AS/400 Advanced Series

ILE COBOL/400 Guía del Programador

Versión 3 Release 7

SC10-9658-01

AS/400 Advanced Series

ILE COBOL/400 Guía del Programador

Versión 3 Release 7

SC10-9658-01

¡Atención!

Antes de utilizar esta información y el producto al que da soporte, lea la información general que se incluye en el apartado “Avisos” en la página xix.

Segunda edición (Noviembre 1996)

Este manual es la traducción del original en inglés *AS/400 Advanced Series ILE COBOL/400 Programmer's Guide*, SC09-2072-01.

Esta edición sólo es aplicable a la Versión 3, Release 7, Nivel de Modificación 0 de IBM Application System/400 ILE COBOL/400 (Número de Programa 5716-CB1) y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en las ediciones nuevas. Asegúrese de que utiliza la edición adecuada para el nivel del producto.

Las modificaciones o adiciones al texto y a las ilustraciones se indican mediante una línea vertical situada a la izquierda de la modificación o adición.

Efectúe el pedido de publicaciones a su representante de ventas IBM o a la sucursal de IBM que preste servicio a su localidad. En la dirección que figura más abajo no hay existencias de las publicaciones.

Al final de la publicación se adjunta una hoja para comentarios del cliente. Si este formulario ha sido extraído, puede enviar sus comentarios a:

IBM, S.A.
National Language Solutions Center
Avda. Diagonal, 571
08029 Barcelona
España

o puede enviar sus comentarios por fax o por correo electrónico.

Cuando se envía información a IBM, se otorga a IBM un derecho no exclusivo para utilizar o distribuir la información en la forma que crea más adecuada sin incurrir por ello en ninguna obligación para con el remitente.

Contenido

Figuras

Tablas

Avisos

Las referencias a programas bajo licencia IBM que se hagan en esta publicación no implican que sólo puedan utilizarse programas bajo licencia IBM. Cualquier servicio, programa o producto funcionalmente equivalente que no infrinja los derechos de propiedad intelectual de IBM puede utilizarse en lugar de los productos, programas o servicios IBM. Es responsabilidad del usuario la evaluación y verificación de su funcionamiento conjuntamente con otros productos excepto aquellos expresamente designados por IBM.

IBM tiene patentes o solicitudes de patente pendientes que tratan la materia objeto de esta publicación. La posesión de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencias, por escrito, a IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, USA.

Los usuarios que posean una licencia de este programa y deseen información sobre la posibilidad de: (i) intercambiar información entre programas creados independientemente y otros programas (incluyendo éste) y (ii) utilizar mutuamente la información que se ha intercambiado, deberían ponerse en contacto con IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Esta información puede estar disponible sujeta a los términos y condiciones adecuados, incluyendo en algunos casos el pago de una tarifa.

Esta publicación contiene ejemplos de datos e informes que se utilizan en operaciones diarias. Para ilustrarlas del modo más completo posible, los ejemplos incluyen nombres de personas, empresas, sucursales y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones de empresas reales es mera coincidencia.

Información sobre la interfaz de programación

La *ILE COBOL/400 Guía del programador* tiene como objetivo ayudarle a crear programas ILE COBOL/400. Contiene información necesaria para utilizar el compilador ILE COBOL/400. Esta publicación aporta información sobre las interfaces de programación de uso general e información asociada proporcionada por el compilador ILE COBOL/400.

Las interfaces de programación de uso general le permiten crear programas que van a utilizar los servicios del compilador ILE COBOL/400.

Marcas registradas y marcas de servicio

Los términos siguientes son marca registrada de International Business Machines Corporation en los Estados Unidos y/o en otros países:

AS/400
Application System/400
C/400
CL/400
CICS
CICS/400
COBOL/400
IBM
ILE Integrated Language Environment
Integrated Language Environment
IBMLink
MVS
Operating System/400
OS/400
PROFS
RPG/400
SAA
SQL/400
System/370
S/370
System/390
S/390
Systems Application Architecture
VM
400

Windows es marca registrada de Microsoft Corporation.

PC DIRECT es marca registrada de Ziff Communications Company; IBM Corporation lo utiliza bajo licencia.

UNIX es marca comercial registrada en los Estados Unidos y en otros países cuya licencia se obtiene exclusivamente a través de X/Open Company Limited.

Otros nombres de servicios, productos y empresas señalados con doble asterisco (**), pueden ser marcas registradas o marcas de servicio de terceros.

Reconocimiento

IBM reconoce la utilización del siguiente producto de investigación en el compilador ILE COBOL/400:

S/SL ©Copyright 1981 by the University of Toronto

Acerca de la publicación ILE COBOL/400 Guía del programador, SC10-9658 (SC09-2072)

La publicación *ILE COBOL/400 Guía del programador* describe cómo escribir, compilar, enlazar, ejecutar, depurar y mantener programas de Entorno de Lenguajes Integrados*(ILE*) COBOL/400* en el AS/400*(Application System/400*). Proporciona información de programación sobre cómo llamar a otros programas ILE COBOL/400 y no ILE COBOL/400, compartir datos con otros programas, utilizar punteros y manejar condiciones de excepción.También describe cómo llevar a cabo operaciones de entrada/salida en dispositivos conectados externamente, archivos de bases de datos, archivos de pantalla y archivos ICF.

Si utiliza este manual, podrá:

- Diseñar y codificar programas ILE COBOL/400
- Entrar, compilar y enlazar programas ILE COBOL/400
- Ejecutar y depurar programas ILE COBOL/400
- Examinar ejemplos ILE COBOL/400 codificados.

Nota: Debería familiarizarse con los capítulos 1 a 6 de esta guía antes de continuar con los otros capítulos.

Este manual hace referencia a otras publicaciones IBM*. Estas publicaciones se listan en el apartado “Bibliografía” en la página 537 con el título completo y el número de pedido base. Cuando se alude a éstas en el texto, se utiliza una versión abreviada del título.

A quién va dirigido este manual

Este manual va dirigido a los programadores de aplicaciones que tienen experiencia en lenguaje de programación COBOL y a los operadores que ejecutan los programas. Es una guía para programar en lenguaje ILE COBOL/400 para usuarios del sistema AS/400.

Para utilizar esta guía debería tener conocimientos básicos sobre:

- Conceptos relacionados con el proceso de datos
- Lenguaje de programación COBOL
- El sistema operativo IBM OS/400* (Operating System/400*)
- Conceptos sobre el Entorno de Lenguajes Integrados (ILE)
- Interfaces de programación de aplicaciones (API)
- Herramientas para el desarrollo, como por ejemplo el Juego de herramientas para el desarrollo de aplicaciones (ADTS/400) para para terminales no programables (NPT).
- Cómo utilizar los controles e indicaciones de la pantalla y cómo utilizar las teclas del teclado como, por ejemplo:
  - Teclas de movimiento del cursor
  - Teclas de función
  - Teclas de salida de campo
  - Teclas Insertar y Suprimir

  - Tecla Restaurar error

- Cómo utilizar la estación de pantalla cuando está conectada al sistema AS/400 de IBM y ejecutando software AS/400. Esto significa saber utilizar el sistema operativo OS/400 y el Lenguaje de control (CL) para:
  - Iniciar y terminar la sesión en la estación de pantalla
  - Interactuar con pantallas
  - Utilizar la ayuda
  - Entrar mandatos CL
  - Utilizar Herramientas para el desarrollo de aplicaciones
  - Responder a mensajes
  - Gestionar archivos.
- Los conceptos básicos de funciones CL de OS/400.
- Cómo utilizar el soporte de gestión de datos para que una aplicación pueda trabajar con archivos.
- Cómo utilizar las siguientes herramientas del Juego de herramientas para el desarrollo de aplicaciones/400:
  - Ayuda para el Diseño de Pantallas (SDA), que se utiliza para diseñar y codificar pantallas.
  - Programa de Utilidad para Entrada del Fuente (SEU) que se utiliza para entrar y actualizar miembros fuente.
- Lenguaje de consulta estructurada (SQL) que se utiliza para insertar sentencias SQL en programas ILE COBOL/400.

Normativa para la industria

En este documento, norma COBOL hace referencia al lenguaje de programación de COBOL tal como se define en el documento:

- American National Standard for Information Systems - Programming Language - COBOL, ANSI X3.23-1985, ISO 1989:1985

  actualizado con el contenido de los siguientes documentos, en el orden en que aparecen:

  - ANSI X3.23a-1989, American National Standard for Information Systems - Programming Language - Intrinsic Function Module for COBOL and ISO 1989:1985/Amd.1:1992, Programming Languages - COBOL, Amendment 1: Intrinsic function module
  - ANSI X3.23b-1993, American National Standard for Information Systems - Programming Language - Correction Amendment for COBOL and ISO/IEC 1989 DAM2 Programming Languages - COBOL, Amendment 2: Correction and clarification amendment for COBOL

El compilador ILE COBOL/400 está diseñado para dar soporte a la norma COBOL (tal como se ha definido) y

- FIPS Publication 21-4, Federal Information Processing Standard 21-4, COBOL

en el nivel de subconjunto intermedio, tal como IBM lo entiende e interpreta con fecha de enero de 1995.

* A partir de estos momentos, se utilizará el término norma COBOL para hacer refe-
* rencia al estándar ANSI que hemos descrito.

* Partes de este manual están copiadas de documentos de la norma COBOL y se reproducen con el permiso de estas publicaciones (copyright 1985 de American National Standards Institute). Puede obtener copias de éstos escribiendo a American National Standard Institute, 1430 Broadway, New York, New York, 10018.

El comité técnico X3J4 de ANSI es el encargado de mantener el lenguaje COBOL.

Consulte el Apéndice A, “Nivel de soporte de lenguaje” en la página 477 para obtener más información sobre la normativa para la industria a que da soporte el compilador ILE COBOL/400.

Reconocimiento

Para su información y guía le presentamos el siguiente extracto del número de formulario 1965-0795689 de la U.S. Government Printing Office:

> Cualquier organización que esté interesada en reproducir las especificaciones y el informe COBOL parcialmente o en su totalidad, utilizando ideas tomadas de este informe como base para una guía de instrucciones o cualquier otro propósito, puede hacerlo. Sin embargo, tales organizaciones deberán reproducir esta sección en el apartado destinado a la introducción del documento. Las organizaciones que utilicen un pasaje corto, como en una revisión de un manual, deben mencionar COBOL en reconocimiento de la fuente, pero no deberán citar literalmente esta sección por completo.
>
> COBOL es un lenguaje de la industria y no es propiedad de ninguna empresa ni grupo de empresas, ni de ninguna organización ni grupo de organizaciones.
>
> Ningún colaborador ni el comité COBOL han otorgado garantías explícitas ni implícitas en relación al funcionamiento y exactitud del lenguaje y del sistema de programación. Es más, no se presume ningún tipo de responsabilidad por parte de ningún colaborador ni del comité en relación a ello.
>
> Se han establecido procedimientos para el mantenimiento de COBOL. Las consultas en relación a estos procedimientos para proponer modificaciones deberían dirigirse al Executive Committee of the Conference on Data Systems Languages.

Los autores y los tenedores del derecho de propiedad intelectual del material bajo copyright:

> FLOW-MATIC** Programming for the UNIVAC (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Unisys Corporation; IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell

han autorizado la utilización de este material parcialmente o por completo, en las especificaciones de COBOL. Tal autorización abarca además la reproducción y utilización de las especificaciones de COBOL en manuales de programación o publicaciones similares.

Notación de la sintaxis de ILE COBOL/400

Los formatos básicos de ILE COBOL/400 se presentan en un sistema uniforme de notación de sintaxis . En los párrafos siguientes encontrará una explicación sobre esta notación, pensada para ayudarle a escribir instrucciones fuente COBOL:

- Las palabras opcionales y las palabras clave de COBOL aparecen en letras mayúsculas; por ejemplo:

  ```
  MOVE
  ```

  Deben escribirse exactamente como se indica. Si falta alguna palabra clave, el compilador lo considera como un error.

- Las variables que representan valores o nombres proporcionados por el usuario aparecen en letras minúsculas; por ejemplo:

  ```
  parmx
  ```

- Para facilitar las referencias de texto, algunas palabras aparecen seguidas de un guión y un dígito o una letra como, por ejemplo:

  ```
  identificador-1
  ```

  Este sufijo no modifica la definición sintáctica de la palabra.

- Los operadores lógicos y aritméticos (+, –, *, /, **, >, <, =, >= y <=) que aparecen en los formatos de sintaxis son obligatorios. Estos operadores son palabras reservadas de *caracteres especiales*. Para obtener una lista completa de palabras ILE COBOL/400 reservadas, consulte el manual *ILE COBOL/400 Reference*.

- La puntuación y demás caracteres especiales que aparecen en el diagrama son necesarios para la sintaxis del formato cuando se muestran; si no los tiene en cuenta, provocará errores en el programa.

- Debe escribir las cláusulas obligatorias y opcionales (cuando se utilicen) en el orden en que aparecen en el diagrama a menos que las normas asociadas determinen explícitamente algo diferente.

Lectura de diagramas de sintaxis

En este manual, la sintaxis se describe utilizando la estructura definida a continuación.

- Lea los diagramas de sintaxis de izquierda a derecha y de principio a fin, siguiendo el sentido de la línea:

  ►►— Indica el principio de una instrucción. Los diagramas de unidades sintácticas que no son instrucciones como, por ejemplo, cláusulas, expresiones y párrafos también empiezan con este símbolo.

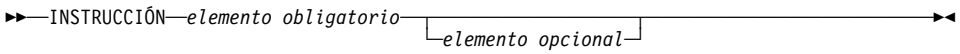
  —► Indica que la sintaxis de la instrucción continua en la línea siguiente.

  ►— Indica que una instrucción es continuación de la línea anterior.

  —►◄ Indica el final de la instrucción. Los diagramas de unidades sintácticas que no son instrucciones como, por ejemplo, cláusulas, expresiones y párrafos también finalizan con este símbolo.

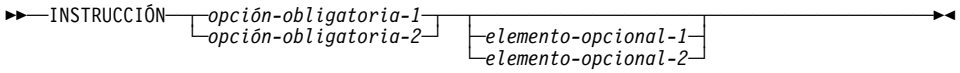
  Nota: Las instrucciones de todo un párrafo incluidas en un diagrama no empiezan con ►►— ni acaban con —►◄ a menos que el principio o el fin coincida con el del párrafo.

- Los elementos obligatorios aparecen en la línea horizontal (la vía principal). Los elementos opcionales aparecen por debajo de la vía principal:

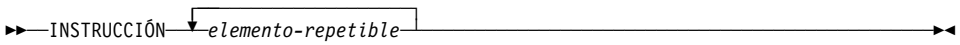


- Cuando se puede elegir entre dos o más elementos, éstos aparecen verticalmente, en una lista.

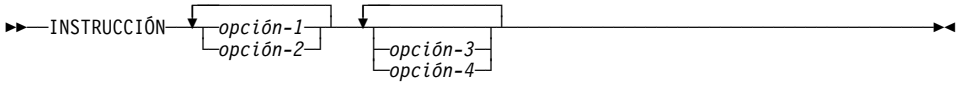
  Si es obligatorio escoger uno de los elementos, un elemento de la lista aparece en la vía principal. Si escoger un elemento es opcional, toda la lista aparece por debajo de la vía principal.



- Una flecha que vuelve hacia la izquierda por encima del elemento indica que el elemento puede repetirse:

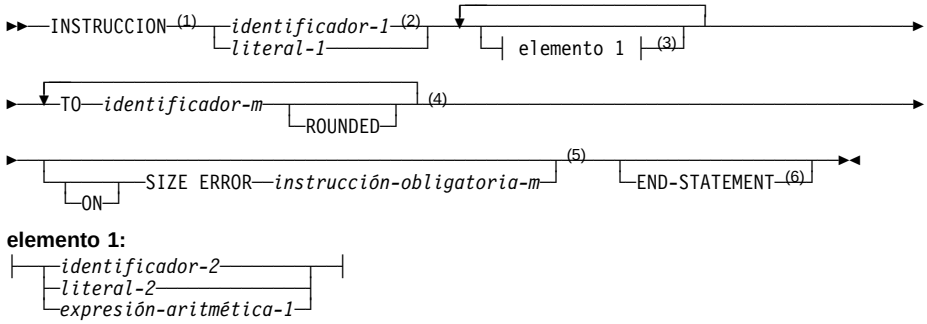


- Una flecha de repetición por encima de una lista de elementos opcionales u obligatorios indica que puede realizar más de una elección entre los elementos listados, o repetir una selección única:



El ejemplo siguiente muestra cómo se utiliza la sintaxis:

Formato



elemento 1:

Notas:

1. La palabra clave INSTRUCCIÓN debe especificarse y codificarse como se indica.
2. Este operando es obligatorio. identificador-1 o literal-1 deben codificarse.
3. El fragmento de elemento 1 es opcional; puede codificarse o no según lo requiera la aplicación. Si el elemento 1 se codifica, puede repetirse con cada entrada separado por uno o más separadores COBOL. Las selecciones de entrada permitidas para este fragmento se describen al final del diagrama.
4. El operando identificador-m y la palabra clave asociada TO son obligatorios y pueden repetirse con uno o más separadores COBOL entre cada entrada. Puede asignarse la palabra clave ROUNDED a cada entrada.
5. La expresión ON SIZE ERROR con la instrucción-obligatoria-m asociada son opcionales. Si la expresión ON SIZE ERROR se codifica, la palabra clave ON es opcional.
6. Puede codificarse la palabra clave END-STATEMENT para finalizar la instrucción. No es un delimitador obligatorio.

Identificación de sintaxis de documentación

Las instrucciones y cláusulas COBOL ilustradas en diagramas de sintaxis cuya sintaxis se comprueba pero que el compilador ILE COBOL/400 trata como documentación, se identifican con notas a pie de página.

Interpretación de códigos de entrada de lenguaje de control (CL) de AS/400

El código que aparece en la esquina superior derecha de cada diagrama de sintaxis de CL contiene los códigos de entrada que especifican el entorno en que el mandato puede entrarse. Los códigos indican si el mandato puede o no puede:

- Utilizarse en un trabajo interactivo o de proceso por lotes (fuera de un programa compilado; Trabajo: B o I)
- Utilizarse en un programa compilado interactivo o de proceso por lotes (Pgm:B o I)
- Utilizarse en un procedimiento REXX interactivo o de proceso por lotes (REXX:B o I)
- Utilizarse como un parámetro del mandato CL CALL o pasarse como una serie de caracteres al programa del sistema QCMDEXC (Exec).

Capítulo 1. Introducción

COBOL (COmmon Business Oriented Language) es un lenguaje de programación parecido al inglés. Como su propio nombre sugiere, COBOL es particularmente eficaz para procesar problemas de gestión.Concede importancia a la descripción y gestión de elementos de datos y registros de entrada/salida; así, COBOL se adapta bien a la gestión de grandes archivos de datos.

Este capítulo proporciona la información siguiente:

- una introducción al Entorno de Lenguajes Integrados*(ILE*)
- una introducción a ILE COBOL/400
- una visión general sobre funciones incorporadas en ILE COBOL/400 que no están disponibles en OPM COBOL/400.
- una visión general sobre los pasos importantes que se deben seguir para crear un objeto de programa ejecutable.
- una visión general sobre otras herramientas para el desarrollo de aplicaciones disponibles para ayudarle a desarrollar aplicaciones ILE COBOL/400 de forma más eficaz.

Entorno de Lenguajes Integrados

El **Entorno de Lenguajes Integrados (ILE)** es la fase en la que se encuentra actualmente la evolución de los modelos de programas OS/400*. Cada fase se ha desarrollado para cumplir con las necesidades actuales de los programadores de aplicaciones. Para obtener una descripción completa de los conceptos y terminología perteneciente a ILE, consulte el manual *ILE Conceptos*..

El entorno de programación que se proporcionó cuando el sistema AS/400 se presentó por primera vez se denomina **modelo de programa original (OPM)**. COBOL, RPG, CL, BASIC y PL/1 funcionaban según este modelo. En la Versión 1 Release 2, se introdujo el **modelo de programa ampliado (EPM)** . EPM se creó para dar soporte a lenguajes como C, Pascal y FORTRAN. Para obtener una descripción de las principales características del OPM y del EPM, consulte el apartado "Historia de ILE" del manual *ILE Conceptos*.

La diferencia más importante entre el entorno OPM COBOL/400 y el entorno ILE COBOL/400 es la forma de crear objetos de programa ejecutables. El compilador ILE COBOL/400 no produce objetos de programa ejecutables. Produce uno o más **objetos de módulo** que pueden **enlazarse** entre sí en diferentes combinaciones para formar una o más unidades ejecutables conocidas como **objetos de programa**.

ILE le permite enlazar objetos de módulo escritos en diferentes lenguajes. Por consiguiente, es posible crear objetos de programa ejecutables que consten de objetos de módulo escritos separadamente en COBOL, RPG, C, C++ y CL.

ILE COBOL/400

Se han incorporado una serie de funciones nuevas en ILE COBOL/400 que no se encuentran disponibles en OPM COBOL/400. Estas funciones nuevas incluyen:

- Soporte de siglo (nuevo en V3R7)

  Se ha añadido la posibilidad de que el usuario trabaje con un año de 4 dígitos en las siguientes instrucciones y funciones:

  – Instrucción ACCEPT con las expresiones YYYYDDD y YYYYMMDD

  – Las siguientes funciones intrínsecas convierten un año de 2 dígitos en uno de 4 dígitos:

    - DATE-TO-YYYYMMDD
    - DAY-TO-YYYYDDD
    - YEAR-TO-YYYY

  – Las siguientes funciones intrínsecas devuelven un año de 4 dígitos:

    - CURRENT-DATE
    - DAY-OF-INTEGER
    - DATE-OF-INTEGER
    - WHEN-COMPILED

- Soporte de comas flotantes (nuevo en V3R7)

  El valor *FLOAT del parámetro CVTOPT de los mandatos CRTCBLMOD y CRTBNDCBL permite utilizar elementos de datos de coma flotante en programas ILE COBOL/400. Además, las instrucciones afectadas (por ejemplo, ACCEPT, DISPLAY, MOVE, COMPUTE, ADD, SUBTRACT, MULTIPLY y DIVIDE) dan soporte a comas flotantes.

- Soporte del área de datos (nuevo en V3R7)

  Se han añadido nuevos formatos de las instrucciones ACCEPT y DISPLAY para posibilitar la recuperación y actualización del contenido de las áreas de datos de AS/400.

- Funciones intrínsecas (nuevo en V3R7)

  Se han añadido las siguientes funciones intrínsecas nuevas:

  ACOS
  ASIN
  ATAN
  CHAR
  COS
  CURRENT-DATE
  DATE-OF-INTEGER
  DATE-TO-YYYYMMDD
  DAY-OF-INTEGER
  DAY-TO-YYYYDDD
  INTEGER-OF-DATE
  INTEGER-OF-DAY
  LENGTH
  LOG
  LOG10
  LOWER-CASE
  MEAN
  NUMVAL
  NUMVAL-C
  ORD
  REVERSE
  SIN
  SQRT
  TAN
  UPPER-CASE
  WHEN-COMPILED
  YEAR-TO-YYYY

- Parámetro de directorio de enlace—BNDDIR (nuevo en V3R7)

| Se ha añadido el parámetro BNDDIR al mandato CRTBNDCBL para permitir la especificación de la lista de directorios enlazados que se utilizan en la resolución de símbolos.

- Parámetro de grupo de activación—ACTGRP (nuevo en V3R7)

  Se ha añadido el parámetro ACTGRP al mandato CRTBNDCBL para permitir la especificación del grupo de activación al que un programa debe estar asociado cuando se llama.

- Objectos de programa y áreas de datos calificados con biblioteca (nuevo en V3R7)

  Se ha añadido la expresión LIBRARY a las siguientes instrucciones ILE COBOL/400 para permitir que los objectos de programa y las áreas de datos de OS/400 se califiquen con un nombre de biblioteca de OS/400:

  - CALL
  - CANCEL
  - SET
  - ACCEPT
  - DISPLAY

- Recogida de datos de rendimiento (nuevo en V3R7)

  Se ha añadido el parámetro ENBPFRCOL a los mandatos CRTCBLMOD y CRTBNDCBL y a la instrucción PROCESS para permitir la generación de código de medida del rendimiento en un módulo o programa. La herramienta de rendimiento del sistema puede utilizar los datos recogidos para crear un perfil del rendimiento de una aplicación.

- Nuevo soporte del depurador de ILE (nuevo en V3R7)

  El depurador de ILE le permite ahora:

  - Depurar la mayor parte de programas OPM
  - Establecer condiciones de observación, peticiones para establecer puntos de interrupción cuando cambia el valor de una variable (o de alguna expresión que determina la dirección de una ubicación de almacenamiento).

* - Nueva expresión EXIT PROGRAM (nuevo en V3R6/V3R2)

  Se ha añadido la expresión AND CONTINUE RUN UNIT a la instrucción EXIT PROGRAM para permitir que pueda abandonarse un programa de llamada sin necesidad de detener la unidad de ejecución.

- Nuevo formato de puntero de la instrucción SET (nuevo en V3R6/V3R2)

  Se ha añadido un nuevo formato de la instrucción SET que permite la actualización de referencias de puntero.

- Soporte de datos DBCS (nuevo en V3R6/V3R2)

  Ahora, se pueden procesar datos de Juego de caracteres de doble byte (DBCS) en ILE COBOL/400. El compilador ILE COBOL/400 da soporte a DBCS, donde cada carácter lógico está representado por dos bytes. DBCS proporciona soporte para lenguajes ideográficos como, por ejemplo, el Juego de caracteres gráficos japoneses de IBM, Kanji.

- Soporte de CALL...BY VALUE y CALL...RETURNING (nuevo en V3R6/V3R2)

  CALL...BY VALUE y CALL...RETURNING le permiten pasar argumentos BY VALUE en lugar de BY REFERENCE y recibir valores RETURN. De este

modo se facilita la migración y se mejora el soporte entre lenguajes ya que tanto ILE C como ILE RPG dan soporte a CALL... BY VALUE y a CALL...RETURNING.

- Soporte a las expresiones BY VALUE y RETURNING de la cabecera PROCEDURE DIVISION (nuevo en V3R6/V3R2)

  La expresión BY VALUE de la cabecera PROCEDURE DIVISION permite a COBOL recibir argumentos BY VALUE de un programa de llamada COBOL o de otros lenguajes ILE como, por ejemplo, RPG, C o VisualAge C++ for OS/400. La expresión RETURNING de la cabecera PROCEDURE DIVISION permite a COBOL devolver un VALUE al procedimiento ILE de llamada.

- Elementos de datos EXTERNAL

  El usuario puede definir los elementos de datos que están disponibles para cada uno de los programas en la unidad de ejecución ILE COBOL/400 utilizando la cláusula EXTERNAL. Ya no es necesario pasar todas las variables que se van a compartir entre programas como argumentos en la instrucción CALL. Este soporte favorece la modularidad de las aplicaciones permitiendo que se compartan datos sin utilizar argumentos ni parámetros en la instrucción CALL.

- Archivos EXTERNAL

  El usuario puede definir los archivos que están disponibles para cada programa en la unidad de ejecución.Puede realizar solicitudes E/S para el mismo archivo desde cualquier programa ILE COBOL/400 de la unidad de ejecución que declare el archivo como EXTERNAL. Para los archivos externos sólo existe un cursor de archivo sin tener en cuenta el número de programas que utilicen el archivo.Se pueden compartir archivos entre programas y, de ese modo, desarrollar programas más pequeños de fácil mantenimiento. Resulta más ventajoso utilizar archivos EXTERNAL que archivos abiertos compartidos ya que tan sólo se necesita una operación OPEN y CLOSE para que todos los programas participantes utilicen el archivo. No obstante, los archivos EXTERNAL no pueden compartirse entre grupos de activación diferentes ni con programas escritos en otros lenguajes de programación.

- Programas fuente anidados

  Un programa fuente ILE COBOL/400 puede contener otros programas fuente ILE COBOL/400. Estos programas contenidos pueden hacer referencia a algunos de los recursos como, por ejemplo, elementos de datos y archivos de los programas en los que se encuentran contenidos o definir sus propios recursos localmente, que únicamente estarán visibles en el programa que los define. Como los programas ILE COBOL/400 son también recursos, su ámbito también está controlado por la estructura de anidamiento y por el atributo de ámbito asociado al programa. De este modo se proporciona más flexibilidad al controlar el conjunto de programas ILE COBOL/400 que puede llamar un programa ILE COBOL/400. Los programas ILE COBOL/400 anidados proporcionan un mecanismo para ocultar recursos que de otro modo serían visibles.

- Cláusula INITIAL

  El usuario tiene un mecanismo por el que un programa ILE COBOL/400 y los programas contenidos en éste pasan a su estado inicial cada vez que se les llama.Esto se consigue especificando INITIAL en el párrafo PROGRAM-ID. Así se proporciona más flexibilidad al controlar la unidad de ejecución COBOL.

- Instrucción REPLACE

La instrucción REPLACE resulta útil para sustituir texto del programa fuente durante el proceso de compilación. Funciona en todo el archivo o hasta que se encuentra otra instrucción REPLACE, a diferencia de lo que ocurre con la instrucción COPY en la expresión REPLACING. Las instrucciones REPLACE se procesan una vez que se han procesado todas las instrucciones COPY. Así se proporciona más flexibilidad al modificar el texto ILE COBOL/400 que se va a compilar.

- Instrucción DISPLAY WITH NO ADVANCING

  Cuando se utiliza la expresión NO ADVANCING en la instrucción DISPLAY, el usuario podrá dejar el cursor a continuación del último carácter que se visualiza. Esto le permite encadenar elementos para que se visualicen en una única línea desde varios puntos del programa ILE COBOL/400.

- Instrucción ACCEPT FROM DAY-OF-WEEK

  ILE COBOL/400 le permite recuperar el día de la semana (Lunes = 1, Martes = 2 ...) y asignarlo a un identificador. Este soporte complementa el soporte ACCEPT FROM DAY/DATE/TIME existente.

- Cláusula SELECT OPTIONAL para archivos relativos

  Permite la creación automática de archivos relativos aunque el archivo sea abierto de E-S.Esto amplía el soporte ya existente para archivos secuenciales.

- Soporte para instrucciones COPY anidadas

  Los miembros a los que se hace referencia en COPY pueden contener instrucciones COPY ampliando de este modo la aplicación de la instrucción COPY. Si un miembro COPY contiene una directiva COPY, ni la directiva COPY contenedora ni la directiva COPY contenida pueden especificar la expresión REPLACING.

- Mejoras realizadas a las instrucciones ACCEPT y DISPLAY ampliadas

  Se puede trabajar con tablas en la instrucción ACCEPT ampliada. Esto le permite actualizar de forma selectiva y fácilmente los elementos de la tabla.

  También se permiten las tablas de longitud variable en las instrucciones ACCEPT y DISPLAY ampliadas.

  Asimismo, también se da soporte a la cláusula SIZE en la instrucción ACCEPT ampliada.

- Soporte de puntero de procedimiento

  Puntero a procedimiento es un nuevo tipo de datos que contienen la dirección de programas ILE COBOL/400 o programas que no son ILE COBOL/400. Los punteros a procedimiento se definen especificando la cláusula USAGE IS PROCEDURE-POINTER en un elemento de datos. Este nuevo tipo de datos resulta útil para llamar programas y/o procedimientos ILE que esperan este tipo de datos como parámetro. Los elementos de datos de tipo puntero de procedimiento también pueden utilizarse como destino de una instrucción CALL para llamar a otro programa.

- Registros especiales nuevos
  - Registro especial RETURN-CODE

    Permite que la información de retorno se pase entre programas ILE COBOL/400. Normalmente, este registro se utiliza para pasar información sobre si ha finalizado satisfactoriamente o no una llamada de programa.

- Registro especial SORT-RETURN

  Devuelve información sobre la finalización satisfactoria de una instrucción SORT o MERGE.También permite al usuario terminar el proceso de una instrucción SORT/MERGE desde un área declarativa de error o un procedimiento de entrada-salida.

- Opciones nuevas del compilador
  - *PICGGRAPHIC/*NOPICGGRAPHIC

    *PICGGRAPHIC es un parámetro nuevo para la opción CVTOPT que permite que el usuario transfiera datos DBCS al programa ILE COBOL/400.

  - opción *IMBEDERR/*NOIMBEDERR

    *IMBEDERR es una opción nueva del compilador que muestra los errores que se producen durante la compilación en el momento en que ocurren en el listado del compilador así como al final del listado.

  - *FLOAT/*NOFLOAT

    *FLOAT es un parámetro nuevo de la opción CVTOPT que le permite transferir elementos de datos de coma flotante a programas ILE COBOL/400 utilizando los nombres de DDS y un USAGE de COMP-1 (precisión simple) o COMP-2 (precisión doble).

  - opción *NOSTDTRUNC/*STDTRUNC

    NOSTDTRUNC es una opción de compilador nueva que suprime el truncamiento de valores en elementos de datos BINARY. Esta opción resulta útil cuando se migran aplicaciones desde IBM System/390 (S/390*).

  - opción *CHGPOSSGN/*NOCHGPOSSGN

    Esta opción es útil cuando se comparten datos entre OS/400 y IBM S/390. Esta opción se proporciona para compatibilidad con IBM System/390. Modifica la representación de bits de elementos de datos con zona y empaquetados con signo cuando se utilizan en instrucciones aritméticas o en instrucciones MOVE y los valores de estos elementos de datos son positivos.

- Soporte de nombres de sistema entrecomillados

  Se ha añadido soporte para aceptar literales donde se permiten nombres de sistema. Se permite la utilización de cualquier nombre al que el sistema dé soporte y no se limita a los nombres COBOL válidos.

- No existe límite de COBOL en las funciones siguientes ya que éstas vienen determinadas por las limitaciones del sistema.
  - Número de archivos declarados
  - Número de parámetros en la instrucción CALL y en la expresión USING de la PROCEDURE DIVISION. Se aplica un límite del sistema de 400 para procedimientos ILE y 255 para objetos de programa.
  - Número de archivos de entrada SORT-MERGE y número de claves SORT-MERGE. El número máximo de archivos de entrada SORT-MERGE es 32 y la longitud máxima de clave SORT-MERGE es 2000 bytes.

- Instrucción START con NO LOCK.

  Cuando se utiliza la expresión NO LOCK en la instrucción START, el cursor del archivo se coloca en el primer registro que se va a leer sin poner un bloqueo

en el registro. Este soporte se proporciona para archivos relativos e indexados y complementa la función READ con NO LOCK ya disponible.

Nota: START con NO LOCK es una instrucción nueva tanto para ILE COBOL/400 como para OPM COBOL/400.

- Soporte de llamada estática de procedimientos

  El usuario puede desarrollar las aplicaciones en objetos de módulo más pequeños de mejor mantenimiento y enlazarlos como un objeto de programa, sin incurrir en el inconveniente de sobrecarga del sistema que supone la llamada dinámica de programas. Este recurso, junto con el entorno en tiempo de ejecución común que proporciona el sistema, también mejora la posibilidad de escribir aplicaciones utilizando varios lenguajes.Los lenguajes de programación ILE permiten el enlace de C, RPG, COBOL y CL en un objeto de programa único haciendo caso omiso de la mezcla de los lenguajes fuente.

  Se ha añadido una nueva sintaxis en la instrucción CALL literal y una opción nueva de compilador a ILE COBOL/400 para diferenciar entre llamadas estáticas de procedimientos y llamadas dinámicas de programas.

- Soporte de registros de longitud variable (cláusula RECORD IS VARYING)

  Puede definir y utilizar con facilidad registros de longitud diferente en el mismo archivo utilizando la sintaxis COBOL ANSI. No sólo supone un ahorro importante de almacenamiento sino que además facilita la tarea de migración de aplicaciones complejas desde otros sistemas.

- Límites ampliados del compilador

  ILE COBOL/400 ofrece límites ampliados del compilador:

  – tamaño de elementos de datos elementales y de grupo
  – tamaño de tablas de longitud variable y fija
  – número de niveles de anidamiento para instrucciones condicionales
  – número de operandos en varias instrucciones de Procedure Division

Pasos principales para crear un objeto de programa ILE COBOL/400 ejecutable

La Figura 1 ilustra los pasos habituales que deben seguirse para desarrollar un objeto de programa ejecutable en ILE COBOL/400:

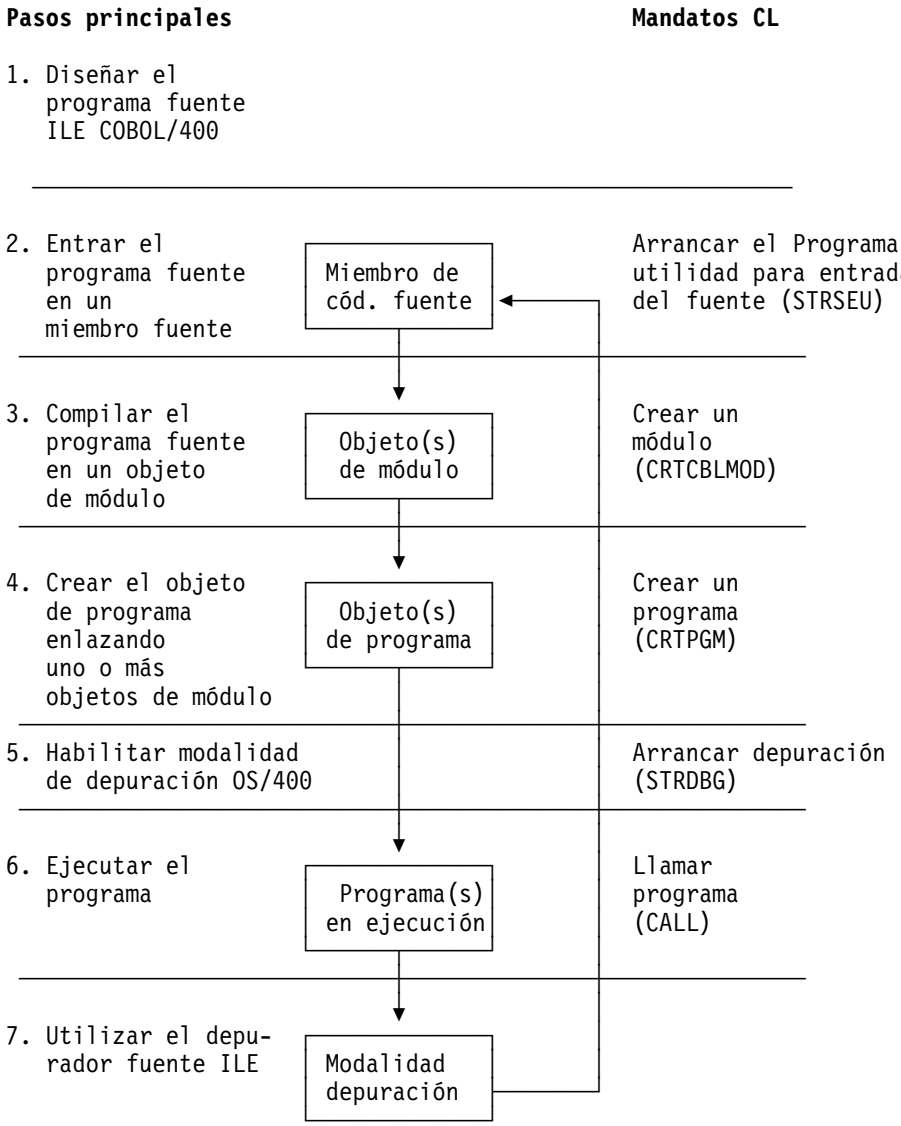


Figura 1. Pasos principales para crear un objeto de programa ILE COBOL/400 ejecutable

Los pasos 3 y 4 pueden llevarse a cabo con un único mandato, CRTBNDCBL. Este mandato crea objetos de módulo temporales a partir del programa fuente ILE COBOL/400 y a continuación crea el objeto de programa. Una vez creado el objeto de programa, los objetos de módulo se suprimen.

Diseño del programa fuente ILE COBOL/400 del usuario

El primer paso que debe llevarse a cabo cuando se crea un objeto de programa ILE COBOL/400 ejecutable es diseñar el programa fuente ILE COBOL/400.

Un **programa fuente ILE COBOL/400** consta de cuatro divisiones. El programa esquemático que aparece en la Figura 2 en la página 10 muestra la estructura de un programa fuente ILE COBOL/400 . Puede utilizarse como ejemplo para diseñar programas fuente ILE COBOL/400.

Los programas ILE COBOL/400 pueden encontrarse dentro de otros programas ILE COBOL/400. Este concepto se conoce como anidamiento y el programa contenido se denomina **programa anidado**. La Figura 2 en la página 10 muestra cómo se incluye un programa ILE COBOL/400 anidado en un programa ILE COBOL/400 más externo. No todas las entradas que se proporcionan en el programa esquemático son obligatorias; la mayoría se proporcionan con carácter informativo.

```
       IDENTIFICATION DIVISION.  1
         PROGRAM-ID. nombre-programa-más externo.
         AUTHOR. entrada-comentario.
         INSTALLATION. entrada-comentario.
         DATE-WRITTEN. entrada-comentario.
         DATE-COMPILED. entrada-comentario.
         SECURITY.
      *   El párrafo SECURITY puede utilizarse para especificar
      *   la información sobre derechos de autor perteneciente
      *   al objeto de módulo generado. Las 8 primeras líneas
      *   del párrafo SECURITY generan la información sobre
      *   derechos de autor que se visualiza en el panel de
      *   Información sobre derechos de autor cuando se
      *   emite el mandato CL Visualizar módulo (DSPMOD).

       ENVIRONMENT DIVISION.  2
        CONFIGURATION SECTION.  3
         SOURCE-COMPUTER. IBM-AS400.
         OBJECT-COMPUTER. IBM-AS400.
         SPECIAL-NAMES.   REQUESTOR IS CONSOLE.
        INPUT-OUTPUT SECTION.  4
         FILE-CONTROL.
           SELECT nombre-archivo ASSIGN TO DISK-nombre-archivo
             ORGANIZATION IS SEQUENTIAL
             ACCESS MODE IS SEQUENTIAL
             FILE STATUS IS nombre-datos.

      DATA DIVISION.  5
       FILE SECTION.
       FD nombre-archivo.
       01 nombre-registro  PIC X(132).
       WORKING-STORAGE SECTION.
       77 nombre-datos PIC XX.
       LINKAGE SECTION.

      PROCEDURE DIVISION.  6
       DECLARATIVES
       END DECLARATIVES.
       proceso-principal SECTION.
        párrafo-proceso-principal
            instrucciones ILE COBOL/400.
               STOP RUN.

      IDENTIFICATION DIVISION.  7
        PROGRAM-ID. nombre-programa-anidado.

      ENVIRONMENT DIVISION.  8
       INPUT-OUTPUT SECTION.
        FILE-CONTROL.
          SELECT nombre-archivo ASSIGN TO DISK-nombre-archivo
            ORGANIZATION IS SEQUENTIAL
            ACCESS MODE IS SEQUENTIAL
            FILE STATUS IS nombre-datos.

      DATA DIVISION.
       FILE SECTION.
       FD nombre-archivo.
       01 nombre-registro  PIC X(132).
       WORKING-STORAGE SECTION.
       77 nombre-datos PIC XX.
       LINKAGE SECTION.

      PROCEDURE DIVISION.
       DECLARATIVES
       END DECLARATIVES.
       proceso-principal SECTION.
        párrafo-proceso-principal
            instrucciones ILE COBOL/400.
               EXIT PROGRAM.

      END PROGRAM nombre-programa-anidado.  9
      END PROGRAM nombre-programa-más externo.
```

Figura 2. Ejemplo de estructura de programa ILE COBOL/400

La Identification Division 1 es la única división que debe incluirse; el resto de divisiones son opcionales.

La Environment Division 2 se compone de dos secciones: La Configuration Section 3 , que describe las especificaciones generales de los sistemas objeto y fuente, y la Input-output Section 4 , que define cada uno de los archivos y especifica la información necesaria para transmitir datos entre un soporte externo y el programa ILE COBOL/400.

La Data Division 5 describe los archivos que van a utilizarse en el programa y los registros que se encuentran en los archivos. También describe los elementos de datos de trabajo del programa en Working-Storage necesarios.

La Procedure Division 6 consta de declarativas opcionales y procedimientos que contienen secciones y/o párrafos, sentencias e instrucciones.

Esta segunda Identification Division 7 marca el principio de un programa ILE COBOL/400 anidado contenido en un programa ILE COBOL/400 más externo.

Los programas anidados no pueden tener una Configuration section 8 en la Environment Division.El programa más externo debe especificar todas las opciones de la Configuration Section que puedan resultar necesarias.

Los programas anidados y el programa más externo deben finalizar con la cabecera END PROGRAM 9 .

Un programa ILE COBOL/400 se identifica mediante el PROGRAM-ID que se encuentra en la IDENTIFICATION DIVISION. Contiene un conjunto de instrucciones autocontenidas que llevan a cabo una tarea determinada.

En ILE, a los programas fuente ILE COBOL/400 se les considera **procedimientos ILE**. Si un programa ILE COBOL/400 contiene programas ILE COBOL/400 anidados, cada uno de los programas ILE COBOL/400 anidados es un procedimiento ILE. El nombre del programa anidado sólo se conoce en el programa contenedor. Si el programa anidado tiene el atributo COMMON, el nombre del pro-

grama anidado también es conocido por otros programas de la misma unidad de compilación. Los procedimientos ILE no deben confundirse con los procedimientos COBOL, que se encuentran en la Procedure Division de un programa COBOL y que contienen secciones, párrafos, sentencias e instrucciones.

Para obtener más información sobre cómo escribir un programa ILE COBOL/400, consulte el manual *ILE COBOL/400 Reference*.

Entrada de instrucciones fuente en un miembro fuente

Después de diseñar el programa ILE COBOL/400, debe entrarlo en un miembro fuente.

El mandato Arrancar Programa de Utilidad para Entrada del Fuente (STRSEU) del Programa de Utilidad para Entrada del Fuente (SEU) se utiliza para entrar y editar las instrucciones fuente de ILE COBOL/400. Para ayudarle a entrar instrucciones ILE COBOL/400 correctas en el sistema, la pantalla del SEU corresponde al formato de codificación de la norma COBOL y a medida que entre o modifique una línea de código, el verificador de sintaxis COBOL comprobará si existen errores en la línea.

Una **unidad de compilación** es el programa ILE COBOL/400 más externo y cualquiera de los programas ILE COBOL/400 anidados en el programa más externo. Pueden entrarse varias unidades de compilación en un único miembro fuente.

Para obtener más información sobre cómo entrar instrucciones fuente, consulte el Capítulo 2, “Entrada de instrucciones fuente en un miembro fuente” en la página 17.

Compilación de un programa fuente en objetos de módulo

Cuando haya acabado de entrar o editar las instrucciones fuente en un miembro fuente, tendrá que crear objetos de módulo utilizando el mandato Crear módulo COBOL (CRTCBLMOD). Este mandato compila las instrucciones fuente del miembro fuente en uno o más objetos de módulo. Cada unidad de compilación del miembro fuente crea un objeto de módulo separado.

Los **objetos de módulo** son la salida del compilador ILE COBOL/400. Se representan en el sistema mediante el tipo *MODULE.Los objetos de módulo no pueden ejecutarse si primero no se han enlazado en objetos de programa.

Para obtener más información sobre la creación de objetos de módulo utilizando el mandato CRTCBLMOD, consulte el Capítulo 3, “Compilación de programas fuente en objetos de módulo” en la página 27.

Creación de un objeto de programa

Para poder crear un **objeto de programa** ejecutable, deben enlazarse los objetos de módulo entre sí. Es posible enlazar uno o más objetos de módulo entre sí para crear un objeto de programa. Los objetos de módulo tienen que enlazarse en objetos de programa ya que únicamente los objetos de programa pueden ejecutarse. Los objetos de módulo escritos en varios lenguajes de programación pueden enlazarse entre sí para crear un objeto de programa. Por ejemplo, un objeto de programa puede constar de objetos de módulo COBOL o RPG para el informe pero de un objeto de módulo C para los cálculos. Un objeto de programa puede crearse utilizando uno de los mandatos siguientes:

- Crear programa (CRTPGM)
- Crear programa COBOL enlazado (CRTBNDCBL)

Para obtener más información sobre estos mandatos, consulte el Capítulo 4, “Creación de un objeto de programa” en la página 71.

Ejecución de un objeto de programa

Se ejecuta un objeto de programa llamándolo. Se puede utilizar uno de los métodos siguientes para llamar a un objeto de programa:

- El mandato CL CALL en una línea de mandatos
- Una instrucción CALL de lenguaje de alto nivel
- Un menú orientado a la aplicación
- Un mandato creado por el usuario

Para obtener información sobre la ejecución de un programa, consulte el Capítulo 6, “Ejecución de un programa ILE COBOL/400” en la página 107.

Depuración de un programa

El depurador fuente ILE se utiliza para detectar errores en los programas de servicio y objetos de programa y eliminarlos. Puede utilizar el depurador fuente ILE para:

- Visualizar el fuente del programa
- Establecer y eliminar puntos de interrupción condicionales e incondicionales.
- Ir a pasos por el programa
- Visualizar el valor de variables, estructuras, registros y matrices
- Modificar el valor de variables
- Modificar el ámbito de referencia
- Equiparar un nombre abreviado a una variable, una expresión o un mandato de depuración.

Para obtener más información sobre el depurador, consulte el Capítulo 7, “Depuración de un programa” en la página 113.

Otras herramientas para el desarrollo de aplicaciones

Los siguientes productos están disponibles para ayudarle a desarrollar aplicaciones ILE COBOL/400 de forma más eficaz.

Gestor para el Desarrollo de Aplicaciones/400

El programa bajo licencia Gestor para el Desarrollo de Aplicaciones/400 proporciona a las organizaciones para el desarrollo de aplicaciones un mecanismo para gestionar de forma eficaz objetos de aplicaciones durante la vida de la aplicación. Este producto permite a un grupo de programadores crear, gestionar y organizar múltiples versiones de la aplicación mediante el producto Workstation Platform/2 o directamente desde la línea de mandatos de AS/400. Para obtener más información sobre el programa bajo licencia Gestor para el Desarrollo de Aplicaciones/400, consulte el manual *ADTS/400: Gestor para el Desarrollo de Aplicaciones ADM/400 Introducción y Guía de Planificación*, GC10-9401 (GC09-1807).

Juego de Herramientas de Desarrollo de Aplicaciones ADTS para OS/400 (ADTS para OS/400)

Juego de Herramientas de Desarrollo de Aplicaciones ADTS para OS/400 (ADTS para OS/400) proporciona un conjunto integrado de herramientas basadas en el sistema diseñadas para satisfacer las necesidades de los profesionales de la informática. Este producto proporciona herramientas para manipular archivos de base de datos, objetos y fuentes en el sistema AS/400. Se proporciona un editor de pantalla completa así como herramientas para diseñar pantallas e informes.

Application Development ToolSet Client Server for OS/400

Entorno de Desarrollo Coorperativo CODE/400 (CODE/400) es una característica de Application Development ToolSet Client Server for OS/400 (ADTS CS for OS/400). CODE/400 ahora da soporte a ILE COBOL/400 con la función de edición/compilación/depuración completa. Además, proporciona verificación de programas y comprobación de sintaxis en el PWS. .*

Compilación, ejecución y depuración de programas ILE COBOL/400

Capítulo 2. Entrada de instrucciones fuente en un miembro fuente

Este capítulo proporciona la información necesaria para entrar las instrucciones fuente ILE COBOL/400. También describe brevemente la metodología y herramientas necesarias para realizarlo.

Para entrar instrucciones fuente ILE COBOL/400 en el sistema, utilice uno de los métodos siguientes:

1. Entre instrucciones fuente utilizando el Programa de utilidad para entrada del fuente (SEU). Este método se explica en este capítulo.
2. Entre instrucciones fuente desde disquete o cinta utilizando los mandatos CL de OS/400 CPYFRMTAP y CPYFRMDKT.

   Para obtener información sobre cómo entrar instrucciones fuente utilizando los mandatos CL, consulte el manual *CL Reference*, SC41-4722.

Creación de una biblioteca y de un archivo físico fuente

Las instrucciones fuente se entran en un miembro de un archivo físico. Para poder entrar el fuente, en primer lugar debe crear una biblioteca y un archivo físico fuente.

Una **biblioteca** es un objeto del sistema que sirve como directorio a otros objetos. Las bibliotecas agrupan objetos relacionados y permiten al usuario buscar objetos por el nombre. El tipo de objeto para las bibliotecas es *LIB.

Un **archivo físico fuente** es un archivo que almacena miembros. Estos miembros contienen instrucciones fuente como, por ejemplo instrucciones fuente ILE COBOL/400.

Para crear una biblioteca denominada MYLIB, utilice el mandato Crear biblioteca (CRTLIB):

```
CRTLIB LIB(MYLIB)
```

Para crear un archivo físico fuente denominado QCBLLESRC en la biblioteca MYLIB, utilice el mandato Crear archivo físico fuente (CRTSRCPF):

```
CRTSRCPF FILE(MYLIB/QCBLLESRC)
```

Nota: En el ejemplo anterior, la biblioteca MYLIB debe existir para poder crear el archivo físico fuente.

Para obtener más información sobre la creación de bibliotecas y archivos físicos fuente, consulte el manual *ADTS/400: Gestor de Desarrollo de Programas (PDM)*, SC10-9419 (SC09-1771).

Cuando haya creado la biblioteca y el archivo físico fuente, podrá iniciar una sesión de edición. Puede utilizar el mandato Arrancar programa de utilidad para entrada del fuente (STRSEU) para iniciar una sesión de edición y entrar las instrucciones fuente.

Nota: Además puede entrar el programa fuente desde disquete o cinta con la función de copia de OS/400. Para obtener más información sobre la función de copia de OS/400, consulte el manual *CL Reference*.

Entrada de instrucciones fuente utilizando el Programa de Utilidad para Entrada del Fuente

El Programa de Utilidad para Entrada del Fuente proporciona formatos de pantalla especiales para COBOL que corresponden al Formato de codificación COBOL y están pensados para facilitarle la entrada de instrucciones fuente COBOL. La Figura 3 muestra un ejemplo de formato de pantalla que el SEU proporciona para COBOL. El SEU puede mostrar una línea de formato para ayudarle a entrar o realizar modificaciones en el código fuente, posición por posición (vea 1).

```
 Columnas. . . :    1  71           Editar                        MYLIB/QCBLLESRC
 SEU==> _____________________________________________________________     XMPLE1
 FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++ 1
        *************** Principio de datos ************************************
0001.00         IDENTIFICATION DIVISION.
0002.00           PROGRAM-ID. XMPLE1.
0003.00
0004.00         ENVIRONMENT DIVISION.
0005.00         CONFIGURATION SECTION.
0006.00           SOURCE-COMPUTER.  IBM-AS400.
0007.00         INPUT-OUTPUT SECTION.
0008.00         FILE-CONTROL.
0009.00             SELECT FILE-1 ASSIGN TO DATABASE-MASTER.
        ****************** Fin de datos *************************************
 Tipo solicitud. .   CB      Número secuencia. . .  0008.00

 Continuación

 Area-A      Area-B
  FILE       -CONTROL.

 F3=Salir  F4=Solicitud   F5=Renovar                  F11=Registro anterior
 F12=Cancelar             F23=Seleccionar solicitud   F24=Más teclas
```

Figura 3. Un formato de pantalla SEU

Para obtener una descripción completa sobre cómo entrar instrucciones fuente .* using SEU, refer to the &1338.. utilizando el SEU, consulte el manual *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)*, SC10-9422 (SC09-1774).

Una **unidad de compilación** es el programa ILE COBOL/400 más externo y cualquiera de los programas ILE COBOL/400 anidados en el programa más externo. Pueden entrarse varias unidades de compilación en un único miembro fuente.

Formato de archivo fuente COBOL

La longitud de registro estándar de los archivos fuente es de 92 caracteres. Estos 92 caracteres se componen de un número de secuencia de 6 caracteres, un campo de datos de 80 caracteres y un área de fecha de última modificación de 6 caracteres.

El compilador ILE COBOL/400 da soporte a una longitud de registro adicional de 102; al final del registro (posiciones 93-102) se sitúa un campo de 10 caracteres que contiene información complementaria. El compilador ILE COBOL/400 no utiliza esta información, pero se coloca en el extremo derecho del listado del compilador. El usuario es el responsable de colocar esta información en dicho campo. Si desea utilizar este campo adicional, cree un archivo fuente cuya longitud de registro sea 102.

Se proporciona un archivo fuente en el que se pueden almacenar los registros fuente si no desea crear su propio archivo. Este archivo, denominado QCBLLESRC, está en la biblioteca QGPL y tiene una longitud de registro de 92 caracteres.

Arranque de SEU

Para entrar el programa fuente ILE COBOL/400 utilizando el SEU, entre el mandato Arrancar el programa de utilidad para entrada del fuente (STRSEU) y especifique CBLLE para el parámetro TYPE. Especifique SQLCBLLE para el parámetro TYPE si el programa fuente contiene instrucciones SQL intercaladas.

Si no especifica un parámetro TYPE, SEU utiliza como valor por omisión el mismo tipo que se utilizó cuando el miembro se editó por última vez. Si no especifica el parámetro TYPE y está creando un miembro nuevo, SEU asigna un tipo de miembro por omisión asociado al nombre del archivo físico fuente. Para ILE COBOL/400, este tipo de miembro por omisión es CBLLE. Para obtener información sobre otros métodos para arrancar el SEU, consulte el manual *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)*, SC10-9422 (SC09-1774).

Utilización del corrector sintáctico COBOL en SEU

Para utilizar el corrector sintáctico COBOL en SEU, especifique el parámetro TYPE (CBLLE) del mandato STRSEU. El corrector sintáctico COBOL comprueba todas las líneas para detectar errores a medida que el usuario entra líneas nuevas o modifica las existentes. Se identifican las instrucciones fuente que no son correctas y aparece un mensaje de error que permite corregir los errores antes de compilar el programa.

Cada vez que entre o modifique una línea fuente otras líneas del código fuente pueden verificarse sintácticamente como parte de esa unidad de verificación de sintaxis. La longitud de una única unidad de verificación de sintaxis se determina realizando una extensión desde una línea modificada o entrada de la forma siguiente:

- Una unidad de verificación de sintaxis se extiende hacia el principio del miembro fuente hasta que se encuentra el principio de la primera línea fuente o bien se halla una línea que finaliza con un punto.
- Una unidad de verificación de sintaxis se extiende hacia el final del miembro fuente hasta que se halla el final de la última línea fuente o se encuentra una línea que finaliza con un punto.

Como el corrector sintáctico COBOL sólo comprueba una sentencia cuando ésta se entra o modifica, con independencia de las sentencias de la precedan o la sigan, sólo se detectan los errores de sintaxis contenidos en cada sentencia fuente. No se detectan errores interrelacionados como, por ejemplo, nombres no

definidos o referencias incorrectas a nombres. El compilador ILE COBOL/400 detecta estos errores cuando se compila el programa.

Sin embargo, si se realiza una modificación en una sentencia que forma parte de una entrada de comentario de un párrafo opcional de la Identification Division, el corrector sintáctico no reconocerá que el contexto permite que se entre cualquier combinación de caracteres. Es posible que genere varios errores al intentar identificar el contenido de la sentencia como una instrucción de COBOL válida. Esto puede evitarse escribiendo la entrada de comentario como una sola sentencia que empiece en la misma línea que el nombre del párrafo o sustituyendo la entrada de comentario por una serie de líneas de comentario.

Si hay un error en una unidad de verificación de sintaxis, la parte de la unidad en la que se identifica el error aparece en contraste invertido. El mensaje que aparece en la parte inferior de la pantalla hace referencia al primer error de la unidad.

La verificación de sintaxis se produce a medida que se entra el código fuente. Los mensajes de error los generan líneas que constan de instrucciones incompletas. Estos desaparecen cuando las instrucciones se completan, como en el siguiente ejemplo:

```
 Columnas. . . :    1  71            Editar                    TESTLIB/QCBLLESRC
 SEU==> _____________________________________________________________      ADDATOB
 FMT CB ......-A+++B+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        *************** Principio de datos ************************************
0000.10         IDENTIFICATION DIVISION.
0000.20         PROGRAM-ID. ADDATOB.
0000.30         ENVIRONMENT DIVISION.
0000.40         CONFIGURATION SECTION.
0000.50           SOURCE-COMPUTER. IBM-AS400.
0000.60           OBJECT-COMPUTER. IBM-AS400.
0000.70         DATA DIVISION.
0000.80         WORKING-STORAGE SECTION.
0000.90         01  A     PIC S9(8) VALUE 5.
0001.00         01  B     PIC S9(8) VALUE 10.
0001.10         PROCEDURE DIVISION.
0001.20         MAINLINE.
0001.30             ADD A
'''''''
        ****************** Fin de datos *************************************


 F3=Salir  F4=Solicitud  F5=Renovar  F9=Recuperar  F10=Cursor  F11=Conmutar
 F16=Repetir búsqueda    F17=Repetir cambio        F24=Más teclas
TO esperado, EOL encontrado. Línea rechazada.
```

Figura 4. Mensaje de error del corrector sintáctico COBOL generado por una instrucción incompleta

```
 Columnas. . . :    1  71           Editar                    TESTLIB/QCBLLESRC
 SEU==> ____________________________________________________________    ADDATOB
 FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ********** Principio de datos ************************************
0000.40         IDENTIFICATION DIVISION.
0000.50         PROGRAM-ID. ADDATOB.
0000.60         ENVIRONMENT DIVISION.
0000.70         CONFIGURATION SECTION.
0000.80           SOURCE-COMPUTER. IBM-AS400.
0000.90           OBJECT-COMPUTER. IBM-AS400.
0000.91         DATA DIVISION.
0000.92         WORKING-STORAGE SECTION.
0000.93         01  A     PIC S9(8) VALUE 5.
0000.94         01  B     PIC S9(8) VALUE 10.
0001.00         PROCEDURE DIVISION.
0001.10         MAINLINE.
0002.00             ADD A
0003.00             TO B.
'''''''
        ****************** Fin de datos *************************************

 F3=Salir  F4=Solicitud  F5=Renovar  F9=Recuperar  F10=Cursor  F11=Conmutar
 F16=Repetir búsqueda    F17=Repetir cambio        F24=Más teclas
```

Figura 5. El mensaje de error del corrector sintáctico COBOL desaparece cuando la instrucción se completa

Un mensaje de error se genera después de entrar la primera línea y desaparece tras entrar la segunda línea, cuando se completa la instrucción.

Las siguientes normas se aplican a la verificación de sintaxis para el código fuente de ILE COBOL/400:

- No se realiza la verificación de sintaxis del código fuente que se encuentra en una línea con un asterisco (*) o con una barra inclinada (/) en la columna 7. Un asterisco indica una línea de comentarios; una barra inclinada indica una línea de comentarios y una expulsión de página.
- Durante la verificación de sintaxis no se respeta ninguna opción del compilador.

  Por ejemplo, el corrector sintáctico acepta comillas y apóstrofos como delimitadores no numéricos siempre que no se encuentren combinados en una unidad de verificación de sintaxis. El corrector sintáctico no comprueba si el delimitador es el que se especificará en los mandatos CRTCBLMOD o CRTBNDCBL o en la instrucción PROCESS.
- La sustitución de caracteres que especifican las cláusulas CURRENCY y DECIMAL-POINT del párrafo SPECIAL-NAMES no se respeta durante la verificación de sintaxis interactiva.
- Cuando se utiliza la cláusula REPLACING *Identificador-1* BY *Identificador-2* de la instrucción COPY y cuando alguno de los identificadores incluye una modificación de referencias, el corrector sintáctico COBOL del SEU comprueba únicamente los paréntesis emparejados.
- En las instrucciones COPY y REPLACE se comprueba la estructura de la sintaxis.
- Se comprueba la sintaxis de las instrucciones SQL intercaladas.

Ejemplo de entrada de instrucciones fuente en un miembro fuente

Este ejemplo le muestra cómo crear una biblioteca y un archivo físico fuente, arrancar una sesión de edición y entrar instrucciones fuente utilizando los mandatos Crear biblioteca (CRTLIB), Crear archivo físico fuente (CRTSRCPF) y Arrancar SEU (STRSEU).

Nota: Para poder llevar a cabo estas tareas con estos mandatos, en primer lugar debe tener autorización para utilizar estos mandatos.

1. Para crear una biblioteca cuyo nombre sea MYLIB, teclee

   ```
   CRTLIB LIB(MYLIB)
   ```

   y pulse Intro.

   El mandato CRTLIB crea una biblioteca cuyo nombre es MYLIB.

2. Para crear un archivo físico fuente cuyo nombre sea QCBLLESRC, teclee

   ```
   CRTSRCPF FILE(MYLIB/QCBLLESRC)
   TEXT ('Archivo físico fuente para un programa ILE COBOL/400')
   ```

   y pulse Intro.

   El mandato CRTSRCPF crea un archivo físico fuente cuyo nombre es QCBLLESRC en la biblioteca MYLIB.

3. Para arrancar una sesión de edición y crear un miembro de archivo físico fuente XMPLE1, teclee

   ```
   STRSEU SRCFILE(MYLIB/QCBLLESRC) SRCMBR(XMPLE1)
   TYPE(CBLLE) OPTION(2)
   ```

   y pulse Intro.

   El mandato STRSEU crea un nuevo miembro XMPLE1 en el archivo QCBLLESRC de la biblioteca MYLIB.

   La pantalla Editar de SEU aparece como se muestra en la Figura 6.

```
 Columnas. . . :    1  71           Editar                        MYLIB/QCBLLESRC
 SEU==> ____________________________________________________________      XMPLE1
 FMT CB ......-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ********** Principio de datos ************************************
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
'''''''
        ****************** Fin de datos ***************************************

 F3=Salir  F4=Solicitud  F5=Renovar  F9=Recuperar  F10=Cursor  F11=Conmutar
 F16=Repetir búsqueda    F17=Repetir cambio        F24=Más teclas
El miembro XMPLE1 se ha añadido al archivo MYLIB/QCBLLESRC.                    +
```

Figura 6. Pantalla Editar para un miembro nuevo

4. Teclee el fuente en la pantalla Editar de SEU.
5. Pulse F3 (Salir) para ir a la pantalla Salir. Teclee Y (Sí) para salvar XMPLE1 y pulse Intro.

   El miembro XMPLE1 se salva.

Utilización de identificadores del juego de caracteres codificados

Un identificador de un juego de caracteres codificados (CCSID) es un número que identifica un juego específico de identificadores de esquemas de codificación, identificadores de juegos de caracteres, identificadores de páginas de códigos y más información relacionada con la codificación que identifica de forma exclusiva la representación de caracteres gráficos codificados que se utiliza.

Todos los caracteres ILE COBOL/400 válidos excepto el signo $ se encuentran entre los caracteres invariables del juego de caracteres sintáctico CS640 y todos los caracteres tienen el mismo elemento de código en las páginas de códigos EBCDIC de un sólo byte excepto la página de códigos Katakana (que se utiliza en Japón).

Nota: En la página de códigos Katakana, a los caracteres Katakana se les asignan los elementos de códigos que se asignan a los caracteres alfabéticos en minúscula en otras páginas de códigos EBCDIC de un sólo byte.

El compilador ILE COBOL/400 aceptará el código fuente escrito en cualquier CCSID EBCDIC de un solo byte excepto el CCSID 290 (Katakana).

Los CCSID pueden ayudarle a mantener la integridad de los datos de tipo carácter entre sistemas.

La Arquitectura de representación de datos de tipo carácter (CDRA) define los valores de CCSID para identificar los elementos de código que se utilizan para representar caracteres y convertir estos códigos como sea necesario para preservar su significado.

Las instrucciones ACCEPT y DISPLAY ampliadas no dan soporte a la conversión de CCSID.

Asignación de un CCSID a un archivo físico fuente

Se asigna un CCSID a todos los archivos fuente en el momento en que se crea en el sistema. Se puede especificar, de forma explícita, el juego de caracteres que se desea utilizar con el parámetro CCSID del mandato CRTSRCPF cuando se crea el archivo físico fuente, o se puede aceptar el valor por omisión *DFTCCSID. Por ejemplo, para crear un archivo físico fuente con CCSID 273, teclee:

```
CRTSRCPF FILE(MYLIB/QCBLLESRC) CCSID(273)
```

Si se acepta el valor por omisión, se asignará el CCSID del trabajo al archivo físico fuente. El CCSID que se asigne depende de la página de códigos que utilizaba la máquina AS/400 en la que se creó el archivo fuente.

El CCSID por omisión para un sistema AS/400 es el CCSID 65535. Si el CCSID del sistema es 65535, el CCSID que se asigne al archivo físico fuente viene determinado por el identificador del lenguaje del trabajo.

Inclusión de miembros de copia con CCSID diferentes en el archivo fuente

El programa fuente ILE COBOL/400 puede constar de más de un archivo fuente. Puede haber un archivo fuente primario y varios archivos fuente secundarios como, por ejemplo, miembros de copia y archivos DDS.

Los archivos fuente secundarios pueden tener CCSID diferentes del CCSID del archivo fuente primario. En este caso, el contenido de los archivos secundarios se convierte al CCSID de los archivos fuente primarios a medida que el compilador ILE COBOL/400 los procesa.

El CCSID 65535 implica que no se va a efectuar ninguna conversión del archivo fuente.Si el archivo fuente primario, el archivo fuente secundario o ambos tienen asignado el CCSID 65535, no se lleva a cabo ninguna conversión. El compilador ILE COBOL/400 podría informar de un error de sintaxis si el archivo fuente secundario contiene caracteres que el juego de caracteres especificado por el CCSID del archivo fuente primario no reconoce.

Cuando una instrucción COPY de Formato 2 se utiliza para incorporar descripciones de archivo DDS en el programa fuente, la conversión CCSID no se lleva a cabo. Si el fuente DDS tiene un CCSID diferente que el miembro fuente en el que se está copiando, el fuente DDS copiado puede contener algunos caracteres que no sean válidos.Estos caracteres se señalarán como errores de sintaxis.

Si el archivo fuente primario y los archivos fuente secundarios tiene un CCSID diferente y ninguno es el CCSID 65535, el rendimiento de la compilación puede verse afectado. El compilador ILE COBOL/400 debe invertir tiempo convirtiendo los archivos fuente secundarios de un CCSID al CCSID del archivo fuente primario. Este tiempo puede resultar significativo según el tamaño de los archivos fuente.

Definición del CCSID para el corrector sintáctico COBOL en el SEU

Para que el corrector sintáctico COBOL en el SEU se comporte de igual manera que el compilador ILE COBOL/400, debe definir el CCSID del trabajo que ejecuta el SEU para que sea igual que el CCSID del archivo fuente primario que está editando. En la mayoría de los casos, ya son iguales.Sin embargo, si son diferentes, se puede modificar el CCSID del trabajo especificando el número de CCSID nuevo en el parámetro CCSID del mandato CHGJOB. Por ejemplo, para modificar el CCSID del trabajo actual y establecerlo en 280, teclee:

```
CHGJOB CCSID(280)
```

Para obtener más información sobre cómo modificar los atributos de un trabajo, consulte el mandato CHGJOB en el manual *CL Reference*.

Manipulación de CCSID diferentes con el depurador fuente ILE

Consulte el apartado “Soporte de idioma para el depurador fuente ILE” en la página 148 para obtener una descripción sobre cómo el depurador fuente ILE maneja los diferentes CCSID.

Capítulo 3. Compilación de programas fuente en objetos de módulo

El compilador ILE COBOL/400 no genera objetos de programa ejecutables. Produce uno o más objetos de módulo que pueden enlazarse entre sí en diferentes combinaciones para formar una o más unidades ejecutables conocidas como objetos de programa. Para obtener más información sobre la creación de objetos de programa ejecutables, consulte el Capítulo 4, “Creación de un objeto de programa” en la página 71.

Este capítulo describe:

- cómo crear un objeto de módulo
- el mandato CRTCBLMOD y sus parámetros
- cómo utilizar la instrucción PROCESS para especificar opciones de compilador
- cómo entender la salida que genera el compilador ILE COBOL/400

Definición de un objeto de módulo

Los **objetos de módulo** son la salida producida por todos los compiladores ILE incluyendo el compilador ILE COBOL/400. Son objetos del sistema de tipo *MODULE. Para ILE COBOL/400, el nombre de cualquier objeto de módulo creado de forma permanente viene determinado por el mandato CRTCBLMOD o el párrafo PROGRAM-ID del programa fuente ILE COBOL/400 más externo. Cada unidad de compilación de un miembro fuente crea un objeto de módulo separado. Un programa ILE COBOL/400 de un objeto de módulo puede llamar al programa ILE COBOL/400 más externo de un objeto de módulo diferente a través de una llamada enlazada de procedimiento. También pueden llamarlo utilizando una llamada de dinámica de programas después de haber enlazado el objeto de módulo en un objeto de programa. Consulte el apartado “Llamada a un programa ILE COBOL/400” en la página 179 para obtener una descripción de llamadas enlazadas de procedimiento y llamadas dinámicas de programa.

Los objetos de módulo no pueden ejecutarse por sí mismos. Primero, deben enlazarse en un objeto de programa. Puede enlazar uno o más objetos de módulo entre sí para crear un objeto de programa (tipo *PGM) o un programa de servicio (tipo *SRVPGM). Esta posibilidad de combinar objetos de módulo le permite:

- Volver a utilizar fragmentos de código dando como resultado programas más pequeños.
- Compartir código entre varios programas y, por consiguiente, evitar que se produzcan errores en otras partes del programa general mientras se actualiza una sección compartida.
- Combinar lenguajes para seleccionar el que mejor lleva a cabo la tarea que se tienen que realizar.

Un objeto de módulo puede constar de uno o varios procedimientos ILE. .* The number of procedures allowed is language dependent.

El mandato Crear módulo COBOL (CRTCBLMOD) crea uno o más objetos de módulo a partir de instrucciones fuente ILE COBOL/400. Estos objetos de módulo

permanecen almacenados en la biblioteca designada hasta que se sustituyan o eliminen de forma explícita. Los objetos de módulo se pueden enlazar más tarde en un objeto de programa ejecutable utilizando el mandato Crear programa (CRTPGM) o en un programa de servicio utilizando el mandato Crear programa de servicio (CRTSRVPGM). Los objetos de módulo todavía se encuentran en la biblioteca después de haber creado el objeto de programa o el programa de servicio. Para obtener más información sobre la creación de un objeto de programa a partir de uno o más objetos de módulo, consulte el apartado “Utilización del mandato Crear programa (CRTPGM)” en la página 73. Para obtener más información sobre la creación de un programa de servicio a partir de uno o más objetos de módulo, consulte el Capítulo 5, “Creación de un programa de servicio” en la página 101.

El mandato Crear programa COBOL enlazado (CRTBNDCBL) crea objeto(s) de programa a partir de instrucciones fuente ILE COBOL/400 en un solo paso.CRTBNDCBL crea objetos de módulo; sin embargo, estos objetos de módulo se crean sólo de forma temporal y no pueden volver a utilizarse. Cuando CRTBNDCBL ha completado la creación de objetos de programa, los objetos de módulo temporales se suprimen.

Para obtener más información sobre la creación de un objeto de programa en un sólo paso, consulte el apartado “Utilización del mandato Crear COBOL enlazado (CRTBNDCBL)” en la página 76.

Cuando se crea un objeto de módulo, éste puede contener:

- Datos de depuración

  Los **datos de depuración** son los datos necesarios para depurar un objeto de programa utilizando el depurador fuente ILE. Estos datos se generan en base a la opción especificada en el parámetro DBGVIEW de los mandatos CRTCBLMOD o CRTBNDCBL.

- Procedimiento de entrada de programa (PEP)

  Un **procedimiento de entrada de programa** es el código generado por el compilador que es el punto de entrada para un objeto de programa en una llamada dinámica de programa. El control se pasa al PEP de un objeto de programa cuando éste se llama utilizando una llamada dinámica de programa. Es semejante al código que se proporciona para el punto de entrada de un programa OPM. EL PEP identifica el procedimiento ILE dentro de un objeto de módulo que debe ejecutarse en primer lugar cuando se llama a su objeto de programa utilizando una llamada dinámica de programa. Cuando el compilador ILE COBOL/400 crea un objeto de módulo, se genera un PEP. Este PEP llama al programa ILE COBOL/400 más externo que se encuentra en la unidad de compilación.

  Cuando se enlazan varios objetos de módulo entre sí para crear un objeto de programa, debe especificarse qué objeto de módulo tendrá el PEP del objeto de programa que se está creando. Esto se lleva a cabo identificando el objeto de módulo en el parámetro ENTMOD del mandato CRTPGM. El PEP de este objeto de módulo se convierte entonces en el PEP del objeto de programa. Los PEP del resto de objetos de módulo se suprimen lógicamente del objeto de programa.

- Procedimiento de entrada de usuario (UEP)

Cuando el compilador ILE COBOL/400 crea un objeto de módulo, el programa ILE COBOL/400 más externo que se encuentra en la unidad de compilación es el **procedimiento de entrada de usuario**. Durante una llamada dinámica de programa, el UEP es el procedimiento ILE que obtiene el control del PEP. Durante llamadas estáticas de procedimiento entre procedimientos ILE de diferentes objetos de módulo, el UEP recibe el control directamente.

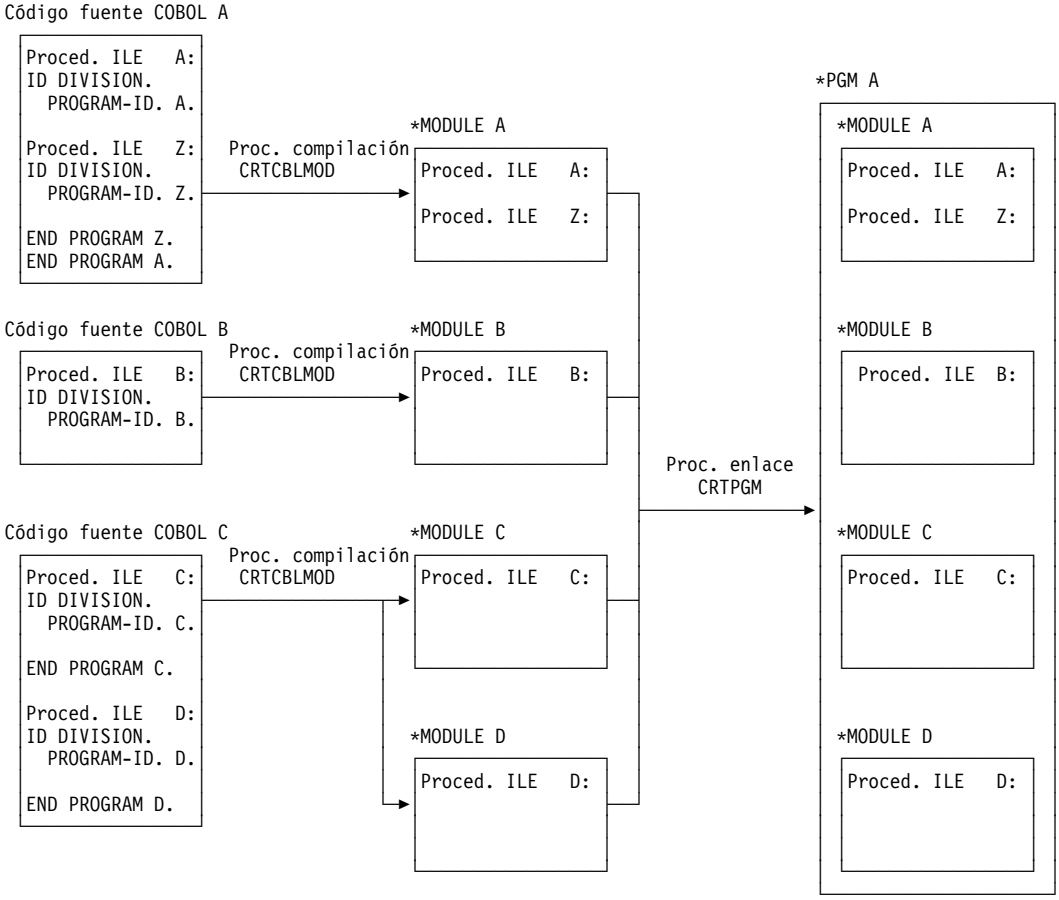


Figura 7. Creación de objetos de módulo utilizando el mandato CRTCBLMOD

En la Figura 7, *PGM A se crea y se designa *MODULE A como el objeto de módulo que contiene el punto de entrada para el objeto de programa. El PEP para *MODULE A llama al Procedimiento ILE A. El PEP para *MODULE A también pasa a ser el PEP para *PGM A así el PEP para *PGM A llama al Procedimiento ILE A. El UEP para *PGM A es también el Procedimiento ILE A. *MODULE B, *MODULE C y *MODULE D también tienen PEP pero *PGM A hace caso omiso de ellos. Además, el Procedimiento Z de ILE sólo puede llamarse desde el procedimiento A de ILE. EL Procedimiento Z de ILE no es visible para los Procedimientos B, C ni D de ILE ya que se encuentran en objetos de módulo separados y el Procedimiento Z de ILE no es el programa COBOL más externo en el *MODULE A. Los Procedimientos A, B, C y D de ILE sí que pueden llamarse entre sí. No se permite la recurrencia.

Cada procedimiento de declaración de un programa fuente ILE COBOL/400 genera un procedimiento ILE separado.

Cada programa COBOL anidado genera un procedimiento ILE separado.

Un objeto de módulo puede tener exportaciones e importaciones de módulo asociados a él.

Una **exportación de módulo** es el nombre de un procedimiento o elemento de datos que otros objetos ILE pueden utilizar mediante el proceso de enlace. La exportación de módulo se identifica por su nombre y el tipo que tiene asociado: procedimiento o datos. El ámbito de las exportaciones de módulo abarca: el objeto de programa y el grupo de activación.No todos los nombres que se exportan al objeto de programa se exportan al grupo de activación. El compilador ILE COBOL/400 crea exportaciones de módulo para cada una de las siguientes construcciones de lenguaje de programación COBOL:

- Un nombre de procedimiento correspondiente al programa ILE COBOL/400 más externo de una unidad de compilación.
- Un nombre de procedimiento de cancelación correspondiente al programa ILE COBOL/400 más externo de una unidad de compilación.
- Una exportación débil de un archivo EXTERNAL o de datos EXTERNAL.

Una **importación de módulo** es la utilización de o referencia al nombre de un procedimiento o elemento de datos no definido en un objeto de módulo al que se hace referencia. La importación de módulo se identifica por su nombre y el tipo que tiene asociado, o bien procedimiento o datos. El compilador ILE COBOL/400 crea importaciones de módulo para cada una de las siguientes construcciones de lenguaje de programación COBOL:

- Un nombre de procedimiento correspondiente al programa ILE COBOL/400 que se llama utilizando una llamada estática de procedimiento.
- Un nombre de procedimiento de cancelación correspondiente a un programa ILE COBOL/400 que se llama utilizando una llamada estática de procedimiento.
- Una importación débil de un archivo EXTERNAL o de datos EXTERNAL.
- Un nombre de procedimiento correspondiente a un programa ILE COBOL/400 establecido mediante la instrucción SET elemento-puntero-procedimiento TO ENTRY nombre-procedimiento, donde el nombre-procedimiento va a interpretarse como un procedimiento ILE.

La importación de módulo se genera cuando el procedimiento destino no está definido en el objeto de módulo al que hace referencia. Una importación débil de elementos de datos se genera cuando se hace referencia a elementos de datos del programa ILE COBOL/400

Utilización del mandato Crear módulo COBOL (CRTCBLMOD)

Para compilar instrucciones fuente ILE COBOL/400 en uno o más objetos de módulo, debe utilizar el mandato CRTCBLMOD. Este mandato arranca el compilador ILE COBOL/400 que crea objeto(s) de módulo en base a las instrucciones ILE COBOL/400 del miembro fuente. El mandato CRTCBLMOD se puede utilizar de forma interactiva, en modalidad de proceso por lotes o desde un programa CL del sistema AS/400.

Nota: Para crear un objeto de módulo con el mandato CRTCBLMOD debe tener autorización para utilizar el mandato.

Si se utiliza el formato 2 de la instrucción COPY en el programa para acceder a archivos descritos externamente, el sistema operativo proporciona al programa compilado información sobre los archivos descritos externamente.

Si el compilador ILE COBOL/400 se detiene, se emitirá el mensaje LNC9001

```
Se ha producido un error en la compilación.
nombre-módulo no se ha creado.
```

Puede utilizar un programa de lenguaje de control que supervise esta excepción utilizando el mandato Supervisar mensaje (MONMSG) de CL.

Utilización de pantallas de solicitud con el mandato CRTCBLMOD

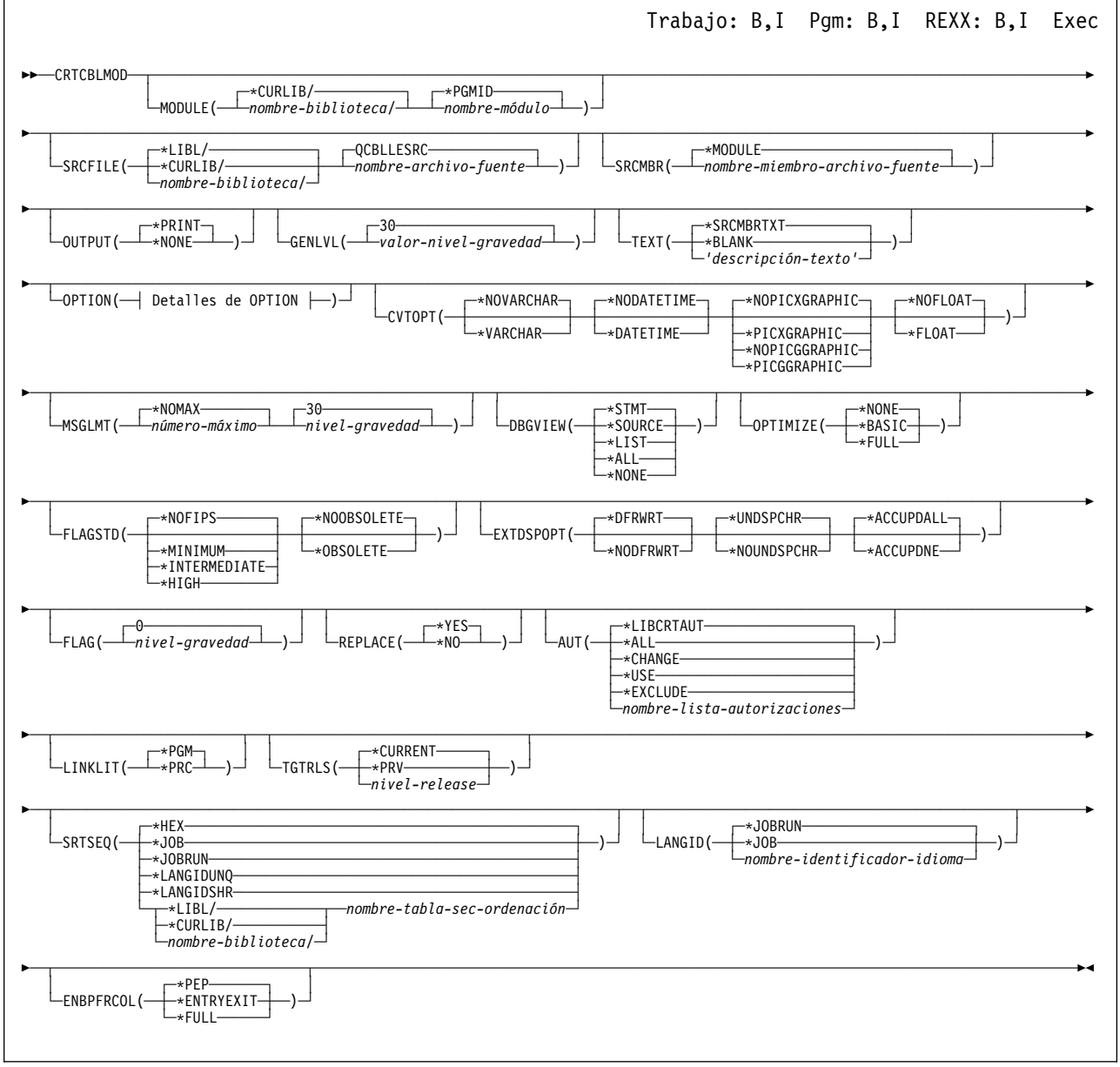
Se puede entrar el mandato CRTCBLMOD utilizando las pantallas de solicitud. Para entrar parámetros del mandato con este método, teclee CRTCBLMOD y pulse F4.

Todos los parámetros de la pantalla muestran un valor por omisión. Teclee sobre los elementos para establecer diferentes opciones o valores. Si no está seguro sobre cómo establecer un valor de un parámetro, teclee un signo de interrogación (?) en la primera posición del campo y pulse Intro, o F4 (Solicitud), para recibir información más detallada. El signo de interrogación debe ir seguido de un blanco. Si ha entrado algunos parámetros antes de pedir la pantalla de solicitud, se visualizarán los valores que haya entrado para estos.

Para obtener la descripción de los parámetros del mandato CRTCBLMOD, consulte el apartado “Parámetros del mandato CRTCBLMOD” en la página 33.

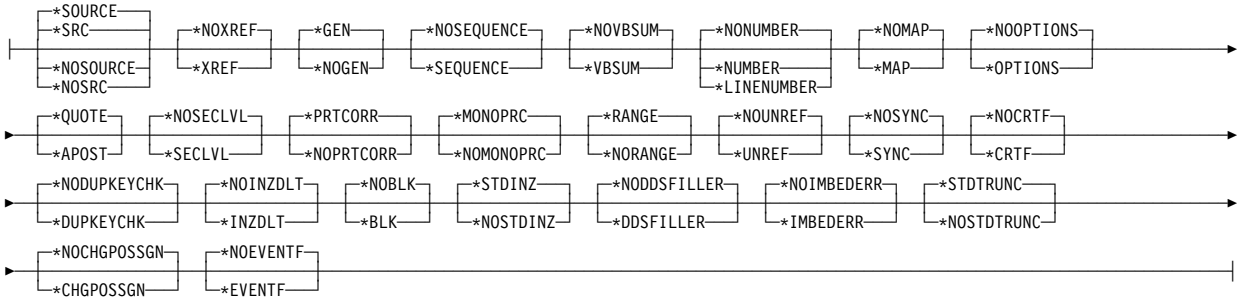
Sintaxis del mandato CRTCBLMOD

Mandato CRTCBLMOD - Formato



Mandato CRTCBLMOD - Formato (continuación)

Detalles de OPTION:



Parámetros del mandato CRTCBLMOD

En este apartado encontrará una descripción de los parámetros del mandato CRTCBLMOD. Los parámetros y opciones se describen en el orden en que aparecen en las pantallas de solicitud.

Los valores por omisión se visualizan primero y aparecen subrayados.

Todos los nombres de objeto especificados para el mandato CRTCBLMOD deben seguir el convenio de denominación de AS/400: los nombres deben tener 10 caracteres de longitud, estar compuestos por caracteres alfanuméricos, siendo el primero un carácter alfabético; o pueden ser nombres entrecomillados, de ocho caracteres de longitud, incluidos entre comillas.

Puede especificar varias opciones de compilador utilizando el parámetro OPTION del mandato CRTCBLMOD o desde el programa fuente utilizando la instrucción PROCESS. Cualquier opción que se especifique en la instrucción PROCESS altera temporalmente las opciones correspondientes del mandato CRTCBLMOD.

Parámetro MODULE:

Especifica el nombre de módulo y el nombre de biblioteca para el objeto de módulo que se está creando. EL nombre de módulo y el nombre de biblioteca deben cumplir las normas del convenio de denominación de AS/400. Los valores posibles son:

***PGMID**

EL nombre del módulo se toma del párrafo PROGRAM-ID del programa fuente ILE COBOL/400 más externo de la unidad de compilación.

nombre-módulo

Entre un nombre para identificar el módulo ILE COBOL/400 compilado. Si se especifica un nombre de módulo para este parámetro y se compila una *secuencia de programas fuente* (varias unidades de compilación en un único miembro de archivo fuente), el primer módulo de la secuencia utiliza este nombre; el resto de módulos utilizan el nombre especificado en el párrafo PROGRAM-ID del correspondiente programa fuente ILE COBOL/400 más externo de la unidad de compilación.

Los posibles valores para la biblioteca son:

***CURLIB**

El objeto de módulo creado se almacena en la biblioteca actual. Si no se ha asignado una biblioteca como biblioteca actual, se utiliza QGPL.

nombre-biblioteca

Entre el nombre de la biblioteca donde se va a almacenar el objeto de módulo creado.

Parámetro SRCFILE:

Especifica el nombre de la biblioteca y el archivo fuente que contiene el código fuente ILE COBOL/400 que se va a compilar. La

longitud de registro de este archivo fuente debería ser 92. Los valores posibles son:

QCBLLESRC
Especifica que el archivo fuente, QCBLLESRC, contiene el código fuente ILE COBOL/400 que se va a compilar.

nombre-archivo-fuente
Entre el nombre del archivo fuente que contiene el código fuente ILE COBOL/400 que se va a compilar.

Los posibles valores para la biblioteca son:

***LIBL**
Se busca en la lista de bibliotecas, la biblioteca donde se ubica el archivo fuente.

***CURLIB**
Se utiliza la biblioteca actual.Si no se ha asignado una biblioteca como biblioteca actual, se utiliza QGPL.

nombre-biblioteca
Entre el nombre de la biblioteca donde se ubica el archivo fuente.

Parámetro SRCMBR:
Especifica el nombre del miembro que contiene el código fuente ILE COBOL/400 que se va a compilar.Se puede especificar este parámetro sólo si el archivo fuente al que se hace referencia en el parámetro SCRFILE es un archivo de base de datos. Los valores posibles son:

***MODULE**
Se utiliza el miembro de archivo fuente cuyo nombre es igual al nombre de módulo especificado en el parámetro MODULE.

Si no especifica un nombre de módulo para el parámetro MODULE, se utiliza el primer miembro de archivo fuente de la base de datos.

nombre-miembro-archivo-fuente
Entre el nombre del miembro que contiene el código fuente ILE COBOL/400.

Parámetro OUTPUT:
Especifica si se genera o no se genera el listado del compilador. Los valores posibles son:

***PRINT**
Se genera un listado de compilador.

***NONE**
No se genera un listado de compilador.

Parámetro GENLVL:
Especifica el nivel de gravedad que determina si se va a crear un objeto de módulo. El nivel de gravedad corresponde al nivel de gravedad de los mensajes que se generan durante la compilación. Este parámetro se aplica de modo individual a cada una de las unidades de compilación de un miembro de archivo fuente. El resto de unidades de compilación del miembro del archivo fuente también se compilarán aunque se produzcan anomalías en una unidad de compilación anterior.

Los valores posibles son:

30 No se crea ningún objeto de módulo si se genera un error cuyo nivel de gravedad sea igual o mayor que 30.

nivel-gravedad
Especifique un número de uno o dos dígitos, de 0 a 30. Este número es el nivel de gravedad que se desea utilizar para determinar si un objeto de módulo se va a crear o no. No se crearán objetos de módulo si se genera un error con un nivel de gravedad igual o mayor que este nivel de gravedad.

Parámetro TEXT:
Le permite entrar texto que describe brevemente el módulo y su función.

***SRCMBRTXT**
Para describir el objeto de módulo se utiliza el mismo texto que describe el miembro del archivo de base de datos que contiene el código fuente ILE COBOL/400. Si el fuente proviene de un dispositivo o archivo incorporado, especificar *SRCMBRTXT tiene el mismo efecto que especificar *BLANK.

***BLANK**
No se especifica texto.

descripción-texto
Entre texto que describa brevemente el módulo y su función.El texto puede tener un máximo de 50 caracteres SBCS de longitud y debe ir incluido entre

apóstrofos.Los apóstrofos no forman parte de la serie de 50 caracteres.

Parámetro OPTION:

Especifica las opciones para utilizar cuando se compila el código fuente ILE COBOL/400.

Las opciones que se especifican en la instrucción PROCESS de un programa fuente ILE COBOL/400 alteran temporalmente las opciones correspondientes del parámetro OPTION.

Los valores posibles para el parámetro OPTION son:

***SOURCE o *SRC**

El compilador genera un listado fuente en el que se incluyen todos los mensajes de error producidos durante la compilación y el programa fuente ILE COBOL/400.

***NOSOURCE o *NOSRC**

El compilador no genera la parte del fuente del listado. Si no necesita un listado fuente debería utilizar esta opción, ya que entonces la compilación se realizará más rápido.

***NOXREF**

El compilador no genera un listado de referencias cruzadas para el programa fuente ILE COBOL/400.

***XREF**

El compilador genera un listado de referencias cruzadas para el programa fuente.

***GEN**

El compilador crea un objeto de módulo una vez que se ha compilado el fuente ILE COBOL/400.

***NOGEN**

El compilador no crea un objeto de módulo una vez que se ha compilado el programa fuente ILE COBOL/400. Puede especificar esta opción si sólo desea mensajes de error o listados.

***NOSEQUENCE**

No se comprueba si existen errores de secuencia en los números de referencia.

***SEQUENCE**

Se comprueba si existen errores de secuencia en los números de referencia. No se producen errores de secuencia si se ha especificado la opción *LINENUMBER.

***NOVBSUM**

No se imprime el recuento de utilización de verbos.

***VBSUM**

Se imprime el recuento de utilización de verbos.

***NONUMBER**

Los números de secuencia del archivo fuente se utilizan para números de referencia.

***NUMBER**

Los números de secuencia proporcionados por el usuario (columnas 1 a 6) se utilizan para números de referencia.

***LINENUMBER**

Se utilizan los números de secuencia creados por el compilador para números de referencia. Esta opción combina el código fuente del programa ILE COBOL/400 y el código fuente introducido mediante las instrucciones COPY en una secuencia numerada consecutivamente. Utilice esta opción si especifica la opción para señalar con distintivos FIPS (Federal Information Processing Standards).

***NOMAP**

El compilador no lista el mapa de la Data Division.

***MAP**

El compilador lista el mapa de la Data Division.

***NOOPTIONS**

Las opciones que se encuentran en vigor no se listan para esta compilación.

***OPTIONS**

Las opciones que se encuentran en vigor se listan para esta compilación.

***QUOTE**

Especifica que el delimitador comillas (") se utiliza para literales no numéricos, literales hexadecimales y literales booleanos. Esta opción también especifica que el valor de la constante figurativa QUOTE tiene el valor EBCDIC de unas comillas.

***APOST**
Especifica que el delimitador apóstrofo (') se utiliza para literales no numéricos, literales hexadecimales y literales booleanos. Esta opción también especifica que el valor de la constante figurativa QUOTE tiene el valor EBCDIC de un apóstrofo.

***NOSECLVL**
No se lista el texto de mensajes de segundo nivel para esta compilación.

***SECLVL**
En la sección de mensajes del listado del compilador, se lista el texto de mensajes de segundo nivel para esta compilación junto con el texto de errores de primer nivel.

***PRTCORR**
Las líneas de comentarios se insertan en el listado de compilador indicando qué elementos primarios se incluyeron como resultado de la utilización de la expresión CORRESPONDING.

***NOPRTCORR**
Las líneas de comentario no se insertan en el listado del compilador cuando se utiliza la expresión CORRESPONDING.

***MONOPRC**
El nombre-programa (literal o palabra) que se encuentra en el párrafo PROGRAM-ID, las instrucciones CALL, CANCEL o SET ENTRY y la cabecera END PROGRAM se convierten a caracteres en mayúsculas y se hacen cumplir las normas para la formación del nombre-programa.

***NOMONOPRC**
El nombre-programa (literal o palabra) que se encuentra en el párrafo PROGRAM-ID, las instrucciones CALL, CANCEL o SET ENTRY y la cabecera END PROGRAM no se convierten a caracteres en mayúsculas y no se hacen cumplir las normas para la formación del nombre-programa. Esta opción permite que se utilicen en el CALL destino caracteres especiales no permitidos para la norma COBOL.

***RANGE**
En el momento de la ejecución, se comprueba si los subíndices se encuentran entre los rangos correctos pero los rangos de índice no se comprueban. También se comprueban las operaciones de subserie generadas por el compilador y la modificación de referencias.

***NORANGE**
Los rangos no se comprueban en el momento de la ejecución.

Nota: La opción *RANGE genera un código para la comprobación de rangos de subíndices. Por ejemplo, se asegura de que no se está intentando acceder al elemento 21 de una matriz de 20 elementos.

La opción *NORANGE no genera un código para comprobar los rangos de subíndice. Como resultado, la opción *NORANGE genera un código de ejecución más rápido.

***NOUNREF**
Los elementos de datos no referenciados no se incluyen en el módulo compilado. Esto reduce la cantidad del almacenamiento utilizado, de forma que puede compilarse un programa más grande. No puede verse ni asignarse a un elemento de datos no referenciado durante la depuración cuando se selecciona la opción *NOUNREF. Los elementos de datos no referenciados seguirán apareciendo en los listados de referencias cruzadas al especificar OPTION (*XREF).

***UNREF**
Los elementos de datos no referenciados se incluyen en el módulo compilado.

***NOSYNC**
Se comprueba sólo la sintaxis de la cláusula SYNCHRONIZED.

***SYNC**
EL compilador compila la cláusula SYNCHRONIZED. La cláusula SYNCHRONIZED hace que los elementos de datos se alineen de tal forma que el extremo derecho (menos significativo) se encuentre en el límite normal del almacenamiento. El límite normal de almacenamiento es el siguiente al límite de 4 bytes, 8 bytes o 16 bytes más cercano según la longitud y el tipo de datos que se

almacenen. Para conseguir este alineamiento se reserva almacenamiento suplementario contiguo al elemento sincronizado. Cada uno de los elementos de datos primarios que se describen como SYNCHRONIZED se alinea con el límite normal de almacenamiento que corresponde a su asignación de almacenamiento de datos.

***NOCRTF**

Los archivos de disco que no se encuentran disponibles cuando se efectúa una operación OPEN no se crean de forma dinámica.

***CRTF**

Los archivos de disco que no se encuentran disponibles cuando se efectúa una operación OPEN se crean de forma dinámica.

Nota: La longitud de registro máxima para un archivo que se va a crear dinámicamente es 32 766. Los archivos indexados no se crearán dinámicamente aunque se haya especificado la opción *CRTF.

***NODUPKEYCHK**

No comprueba la existencia de claves primarias duplicadas para archivos INDEXED.

***DUPKEYCHK**

Comprueba la existencia de claves primarias duplicadas para archivos INDEXED.

***NOINZDLT**

Los archivos relativos con acceso secuencial no se inicializan con registros suprimidos durante la operación CLOSE si éstos se han abierto para OUTPUT. El límite de registros se determina mediante el número de registros grabados durante la operación OPEN OUTPUT. Las operaciones OPEN siguientes sólo permiten el acceso hasta el límite antes indicado.

***INZDLT**

Los archivos relativos con acceso secuencial se inicializan con registros suprimidos durante la operación CLOSE si éstos se abrieron para OUTPUT. Los registros activos de los archivos no se ven afectados. Para las operaciones OPEN siguientes, el límite de registro se define como el tamaño de archivo.

***NOBLK**

El compilador permite la agrupación en bloques sólo de archivos de acceso SEQUENTIAL sin la instrucción START. La cláusula BLOCK CONTAINS, si se especifica, se hace caso omiso excepto para los archivos de cintas.

***BLK**

Cuando se utiliza *BLK y se ha especificado una cláusula BLOCK CONTAINS, el compilador permite la agrupación en bloques para archivos de acceso DYNAMIC y archivos de acceso SEQUENTIAL con una instrucción START.No se permite la agrupación en bloques para archivos RELATIVE abiertos para operaciones de salida. La cláusula BLOCK CONTAINS contiene el número de registros que van a agruparse en bloques.

Cuando *BLK se utiliza y no se especifica una cláusula BLOCK CONTAINS, el compilador permite la agrupación en bloques sólo de archivos de acceso SEQUENTIAL sin instrucción START. El sistema operativo determina el número de registros que van a agruparse en bloques.

***STDINZ**

Para aquellos elementos que no tienen la cláusula VALUE, el compilador inicializa los elementos de datos a los valores por omisión del sistema.

***NOSTDINZ**

Par aquellos elementos que no tienen la cláusula VALUE, el compilador no inicializa los elementos de datos a los valores por omisión del sistema.

***NODDSFILLER**

Si no se encuentran campos coincidentes con una instrucción COPY DDS, no se generan descripciones de campo.

***DDSFILLER**

Si no se encuentran campos coincidentes con una instrucción COPY DDS, siempre se crea una única descripción de campo FILLER de caracteres, "07 FILLER PIC X".

***NOIMBEDERR**

Los mensajes de error no se incluyen en la sección del listado fuente del listado del compilador. Los mensajes de error únicamente aparecen en la sección de mensajes de error del listado del compilador.

***IMBEDERR**

Los mensajes de error de primer nivel se incluyen en la sección de listado fuente del listado del compilador, inmediatamente después de la línea donde se ha producido el error. Los mensajes de error también aparecen en la sección de mensajes de error del listado del compilador.

***STDTRUNC**

Esta opción sólo se aplica a los datos USAGE BINARY. Cuando se selecciona *STDTRUNC, los datos USAGE BINARY se truncan en el número de dígitos que se indica en la cláusula PICTURE del campo receptor BINARY.

***NOSTDTRUNC**

Esta opción sólo se aplica a los datos USAGE BINARY. Cuando se selecciona *NOSTDTRUNC, los campos receptores BINARY se truncan únicamente en los límites de media palabra, palabra completa o palabra doble. Los campos de envío BINARY también se manejan como medias palabras, palabras completas o palabras dobles. Por consiguiente, el contenido binario completo del campo es significativo. Asimismo, la instrucción DISPLAY convertirá todo el contenido de un campo BINARY, sin truncarlo.

Nota: *NOSTDTRUNC no tiene efecto en la cláusula VALUE.

***NOCHGPOSSGN**

Se utiliza el hexadecimal F como el signo positivo por omisión para datos numéricos empaquetados y con zona.El hexadecimal F es el valor por omisión del sistema para el sistema operativo OS/400.

***CHGPOSSGN**

Se utiliza el hexadecimal C como el signo positivo por omisión para datos numéricos empaquetados y con zona. Esto se aplica a todos los resultados de las instrucciones MOVE, ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE e INITIALIZE, así como a los resultados de la cláusula VALUE.

***NOEVENTF**

No crear un Archivo de eventos para su utilización con Entorno de Desarrollo Coorperativo CODE/400 (CODE/400). CODE/400 utiliza este archivo para proporcionar información sobre el error integrada con el editor CODE/400. Un Archivo de eventos suele crearse al crear un módulo o programa desde CODE/400.

***EVENTF**

Crear un Archivo de eventos para su utilización con CODE/400. El Archivo de eventos se crea como un miembro en el archivo EVFEVENT de la biblioteca en la que va a almacenarse el módulo o programa que se ha creado. Si no existe el archivo EVFEVENT, éste se crea de forma automática. El nombre de miembro del Archivo de eventos es el mismo que el del objeto que se crea.

CODE/400 utiliza este archivo para proporcionar información sobre el error integrada con el editor CODE/400. Un Archivo de eventos suele crearse al crear un módulo o programa desde CODE/400.

Parámetro CVTOPT:

Especifica la forma en que el compilador maneja los tipos de campo de fecha, hora e indicación de hora, el tipo de campo DBCS, los tipos de campo de caracteres de longitud variable y los tipos de campo de coma flotante que han pasado de archivos descritos externamente al programa mediante COPY DDS. Los valores posibles son:

***NOVARCHAR**

Los campos de longitud variable se declaran como campos FILLER.

***VARCHAR**

Los campos de longitud variable se declaran como elemento de grupo y son accesibles para el programa fuente ILE COBOL/400.

***NODATETIME**

Los tipos de datos de fecha, hora e indicación de hora se declaran como campos FILLER.

***DATETIME**
Los tipos de datos de fecha, hora e indicación de hora se declaran como campos de tipo carácter y son accesibles para el programa fuente COBOL.

***NOPICXGRAPHIC**
Los tipos de datos gráficos DBCS se declaran como campos FILLER.

***PICXGRAPHIC**
Los tipos de datos gráficos DBCS de longitud fija se declaran como campos alfanuméricos de longitud fija y son accesibles para el programa fuente ILE COBOL/400.

Cuando la opción *VARCHAR también se utiliza, los tipos de datos gráficos DBCS de longitud fija se declaran como elementos de grupo de longitud fija y son accesibles para el programa fuente ILE COBOL/400

***PICGGRAPHIC**
Los tipos de datos gráficos DBCS de longitud fija se declaran como campos tipo G de longitud fija y son accesibles para el programa fuente ILE COBOL/400.

Cuando también se utiliza la opción *VARCHAR, los tipos de datos gráficos DBCS de longitud variable se declaran como elementos de grupo de longitud fija (formados por un campo numérico seguido de un campo tipo G) y son accesibles para el programa fuente ILE COBOL/400.

***NOPICGGRAPHIC**
Los tipos de datos gráficos DBCS se declaran como campos FILLER. *NOPICGGRAPHIC aparecerá como *NOPICXGRAPHIC en el listado.

***NOFLOAT**
Los tipos de datos de coma flotante se declaran como campos FILLER con un USAGE binario.

***FLOAT**
Los tipos de datos de coma flotante se transfieren al programa con los nombres de las DDS y un USAGE de COMP-1 (precisión simple) o COMP-2 (precisión doble). Los campos son accesibles para el programa fuente ILE COBOL/400.

Parámetro MSGLMT:
Especifica el número máximo de mensajes de un nivel de gravedad de error determinado que pueden producirse para cada unidad de compilación antes de que la compilación se detenga. Tan pronto como la unidad de compilación alcance el número máximo, la compilación se detiene para todo el miembro fuente.

Por ejemplo, si se especifica 3 para el número máximo de mensajes y 20 para el nivel de gravedad de error, la compilación se detendrá si se producen tres o más errores cuyo nivel de gravedad sea 20 o superior. Si ningún mensaje iguala o supera el nivel de gravedad de error determinado, la compilación continuará haciendo caso omiso del número de errores que se produzcan.

número-de-mensajes
Especifica el número máximo de mensajes. Los valores posibles son:

***NOMAX**
La compilación continua hasta que termine normalmente haciendo caso omiso del número de errores que se produzcan.

número-máximo
Especifica el número máximo de mensajes que pueden producirse con el nivel de gravedad de error especificado o con un nivel superior antes de que se detenga la compilación. El rango válido es 0-9999.

gravedad-límite-mensaje
Especifica el nivel de gravedad de error que se utiliza para determinar si la compilación va a detenerse o no. Los valores posibles son:

30 La compilación se detiene si el número de errores cuyo nivel de gravedad sea 30 o superior excede el número máximo de mensajes especificado.

nivel-gravedad-error
Entre un número de uno o dos dígitos, de 0 a 30, para indicar el nivel de gravedad que desea utilizar para determinar si detener o no la compilación. La compilación se detiene si el número de errores con este nivel de

gravedad o superior excede el número máximo de mensajes especificado.

Parámetro DBGVIEW:

Especifica opciones que controlan las vistas del programa fuente o listado generado que se encuentran disponibles para depurar el módulo compilado. Los valores posibles son:

***STMT**

El módulo compilado puede depurarse utilizando nombres simbólicos y números de instrucción.

***SOURCE**

El miembro fuente primario, así como los miembros fuente copiados que se incluyeron mediante instrucciones COPY, tendrán vistas fuente disponibles para depurar el módulo compilado. Estas vistas sólo están disponibles si el miembro fuente primario y los miembros fuente copiados provienen de archivos fuente de bases de datos locales. No modifique ni suprima miembros durante el período de tiempo comprendido entre la compilación y la depuración.

***LIST**

Una vista de listado, equivalente al listado de compilador de archivos en spool, estará disponible para la depuración del módulo compilado. Esta opción aumenta el tamaño del módulo compilado, sin afectar el rendimiento de la ejecución del módulo compilado.

Las vistas de listado se ven afectadas por las opciones de compilador especificadas en el parámetro OPTION cuando se compila el módulo. Por ejemplo, OPTION(*XREF) hace que se incluya una tabla de referencias cruzadas en la vista de listado así como al archivo en spool.

Las vistas de listado pueden generarse haciendo caso omiso de la procedencia de los miembros fuente primarios y de los miembros fuente copiados. Las vistas de listado no se ven afectadas por las modificaciones ni las supresiones que se efectúen en miembros fuente después de la compilación.

***ALL**

Equivale a especificar *STMT, *SOURCE y *LIST a la vez.

***NONE**

El módulo compilado no puede depurarse. Esto reduce el tamaño del programa compilado, pero no afecta el rendimiento de la ejecución. Cuando se especifica esta opción, no es posible tener un vuelco con formato.

Parámetro OPTIMIZE:

Especifica el nivel de optimización del módulo. Los valores posibles son:

***NONE**

No se lleva a cabo optimización alguna en el módulo compilado. Cuando esta opción se utiliza, disminuye el tiempo de compilación. Esta opción permite que se visualicen y modifiquen variables durante la depuración.

***BASIC**

Se lleva a cabo cierta optimización (sólo a nivel de agrupación en bloques local) en el módulo compilado. Esta opción permite visualizar pero no modificar variables de usuario durante la depuración.

***FULL**

Se lleva cabo una optimización completa (a nivel global) en el módulo compilado. Esta optimización aumenta el tiempo que se invierte en la compilación pero también genera el código más eficaz. Esta opción permite visualizar pero no modificar variables de usuario durante la depuración. Es posible que los valores visualizados de las variables no sean sus valores actuales. Es posible que algunas variables no puedan visualizarse.

Nota: Haciendo caso omiso del nivel de optimización escogido, se genera toda las información para permitir la optimización completa. El usuario puede pasar los niveles de optimización del objeto de módulo de *NONE a *FULL utilizando el mandato CHGMOD sin tener que volver a compilar el programa fuente.

Parámetro FLAGSTD:

Especifica las opciones para distintivos FIPS.

(Seleccione la opción *LINENUMBER para asegurarse de que los números de referencia que se utilizan en los mensajes FIPS son exclusivos.) Los valores posibles son:

***NOFIPS**
El programa fuente ILE COBOL/400 no se señala con un distintivo FIPS.

***MINIMUM**
Distintivo FIPS para subconjunto mínimo y superior.

***INTERMEDIATE**
Distintivo FIPS para subconjunto intermedio y superior.

***HIGH**
Distintivo FIPS para subconjunto superior.

***NOOBSOLETE**
Los elementos de lenguaje obsoletos no se señalan con distintivos.

***OBSOLETE**
Los elementos de lenguaje obsoletos se señalan con distintivos.

Parámetro EXTDSPOPT:

Especifica las opciones que se pueden utilizar para las instrucciones ACCEPT y DISPLAY ampliadas para E/S de estación de trabajo. Los valores posibles son:

***DFRWRT**
Las instrucciones DISPLAY ampliadas se guardan en un almacenamiento intermedio hasta que se encuentra una instrucción ACCEPT ampliada o hasta que el almacenamiento intermedio se llena.

El contenido del almacenamiento intermedio se escribe en la pantalla cuando se encuentra la instrucción ACCEPT ampliada o cuando el almacenamiento intermedio está lleno.

***NODFRWRT**
Cada una de las instrucciones DISPLAY ampliadas se llevan a cabo a medida que se encuentra.

***UNDSPCHR**
Las instrucciones ACCEPT y DISPLAY ampliadas manejan caracteres visualizables y no visualizables.

***NOUNDSPCHR**
Las instruccioness DISPLAY y ACCEPT ampliadas sólo manejan caracteres visualizables.

Aunque debe utilizar esta opción para estaciones de pantalla conectadas a controladores 3174 y 3274 remotos, también puede utilizarla para estaciones de trabajo locales. Si no utiliza esta opción, los datos deben contener caracteres visualizables exclusivamente. Si los datos contienen valores inferiores al hexadecimal 20, el resultado puede ser imprevisible, desde errores de formato de pantalla inesperados hasta errores graves.

***ACCUPDALL**
Todos los tipos de datos se visualizan previamente en las instrucciones ACCEPT ampliadas haciendo caso omiso de la existencia de la expresión UPDATE.

***ACCUPDNE**
Sólo los datos editados numéricos se visualizan previamente en las instrucciones ACCEPT ampliados que no contienen la expresión UPDATE.

Parámetro FLAG:
Especifica el nivel de gravedad mínimo de mensajes que aparecerán en el listado del compilador. Los valores posibles son:

0 Todos los mensajes aparecerán en el listado del compilador.

nivel-gravedad
Entre un número de uno o dos dígitos que especifique el nivel de gravedad mínimo de los mensajes que desea que aparezcan en el listado del compilador. Los mensajes cuyo nivel de gravedad sea igual o superior que el especificado aparecerán en el listado del compilador.

Parámetro REPLACE:
Especifica si se crea un módulo nuevo cuando ya existe un módulo con el mismo nombre en la biblioteca especificada o presupuesta. Los valores posibles son:

***YES**
Se crea un módulo nuevo que sustituye cualquier módulo existente con el mismo nombre en la biblioteca especificada o presupuesta. El módulo existente con el

mismo nombre que se encuentra en la biblioteca especificada o presupuesta se traslada a la biblioteca QRPLOBJ.

***NO**
No se crea un módulo nuevo si en la biblioteca especificada o presupuesta ya existe un módulo con el mismo nombre. El módulo existente no se sustituye, aparece un mensaje en pantalla y la compilación se detiene.

Parámetro AUT:
Especifica la autorización que se da a los usuarios que no tienen una autorización específica sobre el objeto de módulo, los usuarios que no se encuentran en la lista de autorizaciones o aquéllos cuyo grupo no tiene autorización específica sobre el objeto de módulo. Se puede modificar la autorización para todos lo usuarios, o para determinados usuarios después de crear el objeto de módulo utilizando los mandatos GRTOBJAUT (Otorgar autorización sobre objeto) o RVKOBJAUT (Revocar autorización sobre objeto).

Los valores posibles son:

***LIBCRTAUT**
La autorización pública para el objeto se toma de la palabra clave CRTAUT de la biblioteca destino (la biblioteca que va a contener el objeto de módulo creado). Este valor se determina cuando se crea el objeto de módulo. Si el valor CRTAUT para la biblioteca se modifica después de haber creado el objeto de módulo, el nuevo valor NO afecta a ninguno de los objetos existentes.

***ALL**
Proporciona autorización para realizar todas las operaciones en el objeto de módulo excepto aquellas limitadas al propietario o controladas por la autoridad de gestión de la lista de autorizaciones.El usuario puede controlar la existencia del objeto de módulo, especificar su seguridad, modificarlo y realizar funciones básicas sobre éste, pero no puede transferir su propiedad.

***CHANGE**
Proporciona autorización sobre todos los datos y autorización para realizar todas las operaciones en el objeto de módulo excepto aquellas limitadas al propietario o controladas por la autorización sobre objeto y la autorización de gestión de objetos. El usuario puede modificar el objeto y realizar funciones básicas sobre éste.

***USE**
Proporciona autorización operativa sobre objetos y autorización de lectura; autorización para realizar operaciones básicas en el objeto de módulo. El usuario puede realizar operaciones básicas sobre el objeto pero no puede modificarlo.

***EXCLUDE**
El usuario no puede acceder al objeto de módulo.

nombre-lista-autorización
El nombre de una lista de autorizaciones de usuario y autorizaciones a las que se añade el módulo.El objeto de módulo se protege con esta lista de autorizaciones y la autorización pública para el objeto de módulo se establece en *AUTL. La lista de autorizaciones debe existir en el sistema cuando se emite el mandato CRTCBLMOD. Utilice el mandato Crear lista de autorizaciones (CRTAUTL) para crear una lista de autorizaciones propia.

Parámetro LINKLIT:
Especifica el tipo de enlace para el destino externo de CALL/CANCEL 'literal' y el destino de SET ENTRY. Puede alterar temporalmente esta opción para las listas de destinos externos de CALL/CANCEL 'literal' y de SET ENTRY especificando la instrucción siguiente en el párrafo SPECIAL-NAMES:

```
LINKAGE TYPE IS nombre-implementador FOR lista-destinos.
```

Los valores posibles para LINKLIT son:

***PGM**
El destino para CALL/CANCEL o SET ENTRY es un objeto de programa.

***PRC**
El destino para CALL/CANCEL o SET ENTRY es un procedimiento ILE.

Parámetro TGTRLS:
Especifica el entorno del Release de destino para el módulo que se está creando.

En los ejemplos que aparecen a continuación, para especificar el release se utiliza el formato VxRxMx, en el que Vx es la versión, Rx el release y Mx el nivel de modificación. Por ejemplo, V2R3M0 es la versión 2, release 3, nivel de modificación 0. Los valores posibles son los siguientes:

***CURRENT**
El objeto va a utilizarse en el release del sistema operativo que se está ejecutando en el sistema actualmente. Por ejemplo, si en el sistema se está ejecutando V3R7M0, *CURRENT significa que piensa utilizar el objeto en un sistema en el que se ha instalado V3R7M0. El objeto también puede utilizarse en un sistema que tenga instalado un release posterior del sistema operativo.

Nota: Si en el sistema se está ejecutando V2R3M5 y el objeto debe utilizarse en un sistema que tiene instalado V2R3M0, especifique TGTRLS(V2R3M0) en lugar de TGTRLS(*CURRENT).

***PRV**
El objeto va a utilizarse en el release anterior del sistema operativo, con un nivel de modificación de 0. Por ejemplo, si se está ejecutando V3R7M0 en el sistema del usuario, *PRV significa que intenta utilizar el objeto en un sistema que tiene instalado V3R6M0. También puede utilizar el objeto en un sistema que tenga instalado un release posterior del sistema operativo.

nivel-release
Especifique el nivel de release del entorno destino en el formato VxRxMx. El objeto puede utilizarse en un sistema que tenga instalado el release especificado o un release posterior del sistema operativo.

Los valores válidos dependen de la versión, release y nivel de modificación actuales y cambian con cada release. Si especifica un nivel-release anterior al primer nivel de release al que este mandato da soporte, se envía un mensaje de error que indica el primer release que se soporta.

Parámetro SRTSEQ:
Especifica la secuencia de ordenación que se utiliza cuando se asocia NLSSORT a un nombre-alfabeto en la cláusula ALPHABET. El parámetro SRTSEQ se utiliza junto con el parámetro LANGID para determinar qué tabla de secuencia de ordenación definida por el usuario o definida por el sistema va a utilizar el módulo. Los valores posibles son:

***HEX**
No se utilizará ninguna tabla de secuencia de ordenación; para determinar la secuencia de ordenación se utilizarán los valores hexadecimales de los caracteres.

***JOB**
La secuencia de ordenación se resolverá y asociará con el módulo durante la compilación utilizando la secuencia de ordenación del trabajo de compilación. La tabla de secuencia de ordenación del trabajo de compilación debe existir en el sistema en el momento de la compilación. Si en el momento de la ejecución el CCSID del trabajo en ejecución difiere del CCSID del trabajo en el momento de la compilación, la tabla de secuencia de ordenación que se ha cargado en el momento de la compilación se convierte para que coincida con el CCSID del trabajo en ejecución.

***JOBRUN**
La secuencia de ordenación se resolverá y asociará con el módulo en el momento de la ejecución. Este valor permite que el módulo se compile una vez y se utilice con diferentes secuencias de ordenación en el momento de la ejecución.

***LANGIDUNQ**
Especifica que la tabla de secuencia de ordenación que se utiliza debe contener un peso único para cada carácter de la página de códigos. La tabla de secuencia de ordenación utilizada será la única tabla con significación asociada al idioma especificado en el parámetro LANGID.

***LANGIDSHR**
Especifica que la tabla de secuencia de ordenación que se utiliza puede contener el mismo peso para varios caracteres de la página de códigos. La tabla de secuencia de ordenación utilizada será la

tabla con significación compartida asociada al idioma especificado en el parámetro LANGID.

nombre-tabla
Entre el nombre de la tabla de secuencia de ordenación que se va a utilizar. La tabla contiene pesos para todos los caracteres de una página de códigos determinada. Un peso se asocia al carácter que se define en el elemento de código. Cuando se utiliza un nombre de tabla de secuencia de ordenación, se puede especificar la biblioteca donde reside el objeto. Los valores válidos para la biblioteca son:

***LIBL**
La biblioteca en la que se ubica la tabla de secuencia de ordenación se busca en la lista de bibliotecas.

***CURLIB**
Se utiliza la biblioteca actual. Si no se ha asignado una biblioteca como biblioteca actual, se utiliza QGPL.

nombre-biblioteca
Entre el nombre de la biblioteca donde se encuentra la tabla de secuencia de ordenación.

Parámetro LANGID:
Especifica el identificador de idioma que se utiliza junto con la secuencia de ordenación. El parámetro LANGID se utiliza sólo cuando el valor SRTSEQ en vigor es *LANGIDUNQ o *LANGIDSHR. Los valores posibles son:

***JOBRUN**
El identificador de idioma del módulo se resolverá en el momento de la ejecución. Este valor permite que el módulo se compile una vez y se utilice con diferentes identificadores de idioma durante la ejecución.

***JOB**
El identificador de idioma del módulo se resolverá en el momento de la compilación utilizando el identificador de idioma del trabajo de compilación.

nombre-identificador-idioma
Entre un identificador de idioma de 3 caracteres válido.

Parámetro ENBPFRCOL:
Especifica si debe generarse el código de medida de rendimiento en el módulo o programa. La herramienta de rendimiento del sistema puede utilizar los datos recogidos para crear un perfil del rendimiento de una aplicación. Al generar un código de medida de rendimiento en un módulo o programa compilado, los objetos pasarán a ser ligeramente mayores y es posible que ello afecte el rendimiento.

***PEP**
Sólo se reúnen estadísticas de rendimiento en la entrada y salida del procedimiento de entrada de programa. Seleccione este valor si desea tener infomación general sobre el rendimiento de una aplicación. Este soporte es equivalente al soporte que la herramienta TPST proporcionaba anteriormente. Se trata del valor por omisión.

***ENTRYEXIT**
Se reúnen estadísticas sobre el rendimiento en la entrada y salida de todos los procedimientos del programa. Incluye la rutina del PEP del programa.

Esta opción le resultará de utilidad si desea capturar información sobre todas las rutinas. Utilice esta opción si sabe que todos los programas a los que la aplicación ha llamado se han compilado con la opción *PEP, *ENTRYEXIT o *FULL. En caso contrario, si la aplicación llama a otros programas en los que no se ha habilitado la medida de rendimiento, la herramienta de rendimiento cargará a la aplicación la utilización de recursos de los programas llamados. Esto dificultaría la determinación del lugar en que los recursos se utilizan realmente.

***FULL**
Se reúnen estadísticas sobre el rendimiento en la entrada y salida de todos los procedimientos. Además, también se reúnen estadísticas antes y después de cada llamada a un procedimiento externo.

Utilice esta opción si piensa que la aplicación llamará a otros programas que no se hayan compilado con *PEP, *ENTRYEXIT o *FULL. Esta opción permite que las herramientas de ren-

dimiento distingan entre los recursos utilizados por la aplicación y los utilizados por los programas a los que llama (incluso aunque en estos programas no se haya habilitado la medida de rendimiento). Es la opción más cara pero le permite el análisis selectivo de varios programas dentro de una aplicación.

Ejemplo de compilación de un programa fuente en un objeto de módulo

Este ejemplo muestra cómo crear un objeto de módulo ILE COBOL/400 utilizando el mandato CRTCBLMOD.

1. Para crear un objeto de módulo, teclee:

   ```
   CRTCBLMOD MODULE(MYLIB/XMPLE1)
   SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(MYLIB/XMPLE1)
   OUTPUT(*PRINT)
   TEXT('Programa ILE COBOL/400')
   CVTOPT(*FLOAT)
   ```

   El mandato CRTCBLMOD crea el módulo XMPLE1 en MYLIB, la misma biblioteca que contiene el fuente. La opción de salida OUTPUT(*PRINT) especifica un listado del compilador. La opción de conversión CVTOPT(*FLOAT) especifica que los tipos de datos de coma flotante se transfieren al programa con los nombres de DDS y un USAGE de COMP-1 (precisión simple) o COMP-2 (precisión doble).

2. Teclee uno de los siguientes mandatos CL para ver el listado del compilador.

   Nota: Para poder ver un listado del compilador debe tener autorización para utilizar los mandatos que se listan a continuación.

   - `DSPJOB` y a continuación seleccione la opción 4 (*Visualizar archivos en spool*)
   - `WRKJOB`
   - `WRKOUTQ nombre-cola`
   - `WRKSPLF`

Especificación de un release destino diferente

Se puede compilar un programa ILE COBOL/400 en un sistema AS/400 utilizando el release actual del sistema operativo OS/400.

Nota: El compilador ILE COBOL/400 y el compilador OPM COBOL/400 son opciones de producto individuales separadas. La información que se incluye en este apartado se aplica únicamente al release actual del compilador ILE COBOL/400.

El parámetro Release destino (TGTRLS) de los mandatos CRTCBLMOD y CRTBNDCBL permite especificar el nivel de release en el que se planifica utilizar el objeto de módulo. El parámetro TGTRLS cuenta con tres valores posibles: *CURRENT, *PRV y *nivel-release*.

- Especifique *CURRENT si el objeto de módulo va a utilizarse en el release del sistema operativo que se está ejecutando actualmente en el sistema. Por ejemplo, si en el sistema se está ejecutando V3R1M0, *CURRENT significa que piensa utilizar el programa en un sistema en el que se ha instalado V3R1M0. Este es el valor por omisión.

- Si especifica *PRV el objeto va a utilizarse en el release anterior del sistema operativo, con un nivel de modificación de 0. Por ejemplo, si se está ejecutando V3R7M0 en el sistema del usuario, *PRV significa que intenta utilizar el objeto en un sistema que tiene instalado V3R6M0. También puede utilizar el objeto en un sistema que tenga instalado un release posterior del sistema operativo.
- *nivel-release* permite especificar el nivel de release en el que se piensa utilizar el objeto de módulo. Los valores que se pueden entrar para este parámetro dependen de la versión, release y nivel de modificación actuales y cambian con cada release nuevo.

  Especifique el nivel de release del entorno destino en el formato VxRxMx. El objeto puede utilizarse en un sistema que tenga instalado el release especificado o un release posterior del sistema operativo.

  Por ejemplo, si especifica V3R2M0, el objeto puede utilizarse en un sistema V3R2M0.

Para obtener más información sobre el parámetro TGTRLS, consulte la página 42.

Debería tener en cuenta las limitaciones siguientes :

- Se puede restaurar un programa objeto al release actual o a un release posterior. No se puede restaurar un programa objeto en un release anterior distinto al permitido por el nivel-release de TGTRLS.
- No debería haber ninguna biblioteca del producto en la parte del sistema de la lista de bibliotecas.

Especificación de la secuencia de ordenación de idioma en CRTCBLMOD

En el momento en que se compila el programa fuente ILE COBOL/400, se puede especificar de forma explícita el orden de clasificación que el programa utilizará cuando se ejecute, o se puede especificar cómo se va a determinar el orden de clasificación cuando se ejecute el programa.

Para especificar el orden de clasificación, primero se define un *nombre-alfabeto* en el párrafo SPECIAL-NAMES utilizando la cláusula ALPHABET y se asocia ese *nombre-alfabeto* al nombre de implementador NLSSORT. A continuación, haga referencia a este *nombre-alfabeto* en la cláusula PROGRAM COLLATING SEQUENCE de la ENVIRONMENT DIVISION, o en la expresión COLLATING SEQUENCE de las instrucciones SORT/MERGE, para indicar que el *nombre-alfabeto* especificado determinará el orden de clasificación que va a utilizarse.

Se especifica el orden de clasificación real utilizado con las opciones de los parámetros SRTSEQ y LANGID de los mandatos CRTCBLMOD y CRTBNDCBL.Por ejemplo, si se especifica SRTSEQ(*JOBRUN) y LANGID(*JOBRUN), el orden de clasificación del programa se resolverá en el momento de la ejecución.Este valor permite que el programa fuente se compile una vez y se utilice con diferentes órdenes de clasificación en el momento de la ejecución. Las opciones de la instrucción PROCESS asociadas a SRTSEQ y LANGID también pueden utilizarse para especificar el orden de clasificación (consulte el apartado “Utilización de la instrucción PROCESS para especificar opciones de compilador” en la página 47).

Si el programa fuente no tiene NLSSORT asociado a un *nombre-alfabeto* en su cláusula ALPHABET, o en la cláusula ALPHABET se especifica NLSSORT pero no se hace referencia al *nombre-alfabeto* en ninguna cláusula PROGRAM COLLATING SEQUENCE ni en ninguna expresión COLLATING SEQUENCE de las instrucciones SORT/MERGE, la secuencia de ordenación identificada mediante los parámetros SRTSEQ y LANGID no se utiliza.

El *nombre-alfabeto* asociado a NLSSORT no puede utilizarse para determinar el juego de códigos de caracteres, como en la cláusula CODE-SET de la entrada Descripción de archivo (FD).El *nombre-alfabeto* que se utiliza para determinar el juego de códigos de caracteres debe identificarse en una cláusula ALPHABET separada.

Consulte el manual *ILE COBOL/400 Reference* para obtener una descripción completa de la cláusula ALPHABET, la cláusula PROGRAM COLLATING SEQUENCE y las instrucciones SORT/MERGE. Consulte el apartado “Parámetros del mandato CRTCBLMOD” en la página 33 para obtener una descripción de los parámetros SRTSEQ y LANGID.

Utilización de la instrucción PROCESS para especificar opciones de compilador

La instrucción PROCESS es una parte opcional del programa fuente ILE COBOL/400. Se puede utilizar la instrucción PROCESS para especificar opciones que normalmente especificaría en el momento de la compilación.

Las opciones especificadas en la instrucción PROCESS **alteran temporalmente** las opciones correspondientes especificadas en los mandatos CL CRTCBLMOD o CRTBNDCBL.

Se aplican las siguientes normas:

- La instrucción debe colocarse antes que la primera instrucción fuente del programa fuente ILE COBOL/400 que empiece una unidad de compilación nueva, justo delante de la cabecera IDENTIFICATION DIVISION.
- La instrucción empieza con la palabra PROCESS. Las opciones pueden aparecer en más de una línea; sin embargo, la palabra PROCESS sólo puede aparecer en la primera línea.
- La palabra PROCESS y todas las opciones deben aparecer en las posiciones 8 a 72. La posición 7 debe quedar en blanco. Las restantes posiciones pueden utilizarse igual que en las instrucciones fuente de ILE COBOL/400: las posiciones 1 a 6 para los números de secuencia y las posiciones 73 a 80 para la identificación.
- Las opciones deben separarse con blancos y/o comas.
- Las opciones pueden aparecer en cualquier orden. Si se especifican opciones en conflicto, por ejemplo, XREF y NOXREF, tendrá prioridad la última opción que se encuentre.
- Si la palabra clave de opción es correcta y la subopción tiene un error, se asumirá la subopción por omisión.

No todos los parámetros de los mandatos CRTCBLMOD y CRTBNDCBL tienen una opción correspondiente en la instrucción PROCESS. Consulte las tablas

siguientes que indican las opciones de la instrucción PROCESS permitidas y las opciones y parámetros de los mandatos CRTCBLMOD o CRTBNDCBL equivalentes. Los valores por omisión aparecen subrayados. Las descripciones de las opciones de la instrucción PROCESS corresponden a las descripciones de las opciones y parámetros que se encuentran en el apartado “Parámetros del mandato CRTCBLMOD” en la página 33.

Opciones de la instrucción PROCESS	**CRTCBLMOD/CRTBNDCBL**
	Opciones del parámetro OUTPUT
OUTPUT NOOUTPUT	*PRINT *NONE

Opción de la instrucción PROCESS	**CRTCBLMOD/CRTBNDCBL**
	Opción del parámetro GENLVL
GENLVL(nn)	nn

Opciones de la instrucción PROCESS	**CRTCBLMOD/CRTBNDCBL**
	Opciones del parámetro OPTION
SOURCE SRC NOSOURCE NOSRC	*SOURCE *SRC *NOSOURCE *NOSRC
NOXREF XREF	*NOXREF *XREF
GEN NOGEN	*GEN *NOGEN
NOSEQUENCE SEQUENCE	*NOSEQUENCE *SEQUENCE
NOVBSUM VBSUM	*NOVBSUM *VBSUM
NONUMBER NUMBER LINENUMBER	*NONUMBER *NUMBER *LINENUMBER
NOMAP MAP	*NOMAP *MAP
NOOPTIONS OPTIONS	*NOOPTIONS *OPTIONS
QUOTE APOST	*QUOTE *APOST
NOSECLVL SECLVL	*NOSECLVL *SECLVL
PRTCORR NOPRTCORR	*PRTCORR *NOPRTCORR

Opciones de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro OPTION
MONOPRC NOMONOPRC	*MONOPRC *NOMONOPRC
RANGE NORANGE	*RANGE *NORANGE
NOUNREF UNREF	*NOUNREF *UNREF
NOSYNC SYNC	*NOSYNC *SYNC
NOCRTF CRTF	*NOCRTF *CRTF
NODUPKEYCHK DUPKEYCHK	*NODUPKEYCHK *DUPKEYCHK
NOINZDLT INZDLT	*NOINZDLT *INZDLT
NOBLK BLK	*NOBLK *BLK
STDINZ NOSTDINZ	*STDINZ *NOSTDINZ
NODDSFILLER DDSFILLER	*NODDSFILLER *DDSFILLER
STDTRUNC NOSTDTRUNC	*STDTRUNC *NOSTDTRUNC
CHGPOSSGN NOCHGPOSSGN	*CHGPOSSGN *NOCHGPOSSGN

Opciones de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro CVTOPT
NOVARCHAR VARCHAR	*NOVARCHAR *VARCHAR
NODATETIME DATETIME	*NODATETIME *DATETIME
NOCVTPICXGRAPHIC CVTPICXGRAPHIC CVTPICGGRAPHIC NOCVTPICGGRAPHIC	*NOPICXGRAPHIC *PICXGRAPHIC *PICGGRAPHIC *NOPICGGRAPHIC
NOFLOAT FLOAT	*NOFLOAT *FLOAT

Opciones de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro OPTIMIZE
NOOPTIMIZE BASICOPT FULLOPT	*NONE *BASIC *FULL

Opciones de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro FLAGSTD
NOFIPS MINIMUM INTERMEDIATE HIGH	*NOFIPS *MINIMUM *INTERMEDIATE *HIGH
NOOBSOLETE OBSOLETE	*NOOBSOLETE *OBSOLETE

Opciones de la instrucción PROCESS EXTDSPOPT(*a b c*)	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro EXTDSPOPT
DFRWRT NODFRWRT	*DFRWRT *NODFRWRT
UNDSPCHR NOUNDSPCHR	*UNDSPCHR *NOUNDSPCHR
ACCUPDALL ACCUPDNE	*ACCUPDALL *ACCUPDNE

Opción de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opción del parámetro FLAG
FLAG(nn)	nn

Opciones de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
	Opción del parámetro LINKLIT
LINKPGM LINKPRC	*PGM *PRC

Opciones de la instrucción PROCESS SRTSEQ(*a*)	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro SRTSEQ
HEX	*HEX
JOB	*JOB
JOBRUN	*JOBRUN
LANGIDUNQ	*LANGIDUNQ
LANGIDSHR	*LANGIDSHR
"LIBL/nombre-tabla-sec-ord"	*LIBL/nombre-tabla-sec-ord
"CURLIB/nombre-tabla-sec-ord"	*CURLIB/nombre-tabla-sec-ord
"nombre-biblioteca/nombre-tabla-sec-ord"	nombre-biblioteca/nombre-tabla-sec-ord
"nombre-tabla-sec-ord"	nombre-tabla-sec-ord

Opciones de la instrucción PROCESS LANGID(*a*)	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro LANGID
JOBRUN	*JOBRUN
JOB	*JOB
"nombre-identificador-idioma"	nombre-identificador-idioma

Opciones de la instrucción PROCESS ENBPFRCOL(*a*)	CRTCBLMOD/CRTBNDCBL
	Opciones del parámetro ENBPFRCOL
PEP	*PEP
ENTRYEXIT	*ENTRYEXIT
FULL	*FULL

Opción de la instrucción PROCESS	CRTCBLMOD/CRTBNDCBL
NOGRAPHIC	no aplicable
GRAPHIC	

La opción GRAPHIC de la instrucción PROCESS está disponible para el proceso de caracteres DBCS en literales mixtos. Los **literales mixtos** son literales que combinan caracteres SBCS y caracteres DBCS. Si se especifica la opción GRAPHIC, los literales mixtos se manejarán con la suposición de que hex 0E y hex 0F son caracteres de desplazamiento a teclado estándar y a teclado ideográfico respectivamente y que incluyen los caracteres DBCS en el literal mixto. Los caracteres de desplazamiento a teclado estándar y a teclado ideográfico ocupan 1 byte cada uno. Si se especifica o presupone NOGRAPHIC, el compilador ILE COBOL/400 tratará los literales no numéricos que contienen hex 0E y hex 0F como si sólo contuvieran caracteres SBCS. Hex 0E y hex 0F no se tratan como caracteres de desplazamiento a teclado estándar ni a teclado ideográfico, sino que se considerará que forman parte de una serie de caracteres SBCS. Consulte el Apéndice D, “Soporte a idiomas internacionales con juegos de caracteres de doble byte” en la página 491 para obtener más información sobre el soporte de DBCS.

La opción EXTDSPOPT de la instrucción PROCESS debería codificarse con las opciones asociadas entre paréntesis semejante a la sintaxis de FLAG(nn). Puede especificar más de una opción entre paréntesis para la opción EXTDSPOPT. Por ejemplo, para especificar DFRWRT y UNDSPCHR, teclee

```
EXTDSPOPT(DFRWRT UNDSPCHR)
```

También es válido especificar EXTDSPOPT o EXTDSPOPT( ).

Cuando sólo se especifica EXTDSPOPT en la instrucción PROCESS, todos los valores por omisión para las opciones adicionales quedan en vigor.

Si se especifica EXTDSPOPT( ), no tiene efecto en el programa.

Las opciones SRTSEQ, LANGID i ENBPFRCOL de la instrucción PROCESS deberían codificarse con las opciones adecuadas entre paréntesis, de forma similar a la sintaxis de FLAG(nn).

Compilación de varios programas fuente

La instrucción PROCESS puede colocarse al principio de cada unidad de compilación siguiendo la secuencia de los programas fuente ILE COBOL/400 del miembro fuente de entrada. Al compilar varios programas fuente ILE COBOL/400, los resultados fusionados de todas las opciones especificadas en el mandato CRTCBLMOD o CRTBNDCBL, más todas las opciones por omisión y las opciones especificadas en la última instrucción PROCESS anterior al programa fuente ILE COBOL/400 estarán vigentes para la compilación de este programa fuente ILE COBOL/400. Toda la salida del compilador se dirige a los destinos especificados por los mandatos CRTCBLMOD o CRTBNDCBL.

Todos los objetos de módulo u objetos de programa se almacenan en la biblioteca especificada en el parámetro MODULE o en el parámetro PGM. Si se especifica *nombre-módulo* o *nombre-programa* para el parámetro MODULE o para el parámetro PGM, el primer objeto de módulo u objeto de programa correspondiente al primer programa fuente ILE COBOL/400 de la secuencia de programas fuente ILE COBOL/400 utiliza ese nombre, y todos los objetos de módulo u objetos de programa correspondientes al resto de programas fuente ILE COBOL/400 del mismo miembro fuente de entrada utilizan el nombre especificado en el párrafo PROGRAM-ID del programa fuente ILE COBOL/400.

Utilización de COPY en la instrucción PROCESS

Las instrucciones COPY pueden utilizarse en un programa fuente siempre que puedan utilizarse series de caracteres o separadores. Cada una de las instrucciones COPY debe ir precedida de un espacio y seguida de un punto y un espacio. Para obtener más información sobre la instrucción COPY, consulte el apartado “Instrucción COPY” del manual *ILE COBOL/400 Reference*.

La instrucción COPY Formato 1 puede utilizarse en la instrucción PROCESS para recuperar opciones de compilador previamente almacenadas en una biblioteca fuente e incluirlas en la instrucción PROCESS. COPY puede utilizarse para incluir opciones que alteren temporalmente aquéllas que el compilador especifica como valores por omisión. Con la instrucción COPY pueden recuperarse cualesquiera opciones de la instrucción PROCESS.

Las opciones de compilador pueden tanto preceder como ir a continuación de la instrucción COPY dentro de una instrucción PROCESS. La opción que se

encuentre en último lugar altera temporalmente las anteriores apariciones de esa opción.

El ejemplo siguiente muestra la utilización de la instrucción COPY en la instrucción PROCESS. Observe también que en este caso, NOMAP altera temporalmente la opción correspondiente del miembro de biblioteca:

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/COPYPROC         AS400SYS   96/07/04 11:48:02         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FECHCAM


          000010 PROCESS XREF
          000020 COPY PROCDFLT.
         +000010         MAP, SOURCE, APOST                                          PROCDFLT
          000030         NOMAP, FLAG(20)
    1     000040 IDENTIFICATION DIVISION.
    2     000050 PROGRAM-ID. COPYPROC.
    3     000060 ENVIRONMENT DIVISION.
    4     000070 CONFIGURATION SECTION.
    5     000080   SOURCE-COMPUTER. IBM-AS400.
    6     000090   OBJECT-COMPUTER. IBM-AS400.
    7     000100 PROCEDURE DIVISION.
          000110 MAINLINE.
    8     000120     DISPLAY "HOLA MUNDO".
    9     000130     STOP RUN.

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 8. Utilización de COPY en la instrucción PROCESS

Comprensión de la salida del compilador

La salida del compilador puede incluir:

- Un resumen de opciones de mandatos
- Un listado de opciones: un listado de las opciones en vigor para la compilación. Utilice OPTION(*OPTIONS).
- Un listado fuente: un listado de las instrucciones que se encuentran en el programa fuente. Utilice OPTION(*SOURCE).
- Un listado de utilización de verbos: un listado de los verbos COBOL y las veces que se utiliza cada verbo. Utilice OPTION(*VBSUM).
- Un mapa de Data Division: un glosario de información generada por el compilador sobre los datos. Utilice OPTION(*MAP).
- Mensajes FIPS: una lista de mensajes para un subconjunto FIPS COBOL, para cualquiera de los módulos opcionales, para todos los elementos del lenguaje obsoletos o para una combinación de un subconjunto FIPS COBOL, módulos opcionales y todos los elementos obsoletos. Consulte la información que aparece en el apartado “Parámetro FLAGSTD” en la página 40 para conocer opciones específicas disponibles para distintivos FIPS.
- Listado de referencias cruzadas. Utilice OPTION(*XREF).
- Listado de errores intercalados. Utilice OPTION(*IMBEDERR).
- Mensajes del compilador (incluyendo estadísticas de diagnóstico).

- Estadísticas de compilación.
- Objeto(s) de módulo. Utilice el mandato CRTCBLMOD.
- Objeto(s) de programa. Utilice el mandato CRTBNDCBL.

La presencia o ausencia de alguno de estos tipos de salida de compilador depende de las opciones especificadas en la instrucción PROCESS o mediante el mandato CRTCBLMOD o el mandato CRTBNDCBL. EL nivel de mensajes de diagnóstico impreso depende de la opción FLAG. La opción DBGVIEW indica la clase de datos de depuración que se encuentran en el objeto de programa u objeto de módulo generado.

Especificación del formato del listado

Una barra inclinada (/) en el área de indicador (columna 7) de una línea da como resultado la expulsión de página del listado de programa fuente. También se puede entrar texto de comentario después de la barra inclinada (/) en esta línea. La línea de comentario de la barra inclinada (/) se imprime en la primera línea de la siguiente página.

Si se especifica la instrucción EJECT en el programa, la siguiente instrucción fuente se imprime en la parte superior de la siguiente página del listado del compilador. Esta instrucción pude escribirse en cualquier parte del Área A o del Área B y debe ser la única instrucción de la línea.

La instrucción SKIP1/2/3 provoca que se inserten líneas en blanco en el listado del compilador. Una instrucción SKIP1/2/3 puede escribirse en cualquier parte del Área A o del Área B. Debe ser la única instrucción de la línea.

- SKIP1 inserta una línea en blanco (doble espacio).
- SKIP2 inserta dos líneas en blanco (triple espacio).
- SKIP3 inserta tres líneas en blanco (cuatro espacios).

Cada una de las instrucciones SKIP anteriores provoca una única inserción de una, dos o tres líneas.

Una instrucción TITLE coloca un título en cada una de las páginas que se indique.

Se pueden listar o suprimir de forma selectiva las instrucciones fuente ILE COBOL/400 utilizando las instrucciones *CONTROL, *CBL o COPY:

- *CONTROL NOSOURCE y *CBL NOSOURCE suprimen el listado de instrucciones fuente.
- *CONTROL SOURCE y *CBL SOURCE continúan el listado de instrucciones fuente.
- Una instrucción COPY que incluye una expresión SUPPRESS suprime el listado de instrucciones copiadas. Mientras esté en vigor, esta instrucción altera temporalmente cualquier instrucción *CONTROL o *CBL. Si el miembro copiado contiene instrucciones *CONTROL o *CBL, la última se ejecuta cuando el miembro COPY se ha procesado.

Consulte el manual *ILE COBOL/400 Reference* para obtener información adicional sobre las instrucciones EJECT, SKIP1/2/3, *CONTROL, *CBL, COPY y TITLE.

Caracteres de separación de hora

El parámetro TIMSEP de los mandatos relacionados con el trabajo (como CHGJOB) ahora especifica el carácter de separación de hora que se utiliza en las indicaciones de hora que aparecen en el listado el compilador. Si no se especifica un valor TIMSEP, se utiliza por omisión el valor del sistema QTIMSEP.

Examen del listado del compilador utilizando el SEU

El Programa de utilidad para entrada del fuente (SEU) permite examinar un listado de compilador de una cola de salida. Puede volver a ver el resultado de una compilación anterior mientras realiza las modificaciones necesarias en el código fuente.

Mientras se examina el listado de compilador, se pueden buscar los errores y corregir las instruccioness fuentes que tienen errores. Para buscar errores, teclee `F *ERR` en la línea de mandatos SEU.

Para obtener más información sobre cómo examinar un listado de compilador, consulte el manual *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)*.

Programa de ejemplo y listados

Los siguientes listados de ejemplo muestran las opciones de compilador y listado fuente que se generan para el programa de ejemplo. En el texto siguiente se hace referencia a las figuras. Estas referencias se indican mediante la impresión invertida de las letras sobre fondo oscuro, por ejemplo (Z). Las letras del texto en impresión invertida corresponden a las letras que se encuentran en las figuras.

Resumen de mandatos

Este resumen, resultado de la compilación, lista todas las opciones especificadas en los mandatos CRTCBLMOD y CRTBNDCBL. Consulte el apartado “Utilización del mandato Crear módulo COBOL (CRTCBLMOD) ” en la página 30 para obtener más información sobre las opciones definidas por el usuario.

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27         Página     1

Mandato  . . . . . . . . . . . . . . . :   CRTCBLMOD

Valores reales:

  Módulo . . . . . . . . . . . . . . . :   SAMPLE
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Archivo fuente . . . . . . . . . . . :   QCBLLESRC
    Biblioteca . . . . . . . . . . . . :     TESTLIB
    CCSID  . . . . . . . . . . . . . . :   65535
  Miembro fuente . . . . . . . . . . . :   SAMPLE         94/08/22 13:43:33
  Descripción de texto. . . . . . . . . :  Programa muestra para generar listado

Opciones de mandato:

  Módulo . . . . . . . . . . . . . . . :   SAMPLE
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Archivo fuente . . . . . . . . . . . :   QCBLLESRC
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Miembro fuente . . . . . . . . . . . :   *MODULE
  Salida . . . . . . . . . . . . . . . :   *PRINT
  Nivel de gravedad de generación. . . :   30
  Texto 'descripción'  . . . . . . . . :   *SRCMBRTXT
  Opciones de compilador . . . . . . . :   *IMBEDERR
  Opciones de conversión . . . . . . . :   *NONE
  Límite de mensajes:
    Número de mensajes . . . . . . . . :   *NOMAX
    Gravedad límite de mensajes. . . . :   30
  Vista de depuración. . . . . . . . . :   *STMT
  Nivel de optimización. . . . . . . . :   *NONE
  Distintivo FIPS  . . . . . . . . . . :   *NOFIPS *NOOBSOLETE
  Opciones de pantalla ampliada. . . . :   *NONE
  Gravedad de distintivos. . . . . . . :   0
  Sustituir módulo . . . . . . . . . . :   *YES
  Autorización . . . . . . . . . . . . :   *LIBCRTAUT
  Literal de enlace. . . . . . . . . . :   *PGM
  Release destino. . . . . . . . . . . :   *CURRENT
  Secuencia de ordenación. . . . . . . :   *HEX
    Biblioteca . . . . . . . . . . . . :
  Identificador de idioma. . . . . . . :   *JOBRUN
  Habilitar recogida de rendimiento:
    Nivel de recogida. . . . . . . . . :   *PEP
  Directorio de enlace . . . . . . . . :   *NONE
    Biblioteca . . . . . . . . . . . . :
  Grupo de activación. . . . . . . . . :   QILE
  Compilador . . . . . . . . . . . . . :   IBM ILE COBOL/400
```

Figura 9. Listado resumen del mandato CRTCBLMOD

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página     1

Mandato  . . . . . . . . . . . . . . . :   CRTBNDCBL

Valores reales:

  Programa . . . . . . . . . . . . . . :   SAMPLE2
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Archivo fuente . . . . . . . . . . . :   QCBLLESRC
    Biblioteca . . . . . . . . . . . . :     TESTLIB
    CCSID  . . . . . . . . . . . . . . :   65535
  Miembro fuente . . . . . . . . . . . :   SAMPLE         96/10/22 13:43:33
  Descripción de texto. . . . . . . . . :  Programa muestra para generar listado

Opciones de mandato:

  Programa . . . . . . . . . . . . . . :   SAMPLE2
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Archivo fuente . . . . . . . . . . . :   QCBLLESRC
    Biblioteca . . . . . . . . . . . . :     TESTLIB
  Miembro fuente . . . . . . . . . . . :   SAMPLE
  Salida . . . . . . . . . . . . . . . :   *PRINT
  Nivel de gravedad de generación. . . :   0
  Texto 'descripción'  . . . . . . . . :   *SRCMBRTXT
  Opciones de compilador . . . . . . . :   *IMBEDERR
  Opciones de conversión . . . . . . . :   *NONE
  Límite de mensajes:
    Número de mensajes . . . . . . . . :   *NOMAX
    Gravedad límite de mensajes. . . . :   30
  Vista de depuración. . . . . . . . . :   *NONE
  Nivel de optimización. . . . . . . . :   *NONE
  Distintivo FIPS  . . . . . . . . . . :   *NOFIPS *NOOBSOLETE
  Opciones de pantalla ampliada. . . . :   *NONE
  Gravedad de distintivos. . . . . . . :   0
  Sustituir programa . . . . . . . . . :   *YES
  Programa simple. . . . . . . . . . . :   *YES
  Autorización . . . . . . . . . . . . :   *LIBCRTAUT
  Literal de enlace. . . . . . . . . . :   *PGM
  Release destino. . . . . . . . . . . :   *CURRENT
  Perfil de usuario. . . . . . . . . . :   *USER
  Secuencia de ordenación. . . . . . . :   *HEX
    Biblioteca . . . . . . . . . . . . :
  Identificador de idioma. . . . . . . :   *JOBRUN
  Habilitar recogida de rendimiento:
    Nivel de recogida. . . . . . . . . :   *PEP
  Directorio de enlace . . . . . . . . :   *NONE
    Biblioteca . . . . . . . . . . . . :
  Grupo de activación. . . . . . . . . :   QILE
  Compilador . . . . . . . . . . . . . :   IBM ILE COBOL/400
```

Figura 10. Listado resumen del mandato CRTBNDCBL

Identificación de las opciones del compilador en vigor

En primer lugar se imprime la instrucción PROCESS, si se especifica. La Figura 11 presenta una lista que incluye todas las opciones en vigor para la compilación del programa de ejemplo: las opciones que se especifican en el mandato CRTCBLMOD, tal como las modifica la instrucción PROCESS. Las opciones de compilador se listan al principio de toda salida generada por el compilador si se especifica el parámetro OPTIONS.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página     2

                                   F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


          000010 PROCESS OPTIONS, SOURCE, VBSUM, MAP,
          000020         FLAG(00), MINIMUM, OBSOLETE, XREF
                    Opciones de compilador COBOL en vigor
                         SOURCE
                         XREF
                         GEN
                         NOSEQUENCE
                         VBSUM
                         NONUMBER
                         MAP
                         OPTIONS
                         QUOTE
                         NOSECLVL
                         PRTCORR
                         MONOPRC
                         RANGE
                         NOUNREF
                         NOSYNC
                         NOCRTF
                         NODUPKEYCHK
                         NOINZDLT
                         NOBLK
                         STDINZ
                         NODDSFILLER
                         NOIMBEDERR
                         STDTRUNC
                         NOCHGPOSSGN
                         NOEVENTF
                         OUTPUT
                         GENLVL(30)
                         NOOPTIMIZE
                         MINIMUM
                         OBSOLETE
                         DFRWRT
                         UNDSPCHR
                         ACCUPDALL
                         FLAG(0)
                         LINKPGM
                         SRTSEQ(*HEX      )
                         LANGID(*JOBRUN   )
                         ENBPFRCOL(PEP)
                         NOGRAPHIC


                    Opciones de conversión COBOL en vigor

                         NOVARCHAR
                         NODATETIME
                         NOCVTPICXGRAPHIC
                         NOFLOAT
```

Figura 11. Lista de opciones en vigor

Listado fuente

La Figura 12 muestra un listado fuente. Las instrucciones del programa fuente se listan exactamente de la forma en que fueron sometidas con excepción del texto fuente de programa, que se identifica en la instrucción REPLACE. El texto de sustitución aparecerá en el listado fuente. Después de la página en la que se lista el párrafo PROGRAM-ID, en todas las páginas de salida del compilador el nombre de id-programa aparece listado en la cabecera antes del nombre de sistema.

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE           AS400SYS   96/07/04 14:40:27         Página     3

  INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB
    A   B    C                                                                                  D    E          F

     1     000300 IDENTIFICATION DIVISION.

     2     000500 PROGRAM-ID.     SAMPLE.
     3     000600   AUTHOR.       PROGRAMMER NAME.
     4     000700   INSTALLATION. COBOL DEVELOPMENT CENTRE.
     5     000800   DATE-WRITTEN. 02/24/96.
     6     000900   DATE-COMPILED. 96/09/12 14:40:27


     7     001100 ENVIRONMENT DIVISION.

     8     001300 CONFIGURATION SECTION.
     9     001400 SOURCE-COMPUTER. IBM-AS400.
    10     001500 OBJECT-COMPUTER. IBM-AS400.

    11     001700 INPUT-OUTPUT SECTION.
    12     001800 FILE-CONTROL.
    13     001900     SELECT FILE-1 ASSIGN TO DISK-SAMPLE.


    15     002100 DATA DIVISION.

    16     002300 FILE SECTION.
    17     002400 FD  FILE-1
           002500     LABEL RECORDS ARE STANDARD
*==>                  a
*=a> LNC0848   0  Se comprueba la sintaxis de la cláusula LABEL y no se tiene en cuenta.  G

           002600     RECORD CONTAINS 20 CHARACTERS
           002700     DATA RECORD IS RECORD-1.
*==>                               a
*=a> LNC0848   0  Se comprueba la sintaxis de la cláusula DATA RECORDS y no se tiene en cuenta.

    18     002800 01  RECORD-1.
    19     002900   02 FIELD-A      PIC X(20).

    20     003100 WORKING-STORAGE SECTION.
    21     003200 01  FILLER.
    22     003300   05 KOUNT        PIC S9(2) COMP-3.
    23     003400   05 LETTERS      PIC X(26) VALUE "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
    24     003500   05 ALPHA REDEFINES LETTERS
           003600                   PIC X(1)  OCCURS 26 TIMES.
    25     003700   05 NUMBR        PIC S9(2) COMP-3.
    26     003800   05 DEPENDENTS   PIC X(26) VALUE "01234012340123401234012340".
    27     003900   05 DEPEND REDEFINES DEPENDENTS
           004000                   PIC X(1)  OCCURS 26 TIMES.
           004100 COPY WRKRCD.
    28    +000010 01  WORK-RECORD.                                                    WRKRCD
    29    +000020   05 NAME-FIELD   PIC X(1).                                        WRKRCD
    30    +000030   05 FILLER       PIC X(1)  VALUE SPACE.                          WRKRCD
    31    +000040   05 RECORD-NO    PIC S9(3).                                      WRKRCD
    32    +000050   05 FILLER       PIC X(1)  VALUE SPACE.                          WRKRCD
    33    +000060   05 LOCATION     PIC A(3)  VALUE "NYC".                          WRKRCD
    34    +000070   05 FILLER       PIC X(1)  VALUE SPACE.                          WRKRCD
    35    +000080   05 NO-OF-DEPENDENTS                                            WRKRCD
          +000090                   PIC X(2).                                       WRKRCD
    36    +000100   05 FILLER       PIC X(7)  VALUE SPACES.                         WRKRCD
```

Figura 12 (Parte 1 de 3). Ejemplo de un listado fuente ILE COBOL/400

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   37     004200 77  WORKPTR USAGE POINTER.



          004400*****************************************************
          004500* EL PÁRRAFO SIGUIENTE ABRE EL ARCHIVO DE SALIDA QUE*
          004600* VA A CREARSE E INICIALIZA LOS CONTADORES          *
          004700*****************************************************
   38     004800 PROCEDURE DIVISION.

          005000 STEP-1.
   39     005100     OPEN OUTPUT FILE-1.
   40     005200     MOVE ZERO TO KOUNT, NUMBR.


          005400*****************************************************
          005500* LOS 3 PÁRRAFOS SIGUIENTES CREAN INTERNAMENTE LOS  *
          005600* REGISTROS QUE DEBEN INCLUIRSE EN EL ARCHIVO, LOS  *
          005700* GRABA EN DISCO Y LOS VISUALIZA                    *
          005800*****************************************************
          005900 STEP-2.
   41     006000     ADD 1 TO KOUNT, NUMBR.
   42     006100     MOVE ALPHA  (KOUNT) TO NAME-FIELD.
   43     006200     MOVE DEPEND (KOUNT) TO NO-OF-DEPENDENTS.
   44     006300     MOVE NUMBR          TO RECORD-NO.

          006500 STEP-3.
   45     006600     DISPLAY WORK-RECORD.
   46     006700     WRITE RECORD-1 FROM WORK-RECORD.

          006900 STEP-4.
   47     007000     PERFORM STEP-2 THRU STEP-3 UNTIL KOUNT IS EQUAL TO 26.


          007200*****************************************************
          007300* EL PÁRRAFO SIGUIENTE CIERRA EL ARCHIVO ABIERTO    *
          007400* PARA SALIDA Y VUELVE A ABRIRLO PARA ENTRADA       *
          007500*****************************************************
          007600 STEP-5.
   48     007700     CLOSE FILE-1.
   49     007800     OPEN INPUT FILE-1.


          008000*****************************************************
          008100* LOS PÁRRAFOS SIGUIENTES LEEN DE NUEVO EL ARCHIVO  *                                96/07/04
          008200* Y DISTINGUEN LOS EMPLEADOS SIN SUBORDINADOS       *                                96/07/04
          008300*****************************************************
          008400 STEP-6.
   50     008500     READ FILE-1 RECORD INTO WORK-RECORD
   51     008600       AT END GO TO STEP-8.

          008800 STEP-7.
   52     008900     IF NO-OF-DEPENDENTS IS EQUAL TO "0"
   53     009000       MOVE "Z" TO NO-OF-DEPENDENTS.
   54     009100     GO TO STEP-6.

          009300 STEP-8.
```

Figura 12 (Parte 2 de 3). Ejemplo de un listado fuente ILE COBOL/400

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE           AS400SYS   96/07/04 14:40:27         Página     5

  INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


    55     009400     CLOSE FILE-1.
    56     009500     STOP RUN.
*===> LNC0650   0  Agrupar/desagrupar bloques para archivo 'FILE-1' se llevará a cabo mediante código generado por compilador.

                         * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 12 (Parte 3 de 3). Ejemplo de un listado fuente ILE COBOL/400

La Figura 12 visualiza los siguientes campos:

A *Número de instrucción generado por el compilador:* El número aparece situado a la izquierda del listado de programa fuente. Se hace referencia a estos números en todos los listados de salida de compilador excepto en el caso de .* and SAA messages los listados FIPS. Una instrucción puede ocupar varias líneas y una línea puede contener más de una instrucción. Cuando hay una secuencia de programas fuente ILE COBOL/400 en el miembro fuente de entrada, el número de instrucción se restablece a 1 en cada una de las unidades de compilación nuevas. El número de instrucción no se restablece en una única unidad de compilación que contiene uno o más programas COBOL anidados.

B *Nivel de anidamiento de programa:* El número que aparece en este campo indica el nivel de anidamiento del programa.

C *Número de referencia:* Los números aparecen situados a la izquierda de las instrucciones fuente. Los números que aparecen en este campo y la cabecera de columna (que aparece como NUMSEC en este listado) dependen de una opción especificada en el mandato CRTCBLMOD o CRTBNDCBL o en la instrucción PROCESS, tal como se muestra en la tabla siguiente:

Opción	Cabecera	Origen
NONUMBER	NUMSEC	Números de secuencia del archivo fuente
NUMBER	NUMBER	Números de secuencia proporcionados por el usuario
LINENUMBER	NUMLIN	Números de secuencia generados por el compilador

D *Columna de indicador de error de secuencia:* Una S en esta línea indica que la línea está fuera de secuencia. La comprobación de secuencia se lleva a cabo en el campo de número de referencia sólo si se especifica la opción SEQUENCE.

E *Nombrecopia:* Se indica el nombrecopia, tal como se especifica en la instrucción COPY ILE COBOL/400, para todos los registros que esa instrucción COPY ha incluido en el programa fuente. Si se utiliza la expresión DDS-ALL-FORMATS, aparece el nombre `<--ALL-FMTS` bajo NOMCOPIA.

F *Campo fecha/modificación:* Aquí se indica la fecha en que la línea se modificó por última vez.

G *Error intercalado:* El mensaje de error de primer nivel se incluye en el listado después de la línea en la que se produjo el error.Se identifica la cláusula, instrucción o expresión que provocó el error.

Listado de recuento de utilización de verbos

La Figura 13 muestra la lista alfabética que se genera de todos los verbos utilizados en el programa fuente. Además se incluye un recuento de las veces que se ha utilizado cada verbo. Este listado se genera cuando se especifica la opción VBSUM.

```
5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400      TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página     6

             U t i l i z a c i ó n   d e   v e r b o s   p o r   n ú m e r o

VERBO                   CUENTA


ADD                        1
CLOSE                      2
DISPLAY                    1
GOTO                       2
IF                         1
MOVE                       5
OPEN                       2
PERFORM                    1
READ                       1
STOP                       1
WRITE                      1

               * * * * *  F I N   D E   U T I L I Z A C I Ó N   D E   V E R B O S   P O R   N Ú M E R O  * * * * *
```

Figura 13. Listado de recuento de utilización de verbos

Mapa de la Data Division

El mapa de la Data Division se lista cuando se especifica la opción MAP. Contiene información sobre los nombres del programa fuente ILE COBOL/400. Al final del mapa de la Data Division se indica el número mínimo de bytes necesarios para File Section y Working-Storage Section.

```
5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400      TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página     7

                  M a p a   d e   D a t a   D i v i s i o n

 INST NIV  NOMBRE FUENTE               SECCION    DESP   LONG    TIPO     ATRIBUTOS
   H    I       J                          K        L       M     N           O

   17  FD  FILE-1                          FS                             DEVICE DISK, ORGANIZATION SEQUENTIAL,
                                                                          ACCESS SEQUENTIAL, RECORD CONTAINS 20
                                                                          CHARACTERS
   18  01  RECORD-1                        FS  00000000       20  GROUP
   19  02  FIELD-A                         FS  00000000       20  AN
   21  01  FILLER                          WS  00000000       56  GROUP
   22  05  KOUNT                           WS  00000000        2  PACKED
   23  05  LETTERS                         WS  00000002       26  AN       VALUE
   24  05  ALPHA                           WS  00000002        1  AN       REDEFINES LETTERS, DIMENSION(26)
   25  05  NUMBR                           WS  00000028        2  PACKED
   26  05  DEPENDENTS                      WS  00000030       26  AN       VALUE
   27  05  DEPEND                          WS  00000030        1  AN       REDEFINES DEPENDENTS, DIMENSION(26)
   28  01  WORK-RECORD                     WS  00000000       19  GROUP
   29  05  NAME-FIELD                      WS  00000000        1  AN
   30  05  FILLER                          WS  00000001        1  AN       VALUE
   31  05  RECORD-NO                       WS  00000002        3  ZONED
   32  05  FILLER                          WS  00000005        1  AN       VALUE
   33  05  LOCATION                        WS  00000006        3  A        VALUE
   34  05  FILLER                          WS  00000009        1  AN       VALUE
   35  05  NO-OF-DEPENDENTS                WS  00000010        2  AN
   36  05  FILLER                          WS  00000012        7  AN       VALUE
   37  77  WORKPTR                         WS  00000000       16  POINTR


FILE SECTION utiliza como mínimo 20 bytes de almacenamiento
WORKING-STORAGE SECTION utiliza como mínimo 91 bytes de almacenamiento

               * * * * *  F I N  D E  M A P A  D E  D A T A  D I V I S I O N  * * * * *
```

Figura 14. Mapa de la Data Division

El mapa de la Data Division muestra los campos siguientes:

H *Número de instrucción:* Se lista el número de instrucción generado por el compilador donde se definió el elemento de datos para cada uno de los elementos de datos del mapa de la Data Division.

I *Nivel de elemento de datos:* Se indica el número de nivel del elemento de datos tal como se ha especificado en el programa fuente. Los nombres de índice se identifican mediante *IX* en el número de nivel y campos en blanco en los campos SECCION, DESP, LONG y TIPO.

J *Nombre fuente* Se indica el nombre de datos, como se especifica en el programa fuente.

K *Sección:* Se indica la sección donde se definió el elemento utilizando los códigos siguientes:

```
FS  File Section
WS  Working-Storage Section
LS  Linkage Section
SM  Sort/Merge Section
SR  Special Register.
```

L *Desplazamiento:* Se indica el desplazamiento, en bytes, del elemento desde el elemento de grupo nivel -01.

M *Longitud:* Se indica la longitud decimal del elemento, en bytes.

N *Tipo:* Se indica el tipo de clase de datos del elemento utilizando los códigos siguientes:

Código	Tipo de clase de datos
GROUP	Elemento de grupo
A	Alfabético
AN	Alfanumérico
ANE	Alfanumérico editado
INDEX	Elemento de datos de índice (USAGE INDEX)
BOOLN	Booleano
ZONED	Decimal con zona (decimal externo)
PACKED	Decimal empaquetado (decimal interno) (USAGE COMP, COMP-3 o PACKED-DECIMAL)
BINARY	Binario (USAGE COMP-4 o BINARY)
FLOAT	Coma flotante interno (USAGE COMP-1 o COMP-2)
EFLOAT	Coma flotante externo (USAGE DISPLAY)
NE	Numérico editado
POINTR	Elementos de datos de puntero (USAGE POINTER)
PRCPTR	Elemento de datos de puntero de procedimiento (USAGE PROCEDURE-POINTER) .*
G	DBCS
GE	DBCS-editado

0 *Atributos:* Se indican los atributos del elemento de la forma siguiente:

- Para archivos, se puede dar la información siguiente:

  DEVICE tipo
  ORGANIZATION información
  ACCESS modalidad
  BLOCK CONTAINS información
  RECORD CONTAINS información
  LABEL información
  Se indica RERUN
  Se indica SAME AREA
  Se indica CODE-SET
  Se indica SAME RECORD AREA
  Se indica LINAGE.

- Para elementos de datos, los atributos indican si la siguiente información se especificó para elemento:

  REDEFINES información
  VALUE
  JUSTIFIED
  SYNCHRONIZED
  BLANK WHEN ZERO
  SIGN IS LEADING
  SIGN IS LEADING SEPARATE CHARACTER
  SIGN IS SEPARATE CHARACTER
  INDICATORS.

- Para elementos de tabla, se indican las dimensiones del elemento en el formato `DIMENSION (nn)`. Para cada dimensión se da un valor OCCURS máximo. Cuando una dimensión es una variable, se lista como tal, dando los valores OCCURS inferior y superior.

Mensajes FIPS

Los mensajes FIPS, Figura 15, se listan si se especifica el parámetro FLAGSTD. Consulte la página 40 para obtener más información sobre cómo especificar la opción para señalar con distintivos FIPS. Únicamente se listan los mensajes para el subconjunto FIPS, módulos opcionales y/o elementos obsoletos especificados.

Nota: Se indica el número de secuencia y el número de columna para cada vez que se emite el mensaje.

```
*
  5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400      TESTLIB/SAMPLE         TORAS015   96/07/04 13:57:14        Página     7
                        M e n s a j e s   C O B O L   F I P S
   ID-FIPS    DESCRIPCIÓN Y NÚMEROS DE SECUENCIA CON DISTINTIVO Q
   P

  LNC8100     Los elementos siguientes son elementos de lenguaje obsoletos.
  LNC8102     Párrafo AUTHOR.
                  000060    10
  LNC8103     Párrafo DATE-COMPILED.
                  000090    10
  LNC8104     Párrafo INSTALLATION.
                  000070    10
  LNC8105     Párrafo DATE-WRITTEN.
                  000080    10
  LNC8117     Cláusula LABEL RECORDS.
                  000250    12

  LNC8177     Cláusula DATA RECORDS.
                  000270    12
  LNC8200     Los elementos siguientes estándar no conformes sólo son válidos al nivel FIPS intermedio o superior.  R
  LNC8201     Instrucción COPY.
                  000410     8
  LNC8500     Los elementos siguientes no estándar no conformes son definidos por IBM o extensiones IBM.  S
  LNC8504     Nombre-asignación en cláusula ASSIGN.
                  000190    36
  LNC8518     USAGE IS COMPUTATIONAL-3.
                  000330    36
                  000370    36
  LNC8520     USAGE IS POINTER o PROCEDURE-POINTER.
                  000420    26
  LNC8561     Biblioteca por omisión tomada para instrucción COPY.
                  000410     8
  LNC8572     Instrucción SKIP1/2/3.
                  000040    13
                  000100    13
                  000120    13
                  000160    13
                  000200    13
                  000220    13
                  000300    13
                  000430    13
                  000490    13
                  000530    13
                  000640    13
                  000680    13
                  000710    13
                  000790    13
                  000870    13
                  000920    13
    28 violaciones FIPS con distintivo.
                   * * * * *  F I N  D E  M E N S A J E S  C O B O L  F I P S  * * * * *
```

Figura 15. Mensajes FIPS

Los mensajes FIPS constan de los campos siguientes:

P *ID-FIPS*: Este campo lista el número de mensaje FIPS.

Q *Números de referencia y descripción con distintivo:* Este campo lista una descripción de la condición con distintivo, seguida de una lista de números de referencia del programa fuente donde esta condición se encuentra.

El tipo de números de referencia que se utiliza y sus nombres en la cabecera (que aparece como NÚMEROS DE SECUENCIA en este listado)

dependen de una opción especificada en el mandato CRTCBLMOD o en el mandato CRTBNDCBL o en la instrucción PROCESS, como aparece en la tabla siguiente:

Opción	Cabecera
NONUMBER	DESCRIPCIÓN Y NÚMEROS DE SECUENCIA CON DISTINTIVO
NUMBER	DESCRIPCIÓN Y NÚMEROS PROPORCIONADOS POR EL USUARIO CON DISTINTIVO
LINENUMBER	DESCRIPCIÓN Y NÚMEROS DE LÍNEA CON DISTINTIVO

R *Elementos agrupados según nivel:* Estas cabeceras subdividen los mensajes FIPS según nivel y categoría.

S *Violaciones FIPS con distintivo:* Al final del listado FIPS se incluye el número total de violaciones FIPS con distintivo.

Listado de referencias cruzadas

La Figura 16 muestra el listado de referencias cruzadas que se genera cuando se especifica la opción XREF. Proporciona una lista que incluye todas las referencias de datos, referencias de nombre-procedimiento y referencias de nombre-programa del programa fuente, por número de instrucción.

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página     9

                    R e f e r e n c i a s   c r u z a d a s

Referencias de datos:

Tipo-datos se indica mediante la letra que sigue a una definición de nombre-datos.
Estas letras y su significado son:

    E = EXTERNAL
    G = GLOBAL
    X = EXTERNAL y GLOBAL


NOMBRES DATOS                   DEFINIDO  REFERENCIAS (* = CAMBIADO)
    T                               U         V

ALPHA                               24     42
DEPEND                              27     43
DEPENDENTS                          26     27
FIELD-A                             19
FILE-1                              17     39     48     49     50     55
KOUNT                               22     40*    41*    42     43     47
LETTERS                             23     24
LOCATION                            33
NAME-FIELD                          29     42*
NO-OF-DEPENDENTS                    35     43*    52     53*
NUMBR                               25     40*    41*    44
RECORD-NO                           31     44*
RECORD-1                            18     46*
WORK-RECORD                         28     45     46     50*
WORKPTR                             37


Referencias de procedimiento:

La utilización de contexto se indica mediante la letra que sigue a una referencia de nombre-procedimiento.
Estas letras y su significado son:

    A = ALTER (nombre-procedimiento)
    D = GO TO (nombre-procedimiento) DEPENDING ON
    E = Fin de rango de (PERFORM) hasta (nombre-procedimiento)
    G = GO TO (nombre-procedimiento)
    P = PERFORM (nombre-procedimiento)
    T = (ALTER) TO PROCEED TO (nombre-procedimiento)


NOMBRES PROCEDIMIENTO           DEFINIDO  REFERENCIAS


STEP-1                              38
STEP-2                              40     47P
STEP-3                              44     47E
STEP-4                              46
STEP-5                              47
STEP-6                              49     54D
STEP-7                              51
STEP-8                              54     51D
```

Figura 16 (Parte 1 de 2). Listado de referencias cruzadas

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27         Página    10

Referencias de programa:

Tipo-programa del programa externo se indica mediante la palabra que se encuentra en una definición de nombre-programa.
Estas palabras y su significado son:

    EPGM = un objeto de programa que va a enlazarse dinámicamente
    BPRC = un programa COBOL o una función C o un programa RPG que va a enlazarse
    SYS  = un programa del sistema


NOMBRES PROGRAMA               DEFINIDO  REFERENCIAS


SAMPLE                             2

                * * * * *  F I N  D E  R E F E R E N C I A S  C R U Z A D A S * * * * *
```

Figura 16 (Parte 2 de 2). Listado de referencias cruzadas

El listado de referencias cruzadas muestra los campos siguientes:

T *Campo nombres:* Se indica el nombre de dato o de procedimiento o el nombre de programa al que se hace referencia. Los nombres se listan alfabéticamente. Los nombres de programa pueden estar calificados por un nombre de biblioteca.

U *Campo Definido:* Se indica el número de instrucción en el que se definió el nombre dentro del programa fuente.

V *Campo Referencias:* Se listan todos los números de instrucción en la misma secuencia con que se hace referencia al nombre en el programa fuente. Un * a continuación del número de instrucción indica que el elemento se modificó en esa instrucción.

Mensajes

La Figura 17 muestra los mensajes generados durante la compilación del programa.

```
 5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPLE          AS400SYS   96/07/04 14:40:27        Página    11

                                  M e n s a j e s

   INST

      W                  Y                     X
*    17  IDMEN: LNC0848  GRAVEDAD:  0  NUMSEC:  002500
         Mensaje . . . . :   Se comprueba la sintaxis de la cláusula LABEL y no se tiene en cuenta.    Z

*    17  IDMEN: LNC0848  GRAVEDAD:  0  NUMSEC:  002700
         Mensaje . . . . :   Se comprueba la sintaxis de la cláusula DATA RECORDS y no se tiene
           en cuenta.

*        IDMEN: LNC0650  GRAVEDAD:  0  NUMSEC:
         Mensaje . . . . :   Agrupar/desagrupar bloques para archivo 'FILE-1' se
           realizará mediante código  generado por compilador.


                                       Resumen de mensajes
    AA
Total de mensajes:
   Información (00-04) . . . . . :      3
   Aviso       (05-19) . . . . . :      0
   Error       (20-29) . . . . . :      0
   Graves      (30-39) . . . . . :      0
   Terminales  (40-99) . . . . . :      0
   ------------------------------------------
   Total                                3


                     * * * * *  F I N  D E  M E N S A J E S  * * * * *

    BB
Estadísticas:
   Registros fuente leídos . . . . . :     95
   Registros de copia leídos . . . . :     10
   Miembros de copia procesados. . . . :    1
   Errores de secuencia. . . . . :      0
   Mensajes gravedad superior emitidos :    0


  LNC0899  0  Módulo SAMPLE creado en biblioteca TESTLIB el 08/11/96 a las 14:40:34.

                    * * * * *  F I N  D E  C O M P I L A C I O N  * * * * *
```

Figura 17. Mensajes de diagnóstico

Se visualizan los campos siguientes:

W *Número de instrucción:* Este campo lista el número de instrucción generado por el compilador asociado a la instrucción del programa fuente para la que se emitió el mensaje.[1] para la que se emitió el mensaje.

Nota: El número de instrucción y el número de referencia no aparecen para algunos mensajes que hacen referencia a elementos que faltan. Por ejemplo, si el párrafo PROGRAM-ID falta, el mensaje LBL0031 aparece en el listado sin número de referencia ni de instrucción

X *Número de referencia:* Se indica el número de referencia.[1] Los números que aparecen en este campo y la cabecera de columna (que aparece como NUMSEC) dependen de una opción especificada en el mandato

[1] El número de instrucción y el número de referencia no aparecen en algunos mensajes que hacen referencia a elementos que faltan. Por ejemplo, si falta el párrafo PROGRAM-ID, el mensaje LNC0031 aparece en el listado sin número de referencia ni de instrucción.

CRTCBLMOD o en el mandato CRTBNDCBL o en la instrucción PROCESS, como se indica en la tabla siguiente.

Opción	Cabecera	Origen
NONUMBER	NUMSEC	Números de secuencia del archivo fuente
NUMBER	NUMBER	Números de secuencia proporcionados por el usuario
LINENUMBER	NUMLIN	Números de secuencia generados por el compilador

Cuando se emite un mensaje para un registro de un archivo de copia, el número va precedido de un signo +.

Y *IDMEN* y *Nivel de gravedad:* Estos campos indican el número de mensaje y el nivel de gravedad asociado. Los niveles de gravedad se definen del modo siguiente:

```
00  Informativo
10  Aviso
20  Error
30  Error grave
40  Irrecuperable (normalmente un error del usuario)
50  Irrecuperable (normalmente un error de compilador)
```

Z *Mensaje:* El mensaje identifica la condición e indica la acción que realiza el compilador.

AA *Estadísticas de mensajes:* Este campo indica el número total de mensajes y el número de mensajes según el nivel de gravedad.

El total que se indica es el número de mensajes que el compilador genera para cada nivel de gravedad y no siempre son el número listado. Por ejemplo, si se ha especificado FLAG(10), no se listan aquellos mensajes cuyo nivel de gravedad sea inferior a 10. El recuento, sin embargo, indica el número que se hubiera impreso si estos no se hubieran suprimido.

BB *Estadísticas del compilador:* Este campo lista el número total de registros fuente leídos, registros de copia leídos, miembros de copia procesados, errores de secuencia encontrados y los mensajes de gravedad superior emitidos.

Capítulo 4. Creación de un objeto de programa

Este apartado proporciona información sobre:

- cómo crear un objeto de programa enlazando uno o más objetos de módulo entre sí.
- cómo crear un objeto de programa a partir de instrucciones fuente ILE COBOL/400
- el mandato CRTBNDCBL y sus parámetros
- cómo leer un listado del enlazador
- cómo crear objetos de programa utilizando uno o más objetos de módulo, programas de servicio y directorios de enlace ILE.

Definición de un objeto de programa

Un **objeto de programa** es un objeto del sistema ejecutable de tipo *PGM. Para ILE COBOL/400, el nombre del objeto de programa viene determinado por el mandato CRTBNDCBL, el mandato CRTPGM o el párrafo PROGRAM-ID del programa fuente COBOL más externo. El proceso que crea un objeto de programa a partir de uno o más objetos de módulo y programas de servicio referenciados se conoce como **enlace**. Uno o más objetos de módulo se crean con el mandato CRTCBLMOD, o se crean temporalmente con el mandato CRTBNDCBL antes de crear uno o más objetos de programa enlazados. El enlace es un proceso que toma objetos de módulo producidos por los mandatos CRTCBLMOD o CRTBNDCBL y los combina para crear un programa de servicio o un objeto de programa enlazado ejecutable.

Cuando se crea un objeto de programa, sólo los procedimientos ILE en aquellos objetos de módulo que contienen datos de depuración pueden depurarse con el depurador fuente ILE. Los datos de depuración no afectan el rendimiento del objeto de programa en ejecución. Los datos de depuración aumentan el tamaño del objeto de programa generado.

Los objetos de programa se ejecutan utilizando llamadas dinámicas de programas. El punto de entrada al objeto de programa es el PEP.

El proceso de enlace

El proceso de enlace mejora el rendimiento de la ejecución ya que los objetos de programa pueden utilizar llamadas estáticas de procedimientos a rutinas ya enlazadas como parte de un objeto de programa. Las llamadas dinámicas de programas no son necesarias para acceder a las rutinas. Los objetos de módulo individuales que se han creado con diferentes compiladores HLL ILE pueden enlazarse conjuntamente en el mismo objeto de programa permitiendo que una rutina se codifique en el lenguaje más apropiado y a continuación se enlace a un objeto de programa que la requiera. Los objetos de módulo compilados anteriormente pueden enlazarse en varias secuencias para crear nuevos objetos de programa ejecutables. Los objetos de módulo compilados anteriormente pueden volver a utilizarse para crear nuevos objetos de programa ejecutables sin tener que volver a compilar el programa fuente original. De este modo, el objeto de módulo puede volver a utilizarse cuando sea necesario. En lugar de volver a crear programas

cada vez que el objeto de módulo cambia, pueden utilizarse programas de servicio. Las rutinas comunes pueden crearse como programas de servicio. Si la rutina cambia pero su interfaz no lo hace, o si sólo se realizan cambios compatibles avanzados en la interfaz, el cambio podrá incorporarse volviendo a crear el programa de servicio. Los objetos de programa y programas de servicio que utilizan esas rutinas comunes no tienen que volver a crearse.

Existen dos vías para crear un objeto de programa utilizando el proceso de enlace. El diagrama que aparece a continuación le indica las dos vías disponibles:

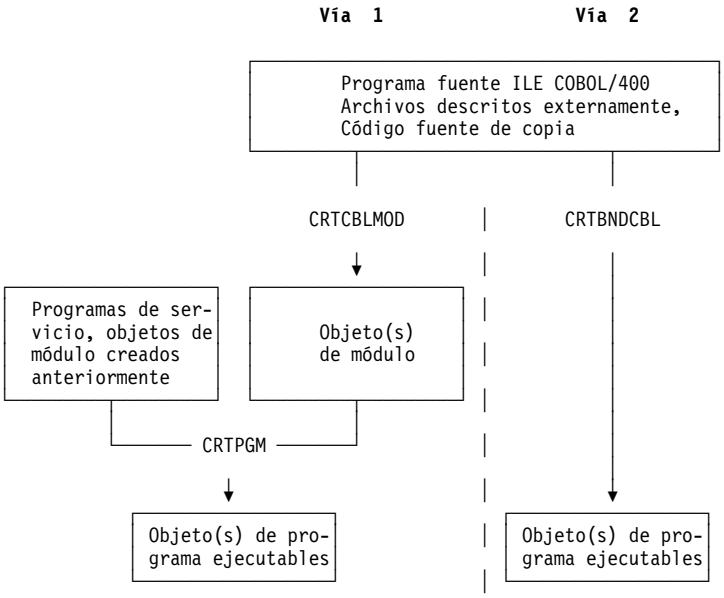


Figura 18. Dos vías para crear un objeto de programa

Estas dos vías utilizan el proceso de enlace. El mandato Crear programa (CRTPGM) crea un objeto de programa con objetos de módulo creados a partir de programas fuente ILE COBOL/400 utilizando el mandato Crear módulo COBOL (CRTCBLMOD) y cero o más programas de servicio.

Nota: Los objetos de módulo creados con los mandatos Crear módulo RPG (CRTRPGMOD), Crear módulo C (CRTCMOD) y Crear módulo CL (CRTCLMOD) también pueden enlazarse utilizando el mandato Crear programa (CRTPGM).

El mandato Crear COBOL enlazado (CRTBNDCBL) crea uno o más objetos de módulo temporales a partir de una más unidades de compilación ILE COBOL/400, y a continuación crea uno o más objetos de programa. Una vez creado el objeto de programa, CRTBNDCBL suprime el o los objetos de módulo que ha creado. Este mandato realiza las tareas combinadas de los mandatos Crear módulo COBOL (CRTCBLMOD) y Crear programa (CRTPGM) en un sólo paso. No obstante, solo puede enlazarse un único miembro fuente de entrada utilizando este paso.

Nota: Cada uno de los objetos de programa sólo reconoce un PEP (y un UEP). Si se enlazan varios objetos de módulo entre sí para crear un objeto de programa utilizando CRTPGM y cada uno de estos objetos de módulo tiene un PEP, es preciso especificar qué PEP de objetos de módulo va a utili-

zarse como PEP del objeto de programa en el parámetro ENTMOD. Asimismo, el orden en que los objetos de módulo y programas de servicio se especifican en el mandato CRTPGM afecta el modo en que se resuelven los símbolos durante el proceso de enlace. Por lo tanto, es importante que se comprenda cómo se realiza el proceso de enlace. Para obtener más información sobre el proceso de enlace, consulte el manual *ILE Conceptos*.

Un **directorio de enlace** contiene los nombres de los módulos y programas de servicio que es posible que necesite para crear un programa de servicio o programa ILE. Los módulos o programas de servicio que aparecen en el directorio de enlace se utilizan cuando proporcionan una exportación que permite satisfacer alguna petición de importación no resuelta. Un directorio de enlace es un objeto del sistema que se identifica mediante el símbolo *BNDDIR.

Los directorios de enlace son opcionales. Los motivos para la utilización de directorios de enlace son la comodidad y el tamaño del programa.

- Ofrecen un método conveniente para empaquetar los módulos o programas de servicio que es posible que necesite al crear su propio programa de servicio o programa ILE. Por ejemplo, un directorio de enlace puede contener todos los módulos y programas de servicio que proporcionan funciones matemáticas. Si desea utilizar algunas de estas funciones, es suficiente con especificar el directorio de enlace sin necesidad de especificar cada módulo o programa de servicio que utilice.
- Los directorios de enlace pueden reducir el tamaño del programa, ya que no es necesario especificar módulos o programas de servicio que no se utilizan.

Las restricciones de las entradas de un directorio de enlace son escasas. El nombre de un módulo o programa de servicio puede añadirse a un directorio de enlace aunque este objeto todavía no exista.

Para obtener una lista de los mandatos CL que se utilizan con los directorios de enlace, consulte el manual *ILE Conceptos*. Características de un objeto *BNDDIR:

- Es un método conveniente para agrupar los nombres de los módulos y programas de servicio que es posible que se necesiten para crear un programa de servicio o un programa ILE.
- Como las entradas de un directorio de enlace son sólo nombres, no es necesario que la lista de objetos ya exista en el sistema.
- Los únicos nombres de biblioteca válidos son *LIBL o la biblioteca específica.
- Los objetos de la lista son opcionales. Los objetos incluidos sólo se utilizan si existen importaciones no resueltas y el objeto proporciona una exportación que permite cumplir la petición de importación no resulta.

Utilización del mandato Crear programa (CRTPGM)

El mandato Crear programa (CRTPGM) crea un objeto de programa a partir de uno o más objetos de módulo creados anteriormente y, si es necesario, uno o más programas de servicio. Es posible enlazar objetos de módulo creados con cualquiera de los mandatos Crear módulo ILE: CRTCBLMOD, CRTCMOD, CRTRPGMOD o CRTCLMOD.

Nota: Para utilizar el mandato CRTPGM, debe tener autorización sobre éste y los módulos necesarios tienen que haberse creado con anterioridad utilizando los mandatos CRTCBLMOD, CRTCMOD, CRTRPGMOD o CRTCLMOD.

Antes de crear un objeto de programa utilizando el mandato CRTPGM, siga los pasos siguientes:

1. Establezca un nombre de programa.
2. Identifique el objeto u objetos de módulo y, si es necesario, el programa o programas de servicio que desea enlazar en el objeto de programa.
3. Identifique los directorios de enlace que va a utilizar. Las referencias implícitas a directorios de enlace para programas de servicio de ejecución ILE COBOL/400 y API enlazables ILE, se realizan a partir de objetos de módulo creados con los mandatos CRTCBLMOD y CRTBNDCBL.
4. Identifique qué PEP de objeto de módulo se va a utilizar como el PEP para el objeto de programa que se está creando.Especifique este objeto de módulo en el parámetro ENTMOD de CRTPGM.

   Si se especifica ENTMOD(*FIRST) en lugar de identificar de forma explícita un objeto de módulo en el parámetro ENTMOD, el orden en que se produce el enlace es importante para decidir qué objeto de módulo contiene el PEP para el objeto de programa que se está creando. Es posible que los objetos de módulo que se listan en el parámetro MODULE o aquellos localizados mediante un directorio de enlace contengan uno o más PEP cuando sólo puede utilizarse uno. El orden en que se realiza el enlace también es importante por otras razones como, por ejemplo, la resolución de símbolos. Para obtener más información sobre el enlace, consulte el manual *ILE Conceptos*.
5. Establezca si el mandato permitirá nombres de variables y/o procedimientos duplicados.

   Es posible que esté enlazando en un objeto de programa objetos de módulo en los que cada uno define los mismos nombres de procedimiento y nombres de variables de varias formas diferentes.
6. Identifique el grupo de activación donde se ejecuta el programa. Consulte el apartado “Activación y grupos de activación” en la página 175 para obtener una descripción sobre los grupos de activación.

Para crear un objeto de programa utilizando el mandato CRTPGM, lleve a cabo los pasos siguiente:

1. Entre el mandato CRTPGM.
2. Entre los valores adecuados para los parámetros del mandato.

La Tabla 1 en la página 75 lista los parámetros del mandato CRTPGM y sus valores por omisión. Para obtener una descripción completa del mandato CRTPGM y sus parámetros, consulte el manual *CL Reference*.

Tabla 1. Parámetros del mandato CRTPGM y sus valores por omisión

Grupo parámetros	Parámetro(Valor por omisión)
Identificación	PGM(*nombre biblioteca/nombre programa*) MODULE(*PGM)
Acceso programa	ENTMOD(*FIRST)
Enlace	BNDSRVPGM(*NONE) BNDDIR(*NONE)
Ejecución	ACTGRP(*NEW)
Varios	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF) DETAIL(*NONE) ALWUPD(*YES) ALWRINZ(*NO) REPLACE(*YES) AUT(*LIBCRTAUT) TEXT(*ENTMODTXT) TGTRLS(*CURRENT) USRPRF(*USER)

Una vez que se ha entrado el mandato CRTPGM, el objeto de programa se crea de la forma siguiente:

1. Los objetos de módulo listados se copian en lo que se convertirá en el objeto de programa.
2. El objeto de módulo que contiene el PEP se identifica y se localiza la primera importación de este módulo.
3. Los objetos de módulo se comprueban en el orden en que se listan y la primera importación se compara con una exportación de módulo.
4. Se vuelve al primer objeto de módulo y se localiza la siguiente importación.
5. Se resuelven todas las importaciones del primer objeto de módulo.
6. Se continua con el siguiente objeto de módulo y se resuelven todas las importaciones.
7. Se resuelven todas las importaciones de cada uno de los objetos de módulo siguientes hasta que se hayan resuelto todas.
8. Si se especifica OPTION(*RSLVREF) y alguna de las importaciones no pueden resolverse con una exportación, el proceso de enlace termina sin crear el objeto de programa. Si se especifica OPTION(*UNRSLVREF) no es necesario que todas las importaciones se resuelvan con exportaciones para crear el objeto de programa. Si el objeto de programa utiliza una de estas importaciones no resueltas durante la ejecución, se emitirá el mensaje de excepción MCH4439.
9. Cuando todas las importaciones se hayan resuelto, el proceso de enlace finalizará y se creará el objeto de programa.

Ejemplo de enlace de varios módulos para crear un objeto de programa

Este ejemplo muestra cómo utilizar el mandato CRTPGM para enlazar los objetos de módulo A, B y C en el objeto de programa ABC. El objeto de módulo que contiene el PEP y el UEP para el objeto de programa es el objeto de módulo indicado en el parámetro ENTMOD.

Deberían resolverse todas las referencias externas para que el mandato CRTPGM enlace varios módulos en un objeto de programa. Las referencias a las funciones de ejecución ILE COBOL/400 se resuelven a medida que se enlazan automáticamente en un objeto de programa que contiene objetos de módulo ILE COBOL/400.

1. Para enlazar varios objetos de módulo para crear un objeto de programa, teclee:

   ```
   CRTPGM PGM(ABC) MODULE(A B C) ENTMOD(*FIRST) DETAIL(*FULL)
   ```

   y pulse Intro.

Utilización del mandato Crear COBOL enlazado (CRTBNDCBL)

El mandato Crear COBOL enlazado (CRTBNDCBL) crea uno o más objetos de programa a partir de un único miembro de archivo fuente ILE COBOL/400 en un solo paso. Este mandato arranca el compilador ILE COBOL/400 y crea objetos de módulo temporales que enlaza en uno o más objetos de programa de tipo *PGM.

A diferencia de lo que ocurre con el mandato CRTPGM, cuando se utiliza el mandato CRTBNDCBL, no es necesario realizar un paso anterior independiente para crear uno o más objetos de módulo utilizando el mandato CRTCBLMOD. El mandato CRTBNDCBL realiza las tareas combinadas de los mandatos Crear módulo COBOL (CRTCBLMOD) y Crear programa (CRTPGM) creando objetos de módulo temporales a partir del miembro de archivo fuente y a continuación creando uno o más objetos de programa. Una vez creado el o los objetos de programa, CRTBNDCBL suprime los objetos de módulo que ha creado.

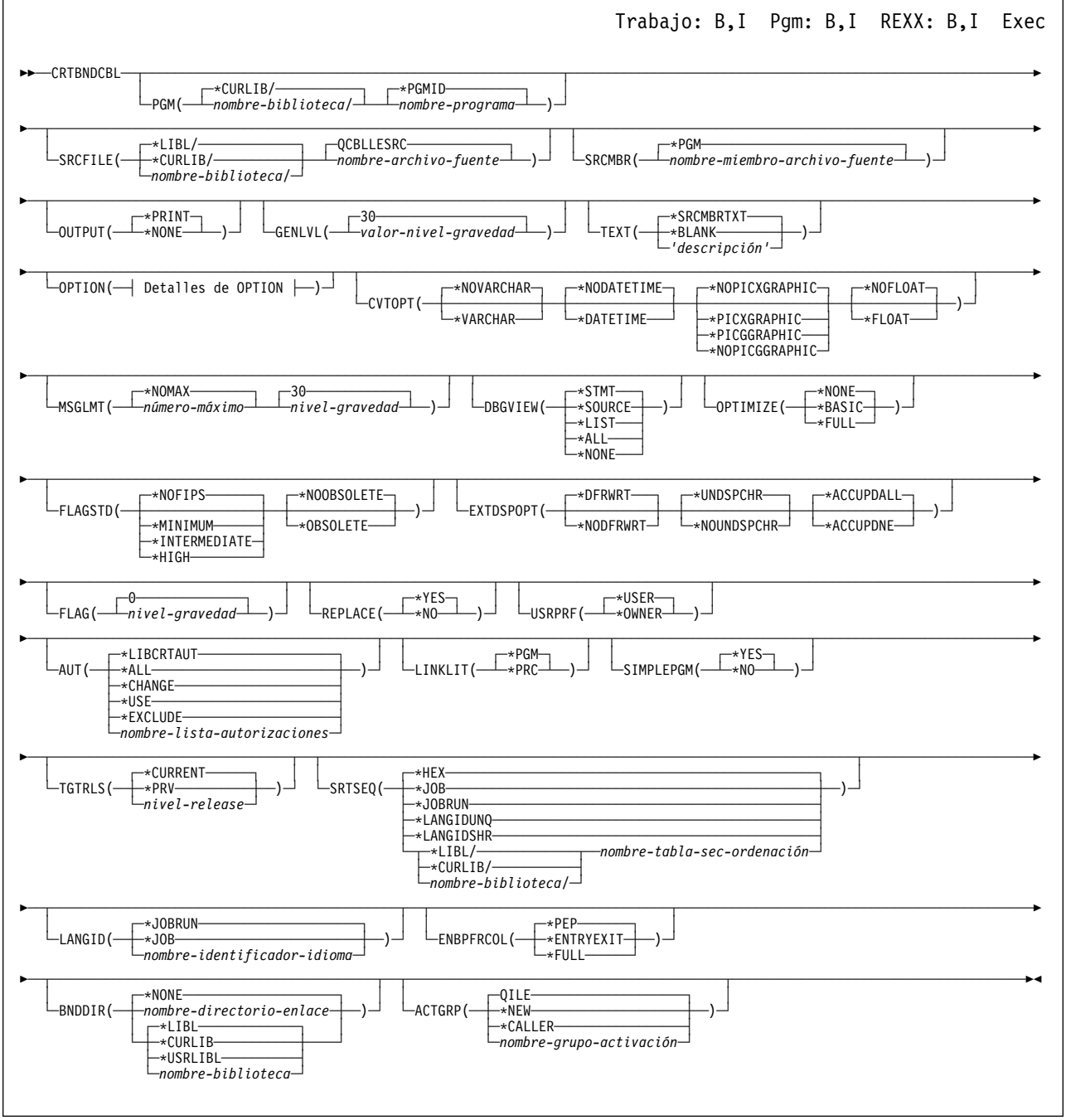
Nota: Si desea conservar los objetos de módulo, utilice CRTCBLMOD en lugar de CRTBNDCBL. Si utiliza CRTCBLMOD, tendrá que utilizar CRTPGM para enlazar los objetos de módulo en uno o más objetos de programa. Además, si desea utilizar opciones de CRTPGM diferentes de las presupuestas por CRTBNDCBL, utilice CRTCBLMOD y CRTPGM.

Utilización de pantallas de solicitud con el mandato CRTBNDCBL

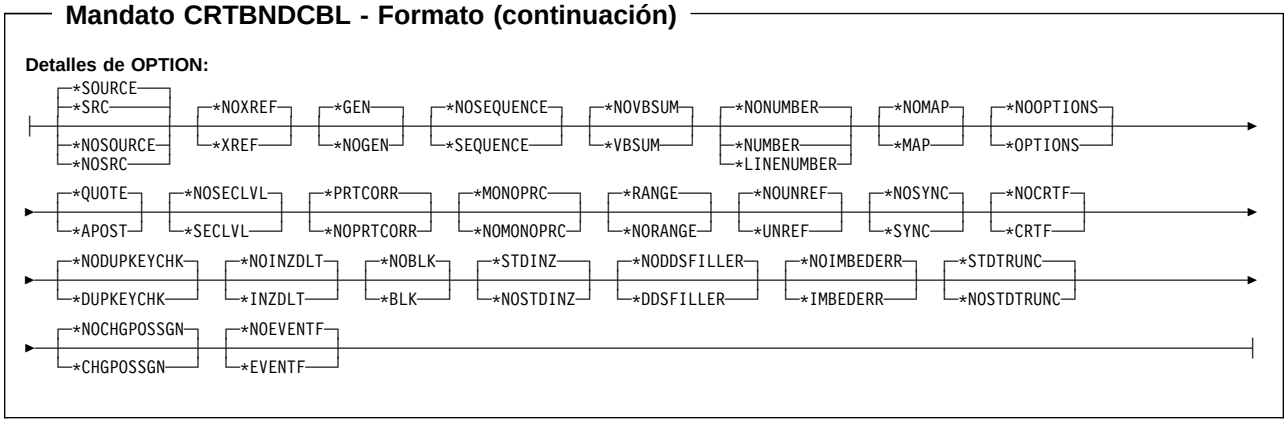
Si necesita solicitudes, teclee CRTBNDCBL y pulse F4 (Solicitud). Aparecerá la pantalla CRTBNDCBL que lista parámetros y proporciona valores por omisión. Si ya se han proporcionado parámetros antes de pedir solicitudes, la pantalla aparecerá con los valores de los parámetros cumplimentados. Si necesita ayuda, teclee CRTBNDCBL y pulse F1 (Ayuda).

Sintaxis del mandato CRTBNDCBL

Mandato CRTBNDCBL - Formato



Mandato CRTBNDCBL - Formato (continuación)



Parámetros del mandato CRTBNDCBL

Casi todos los parámetros de CRTBNDCBL son idénticos a los que se han mostrado para CRTCBLMOD anteriormente. En este apartado únicamente se indicarán las diferencias entre los dos mandatos.

CRTBNDCBL se diferencia de CRTCBLMOD en lo siguiente:

- introducción del parámetro PGM en lugar del parámetro MODULE
- el parámetro SRCMBR utiliza la opción *PGM (en lugar de la opción *MODULE)
- utilización diferente del parámetro REPLACE
- introducción del parámetro USRPRF
- introducción del parámetro SIMPLEPGM
- introducción del parámetro BNDDIR
- introducción del parámetro ACTGRP

Parámetro PGM:

Especifica el nombre de programa y el nombre de biblioteca para el objeto de programa que se está creando. El nombre de programa y de biblioteca deben cumplir las normas del convenio de denominación de AS/400. Los valores posibles son:

***PGMID**

El nombre para el objeto de programa compilado se toma del párrafo PROGRAM-ID del programa fuente ILE COBOL/400. Cuando se especifica SIMPLEPGM(*NO), el nombre del objeto de programa compilado se toma del párrafo PROGRAM-ID del primer programa fuente ILE COBOL/400 de la *secuencia de programas fuente* (varias unidades de compilación en un único miembro de archivo fuente).

Nombre-programa

Entre un nombre para identificar el programa ILE COBOL/400 compilado. Si se especifica un nombre de programa para este parámetro y se compila una secuencia de programas fuente y se especifica SIMPLEPGM(*YES), el primer objeto de programa de la secuencia utiliza este nombre; el resto de objetos de programa utilizan el nombre especificado en el párrafo PROGRAM-ID del programa fuente ILE COBOL/400 correspondiente.

Los posibles valores para la biblioteca son:

***CURLIB**
El objeto de programa que se crea se almacena en la biblioteca actual. Si no se ha asignado una biblioteca como biblioteca actual, se utiliza QGPL.

nombre-biblioteca
Entre el nombre de la biblioteca donde se va a almacenar el objeto de programa creado.

Parámetro REPLACE:
Especifica si se crea un objeto de programa nuevo cuando ya existe un objeto de programa con el mismo nombre en la biblioteca especificada o presupuesta. Los objetos de módulo intermedios que se crean durante el proceso del mandato CRTBNDCBL no están sujetos a las especificaciones de REPLACE y tienen un REPLACE(*NO) presupuesto contra la biblioteca QTEMP. Los objetos de módulo intermedios se suprimen cuando el mandato CRTBNDCBL finaliza el proceso. Los valores posibles para el parámetro REPLACE son:

***YES**
Se crea un objeto de programa nuevo y se sustituyen los objetos de programa con el mismo nombre existentes en la biblioteca especificada o presupuesta. El objeto de programa con el mismo nombre existente en la biblioteca especificada o presupuesta se traslada a la biblioteca QRPLOBJ.

***NO**
No se crea un objeto de programa nuevo si un objeto de programa con el mismo nombre ya existe en la biblioteca especificada. El objeto de programa existente no se sustituye, aparece un mensaje y la compilación se detiene.

Parámetro USRPRF:
Especifica el perfil de usuario que ejecutará el objeto de programa que se ha creado. Se utiliza el perfil del propietario del programa o del usuario del programa para ejecutar el objeto de programa y controlar los objetos que puede utilizar este objeto (incluyendo la autorización que el objeto de programa tiene sobre cada objeto). Este parámetro no se actualiza si el objeto de programa ya existe. Para modificar el valor de USRPRF, suprima el objeto de programa y vuelva a compilar utilizando el valor correcto (o, si los objetos *MODULE componentes existen, puede elegir invocar el mandato CRTPGM).

Los valores posibles son:

***USER**
Cuando se ejecute el objeto de programa se utilizará el perfil del usuario del programa.

***OWNER**
Cuando se ejecute el objeto de programa se utilizarán tanto el perfil de usuario de propietario como el de usuario. La agrupación conjunta de autorizaciones sobre objetos de los perfiles de usuario y de propietario se utilizarán para encontrar y acceder a objetos durante la ejecución del objeto de programa. Todos los objetos que se crean cuando se ejecuta el programa son propiedad del usuario del programa.

Parámetro SIMPLEPGM:
Especifica si se crea un objeto de programa para cada una de las unidades de compilación de la secuencia de programas fuente. Esta opción solo tiene

sentido si el miembro fuente de entrada para este mandato contiene una secuencia de programas fuente que generan varios objetos de módulo. Esta opción no se tendrá en cuenta si se especifica pero el miembro fuente de entrada no tiene una secuencia de programas fuente. Los valores posibles son:

***YES**
Se crea un objeto de programa para cada una de las unidades de compilación de la secuencia de programas fuente. Si se especifica REPLACE(*NO) y un objeto de programa con el mismo nombre ya existe para una unidad de compilación de la secuencia de programas fuente, ese objeto de programa no se sustituye y la compilación continúa en la siguiente unidad de compilación

***NO**
Sólo se crea un objeto de programa de todas las unidades de compilación de la secuencia, siendo la primera unidad de compilación la que represente la Entrada de programa. Con SIMPLEPGM(*NO), si un programa fuente de la secuencia de programas fuente no puede generar un objeto de módulo debido a una anomalía, ninguno de los programas fuente siguientes de la secuencia podrá generar objetos de módulo.

Parámetro BNDDIR:
Especifica la lista de directorios de enlace que se utilizan en la resolución de símbolos. Es posible especificar hasta 50 directorios de enlace.

***NONE**
No se especifica ningún directorio de enlace.

nombre-directorio-enlace
Especifique el nombre del directorio de enlace que se utiliza en la resolución de símbolos. El nombre de directorio puede calificarse con uno de los siguientes valores de biblioteca:

***LIBL**
El sistema busca en la lista de bibliotecas para hallar la biblioteca en la que está almacenado el directorio de enlace. Se trata del valor por omisión.

***CURLIB**
La búsqueda se realiza en la biblioteca actual del trabajo. Si no se especifica ninguna biblioteca como la biblioteca actual del trabajo se utiliza la biblioteca QGPL.

***USRLIBL**
La búsqueda sólo se realiza en las bibliotecas de la parte del usuario de la lista de bibliotecas del trabajo.

nombre-biblioteca
Especifique el nombre de la biblioteca en que debe realizarse la búsqueda.

Parámetro ACTGRP:
Especifica el grupo de activación al que este programa está asociado cuando se llama.

QILE
Cuando se llama a este programa, éste se activa en el grupo de activación QILE indicado. Se trata del valor por omisión.

***NEW**
Cuando se llama a este programa, éste se activa en un grupo de activación nuevo.

***CALLER**
Cuando se llama a este programa, éste se activa en el grupo de activación del llamador.

nombre-grupo-activación
Especifique el nombre del grupo de activación que debe utilizarse cuando se llame a este programa.

Invocación a CRTPGM de forma implícita desde CRTBNDCBL

Existen valores por omisión presupuestos para utilizar cuando CRTPGM se invoca implícitamente desde el mandato CRTBNDCBL. Se describen a continuación.

Los parámetros utilizados en CRTPGM cuando se invoca desde CRTBNDCBL son los siguientes:

PGM
Cuando se especifica o presupone SIMPLEPGM(*YES), se invoca a CRTPGM para cada unidad de compilación en la secuencia de programas fuente. El nombre de programa especificado en el párrafo PROGRAM-ID del programa fuente ILE COBOL/400 más externo de cada unidad de compilación se utiliza con el parámetro PGM para CRTPGM cada vez que éste se invoca. Se crea un objeto de programa enlazado individualmente para cada unidad de compilación.

Cuando se especifica SIMPLEPGM(*NO), CRTPGM se invoca sólo una vez contra todas las unidades de compilación de la secuencia de programas fuente a la vez. Únicamente se utiliza el nombre de programa especificado en el párrafo PROGRAM-ID del programa fuente ILE COBOL/400 más externo correspondiente de la primera unidad de compilación de la secuencia de programas fuente con el parámetro PGM para CRTPGM cuando éste se invoca. Todas las unidades de compilación se enlazan entre sí para crear un objeto de programa.

MODULE
Cuando se especifica o presupone SIMPLEPGM(*YES), el nombre del módulo creado en QTEMP para cada unidad de compilación se utiliza con el parámetro MODULE para CRTPGM cada vez que éste se invoca.

Cuando se especifica SIMPLEPGM(*NO), todos los nombres de los módulos que se han creado en QTEMP para las unidades de compilación se listan en el parámetro MODULE para el mandato CRTPGM cuando éste se invoca.

BNDDIR
Especifica la lista de directorios de enlace que se utilizan en la resolución de símbolos.

Si se especifica *NONE (el valor por omisión), no se utiliza ningún directorio de enlace.

Si se especifica *nombre-directorio-enlace*, el nombre del directorio de enlace que especifique se utiliza en la resolución de símbolos. El nombre de directorio puede calificarse con uno de los siguientes valores de biblioteca:

.

***LIBL** El sistema busca en la lista de bibliotecas para hallar la biblioteca en la que está almacenado el directorio de enlace. Se trata del valor por omisión.

***CURLIB** La búsqueda se realiza en la biblioteca actual del trabajo. Si no se especifica ninguna biblioteca como la biblioteca actual del trabajo se utiliza la biblioteca QGPL.

***USRLIBL**
La búsqueda sólo se realiza en las bibliotecas de la parte del usuario de la lista de bibliotecas del trabajo.

nombre-biblioteca
Especifique el nombre de la biblioteca en que debe realizarse la búsqueda.

ACTGRP
Se utiliza el grupo de activación especificado

REPLACE
Se utiliza la opción REPLACE especificada en el mandato CRTBNDCBL

USRPRF
Se utiliza la opción USRPRF especificada en el mandato CRTBNDCBL

AUT
Se utiliza la opción AUT especificada en el mandato CRTBNDCBL

TEXT
Se utiliza la opción TEXT especificada en el mandato CRTBNDCBL

TGTRLS
Se utiliza la opción TGTRLS especificada en el mandato CRTBNDCBL

Se utilizan los valores por omisión para el resto de parámetros de CRTPGM cuando se invoca desde el mandato CRTBNDCBL. Consulte el mandato CRTPGM en el manual *CL Reference* para obtener una descripción de estos valores por omisión.

Ejemplo de enlace de un objeto de módulo para crear un objeto de programa

Este ejemplo muestra cómo crear un objeto de programa a partir de un módulo utilizando el mandato CRTBNDCBL.

1. Para crear un objeto de programa, teclee:

   ```
   CRTBNDCBL PGM(MYLIB/XMPLE1)
   SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
   OUTPUT(*PRINT)
   TEXT('Programa ILE COBOL/400')
   CVTOPT(*FLOAT)
   ```

   y pulse Intro.

   El mandato CRTBNDCBL crea el programa XMPLE1 en MYLIB. La opción OUTPUT(*PRINT) especifica que desea un listado de compilación.

2. Teclee uno de los siguientes mandatos CL para ver el listado.

Nota: Para poder ver un listado de compilador debe tener autorización para utilizar los mandatos que se listan a continuación.

- `DSPJOB` y a continuación seleccione la opción 4 (*Visualizar archivos en spool*)
- `WRKJOB`
- `WRKOUTQ nombre-cola`
- `WRKSPLF`

Especificación de la secuencia de ordenación de idioma en CRTBNDCBL

En el momento en que se compila el programa fuente COBOL, se puede especificar de forma explícita el orden de clasificación que el programa utilizará cuando se ejecute, o se puede especificar cómo se va a determinar el orden de clasificación cuando se ejecute el programa.

El orden de clasificación se especifica mediante CRTBNDCBL del mismo modo que se hace con CRTCBLMOD. Para obtener una descripción completa de cómo especificar un orden de clasificación NLS, consulte el apartado “Especificación de la secuencia de ordenación de idioma en CRTCBLMOD” en la página 46.

Lectura de un listado del enlazador

El proceso de enlace puede producir un listado que describe los recursos que se han utilizado, los símbolos y objetos encontrados y los problemas resueltos o no producidos durante el proceso de enlace. El listado se produce como un archivo de spool para el trabajo que se utiliza para entrar el mandato CRTPGM. El valor por omisión del mandato, *NONE, indica que no se genere esta información. Sin embargo, puede seleccionar generarla como salida impresa según tres niveles de detalle, que se pueden especificar como valor del parámetro DETAIL:

- *BASIC
- *EXTENDED
- *FULL

El listado del enlazador incluye las secciones siguientes según el valor especificado en DETAIL:

Tabla 2. Secciones del listado del enlazador en base al parámetro DETAIL

Nombre sección	*NONE	*BASIC	*EXTENDED	*FULL
Resumen de opciones de mandato		X	X	X
Tabla de resumen breve		X	X	X
Tabla de resumen ampliado		X	X	
Listado de información del enlazador			X	X
Listado de referencias cruzadas				X
Estadísticas de enlace				X

La información que se incluye en este apartado puede ayudarle a diagnosticar problemas si el enlace no ha sido satisfactorio o dar información sobre lo que el enlazador se ha encontrado durante el proceso.

Ejemplo de un listado del enlazador

Los listados de ejemplo que se presentan a continuación muestran el listado del enlazador que se genera utilizando el mandato CRTPGM. En el texto siguiente se hace referencia a figuras. Estas referencias se indican mediante la impresión invertida de las letras sobre fondo oscuro, por ejemplo (Z). Las letras del texto en impresión invertida corresponden a las letras que se encuentran en las figuras.

Resumen de opciones de mandato

El resumen de opciones de mandato se produce siempre que se solicita un listado del enlazador. Muestra las opciones que se utilizaron cuando el programa ILE se creó. Es posible que desee guardar esta información sobre el mandato para consultarla en un futuro si tiene que crear el programa de nuevo. La Figura 19 muestra el listado de resumen de opciones de mandato

```
                                        Crear Programa                                          Página   1
5716SS1 V3R7M0  961108                                                TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

 Programa . . . . . :   EXTLFL
   Biblioteca . . . . . :     TESTLIB
 Módulo de proc. de entrada de programa . . . . . :   *FIRST
   Biblioteca . . . . . :
 Grupo de activación. . . . . :   *NEW
 Opciones de creación . . . . . :   *GEN        *NODUPPROC  *NODUPVAR   *WARN       *RSLVREF
 Detalle de listados. . . . . :   *FULL
 Permitir actualización . . . . . :   *YES
 Perfil de usuario. . . . . :   *USER
 Sustituir programa existente . . . . . :   *YES
 Autorización . . . . . :   *LIBCRTAUT
 Release destino. . . . . :   *CURRENT
 Permitir reinicialización. . . . . :   *NO
 Texto. . . . . :   *ENTMODTXT

 Módulo      Biblioteca          Módulo      Biblioteca          Módulo      Biblioteca          Módulo      Biblioteca

 EXTLFL      TESTLIB

 Programa                        Programa                        Programa                        Programa
 servicio    Biblioteca          servicio    Biblioteca          servicio    Biblioteca          servicio    Biblioteca

 *NONE

 Directorio                      Directorio                      Directorio                      Directorio
 enlace      Biblioteca          enlace      Biblioteca          enlace      Biblioteca          enlace      Biblioteca

 *NONE
```

Figura 19. Listado resumen de opciones del mandato CRTPGM

Tabla de resumen breve

Esta Tabla de resumen breve, disponible cuando se especifica *BASIC, *EXTENDED o *FULL, refleja los errrores hallados durante el proceso de enlace. La Figura 20 en la página 85 muestra el diseño de la Tabla de resumen breve

```
                                        Crear Programa                                        Página   3
5716SS1 V3R7M0  961108                                                TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

                                             Tabla de resumen breve

Procedimientos de entrada de programa. . . . . :   1   A

   Símbolo   Tipo      Biblioteca  Objeto      Identificador
      D        E           F          G             H
             *MODULE   TESTLIB     EXTLFL      _Qln_pep

 Varias definiciones firmes . . . . . :   0   B

 Referencias no resueltas . . . . . :   0   C


                          * * * * *  F I N  D E  T A B L A  D E  R E S U M E N  B R E V E  * * * * *
```

Figura 20. Listado CRTPGM - Tabla de resumen breve

La tabla consta de tres listas con el número de entradas en cada una de las siguientes categorías:

A *Procedimientos de entrada de programa:* Número de procedimientos controlados por un programa de llamada.

B *Varias definiciones firmes:* Número de procedimientos de exportación de módulo con el mismo nombre. Debería ser 0.

C *Referencias no resueltas:* Número de variables o procedimientos importados para los que no se localizó una exportación. Debería ser 0.

D *Símbolo #:* El número de Símbolo se obtiene del listado de información del enlazador que aparece en el apartado “Listado de información del enlazador” en la página 86. Si se especifica *BASIC en el parámetro DETAIL, este área queda en blanco.

E *Tipo:* En el campo Tipo se indica el tipo de objeto que contiene el identificador.

F *Biblioteca:* En el campo Biblioteca se indica el nombre de la biblioteca que contiene el objeto.

G *Objeto:* En el campo Objeto se indica el objeto que tiene el procedimiento de entrada de programa, la referencia no resuelta o la definición firme.

H *Identificador:* Se indica el nombre del procedimiento o variable del fuente de módulo.

En este ejemplo, el número total de procedimientos de entrada de programa, referencias no resueltas o múltiples definiciones firmes son 1, 0 y 0, respectivamente. Los recuentos de utilización que aparecen en la Figura 20 se indican en formato decimal.

Tabla de resumen ampliado

La Tabla de resumen ampliado se proporciona si se indica *EXTENDED o *FULL. Contiene información estadística que no se incluye en la Tabla de resumen breve. Esta información da una idea general de las importaciones y exportaciones que el enlazador ha resuelto. La Figura 21 en la página 86 muestra el diseño de la Tabla de resumen ampliado.

```
                                        Crear Programa                                          Página   2
5716SS1 V3R7M0  961108                                                TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

                                          Tabla de resumen ampliado

   Definiciones válidas . . . . . . . . . :          150  I
     Firmes . . . . . . . . . . . . . . . :          149
     Débiles. . . . . . . . . . . . . . . :            1
   Referencias resueltas. . . . . . . . . :           15  J
     Para definiciones firmes . . . . . . :           14
     Para definiciones débiles. . . . . . :            1


                        * * * * *  F I N  D E  T A B L A  D E  R E S U M E N  A M P L I A D O * * * * *
```

Figura 21. Listado CRTPGM - Tabla de resumen ampliado

La Tabla de resumen ampliado proporciona información estadística sobre los siguientes elementos:

I *Definiciones válidas:* Este campo indica el número de procedimientos y variables nombrados disponibles para la exportación. Las definiciones además se clasifican entre **definiciones firmes** o **definiciones débiles**. En el caso de definiciones firmes, se asigna almacenamiento para la variable o procedimiento. En el caso de definiciones débiles, se hace referencia al almacenamiento de la variable o el procedimiento.

En ILE COBOL/400, el programa fuente COBOL más externo y el procedimiento CANCEL asociado tendrán definiciones firmes. Los datos EXTERNAL y los archivos EXTERNAL tendrán definiciones débiles. CALL, CANCEL y SET ENTRY para un procedimiento estático externo tendrán importaciones de módulo que son definiciones firmes. Las referencias a datos EXTERNAL y a archivos EXTERNAL tendrán definiciones débiles como importaciones de módulo.

J *Referencias resueltas:* Este campo proporciona el número de importaciones que coinciden con exportaciones entre módulos.

Los recuentos de utilización que aparecen en la Figura 21 se indican en formato decimal.

Listado de información del enlazador

Este listado, que proporciona información mucho más detallada sobre el proceso de enlace, está disponible si se especifica *EXTENDED o *FULL. La Figura 22 en la página 87 muestra el diseño del Listado de información del enlazador

```
                                        Crear Programa                                          Página   4
5716SS1 V3R7M0  961108                                                  TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

                                     Listado de información del enlazador

 Módulo . . . . . :   EXTLFL      K
   Biblioteca . . . . . :     TESTLIB
   Enlazado . . . . . :     *YES      L
   Fecha/hora modificación. . :     96/07/04  09:45:32

       Número    Símbolo     Ref       Identificador                          Tipo      Ámbito      Exportación  Clave
           M       N           O            P                                    Q          R            S          T
       00000001  Def                   EF1                                    Datos     Módulo      Débil        F0
****                                   mejor definición débil
       00000002  Def                   EF1MAIN                                Proc      Módulo      Firme
       00000003  Def                   EF1MAIN_reset                          Proc      Módulo      Firme
       00000004  Ref         0000004D  _Qln_rut                               Datos
       00000005  Ref         00000001  EF1                                    Datos
       00000006  Ref         00000067  Q LE AG_prod_rc                        Datos
       00000007  Ref         00000066  Q LE AG_user_rc                        Datos
       00000008  Ref         0000004E  _Qln_init_mod                         Proc
       00000009  Ref         0000004F  _Qln_init_mod_bdry                     Proc
       0000000A  Ref         00000070  Q LE leBdyEpilog                       Proc
       0000000B  Ref         00000050  _Qln_init_oprg                         Proc
       0000000C  Ref         0000005E  _Qln_recurse_msg                      Proc
       0000000D  Ref         00000027  _Qln_disp_norm                        Proc
       0000000E  Ref         00000061  _Qln_stop_prg                         Proc
       0000000F  Ref         00000023  _Qln_cancel_msg                       Proc
       00000010  Ref         0000006F  Q LE leBdyCh                          Proc
       00000011  Ref         00000068  Q LE leDefaultEh                      Proc
       00000012  Ref         00000060  _Qln_fc_hdlr                          Proc

 Programa servicio. . . . . :   QLNRACPT
   Biblioteca . . . . . :     *LIBL
   Enlazado . . . . . :     *NO
   Fecha/hora modificación. . :     96/07/04  10:05:44

       Número    Símbolo     Ref       Identificador                          Tipo      Ámbito      Exportación  Clave

       00000013  Def                   _Qln_acpt_norm                         Proc                  Firme
       00000014  Def                   _Qln_acpt_console                      Proc                  Firme
       00000015  Def                   _Qln_acpt_session                      Proc                  Firme
       00000016  Def                   _Qln_acpt_time                         Proc                  Firme
       00000017  Def                   _Qln_acpt_date                         Proc                  Firme
       00000018  Def                   _Qln_acpt_day                          Proc                  Firme
       00000019  Def                   _Qln_acpt_day_of_week                  Proc                  Firme
       0000001A  Def                   _Qln_acpt_attribute                    Proc                  Firme
       0000001B  Def                   _Qln_acpt_pip                          Proc                  Firme
       0000001C  Def                   _Qln_acpt_lda                          Proc                  Firme
       0000001D  Def                   _Qln_acpt_open_feed                    Proc                  Firme
       0000001E  Def                   _Qln_acpt_io_feed                      Proc                  Firme

 Programa servicio. . . . . :   QLNRCALL
   Biblioteca . . . . . :     *LIBL
   Enlazado . . . . . :     *NO
   Fecha/hora modificación. . :     96/07/04  10:05:48
```

Figura 22 (Parte 1 de 2). Listado CRTPGM - Listado de información del enlazador

```
                                        Crear Programa                                          Página   5
5716SS1 V3R7M0  961108                                                  TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

      Número    Símbolo     Ref       Identificador                     Tipo      Ámbito      Exportación  Clave

      0000001F  Def                   _Qln_call_resolve                 Proc                  Firme
      00000020  Def                   _Qln_call_resolve_external        Proc                  Firme
      00000021  Def                   _Qln_set_proc_pointer             Proc                  Firme

 Programa servicio. . . . . :   QLNRCNCL
   Biblioteca . . . . . :     *LIBL
   Enlazado . . . . . :       *YES
   Fecha/hora modificación. . :     96/07/04  10:05:51

      Número    Símbolo     Ref       Identificador                     Tipo      Ámbito      Exportación  Clave

      00000022  Def                   _Qln_cancel_resolve_external      Proc                  Firme
      00000023  Def                   _Qln_cancel_msg                   Proc                  Firme




   .
   .
   .
   .


 Programa servicio. . . . . :   QBNPREPR
   Biblioteca . . . . . :     QSYS
   Enlazado . . . . . :       *NO
   Fecha/hora modificacón . . :     96/07/04  19:49:37

      Número    Símbolo     Ref       Identificador                     Tipo      Ámbito      Exportación  Clave

      0000009F  Def                   QbnStartPreProcessor              Proc                  Firme
      000000A0  Def                   QbnAddExtendedAttributeData       Proc                  Firme
      000000A1  Def                   QbnAddAssociatedSpaceEntry        Proc                  Firme
      000000A2  Def                   QbnRetrieveAssociatedSpace        Proc                  Firme
      000000A3  Def                   QbnAddPreProcessorLevelData       Proc                  Firme
      000000A4  Def                   QbnAddBindtimeExit                Proc                  Firme
      000000A5  Def                   QbnEndPreProcessor                Proc                  Firme

                    * * * * *  F I N  D E  L I S T A D O  D E  I N F O R M A C I Ó N  D E  E N L A Z A D O R * * * * *
```

Figura 22 (Parte 2 de 2). Listado CRTPGM - Listado de información del enlazador

Las columnas del listado contienen la información siguiente:

K *Módulo y biblioteca:* Este campo identifica la biblioteca y el nombre del objeto de módulo o programa de servicio que se ha procesado.

L *Enlazado:* Si en este campo aparece el valor *YES para un objeto de módulo, significa que el objeto de módulo se ha enlazado por copia. Si en este campo aparece el valor *YES para un programa de servicio, significa que el programa de servicio se ha enlazado por referencia. Si en este campo aparece el valor *NO para un objeto de módulo o programa de servicio, ese objeto no se incluye en el enlace.

M *Número:* Un identificador exclusivo asignado a cada uno de los elementos de datos o procedimiento ILE de este programa. Este número se utiliza en las referencias cruzadas.

N *Símbolo:* Este campo identifica el símbolo como una exportación o una importación. Si este campo muestra el valor Def significa que el símbolo es una exportación. Si este campo muestra el valor Ref significa que el símbolo es una importación.

O *Ref:* Este campo queda en blanco si Símbolo es Def. Contiene un número de símbolo si el valor de la columna Símbolo es Ref. Si la columna Símbolo es Ref, este campo contiene el número exclusivo que identifica la exportación (Def) que satisface la solicitud de importación.

P *Identificador:* Este es el nombre del símbolo que se exporta o importa.

Q *Tipo:* Si el nombre de símbolo es un procedimiento ILE, este campo contiene Proc. Si el nombre de símbolo es un elemento de datos, este campo contiene Datos.

R *Ámbito:* Este campo identifica el nivel al que se puede acceder al nombre de símbolo exportado.

S *Exportación:* Este campo identifica si los elementos de datos exportados tienen una definición débil o firme.

T *Clave:* Este campo contiene la longitud de los elementos débiles exportados. Los valores que aparecen en este campo se indican en formato hexadecimal.

Listado de referencias cruzadas

El listado de referencias cruzadas, proporcionado únicamente si se especifica *FULL, es útil para el programador que tiene un listado del enlazador muy largo y desea un índice más manejable. El listado de referencias cruzadas lista alfabéticamente todos los identificadores exclusivos del listado del enlazador con una lista correspondiente de todas las definiciones y referencias resueltas de ese identificador. La Figura 23 en la página 90muestra el diseño del Listado de referencias cruzadas.

```
                                        Crear Programa                                          Página   9
5716SS1 V3R7M0  961108                                                TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

                                     Listado de referencias cruzadas

                                                    --------Refs--------
        Identificador                          Defs        Ref         Ref         Tipo        Biblioteca    Objeto
             U                                   V                       W           X             Y            Z
        __CEEDOD                               0000008D                            *SRVPGM     *LIBL         QLEAWI
        __CEEGSI                               0000008E                            *SRVPGM     *LIBL         QLEAWI
        __CEEHDLR                              0000007A                            *SRVPGM     *LIBL         QLEAWI
        __CEEHDLU                              0000007B                            *SRVPGM     *LIBL         QLEAWI
        __CEERTX                               00000073                            *SRVPGM     *LIBL         QLEAWI
        __CEETSTA                              0000008C                            *SRVPGM     *LIBL         QLEAWI
        __CEEUTX                               00000074                            *SRVPGM     *LIBL         QLEAWI
        _C_session_cleanup                     00000082                            *SRVPGM     *LIBL         QLEAWI
        _C_session_open                        00000083                            *SRVPGM     *LIBL         QLEAWI
        _Qln_acpt_attribute                    0000001A                            *SRVPGM     *LIBL         QLNRACPT
        _Qln_acpt_console                      00000014                            *SRVPGM     *LIBL         QLNRACPT




        .
        .
        .
        .



        CEE4RAGE                               00000072                            *SRVPGM     *LIBL         QLEAWI
        CEE4RIN                                0000007D                            *SRVPGM     *LIBL         QLEAWI
        EF1                                    00000001    00000005                *MODULE     TESTLIB       EXTLFL
        EF1MAIN                                00000002                            *MODULE     TESTLIB       EXTLFL
        EF1MAIN_reset                          00000003                            *MODULE     TESTLIB       EXTLFL
        Q LE leBdyCh                           0000006F    00000010                *SRVPGM     *LIBL         QLEAWI
        Q LE leBdyEpilog                       00000070    0000000A                *SRVPGM     *LIBL         QLEAWI
        Q LE leDefaultEh                       00000068    00000011                *SRVPGM     *LIBL         QLEAWI
        Q LE AG_prod_rc                        00000067    00000006                *SRVPGM     *LIBL         QLEAWI
        Q LE AG_user_rc                        00000066    00000007                *SRVPGM     *LIBL         QLEAWI
        Q LE HdlrRouterEh                      00000076                            *SRVPGM     *LIBL         QLEAWI
        Q LE RtxRouterCh                       00000075                            *SRVPGM     *LIBL         QLEAWI
        QbnAddAssociatedSpaceEntry             000000A1                            *SRVPGM     QSYS          QBNPREPR
        QbnAddBindtimeExit                     000000A4                            *SRVPGM     QSYS          QBNPREPR
        QbnAddExtendedAttributeData            000000A0                            *SRVPGM     QSYS          QBNPREPR
        QbnAddPreProcessorLevelData            000000A3                            *SRVPGM     QSYS          QBNPREPR
        QbnEndPreProcessor                     000000A5                            *SRVPGM     QSYS          QBNPREPR
        QbnRetrieveAssociatedSpace             000000A2                            *SRVPGM     QSYS          QBNPREPR
        QbnStartPreProcessor                   0000009F                            *SRVPGM     QSYS          QBNPREPR
        QlnDumpCobol                           00000063                            *SRVPGM     *LIBL         QLNRMAIN
        QlnRtvCobolErrorHandler                00000064                            *SRVPGM     *LIBL         QLNRMAIN
        QlnSetCobolErrorHandler                00000065                            *SRVPGM     *LIBL         QLNRMAIN

                    * * * * *  F I N  D E  L I S T A D O  D E  R E F E R E N C I A S  C R U Z A D A S * * * * *
```

Figura 23. Listado CRTPGM - Listado de referencias cruzadas

Los campos contienen la información siguiente:

U *Identificador:* Nombre de la exportación que se ha procesado durante la resolución del símbolo.

V *Defs:* Número de identificación exclusivo asociado a cada exportación.

W *Refs:* Lista los números de identificación exclusivos de las importaciones (Ref) que se han resuelto para esta importación (Def).

X *Tipo:* Identifica si la exportación proviene de un objeto de módulo (*MODULE) o de un programa de servicio (*SRVPGM).

Y *Biblioteca:* Nombre de la biblioteca en la que se ha definido el símbolo descrito en esta línea.

Z *Objeto:* Nombre del objeto de módulo o programa de servicio en el que se ha definido el símbolo descrito en esta línea.

Estadísticas de enlace

La sección de Estadísticas de enlace solo se produce si se utiliza el valor *FULL en el parámetro DETAIL. Muestra cuánto tiempo de CPU del sistema se ha utilizado para enlazar determinadas partes del programa. Es posible que estos valores sólo tengan sentido cuando se comparan con salidas parecidas de otros programas ILE u otras veces en que se ha creado un programa determinado. El valor para el tiempo de CPU de compilación de lenguaje de enlace es siempre cero para un programa ILE. La Figura 24 muestra el diseño de la sección Estadísticas de enlace

```
                                        Crear Programa                                          Página  12
5716SS1 V3R7M0  961108                                                TESTLIB/EXTLFL    AS400SYS  96/07/04  09:46:41

                                              Estadísticas de enlace

 Tiempo de CPU para recogida de símbolos. . . . . . . . . . . :              .008
 Tiempo de CPU para resolución de símbolos. . . . . . . . . . :              .004
 Tiempo de CPU para resolución de directorio de enlace. . . . :              .138
 Tiempo de CPU para compilación de lenguaje enlazador . . . . :              .000
 Tiempo de CPU para creación de listados. . . . . . . . . . . :              .462
 Tiempo de CPU para creación de programas servicio/programas. :              .100

 Tiempo CPU total. . . . . . . . . . . . . . . . . . . . . . :              .999
 Tiempo CPU transcurrido. . . . . . . . . . . . . . . . . . . :             3.379


                            * * * * *  F I N  D E  E S T A D Í S T I C A S  D E  E N L A C E * * * * *

*CPC5D07 - Programa EXTLFL creado en la biblioteca TESTLIB.


                          * * * * *  F I N  D E  L I S T A D O  D E  C R E A R  P R O G R A M A  * * * * *
```

Figura 24. Listado CRTPGM - Estadísticas de enlace

Modificación de un objeto de módulo y enlace del objeto de programa de nuevo

Después de haber creado un objeto de programa, es posible que haya que modificarlo para solucionar problemas o para que cumpla con nuevos requisitos. Un objeto de programa se crea a partir de objetos de módulo, por lo tanto es posible que no se tenga que modificar el objeto de programa por completo. Se puede aislar el módulo que se tiene que modificar, modificarlo y después enlazarlo al objeto de programa de nuevo. Los cambios que se realizan en el objeto de módulo dependen de lo que deba modificarse.

Un objeto de módulo se puede cambiar de cuatro formas:

- Cambiar el programa fuente ILE COBOL/400 del objeto de módulo
- Cambiar el nivel de optimización del objeto de módulo
- Cambiar la observabilidad del objeto de módulo.
- Cambiar la habilitación de recogida de datos sobre el rendimiento.

Nota: Necesitará autorización sobre el código fuente y sobre los mandatos necesarios para realizar cualquiera de estas modificaciones en el objeto de módulo.

Si desea cambiar el nivel de optimización u observabilidad de un objeto de módulo, no tendrá que volver a crearlo de nuevo. Esto ocurre muchas veces cuando desea depurar un objeto de programa o cuando el objeto de programa está preparado para el proceso de producción. Tales modificaciones pueden llevarse a cabo con más rapidez y utilizando menos recursos del sistema que si se crea el objeto de módulo de nuevo.

En estas situaciones puede tener muchos objetos de módulo para crear a la vez. Puede utilizar el mandato Trabajar con módulos (WRKMOD) para obtener una lista de los objetos de módulo seleccionados por biblioteca, nombre, símbolo genérico o *ALL. Se puede limitar la lista para que tan sólo aparezcan los objetos de módulo que ha creado el compilador ILE COBOL/400.

Cuando se haya realizado una modificación en el objeto de módulo, se debe utilizar el mandato CRTPGM o el mandato UPDPGM para volver a enlazar el objeto de programa.

Modificación del programa fuente ILE COBOL/400

Cuando tenga que realizar modificaciones en el programa fuente ILE COBOL/400, siga los pasos siguientes:

1. Modifique el programa fuente ILE COBOL/400 donde sea necesario utilizando el SEU.Consulte el apartado “Entrada de instrucciones fuente utilizando el Programa de Utilidad para Entrada del Fuente” en la página 18 para obtener más detalles sobre cómo modificar el código fuente.
2. Compile el programa fuente ILE COBOL/400 utilizando el mandato CRTCBLMOD para crear uno o más objetos de módulo nuevos.Consulte el apartado “Utilización del mandato Crear módulo COBOL (CRTCBLMOD) ” en la página 30 para obtener una descripción de cómo compilar el programa fuente ILE COBOL/400.
3. Enlace los objetos de módulo utilizando el mandato CRTPGM o el mandato UPDPGM para crear un objeto de programa nuevo. Consulte el apartado “Utilización del mandato Crear programa (CRTPGM)” en la página 73 para obtener información sobre la creación de un objeto de programa.

Modificación de los niveles de optimización

Se pueden modificar los niveles a los que se optimiza el código generado para su ejecución en el sistema. Cuando el compilador optimiza el código, busca el método de proceso más corto para reducir la cantidad de recursos del sistema necesarios para producir la misma salida. Entonces convierte este método a código de máquina.

Por ejemplo: `a = (x + y) + (x + y) + 10.` Para resolver "a", el compilador reconoce la equivalencia entre las dos expresiones "(x + y)" y utiliza el valor que se ha calculado para suministrar el valor de la segunda expresión.

Una mayor optimización aumenta la eficacia con que el objeto de programa se ejecuta en el sistema. No obstante, si la optimización es mayor, se aumentará el tiempo de compilación y también es posible que no pueda ver variables que se hayan optimizado. Se puede modificar el nivel de optimización de un objeto de

módulo para visualizar variables con precisión mientras se depura un objeto de programa y a continuación modificar el nivel de optimización cuando el objeto de programa esté preparado para la producción.

Los compiladores ILE dan soporte a un rango de niveles de optimización. Actualmente, existen cuatro niveles de optimización, tres de los cuales están disponibles para usuarios ILE COBOL/400:

***NONE o 10** No se lleva a cabo optimización adicional en el código generado. Este nivel de optimización permite visualizar variables y modificarlas cuando el objeto de programa se está depurando. Este valor proporciona el nivel de rendimiento más bajo durante la ejecución.

***BASIC o 20** Se lleva a cabo cierta optimización (sólo a nivel de bloque local) en el código generado. Las variables pueden visualizarse pero no modificarse cuando se está depurando el objeto de programa. Este nivel de optimización mejora ligeramente el rendimiento de la ejecución.

***FULL o 30** Se lleva cabo una optimización completa (a nivel global) en el código generado. Las variables no pueden modificarse pero pueden visualizarse mientras el objeto de programa se depura. Sin embargo, puede ser que el valor que se visualice para las variables durante la depuración no sea su valor actual.

El efecto de la optimización en el rendimiento de la ejecución varía según el tipo de la aplicación.Por ejemplo, en una aplicación que realice gran cantidad de cálculos, la optimización puede mejorar el rendimiento de la ejecución de forma significativa mientras que en una aplicación con gran cantidad de E/S, la optimización puede mejorar el rendimiento de la ejecución mínimamente.

Para modificar el nivel de optimización de un objeto de módulo en un objeto de programa, utilice el mandato Trabajar con módulos (WRKMOD). Teclee WRKMOD en la línea de mandatos y aparecerá la pantalla Trabajar con módulos. Seleccione la opción 5 (Visualizar) de la pantalla Trabajar con módulos para ver los valores de atributo que deben modificarse. La pantalla Visualizar información de módulo aparece en la Figura 25 en la página 94.

```
                          Visualizar información de módulo
                                                                  Pantalla 1 de 7
 Módulo . . . . . :   COPYPROC
   Biblioteca . . . . . :     TESTLIB
 Detalle. . . . . :   *BASIC
 Atributo de módulo . . . . . :   CBLLE

 Información de módulo
   Fecha/hora creación de módulo . . . . . :   96/07/04  11:46:06
   Archivo fuente. . . . . :
     Biblioteca. . . . . :
   Miembro fuente. . . . . :
   Fecha/hora modificación archivo fuente. . . . . :
   Propietario . . . . . :   TESTLIB
   Identificador juego caracteres codificados. . . . . :   65535
   Descripción de texto. . . . . :   PG - COPY en Ejemplo
Instrucción PROCESS
   Datos de creación . . . . . :   *YES
   Tabla secuencia ordenación. . . . . :   *HEX
                                                                  Más....
 Pulse Intro para continuar

 F3=Salir  F12=Cancelar
```

Figura 25. Primera pantalla Visualizar información de módulo

Primero, compruebe si el valor de *plantilla de instrucciones de máquina incluida* es *YES. Esto significa que el objeto de módulo puede convertirse de nuevo después de modificar el valor del nivel de optimización. Si el valor es *NO, debe crear el objeto de módulo de nuevo e incluir la plantilla de instrucciones de máquina para cambiar el nivel de optimización.

A continuación, pulse la tecla Giro abajo, para ver más información sobre el objeto de módulo.

```
                        Visualizar información de módulo
                                                                Pantalla 1 de 7
Módulo . . . . . :   COPYPROC
  Biblioteca . . . . . :     TESTLIB
Detalle. . . . . :   *BASIC
Atributo de módulo . . . . . :   CBLLE

  Identificador de idioma . . . . . :   *JOBRUN
  Nivel de optimización . . . . . :   *NONE
  Nivel optimización máximo . . . . . :   *FULL
  Datos depuración. . . . . :   *NO
  Compactado  . . . . . :   *NO
  Nombre procedimiento entrada programa . . . . . :   _Qln_pep
  Número de parámetros  . . . . . :   0
  Estado de módulo. . . . . :   *USER
  Dominio de módulo . . . . . :   *SYSTEM
  Número de símbolos definidos exportados . . . . . :   2
  Número de símbolos (no resueltos) importados. . . . . :   14

                                                                      Más....
Pulse Intro para continuar

F3=Salir  F12=Cancelar
```

Figura 26. Segunda pantalla Visualizar información de módulo

Compruebe el valor del nivel de optimización. Puede ser que ya esté en el nivel que desee.

Si el módulo tiene la plantilla de instrucciones de máquina y desea cambiar el nivel de optimización, pulse F12 (Cancelar). Aparecerá la pantalla Trabajar con módulos. Seleccione la opción 2 (Cambiar) para el objeto de módulo cuyo nivel de optimización desea cambiar. Aparecerá la pantalla del mandato CHGMOD como se ilustra en la Figura 28 en la página 97. Teclee sobre el valor especificado para la solicitud *Optimizar módulo*.

| A continuación, pulse la tecla Giro abajo, para ver el final de la información sobre
| el objeto de módulo.

```
                          Visualizar información de módulo
                                                                  Pantalla 1 de 7
 Módulo . . . . . . . . . . . . :   COPYPROC
   Biblioteca . . . . . . . . . :     TESTLIB
 Detalle. . . . . . . . . . . . :   *BASIC
 Atributo de módulo . . . . . . :   CBLLE

Habilitar recogida de rendimiento . . . . . . . . . . :   *PEP


 Compatibilidad de módulo:
   Módulo creado en. . . . . . . . . . . . . . . . . . :   V3R7M0
   Módulo creado para. . . . . . . . . . . . . . . . . :   V3R7M0
   Primer release en que puede restaurarse el módulo . . :   V3R7M0
   Conversión requerida. . . . . . . . . . . . . . . . :   *NO




                                                                        Más....
 Pulse Intro para continuar

 F3=Salir  F12=Cancelar
```

Figura 27. Tercera pantalla Visualizar información de módulo

La solicitud *Habilitar recogida de rendimiento* muestra que el módulo se ha creado con un código de medida de rendimiento solamente para la entrada al punto de entrada del programa y la salida del mismo. Las solicitudes de compatibilidad de módulo muestran el release y la versión del sistema operativo con el que el módulo es compatible.

```
                          Cambiar módulo (CHGMOD)

Teclee opciones, pulse Intro.

Módulo . . . . . > COPYPROC      Nombre, generic*, *ALL
  Biblioteca . . . . . >   TESTLIB     Nombre, *USRLIBL, *LIBL
Optimizar módulo . . . . . *NONE_        *SAME, *FULL, *BASIC. . .
Eliminar info observable . . . . *DBGDTA       *SAME, *ALL, *NONE...
Habilitar recogida de rendimiento:
  Nivel de recogida. . . . . *PEP          *SAME, *NONE, *PEP, *FULL...
  Procedimientos . . . . . *ALLPRC, *NONLEAF
Forzar recreación de módulo. . . *NO_          *NO, *YES
Texto 'descripción'. . . . . 'PG - COPY en ejemplo instrucción PROCESS
      '
       _____________________________

                                                                    Final
F3=Salir  F4=Solicitud F5=Renovar  F12=Cancelar   F13=Utilización de pantalla
F24=Más teclas
```

Figura 28. Pantalla de solicitud del mandato CHGMOD

Cambiar el objeto de módulo a un nivel de optimización inferior permite visualizar y posiblemente cambiar los valores de las variables durante la depuración.

Repita el proceso para cualquier otro objeto de módulo cuyo nivel de optimización desee cambiar. No importa si se modifica uno o varios objetos de módulo en el mismo programa; el tiempo de creación del programa es el mismo ya que todas las importaciones se resuelven cuando el sistema las encuentra.

Cuando haya terminado de modificar el nivel de optimización para los objetos de módulo de un objeto de programa, cree el objeto de programa de nuevo utilizando el mandato CRTPGM o actualice el objeto de programa existente con los objetos de módulo nuevos utilizando el mandato UPDPGM.

Eliminación de observabilidad de módulo

La observabilidad de módulo hace referencia a dos tipos de datos que pueden almacenarse con un objeto de módulo. Estos datos permiten depurar el objeto de módulo o cambiarlo sin tener que crearlo de nuevo. Una vez creado el objeto de módulo, sólo se pueden eliminar estos datos. Una vez eliminados los datos, debe crear el objeto de módulo de nuevo para sustituirlo. Los dos tipos de datos son:

Datos de creación

Representados mediante el valor *CRTDTA. Estos datos son necesarios para convertir el código a instrucciones de máquina. La plantilla de instrucciones de máquina (MI) se incluye con el objeto de módulo cuando éste se crea utilizando el mandato CRTCBLMOD. La plantilla MI continúa allí hasta que se elimine de

forma explícita. El objeto de módulo debe contener estos datos para poder cambiar el nivel de optimización.

Datos de depuración
Representados mediante el valor *DBGDTA. Estos datos son necesarios para que el objeto de módulo pueda depurarse. Los datos de depuración se incluyen con el objeto de módulo cuando éste se crea utilizando el mandato CRTCBLMOD. El tipo y cantidad de datos de depuración viene determinado por el parámetro DBGVIEW.

Eliminar toda observabilidad reduce el objeto de módulo a su tamaño mínimo (con compresión). Una vez eliminada la observabilidad, no se puede modificar el objeto de módulo de ninguna forma a menos que vuelva a crearlo.

Para eliminar cualquier tipo de datos del objeto de módulo, eliminar ambos tipos o no eliminar ninguno, utilice el mandato Trabajar con módulos (WRKMOD). Teclee WRKMOD en la línea de mandatos y aparecerá la pantalla Trabajar con módulos. Seleccione la opción 5 (Visualizar) para ver los valores de atributo que deben modificarse. La pantalla Visualizar información de módulo aparece en la Figura 25 en la página 94.

Primero, compruebe el valor del parámetro *plantilla instrucciones máquina incluida*. Si es *YES, los datos de creación existen y pueden eliminarse. Si el valor es *NO, no existen datos de creación para eliminar. El objeto de módulo no puede convertirse de nuevo a menos que vuelva a crearlo e incluya la plantilla de instrucciones de máquina.

A continuación, pulse la tecla Giro abajo, para ver más información sobre el objeto de módulo. Compruebe el valor del parámetro *Datos de depuración*. Si es *YES, los datos de depuración existen y el objeto de módulo puede depurarse. Si es *NO, los datos de depuración no existen y el objeto de módulo no puede depurarse a menos que vuelva a crearlo e incluya los datos de depuración. Seleccione la opción 2 (Cambiar) para el objeto de módulo cuya observabilidad desea cambiar. Aparecerá la pantalla de solicitud del mandato CHGMOD. Teclee sobre el valor especificado para la solicitud *Eliminar info observable*.

Para asegurar que el objeto de módulo se crea de nuevo, utilice el parámetro *Forzar recreación de módulo*. Cuando el nivel de optimización se modifica, el objeto de módulo se vuelve a crear siempre si los datos de creación no se han eliminado. Si desea que el objeto de programa se convierta de nuevo eliminando los datos de depuración y sin cambiar el nivel de optimización, debe cambiar el valor del parámetro *Forzar recreación de módulo* a *YES.

Repita el proceso para cualquier otro objeto de módulo que desee cambiar. No importa si se modifica uno o varios objetos de módulo en el mismo programa; el tiempo de creación del programa es el mismo ya que todas las importaciones se resuelven cuando el sistema las encuentra.

Cuando haya terminado de modificar el nivel de optimización para los objetos de módulo de un objeto de programa, cree el objeto de programa de nuevo utilizando el mandato CRTPGM o actualice el objeto de programa existente con los objetos de módulo nuevos utilizando el mandato UPDPGM.

Habilitación de recogida de rendimiento

A continuación se indican las opciones que pueden especificarse cuando se solicitan datos estadísticos sobre el rendimiento para una unidad de compilación.

Niveles de recogida

Los niveles de recogida son:

***PEP** Sólo se reunen estadísticas de rendimiento en la entrada y salida del procedimiento de entrada de programa. Seleccione este valor si desea tener la infomación general sobre el rendimiento de una aplicación. Este soporte es equivalente al soporte que la herramienta TPST proporcionaba anteriormente. Se trata del valor por omisión.

***ENTRYEXIT** Se reúnen estadísticas sobre el rendimiento en la entrada y salida de todos los procedimientos del programa. Incluye la rutina del PEP del programa.

Esta opción le resultará de utilidad si desea capturar información sobre todas las rutinas. Utilice esta opción si sabe que todos los programas a los que la aplicación ha llamado se han compilado con la opción *PEP, *ENTRYEXIT o *FULL. En caso contrario, si la aplicación llama a otros programas en los que no se ha habilitado la medida de rendimiento, la herramienta de rendimiento cargará a la aplicación la utilización de recursos de estos. Esto dificultaría la determinación del lugar en que los recursos se utilizan realmente.

***FULL** Se reúnen estadísticas sobre el rendimiento en la entrada y salida de todos los procedimientos. Además, también se reúnen estadísticas antes y después de cada llamada a un procedimiento externo.

Utilice esta opción si piensa que la aplicación llamará a otros programas que no se hayan compilado con *PEP, *ENTRYEXIT o *FULL. Esta opción permite que las herramientas de rendimiento distingan entre los recursos utilizados por la aplicación y los utilizados por los programas a los que llama (incluso aunque en estos programas no se haya habilitado la medida de rendimiento). Es la opción más cara pero le permite el análisis selectivo de varios programas dentro de una aplicación.

Procedimientos

Los valores para el nivel de procedimiento son:

***ALLPRC** Se recogen datos de rendimiento para todos los procedimientos.

***NONLEAF** Se recogen datos sobre el rendimiento de procedimientos que no son procedimientos de hoja y del PEP.

Nota: *NOLEAF no afecta a los programas ILE COBOL/400.

Capítulo 5. Creación de un programa de servicio

Un programa de servicio es un tipo especial de objeto del sistema que proporciona un conjunto de servicios a los objetos de programa ILE que se enlazan a él.

Este capítulo describe:

- cómo pueden utilizarse los programas de servicio
- cómo escribir mandatos de lenguaje enlazador para un programa de servicio
- cómo crear un programa de servicio utilizando el mandato CRTSRVPGM
- cómo llamar y compartir datos con un programa de servicio.

Definición de un programa de servicio

Un **programa de servicio** es un conjunto de procedimientos ejecutables y elementos de datos disponibles que utilizan otros programas de servicio y objetos de programa ILE. Los programas de servicio son objetos del sistema de tipo *SRVPGM y se les especifica un nombre cuando el programa de servicio se crea.

Utilice el mandato Crear programa de servicio (CRTSRVPGM) para crear un programa de servicio. Un programa de servicio se parece a un objeto de programa en que ambos constan de uno o más objetos de módulo que se han enlazado entre sí para formar un objeto ejecutable. Sin embargo, un programa de servicio se diferencia en que no tiene PEP. Y como no tiene PEP no puede llamarse ni cancelarse. En lugar de un PEP, el programa de servicio puede exportar procedimientos. Sólo los procedimientos exportados desde un programa de servicio pueden llamarse mediante llamadas de procedimiento estáticas realizadas desde el exterior del programa de servicio. Las exportaciones de programas de servicio se definen utilizando el lenguaje enlazador.

Consulte el manual *ILE Conceptos* para obtener más información sobre programas de servicio.

Utilización de programas de servicio

Normalmente, los programas de servicio se utilizan para rutinas comunes que se llaman con frecuencia desde dentro de una aplicación y entre aplicaciones. Por ejemplo, el compilador ILE COBOL/400 utiliza programas de servicio para proporcionar servicios de ejecución como, por ejemplo, funciones matemáticas y rutinas de entrada/salida. Los programas de servicio permiten volver a utilizar los programas fuente, simplifican el mantenimiento y reducen los requisitos de almacenamiento. En muchos aspectos, los programas de servicio se parecen a las bibliotecas de subrutinas o a las bibliotecas de procedimiento.

Se puede actualizar un programa de servicio sin tener que volver a crear el resto de objetos de programa o programas de servicio que utilizan el programa de servicio actualizado siempre que la interfaz no se modifique o tan sólo se modifique de forma compatible ascendente. El usuario es quien controla si los cambios son compatibles con el soporte existente proporcionado por el programa de servicio. Para realizar cambios compatibles en un programa de servicio, debería añadir nombres de procedimientos o nombres de datos nuevos al final de la lista de exportaciones y retener la misma signatura que antes.

Escritura de mandatos del lenguaje enlazador para un programa de servicio ILE COBOL/400

El **lenguaje enlazador** permite definir la lista de elementos de datos y nombres de procedimiento que pueden exportarse desde un programa de servicio.Para obtener una descripción completa del lenguaje enlazador y los mandatos del lenguaje enlazador, consulte el manual *ILE Conceptos*.

Las **signaturas** se generan a partir de elementos de datos y nombres de procedimientos y a partir del orden en que se especifican en el lenguaje enlazador.Una signatura es un valor que identifica la interfaz a la que da soporte el programa de servicio. Se puede especificar de forma explícita la signatura con el parámetro SIGNATURE en el lenguaje enlazador.

Para los programas de servicio creados a partir de programas fuente ILE COBOL/400, los elementos de lenguaje siguientes son exportaciones de módulo que pueden incluirse en la lista de exportaciones del lenguaje enlazador:

- El nombre que se indica en el párrafo PROGRAM-ID del programa ILE COBOL/400 más externo de una unidad de compilación.
- El nombre generado por el compilador ILE COBOL/400 derivado del nombre del párrafo PROGRAM-ID del programa ILE COBOL/400 más externo de una unidad de compilación siempre que el programa no tenga el atributo INITIAL. El nombre se forma añadiendo el sufijo `_reset` al nombre del párrafo PROGRAM-ID. Este nombre tiene que incluirse en la lista de exportaciones únicamente si el programa ILE COBOL/400 del programa de servicio tiene que cancelarse.

Utilización del mandato Crear programa de servicio (CRTSRVPGM)

Los programas de servicio se crean utilizando el mandato Crear programa de servicio (CRTSRVPGM). Cualquier objeto de módulo ILE puede enlazarse en un programa de servicio.Los objetos de módulo deben existir para poder crear un programa de servicio con ellos. Se pueden crear objetos de módulo a partir de programas fuente ILE COBOL/400 utilizando el mandato CRTCBLMOD. Consulte el apartado “Utilización del mandato Crear módulo COBOL (CRTCBLMOD) ” en la página 30 para obtener una descripción de cómo crear objetos de módulo utilizando el mandato CRTCBLMOD.

La Tabla 3 en la página 103 lista los parámetros de CRTSRVPGM y sus valores por omisión. Para obtener una descripción completa del mandato CRTSRVPGM y sus parámetros, consulte el manual *CL Reference*.

Tabla 3. Parámetros del mandato CRTSRVPGM y sus valores por omisión

Grupo parámetros	Parámetro(Valor por omisión)
Identificación	SRVPGM(*nombre-biblioteca*/*nombre-programa-servicio*) MODULE(*SRVPGM)
Acceso programa	EXPORT(*SRCFILE) SRCFILE(*LIBL/QSRVSRC) SRCMBR(*SRVPGM)
Enlace	BNDSRVPGM(*NONE) BNDDIR(*NONE)
Ejecución	ACTGRP(*CALLER)
Varios	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF) DETAIL(*NONE) REPLACE(*YES) AUT(*LIBCRTAUT) ALWUPD(*YES) ALWRINZ(*NO) TEXT(*BLANK) USRPRF(*USER) TGTRLS(*CURRENT)

Ejemplo de creación de un programa de servicio

Este ejemplo muestra cómo utilizar el lenguaje enlazador para crear un programa de servicio para realizar cálculos económicos.

Supongamos que los siguientes programas fuente ILE COBOL/400 contienen los objetos de módulo que constituirán el programa de servicio.

- RATE

  Calcula el tipo de interés, conociendo el importe del préstamo, el plazo y el importe de la cuota.

- AMOUNT

  Calcula el importe del préstamo, conociendo el tipo de interés, el plazo y el importe de la cuota.

- PAYMENT

  Calcula el importe de la cuota, conociendo el tipo de interés, el plazo y el importe del préstamo.

- TERM

  Calcula el plazo de la cuota conociendo el tipo de interés, el importe del préstamo y el importe de la cuota.

1. El lenguaje enlazador para el programa de servicio que hace disponibles los programas ILE COBOL/400 RATE, AMOUNT, PAYMENT y TERM tiene el siguiente aspecto:

```
FILE: MYLIB/QSRVSRC  MEMBER: FINANCIAL

STRPGMEXP  PGMLVL(*CURRENT)
  EXPORT SYMBOL('TERM')
  EXPORT SYMBOL('RATE')
  EXPORT SYMBOL('AMOUNT')
  EXPORT SYMBOL('PAYMENT')
ENDPGMEXP
```

Se puede utilizar el SEU para entrar instrucciones fuente de lenguaje enlazador. El comprobador de sintaxis de SEU solicitará y validará la entrada de lenguaje enlazador cuando se especifique el tipo fuente BND. Para arrancar una sesión de edición para entrar el fuente de lenguaje enlazador, teclee

```
STRSEU SRCFILE(MYLIB/QSRVSRC) SRCMBR(FINANCIAL)
TYPE(BND) OPTION(2)
```

y pulse Intro.

2. Compile los cuatro programas fuente ILE COBOL/400 en objetos de módulo utilizando el mandato CRTCBLMOD. Suponga que los objetos de módulo también tienen los nombres RATE, AMOUNT, PAYMENT y TERM.

   Para crear un programa de servicio se pueden ejecutar las instrucciones de enlazador necesarias con este mandato:

   ```
   CRTSRVPGM  SRVPGM(MYLIB/FINANCIAL)
              MODULE(MYLIB/TERM MYLIB/RATE MYLIB/AMOUNT MYLIB/PAYMENT)
              EXPORT(*SRCFILE)
              SRCFILE(MYLIB/QSRVSRC)
              SRCMBR(*SRVPGM)
   ```

   Notas:

   a. El archivo fuente QSRVSRC que se encuentra en la biblioteca MYLIB es el archivo que contiene el fuente del lenguaje enlazador.

   b. No es necesario un directorio de enlace ya que todos los objetos de módulo necesarios para crear el programa de servicio se han especificado con el parámetro MODULE.

   Podrá encontrar más ejemplos de la utilización del lenguaje enlazador y creación de programas de servicio en el manual *ILE Conceptos*.

Llamada a procedimientos ILE exportados en programas de servicio

Los procedimientos ILE exportados en programas de servicio sólo pueden llamarse desde un objeto de programa ILE o desde otro programa de servicio utilizando una llamada estática de procedimientos.

Se puede llamar un procedimiento ILE exportado en un programa de servicio desde un programa ILE COBOL/400 utilizando la instrucción CALL *literal* (donde *literal* es el nombre de un procedimiento ILE del programa de servicio). Consulte el apartado “Realización de llamadas estáticas de procedimiento utilizando CALL literal” en la página 187 para obtener información detallada sobre cómo escribir la instrucción CALL en el programa ILE COBOL/400 de forma que llame a un procedimiento ILE exportado en un programa de servicio.

Compartimiento de datos con programas de servicio

Los datos externos pueden compartirse entre objetos de módulo en un programa de servicio, entre programas de servicio, entre objetos de programa y entre objetos de programa y programas de servicio.

En el programa ILE COBOL/400, los elementos de datos que se van a compartir entre objetos de módulo diferentes deben describirse con la cláusula EXTERNAL en la Working Storage Section. Consulte el apartado “Compartimento de datos EXTERNAL” en la página 203 o consulte el apartado sobre la cláusula EXTERNAL del manual *ILE COBOL/400 Reference* para obtener más información sobre cómo se utilizan datos externos en un programa ILE COBOL/400.

Los archivos y datos declarados como EXTERNAL en un programa ILE COBOL/400 de un programa de servicio no pueden incluirse en la lista de exportaciones del lenguaje enlazador para el programa de servicio. Los datos y archivos declarados como EXTERNAL en un programa ILE COBOL/400 que esté fuera del programa de servicio pueden compartirse con un programa ILE COBOL/400 que está dentro del programa de servicio mediante la resolución en tiempo de activación de los datos EXTERNAL y de los archivos EXTERNAL. Este mismo mecanismo también permite compartir datos EXTERNAL y archivos EXTERNAL entre dos objetos de programa completamente separados que estén activados en el mismo grupo de activación.

Cancelación de un programa ILE COBOL/400 en un programa de servicio

Para cancelar un programa ILE COBOL/400 que forma parte de un programa de servicio desde fuera de ese programa de servicio, debe especificar CANCEL para el nombre de procedimiento del programa ILE COBOL/400 en la lista de exportaciones del lenguaje enlazador.

Capítulo 6. Ejecución de un programa ILE COBOL/400

Este capítulo proporciona la información necesaria para ejecutar el programa ILE COBOL/400.

Los métodos más habituales para ejecutar un programa ILE COBOL/400 son:

- Utilizar un mandato CALL de Lenguaje de control (CL)
- Utilizar una instrucción CALL de un Lenguaje de alto nivel (por ejemplo, la instrucción CALL del ILE COBOL/400)
- Utilizar un programa de aplicaciones dirigido por menús
- Emitir un mandato creado por el usuario.

Ejecución de un programa COBOL utilizando el mandato CALL de CL

Se puede utilizar el mandato CALL de CL para ejecutar un programa ILE COBOL/400. Se puede utilizar un mandato CALL de CL de forma interactiva, como parte de un trabajo de proceso por lotes, o incluirlo en un programa CL. A continuación se presenta un ejemplo de un mandato CALL de CL:

```
CALL nombre-programa
```

El objeto de programa especificado mediante nombre-programa debe existir en una biblioteca y esta biblioteca debe encontrarse en la lista de bibliotecas *LIBL. También se puede especificar la biblioteca explícitamente en el mandato CALL de CL de la forma siguiente:

```
CALL nombre-biblioteca/nombre-programa
```

Consulte el manual *CL Reference* para obtener más información sobre la utilización del mandato CALL de CL.

Cuando se ejecuta un trabajo de proceso por lotes que llama a un programa ILE COBOL/400 que utiliza el Formato 1 de la instrucción ACCEPT, los datos de entrada se toman de la corriente de trabajos. Estos datos deben colocarse inmediatamente después del CALL de CL para el programa ILE COBOL/400. Debe asegurarse de que el programa solicita (mediante varias instrucciones ACCEPT) la misma cantidad de datos que hay disponible. Consulte el apartado “Instrucción ACCEPT” del manual *ILE COBOL/400 Reference* para obtener más información.

Si se solicitan más datos que los disponibles, el mandato CL que sigue a los datos se tratará como datos de entrada. Si hay más datos disponibles que los solicitados, cada línea de datos adicional se trata como un mandato CL. En cada caso, pueden producirse resultados indeseables.

Cómo pasar parámetros a un programa ILE COBOL/400 mediante el mandato CALL de CL

Se utiliza la opción PARM del mandato CALL de CL para pasar parámetros al programa ILE COBOL/400 cuando se ejecuta.

```
CALL PGM(nombre-programa) PARM(parámetro-1 parámetro-2 parámetro-3)
```

Cada uno de los valores de parámetro sólo puede especificarse en una de las formas siguientes:

- constante de serie de caracteres
- constante numérica
- constante lógica
- constante de coma flotante de precisión doble
- variable de programa

Consulte el mandato CALL en el manual *CL Reference* o el apartado "Pasar parámetros entre programas" del manual *CL Programación* para obtener información completa sobre cómo se manejan los parámetros.

Ejecución de un programa ILE COBOL/400 utilizando la instrucción CALL de HLL

Se puede ejecutar un programa ILE COBOL/400 llamándolo desde otro programa HLL.

Se puede utilizar la instrucción CALL ILE COBOL/400 en un programa ILE COBOL/400 para llamar a otro programa ILE COBOL/400. Si la llamada de ILE COBOL/400 es una llamada dinámica de programas, el objeto de programa puede calificarse con una biblioteca utilizando la expresión IN LIBRARY. Por ejemplo, para llamar al objeto de programa PGMNAME de la biblioteca LIBNAME, debería especificar:

```
CALL "PGMNAME" IN LIBRARY "LIBNAME" USING variable1.
```

Sin la expresión IN LIBRARY, la búsqueda del objeto de programa se realiza en la lista de bibliotecas *LIBL. Consulte el apartado “Instrucción CALL” del manual *ILE COBOL/400 Reference* para obtener más información.

Para ejecutar un programa ILE COBOL/400 desde ILE C/400, utilice una llamada de función ILE C/400. El nombre de la función corresponde al nombre del programa ILE COBOL/400. Por omisión, esta llamada de función es una llamada estática de procedimientos. Para llevar a cabo una llamada dinámica de programas, utilice la instrucción directriz `#pragma linkage (PGMNAME, OS)`. `PGMNAME` representa el nombre del programa ILE COBOL/400 que desea ejecutar desde el programa ILE C/400. Una vez utilizada la instrucción directriz `#pragma linkage (PGMNAME, OS)` para indicar al compilador ILE C/400 que `PGMNAME` es un programa externo, se puede ejecutar el programa ILE COBOL/400 con una llamada de función ILE C/400. Para obtener más información, consulte el capítulo que trata sobre cómo escribir programas que llaman a otros programas del manual *ILE C/400 Programmer’s Guide*.

Para ejecutar un programa ILE COBOL/400 desde un programa ILE RPG/400, utilice el código de operación CALL para realizar una llamada dinámica de programas o el código de operación CALLB para realizar una llamada estática de procedimientos. El programa que se va a llamar se identifica especificando su nombre como la entrada Factor 2. Para obtener más información, consulte el capítulo sobre procedimientos y programas de llamada en el manual *ILE RPG/400 Guía del programador*.

Para ejecutar un programa ILE COBOL/400 desde VisualAge C++ for OS/400, utilice la llamada de función VisualAge C++ for OS/400. El nombre de la función corresponde al nombre del programa ILE COBOL/400.Para evitar que VisualAge C++ for OS/400 modifique internamente el nombre de la función, es decir, para

evitar que el nombre de función ILE C++ se mutile, debe realizar un prototipo de la llamada de función utilizando la palabra clave `extern`.Para llamar a un procedimiento ILE COBOL/400 que no devuelve nada y que toma un número binario de dos bytes, el prototipo VisualAge C++ for OS/400 sería:

```
extern "COBOL" void PGMNAME(short int);
```

Para llamar al mismo objeto de programa COBOL, se especificaría un enlace de "OS". El prototipo se convierte en:

```
extern "OS" void PGMNAME(short int);
```

Un enlace de "COBOL" en una llamada de función VisualAge C++ for OS/400 no sólo evita que el nombre de función se mutile sino que provoca que los argumentos que se han pasado al procedimiento ILE COBOL/400 se pasen BY REFERENCE. Si el procedimiento ILE COBOL/400 espera un parámetro BY VALUE, debería especificarse un enlace "C".

Ejecución de un programa ILE COBOL/400 desde una aplicación dirigida por menús

Otra forma de ejecutar un programa ILE COBOL/400 es desde una aplicación dirigida por menús. El usuario de la estación de trabajo selecciona una opción desde un menú, llamando al programa adecuado. La figura siguiente muestra un ejemplo de un menú de aplicación

```
                          MENÚ DEPARTAMENTO NÓMINAS


  1.  Consultar en archivo principal de empleados
  2.  Cambiar archivo principal de empleados
  3.  Añadir empleado nuevo
  4.  Volver


Opción:
```

Figura 29. Ejemplo de un menú de aplicación

El menú que aparece en esta figura normalmente se visualiza mediante un programa CL en el que cada opción llama a un programa COBOL separado.

La DDS para el archivo de pantalla del anterior MENÚ DEPARTAMENTO NÓMINA tiene el aspecto siguiente:

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* MENU PAYROLLD MENÚ DEPARTAMENTO NÓMINAS
     A
     A          R MENU                      TEXT('MENÚ DEPARTAMENTO NÓMINAS')
     A                                   1 29'MENÚ DEPARTAMENTO NÓMINAS'
     A                                   5  4'1.  Consultar en archivo principal de empleados'
     A                                   6  4'2.  Cambiar archivo principal de empleados'
     A                                   7  4'3.  Añadir empleado nuevo'
     A                                   8  4'4.  Volver'
     A                                  12  2'Opción:'
     A            RESP                  12 10VALUES(1 2 3 4)
     A                                       DSPATR(MDT)
```

Figura 30. Especificación de descripción de datos de un menú de aplicación

La Figura 30 muestra un ejemplo del programa CL para el menú de aplicación de la Figura 29.

```
            PGM /* PAYROLL Menú departamento nóminas*/
            DCLF FILE (PAYROLLD)
      START: SNDRCVF RCDFMT(MENU)
            IF (&RESP=1) THEN(CALL CBLINQ)
            /* Consultar */
                ELSE +
              IF (&RESP=2) THEN(CALL CBLCHG)
              /* Cambiar */
                ELSE +
                IF (&RESP=3) THEN(CALL CBLADD)
                /* Añadir */
                ELSE +
                  IF (&RESP=4) THEN(RETURN)
                  /* Volver */
            GOTO START
            ENDPGM
```

Figura 31. Ejemplo de un programa CL que llama a programas ILE COBOL/400.

Si el usuario entra 1, 2 ó 3 desde el menú de aplicación, el programa CL de la Figura 31 llama a los programas ILE COBOL/400 CBLINQ, CBLCHG o CBLADD respectivamente. Si el usuario entra 4 en el menú de la aplicación, el programa CL vuelve al programa que lo llamó.

Ejecución de un programa ILE COBOL/400 utilizando un mandato creado por el usuario

El usuario puede crear sus propios mandatos para ejecutar un programa ILE COBOL/400 utilizando una definición de mandato. Una **definición de mandato** es un objeto que contiene la definición de un mandato (incluyendo el nombre del mandato, las descripciones de los parámetros y la información de comprobación de validez) e identifica al programa que realiza la función que solicita el mandato. El tipo de objeto reconocido por el sistema es *CMD.

Por ejemplo, se puede crear un mandato, PAY, que llama a un programa, PAYROLL. PAYROLL es el nombre de un programa ILE COBOL/400 que se llama y se ejecuta. Se puede entrar el mandato de forma interactiva o en un trabajo de proceso por lotes. Consulte el manual *CL Programación* para obtener más información sobre la utilización de la definición de mandato.

Finalización de un programa ILE COBOL/400

Cuando un programa ILE COBOL/400 termina de formal normal, el sistema devuelve el control al llamador. El llamador podría ser un usuario de una estación de trabajo, un programa CL (como, por ejemplo, el programa de manejo de menús) u otro programa HLL.

Si un programa ILE COBOL/400 termina anormalmente durante la ejecución, se emite el mensaje de escape CEE9901

Error de aplicación. id-mensaje no supervisado por nombre-programa en instrucción número-instrucción, instrucción número-instrucción.

al llamador de la unidad de ejecución. Un programa CL puede supervisar esta excepción utilizando el mandato Supervisar mensajes (MONMSG). Consulte el manual *CL Reference* para obtener más información sobre los mandatos de lenguaje de control.

Si un programa terminar por una causa que no es una de las siguientes:

- La utilización de la instrucción STOP RUN
- La utilización de la instrucción GOBACK en el programa principal
- La utilización de la instrucción EXIT-PROGRAM AND CONTINUE RUN UNIT en el programa principal
- La desactivación al final del programa,

el atributo de trabajo RTNCDE se establece en 2.

Consulte los mandatos RTVJOBA y DSPJOB del manual *CL Programación* para obtener más información sobre los códigos de retorno.

Respuesta a mensajes de consulta durante la ejecución

Cuando se ejecuta un programa ILE COBOL/400, es posible que se generen mensajes de consulta de ejecución. Estos mensajes requieren una respuesta para que el programa continúe la ejecución.

Los mensajes de consulta se pueden añadir a una lista de respuestas del sistema para proporcionar respuestas automáticas a los mensajes. Las respuestas para estos mensajes pueden especificarse individualmente o de forma general. Este método de respuesta a mensajes de consulta es especialmente adecuado para programas de proceso por lotes que, de otro modo, necesitarían un operador que emitiera las respuestas.

Los siguientes mensajes de consulta ILE COBOL/400 se pueden añadir a la lista de respuestas del sistema:

LNR7200
LNR7201
LNR7203
LNR7204
LNR7205
LNR7206
LNR7207
LNR7208
LNR7209
LNR7210
LNR7211
LNR7212
LNR7213
LNR7214
LNR7604.

La lista de respuestas sólo se utiliza cuando un trabajo que tiene el atributo Respuesta de mensajes de consulta (INQMSGRPY) especificado como `INQMSGRPY(*SYSRPYL)`, envía un mensaje de consulta.

El parámetro INQMSGRPY se produce en los mandatos CL siguientes:

- Cambiar trabajo (CHGJOB)
- Cambiar descripción de trabajo (CHGJOBD)
- Crear descripción de trabajo (CRTJOBD)
- Someter trabajo (SBMJOB).

Se puede seleccionar una de las cuatro modalidades de respuesta especificando uno de los valores siguientes para el parámetro INQMSGRPY:

SAME — No se realiza ninguna modificación en la forma en que las respuestas se envían a los mensajes de consulta.

RQD — Todos los mensajes de consulta precisan una respuesta por parte del receptor de los mensajes de consulta.

DFT — Se emite una respuesta por omisión.

SYSRPYL — Se comprueba si en las lista de respuestas del sistema existe una entrada de la lista de respuestas coincidente. Si se produce una coincidencia, se utiliza el valor de respuesta de esa entrada. Si no existe ninguna entrada para ese mensaje de consulta, es necesario proporcionar una respuesta.

Se puede utilizar el mandato Añadir entrada en lista de respuestas (ADDRPYLE) para añadir entradas a la lista de respuestas del sistema, o el mandato Trabajar con entradas de lista de respuestas (WRKRPYLE) para modificar o eliminar entradas de la lista de respuestas del sistema. Consulte el manual *CL Reference* para obtener detalles sobre los mandatos ADDRPYLE y WRKRPYLE. También se puede responder a los mensajes de consulta de ejecución con un manejador de errores definido por el usuario. Para obtener más información sobre las API de manejo de errores, consulte el manual System API Reference.

Capítulo 7. Depuración de un programa

La depuración permite detectar, diagnosticar y eliminar errores de un programa. Puede depurar los programas OPM y ILE COBOL/400 utilizando el depurador fuente ILE.

Este capítulo describe cómo utilizar el depurador fuente ILE para:

- Preparar el programa ILE COBOL/400 para la depuración
- Arrancar una sesión de depuración
- Añadir y eliminar programas de una sesión de depuración
- Visualizar el fuente del programa desde una sesión de depuración
- Establecer y eliminar puntos de interrupción condicionales e incondicionales.
- Establecer y eliminar condiciones de observación
- Ejecutar un programa por pasos
- Visualizar el valor de variables, registros, elementos de grupo y matrices
- Modificar el valor de variables
- Modificar el ámbito de referencia
- Equiparar un nombre abreviado a una variable, una expresión o un mandato de depuración.

Mientras depure y pruebe los programas, asegúrese de que la lista de bibliotecas se ha modificado para dirigir los programas a una biblioteca de prueba que contenga datos de prueba de tal forma que los datos reales existentes no se vean afectados.

Puede evitar que los archivos de base de datos de las bibliotecas de producción se modifiquen de forma no intencionada utilizando uno de los siguientes mandatos CL:

- Utilice el mandato Arrancar depuración (STRDBG) y especifique el parámetro UPDPROD(*NO)
- Utilice el mandato Cambiar depuración (CHGDBG) y especifique el valor *NO del parámetro UPDPROD
- Utilice el mandato de depuración SET de la pantalla Visualizar fuente del módulo. La sintaxis para impedir que se produzcan modificaciones en el archivo sería la siguiente:

  ```
  SET UPDPROD NO
  ```

  que puede abreviarse a

  ```
  SET U N
  ```

Consulte el manual *CL Reference* para obtener más información.

Consulte el capítulo sobre depuración del manual *ILE Conceptos* para obtener más información sobre el depurador fuente ILE (incluyendo la autorización necesaria para depurar un objeto de programa o un programa de servicio y los efectos de los niveles de optimización).

El depurador fuente ILE

El depurador fuente ILE se utiliza para detectar errores en los programas de servicio y objetos de programa y eliminarlos. Si utiliza los mandatos de depuración con cualquier programa ILE que contenga datos de depuración podrá:

- Depurar aplicaciones de lenguaje ILE COBOL/400 o ILE mixtas.
- Supervisar el flujo de un programa utilizando los mandatos de depuración mientras se ejecuta el programa.
- Visualizar el fuente del programa o modificar la vista de depuración
- Establecer y eliminar puntos de interrupción condicionales e incondicionales.
- Establecer y eliminar condiciones de observación
- Ejecutar por pasos un número determinado de instrucciones
- Visualizar o modificar los valores de variables, registros, elementos de grupo y matrices.

Nota: El depurador fuente ILE no da soporte a la cláusula COLLATING SEQUENCE ILE COBOL/400. Si utiliza esta cláusula en el programa ILE COBOL/400 para especificar un orden de clasificación propio, el depurador fuente ILE no utilizará este orden.

Si un programa se detiene debido a un punto de interrupción o a un mandato de paso, la vista del objeto de módulo pertinente aparece en la pantalla en el punto en que se ha detenido el programa. En este punto se pueden entrar más mandatos de depuración.

Para poder utilizar el depurador fuente, debe especificar el parámetro DBGVIEW con un valor diferente de *NONE cuando cree el objeto de módulo o el objeto de programa utilizando el mandato CRTCBLMOD o CRTBNDCBL. Una vez haya arrancado el depurador, puede establecer puntos de interrupción u otras opciones de depuración fuente ILE y a continuación deberá ejecutar el programa.

Mandatos de depuración

Con el depurador fuente ILE pueden utilizarse muchos mandatos de depuración.Los mandatos de depuración y sus parámetros se entran en la línea de mandatos de depuración que se visualiza en la parte inferior de las pantallas Visualizar fuente del módulo y Evaluar expresión. Estos mandatos se pueden entrar en letras mayúsculas, minúsculas o mixtas. Consulte el manual *ILE Conceptos* para obtener más detalles sobre los mandatos de depuración.

Nota: Los mandatos de depuración que se entran en la línea de mandatos de depuración no son mandatos CL.

La Tabla 4 en la página 115 resume estos mandatos de depuración. La ayuda en línea disponible para el depurador fuente ILE describe los mandatos de depuración y explica las abreviaturas que se permiten.

Tabla 4. Mandatos del depurador fuente ILE

Mandato de depuración	Descripción
ATTR	Permite visualizar los atributos de una variable. Los atributos son el tamaño y el tipo de la variable como se registra en la tabla de símbolos de depuración. Consulte la Tabla 5 en la página 116 para obtener una lista de atributos y de sus equivalencias en ILE COBOL/400. Estos atributos no son los mismos que los definidos por ILE COBOL/400.
BREAK	Permite entrar un punto de interrupción condicional o incondicional en una posición del programa que se está probando. Utilice BREAK *posición* WHEN *expresión* para entrar un punto de interrupción condicional.
CLEAR	Permite eliminar puntos de interrupción condicionales e incondicionales o eliminar una o todas las condiciones de observación activas.
DISPLAY	Permite visualizar los nombres y definiciones que se han asignado utilizando el mandato EQUATE. También permite visualizar un módulo fuente diferente del que aparece actualmente en la pantalla Visualizar fuente del módulo. El objeto de módulo debe existir en el objeto de programa actual.
EQUATE	Permite asignar una expresión, una variable o un mandato de depuración a un nombre para su utilización abreviada.
EVAL	Permite visualizar o modificar el valor de una variable o visualizar los valores de expresiones, registros, elementos de grupo o matrices.
QUAL	Permite definir el ámbito de las variables que aparecen en los mandatos EVAL posteriores.
SET	Permite modificar las opciones de depuración, como por ejemplo la posibilidad de actualizar archivos de producción o de habilitar el soporte de depuración fuente OPM.
STEP	Permite ejecutar una o más instrucciones del programa que se está depurando.
WATCH	Permite solicitar un punto de interrupción cuando se cambia el valor actual del contenido de una ubicación de almacenamiento específica.
FIND	Busca avanzando en el módulo que se visualiza actualmente un texto, una serie o un número de línea determinado.
UP	Mueve la ventana del fuente que se visualiza hacia el principio de la vista tanto como se indique.
DOWN	Mueve la ventana del fuente que se visualiza hacia el final de la vista tanto como se indique.
LEFT	Mueve la ventana del fuente que se visualiza hacia la izquierda el número de caracteres que se haya entrado.
RIGHT	Mueve la ventana del fuente que se visualiza hacia la derecha el número de caracteres que se haya entrado.
TOP	Coloca la vista para que se muestre la primera línea.
BOTTOM	Coloca la vista para que se muestre la última línea.
NEXT	Coloca la vista en el siguiente punto de interrupción del fuente que se visualiza actualmente.
PREVIOUS	Coloca la vista en el punto de interrupción anterior del fuente que se visualiza actualmente.
HELP	Muestra la información de ayuda en línea para los mandatos disponibles del depurador fuente.

Atributos de variables

El depurador fuente ILE no describe los atributos de las variables de la misma forma que ILE COBOL/400. La Tabla 5 muestra la equivalencia entre los atributos de variables descritas por el depurador fuente ILE y por las categorías de datos ILE COBOL/400.

Tabla 5. Equivalencia entre atributos de variables del depurador fuente ILE y categorías de datos ILE COBOL/400

Atributos de variables del depurador fuente ILE	Categorías de datos ILE COBOL/400
FIXED LENGTH STRING	Alfabético Alfanumérico Alfanumérico editado Numérico editado Coma flotante externo
GRAPHIC	DBCS DBCS-editado
CHAR	Booleano
BINARY	Binario
ZONED(2,0)	Decimal con zona
PACKED(2,0)	Decimal empaquetado
PTR	Puntero Puntero de procedimiento
Real	Coma flotante interno

Preparación de un objeto de programa para una sesión de depuración

Para poder utilizar el depurador fuente ILE, se tiene que utilizar el mandato CRTCBLMOD o el mandato CRTBNDCBL especificando el parámetro DBGVIEW.

Se puede crear una de las tres vistas para cada objeto de módulo ILE COBOL/400 que desee depurar. Estas son:

- Vista de listado
- Vista fuente
- Vista de instrucción

Utilización de una vista de listado

Una vista de listado es parecida a la parte del listado fuente del listado de compilación o archivo de spool que genera el compilador ILE COBOL/400.

Para poder depurar un objeto de módulo ILE COBOL/400 utilizando una vista de listado, debe utilizar el valor *LIST o *ALL en el parámetro DBGVIEW para el mandato CRTCBLMOD o CRTBNDCBL cuando se crea el objeto de módulo o el objeto de programa.

A continuación se indica un método para crear una vista de listado:

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('programa CBL') DBGVIEW(*LIST)
```

Cuando se genera una vista de listado especificando el parámetro DBGVIEW(*LIST) en el mandato CRTCBLMOD o en el mandato CRTBNDCBL, el tamaño del objeto de módulo que se crea aumenta debido a la vista de listado. La vista de listado proporciona todas las ampliaciones (por ejemplo, las instrucciones COPY y REPLACE) que realiza el compilador ILE COBOL/400 cuando crea el objeto de módulo o el objeto de programa. La vista de listado existe independientemente del miembro fuente. El miembro fuente puede modificarse o suprimirse sin afectar a la vista de listado.

Si el miembro fuente contiene varias unidades de compilación, la vista de listado contendrá los listados fuente de todas las unidades de compilación, aunque sólo una de ellas se depure. Sin embargo, cualquier mandato de depuración que se emita desde la pantalla Visualizar fuente del módulo se aplicará únicamente a la unidad de compilación que pueda depurarse.

Utilización de una vista fuente

Las vistas fuente contienen referencias a las instrucciones fuente del miembro fuente.

Para utilizar la vista fuente con el depurador fuente ILE, el compilador ILE COBOL/400 crea referencias al miembro fuente mientras se está creando el objeto de módulo (*MODULE).

Nota: El objeto de módulo se crea utilizando referencias a ubicaciones de las instrucciones fuente en el miembro fuente raíz en lugar de copiar las instrucciones fuente en la vista. Por consiguiente, no debería modificar, renombrar ni mover miembros fuente raíz entre la creación del módulo y la depuración del módulo que se ha creado a partir de estos miembros.

Para depurar un objeto de módulo ILE COBOL/400 utilizando una vista fuente, utilice los valores *SOURCE o *ALL del parámetro DBGVIEW para el mandato CRTCBLMOD o para el mandato CRTBNDCBL.

A continuación se indica un método para crear una vista fuente:

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('programa CBL') DBGVIEW(*SOURCE)
```

Cuando se genera la vista fuente especificando DBGVIEW(*SOURCE) en el mandato CRTCBLMOD o en el mandato CRTBNDCBL, el tamaño del objeto de módulo que se ha creado aumenta debido a la vista fuente, pero el tamaño es menor que aquél generado con la vista de listado. El tamaño del objeto de módulo generado será el mismo que para la vista de instrucción. La vista fuente no proporciona ninguna de las ampliaciones que el compilador ILE COBOL/400 realiza cuando crea un objeto de módulo o un objeto de programa. La vista fuente depende de la existencia no modificada del miembro fuente. Cualquier cambio realizado en el miembro fuente afectará a la vista fuente.

Si el miembro fuente contiene varias unidades de compilación, la vista fuente contendrá el código fuente de todas las unidades de compilación, aunque sólo una de ellas pueda depurarse. Sin embargo, los mandatos de depuración que se emitan desde la pantalla Visualizar fuente del módulo se aplicará únicamente a la unidad de compilación que se esté depurando.

Utilización de una vista de instrucción

Las vista de instrucción no contienen instrucciones fuente. Contienen números de línea y números de instrucción. Para depurar un objeto de módulo ILE COBOL/400 utilizando una vista de instrucción, es necesario una copia impresa del listado del compilador.

Nota: No aparece ningún código fuente en la pantalla Visualizar fuente del módulo cuando se utiliza una vista de instrucción para depurar un objeto de módulo ILE COBOL/400.

Para depurar un objeto de módulo ILE COBOL/400 utilizando una vista de instrucción, utilice los valores *STMT, *SOURCE, *LIST o *ALL en el parámetro DBGVIEW para el mandato CRTCBLMOD o para el mandato CRTBNDCBL cuando cree el módulo.

A continuación se indica un método para crear una vista de instrucción:

```
CRTCBLMOD MODULE(MYLIB/xxxxxxxx)
SRCFILE(MYLIB/QCBLLESRC) SRCMBR(xxxxxxxx)
TEXT('programa CBL') DBGVIEW(*STMT)
```

Cuando se genera la vista de instrucción especificando DBGVIEW(*STMT) en los mandatos CRTCBLMOD o CRTBNDCBL, el tamaño del objeto de módulo creado aumenta mínimamente debido a la vista de instrucción. El tamaño del objeto de módulo creado es menor que aquellos generados con la vista de listado o la vista fuente. La vista de instrucción minimiza el tamaño del objeto de módulo creado a la vez que sigue siendo posible algún tipo de depuración. La vista de instrucción sólo proporciona la tabla de símbolos y una correlación entre números de instrucciones y números de línea de depuración.

Arranque del depurador fuente ILE

Una vez creada la vista de depuración, se puede empezar la depuración de la aplicación.

Para arrancar el depurador fuente ILE, utilice el mandato Arrancar depurador (STRDBG). Una vez arrancado, el depurador permanecerá activo hasta que se entre el mandato Finalizar depurador (ENDDBG). Los atributos de la modalidad de depuración se pueden modificar más adelante en el trabajo utilizando el mandato Cambiar depuración (CHGDBG).

La Tabla 6 en la página 119 lista los parámetros y los valores por omisión para el mandato STRDBG y para el mandato CHGDBG. El mandato ENDDBG no tiene ningún parámetro asociado. Para obtener una descripción completa de los mandatos STRDBG, CHGDBG ENDDBG y sus parámetros, consulte el manual *CL Reference*.

Tabla 6. Parámetros para los mandatos STRDBG y CHGDBG y sus valores por omisión

Grupo parámetros	Mandato STRDBG Parámetro(Valor por omisión)	Mandato CHGDBG Parámetro(Valor por omisión)
Identificación	PGM(*NONE) DFTPGM(*PGM)	DFTPGM(*SAME)
Rastreo	MAXTRC(200) TRCFULL(*STOPTRC)	MAXTRC(*SAME) TRCFULL(*SAME)
Varios	UPDPROD(*NO) OPMSRC(*NO) DSPMODSRC(*PGMDEP) SRCDBGPGM(*SYSDFT) UNMONPGM(*NONE)	UPDPROD(*SAME) OPMSRC(*SAME)

Nota: El rastreo sólo se aplica a programas OPM y no se aplica a programas de servicio ni a programas ILE.

Inicialmente, puede añadir hasta 20 objetos de programa a una sesión de depuración utilizando el parámetro Program (PGM) en el mandato STRDBG. (Según cómo se hayan compilado los programas OPM y de cuáles sean las definiciones del entorno de depuración, es posible que pueda depurarlos utilizando el depurador fuente ILE). Puede tratarse de cualquier combinación de programas ILE o OPM.

En el parámetro PGM del mandato STRDBG sólo pueden especificarse objetos de programa; aquí no se pueden especificar programas de servicio. Los programas de servicio se pueden añadir a la sesión de depuración después de haberla arrancado.

Aparece el primer programa especificado en el mandato STRDBG si éste tiene datos de depuración y si el OPM, el parámetro OPMSRC está definido en *YES. Si se trata de un programa ILE, el primer módulo de entrada, sólo se muestra si tiene datos de depuración; en caso contrario, aparece el primer módulo enlazado al programa ILE que tenga datos de depuración.

Para depurar un programa OPM utilizando el depurador fuente, deben cumplirse las siguientes condiciones:

1. El programa OPM debe estar compilado con OPTION(*LSTDBG) u OPTION(*SRCDBG). (Se da soporte a tres lenguajes OPM: RPG, COBOL y CL. Los programas RPG y COBOL pueden compilarse con *LSTDBG o *SRCDBG, pero los programas CL deben compilarse con *SRCDBG).
2. El entorno de depuración de ILE debe estar definido para aceptar programas OPM. Para ello, especifique OPMSRC(*YES) en el mandato STRDBG. (El valor por omisión del sistema es OPMSRC(*NO)).

Si no se cumplen estas dos condiciones, deberá depurar el programa OPM con el depurador OPM del sistema.

Por ejemplo, para arrancar una sesión de depuración para el programa de depuración de muestra MYPGM1 y para un programa OPM de llamada, MYPGM2, teclee:

```
STRDBG PGM(TESTLIB/MYPGM1 MYLIB/MYPGM2) OPMSRC(*YES)
```

Nota: Para poder añadir un objeto de programa a una sesión de depuración debe tener autorización *CHANGE sobre éste.

Después de entrar el mandato STRDBG, aparece la pantalla Visualizar fuente del módulo. Si se especifica un conjunto de programas ILE y programas OPM que permite el uso del depurador en el mandato STRDBG, se muestra el primer programa que presenta datos de depuración. Si este primer programa es un programa ILE, se muestra el primer objeto de módulo enlazado al objeto de programa que presente datos de depuración como se observa en la Figura 32.

```
                          Visualizar fuente del módulo

Programa:  MYPGM1        Biblioteca: TESTLIB        Módulo:   MYPGM1
     1         IDENTIFICATION DIVISION.
     2         PROGRAM-ID.  MYPGM1.
     3        *
     4        * Es el programa principal que controla
     5        * el proceso de archivos externos.
     6        *
     7
     8         ENVIRONMENT DIVISION.
     9         INPUT-OUTPUT SECTION.
    10         FILE-CONTROL.
    11              SELECT EF1
    12                 ASSIGN TO DISK-EFILE1
    13                 FILE STATUS IS EFS1
    14                 ORGANIZATION IS SEQUENTIAL.
    15                                                                  Más....
Depuración... _____________________________________________________________
F3=Fin programa  F6=Añadir/Borrar punto int. F10=Paso  F11=Visualizar variable
F12=Reanudar     F13=Trabajar con punt.interr.módulo   F24=Más teclas
```

Figura 32. Arranque de una sesión de depuración

Establecimiento de opciones de depuración

Una vez haya arrancado una sesión de depuración, puede definir o modificar las siguientes opciones de depuración utilizando el mandato de depuración SET en la línea de mandatos:

- Si los archivos de la base de datos pueden actualizarse mientras el programa se está depurando. (Esta opción corresponde al parámetro UPDPROD del mandato STRDBG).
- Si las búsquedas de texto utilizando la opción FIND son sensibles a las mayúsculas y minúsculas.
- Si los programas OPM deben depurarse utilizando el depurador fuente ILE. (Esta opción corresponde al parámetro OPMSRC).

Al modificar las opciones de depuración utilizando el mandato de depuración SET sólo se afecta al valor del parámetro correspondiente especificado en el mandato STRDBG, si éste está definido. También puede utilizar el mandato Cambiar depuración (CHGDBG) para establecer las opciones de depuración.

Suponga que se halla en una sesión de depuración trabajando con un programa ILE y decide que también debería depurar un programa OPM que presenta datos

de depuración. Para que el depurador fuente ILE acepte los programas OPM, debe seguir los siguientes pasos:

1. Después de entrar STRDBG, si la pantalla actual *no* es la pantalla Visualizar fuente de módulo teclee:

   ```
   DSPMODSRC
   ```

2. Teclee:

   ```
   SET
   ```

   Aparece la pantalla Establecer opciones de depuración.

3. En esta pantalla, teclee `Y` (Sí) en el campo *soporte de depuración fuente OPM* y pulse la tecla Intro para volver a la pantalla Visualizar fuente de módulo.

A continuación puede añadir el programa OPM, utilizando la pantalla Trabajar con módulo o procesando una instrucción de llamada para este programa.

Ejecución de un objeto de programa en una sesión de depuración

Cuando la sesión de depuración ya se ha arrancado, se puede ejecutar un objeto de programa en la sesión de depuración, pulsando:

- F3 (Fin programa),
- F12 (Reanudar), o
- F21 (Línea de mandatos)

en la pantalla Visualizar fuente del módulo.A continuación, se llama al objeto de programa desde la línea de mandatos utilizando el mandato CL CALL.

Cuando se produce una excepción en un objeto de programa durante una sesión de depuración, la excepción se maneja con las rutinas de manejo de excepciones y errores especificadas para el objeto de programa. Si la excepción no se maneja con un manejador de excepciones antes de que se convierta en una comprobación de función, se invoca el depurador y aparece la pantalla Visualizar fuente del módulo. El objeto de módulo donde se ha producido la excepción se visualiza en la instrucción que provocó la excepción. Consulte el Capítulo 12, “Manejo de errores y excepciones ILE COBOL/400” en la página 261 para obtener más información sobre el manejo de excepciones y errores.

La ejecución de un programa se puede detener estableciendo puntos de interrupción o pulsando F3 (Fin programa) en la pantalla Visualizar fuente del módulo. Consulte el apartado “Establecimiento y eliminación de puntos de interrupción” en la página 126 para obtener más información acerca de cómo se establecen puntos de interrupción.

Adición de objetos de programa y programas de servicio a una sesión de depuración

Se pueden añadir más objetos de programa y programas de servicio a una sesión de depuración una vez arrancada.

Para añadir programas de servicio y **objetos de programa ILE** a una sesión de depuración, utilice la opción 1 (Añadir programa) y teclee el nombre del objeto de programa en la primera línea de la pantalla Trabajar con lista de módulos (vea la Figura 33 en la página 122). Se puede acceder a la pantalla Trabajar con lista de

módulos pulsando F14 (Trabajar con lista de módulos) en la pantalla Visualizar fuente del módulo. Para añadir un programa de servicio, cambie el tipo de programa por omisión de *PGM a *SRVPGM. No hay límite para el número de programas de servicio y objetos de programa ILE que pueden incluirse en una sesión de depuración en un momento determinado.

Para añadir **objetos de programa OPM** a una sesión de depuración, tiene dos opciones en función del valor especificado para OPMSRC. Si ha especificado OPMSRC(*YES) utilizando STRDBG, el mandato de depuración SET o CHGDBG, debe añadir un programa OPM utilizando la pantalla Trabajar con lista de módulos. (Observe que no aparecerá ningún nombre de módulo para un programa OPM). No hay límite para el número de programas OPM que pueden incluirse en una sesión de depuración en un momento determinado cuando se especifica OPMSRC(*YES). Si ha especificado OPMSRC(*NO), deberá utilizar el mandato Añadir programa (ADDPGM). Cuando se especifica OPMSRC(*NO), sólo puede haber 20 programas OPM en una sesión de depuración.

Nota: No es posible depurar un programa OPM con datos de depuración que provengan de una sesión de depuración ILE y OPM. Si un programa OPM ya se encuentra en una sesión de depuración OPM, debe eliminarlo en primer lugar de esta sesión antes de añadirlo a la sesión de depuración de ILE o entrar en él desde una instrucción de llamada. De forma similar, si desea depurarlo desde una sesión de depuración OPM, debe eliminarlo primero de una sesión de depuración ILE.

```
                          Trabajar con lista de módulos
                                                          Sistema:   AS400SYS
Teclee opciones, pulse Intro.
  1=Añadir programa 4= Eliminar programa 5= Visualizar fuente del módulo
  8=Trabajar con puntos de interrupción de módulos

                                                Tipo de
Opc      Programa/módulo    Biblioteca         programa
1         TEST_________     TESTLIB___         *PGM___
_         MYPGM1            TESTLIB            *PGM
_           MYPGM1                             *MODULE     Selected
_         USERDSP           DSPLIB             *SRVPGM
_           SAMPMDF                            *MODULE
_           GETUSER                            *MODULE




                                                                        Final
Mandato
===>_____________________________________________________________________
F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
```

Figura 33. Adición de un objeto de programa ILE a una sesión de depuración.

Cuando termine de añadir objetos de programa o programas de servicio a la sesión de depuración, pulse F3 (Salir) en la pantalla Trabajar con lista de módulos para volver a la pantalla Visualizar fuente del módulo.

Nota: Debe tener autorización *CHANGE sobre un programa para añadirlo a una sesión de depuración. Los programas de servicio ILE únicamente pueden

añadirse a una sesión de depuración utilizando la opción 1 de la pantalla Trabajar con lista de módulos. Los programas de servicio ILE no pueden especificarse en el mandato STRDBG.

Eliminación de objetos de programa o programas de servicio de una sesión de depuración

Puede eliminar objetos de programa o programas de servicio de una sesión de depuración después de arrancar la sesión.

Para eliminar objetos de programa y programas de servicio ILE de una sesión de depuración, utilice la opción 4 (Eliminar programa), junto al objeto de programa o programa de servicio que desea eliminar, en la pantalla Trabajar con lista de módulos (vea la Figura 34). Se puede acceder a la pantalla Trabajar con lista de módulos pulsando F14 (Trabajar con lista de módulos) en la pantalla Visualizar fuente del módulo.

Para eliminar **objetos de programa OPM** de una sesión de depuración, tiene dos opciones en función del valor especificado para OPMSRC. Si ha especificado OPMSRC(*YES) utilizando STRDBG, el mandato de depuración SET o CHGDBG, debe eliminar un programa OPM utilizando la pantalla Trabajar con lista de módulos. (Observe que no aparecerá ningún nombre de módulo para un programa OPM). No hay límite para el número de programas OPM que pueden eliminarse de una sesión de depuración en un momento determinado cuando se especifica OPMSRC(*YES). Si ha especificado OPMSRC(*NO), deberá utilizar el mandato Eliminar programa (RMVPGM). Cuando se especifica OPMSRC(*NO), sólo puede haber diez programas OPM en una sesión de depuración.

```
                          Trabajar con lista de módulos
                                                              Sistema:   AS400SYS
Teclee opciones, pulse Intro.
  1=Añadir programa 4= Eliminar programa 5= Visualizar fuente del módulo
  8=Trabajar con puntos de interrupción de módulos

                                                 Tipo de
Opc      Programa/módulo   Biblioteca          programa
_         ____________      *LIBL_____          *PGM___
4         TEST              TESTLIB             *PGM
_           SAMPMDF                           *MODULE
_         MYPGM1            TESTLIB             *PGM
_           MYPGM1                            *MODULE     Selected
_         USERDSP           DSPLIB              *SRVPGM
_           SAMPMDF                           *MODULE
_           GETUSER                           *MODULE
_


                                                                          Final
Mandato
===> ____________________________________________________________________________
F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
```

Figura 34. Eliminación de un objeto de programa ILE de una sesión de depuración

Cuando termine de eliminar objetos de programa o programas de servicio de una sesión de depuración, pulse F3 (Salir) en la pantalla Trabajar con lista de módulos para volver a la pantalla Visualizar fuente del módulo.

Nota: Debe tener autorización *CHANGE sobre un programa para eliminarlo de una sesión de depuración.

Visualización del fuente del programa

La pantalla Visualizar fuente del módulo muestra el fuente de un objeto de programa o programa de servicio ILE, visualizando un objeto de módulo cada vez. Los fuentes de los objetos de módulo pueden mostrarse si los objetos de módulo están creados con datos de depuración, utilizando una de las siguientes opciones de vista de depuración:

- DBGVIEW(*STMT)
- DBGVIEW(*SOURCE)
- DBGVIEW(*LIST)
- DBGVIEW(*ALL)

El fuente de un programa OPM puede mostrarse si se cumplen las siguientes condiciones:

1. El programa OPM debe estar compilado con OPTION(*LSTDBG) u OPTION(*SRCDBG). (Sólo los programas RPG y COBOL pueden compilarse con *LSTDBG).
2. El entorno de depuración de ILE debe estar definido para aceptar programas OPM, es decir, el valor de OPMSCR deber ser *YES. (El valor por omisión del sistema es OPMSRC(*NO)).

Hay dos métodos para modificar lo que se muestra en la pantalla Visualizar fuente del módulo:

- Cambiar el objeto de módulo que aparece.
- Cambiar la vista del objeto de módulo que aparece.

El depurador fuente ILE recuerda la última posición en la que visualizó el objeto de módulo y lo muestra en la misma posición cuando vuelve a visualizarlo. Se resaltan los números de las líneas en las que se han establecido puntos de interrupción. Cuando un punto de interrupción, un paso o un mensaje provoque la detención del programa y su aparición en la pantalla, se resaltará la línea fuente donde se produjo el suceso.

Modificación del objeto de módulo que aparece

Se puede cambiar el objeto de módulo que aparece en la pantalla Visualizar fuente del módulo utilizando la opción 5 (Visualizar fuente del módulo) de la pantalla Trabajar con lista de módulos. Se puede acceder a la pantalla Trabajar con lista de módulos pulsando F14 (Trabajar con lista de módulos) en la pantalla Visualizar fuente del módulo. La pantalla Trabajar con lista de módulos se muestra en la Figura 35 en la página 125.

Para seleccionar un objeto de módulo, teclee 5 (Visualizar fuente del módulo) junto al objeto de módulo que desea mostrar. Si utiliza esta opción con un objeto de programa ILE, aparece el objeto de módulo que contiene la vista fuente (si existe). En caso contrario, se muestra el primer objeto de módulo enlazado al objeto de programa que presente datos de depuración. Si utiliza esta opción con un objeto de programa OPM, aparece la vista fuente o de listado (si existe).

```
                          Trabajar con lista de módulos
                                                                Sistema:   AS400SYS
Teclee opciones, pulse Intro.
  1=Añadir programa 4= Eliminar programa 5= Visualizar fuente del módulo
  8=Trabajar con puntos de interrupción de módulos

                                                 Tipo de
Opc      Programa/módulo     Biblioteca          programa
_          _____________     *LIBL_____         *PGM___
_          TEST              TESTLIB            *PGM
5            SAMPMDF                          *MODULE
_          MYPGM1            TESTLIB            *PGM
_            MYPGM1                           *MODULE     Selected
_          USERDSP           DSPLIB             *SRVPGM
_            SAMPMDF                          *MODULE
_            GETUSER                          *MODULE


                                                                          Final
Mandato
===> ____________________________________________________________________________
F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar   F12=Cancelar
```

Figura 35. Visualización de una vista de módulo

Cuando haya seleccionado el objeto de módulo que desea ver, pulse Intro y la vista seleccionada aparecerá en la pantalla Visualizar fuente del módulo.

Otro método alternativo para cambiar el objeto de módulo que se muestra es utilizar el mandato de depuración DISPLAY. En la línea de mandatos de depuración, teclee:

```
DISPLAY MODULE nombre-módulo
```

Se mostrará el objeto de módulo *nombre-módulo*. El objeto de módulo debe existir en un objeto de programa que se haya añadido a la sesión de depuración.

Modificación de la vista del objeto de módulo que se muestra

Hay disponibles diferentes vistas de un objeto de módulo ILE COBOL/400 según los valores que se especifiquen cuando éste se crea. Estas vistas son:

- Vista de listado ILE COBOL/400
- Vista fuente ILE COBOL/400

Se puede cambiar la vista del objeto de módulo que aparece en la pantalla Visualizar fuente del módulo a través de la pantalla Seleccionar vista. Puede accederse a la pantalla Seleccionar vista pulsando F15 (Seleccionar vista) en la pantalla Visualizar fuente del módulo. En la Figura 36 en la página 126 se muestra la pantalla Seleccionar vista. La vista actual se lista en la parte superior de la pantalla y debajo el resto de vistas disponibles. Cada uno de los objetos de módulo de un objeto de programa o programa de servicio puede tener un conjunto diferente de vistas disponibles, según las opciones de depuración que se utilizaron para crearlo.

Para seleccionar una vista, teclee 1 (Seleccionar) junto a la vista que desea mostrar.

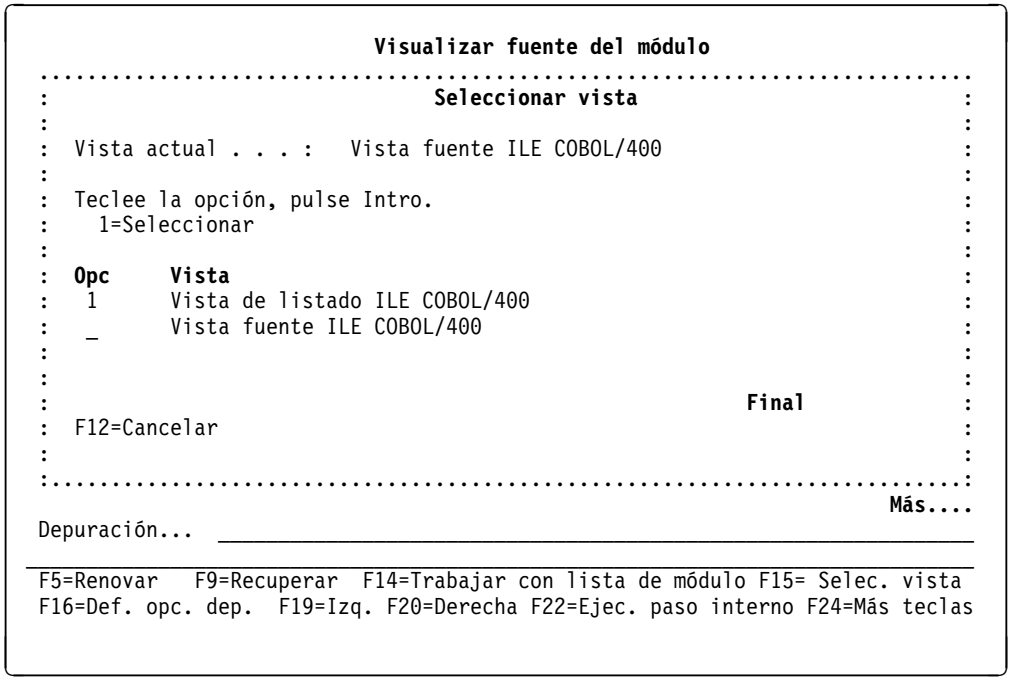


Figura 36. Modificación de la vista de un objeto de módulo

Cuando haya seleccionado la vista del objeto de módulo que desea mostrar, .* press F3 (Exit) pulse Intro y la vista seleccionada del objeto de módulo aparecerá en la pantalla Visualizar fuente del módulo.

Establecimiento y eliminación de puntos de interrupción

Puede utilizar puntos de interrupción para detener un objeto de programa o programa de servicio en un punto determinado cuando éste se ejecuta. Un **punto de interrupción incondicional** detiene el objeto de programa o programa de servicio en una instrucción determinada. Un **punto de interrupción condicional** detiene el objeto de programa o programa de servicio cuando se cumple una condición específica en una instrucción determinada.

Cuando se detiene el objeto de programa o programa de servicio, aparece la pantalla Visualizar fuente del módulo. El objeto de módulo apropiado se muestra con el fuente situado en la línea donde se ha producido el punto de interrupción.Esta línea se resalta. En este punto, puede evaluar las variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

Antes de utilizar puntos de interrupción, debería conocer las siguientes características:

- Si un punto de interrupción se salta, por ejemplo, con la instrucción GO TO, ese punto de interrupción no se procesa.
- Cuando un punto de interrupción se establece en una instrucción, éste se produce antes de procesar la instrucción.
- Cuando se alcanza una instrucción que tiene un punto de interrupción condicional, la expresión condicional asociada al punto de interrupción se evalúa antes de procesar la instrucción.

- Las funciones de los puntos de interrupción se especifican con los mandatos de depuración.

  Estas funciones incluyen:

  – Añadir puntos de interrupción
  – Eliminar puntos de interrupción
  – Visualizar información acerca de puntos de interrupción
  – Reanudar la ejecución de un objeto de programa o programa de servicio después de haber alcanzado un punto de interrupción.

Establecimiento y eliminación de puntos de interrupción incondicionales

Puede establecer o eliminar un punto de interrupción incondicional utilizando:

- F6 (Añadir/Borrar punto de interrupción) en la pantalla Visualizar fuente del módulo
- F13 (Trabajar con puntos de interrupción de módulo) en la pantalla Visualizar fuente del módulo
- El mandato de depuración BREAK para establecer un punto de interrupción
- El mandato de depuración CLEAR para eliminar un punto de interrupción

El método más sencillo para establecer y eliminar un punto de interrupción es utilizar F6 (Añadir/Borrar punto de interrupción) en la pantalla Visualizar fuente del módulo. Para establecer un punto de interrupción incondicional utilizando F6 (Añadir/Borrar punto de interrupción), sitúe el cursor en la línea donde desea añadir el punto de interrupción y pulse F6 (Añadir/Borrar punto de interrupción).Los puntos de interrupción incondicionales se establecen en la línea. Para eliminar un punto de interrupción incondicional utilizando F6 (Añadir/Borrar punto de interrupción), sitúe el cursor en la línea de la que quiere eliminar el punto de interrupción y pulse F6 (Añadir/Borrar punto de interrupción).Los puntos de interrupción se eliminan de la línea.

Repita los pasos anteriores para cada punto de interrupción incondicional que desee establecer.

Si la línea en la que desea establecer el punto de interrupción contiene varias instrucciones, al pulsar F6 (Añadir/Borrar punto de interrupción) se establecerá el punto de interrupción en la primera instrucción de la línea.

Nota: Si la línea donde desea establecer el punto de interrupción no es una instrucción ejecutable, el punto de interrupción se establecerá en la siguiente instrucción ejecutable.

Cuando haya establecido los puntos de interrupción, pulse F3 (Fin programa) para salir de la pantalla Visualizar fuente del módulo. También puede utilizar F21 (Línea de mandatos) en la pantalla Visualizar fuente del módulo para llamar al programa desde una línea de mandatos.

Llame al objeto de programa. Cuando se alcanza un punto de interrupción, se detiene el objeto de programa o programa de servicio y aparece de nuevo la pantalla Visualizar fuente del módulo. En este punto, puede evaluar las variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

Un método alternativo para establecer y eliminar puntos de interrupción incondicionales es utilizar los mandatos de depuración BREAK y CLEAR.

Para establecer un punto de interrupción incondicional utilizando el mandato de depuración BREAK, teclee:

```
BREAK número-línea
```

en la línea de mandatos de depuración.*Número-línea* es el .* number of the debugger line número de línea de la vista del objeto de módulo que se visualiza actualmente donde desea establecer un punto de interrupción.

Si la línea en la que desea establecer un punto de interrupción contiene varias instrucciones, el mandato de depuración BREAK establecerá el punto de interrupción en la primera instrucción de la línea.

Para eliminar un punto de interrupción incondicional utilizando el mandato de depuración CLEAR, teclee:

```
CLEAR número-línea
```

en la línea de mandatos de depuración.*Número-línea* es el número de línea de la vista del objeto de módulo que se visualiza actualmente de la que desea eliminar un punto de interrupción.

Establecimiento y eliminación de puntos de interrupción condicionales

Puede establecer o eliminar un punto de interrupción condicional utilizando:

- La pantalla Trabajar con puntos de interrupción de módulos
- El mandato de depuración BREAK para establecer un punto de interrupción
- El mandato de depuración CLEAR para eliminar un punto de interrupción

Nota: Los operadores relacionales que se soportan para puntos de interrupción condicionales son <, >, =, =<, => y <>.

Un método para establecer o eliminar puntos de interrupción condicionales es utilizar la pantalla Trabajar con puntos de interrupción de módulos. Se puede acceder a la pantalla Trabajar con puntos de interrupción de módulos pulsando F13 (Trabajar con puntos de interrupción de módulos) en la pantalla Visualizar fuente del módulo. En la Figura 37 en la página 129 se muestra la pantalla Trabajar con puntos de interrupción de módulos. Para establecer un punto de interrupción condicional, teclee 1 (Añadir) en el campo *Opc*, teclee el número de línea del depurador donde desea establecer el punto de interrupción en el campo *Línea*, teclee una expresión condicional en el campo *Condición* y pulse Intro. Por ejemplo, para establecer un punto de interrupción condicional en la línea de depurador 35, teclee 1 (Añadir) en el campo *Opc*, teclee 35 en el campo *Línea*, teclee I=21 en el campo *Condición* y pulse Intro, como se muestra en la Figura 37 en la página 129. La expresión condicional sólo puede ser una expresión simple. El término situado en la parte derecha de la ecuación sólo puede contener un valor numérico único.Por ejemplo, I=21 se acepta pero I=A+2 o I=3*2 no se aceptan.

Para eliminar un punto de interrupción condicional, teclee 4 (Borrar) en el campo *Opc* junto al punto de interrupción que desee eliminar y pulse Intro. También puede eliminar puntos de interrupción incondicionales de este modo.

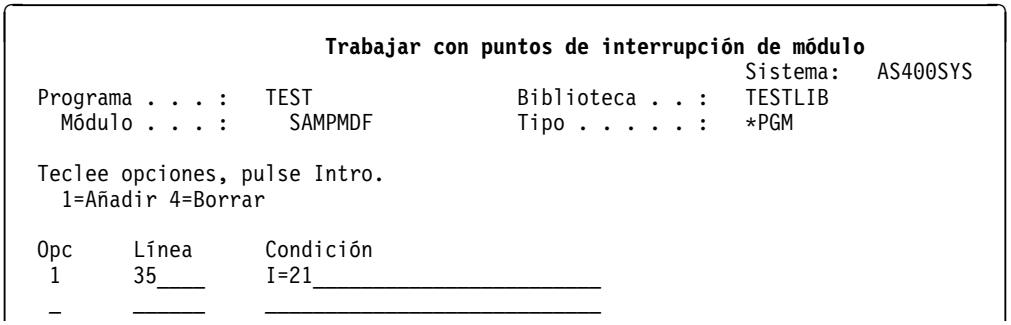


Figura 37. Establecimiento de un punto de interrupción condicional

Repita los pasos anteriores para cada punto de interrupción condicional que desee establecer o eliminar.

Si la línea en la que desea establecer el punto de interrupción contiene varias instrucciones, el punto de interrupción se establece en la primera instrucción de la línea.

Nota: Si la línea donde desea establecer el punto de interrupción no es una instrucción ejecutable, el punto de interrupción se establecerá en la siguiente instrucción ejecutable.

Cuando haya terminado de especificar todos los puntos de interrupción que desea establecer o eliminar, pulse F3 (Salir) para volver a la pantalla Visualizar fuente del módulo.

A continuación, pulse F3 (Fin programa) para salir de la pantalla Visualizar fuente del módulo. También puede utilizar F21 (Línea de mandatos) en la pantalla Visualizar fuente del módulo para llamar al programa desde una línea de mandatos.

Ejecute el objeto de programa o el programa de servicio. Cuando se alcanza una instrucción que tiene un punto de interrupción condicional, la expresión condicional asociada al punto de interrupción se evalúa antes de ejecutar la instrucción. Si el resultado es falso, el objeto de programa continúa la ejecución. Si el resultado es verdadero, el objeto de programa se detiene y aparece la pantalla Visualizar fuente del módulo. En este punto, puede evaluar las variables, establecer más puntos de interrupción y ejecutar cualquiera de los mandatos de depuración.

Un método alternativo para establecer y eliminar puntos de interrupción condicionales es utilizar los mandatos de depuración BREAK y CLEAR.

Para establecer un punto de interrupción condicional utilizando el mandato de depuración BREAK, teclee:

```
BREAK número-línea WHEN expresión
```

en la línea de mandatos de depuración.*Número-línea* es el número de línea de la vista del objeto de módulo que actualmente se visualiza en la que desea establecer el punto de interrupción y *expresión* es la expresión condicional que se evalúa cuando se encuentra el punto de interrupción. La expresión condicional sólo puede ser una expresión simple. El término situado en la parte derecha de la ecuación sólo puede contener un valor numérico único.Por ejemplo, I=21 se acepta pero I=A+2 o I=3*2 no se aceptan.

Si la línea en la que desea establecer un punto de interrupción contiene varias instrucciones, el mandato de depuración BREAK establecerá el punto de interrupción en la primera instrucción de la línea.

Para eliminar un punto de interrupción condicional utilizando el mandato de depuración CLEAR, teclee:

```
CLEAR número-línea
```

en la línea de mandatos de depuración.*Número-línea* es el número de línea de la vista del objeto de módulo que se visualiza actualmente de la que desea eliminar un punto de interrupción.

Eliminación de todos los puntos de interrupción

Se pueden eliminar todos los puntos de interrupción, condicionales o incondicionales, de un objeto de programa que tiene un objeto de módulo que aparece en la pantalla Visualizar fuente del módulo utilizando el mandato de depuración CLEAR PGM.Para utilizar este mandato de depuración, teclee:

```
CLEAR PGM
```

en la línea de mandatos de depuración. Los puntos de interrupción se eliminan de todos los módulos enlazados al programa.

Establecimiento y eliminación de condiciones de observación

Una **condición de observación** le permite solicitar un punto de interrupción cuando se cambia el valor actual del contenido de una variable especificada (o de una expresión que se relaciona con una subserie o con un elemento de matriz). La manera de establecer condiciones de observación es similar a la forma de establecer puntos de interrupción condicionales pero existen las siguientes diferencias, que son importantes:

- Las condiciones de observación detienen el programa tan pronto como el cambia el valor actual de la expresión o variable observada.
- Los puntos de interrupción condicionales sólo detienen el programa si una variable llega al valor especificado en la condición.

El depurador observa una expresión o una variable en función del contenido de una **dirección de almacenamiento**, calculado en el momento en que se establece la condición de observación. Cuando el contenido de la dirección de almacenamiento es distinto del valor que tenía cuando se estableció la condición de observación o cuando se produjo la última condición de observación, se establece un punto de interrupción y el programa se detiene.

Nota: Después de registrarse una condición de observación, el contenido nuevo en la ubicación de almacenamiento observada se guarda como el nuevo valor actual de la expresión o variable correspondiente. La siguiente condición de observación se registrará si el nuevo contenido de la ubicación de almacenamiento observada cambia con posterioridad.

Características de las observaciones

Antes de utilizar observaciones, debería conocer las siguientes características:

- Las observaciones se supervisan en todo el sistema y puede activarse un máximo de 256 observaciones de forma simultánea. Este número incluye las observaciones establecidas por el propio sistema.

  En función de la utilización general del sistema, es posible que el número de condiciones de observación que puede establecer en un momento determinado esté restringido. Si intenta establecer una condición de observación mientras se ha excedido el número máximo de observaciones activas en el sistema, recibirá un mensaje de error y no se establecerá la condición de observación.

  Nota: Si una expresión o variable pasa el límite de página, se utilizan dos observaciones internamente para supervisar las ubicaciones de almacenamiento. Por este motivo, el número máximo de expresiones o variables que se puede observar de forma simultánea en todo el sistema oscila entre 128 y 256.

- Las condiciones de observación sólo pueden establecerse cuando un programa se detiene mientras realiza una depuración y la expresión o variable que debe observarse se encuentra en su ámbito. En caso contrario, se emitirá un mensaje de error cuando se solicite una observación, indicando que no existe la entrada de la pila de llamadas correspondiente.

- Una vez se haya establecido la condición, la dirección de una ubicación de almacenamiento bajo observación no cambia. Por consiguiente, si una observación está definida en una ubicación temporal, podrían producirse falsos avisos de condiciones de observación.

  Un ejemplo de este caso lo tenemos en el almacenamiento automático de un procedimiento ILE COBOL/400, que puede volverse a utilizar una vez finaliza el procedimiento.

  Es posible que se registre una condición de observación aunque la variable observada ya no se encuentre en su ámbito. No debe asumir que una variable está en su ámbito simplemente porque se ha comunicado una condición de observación.

- Dos ubicaciones de observación que se encuentren en un mismo trabajo no deben solaparse de ninguna forma. Dos ubicaciones de observación que se encuentren en trabajos distintos no deben empezar en la misma dirección de almacenamiento; en caso contrario, se permite el solapamiento. Si se violan estas restricciones, se emite un mensaje de error.

  Nota: Las modificaciones realizadas en una ubicación de almacenamiento que esté bajo observación no se tienen en cuenta a menos que se realicen por el trabajo que ha establecido la condición de observación.

- Después de ejecutar el mandato de forma satisfactoria, la aplicación se detiene si algún programa de la sesión modifica el contenido de la ubicación de almacenamiento que se observa, en cuyo caso aparece la pantalla Visualizar fuente del módulo.

  Si el programa presenta datos de depuración y la vista de texto fuente está disponible, ésta se visualizará. Se resalta la línea fuente de la instrucción que estaba a punto de ser ejecutada cuando se detectó el cambio de contenido de la ubicación de almacenamiento. Un mensaje indica la condición de observación que se ha cumplido.

Si no es posible depurar el programa, el área de texto de la pantalla aparecerá en blanco.

- Los programas seleccionables se añaden de forma automática a la sesión de depuración si son responsables de la condición por la que se detuvo la observación.
- Cuando existen varias condiciones de observación en la misma instrucción de un programa, sólo se informa de la primera.
- También es posible establecer condiciones de observación al utilizar trabajos de servicio para realizar la depuración, es decir, al depurar un trabajo desde otro.

Establecimiento de condiciones de observación

Antes de establecer una condición de observación, el programa debe detenerse en una depuración y la expresión o variable que desea observar debe estar en su ámbito.

- Para observar una variable global, debe asegurarse de que el programa COBOL en el que está definida la variable esté activo antes de establecer la condición de observación.
- Para observar una variable local, debe entrar en el programa COBOL en el que está definida la variable antes de establecer la condición de observación.

Puede establecer una condición de observación utilizando:

- F18 (Trabajar con observación) para mostrar la pantalla Trabajar con observación desde la que puede borrar o visualizar las observaciones
- F17 (Variable de observación) para establecer una condición de observación para una variable (un elemento de datos de COBOL) en la que se encuentra situado el cursor
- El mandato de depuración WATCH, tanto si tiene parámetros especificados como si no.

Utilización del mandato WATCH

Si utiliza el mandato WATCH, éste debe entrarse como un único mandato; no se permite ningún otro mandato de depuración en la misma línea de mandatos.

- Para acceder a la pantalla Trabajar con observación que se muestra a continuación, teclee:

  ```
  WATCH
  ```

  en la línea de mandatos, sin especificar ningún parámetro.

```
                              Trabajar con observación

                                                        Sistema:  DEBUGGER
Teclee opciones, pulse Intro.
   4=Borrar  5=Visualizar

 Opc    Núm      Variable                 Dirección          Longitud
  -     1        KOUNT                    080090506F027004   4

                                                                      Final
Mandato
===>_____________________________________________________________________
F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar  F12=Cancelar
```

Figura 38. Ejemplo de una pantalla Trabajar con observación

La pantalla Trabajar con observación muestra todas las observaciones que actualmente están activas en la sesión de depuración. Puede eliminar o visualizar observaciones desde esta pantalla. Si selecciona la Opción 5 (Visualizar), la ventana Visualizar observación que aparece a continuación muestra información acerca de la observación que está activa actualmente.

```
                         Trabajar con observación
.....................................................
:                    Visualizar observación         :  DEBUGGER
:                                                   :
:  Núm. observación..:  1                           :
:  Dirección.........:  080090506F027004            :
:  Longitud..........:  4                           :
:  Núm. coincidencias:  0                           :
:                                                   :
:  Ámbito cuando se definió el programa:            :
:    Programa/biblioteca/tipo:   PAYROLL   ABC   *PGM    :
:                                                   :
:    Módulo...:    PAYROLL                          :
:    Procedim.:    main                             :
:    Variable.:    KOUNT                            :
:                                                   :
:  F12=Cancelar                                     :
:                                                   :
.....................................................
                                                                  Final
Mandato
===>_____________________________________________________________________
F3=Salir  F4=Solicitud  F5=Renovar   F9=Recuperar  F12=Cancelar
```

Figura 39. Ejemplo de una ventana Visualizar observación

- Para especificar una variable o expresión que debe observarse, realice una de las siguientes acciones:
  - Para especificar una variable que desea observar, teclee:

```
WATCH SALARY
```

en la línea de mandatos de depuración, en la que SALARY es una variable.

Este mandato solicita el establecimiento de un punto de interrupción si cambia el valor actual de `SALARY`.

El ámbito de las variables de expresión en una observación viene definido por el mandato QUAL que se haya emitido más recientemente.

– Para especificar una expresión que debe observarse:

```
WATCH %SUBSTR(A 1 3)
```

donde A forma parte de una expresión de subserie. Para obtener más información sobre subserie consulte el apartado “Visualización de una subserie de una variable de serie de caracteres” en la página 143.

Nota: En ILE COBOL/400, sólo es posible observar expresiones relacionadas con subseries o elementos de matriz.

- Para establecer una condición de observación y especificar una longitud de observación, teclee:

```
WATCH expresión : longitud-observación
```

en una línea de mandatos de depuración.

Cada observación le permite supervisar y comparar 128 bytes de almacenamiento contiguo como máximo. Si se sobrepasa la longitud máxima de 128 bytes, no se establecerá la condición de observación y el depurador emitirá un mensaje de error.

Por omisión, la longitud del tipo de expresión coincide con la longitud de la comparación de observación. El *parámetro longitud-observación* altera este valor por omisión temporalmente. Determina el número de bytes de una expresión que deben compararse para determinar si se ha producido algún cambio en un valor.

Por ejemplo, si se especifica un entero binario de 4 bytes como la variable, sin incluir el parámetro longitud-observación, la longitud de la comparación será de cuatro bytes. Sin embargo, si se especifica el parámetro longitud-observación, éste altera la longitud de la expresión temporalmente y determina la longitud de la observación.

Visualización de observaciones activas

Para visualizar una lista de todas las observaciones activas del sistema y mostrar el trabajo que las ha establecido, teclee:

```
DSPDBGWCH
```

en una línea de mandatos CL. El mandato visualiza la pantalla Visualizar observaciones de depuración que se muestra a continuación.

```
                          Visualizar observaciones de depuración

                                                                  Sistema:  DEBUGGER
-----------Trabajo------------        NÚM     LONGITUD   DIRECCIÓN
MYJOBNAME1  MYUSERPRF1  123456          1          4     080090506F027004
JOB4567890  PRF4567890  222222          1          4     09849403845A2C32
JOB4567890  PRF4567890  222222          2          4     098494038456AA00
JOB         PROFILE     333333         14          4     040689578309AF09
SOMEJOB     SOMEPROFIL  444444          3          4     005498348048242A









                                                                        Final
Pulse Intro para continuar

F3=Salir   F5=Renovar   F12=Cancelar
```

Figura 40. Ejemplo de una pantalla Visualizar observaciones de depuración

Nota: Esta pantalla no muestra las condiciones de observación establecidas por el sistema.

Eliminación de las condiciones de observación

Es posible eliminar observaciones de las formas siguientes:

- El mandato CLEAR utilizado con la palabra clave WATCH termina una o todas las observaciones de forma selectiva. Por ejemplo, para borrar la observación identificada mediante *número-observación*, teclee:

  ```
  CLEAR WATCH número-observación
  ```

  El número de observación puede obtenerse de la pantalla Trabajar con observaciones.

  Para borrar todas las observaciones de la sesión, teclee:

  ```
  CLEAR WATCH ALL
  ```

  en una línea de mandatos de depuración.

  Nota: Aunque el mandato CLEAR PGM elimina todos los puntos de interrupción del programa que contiene el módulo que se visualiza, su efecto sobre las observaciones es nulo. Es preciso utilizar la palabra clave WATCH de forma explícita con el mandato CLEAR para eliminar las condiciones de observación.

- El mandato de CL Finalizar depuración (ENDDBG) elimina las observaciones establecidas en el trabajo local o en un trabajo de servicio.

  Nota: En situaciones anormales, ENDDBG se llama de forma automática para asegurar la eliminación de todas las observaciones afectadas.

- La carga del programa inicial (IPL) del sistema AS/400 elimina todas las condiciones de observación del sistema.

Ejemplo del establecimiento de una condición de observación

En este ejemplo, se muestra una variable *kount* del programa MYLIB/PAYROLL. Para establecer la condición de observación, teclee:

```
WATCH kount
```

en una línea de depuración, aceptando el valor por omisión de longitud-observación.

Si el valor de la variable *kount* cambia posteriormente, la aplicación se detiene y se muestra la pantalla Visualizar fuente del módulo, tal como se ilustra en la Figura 41.

```
                          Visualizar fuente del módulo

Programa:  PAYROLL       Biblioteca:   MYLIB       Módulo:  PAYROLL
    42        * LOS 3 PÁRRAFOS SIGUIENTES CREAN INTERNAMENTE LOS  *
    43        * REGISTROS QUE DEBEN INCLUIRSE EN EL ARCHIVO, LOS  *
    44        * GRABA EN DISCO Y LOS VISUALIZA                    *
    45        ******************************************************
    46         STEP-2.
    47             ADD 1 TO KOUNT, NUMBR.
    48             MOVE ALPHA  (KOUNT) TO NAME-FIELD.
    49             MOVE DEPEND (KOUNT) TO NO-OF-DEPENDENTS.
    50             MOVE NUMBR          TO RECORD-NO.

    51         STEP-3.
    52             DISPLAY WORK-RECORD.
    53             WRITE RECORD-1 FROM WORK-RECORD.

    54         STEP-4.
    55             PERFORM STEP-2 THRU STEP-3 UNTIL KOUNT IS =
                                                                  Más....
Depuración . . .  _____________________________________________________
_______________________________________________________________________
F3=Fin programa F6=Añadir/Borrar punto int. F10=Paso F11=Visualizar variable
F12=Reanudar    F17=Observar variable  F18=Trabajar observación  F24=Más teclas
Número observación 1 en la línea 55, variable: KOUNT
```

Figura 41. Ejemplo de una pantalla Visualizar fuente del módulo

- Se resalta el número de línea de la instrucción en la que se detectó la modificación en la variable de observación. Suele tratarse de la primera línea ejecutable *después* de la instrucción que ha modificado la variable.
- Un mensaje indica que se ha cumplido la condición de observación.

  Nota: Si no está disponible una vista de texto, aparece una pantalla Visualizar fuente del módulo en blanco y en el área de mensajes de aparece este mismo mensaje.

  No es posible añadir los siguientes programas al entorno de depuración ILE:

  1. Programas ILE sin datos de depuración
  2. Programas OPM que sólo contienen datos de depuración que no son fuente
  3. Programas OPM sin datos de depuración

  En los dos primeros casos, se muestra el número de la instrucción en el que se ha detenido. En el tercer caso, se muestra la instrucción MI en la que se

ha detenido. La información se visualiza al final de una pantalla Visualizar fuente del módulo en blanco, tal como se muestra a continuación. En lugar del número de línea, se proporciona la instrucción MI o el número de ésta.

```
                          Visualizar fuente del módulo

Programa:  PAYROLL       Biblioteca:   MYLIB       Módulo:  PAYROLL

  (Fuente no disponible).









                                                                      Final
Depuración . . .  _____________________________________________________
_____________________________________________________________________
F3=Fin programa F6=Añadir/Borrar punto int. F10= Paso F11= Visualizar variable
F12=Reanudar    F17=Observar variable  F18=Trabajar observación  F24=Más teclas
Número observación 1 en la instrucción 18, variable: KOUNT
```

Figura 42. Ejemplo de una ventana Visualizar fuente del módulo

Ejecución de un objeto de programa o procedimiento ILE después de un punto de interrupción

Después de encontrar un punto de interrupción, se puede reanudar la ejecución de un objeto de programa o procedimiento ILE de dos maneras:

- reanude la ejecución del objeto de programa o procedimiento ILE en la siguiente instrucción después del punto de interrupción y deténgala en el siguiente punto de interrupción o cuando el programa finalice.
- ejecute por pasos un número de instrucciones determinado después del punto de interrupción y detenga el objeto de programa de nuevo.

Reanudación de un objeto de programa o procedimiento ILE

Después de encontrar un punto de interrupción, se puede reanudar un objeto de programa o procedimiento ILE pulsando F12 (Reanudar) en la pantalla Visualizar fuente del módulo. El objeto de programa o procedimiento ILE empieza a ejecutarse en la siguiente instrucción del objeto de módulo donde el programa se detuvo. El objeto de programa o procedimiento ILE se detendrá de nuevo en el siguiente punto de interrupción o cuando el objeto de programa finalice.

Ejecución por pasos del objeto de programa o procedimiento ILE

Después de encontrar un punto de interrupción, puede ejecutar un número especificado de instrucciones de un objeto de programa o procedimiento ILE, detener el programa de nuevo y volver a la pantalla Visualizar fuente del módulo. El objeto de programa o procedimiento ILE empieza a ejecutarse en la siguiente instrucción del objeto de módulo donde el programa se detuvo.

| Es posible ejecutar un programa OPM por pasos si éste tiene disponibles datos de
| depuración y si la sesión de depuración acepta la depuración de programas OPM.

Se puede ejecutar por pasos un objeto de programa o procedimiento ILE utilizando:

- F10 (Paso) o F22 (Ejecutar pasos internos) en la pantalla Visualizar fuente del módulo
- El mandato de depuración STEP

El método más sencillo para ejecutar por pasos un objeto de programa o procedimiento ILE es utilizar F10 (Paso) o F22 (Ejecutar pasos internos) en la pantalla Visualizar fuente del módulo. Cuando se pulsa F10 (Paso) o F22 (Ejecutar pasos internos), se ejecuta la siguiente instrucción del objeto de módulo que aparece en la pantalla Visualizar fuente del módulo y vuelve a detenerse el objeto de programa o procedimiento ILE. Si hay varias instrucciones en la línea en la que se pulsa F10 (Paso) o F22 (Ejecutar pasos internos), todas las instrucciones de esta línea se ejecutan y el objeto de programa o procedimiento ILE se detiene en la siguiente instrucción de la siguiente línea.

Nota: No se puede especificar el número de instrucciones para ejecutar por pasos cuando se utiliza F10 (Paso) o F22 (Ejecutar pasos internos). Al pulsar F10 (Paso) o F22 (Ejecutar pasos internos) se efectúa un sólo paso.

Otro método para ejecutar por pasos un programa o procedimiento ILE consiste en utilizar el mandato de depuración STEP. El mandato de depuración STEP permite ejecutar más de una instrucción en un solo paso. El número por omisión de instrucciones que se ejecutan utilizando el mandato de depuración STEP, es una. Para ejecutar por pasos un objeto de programa o procedimiento ILE utilizando el mandato de depuración STEP, teclee:

```
STEP número-de-instrucciones
```

en la línea de mandatos de depuración. *Número-de-instrucciones* es el número de instrucciones que desea ejecutar en el paso siguiente antes de que la aplicación se detenga de nuevo. Por ejemplo, si teclea

```
STEP 5
```

en la línea de mandatos, se ejecutan las cinco instrucciones siguientes del objeto de programa o procedimiento ILE, el objeto de programa o procedimiento ILE se detiene de nuevo y aparece la pantalla Visualizar fuente del módulo.

Cuando se encuentra una instrucción CALL a otro objeto de programa o procedimiento ILE en una sesión de depuración, es posible:

- Ejecutar pasos externos del objeto de programa o procedimiento ILE llamado o
- Ejecutar pasos internos del objeto de programa o procedimiento ILE llamado.

Si selecciona **ejecutar pasos externos** del objeto de programa o procedimiento ILE llamado, la instrucción CALL y el objeto de programa llamado se ejecutan

como un único paso. El objeto de programa o procedimiento ILE llamado se ejecuta hasta el final antes de que el objeto de programa o procedimiento ILE de llamada se detenga en el siguiente paso. La ejecución por pasos externos es la modalidad de paso por omisión.

Si selecciona **ejecutar pasos internos** en el objeto de programa o procedimiento ILE llamado, cada una de las instrucciones del objeto de programa o procedimiento ILE llamado se ejecuta como un único paso. Si el siguiente paso en el que el objeto de programa o procedimiento ILE en ejecución debe detenerse se encuentra dentro del objeto de programa o procedimiento ILE llamado, el objeto de programa o procedimiento ILE llamado se detiene en este punto y el objeto de programa o procedimiento ILE llamado aparece en la pantalla Visualizar fuente del módulo.

Ejecución por pasos externos de objetos de programa o procedimientos ILE

Es posible ejecutar objetos de programa o procedimiento ILE por pasos internos utilizando:

- F10 (Paso) en la pantalla Visualizar fuente del módulo
- El mandato de depuración STEP OVER

Puede utilizar F10 (Paso) en la pantalla Visualizar fuente del módulo para ejecutar un objeto de programa o procedimiento ILE llamado en una sesión de depuración por pasos internos. Si la siguiente instrucción que va a ejecutarse es una instrucción CALL a otro objeto de programa o procedimiento ILE, al pulsar F10 (Paso) se ejecutará el objeto de programa o procedimiento ILE llamado hasta el final antes de que el objeto de programa o procedimiento ILE se detenga de nuevo.

De modo alternativo, puede utilizar el mandato de depuración STEP OVER para ejecutar un objeto de programa o procedimiento ILE llamado en una sesión de depuración por pasos externos. Para utilizar el mandato de depuración STEP OVER, teclee:

```
STEP número-de-instrucciones OVER
```

en la línea de mandatos de depuración. *Número-de-instrucciones* es el número de instrucciones que desea ejecutar en el paso siguiente antes de que la aplicación se detenga de nuevo. Si una de las instrucciones que se ejecuta contiene una instrucción CALL para otro objeto de programa o procedimiento ILE, el depurador fuente ILE ejecutará el objeto de programa o procedimiento ILE llamado por pasos externos.

Ejecución por pasos internos de objetos de programa o procedimientos ILE

Es posible ejecutar un objeto de programa o procedimientos ILE por pasos externos utilizando:

- F22 (Ejecutar pasos internos) en la pantalla Visualizar fuente del módulo
- El mandato de depuración STEP INTO

Puede utilizar F22 (Ejecutar pasos internos) en la pantalla Visualizar fuente del módulo para ejecutar un objeto de programa o procedimiento ILE llamado en una sesión de depuración por pasos internos. Si la siguiente instrucción que va a ejecutarse es una instrucción CALL para otro objeto de programa o procedimiento ILE, al pulsar F22 (Ejecutar pasos internos) se ejecutará la primera instrucción ejecutable del objeto de programa o procedimiento ILE llamado. El objeto de pro-

grama o procedimiento ILE llamado aparecerá en la pantalla Visualizar fuente del módulo.

Nota: Un objeto de programa o procedimiento ILE llamado debe tener datos de depuración asociados para que pueda aparecer en la pantalla Visualizar fuente del módulo. Un programa OPM llamado aparecerá en la pantalla Visualizar módulo del fuente si el depurador fuente ILE está configurado para que acepte programas OPM y el programa OPM presenta datos de depuración. (Un programa OPM presentará datos de depuración si se ha compilado con OPTION(*SRCDBG) u OPTION(*LSTDBG)).

De modo alternativo, puede utilizar el mandato de depuración STEP INTO para ejecutar un objeto de programa o procedimiento ILE llamado en una sesión de depuración por pasos externos. Para utilizar el mandato de depuración STEP INTO, teclee:

```
STEP número-de-instrucciones INTO
```

en la línea de mandatos de depuración. *Número-de-instrucciones* es el número de instrucciones del objeto de programa o procedimiento ILE que desea ejecutar en el paso siguiente antes de que el objeto de programa o procedimiento ILE se detenga de nuevo. Si una de las instrucciones que se ejecuta contiene una instrucción CALL para otro objeto de programa o procedimiento ILE, el depurador ejecutará el objeto de programa o procedimiento ILE llamado por pasos internos. Cada una de las instrucciones del objeto de programa o procedimiento ILE llamado se contará en los pasos. Si los pasos finalizan en el objeto de programa o procedimiento ILE llamado, el objeto de programa o procedimiento ILE llamado aparecerá en la pantalla Visualizar fuente del módulo. Por ejemplo, si teclea

```
STEP 5 INTO
```

en la línea de mandatos de depuración, se ejecutan las cinco instrucciones siguientes del objeto de programa o procedimiento ILE. Si la tercera instrucción es una instrucción CALL para otro objeto de programa o procedimiento ILE, se ejecutan dos instrucciones del objeto de programa o procedimiento ILE de llamada y las tres primeras instrucciones del objeto de programa o procedimiento ILE llamado.

Visualización de variables, expresiones, registros, elementos de grupo y matrices

Puede visualizar los valores de variables, expresiones, elementos de grupo registros y matrices utilizando:

- F11 (Visualizar variable) en la pantalla Visualizar fuente del módulo
- El mandato de depuración EVAL

El ámbito de las variables que se utilizan en el mandato EVAL se define utilizando el mandato QUAL.

Nota: El depurador fuente ILE no da soporte a los registros especiales ILE COBOL/400.Así, los valores que contienen los registros especiales ILE COBOL/400 no pueden visualizarse en una sesión de depuración. El depurador fuente ILE no puede evaluar el resultado de un identificador de función COBOL.

Visualización de variables y expresiones

El método más simple para visualizar el valor de una variable es utilizar F11 (Visualizar variable) en la pantalla Visualizar fuente del módulo. Para visualizar una variable utilizando F11 (Visualizar variable), sitúe el cursor en la variable que desea visualizar y pulse F11 (Visualizar variable).El valor actual de la variable aparece en la línea de mensajes situada en la parte inferior de la pantalla Visualizar fuente del módulo.Por ejemplo, si desea ver el valor de la variable *COUNTER* en la línea 221 del objeto de módulo que aparece en la Figura 43, sitúe el cursor sobre la variable y pulse F11 (Visualizar variable). El valor actual de la variable *COUNTER* aparece en la línea de mensajes.

```
                          Visualizar fuente del módulo

Programa:  TEST           Biblioteca:   TESTLIB        Módulo:   SAMPMDF
   213
   214         PROCEDURE-SECTION SECTION.
   215         FILL-TERMINAL-LIST.
   216             READ TERMINAL-FILE RECORD INTO LIST-OF-TERMINALS(COUNTER)
   217                 AT END
   218                     SET END-OF-TERMINAL-LIST TO TRUE
   219                     SUBTRACT 1 FROM COUNTER
   220                     MOVE COUNTER TO NO-OF-TERMINALS.
   221             ADD 1 TO COUNTER.
   222
   223         ACQUIRE-AND-INVITE-TERMINALS.
   224             ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE FILE.
   225             WRITE MULTIPLE-REC
   226                 FORMAT IS "SIGNON"
   227                 TERMINAL IS LIST-OF-TERMINALS(COUNTER).
                                                                      Más....
Depuración... _____________________________________________________________
_____________________________________________________________________________
F3=Fin programa  F6=Añadir/Borrar punto int. F10= Paso F11= Visualizar variable
F12=Reanudar     F13=Trabajar con punt. int. módulo   F24=Más teclas
COUNTER = 89.
```

Figura 43. Visualización de una variable utilizando F11 (Visualizar variable)

Cuando se evalúan registros, elementos de grupo o matrices, el mensaje que se devuelve al pulsar F11 (Visualizar variable) puede ocupar varias líneas. Los mensajes que ocupan varias líneas aparecen en la pantalla Evaluar expresión que muestra el texto completo del mensaje. Cuando haya terminado de visualizar el mensaje de la pantalla Evaluar expresión, pulse Intro para volver a la pantalla Visualizar fuente del módulo.

También puede utilizar el mandato de depuración EVAL para determinar el valor de una variable.Primero, utilice el mandato de depuración QUAL para identificar el número de línea de la variable que desea mostrar. Desde esta línea se aplican las normas para determinar el ámbito de una variable.

Nota: La posición de QUAL por omisión es la línea actual.

Para visualizar el valor de una variable por omisión utilizando el mandato de depuración EVAL, teclee:

```
EVAL nombre-variable
```

en la línea de mandatos de depuración.*Nombre-variable* es el nombre de la variable que desea visualizar. El valor de la variable aparece en la línea de men-

sajes si el mandato de depuración EVAL se entra desde la pantalla Visualizar fuente del módulo y el valor puede mostrarse en una única línea. De lo contrario, el valor de la variable aparece en la pantalla Evaluar expresión.

Por ejemplo, para visualizar el valor de la variable *COUNTER* en la línea 221 del objeto de módulo que aparece en la Figura 43 en la página 141, teclee:

```
EVAL COUNTER
```

para visualizar la variable *COUNTER*. La línea de mensajes de la pantalla Visualizar fuente del módulo muestra `COUNTER = 89` como en la Figura 44.

```
                          Visualizar fuente del módulo

Programa:  TEST           Biblioteca:   TESTLIB        Módulo:   SAMPMDF
   213
   214         PROCEDURE-SECTION SECTION.
   215         FILL-TERMINAL-LIST.
   216             READ TERMINAL-FILE RECORD INTO LIST-OF-TERMINALS(COUNTER)
   217                 AT END
   218                     SET END-OF-TERMINAL-LIST TO TRUE
   219                     SUBTRACT 1 FROM COUNTER
   220                     MOVE COUNTER TO NO-OF-TERMINALS.
   221             ADD 1 TO COUNTER.
   222
   223         ACQUIRE-AND-INVITE-TERMINALS.
   224             ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE FILE.
   225             WRITE MULTIPLE-REC
   226                 FORMAT IS "SIGNON"
   227                 TERMINAL IS LIST-OF-TERMINALS(COUNTER).
                                                                      Más....
Depuración... _____________________________________________________________
_____________________________________________________________________________
F3=Fin programa  F6=Añadir/Borrar punto int. F10= Paso F11= Visualizar variable
F12=Reanudar     F13=Trabajar con punt. int. módulo   F24=Más teclas
COUNTER = 89.
```

Figura 44. Visualización de una variable utilizando el mandato de depuración EVAL

Visualización de variables como valores hexadecimales

Se puede utilizar el mandato de depuración EVAL para visualizar el valor de una variable en formato hexadecimal.Para visualizar una variable en formato hexadecimal, teclee:

```
EVAL nombre-variable: x 32
```

en la línea de mandatos de depuración.*Nombre-variable* es el nombre de la variable que desea visualizar en formato hexadecimal.`'x'` especifica que la variable va a visualizarse en formato hexadecimal y `'32'` indica que se va a visualizar un vuelco de 32 bytes después del principio de la variable. El valor hexadecimal de la variable aparece en la pantalla Evaluar expresión como se indica en la Figura 45 en la página 143. Si no se especifica la longitud a continuación de `'x'`, el tamaño de la variable se utiliza como longitud. Siempre se visualiza un mínimo de 16 bytes. Si la longitud de la variable es inferior a 16 bytes, el espacio restante se rellena con ceros hasta que se alcanza el límite de 16 bytes.

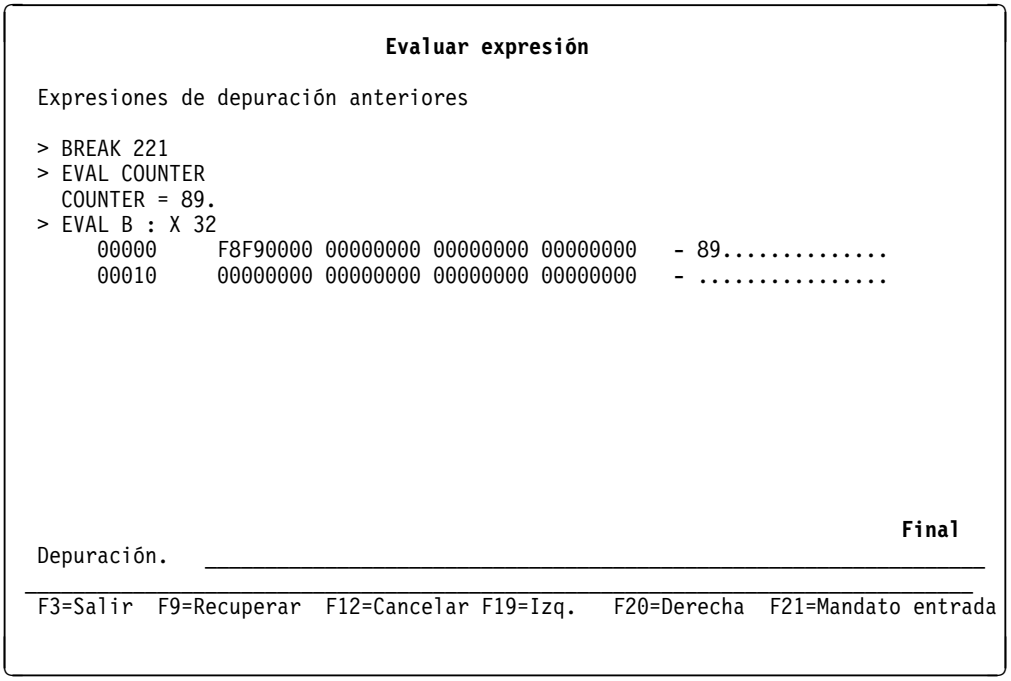


Figura 45. Visualización del valor hexadecimal de una variable utilizando el mandato de depuración EVAL

Visualización de una subserie de una variable de serie de caracteres

El depurador fuente ILE no da soporte a la sintaxis de modificación de referencias ILE COBOL/400. En su lugar, puede utilizar el operador %SUBSTR con el mandato EVAL para visualizar una subserie de una variable de serie de caracteres. El operador %SUBSTR obtiene la subserie de una variable de serie de caracteres a partir de una posición del elemento inicial para determinado número de elementos.

Nota: El depurador fuente ILE no da soporte a la sintaxis de modificación de referencias de COBOL para el manejo de subseries. Es necesario utilizar el operador %SUBSTR del depurador fuente ILE para manejar subseries.

La sintaxis del operador %SUBSTR es la siguiente:

```
%SUBSTR(identificador elemento-inicial número-de-elementos)
```

Donde, *identificador* debe ser una variable de serie de caracteres y *elemento-inicial* y *número-de-elementos* deben ser literales enteros positivos distintos de cero. *Identificador* puede ser una variable indexada, subindexada o calificada. *Elemento-inicial + número-de-elementos - 1* no puede ser superior al número total de elementos de *identificador*.

Por ejemplo, puede obtener los 10 primeros elementos de una serie de caracteres de 20 elementos utilizando %SUBSTR(char20 1 10). Puede obtener los últimos 5 elementos de una serie de caracteres de 8 elementos utilizando %SUBSTR(char8 4 5). En el caso de un elemento DBCS o DBCS editado, elemento hace referencia a un carácter DBCS (es decir, un carácter de dos bytes).

Se puede utilizar un operador %SUBSTR para asignar una subserie de una variable de serie de caracteres a otra variable o subserie de una variable. Los

datos se copian de la variable fuente a la variable destino de izquierda a derecha.Cuando la variable fuente o la variable destino o ambas son subseries, el operando es la parte de subserie de la variable de serie de caracteres, no la variable de serie de caracteres entera. Cuando los tamaños de las variables fuente y destino son diferentes, se aplican las normas de relleno y truncamiento siguientes:

- Si la longitud de la variable fuente es mayor que la variable de destino, la serie de caracteres se trunca en la longitud de la variable destino.
- Si la longitud de la variable fuente es menor que longitud de la variable de destino, la serie de caracteres se justifica por la izquierda en la variable de destino y las posiciones restantes se rellenan con blancos.
- Si la longitud de la variable fuente es igual a la longitud de la variable destino, los dos valores serán copias exactas después de la asignación.

Nota: Es posible utilizar una subserie de la misma variable de serie de caracteres en las variables fuente y destino; sin embargo, si alguna parte de la serie destino se solapa a la serie fuente, se producirá un error.

La Figura 46 muestra ejemplos de cómo puede utilizarse el operador %SUBSTR.

```
                          Evaluar expresión

Expresiones de depuración anteriores

> EVAL CHAR10
  CHAR10 = '10CHARLONG'
> EVAL CHARA
  CHARA = 'A'
> EVAL CHARA = %SUBSTR(CHAR10 3 5)
  CHARA = 'C'
> EVAL %SUBSTR(CHAR10 1 2) = 'A'
  CHAR10 = 'A CHARLONG'
> EVAL %SUBSTR(CHAR10 1 2) = 'XYZ'
  CHAR10 = 'XYCHARLONG'
> EVAL %SUBSTR(CHAR10 7 4) = 'ABCD'
  CHAR10 = 'XYCHARABCD'
> EVAL %SUBSTR(CHAR10 1 2) = %SUBSTR(CHAR10 7 4)
  CHAR10 = 'ABCHARABCD'

                                                                      Final
Depuración.  ___________________________________________________________________
_______________________________________________________________________________
F3=Salir  F9=Recuperar  F12=Cancelar F19=Izq.   F20=Derecha  F21=Mandato entrada
```

Figura 46. Visualización de una subserie de una variable de serie de caracteres utilizando el operador %SUBSTR

Visualización de registros, elementos de grupo y matrices

Puede utilizar el mandato de depuración EVAL para visualizar estructuras, registros y matrices. A menos que el registro, elemento de grupo o matriz se encuentre en la línea actual, primero debe calificar el registro, elemento de grupo o matriz que desea visualizar identificando el número de línea donde se encuentra utilizando el mandato de depuración QUAL. Consulte el apartado “Visualización de variables y expresiones” en la página 141 para obtener una descripción de cómo utilizar el

mandato de depuración QUAL. Para visualizar un registro, elemento de grupo o matriz, teclee:

```
EVAL nombre-datos
```

en la línea de mandatos de depuración.*Nombre-datos* es el nombre del registro, elemento de grupo o matriz que desea visualizar.El valor del registro, elemento de grupo o matriz aparecerá en la pantalla Evaluar expresión.

El ejemplo siguiente muestra cómo visualizar el contenido de un elemento de grupo ILE COBOL/400.

```
01  ACCOUNT.
    02  NUMBER              PIC 9(5).
    02  FULL-NAME.
        03  LAST-NAME       PIC X(20).
        03  FIRST-NAME      PIC X(10).
```

Para visualizar el contenido del elemento de grupo *ACCOUNT*, teclee:

```
EVAL ACCOUNT
```

en la línea de mandatos de depuración. El contenido actual del elemento de grupo *ACCOUNT* aparecerá en la pantalla Evaluar expresión como se muestra en la Figura 47.

Para visualizar el contenido de un único componente del elemento de grupo *ACCOUNT*, como, por ejemplo, el elemento FIRST-NAME OF FULL-NAME OF ACCOUNT, teclee:

```
EVAL FIRST-NAME OF FULL-NAME OF ACCOUNT
```

en la línea de mandatos de depuración. El contenido actual del elemento *FIRST-NAME OF FULL-NAME OF ACCOUNT* aparecerá en la pantalla Evaluar expresión como se muestra en la Figura 47. Pulse Intro para volver a la pantalla Visualizar fuente del módulo. También puede visualizar elementos utilizando nombres calificados parcialmente siempre que el nombre esté suficientemente calificado para resolver posibles ambigüedades de nombre.

```
                          Evaluar expresión

Expresiones de depuración anteriores

> EVAL ACCOUNT
  NUMBER OF ACCOUNT = 12345
  LAST-NAME OF FULL-NAME OF ACCOUNT = 'SMITH               '
  FIRST-NAME OF FULL-NAME OF ACCOUNT = 'JOHN      '
> EVAL FIRST-NAME OF FULL-NAME OF ACCOUNT
  FIRST-NAME OF FULL-NAME OF ACCOUNT = 'JOHN      '
```

Figura 47. Visualización de un elemento de grupo utilizando el mandato de depuración EVAL

El ejemplo siguiente muestra cómo visualizar el contenido de una matriz ILE COBOL/400.

```
05  A       PIC X(5) OCCURS 5 TIMES.
```

Para visualizar el contenido de la matriz *A*, teclee:

```
EVAL A
```

en la línea de mandatos de depuración. El contenido actual de la matriz *A* aparecerá en la pantalla Evaluar expresión como se muestra en la Figura 48 en la página 146.

Para visualizar el contenido de un rango de elementos de la matriz *A*, teclee:

```
EVAL A(2..4)
```

en la línea de mandatos de depuración. El contenido actual de los elementos A(2), A(3) y A(4) de la matriz *A* aparecerá en la pantalla Evaluar expresión como se muestra en la Figura 48.

Para visualizar el contenido de un único elemento de la matriz *A* como, por ejemplo, el elemento A(4), teclee:

```
EVAL A(4)
```

en la línea de mandatos de depuración. El contenido actual del elemento *A(4)* aparecerá en la pantalla Evaluar expresión como se muestra en la Figura 48. Pulse F3 (Salir) para volver a la pantalla Visualizar fuente del módulo.

Nota: El valor de subíndice especificado en el mandato de depuración EVAL sólo puede ser un valor numérico.Por ejemplo, A(4) se acepta pero A(I+2) o A(2*3) no se aceptan.

```
                          Evaluar expresión

Expresiones de depuración anteriores

> EVAL A
  A(1) = 'ONE  '
  A(2) = 'TWO  '
  A(3) = 'THREE'
  A(4) = 'FOUR '
  A(5) = 'FIVE '
> EVAL A(2..4)
  A(2) = 'TWO  '
  A(3) = 'THREE'
  A(4) = 'FOUR '
> EVAL A(4)
  A(4) = 'FOUR '
```

Figura 48. Visualización de una matriz utilizando el mandato de depuración EVAL

Modificación del valor de las variables

Puede modificar el valor de las variables utilizando el mandato EVAL con un operador de asignación. A menos que la variable esté en la línea actual, primero debe calificar la variable que desea modificar identificando el número de línea donde se encuentra utilizando el mandato de depuración QUAL. Consulte el apartado “Visualización de variables y expresiones” en la página 141 para obtener una descripción de cómo utilizar el mandato de depuración QUAL. Para cambiar el valor de la variable, teclee:

```
EVAL nombre-variable = valor
```

en la línea de mandatos de depuración.*Nombre-variable* es el nombre de la variable que desea modificar y *valor* es un identificador, literal o valor constante que desea asignar a la variable *nombre-variable*. Por ejemplo,

```
EVAL COUNTER=3
```

modifica el valor de *COUNTER* a 3 y muestra

```
COUNTER=3 = 3
```

en la línea de mensajes de la pantalla Visualizar fuente del módulo.

Puede utilizar el mandato de depuración EVAL para asignar datos numéricos, alfabéticos, alfanuméricos, DBCS, booleanos y de coma flotante a las variables siempre que éstos coincidan con la definición de la variable.

Nota: Si el valor que se asigna a la variable utilizando el mandato de depuración EVAL no coincide con la definición de la variable, se emitirá un mensaje de aviso y el valor de la variable no se modificará.

Si el valor que se asigna a la variable es una serie de caracteres, se aplican las normas siguientes:

- La longitud de la serie de caracteres que se asigna a la variable debe ser igual a la longitud de la variable.
- Si la longitud de la serie de caracteres que se asigna a la variable es menor que la longitud de la variable, la serie de caracteres se justifica por la izquierda y las posiciones restantes se rellenan con blancos.
- Si la longitud de la serie de caracteres que se asigna a la variable es mayor que la longitud de la variable, la serie de caracteres se trunca por la longitud de la variable.

A continuación se presentan algunos ejemplos que muestran cómo se pueden asignar distintos tipos de datos a variables utilizando el mandato de depuración EVAL.

```
EVAL COUNTER=3                    (COUNTER es una variable numérica)

EVAL COUNTER=LIMIT                (LIMIT es otra variable numérica)

EVAL END-OF-FILE='1'              (END-OF-FILE es una variable booleana)

EVAL BOUNDARY=x'C9'               (BOUNDARY es una variable alfanumérica)

EVAL COMPUTER-NAME='AS400'        (COMPUTER-NAME es una variable alfanumérica)

EVAL INITIALS=%SUBSTR(NAME 17 3)  (INITIALS y NAME son variables alfanuméricas)

EVAL DBCS-NAME= G'OEKIK2K3OF'       (K1K2K3 son caracteres DBCS)

EVAL LONG-FLOAT(3) = -30.0E-3
        (LONG-FLOAT es una matriz de 3 elementos de datos de coma flotante
         de precisión doble - COMP-2)

EVAL SHORT-FLOAT = 10
        (SHORT-FLOAT es un elemento de datos de coma flotante de precisión
         simple - COMP-1)
```

Nota: No se puede asignar una constante figurativa a una variable utilizando el mandato de depuración EVAL. El mandato de depuración EVAL no da soporte a las constantes figurativas.

Equivalencia entre un nombre y una variable, expresión o mandato

Se puede utilizar el mandato de depuración EQUATE para equiparar un nombre con una variable, expresión o mandato de depuración para su uso abreviado. Se puede utilizar un nombre solo o incluido en otra expresión. Si lo utiliza incluido en otra expresión, el valor del nombre se determina antes de evaluar la expresión. Estos nombres permanecen activos hasta que la sesión de depuración finaliza o el nombre se elimina.

Para equiparar un nombre con una variable, expresión o mandato de depuración, teclee:

```
EQUATE nombre-abreviado definición
```

en la línea de mandatos de depuración.*Nombre-abreviado* es el nombre que desea equiparar con una variable, expresión o mandato de depuración y *definición* es la variable, expresión o mandato de depuración que está equiparando al nombre.

Por ejemplo, para definir un nombre abreviado llamado *DC* que visualiza el contenido de la variable *COUNTER*, teclee:

```
EQUATE DC EVAL COUNTER
```

en la línea de mandatos de depuración.Ahora, siempre que se teclee *DC* en la línea de mandatos de depuración, se llevará a cabo el mandato EVAL *COUNTER*.

El número máximo de caracteres que pueden teclearse en un mandato EQUATE es 144. Si no se proporciona ninguna definición y un mandato EQUATE anterior definía el nombre, se eliminará la definición anterior. Si el nombre no se definió anteriormente, aparece un mensaje de error.

Para ver los nombres que se han definido con el mandato de depuración EQUATE para una sesión de depuración, teclee:

```
DISPLAY EQUATE
```

en la línea de mandatos de depuración. Aparecerá una lista de los nombres activos en la pantalla Evaluar expresión.

Soporte de idioma para el depurador fuente ILE

Cuando se trabaja con Soporte de idioma para el depurador fuente ILE, se aplican las condiciones siguientes:

- Cuando se visualiza una vista en la pantalla Visualizar fuente del módulo, el depurador fuente ILE convierte todos los datos al CCSID del trabajo de depuración.
- Cuando se asignan literales a variables, el depurador fuente ILE no realiza la conversión del CCSID en literales entrecomillados (por ejemplo, `'abc'`). Además, los literales entrecomillados son sensibles a las mayúsculas y minúsculas.

Cuando se trabaja con la vista fuente, si el CCSID del archivo fuente del que se obtiene la vista fuente es diferente del CCSID del objeto de módulo, es posible que el depurador fuente ILE no reconozca un identificador ILE COBOL/400 que contenga caracteres variantes.

Si se da alguna de estas condiciones:

- El CCSID del trabajo de depuración es 290, 930 ó 5026 (Katakana japonés)
- La página de códigos de la descripción de dispositivos que se utiliza para la depuración es 290, 930 ó 5026 (Katakana japonés)

los mandatos de depuración, las funciones y los literales hexadecimales deberían entrarse en letras mayúsculas. Por ejemplo,

```
BREAK 16 WHEN var=X'A1B2'
EVAL var:X
```

Sin embargo, cuando se depuran objetos de módulo ILE COBOL/400, ILE RPG/400 o ILE CL, el depurador fuente convierte los nombres de identificador de los mandatos de depuración a mayúsculas y, por lo tanto, pueden visualizarse de forma diferente.

ILE COBOL/400 Consideraciones sobre la programación

Capítulo 8. Trabajo con elementos de datos

Este capítulo explica cómo se puede trabajar con datos numéricos ILE COBOL/400 y cómo se pueden representar los datos numéricos y las operaciones aritméticas de la forma más eficaz. Los temas son los siguientes:

Visualización general de los números en ILE COBOL/400 (cláusula PICTURE)

En general, puede visualizar los datos numéricos de ILE COBOL/400 de forma similar a los datos de una serie de caracteres: como series de posiciones de dígitos decimales. Sin embargo, los elementos numéricos cuentan con propiedades especiales como por ejemplo un signo aritmético.

Definición de elementos numéricos

Para definir elementos numéricos, debe utilizar el carácter "9" en la descripción de datos, que representa los dígitos decimales del número, en vez de utilizar una "x" como para los elementos alfanuméricos:

```
05  COUNT-X         PIC 9(4)   VALUE 25.
05  CUSTOMER-NAME   PIC X(20)  VALUE "González".
```

Puede codificar un máximo de 18 dígitos en la cláusula PICTURE, además de otros caracteres que tienen una importancia especial. La "s" del siguiente ejemplo otorga un signo al valor.

```
05  PRICE           PIC S99V99.
```

El campo puede contener un valor positivo o negativo. La "V" indica la posición de una coma decimal implícita. La "S" y la "V" no se cuentan en el tamaño del elemento y tampoco requieren posiciones de almacenamiento adicionales, a no ser que el elemento se codifique como USAGE DISPLAY y contenga la cláusula SIGN IS SEPARATE. Una excepción la constituyen los datos de coma flotante internos (COMP-1 y COMP-2), para los que no existe ninguna cláusula PICTURE. Por ejemplo, un elemento de datos de coma flotante interno se define de la siguiente forma:

```
05  GROMMET-SIZE-DEVIATION    USAGE COMP-1    VALUE 02.35E-5
```

Para obtener información sobre cómo controlar la manera en que el compilador maneja los elementos de datos de coma flotante, consulte la descripción de

*FLOAT y *NOFLOAT en la página 38 y en el apartado “Utilización de la instrucción PROCESS para especificar opciones de compilador” en la página 47.

Posición de signo separada (para mejorar la portabilidad)

Si piensa transferir el programa o datos a una máquina distinta, es posible que desee codificar el signo como un dígito separado en el almacenamiento:

```
05  PRICE            PIC S99V9   SIGN IS LEADING, SEPARATE.
```

Esto le asegura que la convención que utiliza la máquina para almacenar el signo no separado no provocará resultados extraños al utilizar una máquina con una convención distinta.

Posiciones adicionales para símbolos visualizables (edición numérica)

También puede definir elementos numéricos utilizando ciertos símbolos de edición (como por ejemplo comas decimales, puntos y el signo de dólar) para facilitar la lectura y la comprensión de los datos cuando éstos se visualicen o impriman en un informe. Por ejemplo:

```
05 PRICE           PIC   9(5)V99.
05 EDITED-PRICE    PIC  $ZZ.ZZ9V99.
   .
   .
   .
   MOVE PRICE to EDITED-PRICE
   DISPLAY EDITED-PRICE
```

Si el contenido de `PRICE` es 0150099 (que representa el valor 1.500,99), se visualizaría `$ 1.500,99` después de ejecutar el código.

Cómo utilizar elementos de edición numérica como números

Los elementos de edición numérica están clasificados como elementos de datos alfanuméricos, no como números. Por consiguiente, no pueden funcionar como operandos en expresiones aritméticas ni en las instrucciones ADD, SUBTRACT, MULTIPLY, DIVIDE y COMPUTE.

Los elementos de edición numérica puede convertirse en elementos numéricos y de edición numérica. En el siguiente ejemplo, se *des-edita* el elemento de edición numérica y se traslada su valor al elemento de datos numérico.

```
MOVE EDITED-PRICE to PRICE.
DISPLAY PRICE.
```

Si estas dos instrucciones siguieran de forma inmediata a las sentencias que aparecían en el ejemplo anterior, `PRICE` se visualizaría como 0150099, representando el valor 1.500,99.

Para obtener información más completa sobre las descripciones de datos para datos numéricos, consulte el manual *ILE COBOL/400 Reference*.

Representación computacional de datos (cláusula USAGE)

Es posible controlar la forma en que el sistema almacena los elementos de datos numéricos internamente codificando la cláusula USAGE en las entradas de descripción de datos. Los datos numéricos que utilice en el programa tendrán unos de los formatos disponibles con ILE COBOL/400:

- Decimal externo (USAGE DISPLAY)
- Decimal interno (USAGE PACKED-DECIMAL o COMP-3)
- Binario (USAGE BINARY o COMP-4)
- Coma flotante externo (USAGE DISPLAY)
- Coma flotante interno (USAGE COMP-1, USAGE COMP-2)

COMP-4 es sinónimo de BINARY; COMP y COMP-3 son sinónimos de PACKED-DECIMAL.

Con independencia de la cláusula USAGE que utilice para controlar la representación interna del valor que el sistema genera, puede utilizar las convenciones y el valor decimal aplicables a la cláusula PICTURE en la cláusula VALUE, con excepción de los datos de coma flotante.

Elementos decimales externos (USAGE DISPLAY)

Al codificar USAGE DISPLAY u omitir la cláusula USAGE, cada posición (o byte) de almacenamiento contiene un dígito decimal. Esto corresponde al formato utilizado para imprimir o visualizar la salida e indica que los elementos se almacenan de forma visualizable.

Para qué sirven los elementos USAGE DISPLAY

Los elementos decimales externos están pensado principalmente para la recepción y envío de números entre programas y archivos, terminales e impresoras. Sin embargo, también es posible utilizar elementos de decimales externos como operandos y receptores durante el proceso aritmético del programa y a menudo es conveniente programar de esta forma.

Cuándo debería utilizarlos para operaciones aritméticas

Si el programa realiza operaciones aritméticas intensas con frecuencia y la eficacia constituye una prioridad, es posible que desee utilizar uno de los tipos de datos numéricos computacionales de ILE COBOL/400 para los elementos de datos que utilice en aritmética.

El sistema tiene que convertir de forma automática los números visualizables a la representación *interna* del valor numérico antes de que éstos puedan utilizarse en operaciones aritméticas. Por este motivo, a menudo resulta más eficaz definir los elementos de datos como elementos computacionales desde el principio y no como elementos DISPLAY. Por ejemplo:

```
05 COUNT-X          PIC S9V9(5)   USAGE COMP   VALUE 3.14159.
```

Elementos decimales internos (USAGE PACKED-DECIMAL o COMP-3)

El formato decimal empaquetado ocupa 1 byte de almacenamiento cada dos posiciones decimales de la descripción PICTURE, con excepción del byte situado más a la derecha, que sólo contiene 1 dígito y el signo. Para utilizar este formato de forma más eficaz, debe codificar un número impar de dígitos en la descripción PICTURE, de forma que se utilice completamente el byte situado más a la izquierda. En operaciones aritméticas, el formato decimal empaquetado se trata como un número de coma fija.

Razones para la utilización de decimales empaquetados

Formato decimal empaquetado:

- Requiere un almacenamiento inferior por dígito que el necesario con el formato DISPLAY.
- Es más adecuado para la alineación decimal que el formato binario.
- Las conversiones a y desde el formato DISPLAY son más sencillas que con el formato binario.
- Es apropiado para contener operandos o resultados aritméticos.

Elementos binarios (USAGE BINARY o COMP-4)

El formato binario ocupa 2, 4 u 8 bytes de almacenamiento y en aritmética se trata como un número de coma fija en el que el bit más a la izquierda es el signo operativo. Para datos binarios con inversión de byte, el bit del signo es el bit situado más a la izquierda del byte situado más a la derecha.

Cuánto almacenamiento ocupa BINARY

Un descripción PICTURE con un número que no sobrepase los 4 dígitos decimales ocupa 2 bytes; si tiene entre 5 y 9 dígitos decimales ocupa 4 bytes y entre 10 y 18 dígitos decimales ocupa 8 bytes.

Los elementos binarios son adecuados para subíndices y para las posiciones de inicio y de longitud de modificaciones de referencia.

Sin embargo, el formato BINARY no es adecuado para la alineación decimal, por lo que ILE COBOL/400 convierte los número BINARY de expresiones aritméticas a formato PACKED DECIMAL. Por consiguiente, es preferible utilizar el formato PACKED DECIMAL para las expresiones aritméticas.

También es preferible la utilización del formato PACKED DECIMAL frenta a BINARY al convertir números a formato de visualización. La conversión de un número de formato BINARY a formato DISPLAY es más difícil que su conversión de formato PACKED DECIMAL a formato DISPLAY.

Truncamiento de datos binarios (Opción del compilador *STDTRUNC)

Utilice las opciones del compilador *STDTRUNC y *NOSTDTRUNC (descritas en la página 35) para indicar la forma en que deben truncarse los datos de tipo BINARY y COMP-4.

Elementos de coma flotante interno (USAGE COMP-1 y COMP-2)

COMP-1 hace referencia al formato breve de coma flotante (de precisión simple) y COMP-2 hace referencia al formato largo de coma flotante (de precisión doble), que ocupan, respectivamente, 4 y 8 bytes de almacenamiento. El bit situado más a la izquierda contiene el signo; los siete bits que aparecen a continuación contienen el exponente y los 3 ó 7 bytes restantes contienen la mantisa.

En OS/400, los elementos de datos COMP-1 y COMP-2 se representan en formato IEEE.

No se permite una cláusula PICTURE en la descripción de datos de elementos de datos de coma flotante aunque sí que puede proporcionar un valor inicial utilizando un literal de coma flotante en la cláusula VALUE:

```
05 COMPUTE-RESULT     USAGE COMP-1   VALUE 06.23E-24.
```

Las características de las conversiones entre formato de coma flotante y otros formatos numéricos se discuten en el apartado “Conversiones de formato de datos” en la página 158.

El formato de coma flotante está indicado para operandos y resultados aritméticos y para conseguir un alto nivel de precisión en operaciones aritméticas.

Para obtener información más completa sobre las descripciones de datos para datos numéricos, consulte el manual *ILE COBOL/400 Reference*.

Elementos de coma flotante externos (USAGE DISPLAY)

Los números visualizables codificados en formato de coma flotante se denominan elementos **de coma flotante externos**. Igual que ocurre con los elementos decimales externos, puede definir elementos de coma flotante externos de forma explícita con USAGE DISPLAY o de forma implícita omitiendo la cláusula USAGE.

En el siguiente ejemplo, `COMPUTE-RESULT` está definido de forma explícita como un elemento de coma flotante externo. Cada byte de almacenamiento contiene un carácter (excepto V).

```
05 COMPUTE-RESULT     PIC -9V(9)E-99.
```

La cláusula VALUE no se permite en la descripción de datos para elementos de coma flotante externos. Además, el signo menos (–) no significa que la mantisa y el exponente siempre serán números negativos, sino que, cuando se visualice el signo, éste aparecerá como un blanco para los valores positivos y como un menos para los negativos. Si se ha utilizado un signo más (+), los valores positivos se visualizarán con un signo más y los negativos con un signo menos.

Igual que ocurre con números decimales externos, los números de coma flotante externos deben convertirse (de forma automática, mediante el compilador) a una representación interna del valor numérico antes de que se pueda operar con los mismos. Los números de coma flotante externos siempre se convierten a formato largo de coma decimal interno.

Conversiones de formato de datos

Cuando el código del programa implica la interacción de elementos con formatos de datos distintos, el compilador convierte estos elementos:

- De forma temporal, para comparaciones y operaciones aritméticas
- De forma permanente, para la asignación al receptor en una instrucción MOVE o COMPUTE.

Significado de una conversión

Una conversión es en realidad el traslado de un valor de un elemento de datos a otro. El compilador realiza las conversiones necesarias durante la ejecución de las operaciones aritméticas y comparaciones siguiendo las mismas reglas que se utilizan para instrucciones MOVE y COMPUTE. Las reglas para realizar traslados se definen en *ILE COBOL/400 Reference*. Cuando es posible, el compilador realiza un traslado que le permita conservar el *valor* numérico en lugar de realizar un traslado directo dígito a dígito. (Para obtener más información acerca del truncamiento y la pérdida de dígitos significativos, consulte el apartado "Conversiones y precisión".)

Una conversión requiere su tiempo

Una conversión suele necesitar almacenamiento y tiempo de proceso adicional, ya que es preciso el traslado de los datos a un área de trabajo interna y la conversión de éstos antes de realizar la operación. Es posible que el resultado deba volverse a trasladar al área de trabajo para realizar una nueva conversión.

Conversiones y precisión

Las conversiones entre formatos de datos de coma fija (decimales externos, decimales empaquetados y binarios) se completan sin pérdida de precisión siempre que los campos de destino puedan contener todos los dígitos del operando fuente.

Conversiones en las que es posible la pérdida de datos

Es posible una pérdida de precisión en las conversiones entre formatos de datos de coma fija y formatos de datos de coma flotante (coma flotante breve, coma flotante largo y coma flotante interno). Estas conversiones se producen durante las evaluaciones aritméticas que tienen operandos de coma fija y de coma flotante. (Como tanto los elementos de coma fija como los elementos de coma flotante externos tienen características decimales, la referencia a elementos de coma fija en los siguientes ejemplos también incluyen elementos de coma flotante externos a menos que se especifique lo contrario).

Al realizar una conversión de formato de coma fija a formato de coma flotante externo, los números de coma fija en base 10 se convierten al sistema numérico que se utiliza internamente, base 16.

Aunque el compilador convierte el formato breve en largo para las comparaciones, se utilizan ceros para rellenar el número breve.

Cuando se traslada un elemento de datos USAGE COMP-1 a un elemento de datos de coma fija que tiene 6 o más dígitos, el elemento de datos de coma fija sólo recibirá 6 dígitos significativos y el resto de dígitos será cero.

Conversiones que mantienen la precisión: Si se traslada un elemento de datos de coma fija que tiene 6 dígitos o menos a un elemento de datos USAGE COMP-1 y se devuelve al elemento de datos de coma fija, se recupera el valor original.

Si se traslada un elemento de datos USAGE COMP-1 a un elemento de datos de coma fija que tiene 6 o más dígitos y se devuelve al elemento de datos USAGE COMP-1, se recupera el valor original.

Si se traslada un elemento de datos de coma fija que tiene 15 dígitos o menos a un elemento de datos USAGE COMP-2 y éste se devuelve al elemento de datos de coma fija, se recupera el valor original.

Si se traslada un elemento de datos USAGE COMP-2 a un elemento de datos de coma fija (no de coma flotante externo) que tiene 18 dígitos y se devuelve al elemento de datos USAGE COMP-2, se recupera el valor original.

Conversiones que producen redondeo: Si se traslada un elemento de datos USAGE COMP-1, un elemento de datos USAGE COMP-2, un elemento de datos de coma flotante externo o un literal de coma flotante a un elemento de datos de coma fija, se produce el redondeo de la posición de orden inferior del elemento de datos destino.

Si se traslada un elemento de datos USAGE COMP-2 a un elemento de datos USAGE COMP-1, se produce el redondeo de la posición de orden inferior del elemento de datos destino.

Si se traslada un elemento de datos de coma fija a un elemento de datos de coma flotante externo en el que el PICTURE del elemento de datos de coma fija contiene más posiciones de dígito que el PICTURE del elemento de datos de coma flotante, se produce el redondeo de la posición de orden inferior del elemento de datos destino.

Representación y proceso del signo

La representación del signo afecta al proceso y a la interacción de los datos numéricos.

Dado X'sd', donde s es la representación del signo y d representa el dígito, las representaciones de signo válidas para decimales externos (USAGE DISPLAY que no tengan la cláusula SIGN IS SEPARATE) son las siguientes:

Positivas: A, C, E y F.

Negativas: B y D.

Los signos que se generan de forma interna son F para un valor positivo o sin signo y D para uno negativo.

Dado X'ds', donde d representa el dígito y s es la representación del signo, las representaciones de signo válidas para datos decimales internos (USAGE PACKED-DECIMAL) ILE COBOL/400 son las siguientes:

Positivas: A, C, E y F.

Negativas: B y D.

Los signos que se generan de forma interna son F para un valor positivo o sin signo y D para uno negativo.

Con la opción *CHGPOSSN del compilador

La opción *CHGPOSSGN del compilador ILE COBOL/400 afecta al proceso del signo para datos decimales internos y decimales externos. *CHGPOSSGN no tiene ningún efecto en datos binario ni en datos de coma flotante. Para obtener más información, lea la discusión sobre *CHGPOSSN en la página 38.

Los signos positivos generados por el compilador, que normalmente son F, se convierten en C en MOVE y en instrucciones aritméticas, así como la cláusula VALUE.

La opción *CHGPOSSGN del compilador es menos eficaz que la opción por omisión, *NOCHGPOSSGN, y debería utilizarse exclusivamente al compartir datos con MVS, VM y otros sistemas que tengan signos preferentes distintos.

Comprobación de datos incompatibles (prueba de clase numérica)

El compilador asume que los valores que el usuario proporciona para un elemento de datos son válidos para la cláusula PICTURE y USAGE del elemento y asigna el valor proporcionado sin comprobar su validez. Cuando se proporciona a un elemento un valor incompatible con su descripción de datos, las referencias a este elemento en la PROCEDURE DIVISION no estarán definidas y será imposible prever el resultado.

Con frecuencia, los valores se pasan al programa y se asignan a elementos que tienen descripciones de datos incompatibles para estos valores. Por ejemplo, es posible que los datos no numéricos se muevan o pasen a un campo del programa que esté definido como un número sin signo. En cualquier caso, los datos que contienen estos campos no son válidos. Asegúrese de que el contenido del elemento de datos sea el adecuado para las cláusulas PICTURE y USAGE antes de utilizar el elemento de datos en otros pasos del proceso.

Cómo realizar una prueba de clase numérica

Es posible utilizar una prueba de clase numérica para realizar la validación de los datos. Por ejemplo:

```
LINKAGE SECTION.
01   COUNT-X     PIC 999.
   .
   .
   .
   PROCEDURE DIVISION USING COUNT-X.
     IF COUNT-X IS NUMERIC THEN DISPLAY "LOS DATOS SON VÁLIDOS".
   .
   .
   .
```

La prueba de clase numérica verifica el contenido de un elemento de datos en relación a un conjunto de valores válidos para PICTURE y USAGE del elemento de datos concreto. Por ejemplo, en un elemento decimal empaquetado se comprobarían los valores hexadecimales del X'0' al X'9' de las posiciones de los dígitos y se verificaría la existencia de un valor de signo válido en la posición de signo (tanto si la posición del signo está separada como si no).

Realización de operaciones aritméticas

Al ejecutar ILE COBOL/400 con ILE obtendrá diversas funciones para realizar operaciones aritméticas:

- Instrucciones ADD, SUBTRACT, MULTIPLY, DIVIDE y COMPUTE (tratadas en el apartado “COMPUTE y otras instrucciones aritméticas”).
- Expresiones aritméticas (tratadas en el apartado “Expresiones aritméticas” en la página 162).
- Funciones intrínsecas (tratadas en el apartado “Funciones numéricas intrínsecas” en la página 162).
- Servicios que pueden llamarse desde ILE (API)

  ILE proporciona varios grupos de API de enlace, disponibles para todos los compiladores ILE. Las API matemáticas incluyen CEE4SIFAC para calcular factores y CEESDCOS para calcular cosenos.

  Para obtener más información acerca de las API de enlace que es posible utilizar, consulte el manual *AS/400 System API Reference*.

Para obtener información más detallada sobre la sintaxis y la utilización de contructores de lenguaje ILE COBOL/400, consulte el manual *ILE COBOL/400 Reference*.

COMPUTE y otras instrucciones aritméticas

La práctica general consiste en utilizar la instrucción COMPUTE para la mayor parte de evaluaciones aritméticas en lugar de usar las instrucciones ADD, SUBTRACT, MULTIPLY y DIVIDE. Ello se debe a que a menudo es posible codificar una instrucción COMPUTE en lugar de varias instrucciones individuales.

La instrucción COMPUTE asigna el resultado de una expresión aritmética a un elemento de datos:

```
COMPUTE Z = A + B / C ** D - E
```

o a varios elementos de datos:

```
COMPUTE X Y Z = A + B / C ** D - E
```

Cuándo utilizar otras instrucciones aritméticas

Es posible que algunas operaciones aritméticas sean más intuitivas si utiliza otras instrucciones aritméticas. Por ejemplo:

```
ADD 1 TO INCREMENTO.
```

en vez de:

```
COMPUTE INCREMENTO = INCREMENTO + 1.
```

O bien,

```
SUBTRACT DÉBITO FROM SALDO.
```

en vez de:

```
COMPUTE SALDO = SALDO - DÉBITO.
```

O bien,

```
ADD 1 TO INCREMENTO-1, INCREMENTO-2, INCREMENTO-3.
```

en vez de:

```
COMPUTE INCREMENTO-1 = INCREMENTO-1 + 1
COMPUTE INCREMENTO-2 = INCREMENTO-2 + 1
COMPUTE INCREMENTO-3 = INCREMENTO-3 + 1
```

También es posible que prefiera utilizar la instrucción DIVIDE (con la expresión REMAINDER) para realizar divisiones en las que desee procesar un resto.

Expresiones aritméticas

En los ejemplo de COMPUTE que se han mostrado, todo lo que aparecería a la derecha del signo igual representa una expresión aritmética. Las expresiones aritméticas pueden estar formadas por un solo literal numérico, un solo elemento de datos numérico o una sola referencia de función intrínseca. También pueden estar formadas por varios de los anteriores conectados mediante operadores aritméticos. Estos operadores se evalúan en orden jerárquico.

Tabla 7. Evaluación de operadores

Operador	Significado	Orden de evaluación
Unario + o –	Signo algebraico	Primero
**	Potenciación	Segundo
/ o *	División o multiplicación	Tercero
Binario + o –	Adición o sustracción	Último

Los operadores del mismo nivel se evalúan de izquierda a derecha; sin embargo, puede utilizar paréntesis con estos operadores para modificar el orden de evaluación. Las expresiones entre paréntesis se evalúan antes que todos los operadores individuales. Los paréntesis, tanto si son necesarios como si no, facilitan la lectura del programa.

Además de utilizar expresiones aritméticas en las instrucciones COMPUTE, también puede utilizarlas en otros lugares en los que se permitan elementos de datos numéricos. Por ejemplo, puede utilizar expresiones aritméticas a modo de comparandos en condiciones de relación:

```
IF (A + B) > (C - D + 5) THEN...
```

Funciones numéricas intrínsecas

Las funciones intrínsecas pueden devolver un valor alfanumérico, DBCS o numérico.

Las funciones numéricas intrínsecas:

- Devuelven un valor numérico con signo.
- Se consideran elementos de datos numéricos temporales.
- Sólo pueden utilizarse en aquellos lugares en los que la sintaxis del lenguaje permite expresiones.
- Pueden ahorrarle tiempo, ya que no necesita proporcionar la operación aritmética para los diferentes tipos de cálculo comunes que realizan estas funciones.

Para obtener más información acerca de la aplicación práctica de las funciones intrínsecas, consulte el apartado “Ejemplos de funciones intrínsecas” en la página 164.

Tipos de funciones numéricas

Las funciones numéricas se clasifican en las siguientes categorías:

Enteras
: Aquellas que devuelven un entero.

Coma flotante
: Aquellas que devuelven un valor de coma flotante largo.

Las funciones numéricas disponibles en ILE COBOL/400 en cada una de estas categorías se enumeran en la Tabla 8.

Tabla 8. Tipos de datos que manejan las funciones numéricas

Enteros	De coma flotante
DATE-OF-INTEGER	ACOS
DATE-TO-YYYYMMDD	ASIN
DAY-OF-INTEGER	ATAN
DAY-TO-YYYYDDD	COS
INTEGER-OF-DATE	LOG
INTEGER-OF-DAY	LOG10
LENGTH	MEAN
ORD	MEDIAN
YEAR-TO-YYYY	NUMVAL
	NUMVAL-C
	SIN
	SQRT
	TAN

Anidamiento de funciones y expresiones aritméticas

Es posible anidar las funciones numéricas; es decir, se puede hacer referencia a una función como argumento de otra. Una función anidada se evalúa con independencia de la función exterior.

Como las funciones numéricas y las expresiones aritméticas tienen una categoría sintáctica similar, también es posible anidar una expresión aritmética como argumento de una función numérica:

```
COMPUTE X = FUNCTION MEAN (A, B, C / D).
```

En este ejemplo, sólo existen tres argumentos de función: A, B y la expresión aritmética (C / D).

Subíndice ALL y registros especiales

Otras dos funciones útiles de las funciones intrínsecas son el subíndice ALL y los registros especiales.

Puede hacer referencia a todos los elementos de una matriz como argumentos de una función utilizando el subíndice ALL. Esta función se utiliza con tablas.

Los registros especiales de tipo entero se permiten como argumentos siempre que es posible utilizar argumentos de enteros.

Ejemplos de funciones intrínsecas

Puede utilizar funciones intrínsecas para realizar distintos tipos de operaciones aritméticas, tal como se indica en la Tabla 9.

Tabla 9. Tipos de operaciones aritméticas que manejan las funciones numéricas

Manejo de números	Fecha/hora	Matemáticas	Estadística
LENGTH	CURRENT-DATE	ACOS	MEAN
NUMVAL	DATE-OF-INTEGER	ASIN	
NUMVAL-C	DAY-TO-YYYYDDD	ATAN	
	DATE-TO-YYYYMMDD	COS	
	DAY-OF-INTEGER	LOG	
	INTEGER-OF-DATE	LOG10	
	INTEGER-OF-DAY	SIN	
	WHEN-COMPILED	SQRT	
	YEAR-TO-YYYY	TAN	

Los siguientes ejemplos y las explicaciones que los acompañan muestran funciones intrínsecas de cada una de las categorías enumeradas en la tabla anterior.

Manejo general de números: Suponga que desea hallar la media aritmética de tres precios (representados como elementos alfanuméricos con signos de dólar), colocar este valor en un campo numéricos de un registro de salida y determinar la longitud del registro de salida. Para conseguirlo, podría utilizar NUMVAL-C (una función que devuelve el valor numérico de una serie alfanumérica) y la función MEAN:

```
01 X                PIC 9(2).
01 PRICE1           PIC X(8)     VALUE "$8000".
01 PRICE2           PIC X(8)     VALUE "$4000".
01 PRICE3           PIC X(8)     VALUE "$6000".
01 OUTPUT-RECORD.
   05 PRODUCT-NAME   PIC X(20).
   05 PRODUCT-NUMBER PIC 9(9).
   05 PRODUCT-PRICE  PIC 9(6).
   .
   .
   .
 PROCEDURE DIVISION.
   COMPUTE PRODUCT-PRICE =
       FUNCTION MEAN (FUNCTION NUMVAL-C(PRICE1)
                      FUNCTION NUMVAL-C(PRICE2)
                      FUNCTION NUMVAL-C(PRICE3)).
       COMPUTE X = FUNCTION LENGTH(OUTPUT-RECORD).
```

Adicionalmente, para asegurarse de que el contenido de PRODUCT-NAME aparezca en mayúsculas, podría utilizar la instrucción siguiente:

```
MOVE FUNCTION UPPER-CASE(PRODUCT-NAME) TO PRODUCT-NAME.
```

Fecha y hora: El ejemplo siguiente muestra cómo calcular una fecha a 90 días a partir de la actual. Los ochos primeros caracteres que devuelve la función CURRENT-DATE representan la fecha en formato de 4 dígitos para el año, 2 dígitos para el mes y 2 dígitos para el día (YYYYMMDD). En el ejemplo, la fecha se convierte a un valor entero. Entonces se añade 90 a este valor y el entero vuelve a convertirse de nuevo a formato YYYYMMDD.

```
01 YYYYMMDD              PIC 9(8).
01 INTEGER-FORM          PIC S9(9).
   .
   .
   .
   MOVE FUNCTION CURRENT-DATE(1:8) TO YYYYMMDD.
   COMPUTE INTEGER-FORM = FUNCTION INTEGER-OF-DATE(YYYYMMDD).
   ADD 90 TO INTEGER-FORM.
   COMPUTE YYYYMMDD = FUNCTION DATE-OF-INTEGER(INTEGER-FORM).
   DISPLAY 'Fecha de vencimiento: ' YYYYMMDD.                                  i
```

Matemáticas: La siguiente instrucción ILE COBOL/400 demuestra cómo anidar funciones intrínsecas, cómo convertir argumentos en expresiones aritméticas y cómo realizar de forma sencilla cálculos matemáticos que antes eran complejos:

```
COMPUTE Z = FUNCTION LOG(FUNCTION SQRT (2 * X + 1))
```

Estadística: Las funciones intrínsecas también facilitan el cálculo de información estadística en datos. Supongamos que está analizando los impuestos de diferentes ciudades y desea calcular media aritmética.

```
01 TAX-S               PIC 99V999 VALUE .045.
01 TAX-T               PIC 99V999 VALUE .02.
01 TAX-W               PIC 99V999 VALUE .035.
01 TAX-B               PIC 99V999 VALUE .03.
01 MEAN-TAX            PIC 99V999.
   .
   .
   .
   COMPUTE MEAN-TAX = FUNCTION MEAN(TAX-S TAX-W TAX-B)
```

Conversión de elementos de datos (funciones intrínsecas)

Las funciones intrínsecas permiten convertir elementos de datos de serie de caracteres a los siguientes tipos:

- Mayúsculas o minúsculas
- Orden inverso
- Números

Además de utilizar las funciones intrínsecas para convertir caracteres, también puede utilizar la instrucción INSPECT.

Conversión a mayúsculas o minúsculas (UPPER-CASE, LOWER-CASE)

El siguiente código:

```
01 ITEM-1              PIC X(30)     VALUE "¡Hola a todos!".
01 ITEM-2              PIC X(30).
   .
   .
   .
   DISPLAY ITEM-1.
   DISPLAY FUNCTION UPPER-CASE(ITEM-1).
   DISPLAY FUNCTION LOWER-CASE(ITEM-1).
   MOVE FUNCTION UPPER-CASE(ITEM-1) TO ITEM-2.
   DISPLAY ITEM-2.
```

visualizaría los siguientes mensajes en el terminal:

```
¡Hola a Todos!
¡HOLA A TODOS!
¡hola a todos!
¡HOLA A TODOS!
```

Las instrucciones DISPLAY no modifican el contenido de ITEM-1 y sólo afectan la forma en que se visualizan las letras. Sin embargo, la instrucción MOVE hace que las letras en mayúsculas se trasladen al contenido de ITEM-2.

Conversión a orden inverso (REVERSE)

El código siguiente:

```
   MOVE FUNCTION REVERSE(ORIG-CUST-NAME) TO ORIG-CUST-NAME.
```

invertiría el orden de los caracteres de ORIG-CUST-NAME. Por ejemplo, si el valor inicial fuese GONZALEZ el valor después de realizar la instrucción sería ZELAZNOG.

Conversión a números (NUMVAL, NUMVAL-C)

Las funciones NUMVAL y NUMVAL-C convierten series de caracteres a números. Utilice estas funciones para convertir a formato numérico elementos de datos alfanuméricos que contienen números de representación de caracteres de formato libre y procesarlos de forma numérica. Por ejemplo:

```
01 R            PIC X(20)   VALUE "- 1234.5678".
01 S            PIC X(20)   VALUE "-$12.345,67CR".
01 TOTAL        USAGE IS COMP-2.
   .
   .
   .
   COMPUTE TOTAL = FUNCTION NUMVAL(R) + FUNCTION NUMVAL-C(S).
```

La diferencia entre NUMVAL y NUMVAL-C es que NUMVAL-C se utiliza cuando el argumento incluye un símbolo de moneda o un punto, como se muestra en el ejemplo. También puede colocar un signo algebraico al principio o al final, y éste se procesará. Los argumentos no deben sobrepasar los 18 dígitos (sin contar los símbolos de edición). En el manual *ILE COBOL/400 Reference* hallará las reglas sintácticas aplicables.

Nota: NUMVAL y NUMVAL-C devuelven un valor de coma flotante largo (de precisión doble). Por consiguiente, cualquier referencia a una de estas funciones representa una referencia a un elemento de datos numérico.

¿Por qué utilizar NUMVAL y NUMVAL-C?: Si utiliza NUMVAL o NUMVAL-C, no es necesario que declare datos numéricos en un formato fijo estáticamente, ni que declare los datos de entrada de una forma concreta. Por ejemplo, para este código:

```
01 X         PIC S999V99   LEADING SIGN IS SEPARATE.
.
.
.
ACCEPT X FROM CONSOLE.
```

El usuario de la aplicación debe entrar los número de la misma forma en que se hayan definido en la cláusula PICTURE. Por ejemplo:

```
+001,23
-300,00
```

Sin embargo, si utilizara la función NUMVAL, podría codificar lo siguiente:

```
01 A         PIC X(10).
01 B         PIC S999V99.
.
.
.
ACCEPT A FROM CONSOLE.
COMPUTE B = FUNCTION NUMVAL(A).
```

y la entrada podría ser:

```
1,23
-300
```

Evaluación de elementos de datos (funciones intrínsecas)

Es posible utilizar distintas funciones intrínsecas al evaluar elementos de datos:

- CHAR y ORD para evaluar enteros y caracteres numéricos simples con respecto al orden de clasificación que el programa utiliza.
- LENGTH para hallar la longitud de los elementos de datos.
- WHEN-COMPILED para hallar la fecha y hora en que se compiló el programa.

Evaluación de caracteres simples según el orden de clasificación (CHAR, ORD): Si desea conocer la posición ordinal de cierto carácter en el orden de clasificación, haga referencia a la función ORD utilizando el carácter en cuestión como argumento y ORD devolverá un entero que representará esta posición. Una forma conveniente de realizar esta operación es utilizar la subserie de un elemento de datos como el argumento de ORD:

```
IF FUNCTION ORD (CUSTOMER-RECORD(1:1)) IS > 194 THEN ...
```

Por otra parte, si conoce la posición del orden de clasificación que desea pero no sabe a qué carácter corresponde, haga referencia a la función CHAR utilizando la posición ordinal entera como argumento y CHAR devolverá el carácter deseado:

```
INITIALIZE CUSTOMER-NAME REPLACING ALPHABETIC BY FUNCTION CHAR(65).
```

Obtención de la longitud de elementos de datos (LENGTH)

La función LENGTH es útil en muchos contextos de programación para determinar la longitud de elementos de serie. La siguiente instrucción ILE COBOL/400 muestra cómo mover un elemento de datos, como por ejemplo el nombre de un cliente, al campo del registro específico para nombres de clientes:

```
MOVE CUSTOMER-NAME TO CUSTOMER-RECORD(1:FUNCTION LENGTH(CUSTOMER-NAME)).
```

Nota: La función LENGTH también puede utilizarse sobre un elemento de datos numérico o una entrada de tabla.

Registro especial LENGTH OF: Además de la función LENGTH, otra técnica para hallar la longitud de un elemento de datos es la utilización del registro especial LENGTH OF.

Existe una diferencia fundamental entre el registro especial LENGTH OF y la función intrínseca LENGTH. FUNCTION LENGTH devuelve la longitud de un elemento en posiciones de caracteres, mientras que LENGTH OF devuelve la longitud de un elemento en bytes. En la mayor parte de los casos, esto tiene poca importancia excepto cuando se trata de elementos de la clase DBCS. Por ejemplo:

```
77 CUSTOMER-NAME PIC X(30).
77 CUSTOMER-LOCATION-ASIA PIC G(50).
```

Tanto la codificación de `FUNCTION LENGTH(CUSTOMER-NAME)` como la de `LENGTH OF CUSTOMER-NAME` devolverá 30; sin embargo, la codificación de `FUNCTION LENGTH(CUSTOMER-LOCATION-ASIA)` devolverá 50, mientras que `LENGTH OF CUSTOMER-LOCATION-ASIA` devolverá 100.

Mientras que la función LENGTH sólo puede utilizarse en los casos en que se permiten expresiones aritméticas, el registro especial LENGTH OF puede utilizarse en una mayor variedad de contextos. Por ejemplo, el registro especial LENGTH OF puede utilizarse como argumento de una función intrínseca que permita argumentos enteros. (No es posible utilizar una función intrínseca como operando del registro especial LENGTH OF). El registro especial LENGTH OF también puede utilizarse como parámetro en una instrucción CALL.

Obtención de la fecha de compilación (WHEN-COMPILED)

Si desea conocer la fecha y hora en que se compiló el programa, tal como la proporcione el sistema en el que se compiló el programa, utilice la función WHEN-COMPILED. Se devolverá un resultado de 21 posiciones de caracteres de las que las 16 primeras tendrán el siguiente formato:

```
YYYYMMDDhhmmsshh
```

que indica los cuatro dígitos del año, el mes, el día y la hora (en horas, minutos, segundos y décimas de segundo) relativos a la compilación.

Registro especial WHEN-COMPILED: El registro especial WHEN-COMPILED constituye otra técnica que puede utilizarse para hallar la fecha y hora de la compilación. Presenta el siguiente formato:

```
MM/DD/YYhh.mm.ss
```

El registro especial WHEN-COMPILED sólo proporciona soporte a un año de dos dígitos y precisa la hora sólo hasta los segundos. El registro especial sólo puede utilizarse como el campo de envío de una instrucción MOVE.

Aritmética de coma fija frente a aritmética de coma flotante

Es posible que algunas de las instrucciones del programa impliquen operaciones aritméticas. Por ejemplo, cada una de las instrucciones de COBOL que aparecen a continuación requiere algún tipo de evaluación aritmética:

- Aritmética general.

```
COMPUTE REPORT-MATRIX-COL = (EMP-COUNT ** .5) + 1
ADD REPORT-MATRIX-MIN TO REPORT-MATRIX-MAX GIVING
    REPORT-MATRIX-TOT.
```

- Expresiones y funciones.

```
COMPUTE REPORT-MATRIX-COL = FUNCTION SQRT(EMP-COUNT) + 1
COMPUTE CURRENT-DAY = FUNCTION DAY-OF-INTEGER(NUMBER-OF-DAYS + 1)
```

- Comparaciones aritméticas.

```
IF REPORT-MATRIX-COL < FUNCTION SQRT(EMP-COUNT) + 1
IF CURRENT-DAY not = FUNCTION DAY-OF-INTEGER(NUMBER-OF-DAYS + 1)
```

  Para cada una de las evaluaciones aritméticas del programa -tanto si se trata de una instrucción, una función intrínseca, una expresión o bien una combinación de las anteriores anidadas entre sí- la forma en que se codifique la operación aritmética determinará si se realizará una evaluación de coma flotante o de coma fija.

La siguiente discusión explica cuándo una operación o una expresión aritmética se evalúa en forma de coma fija o en forma de coma flotante. Para obtener información detallada sobre la precisión de las evaluaciones aritméticas, consulte el apartado “Conversiones y precisión” en la página 158.

Evaluaciones de coma flotante

En general, si la evaluación aritmética tiene alguna de las características que se enumeran a continuación, el compilador la evaluará en aritmética de coma flotante:

- Un operando o un campo de resultado es de coma flotante.

  Un elemento de datos es de coma flotante si está codificado como literal de coma flotante o está definido como USAGE COMP-1, USAGE COMP-2 o coma flotante externo (USAGE DISPLAY con PICTURE de coma flotante).

  Un operando que sea una expresión aritmética anidada o una referencia a una función intrínseca numérica será de coma flotante si:

  – Un argumento de una expresión aritmética resulta en coma flotante.

  – La función es una función de coma flotante.

- Se trata del argumento de una función de coma flotante.

  Las funciones como COS y SIN son funciones de coma flotante que precisan un argumento. Como estas funciones son de coma flotante, el argumento se calculará como de coma flotante.

Evaluaciones de coma fija

En general, si la evaluación aritmética no contiene ninguna de las características que se han enumerado para comas flotantes, el compilador la evaluará en aritmética de coma fija. Es decir, el compilador manejará las evaluaciones aritméticas como de coma fija sólo si todos los operandos se proporcionan como de coma fija y el resultado se define como de coma fija. Las expresiones aritméticas anidadas y las referencias a funciones deben representar valores de coma fija.

Comparaciones aritméticas (condiciones de relación)

Si la operación aritmética es una comparación (contiene un operador relacional), las expresiones numéricas que se comparan -tanto si se trata de elementos de datos, de expresiones aritméticas, de referencias a funciones o de una combinación de las mismas- son realmente operandos (comparandos) en el contexto de toda la evaluación. Este también es el caso con comparaciones abreviadas; aunque un comparando no aparezca de forma explícita, ambos son operandos de la comparación. Si utiliza expresiones que contengan comparaciones en ILE COBOL/400, la expresión se evalúa como de coma flotante si por lo menos uno de los comparandos es de coma flotante o se resuelve como tal; en caso contrario, la expresión se calcula como de coma fija.

Por ejemplo, considere la siguiente instrucción:

```
IF (A + B) = C or D = (E + F)
```

En el ejemplo anterior hay dos comparaciones y por lo tanto cuatro comparandos. Si alguno de los cuatro comparandos es un valor de coma flotante o se resuelve como tal, todas las operaciones aritméticas de la instrucción IF se realizarán en forma de coma flotante; en caso contrario, todas las operaciones aritméticas se llevarán a cabo en forma de coma fija.

En el caso de la instrucción EVALUATE:

```
EVALUATE (A + D)
  WHEN (B + E) THRU C
  WHEN (F / G) THRU (H * I)
    .
    .
    .
END-EVALUATE.
```

Una instrucción EVALUATE puede volverse a escribir como una instrucción IF o una serie de instrucciones IF equivalentes. En este ejemplo, las instrucciones IF equivalentes son las siguientes:

```
if ( (A + D) >= (B + E) ) AND
   ( (A + D) <= C)

if ( (A + D) >= (F / G) ) AND
   ( (A + D) <= (H * I) )
```

Por lo tanto, siguiendo estas reglas que acabamos de ver para la instrucción IF debemos mirar todos los comparandos de cada una de las instrucciones IF para determinar si todas las operaciones aritméticas de la instrucción IF serán de coma fija o de coma variable.

Ejemplos de evaluaciones de coma fija y de coma flotante

Para los ejemplos que se muestran en el apartado “Aritmética de coma fija frente a aritmética de coma flotante” en la página 169, si define los elementos de datos de la siguiente forma:

```
01  EMPLOYEE-TABLE.
    05  EMP-COUNT           PIC  9(4).
    05  EMPLOYEE-RECORD OCCURS 1 TO 1000 TIMES
                         DEPENDING ON EMP-COUNT.
        10 HOURS             PIC +9(5)E+99.
.
.
.
01  REPORT-MATRIX-COL       PIC  9(3).
01  REPORT-MATRIX-MIN       PIC  9(3).
01  REPORT-MATRIX-MAX       PIC  9(3).
01  REPORT-MATRIX-TOT       PIC  9(3).
01  CURRENT-DAY             PIC  9(7).
01  NUMBER-OF-DAYS          PIC  9(3).
```

- Estas evaluaciones se realizarían en aritmética de coma flotante:

```
COMPUTE REPORT-MATRIX-COL = FUNCTION SQRT(EMP-COUNT) + 1
IF REPORT-MATRIX-TOT < FUNCTION SQRT(EMP-COUNT) + 1
```

- Estas evaluaciones se realizarían en aritmética de coma fija:

```
ADD REPORT-MATRIX-MIN TO REPORT-MATRIX-MAX GIVING REPORT-MATRIX-TOT.
IF CURRENT-DAY NOT = FUNCTION DAY-OF-INTEGER((NUMBER-OF-DAYS) + 1)
```

Proceso de elementos de tablas

Es posible procesar elementos de tabla alfanuméricos o numéricos utilizando funciones intrínsecas siempre que la descripción de datos del elemento de tabla sea compatible con los requisitos de argumentación de la función. En la publicación *ILE COBOL/400 Reference* se describen los formatos de datos necesarios para los argumentos de las distintas funciones intrínsecas.

Utilice un subíndice o un índice para hacer referencia a un elemento de datos individual como argumento de la función. Suponiendo que `TABLE-ONE` sea una matriz 3X3 de elementos numéricos, puede hallar la raíz cuadrada del elemento central con una instrucción como la siguiente:

```
COMPUTE X = FUNCTION SQRT(TABLE-ONE(2,2)).
```

¿Cuál es el problema con el año 2000?

El problema con el año 2000 estriba en la utilización de dos dígitos para representar el año. Si los campos de fecha del programa sólo tienen los 2 últimos dígitos del año, al llegar al 1/1/2000 el año acutal se representará como 00. Ello significa que el año actual será inferior al año anterior, ya que 00 es menor que 99.

En el release de ILE COBOL/400, se ha añadido el soporte al siglo para el siglo 21. Esto significa que si recupera un año cuyos 2 últimos dígitos estén comprendido en el rango 40 – 99, los dígitos “19” se añadirán como prefijo y así obtendrá un año de cuatro dígitos dentro del rango 1940 – 1999. Por el contrario, si recupera una año cuyos 2 últimos dígitos se encuentran en el rango 00 – 39, los

dígitos “20” se añadirán como prefijo y recuperará un año de cuatro dígitos dentro del rango 2000 – 2039.

Solución a largo plazo

Para que los programas puedan llegar hasta el año 9999, deberá realizar las siguientes operaciones:

1. Volver a escribir las aplicaciones utilizando uno de los métodos siguientes:
   - Utilizar las expresiones YYYYMMDD y YYYYDDD nuevas de la instrucción ACCEPT para conseguir un año de 4 dígitos *o*
   - Utilizar las funciones intrínsecas para obtener una fecha con un año de 4 dígitos (como por ejemplo CURRENT-DATE, DATE-OF-INTEGER y DAY-OF-INTEGER) *o*
   - Utilizar los servicios invocables del Entorno de Lenguajes Integrados para obtener fechas con un año de 4 dígitos.
2. Volver a crear las bases de datos con años de 4 dígitos.

Sin embargo, existe una solución a corto plazo que es más fácil.

Solución a corto plazo

Si no puede modificar todas las aplicaciones y datos antes del año 2000, utilice las funciones intrínsecas de la ventana de siglo, que permiten la interpretación de los años de 2 dígitos en una ventana de 100 años (ya que cada número de 2 dígitos sólo puede producirse una vez cada período de 100 años). Seleccione el período, proporcione a la función intrínseca un año de 2 dígitos o una fecha o día con un año de dos dígitos y la función intrínseca devolverá el valor adecuado con un año de 4 dígitos en esta ventana de 100 años.

El compilador ILE COBOL/400 proporciona tres funciones intrínsecas en la ventana de siglo: YEAR-TO-YYYY, DAY-TO-YYYYDDD y DATE-TO-YYYYMMDD. La función intrínseca YEAR-TO-YYYY toma un año de 2 dígitos y devuelve un año de 4 dígitos en una ventana de 100 años especificada. Las otras dos funciones intrínsecas toman una fecha que contiene un año de 2 dígitos y devuelven una fecha con un año de 4 dígitos en una ventana de 100 años especificada. Para la función intrínseca DAY-TO-YYYYDDD, la fecha que se toma es un número de 5 dígitos con un formato como el de la instrucción ACCEPT FROM DAY. De forma similar, la función intrínseca DATE-TO-YYYYMMDD toma un número de 6 dígitos con un formato como el de la instrucción ACCEPT FROM DATE.

Para obtener más información acerca de las funciones intrínsecas de la ventana de siglo, consulte el manual *ILE COBOL/400 Reference*, SC09-2073.

Ventajas de la solución a corto plazo

La ventaja de la solución a corto plazo es que sólo necesita modificar unos cuantos programas y no necesita cambiar las bases de datos. Este método es más barato, más rápido y más fácil que la solución a largo plazo.

Sin embargo, puede utilizar las funciones intrínsecas de la ventana de siglo para convertir las bases de datos o los archivos que tengan fechas con un año de 2 dígitos a fechas con un año de 4 dígitos. Para ello debe leer las fechas de los años de 2 dígitos, interpretarlas para obtener años de 4 dígitos y a continuaicón volver a grabar los datos en una copia del original que se haya ampliado para que

pueda contener datos con años de 4 dígitos. Todos los datos nuevos entrarían entonces en el archivo o base de datos nuevos.

Inconvenientes de la solución a corto plazo

Sólo podrá utilizar este método durante unos cuantos años, según de que aplicación se trate. Al final deberá cambiar todos los programas y bases de datos que contengan fechas.

No puede utilizar la ventana de siglo eternamente, ya que un año de 2 dígitos sólo puede ser único en un período de 100 años. Con el tiempo, necesitará más de 100 años para la ventana de datos. De hecho, muchas empresas ya necesitan ahora más de 100 años.

Una forma para que la ventana de siglo proporcione más tiempo consiste en que el usuario sepa distinguir en cualquier sección del código ILE COBOL/400 si una fecha es antigua (la fecha es del pasado) o si todavía no se ha alcanzado (la fecha es del futuro). Puede utilizar este conocimiento para determinar cómo establecer la ventana de siglo.

Sin embargo, hay limitaciones. Por ejemplo, la ventana de siglo no puede resolver el problema de intentar averiguar cuánto tiempo ha estado un cliente en el empresa si el tiempo supera los 100 años y sólo tiene años de 2 dígitos en las fechas. Otro ejemplo lo encontramos en la ordenación. Todos los registros que desee ordenar por fecha deberán tener fechas con años de 4 dígitos. Para solucionar estos y otros problemas, deberá utilizar instrucciones ACCEPT, funciones intrínsecas o servicios de datos ILE, que devuelven un años de 4 dígitos.

Capítulo 9. Llamada y compartimiento de datos entre programas ILE COBOL/400

A veces, una aplicación es lo suficientemente sencilla como para codificarla como un programa autosuficiente único. Sin embargo, en muchos casos, una solución de aplicación constará de varios programas compilados por separado que se utilizan conjuntamente.

El sistema AS/400 proporciona comunicación entre programas ILE COBOL/400, y entre programas ILE COBOL/400 y programas que no son de ILE COBOL/400.

Este capítulo describe:

- varios métodos utilizados para llamar a otro programa ILE COBOL/400
- cómo se devuelve de nuevo el control al programa de llamada una vez el programa llamado ha finalizado la ejecución
- cómo pasar datos entre el programa de llamada y el programa llamado.
- cómo cancelar un programa ILE COBOL/400.

Conceptos de ejecución

Se crea un objeto de programa desde uno o más objetos de módulo. Cada objeto de programa tiene asignado un sólo objeto de módulo como punto de entrada principal cuando se activa el objeto de programa. Cuando el compilador ILE COBOL/400 crea un objeto de módulo, se genera un PEP, el cual llama al programa ILE COBOL/400 más externo que se encuentre en la unidad de compilación. Al enlazar múltiples objetos de módulo juntos para crear un objeto de programa, debe especificar qué objeto de módulo contiene el PEP del objeto de programa que se está creando. Esto se lleva a cabo identificando el objeto de módulo en el parámetro ENTMOD del mandato CRTPGM. El PEP de este objeto de módulo se convierte en el PEP para el objeto de programa.

Cuando un objeto de programa se activa utilizando una llamada dinámica de programa, se otorga el control al PEP. Entonces el PEP llama al UEP, que es el programa ILE COBOL/400 más externo del objeto de módulo que se ha de ejecutar primero. Consulte la publicación *ILE Conceptos* para obtener información sobre los PEP y los UEP.

Activación y grupos de activación

El proceso de preparar un objeto de programa o un programa de servicio para ejecutarlo se denomina **activación**. El sistema realiza la activación cuando se llama a un objeto de programa. Como los programas de servicio no se llaman en su totalidad, estos se activan durante la llamada a un objeto de programa que, directa o indirectamente, necesite sus servicios. Los procedimientos ILE dentro de programas de servicio se llaman utilizando llamadas estáticas de procedimientos; no pueden llamarse mediante llamadas dinámicas de programas.

La activación realiza las funciones siguientes:

- Asigna de forma exclusiva los datos estáticos que el objeto de programa o el programa de servicio necesitan

- Cambia los enlaces simbólicos a programas de servicio utilizados, a enlaces a direcciones físicas.

Cuando la activación asigna el almacenamiento necesario para las variables estáticas utilizadas por un objeto de programa, el espacio se asigna desde un **grupo de activación**. Cada grupo de activación tiene un nombre. El nombre del grupo de activación lo proporciona el usuario (o el sistema cuando se especifica *NEW). En el momento de la creación del objeto de programa o del programa de servicio mediante CRTPGM o CRTSRVPGM, puede especificar el grupo de activación en que se activará el objeto de programa o el programa de servicio. Consulte el manual *ILE Conceptos* para obtener información más detallada sobre la activación y los grupos de activación.

Unidad de ejecución COBOL

Una **unidad de ejecución** COBOL es un conjunto de uno o más programas que funcionan como una unidad en la ejecución para proporcionar la solución de un problema.Una unidad de ejecución COBOL es una entidad independiente que puede ejecutarse sin comunicarse ni coordinarse con ninguna otra unidad de ejecución, a menos que pueda procesar archivos de datos y mensajes o establecer y comprobar conmutadores utilizados por otras unidades de ejecución. Una unidad de ejecución también puede contener objetos de programa y programas de servicios creados desde objetos de módulo que se hayan creado a partir de la compilación de programas escritos en lenguajes distintos a ILE COBOL/400.

En ILE, una unidad de ejecución COBOL está compuesta por objetos de programa y programas de servicio que se ejecutan todos en un único grupo de activación ILE. Para mantener la compatibilidad de la semántica de la unidad de ejecución OPM COBOL/400, la aplicación ILE COBOL/400 debe cumplir las condiciones siguientes:

- Cada unidad de compilación ILE COBOL/400 debe compilarse y luego enlazarse a un único objeto de programa.
- Todos los participantes de la unidad de ejecución (ILE COBOL/400 u otros programas/procedimientos ILE) deben ejecutarse en un único grupo de activación ILE.

  Nota: Debería utilizar un grupo de activación ILE con nombre en el cual poder ejecutar la aplicación para mantener adecuadamente la semántica de la unidad de ejecución COBOL. Utilizando un grupo de activación ILE con nombre para todos los objetos de programa participantes, no tiene que especificar un objeto de programa ILE COBOL/400 concreto para que sea el programa principal antes de ejecutar la aplicación. Por otro lado, si se sabe que un objeto de programa ILE COBOL/400 concreto es el programa principal antes de ejecutar la aplicación, puede especificar el atributo *NEW para la opción ACTGRP al crear un objeto *PGM, utilizando el programa ILE COBOL/400 como el UEP. Todos los demás objetos de programa participantes deben especificar el atributo *CALLER para la opción ACTGRP.
- La llamada más antigua del grupo de activación ILE (correspondiente a la unidad de ejecución) debe ser la de ILE COBOL/400. Este es el programa principal de la unidad de ejecución.

Si no se cumplen estas condiciones, puede haber un límite de control que enlace el ámbito de STOP RUN, de manera que no se renueve el estado de la aplicación entera.

Nota: La condición anterior indica que un programa ILE COBOL/400 que se ejecute en *DFTACTGRP, se ejecuta generalmente en una unidad de ejecución no compatible con una unidad de ejecución OPM COBOL/400.

Límites de control

Todos los lenguajes ILE, incluyendo ILE COBOL/400, utilizan un mecanismo común denominado **pila de llamadas** para transferir control a y desde procedimientos ILE u objetos de programas OPM llamados. La pila de llamadas consta de una lista de entradas del tipo última en entrar primera en salir con una entrada por cada procedimiento ILE u objeto de programa llamado. Cada entrada de la pila de llamadas tiene información sobre las variables automáticas para el procedimiento ILE y otros recursos del ámbito de la entrada de la pila de llamadas, como por ejemplo manejadores de condiciones y de cancelación.

En ILE COBOL/400, cada programa ILE COBOL/400 o programa anidado llamado tiene una entrada en la pila de llamadas. Cada procedimiento declarativo llamado también tiene su propia entrada en la pila de llamadas.

Una llamada añade una nueva entrada a la pila para el procedimiento ILE o para el objeto de programa OPM llamado y transfiere el control al objeto llamado. Una devolución de control elimina la entrada de la pila de llamadas y transfiere el control de nuevo al procedimiento ILE o al objeto de programa llamado en la entrada de la pila de llamadas anterior.

En ILE, puede crear una aplicación que ejecute objetos de programa en grupos de activación múltiples. Puede llamar a un objeto de programa ILE COBOL/400 que se esté ejecutando en un grupo de activación distinto al del programa de llamada. En este caso, la entrada de la pila de llamadas para el objeto de programa llamado se conoce como un **límite de control**. Un límite de control se define como cualquier entrada de la pila de llamadas ILE para la cual la entrada de la pila de llamadas inmediatamente anterior es para un procedimiento u objeto de programa ILE de un grupo de activación distinto. Una entrada de la pila de llamadas ILE la entrada de la pila de llamadas inmediatamente anterior sea para un objeto de programa OPM, también constituye un límite de control.

Si el objeto de programa llamado es el primer objeto de programa a activar en un grupo de activación concreto, entonces la entrada de la pila de llamadas se conoce como **límite de control fijo**. Si el objeto de programa llamado, que es un límite de control, no es el primer objeto de programa que se va a activar en un grupo de activación, entonces su entrada de la pila de llamadas se conoce como un **límite de control variable**. El programa principal de una unidad de ejecución con la que es compatible y la unidad de ejecución OPM COBOL/400 se encuentran en el límite de control fijo del grupo de activación.

Cuando una instrucción STOP RUN (o una instrucción GOBACK de un programa principal ILE COBOL/400) se encuentra en un programa ILE COBOL/400 llamado, el control se transfiere al llamador del límite de control. En una unidad de ejecución compatible con una unidad de ejecución OPM COBOL/400, STOP RUN finalizará la unidad de ejecución. Se realiza una operación COMMIT implícita en los archivos bajo el control de compromiso si éste está en el ámbito del grupo de acti-

vación y la activación finaliza normalmente sin errores cerrando los archivos. Se realiza una operación ROLLBACK si el grupo de activación finaliza anormalmente o si existen errores al cerrar los archivos.No ocurre nada si el control de compromiso está en el ámbito del trabajo.

El límite de control también se encuentra cuando un error no manejado se convierte en una comprobación de función. Si sigue sin manejarse la comprobación de función, ésta se pasa a la condición de error ILE genérica, CEE9901, en el límite de control y se envía al llamador del límite de control.

Programas principales y subprogramas

El primer programa del grupo de activación a activar inicia la unidad de ejecución COBOL y es el **programa principal**. El programa principal se encuentra en el límite de control fijo del grupo de activación. Ninguna opción o instrucción fuente específica identifica a un programa ILE COBOL/400 como programa principal o subprograma.

Un **subprograma** es un programa de la unidad de ejecución situado debajo del programa principal en la pila de llamadas. Para obtener más información sobre las pilas de programas y otros términos relacionados con la comunicación entre programas, vea el manual *CL Programación*.

Inicialización del almacenamiento

La primera vez que se llama a un programa ILE COBOL/400 de una unidad de ejecución, se inicializa su almacenamiento. El almacenamiento se inicializa de nuevo bajo las circunstancias siguientes:

- El párrafo PROGRAM-ID del programa ILE COBOL/400 posee la cláusula INITIAL. El almacenamiento se reinicializa cada vez que se llama al programa.
- Se finaliza la unidad de ejecución y luego se reinicializa.
- El programa se cancela (utilizando la instrucción CANCEL de ILE COBOL/400) y luego se llama de nuevo.
- El final de las direcciones de bifurcación de nombre de sección y nombre de párrafo se reinicializa siempre (como consecuencia de instrucciones PERFORM anteriores).

Transferencia de control a otro programa

En la PROCEDURE DIVISION, un programa puede llamar a otro programa (generalmente denominado subprograma en términos COBOL), y este programa llamado puede a su vez llamar a otro programa. El programa que llama a otro programa se denomina el programa **de llamada** y el programa al que llama se denomina el programa **llamado**.

El programa ILE COBOL/400 llamado empieza a ejecutarse al principio de la parte no declarativa de la PROCEDURE DIVISION. Si un programa ILE COBOL/400 llamado no tiene una Procedure Division o no tiene una parte no declarativa en la PROCEDURE DIVISION, simplemente volverá al programa de llamada ILE COBOL/400.

Una vez finalizado el proceso del programa llamado, el programa puede transferir de nuevo el control al programa de llamada o finalizar la unidad de ejecución. La

unidad de ejecución finaliza después de emitir STOP RUN siempre que el límite de control más cercano sea un límite de control fijo. Si el límite de control más cercano es un límite de control variable, entonces el control vuelve al llamador del límite de control, pero la unidad de ejecución permanece activa.

Un programa llamado no debe llamar directa ni indirectamente a su llamador (como por ejemplo, el programa X llama al programa Y; el programa Y llama al programa Z y el programa Z entonces llama al programa X). Esto se denomina una llamada **recursiva**. ILE COBOL/400 **no** permite la recursión en programas principales ni subprogramas.Si se intenta una repetición, se generará un mensaje de error de ejecución.

Llamada a un programa ILE COBOL/400

Para llamar a otro programa ILE COBOL/400, puede utilizar uno de los métodos siguientes:

- Llamadas a programas anidados
- Llamadas estáticas de procedimientos
- Llamadas dinámicas de programas

Las **llamadas a programas anidados** le permiten crear aplicaciones utilizando técnicas de programación estructuradas. También pueden utilizarse en lugar de procedimientos PERFORM para evitar la modificación accidental de elementos de datos.Las llamadas a programas anidados pueden realizarse utilizando la instrucción de CALL *literal* o la instrucción de CALL *identificador*. Para obtener más información sobre programas anidados, vea el apartado “Llamada a programas anidados” en la página 182.

Una **llamada estática de procedimientos** transfiere el control a un programa ILE COBOL/400 llamado que esté enlazado por copia o por referencia al mismo objeto de programa que el programa ILE COBOL/400 de llamada. Las llamadas estáticas de procedimientos pueden realizarse utilizando la instrucción CALL *literal* o la instrucción CALL *puntero de procedimiento*. Una llamada estática de procedimientos puede utilizarse para llamar a cualquiera de los procedimientos siguientes:

- Un procedimiento ILE dentro del mismo objeto de módulo
- Un programa ILE COBOL/400 anidado (utilizando CALL *literal*)
- Un procedimiento ILE en un objeto de módulo separado que esté enlazado al programa de llamada ILE COBOL/400
- Un procedimiento ILE en un programa de servicio separado.

Una **llamada dinámica de programa** transfiere el control a un programa ILE COBOL/400 llamado que esté enlazado a un objeto de programa separado del programa de llamada ILE COBOL/400. El programa ILE COBOL/400 llamado debe ser el UEP del objeto de programa. Sólo el programa ILE COBOL/400, que es el UEP del objeto de programa, puede llamarse desde otro programa ILE COBOL/400 que se encuentre en un objeto de programa distinto.Los programas ILE COBOL/400 distintos al designado como UEP, son sólo visibles dentro del objeto de programa. Con una llamada dinámica de programa, el objeto de programa llamado se activa la primera vez que se llama dentro del grupo de activación. Las llamadas dinámicas de programa pueden realizarse utilizando las instrucciones CALL *literal*, CALL *identificador* o CALL *elemento-de-datos-de-puntero-de-procedimiento*. Utilice la instrucción SET *elemento-de-datos-de-puntero-de-procedimiento* TO ENTRY *nombre-de-objeto-de-programa* para establecer el

elemento-de-datos-de-puntero-de-procedimiento antes de utilizar la instrucción CALL *elemento-de-datos-de puntero-de-procedimiento*.

Para obtener información adicional sobre llamadas estáticas de procedimiento y las llamadas dinámicas de programa, consulte el apartado “Utilización de llamadas estáticas de procedimiento y llamadas dinámicas de programa” en la página 186.

Identificación del tipo de enlace de programas y procedimientos llamados

Al llamar a otro programa ILE COBOL/400 que no esté en el mismo objeto de módulo que el programa de llamada, si la llamada se realiza mediante una instrucción CALL *literal*, debe especificar si el programa llamado es un objeto de programa ILE o un procedimiento ILE.

Identifique si está llamando a un objeto de programa o a un procedimiento especificando el tipo de enlace de la llamada.

El tipo de LINKAGE de la llamada puede especificarse de forma explícita u obligada, especificando una expresión que esté asociada con un enlace concreto. Por ejemplo, la expresión IN LIBRARY obliga a que una llamada sea un LINKAGE programa. En los casos en los que ninguna expresión obligue al enlace, hay tres formas de especificar un enlace de forma explícita. Se listan en orden de prioridad:

1. La cláusula LINKAGE de las instrucciones CALL, CANCEL o SET...ENTRY

   Para llamar o cancelar un objeto de programa, especifique LINKAGE TYPE IS PROGRAM en la instrucción CALL, CANCEL o SET...ENTRY.

   ```
   PROCEDURE DIVISION.
         ⋮
        CALL LINKAGE TYPE IS PROGRAM literal-1
         ⋮
        CALL LINKAGE PROGRAM literal-2 IN LIBRARY literal-3
         ⋮
        CANCEL LINKAGE PROGRAM literal-2 IN LIBRARY literal-3
         ⋮
        CANCEL LINKAGE TYPE IS PROGRAM literal-1
   ```

   Para llamar o cancelar un procedimiento, especifique LINKAGE TYPE IS PROCEDURE en la instrucción CALL, CANCEL o SET...ENTRY. La expresión IN LIBRARY no puede especificarse para una instrucción CALL, CANCEL o SET que incluya una expresión LINKAGE TYPE IS PROCEDURE. La expresión IN LIBRARY se utiliza para especificar un nombre de biblioteca de OS/400 para una objeto de programa (*PGM).

   ```
   PROCEDURE DIVISION.
         ⋮
        CALL LINKAGE TYPE IS PROCEDURE literal-1
         ⋮
        CANCEL LINKAGE TYPE IS PROCEDURE literal-1
   ```

2. La expresión LINKAGE TYPE del párrafo SPECIAL-NAMES

   Para llamar o cancelar un objeto de programa, especifique LINKAGE TYPE IS PROGRAM FOR literal-1 en el párrafo SPECIAL-NAMES donde literal-1 es el nombre del objeto de programa al que se está llamando. No tiene

que especificar la palabra clave LINKAGE TYPE con la instrucción CALL, CANCEL o SET...ENTRY cuando el enlace se haya definido en el párrafo SPECIAL-NAMES.

```
       ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.
                ⋮
         SPECIAL-NAMES.
              LINKAGE TYPE IS PROGRAM FOR literal-1.
                ⋮
 PROCEDURE DIVISION.
                ⋮
              CALL literal-1.
                ⋮
              CANCEL literal-1.
```

Para llamar o cancelar un procedimiento, especifique LINKAGE TYPE IS PROCEDURE FOR literal-1 en el párrafo SPECIAL-NAMES, donde literal-1 es el nombre del procedimiento al que está llamando. No tiene que especificar la expresión LINKAGE TYPE con la instrucción CALL, CANCEL o SET...ENTRY cuando el enlace se ha definido en el párrafo SPECIAL-NAMES.

```
       ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.
                ⋮
         SPECIAL-NAMES.
              LINKAGE TYPE IS PROCEDURE FOR literal-1.
                ⋮
 PROCEDURE DIVISION.
                ⋮
              CALL literal-1.
                ⋮
              CANCEL literal-1.
```

3. El parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL o la opción de instrucción PROCESS asociada.

   El parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL le permite especificar, en el momento de la compilación, el tipo de enlace para todas las instrucciones externas CALL literal-1, CANCEL literal-1 o SET elemento-de-datos-de-puntero-de-procedimiento TO ENTRY literal-1 del programa ILE COBOL/400. No tiene que especificar la cláusula LINKAGE TYPE en el párrafo SPECIAL-NAMES ni la expresión LINKAGE TYPE con la instrucción CALL, CANCEL o SET...ENTRY cuando el enlace haya sido definido mediante el parámetro LINKLIT de CRTCBLMOD o CRTBNDCBL.

   Para crear un módulo que llame a objetos de programa, teclee:

   ```
   CRTCBLMOD MODULE(MYLIB/XMPLE1)
   SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
   LINKLIT(*PGM)
   ```

   Para crear un módulo que llame a procedimientos, teclee:

   ```
   CRTCBLMOD MODULE(MYLIB/XMPLE1)
   SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
   LINKLIT(*PRC)
   ```

Codifique las instrucciones CALL y CANCEL de la siguiente forma al utilizar el parámetro LINKLIT de CRTCBLMOD para especificar el tipo de enlace:

```
PROCEDURE DIVISION.
              ⋮
             CALL literal-1.
              ⋮
             CANCEL literal-1.
```

Llamada a programas anidados

Los programas anidados le proporcionan un método para crear funciones modulares para la aplicación y para mantener técnicas de programación estructuradas. Los programas anidados le permiten definir múltiples funciones separadas, cada una con su propio ámbito de control, dentro de una única unidad de compilación. Pueden utilizarse como procedimientos PERFORM con la capacidad adicional de proteger elementos de datos **locales**.

Cuando se compilan, los programas anidados residen en el mismo módulo que su programa de llamada.Por consiguiente, los programas anidados siempre se ejecutan en el mismo grupo de activación que sus programas de llamada.

Estructura de los programas anidados

Un programa ILE COBOL/400 puede **contener** otros programas ILE COBOL/400. Los programas **contenidos** pueden a su vez contener otros programas.Un programa contenido puede, **directa** o **indirectamente**, estar contenido dentro de otro programa.

La Figura 49 en la página 183 describe la estructura de un programa anidado con programas contenidos directa e indirectamente.

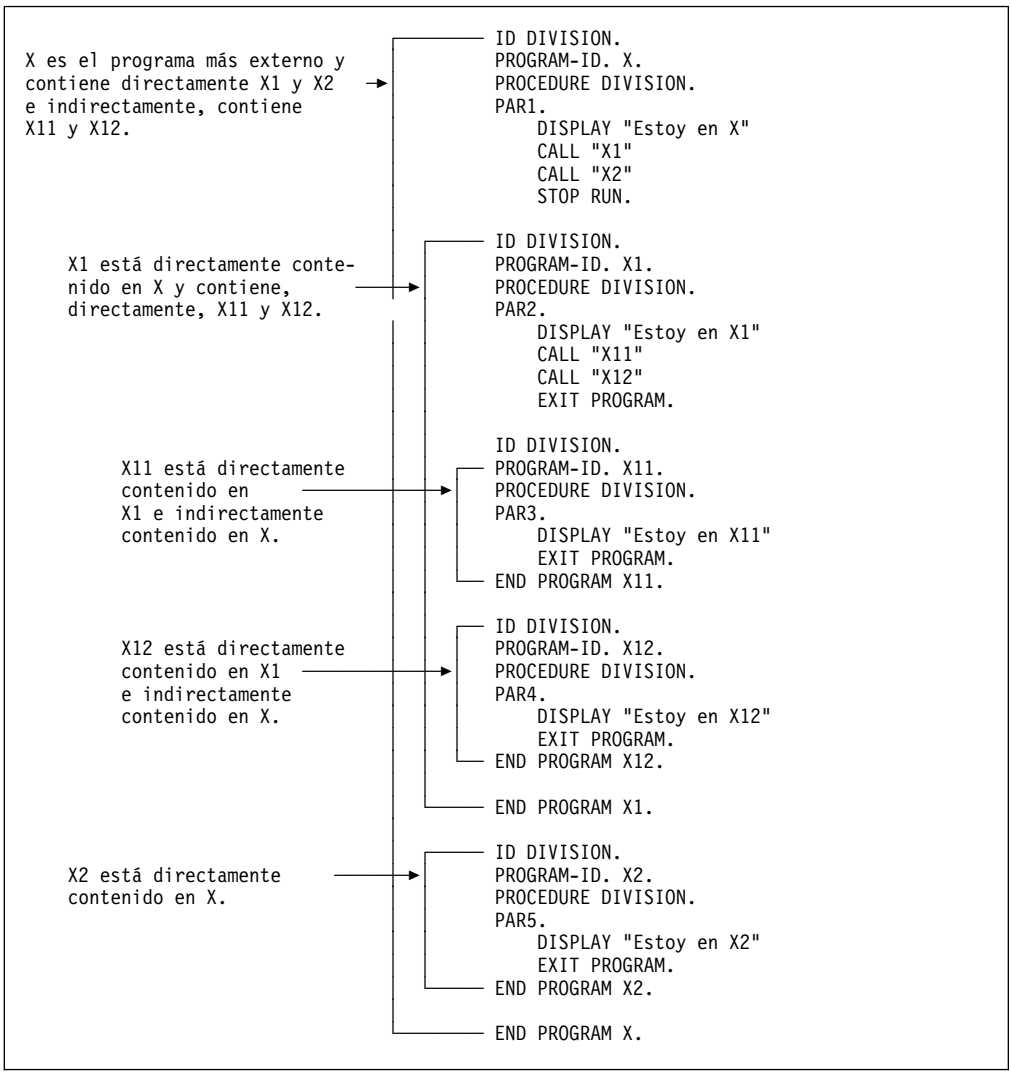


Figura 49. Estructura de programas anidados con programas contenidos directa e indirectamente

Convenciones para utilizar la estructura de programas anidados

Existen varias convenciones aplicables al utilizar estructuras de programas anidados.

1. En cada programa se requiere la IDENTIFICATION DIVISION. Todas las otras divisiones son opcionales.
2. El nombre de programa del párrafo PROGRAM-ID debe ser exclusivo.
3. Los nombres de los programas anidados pueden ser cualquier palabra COBOL válida o un literal no numérico.
4. Los programas anidados no pueden tener una CONFIGURATION SECTION. El programa más externo debe especificar cualquier opción en la Configuration Section que se necesite.
5. Cada programa anidado se incluye en el programa que lo contiene inmediatamente antes de su cabecera END PROGRAM (vea la Figura 49).
6. Cada programa ILE COBOL/400 debe terminar con una cabecera END PROGRAM.

7. Los programas anidados sólo pueden llamarse o cancelarse desde un programa ILE COBOL/400 en el mismo objeto de módulo.
8. Las llamadas a programas anidados sólo pueden realizarse utilizando una instrucción CALL *literal* o CALL *identificador*. Las llamadas a programas anidados no pueden realizarse utilizando CALL *puntero-de-procedimiento*. Las llamadas a programas anidados siguen las mismas normas que las llamadas estáticas a procedimiento.

Jerarquía de llamadas a programas anidados

A un programa anidado sólo lo puede llamar el programa que lo contiene directamente, a menos que el programa anidado esté identificado como COMMON en el párrafo PROGRAM-ID. En ese caso, al programa COMMON también puede llamarle cualquier programa contenido (directa o indirectamente) dentro del mismo programa que el que contiene directamente al programa COMMON. No se permiten las llamadas recursivas.

La Figura 50 muestra el perfil de una estructura anidada con algunos programas contenidos identificados como COMMON.

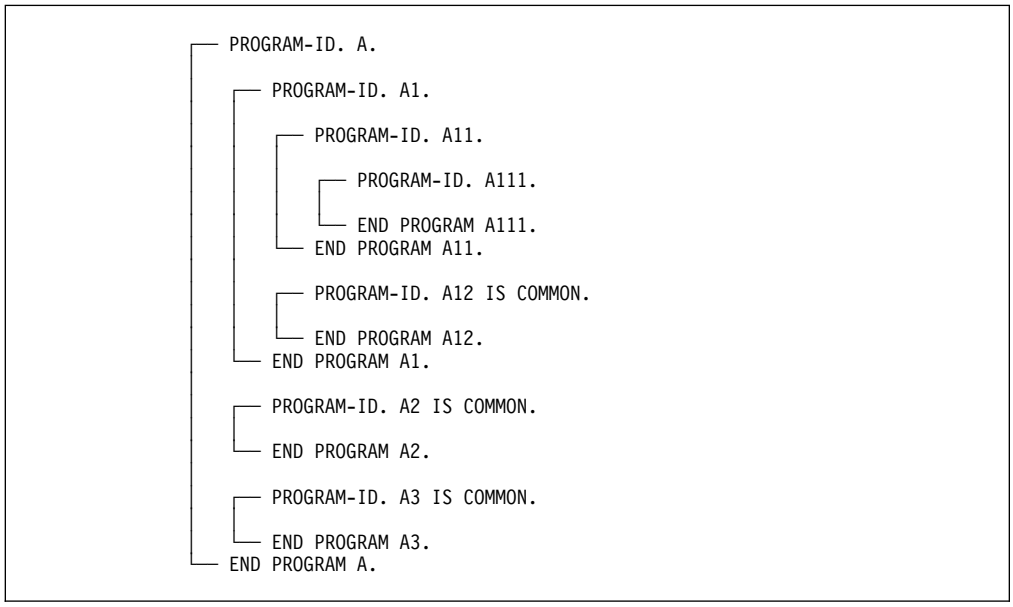


Figura 50. Estructura de programas anidados con programas contenidos directa e indirectamente

La tabla siguiente describe la **jerarquía de llamadas** para la estructura mostrada en la Figura 50. Observe que A12, A2 y A3 se identifican como COMMON y las diferencias resultantes en las llamadas asociadas con los mismos.

Tabla 10. Jerarquía de llamadas para estructuras anidadas con programas COMMON

Este programa	Puede llamar a estos programas	Y puede ser llamado por estos programas
A	A1, A2, A3	Ninguno
A1	A11, A12, A2, A3	A
A11	A111, A12, A2, A3	A1
A111	A12, A2, A3	A11
A12	A2, A3	A1, A11, A111
A2	A3	A, A1, A11, A111, A12, A3
A3	A2	A, A1, A11, A111, A12, A2

Observe que:

- A2 no puede llamar a A1 porque A1 no es COMMON y no está directamente contenido en A2.
- A111 no puede llamar a A11 porque eso sería una llamada recursiva.
- A1 puede llamar a A2 porque A2 es COMMON.
- A1 puede llamar a A3 porque A3 es COMMON.

Ámbito de nombres dentro de una estructura anidada

Existen dos clases de nombres dentro de las estructuras anidadas— **locales** y **globales**. La clase determinará si se conoce un nombre más allá del ámbito del programa que lo declara.

Nombres locales: Los nombres son locales a menos que se indique que estén definidos como GLOBAL (con excepción del nombre del programa). Estos nombres locales no son visibles ni accesibles para ningún programa fuera del programa en que se declaran; esto incluye los programas contenidos y los que los contienen.

Nombres globales: Un nombre especificado como global (utilizando la cláusula GLOBAL) es visible y accesible para el programa en que se declara y para todos los programas contenidos, directa e indirectamente, dentro del programa.Esto permite a los programas contenidos compartir datos y archivos comunes con el programa que los contiene, simplemente haciendo referencia al nombre del elemento.

Cualquier elemento que sea subordinado al elemento global (incluyendo los nombres e índices de condición) es global automáticamente.

Puede declararse el mismo nombre con la cláusula GLOBAL múltiples veces, siempre que cada declaración ocurra en un programa distinto. Recuerde que enmascarar u ocultar un nombre dentro de una estructura anidada es posible, utilizando el mismo nombre en programas distintos de la misma estructura que los contienen.

Búsqueda de declaraciones de nombres: Cuando en un programa se hace referencia a un nombre, se realiza una búsqueda para localizar la declaración para dicho nombre.La búsqueda empieza dentro del programa que contiene la referencia y continúa **hacia fuera** a programas continentes hasta que se encuentra una coincidencia.La búsqueda sigue este proceso:

1. Primero se buscan las declaraciones dentro del programa.
2. Si no se encuentra ninguna coincidencia, entonces se buscarán **sólo** las declaraciones globales en sucesivos programas externos continentes.
3. La búsqueda finaliza cuando se encuentra el primer nombre coincidente; de lo contrario, si no se encuentra ninguna coincidencia, aparecerá un error.

Utilización de llamadas estáticas de procedimiento y llamadas dinámicas de programa

La información siguiente sólo es aplicable a subprogramas compilados por separado, no a programas anidados. Para obtener más información sobre las llamadas dentro de una estructura de programas anidados, consulte el apartado “Llamada a programas anidados” en la página 182.

El proceso de enlace difiere según si el programa ILE COBOL/400 utiliza llamadas estáticas de procedimiento o llamadas dinámicas de programa. Cuando se utiliza una llamada estática de procedimiento para llamar a un subprograma ILE COBOL/400, primero debe compilarse en un objeto de módulo y luego enlazarse, por copia o por referencia, al mismo objeto de programa que el programa de llamada ILE COBOL/400. Cuando se utiliza una llamada dinámica de programa para llamar a un subprograma ILE COBOL/400, éste debe compilarse y enlazarse como un objeto de programa separado. Para obtener más información sobre el proceso de enlace, vea el manual *ILE Conceptos*.

Las llamadas estáticas de procedimiento ofrecen ventajas en rendimiento sobre las llamadas dinámicas de programa.

Cuando se llama a un subprograma ILE COBOL/400 utilizando una llamada estática de procedimiento, éste ya está activado, ya que está enlazado en el mismo objeto de programa que el programa de llamada y se ejecuta inmediatamente al recibir el control desde el programa de llamada ILE COBOL/400.

Cuando se llama a un subprograma ILE COBOL/400 utilizando una llamada dinámica de programa, se necesitan muchas más tareas antes de que se ejecute el programa llamado ILE COBOL/400. Estas tareas incluyen lo siguiente:

- Si el grupo de activación en el cual se debe activar el programa llamado ILE COBOL/400 no existe, primero debe crearse antes de que el programa pueda activarse.
- Si el programa llamado ILE COBOL/400 no se ha activado previamente, primero debe activarse antes de poderse ejecutar.Activar el programa llamado ILE COBOL/400 también implica activar todos los programas de servicio enlazados (directa o indirectamente) a él. La activación implica realizar las funciones siguientes:
  - Asignar de forma exclusiva los datos estáticos que el objeto de programa o el programa de servicio necesitan
  - cambiar los enlaces simbólicos a programas de servicio utilizados a enlaces a direcciones físicas.

Así, una llamada dinámica de programa es más lenta que una estática de procedimientos debido al coste de activación de la primera vez que se realiza en un grupo de activación.

Las llamadas dinámicas de programa y las llamadas estáticas de procedimiento también difieren en el número de operandos que pueden pasarse del programa de llamada ILE COBOL/400 al programa llamado ILE COBOL/400. Puede pasar hasta 255 operandos utilizando una llamada dinámica de programa. Con una llamada estática de procedimiento puede pasar hasta 400 operandos.

Los argumentos diseñados como OMITTED o que tienen descriptores operativos asociados sólo pueden pasarse utilizando una llamada estática de procedimiento. Estos argumentos no pueden pasarse utilizando llamadas dinámicas de programa.

Realización de llamadas estáticas de procedimiento utilizando CALL literal

Puede realizar una llamada estática de procedimiento utilizando la instrucción de CALL *literal* (donde *literal* es el nombre de un subprograma). Existen tres formas de especificar que la llamada será una llamada estática de procedimiento. Se listan en orden de prioridad:

Nota: La expresión IN LIBRARY es incompatible con una llamada estática de procedimiento.

1. Utilice la expresión LINKAGE de la instrucción CALL.

   Especifique LINKAGE TYPE IS PROCEDURE en la instrucción CALL para asegurarse de que el programa llamado se llamará utilizando una llamada estática de procedimiento.

   ```
   PROCEDURE DIVISION.
                 ⋮
                CALL LINKAGE TYPE IS PROCEDURE literal-1
   ```

2. Utilice la cláusula LINKAGE TYPE del párrafo SPECIAL-NAMES.

   Especifique LINKAGE TYPE IS PROCEDURE FOR *literal-1* en el párrafo SPECIAL-NAMES, donde *literal-1* es el nombre del programa ILE COBOL/400 al que está llamando. No tiene que especificar la expresión LINKAGE TYPE con la instrucción CALL cuando el enlace se haya especificado en el párrafo SPECIAL-NAMES.

   ```
   ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
           ⋮
     SPECIAL-NAMES.
          LINKAGE TYPE IS PROCEDURE FOR literal-1.
           ⋮
     PROCEDURE DIVISION.
           ⋮
          CALL literal-1.
   ```

3. Utilice el parámetro LINKLIT de los mandatos CRTCBLMOD CRTBNDCBL o la opción de la instrucción PROCESS asociada.

   Especifique *PRC con el parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL, en el momento de la compilación, para indicar que se realizarán llamadas estáticas de procedimiento para todas las instrucciones CALL *literal-1* externas del programa ILE COBOL/400. No tiene que especificar la cláusula LINKAGE TYPE en el párrafo SPECIAL-NAMES ni la expresión LINKAGE TYPE con la instrucción CALL o CANCEL cuando el enlace haya sido definido por el parámetro LINKLIT de CRTCBLMOD.

```
CRTCBLMOD MODULE(MYLIB/XMPLE1)
SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
LINKLIT(*PRC)
```

Codifique las instrucciones CALL de la forma siguiente si ha utilizado el parámetro LINKLIT de CRTCBLMOD para especificar el tipo de enlace:

```
PROCEDURE DIVISION.
      ⋮
     CALL literal-1.
```

Realización de llamadas dinámicas de programa utilizando CALL literal

Puede realizar una llamada dinámica de programa utilizando la instrucción CALL *literal* (donde *literal* es el nombre de un subprograma) o la instrucción CALL *identificador*. Consulte el apartado “Utilización de CALL identificador” en la página 189 para obtener más información sobre CALL *identificador*. Utilizando CALL *literal* existen tres formas de especificar que la llamada será una llamada dinámica de programa. Se listan en orden de prioridad:

1. Utilice la expresión LINKAGE de la instrucción CALL.

   Especifique LINKAGE TYPE IS PROGRAM en la instrucción CALL para asegurarse de que el programa llamado se llamará utilizando una llamada dinámica de programa.

   ```
   PROCEDURE DIVISION.
         ⋮
        CALL LINKAGE TYPE IS PROGRAM literal-1
   ```

2. Utilice la cláusula LINKAGE TYPE del párrafo SPECIAL-NAMES.

   Especifique LINKAGE TYPE IS PROGRAM FOR *literal-1* en el párrafo SPECIAL-NAMES, donde *literal-1* es el nombre del programa ILE COBOL/400 al que está llamando. No tiene que especificar la expresión LINKAGE TYPE con la instrucción CALL cuando el enlace se haya especificado en el párrafo SPECIAL-NAMES.

   ```
   ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
           ⋮
     SPECIAL-NAMES.
          LINKAGE TYPE IS PROGRAM FOR literal-1.
           ⋮
   PROCEDURE DIVISION.
           ⋮
          CALL literal-1.
   ```

3. Utilice el parámetro LINKLIT de los mandatos CRTCBLMOD CRTBNDCBL o la opción de la instrucción PROCESS asociada.

   Especifique *PGM con el parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL, en el momento de la compilación, para indicar que se realizarán llamadas dinámicas de programa para todas las instrucciones CALL *literal-1* externas del programa ILE COBOL/400. No tiene que especificar la cláusula LINKAGE TYPE en el párrafo SPECIAL-NAMES ni la expresión LINKAGE TYPE con la instrucción CALL o CANCEL cuando el enlace haya sido definido por el parámetro LINKLIT de CRTCBLMOD.

```
CRTCBLMOD MODULE(MYLIB/XMPLE1)
SRCFILE(MYLIB/QCBLLESRC)  SRCMBR(XMPLE1)
LINKLIT(*PGM)
```

Codifique las instrucciones CALL de la forma siguiente si ha utilizado el parámetro LINKLIT de CRTCBLMOD para especificar el tipo de enlace:

```
PROCEDURE DIVISION.
                ⋮
              CALL literal-1.
```

Una llamada dinámica de programa activa el subprograma en la ejecución. Utilice una instrucción de llamada dinámica cuando:

- Desee simplificar las tareas de mantenimiento y aprovechar la reutilización de código.

  Cuando se cambia un subprograma, deben volverse a enlazar todos los objetos de módulo, excepto los programas de servicio que lo llamen de forma estática y estén enlazados por copia. Si están enlazados por referencia, tampoco tienen que volver a enlazarse siempre que no se cambie la interfaz entre el subprograma y los objetos de módulo. Si el subprograma cambiado se llama dinámicamente, sólo deberá volver a enlazarse dicho subprograma. Así, las llamadas dinámicas hacen más fácil el mantener una copia de un subprograma con un número mínimo de enlaces.

- Los subprogramas llamados con CALL *literal* se utilizan con poca frecuencia o son muy grandes.

  Si los subprogramas se llaman sólo en unas pocas condiciones, las llamadas dinámicas pueden activar los subprogramas sólo cuando sea necesario.

  Si los subprogramas son muy grandes o hay muchos subprogramas, la utilización de llamadas estáticas puede requerir un mayor tamaño del conjunto de trabajo en el almacenamiento principal.

Utilización de CALL identificador

Puede utilizar CALL *identificador* (donde *identificador* no es un puntero de procedimiento) para llamar a un programa ILE COBOL/400 anidado o a un objeto de programa. El contenido del identificador determina, en el momento de la ejecución, si se llama a un programa anidado o a un objeto de programa. Si el contenido del identificador coincide con el nombre de un programa anidado visible, la llamada se dirige al programa anidado. De lo contrario, se realiza una llamada dinámica de programa a un objeto de programa que tenga el nombre especificado en el contenido del identificador.

| Si se especifica la expresión IN LIBRARY en una instrucción CALL identificador la
| llamada se pasa obligatoriamente a un objeto de programa.

La primera vez que utilice el identificador en una instrucción CALL, se establecerá
| un puntero abierto que asociará un identificador CALL (y cualquier elemento aso-
| ciado IN LIBRARY) con un objeto.

Si lleva a cabo una llamada por identificador a un objeto de programa que posteriormente suprime o renombra, debe utilizar la instrucción CANCEL para anular el puntero abierto asociado con el identificador. Esto asegura que la próxima vez que utilice el identificador para llamar al objeto de programa, el puntero abierto asociado se establecerá de nuevo.

El ejemplo siguiente muestra cómo aplicar la instrucción CANCEL a un identificador:

```
MOVE "ABCD" TO IDENT-1.
CALL IDENT-1.
CANCEL IDENT-1.
```

Si aplica la instrucción CANCEL directamente al literal "ABCD", *no* anulará el puntero abierto asociado con IDENT-1. En su lugar, puede continuar llamando al programa ABCD simplemente utilizando IDENT-1 en la instrucción CALL.

El valor del puntero abierto también se modifica si se cambia el valor del identificador CALL y se realiza una llamada utilizando este nuevo valor. El valor del puntero abierto también se ve afectado por los elementos asociados IN LIBRARY. Si se especifica una biblioteca distinta para una instrucción CALL to IDENT-1 distinta a la llamada anterior a IDENT-1, se restablece el puntero abierto.

Utilización de CALL puntero de procedimiento

Puede realizar una llamada estática de procedimiento o una llamada dinámica de programa utilizando la instrucción CALL *puntero de procedimiento*.

Antes de utilizar la instrucción CALL *puntero de procedimiento*, debe establecer el elemento de datos *puntero de procedimiento* en un valor de dirección. El elemento de datos *puntero de procedimiento* puede establecerse al programa COBOL más externo (un procedimiento ILE), un procedimiento ILE en otra unidad de compilación o un objeto de programa. Utilice el formato 6 de la instrucción SET para establecer el valor del elemento de datos *puntero de procedimiento*.

Especifique LINKAGE TYPE IS PROCEDURE en la instrucción SET para establecer el elemento de datos *puntero de procedimiento* a procedimiento ILE.

Especifique LINKAGE TYPE IS PROGRAM en la instrucción SET para establecer el elemento de datos *puntero de procedimiento* a objeto de programa.

También puede utilizar la cláusula LINKAGE TYPE del párrafo SPECIAL-NAMES o el parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL para determinar el tipo de objeto al que se establece el elemento de datos *puntero de procedimiento*. Consulte el apartado “Identificación del tipo de enlace de programas y procedimientos llamados” en la página 180 para obtener más información sobre el establecimiento del tipo de enlace utilizando la claúsula LINKAGE TYPE del párrafo SPECIAL-NAMES o el parámetro LINKLIT de los mandatos CRTCBLMOD y CRTBNDCBL.

Codifique las instrucciones SET y CALL de la forma siguiente al utilizar CALL *puntero de procedimiento* para realizar una llamada estática de procedimiento:

```
PROCEDURE DIVISION.
             ⋮
            SET puntero-procedimiento
                TO ENTRY LINKAGE TYPE IS PROCEDURE literal-1.
             ⋮
            CALL puntero-procedimiento.
```

Codifique las instrucciones SET y CALL de la forma siguiente al utilizar CALL *puntero de procedimiento* para realizar una llamada dinámica de programa:

```
PROCEDURE DIVISION.
                ⋮
              SET puntero-procedimiento
                  TO ENTRY LINKAGE TYPE IS PROGRAM literal-1.
                ⋮
              CALL puntero-procedimiento.
```

Devolución del control desde un programa ILE COBOL/400

Puede emitir una instrucción STOP RUN, EXIT PROGRAM o GOBACK para devolver el control desde un programa ILE COBOL/400 llamado.

Debe saber si un programa ILE COBOL/400 es un programa principal o un subprograma para determinar cómo se devuelve el control desde un programa llamado cuando ocurre un error o finaliza un programa. Vea “Programas principales y subprogramas” en la página 178 para obtener una descripción de los programas principales y los subprogramas.

Devolución del control desde un programa principal

Para devolver el control desde un programa principal, utilice STOP RUN, GOBACK o EXIT PROGRAM junto con la expresión CONTINUE. Las instrucciones STOP RUN y GOBACK finalizan la unidad de ejecución y se devuelve el control al llamador del programa principal. No puede utilizarse EXIT PROGRAM sin la expresión CONTINUE para devolver el control desde un programa principal. Cuando se encuentra EXIT PROGRAM sin la expresión CONTINUE en un programa principal, no se realiza ninguna operación y el proceso continúa en la siguiente instrucción del programa principal.

Devolución del control desde un grupo de activación *NEW

Si realiza STOP RUN, GOBACK o EXIT PROGRAM con la expresión CONTINUE desde un programa principal ILE COBOL/400 llamado de un grupo de activación *NEW, el grupo de activación finaliza cuando se devuelve el control al programa de llamada. El grupo de activación cerrará todos los archivos y devolverá todos los recursos al sistema.

Como resultado de la finalización del grupo de activación, el programa ILE COBOL/400 llamado se colocará en su estado inicial.

Devolución del control desde un grupo de activación con nombre

Si realiza EXIT PROGRAM con la expresión CONTINUE desde un programa principal ILE COBOL/400 llamado en un grupo de activación con nombre, el grupo de activación permanece activo y se devuelve el control al programa de llamada. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que estuvieran cuando se utilizaron por última vez.

Si ejecuta las instrucciones STOP RUN o GOBACK desde un programa principal ILE COBOL/400 llamado en un grupo de activación con nombre, el grupo de activación finaliza cuando se devuelve el control al programa de llamada. El grupo de activación cerrará todos los archivos y devolverá todos los recursos al sistema.

Devolución desde el grupo de activación por omisión (*DFTACTGRP)

Si ejecuta las instrucciones STOP RUN o GOBACK desde un programa principal ILE COBOL/400 llamado en el grupo de activación por omisión (*DFTACTGRP) el grupo de activación continúa activo y se devuelve el control al programa de llamada. Todos los archivos y recursos utilizados en el grupo de activación se dejan en su último estado utilizado.

Devolución del control desde un subprograma

Para devolver el control desde un subprograma, éste puede finalizar con una instrucción EXIT PROGRAM, GOBACK o STOP RUN. Si el subprograma finaliza con una instrucción EXIT PROGRAM o GOBACK, el control se devuelve a su llamador inmediato sin finalizar la unidad de ejecución. Se genera una instrucción EXIT PROGRAM implícita si no hay ninguna instrucción ejecutable más en un programa llamado. Si el subprograma finaliza con una instrucción STOP RUN, se finalizan todos los programas de la unidad de ejecución hasta el límite de control más cercano y el control se devuelve al programa anterior al límite de control.

Normalmente, un subprograma se deja en su **último estado utilizado** cuando finaliza con EXIT PROGRAM o GOBACK. La próxima vez que se llame en la unidad de ejecución, los valores internos serán los mismos que se han dejado, aunque todas las instrucciones PERFORM se considerarán finalizadas y se restaurarán a sus valores iniciales. Por otro lado, se inicializará un programa principal cada vez que se llame. Existen dos excepciones:

- Un subprograma que se llame dinámicamente y luego se cancele estará en el estado inicial la próxima vez que se llame.
- Un programa que tenga la cláusula INITIAL especificada en el párrafo PROGRAM-ID estará en estado inicial cada vez que se llame.

Mantenimiento de la semántica de STOP RUN definida por la unidad de ejecución OPM COBOL/400

Para que la instrucción STOP RUN se comporte de forma compatible con una unidad de ejecución OPM COBOL/400, debe crearse la aplicación ILE COBOL/400 utilizando condiciones específicas. Consulte el apartado “Unidad de ejecución COBOL” en la página 176 para obtener una descripción de estas condiciones.

Ejemplos de devolución de control desde un programa ILE COBOL/400

Los ejemplos siguientes muestran el comportamiento de EXIT PROGRAM, STOP RUN y GOBACK en varias combinaciones de grupos de activación con Nombre, *NEW y *DFTACTGP.

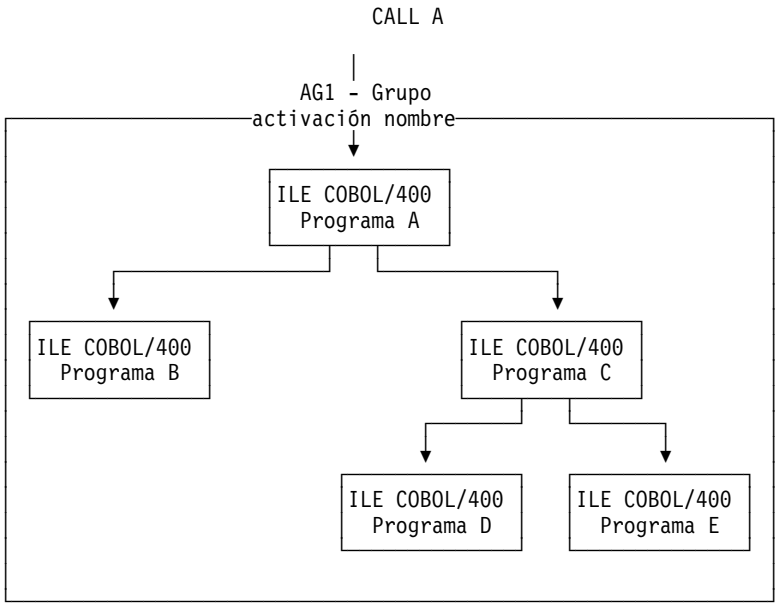


Figura 51. Ejemplo de comportamiento de EXIT PROGRAM, STOP RUN y GOBACK en un único grupo de activación con nombre

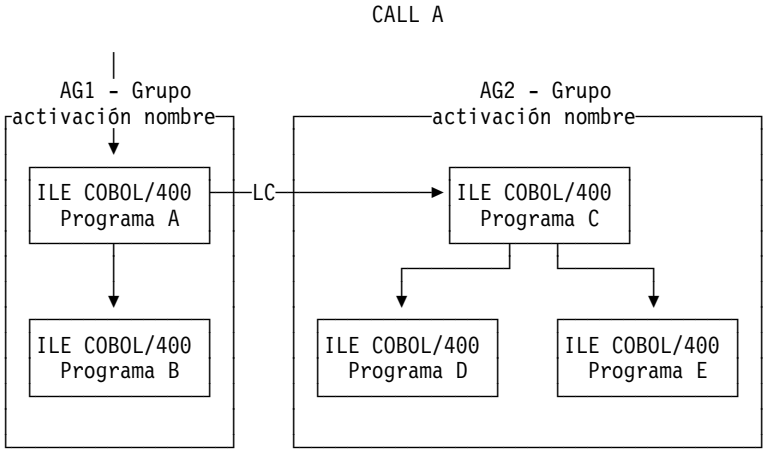
Instrucción	Programa A	Programa B	Programa C	Programa D	Programa E
EXIT PROGRAM	1	4	4	2	2
STOP RUN	3	3	3	3	3
GOBACK	3	4	4	2	2

1 Si se codifica EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa. EXIT PROGRAM con la expresión CONTINUE devuelve el control al llamador del Programa A y deja activo el grupo de activación. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.

2 El grupo de activación permanece activo y se devuelve el control al Programa C. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.

3 El grupo de activación finaliza y el control se devuelve al llamador del programa principal. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se restablecerán a su estado inicial.

4 El grupo de activación permanece activo y se devuelve el control al Programa A.Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.



NOTA: LC indica un límite de control.

Figura 52. Ejemplo de comportamiento de EXIT PROGRAM, STOP RUN y GOBACK en dos grupos de activación con nombre

Instrucción	Programa A	Programa B	Programa C	Programa D	Programa E
EXIT PROGRAM	1	5	1	2	2
STOP RUN	3	3	4	4	4
GOBACK	3	5	4	2	2

1 Si se ha utilizado una instrucción EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa. Si se ha utilizado una instrucción EXIT PROGRAM con la expresión CONTINUE, el grupo de activación permanece activo y se devuelve el control al programa o mandato de llamada. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.

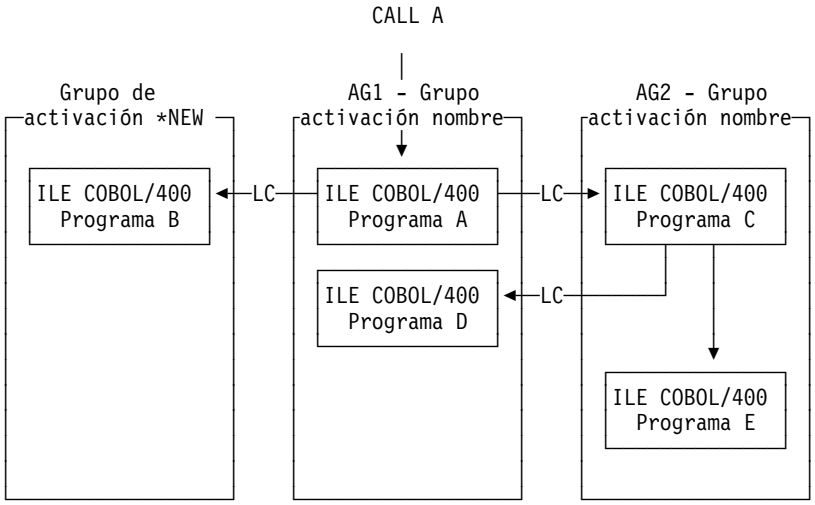
2 El grupo de activación permanece activo y se devuelve el control al Programa C. Todos los archivos y recursos utilizados en el grupo de activación se dejan en su último estado utilizado.

3 El grupo de activación se finaliza y el control se devuelve al llamador del programa principal. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

4 El grupo de activación se finaliza y el control se devuelve al Programa A. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de

compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

5 El grupo de activación permanece activo y se devuelve el control al Programa A.Todos los archivos y recursos utilizados en el grupo de activación se dejan en su último estado utilizado.



NOTA: LC indica un límite de control.

*Figura 53. Ejemplo de comportamiento de EXIT PROGRAM, STOP RUN y GOBACK en múltiples grupos de activación *NEW y con nombre*

Instrucción	Programa A	Programa B	Programa C	Programa D	Programa E
EXIT PROGRAM	1	5	1	2	2
STOP RUN	3	4	4	2	4
GOBACK	3	4	4	2	2

1 Si se ha utilizado una instrucción EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa. Si se ha utilizado una instrucción EXIT PROGRAM con la expresión CONTINUE, el grupo de activación permanece activo y se devuelve el control al programa o mandato de llamada. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.

2 El grupo de activación permanece activo y se devuelve el control al Programa C. Todos los archivos y recursos utilizados en el grupo de activación se dejan en su último estado utilizado.

3 El grupo de activación se finaliza y el control se devuelve al llamador del programa principal. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como

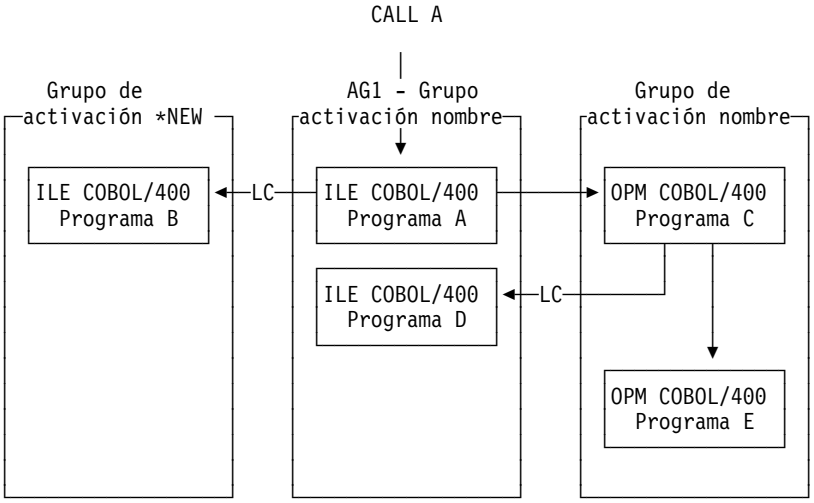
resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

4 El grupo de activación se finaliza y el control se devuelve al Programa A. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas que estuvieran activos en el grupo de activación se devolverán a su estado inicial.

5 Si se ha utilizado una instrucción EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa.

Si se ha utilizado una instrucción EXIT PROGRAM con la expresión CONTINUE, se devuelve el control al programa o mandato de llamada. En un grupo de activación *NEW, cuando un programa principal devuelve el control al llamador, se finaliza el grupo de activación. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Las operaciones pendientes de compromiso del ámbito del grupo de activación se comprometerán de forma implícita.

Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas que estuvieran activos en el grupo de activación se devolverán a su estado inicial.



NOTA: LC indica un límite de control.

*Figura 54. Ejemplo de comportamiento de EXIT PROGRAM, STOP RUN y GOBACK en grupos de activación *NEW, con nombre y *DFTACTGP*

Instrucción	Programa A	Programa B	Programa C	Programa D	Programa E
EXIT PROGRAM	1	6	7	2	2
STOP RUN	3	4	5	2	5
GOBACK	3	4	5	2	2

1 Si se ha utilizado una instrucción EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa. Si se ha utilizado una instrucción EXIT PROGRAM con la expresión CONTINUE, el grupo de activación permanece activo y se devuelve el control al programa o mandato de llamada. Todos los archivos y recursos utilizados en el grupo de activación se dejan en el estado en que se utilizaron por última vez.

2 El grupo de activación permanece activo y se devuelve el control al Programa C. Todos los archivos y recursos utilizados en el grupo de activación se dejan en su último estado utilizado.

3 El grupo de activación se finaliza y el control se devuelve al llamador del programa principal. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

4 El grupo de activación se finaliza y el control se devuelve al Programa A. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Se comprometerá implícitamente cualquier operación pendiente de compromiso del ámbito del grupo de activación. Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

5 El grupo de activación permanece activo y se devuelve el control al Programa A. Se cierran todos los archivos abiertos por el Programa C o el Programa E. Se comprometerá implícitamente cualquier operación pendiente de compromiso para archivos abiertos por el Programa C o el Programa E. El almacenamiento se libera para el Programa C y para el Programa E.

6 Si se ha utilizado una instrucción EXIT PROGRAM sin la expresión CONTINUE, no se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa.

Si se ha utilizado una instrucción EXIT PROGRAM con la expresión CONTINUE, se devuelve el control al programa o mandato de llamada. En un grupo de activación *NEW, cuando un programa principal devuelve el control al llamador, se finaliza el grupo de activación. El grupo de activación cerrará todos los archivos del ámbito del grupo de activación. Las operaciones pendientes de compromiso del ámbito del grupo de activación se comprometerán de forma implícita.

Todos los recursos asignados al grupo de activación se devolverán al sistema. Como resultado de la finalización del grupo de activación, todos los programas activos del grupo se colocarán en su estado inicial.

7 No se procesa ninguna operación ya que la instrucción se encuentra en un programa principal. El proceso continúa con la siguiente instrucción del programa.

Transferencia de información sobre códigos de devolución de control (registro especial RETURN-CODE)

Puede utilizar el registro especial RETURN-CODE para transferir y recibir códigos de retorno entre programas ILE COBOL/400. Puede establecer el registro especial RETURN-CODE antes de volver de un programa ILE COBOL/400 llamado.

Cuando se utiliza en programas anidados, el registro especial RETURN-CODE se define implícitamente como GLOBAL en el programa ILE COBOL/400 más externo. Cualquier cambio realizado al registro especial RETURN-CODE es global para todos los programas ILE COBOL/400 incluidos en el objeto de módulo.

Cuando un programa ILE COBOL/400 vuelve a su llamador, el contenido de su registro especial RETURN-CODE se transfiere al registro especial RETURN-CODE del programa de llamada.

Cuando se devuelve el control al sistema operativo desde un programa principal ILE COBOL/400, el contenido del registro especial RETURN-CODE se devuelve como un código de retorno de usuario.

Transferencia y compartimiento de datos entre programas

Existen muchas formas de transferir o compartir datos entre programas ILE COBOL/400:

- Los datos pueden declararse como GLOBAL para que puedan utilizarlos los programas anidados.
- Los datos pueden devolverse al programa de llamada utilizando la expresión RETURNING de la instrucción CALL.
- Cuando se ejecuta la instrucción CALL, los datos pueden transferirse a un programa llamado BY REFERENCE, BY VALUE o BY CONTENT (POR REFERENCIA, POR VALOR o POR CONTENIDO).
- Los datos declarados como EXTERNAL pueden compartirlos programas compilados por separado. Los datos EXTERNAL también pueden compartirse entre programas anidados ILE COBOL/400 incluidos en un objeto de módulo.
- Los archivos declarados como EXTERNAL pueden compartirlos programas compilados por separado. Los archivos EXTERNAL también pueden compartirse entre programas anidados ILE COBOL/400 incluidos en un objeto de módulo.
- Los punteros pueden utilizarse cuando se desee transferir y recibir direcciones de elementos de datos ubicados dinámicamente.
- Los datos pueden transferirse utilizando Áreas de datos

Comparación de datos locales y globales

El concepto de datos locales y globales es aplicable sólo a los programas anidados.

Los **datos locales** son sólo accesibles desde dentro del programa en el cual se declaran los datos locales.Los datos locales no son visibles ni accesibles para ningún programa fuera del programa donde se han declarado; esto incluye tanto programas contenidos como los programas que los contienen.

Todos los datos se consideran datos locales a menos que se declaren explícitamente como datos globales.

Los **datos globales** son accesibles desde dentro del programa en el que se han declarado los datos globales o desde dentro de cualquier otro programa anidado contenido, directa o indirectamente, en el programa donde se han declarado los datos globales.

Los nombres de datos, los nombres de archivos y los nombres de registros pueden declararse como globales.

Para declarar un nombre de datos como global, especifique la palabra GLOBAL en la entrada de la descripción de datos donde se declara el nombre de datos o en otra entrada a la cual esté subordinada dicha entrada de descripción de datos.

Para declarar un nombre de archivo como global, especifique la palabra GLOBAL en la entrada de descripción de archivos para ese nombre de archivo.

Para declarar un nombre de registro como global, especifique la palabra GLOBAL en la entrada de descripción de registro donde se declara el nombre de registro o, en el caso de entradas de descripción de registros en la FILE SECTION, especifique la palabra GLOBAL en la entrada de descripción de archivos para el nombre de archivo asociado con la entrada de descripción de registros.

Para obtener una descripción más detallada de la cláusula GLOBAL, consulte la *ILE COBOL/400 Reference*.

Transferencia de datos utilizando CALL...BY REFERENCE, BY VALUE o BY CONTENT

BY REFERENCE significa que cualquier cambio que el subprograma realice en las variables que recibe será visible para el programa de llamada.

BY CONTENT significa que el programa de llamada sólo está transfiriendo el **contenido** del *literal* o del *identificador*. Con una instrucción CALL...BY CONTENT, el programa llamado no puede cambiar el valor del *literal* o del *identificador* del programa de llamada, ni siquiera modificando los parámetros recibidos.

BY VALUE significa que el programa de llamada está transfiriendo el valor del *literal* o del *identificador*, no una referencia al elemento de envío. El programa llamado puede cambiar el parámetro en el programa llamado. Sin embargo, como el subprograma sólo tiene acceso a una copia temporal del elemento de envío, dichos cambios no afectan al argumento del programa de llamada.

Transferir elementos de datos BY REFERENCE, BY VALUE o BY CONTENT depende de lo que desee que el programa haga con los datos:

- Si desea que la definición del argumento de la instrucción CALL en el programa de llamada y la definición del parámetro en el programa llamado compartan la misma memoria, especifique:

  ```
  CALL...BY REFERENCE identificador
  ```

  Cualquier cambio realizado por el subprograma al parámetro afectará al argumento del programa de llamada.

- Si desea transferir la dirección de una área de registro a un programa llamado, especifique:

  ```
  CALL...BY REFERENCE ADDRESS OF nombre-de-registro
  ```

  El subprograma recibe el registro especial ADDRESS OF para el nombre de registro que especifique.

  Debe definir el nombre de registro como un elemento de nivel-01 o nivel-77 de la LINKAGE SECTION del programa llamado y de llamada. Se proporciona un registro especial ADDRESS OF separado para cada registro de la Linkage Section.

- Si desea transferir la dirección de cualquier elemento de datos en la DATA DIVISION a un programa llamado, especifique:

  ```
  CALL...BY CONTENT ADDRESS OF nombre-de-elemento-de-datos
  ```

- Si no desea que la definición del argumento de la instrucción CALL en el programa de llamada y la definición del parámetro en el subprograma llamado compartan la misma memoria, especifique:

  ```
  CALL...BY CONTENT identificador
  ```

- Si desea transferir datos a programas ILE que requieren parámetros BY VALUE utilice:

  ```
  CALL...BY VALUE elemento
  ```

- Si desea transferir un entero numérico de distintas longitudes especifique:

  ```
  CALL...BY VALUE entero-1 SIZE entero-2
  ```

  El entero numérico se transfiere como un valor binario de longitud entero-2.La expresión SIZE es opcional. Si no se especifica, entero-1 se transfiere como un número binario de 4 bytes.

- Si desea llamar a una función ILE C, C++ o RPG con un valor de retorno de función, utilice:

  ```
  CALL...RETURNING identificador
  ```

- Si desea transferir un valor literal a un programa llamado, especifique:

  ```
  CALL...BY CONTENT literal
  ```

  El programa llamado no puede cambiar el valor del literal.

- Si desea transferir la longitud de un elemento de datos especifique:

  ```
  CALL...BY CONTENT LENGTH OF identificador
  ```

  El programa de llamada transfiere la longitud de *identificador* desde su registro especial LENGTH OF.

- Si desea transferir un elemento de datos y su longitud a un subprograma, especifique una combinación de BY REFERENCE y BY CONTENT. Por ejemplo:

  ```
  CALL 'ERRPROC' USING BY REFERENCE A
       BY CONTENT LENGTH OF A.
  ```

- Si no desea que el programa llamado reciba un argumento correspondiente o si desea que el programa llamado utilice el valor por omisión para el argumento, especifique la palabra OMITTED en lugar del elemento de datos en la instrucción CALL... BY REFERENCE o CALL... BY CONTENT. Por ejemplo:

```
CALL...BY REFERENCE OMITTED

CALL...BY CONTENT OMITTED
```

En el programa llamado, puede utilizar la API CEETSTA para determinar si un parámetro especificado es de tipo OMITTED o no.

- Si desea transferir elementos de datos con **descriptores operativos**, especifique la cláusula LINKAGE TYPE IS PRC...USING ALL DESCRIBED en el párrafo SPECIAL-NAMES. Luego utilice la instrucción CALL...BY REFERENCE, CALL...BY CONTENT o CALL...BY VALUE para transferir los datos. Los descriptores operativos proporcionan información descriptiva al procedimiento ILE llamado en los casos en que el procedimiento ILE llamado no puede anticipar con precisión la forma de los elementos de datos que se transfieren. Utilice los descriptores operativos cuando los espere un procedimiento ILE llamado, escrito en un lenguaje ILE distinto, y una API de enlace ILE. Consulte el manual *ILE Conceptos* para obtener más información sobre los descriptores operativos. Por ejemplo:

```
SPECIAL-NAMES.  LINKAGE TYPE PRC FOR 'ERRPROC'
                USING ALL DESCRIBED.

                ⋮
CALL 'ERRPROC' USING BY REFERENCE identificador.
```

o

```
SPECIAL-NAMES.  LINKAGE TYPE PRC FOR 'ERRPROC'
                USING ALL DESCRIBED.

                ⋮
CALL 'ERRPROC' USING BY CONTENT identificador.
```

Los elementos de datos de un programa de llamada pueden describirse en la Linkage Section de todos los programas que los llamen directa o indirectamente. En este caso, el almacenamiento para estos elementos se asigna en el programa más externo de llamada.

Descripción de argumentos en el programa de llamada

Los datos que se transfieren desde un programa de llamada se denominan **argumentos**. En el programa de llamada, los argumentos se describen en la Data Division de la misma forma que otros elementos de datos de la Data Division. A menos que estén en la Linkage Section, el almacenamiento para estos elementos se asigna en el programa de llamada. Si se hace referencia a unos datos en un archivo, éste debe estar abierto cuando se haga referencia a los datos. Codifique la cláusula USING de la instrucción CALL para transferir los argumentos.

Descripción de parámetros en el programa llamado

Los datos recibidos en un programa llamado se denominan **parámetros**. En el programa llamado, los parámetros se describen en la Linkage Section. Codifique la cláusula USING después de la cabecera PROCEDURE-DIVISION para recibir los parámetros.

Escritura de la Linkage Section en el programa llamado: Debe conocer qué se está transfiriendo desde el programa de llamada y configurar la Linkage Section en el programa llamado para que lo acepte. Para el programa llamado, no tiene importancia qué cláusula de la instrucción CALL utilice para transferir los datos (BY

REFERENCE, BY VALUE o BY CONTENT). En todos los casos, el programa llamado debe describir los datos que recibe. Lo hace en la Linkage Section.

El número de *nombres-de-datos* de la lista de *identificadores* de un programa llamado no debe ser superior al número de *nombres-de-datos* de la lista de *identificadores* del programa de llamada. Existe una correspondencia situacional de uno a uno; es decir, el primer *identificador* del programa de llamada se transfiere al primer *identificador* del programa llamado, y así sucesivamente. El compilador ILE COBOL/400 no exige la coherencia en cuanto al número de argumentos y al número de parámetros, como tampoco lo hace en cuanto al tipo y tamaño entre un argumento y su parámetro correspondiente.

Las incoherencias en cuanto a número de argumentos y número de parámetros pueden provocar excepciones en la ejecución. Para una llamada dinámica de programa, cuando el número de argumentos es superior al número de parámetros, se genera una excepción de ejecución en el programa de llamada al intentar ejecutar la instrucción CALL. Esta excepción puede capturarse si la expresión ON EXCEPTION está especificada en la instrucción CALL. Cuando el número de argumentos es inferior al número de parámetros, no se genera una excepción de ejecución en el programa de llamada al ejecutar la instrucción CALL. En su lugar, se genera una excepción de puntero en el programa llamado cuando éste intenta acceder a un parámetro no proporcionado.

Si un argumento se transfirió BY VALUE, la cabecera PROCEDURE DIVISION del subprograma debe indicarlo:

```
PROCEDURE DIVISION USING BY VALUE DATA-ITEM.
```

Si un argumento se transfirió BY REFERENCE o BY CONTENT, la cabecera PROCEDURE DIVISION no tiene que indicar cómo se transfirió el argumento. La cabecera puede ser:

```
PROCEDURE DIVISION USING DATA-ITEM
```

o:

```
PROCEDURE DIVISION USING BY REFERENCE DATA-ITEM
```

Agrupación de datos para su transferencia

Considere agrupar todos los elementos de datos que desea transferir entre programas y en colocarlos bajo un elemento nivel-01. Si lo hace, puede transferir un único registro nivel-01 entre los programas. Como ejemplo de este método, vea la Figura 55 en la página 203.

Para disminuir aún más la posibilidad de registros no coincidentes, coloque el registro de nivel-01 en un miembro de copia y cópielo en ambos programas. (Es decir, cópielo en la Working-Storage Section del programa de llamada y en la Linkage Section del programa llamado).

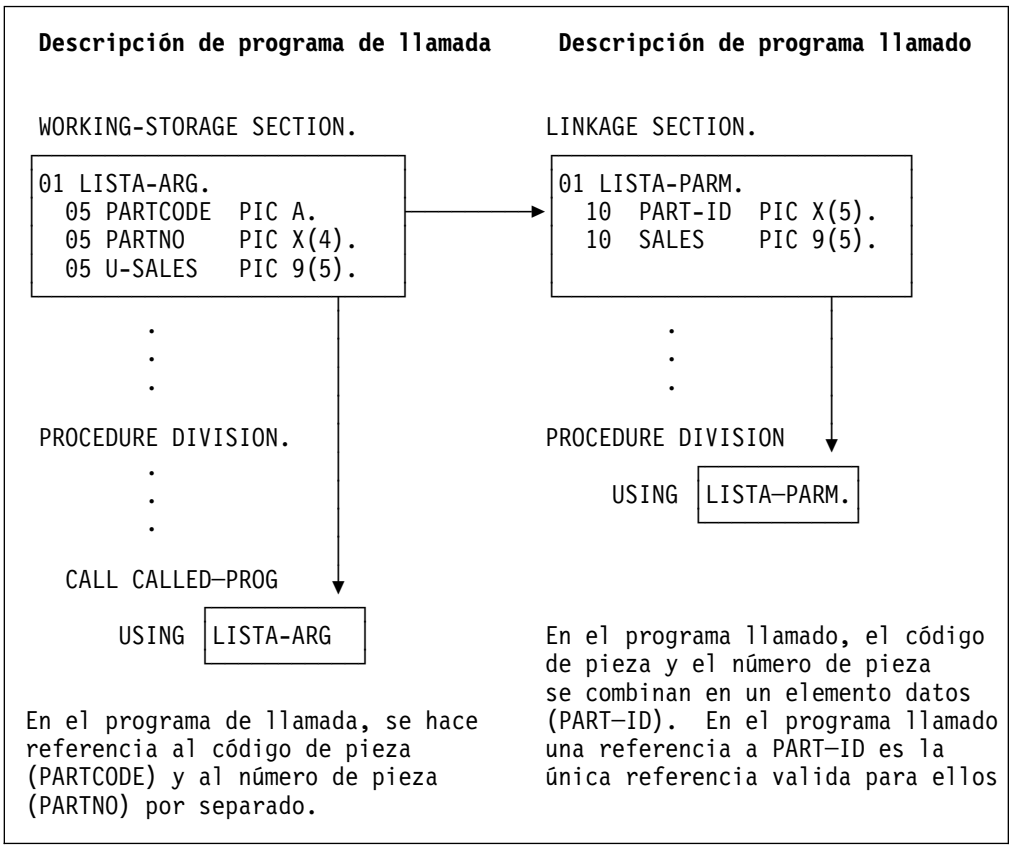


Figura 55. Elementos de datos comunes durante el enlace de subprogramas

Compartimento de datos EXTERNAL

Los programas ILE COBOL/400 compilados por separado (incluso los programas dentro de una secuencia de programas fuente ILE COBOL/400) pueden compartir elementos de datos utilizando la cláusula EXTERNAL. Estos datos EXTERNAL se manejan como exportación débil. Consulte la publicación *ILE Conceptos* para obtener más información sobre exportaciones fuertes y débiles.

Especifique la cláusula EXTERNAL en la descripción de datos de nivel 01 de la Working-Storage Section del programa ILE COBOL/400 y se aplicarán las normas siguientes:

1. Los elementos subordinados a un elemento de grupo EXTERNAL son también EXTERNAL.
2. El nombre utilizado para el elemento de datos no puede utilizarse en otro elemento EXTERNAL dentro del mismo programa.
3. La cláusula VALUE no puede especificarse para ningún elemento de grupo ni elemento subordinado de tipo EXTERNAL.
4. Los datos EXTERNAL no pueden inicializarse y su valor inicial durante la ejecución es cero hexadecimal.

Cualquier programa ILE COBOL/400 dentro de una unidad de ejecución, que tenga la misma descripción de datos para el elemento que el programa que contiene el elemento, puede acceder y procesar el elemento de datos.Por ejemplo, si el programa A tuviera la descripción de datos siguiente:

```
01 EXT-ITEM1          PIC 99 EXTERNAL.
```

El programa B podría acceder al elemento de datos si tuviera la descripción de datos idéntica en su Working-Storage Section.

El tamaño debe ser el mismo para el mismo elemento de datos EXTERNAL con nombre en todos los objetos de módulo que lo declaren.Si en programas ILE COBOL/400 múltiples de una unidad de compilación se declaran elementos de datos EXTERNAL de tamaños distintos, se utilizará el tamaño superior para representar el elemento de datos.

Además, cuando elementos de datos EXTERNAL de tamaño distinto del mismo nombre se representan en objetos de programa o en programas de servicio múltiples que se activan en el mismo grupo de activación, y el último objeto de programa o programa de servicio activado tiene un tamaño superior para el mismo elemento de datos EXTERNAL con nombre, la activación del objeto de programa o del programa de servicio activado en último lugar fallará.

El compilador ILE COBOL/400 no impone la coherencia de tipos entre los elementos de datos del mismo nombre declarados en programas ILE COBOL/400 múltiples. El usuario debe asegurarse de que la utilización de estos elementos de datos sea coherente.

Recuerde, cualquier programa que tenga acceso a un elemento de datos EXTERNAL puede cambiar su valor.No utilice esta cláusula para elementos de datos que deba proteger.

Compartimento de archivos EXTERNAL

Utilizar la cláusula EXTERNAL para archivos permite a los programas compilados por separado dentro de la unidad de ejecución acceder a los archivos comunes. Estos archivos EXTERNAL se manejan como exportaciones débiles. Consulte publicación *ILE Conceptos* para obtener más información sobre exportaciones fuertes y débiles.

Cuando un archivo EXTERNAL se define en programas ILE COBOL/400 múltiples, una vez lo abre uno de estos programas ILE COBOL/400, el archivo es accesible a todos los programas. De forma similar, si uno de los programas cierra el archivo EXTERNAL, los demás programas ya no podrán acceder al mismo.

Para programas ILE COBOL/400 múltiples en objetos de módulo múltiples, se realiza una comprobación de coherencia de ejecución la primera vez que se llama al programa ILE COBOL/400 que declara un archivo EXTERNAL dado, para ver si la definición en ese objeto de módulo es coherente con las definiciones de programas ILE COBOL/400 ya llamados en otros objetos de módulos. Si se encuentra alguna incoherencia, se emite el mensaje de excepción de ejecución.

El ejemplo de la Figura 56 en la página 206 muestra algunas de las ventajas de utilizar archivos EXTERNAL:

- El programa principal puede hacer referencia al área de registro del archivo, aunque el programa principal no contenga ninguna instrucción de entrada-salida.
- Cada subprograma puede controlar una única función de entrada-salida, como por ejemplo OPEN o READ.

- Cada programa tiene acceso al archivo.

La tabla siguiente proporciona el nombre del programa (o del subprograma) para el ejemplo de la Figura 56 en la página 206 y describe su función.

Tabla 11. Nombres de programas para un ejemplo de entrada-salida que utilice archivos EXTERNAL

Nombre	Función
EF1MAIN	Este es el programa principal. Llama a todos los subprogramas y luego verifica el contenido de una área de registro.
EF1OPENO	Este programa abre el archivo externo para salida y comprueba el Código de estado del archivo.
EF1WRITE	Este programa graba un registro en el archivo externo y comprueba el Código de estado del archivo.
EF1OPENI	Este programa abre el archivo externo para entrada y comprueba el Código de estado del archivo.
EF1READ	Este programa lee el registro del archivo externo y comprueba el Código de estado del archivo.
EF1CLOSE	Este programa cierra el archivo externo y comprueba el Código de estado del archivo.

El programa de ejemplo también utiliza la cláusula EXTERNAL para un elemento de datos de la Working-Storage Section. Este elemento se utiliza para comprobar los Códigos de estado de los archivos.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10/49:12        Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  EF1MAIN.
          000030*
          000040* Este es el programa principal que controla
          000050* el proceso del archivo externo.
          000060*
          000070
    3     000080 ENVIRONMENT DIVISION.
    4     000090 INPUT-OUTPUT SECTION.
    5     000100 FILE-CONTROL.
    6     000110      SELECT EF1
    7     000120         ASSIGN TO DISK-EFILE1
    8     000130         FILE STATUS IS EFS1
    9     000140         ORGANIZATION IS SEQUENTIAL.
          000150
   10     000160 DATA DIVISION.
   11     000170 FILE SECTION.
   12     000180 FD  EF1 IS EXTERNAL
          000190           RECORD CONTAINS 80 CHARACTERS.
   13     000200 01  EF-RECORD-1.
   14     000210     05  EF-ITEM-1     PIC X(80).
          000220
   15     000230 WORKING-STORAGE SECTION.
   16     000240 01  EFS1              PIC 99 EXTERNAL.
          000250
   17     000260 PROCEDURE DIVISION.
          000270 EF1MAIN-PROGRAM SECTION.
          000280 MAINLINE.
   18     000290      CALL "EF1OPENO"
   19     000300      CALL "EF1WRITE"
   20     000310      CALL "EF1CLOSE"
   21     000320      CALL "EF1OPENI"
   22     000330      CALL "EF1READ"
   23     000340      IF EF-RECORD-1 = "Primer registro" THEN
   24     000350         DISPLAY "Primer registro correcto"
          000360      ELSE
   25     000370         DISPLAY "Primer registro incorrecto"
   26     000380         DISPLAY "Esperaba:  Primer registro"
   27     000390         DISPLAY "Encontró:    " EF-RECORD-1
          000400      END-IF
   28     000410      CALL "EF1CLOSE"
   29     000420      GOBACK.
   30     000430 END PROGRAM EF1MAIN.
          000440
          000460

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 1 de 6). Entrada-salida utilizando archivos EXTERNAL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10:49:12        Página     5

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000470 IDENTIFICATION DIVISION.
    2     000480 PROGRAM-ID.  EF1OPENO.
          000490*
          000500* Este programa abre el archivo externo para salida.
          000510*
          000520
    3     000530 ENVIRONMENT DIVISION.
    4     000540 INPUT-OUTPUT SECTION.
    5     000550 FILE-CONTROL.
    6     000560      SELECT EF1
    7     000570         ASSIGN TO DISK-EFILE1
    8     000580         FILE STATUS IS EFS1
    9     000590         ORGANIZATION IS SEQUENTIAL.
          000600
   10     000610 DATA DIVISION.
   11     000620 FILE SECTION.
   12     000630 FD  EF1 IS EXTERNAL
          000640            RECORD CONTAINS 80 CHARACTERS.
   13     000650 01  EF-RECORD-1.
   14     000660     05  EF-ITEM-1     PIC X(80).
          000670
   15     000680 WORKING-STORAGE SECTION.
   16     000690 01  EFS1              PIC 99 EXTERNAL.
          000700
   17     000710 PROCEDURE DIVISION.
          000720 EF1OPENO-PROGRAM SECTION.
          000730 MAINLINE.
   18     000740      OPEN OUTPUT EF1
   19     000750      IF EFS1 NOT = 0 THEN
   20     000760         DISPLAY "Estado archivo " EFS1 " en OPEN OUTPUT"
   21     000770         STOP RUN
          000780      END-IF
   22     000790      GOBACK.
   23     000800 END PROGRAM EF1OPENO.
          000810
          000830

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 2 de 6). Entrada-salida utilizando archivos EXTERNAL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10:49:12        Página     8

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000840 IDENTIFICATION DIVISION.
    2     000850 PROGRAM-ID.  EF1WRITE.
          000860*
          000870* Este programa graba un registro en el archivo externo.
          000880*
          000890
    3     000900 ENVIRONMENT DIVISION.
    4     000910 INPUT-OUTPUT SECTION.
    5     000920 FILE-CONTROL.
    6     000930      SELECT EF1
    7     000940         ASSIGN TO DISK-EFILE1
    8     000950         FILE STATUS IS EFS1
    9     000960         ORGANIZATION IS SEQUENTIAL.
          000970
   10     000980 DATA DIVISION.
   11     000990 FILE SECTION.
   12     001000 FD  EF1 IS EXTERNAL
          001010            RECORD CONTAINS 80 CHARACTERS.
   13     001020 01  EF-RECORD-1.
   14     001030     05  EF-ITEM-1     PIC X(80).
          001040
   15     001050 WORKING-STORAGE SECTION.
   16     001060 01  EFS1              PIC 99 EXTERNAL.
          001070
   17     001080 PROCEDURE DIVISION.
          001090 EF1WRITE-PROGRAM SECTION.
          001100 MAINLINE.
   18     001110      MOVE "Primer registro" TO EF-RECORD-1
   19     001120      WRITE EF-RECORD-1
   20     001130      IF EFS1 NOT = 0 THEN
   21     001140         DISPLAY "Estado archivo " EFS1 " en WRITE"
   22     001150         STOP RUN
          001160      END-IF
   23     001170      GOBACK.
   24     001180 END PROGRAM EF1WRITE.
          001190
          001210

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 3 de 6). Entrada-salida utilizando archivos EXTERNAL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10:49:12        Página    11

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     001220 IDENTIFICATION DIVISION.
    2     001230 PROGRAM-ID.  EF1OPENI.
          001240*
          001250* Este programa abre el archivo externo para entrada.
          001260*
          001270
    3     001280 ENVIRONMENT DIVISION.
    4     001290 INPUT-OUTPUT SECTION.
    5     001300 FILE-CONTROL.
    6     001310      SELECT EF1
    7     001320         ASSIGN TO DISK-EFILE1
    8     001330         FILE STATUS IS EFS1
    9     001340         ORGANIZATION IS SEQUENTIAL.
          001350
   10     001360 DATA DIVISION.
   11     001370 FILE SECTION.
   12     001380 FD  EF1 IS EXTERNAL
          001390            RECORD CONTAINS 80 CHARACTERS.
   13     001400 01  EF-RECORD-1.
   14     001410     05  EF-ITEM-1     PIC X(80).
          001420
   15     001430 WORKING-STORAGE SECTION.
   16     001440 01  EFS1              PIC 99 EXTERNAL.
          001450
   17     001460 PROCEDURE DIVISION.
          001470 EF1OPENI-PROGRAM SECTION.
          001480 MAINLINE.
   18     001490      OPEN INPUT EF1
   19     001500      IF EFS1 NOT = 0 THEN
   20     001510         DISPLAY "Estado archivo " EFS1 " en OPEN INPUT"
   21     001520         STOP RUN
          001530      END-IF
   22     001540      GOBACK.
   23     001550 END PROGRAM EF1OPENI.
          001560
          001580

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 4 de 6). Entrada-salida utilizando archivos EXTERNAL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10:49:12         Página    14

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     001590 IDENTIFICATION DIVISION.
    2     001600 PROGRAM-ID.  EF1READ.
          001610*
          001620* Este programa lee un registro del archivo externo.
          001630*
          001640
    3     001650 ENVIRONMENT DIVISION.
    4     001660 INPUT-OUTPUT SECTION.
    5     001670 FILE-CONTROL.
    6     001680      SELECT EF1
    7     001690         ASSIGN TO DISK-EFILE1
    8     001700         FILE STATUS IS EFS1
    9     001710         ORGANIZATION IS SEQUENTIAL.
          001720
   10     001730 DATA DIVISION.
   11     001740 FILE SECTION.
   12     001750 FD  EF1 IS EXTERNAL
          001760            RECORD CONTAINS 80 CHARACTERS.
   13     001770 01  EF-RECORD-1.
   14     001780     05  EF-ITEM-1     PIC X(80).
          001790
   15     001800 WORKING-STORAGE SECTION.
   16     001810 01  EFS1              PIC 99 EXTERNAL.
          001820
   17     001830 PROCEDURE DIVISION.
          001840 EF1READ-PROGRAM SECTION.
          001850 MAINLINE.
   18     001860      READ EF1
   19     001870      IF EFS1 NOT = 0 THEN
   20     001880         DISPLAY "Estado archivo " EFS1 " en READ"
   21     001890         STOP RUN
          001900      END-IF
   22     001910      GOBACK.
   23     001920 END PROGRAM EF1READ.
          001930
          001950

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 5 de 6). Entrada-salida utilizando archivos EXTERNAL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/EXTLFL          AS400SYS   96/07/04 10:49:12        Página    17

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     001960 IDENTIFICATION DIVISION.
    2     001970 PROGRAM-ID.  EF1CLOSE.
          001980*
          001990* Este programa cierra el archivo externo.
          002000*
          002010
    3     002020 ENVIRONMENT DIVISION.
    4     002030 INPUT-OUTPUT SECTION.
    5     002040 FILE-CONTROL.
    6     002050      SELECT EF1
    7     002060         ASSIGN TO DISK-EFILE1
    8     002070         FILE STATUS IS EFS1
    9     002080         ORGANIZATION IS SEQUENTIAL.
          002090
   10     002100 DATA DIVISION.
   11     002110 FILE SECTION.
   12     002120 FD  EF1 IS EXTERNAL
          002130            RECORD CONTAINS 80 CHARACTERS.
   13     002140 01  EF-RECORD-1.
   14     002150     05  EF-ITEM-1     PIC X(80).
          002160
   15     002170 WORKING-STORAGE SECTION.
   16     002180 01  EFS1              PIC 99 EXTERNAL.
          002190
   17     002200 PROCEDURE DIVISION.
          002210 EF1CLOSE-PROGRAM SECTION.
          002220 MAINLINE.
   18     002230      CLOSE EF1
   19     002240      IF EFS1 NOT = 0 THEN
   20     002250         DISPLAY "Estado archivo " EFS1 " en CLOSE"
   21     002260         STOP RUN
          002270      END-IF
   22     002280      GOBACK.
   23     002290 END PROGRAM EF1CLOSE.
          002300

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 56 (Parte 6 de 6). Entrada-salida utilizando archivos EXTERNAL

Transferencia de datos utilizando punteros

Puede utilizar un puntero en un programa ILE COBOL/400 cuando desee transferir y recibir direcciones de un elemento de datos ubicado dinámicamente.

Para obtener una descripción completa de cómo se utilizan los punteros en un programa ILE COBOL/400, consulte Capítulo 11, “Utilización de punteros en un programa ILE COBOL/400” en la página 235.

Transferencia de datos utilizando áreas de datos

Un área de datos es un objeto OS/400 utilizado para comunicar datos, como valores variables, entre programas de un trabajo y entre trabajos. Un área de datos puede crearse y declararse en un programa antes de que se utilice en ese programa o trabajo. Para obtener información sobre cómo crear y declarar un área de datos, vea el manual *CL Programación*.

Utilización del área de datos local

El área de datos local puede utilizarse para transferir cualquier información que desee entre programas de un trabajo. Esta información puede estar formada por datos sin formato, como mensajes informales, o por un conjunto de campos completamente estructurados o con formato.

Los elementos de datos de coma flotante internos y externos pueden transferirse utilizando el área de datos local. Los números de coma flotante internos que se graban en el área de datos local utilizando la instrucción DISPLAY se convierten a números de coma flotante externos.

El sistema crea automáticamente una área de datos local para cada trabajo. El área de datos local se define fuera del programa ILE COBOL/400 como un área de 1024 bytes.

Cuando se somete un trabajo, el área de datos local del trabajo de sumisión se copia en el área de datos local del trabajo sometido. Si no hay ningún trabajo de sumisión, el área de datos local se inicializa en espacios en blanco.

Un programa ILE COBOL/400 puede acceder al área de datos local para su trabajo con las instrucciones ACCEPT y DISPLAY, utilizando un nombre nemotécnico asociado con el nombre de entorno en LOCAL-DATA.

Sólo hay una área de datos local asociada con cada trabajo. Incluso si un sólo trabajo adquiere varias estaciones de trabajo, sólo existe una área de datos local para ese trabajo. *No* existe un área de datos local para cada estación de trabajo.

Utilización de las áreas de datos que cree

Puede transferir datos entre programas utilizando las áreas de datos que cree. Esta información puede estar formada por datos sin formato, como mensajes informales o por un conjunto de campos completamente estructurados o con formato. Debe especificar la biblioteca y el nombre del área de datos cuando los cree.

Si utiliza los formatos de Área de datos (frente a los formatos de Área de datos local) de las instrucciones ACCEPT y DISPLAY, podrá acceder a estas áreas de datos. La expresión FOR le permite especificar el nombre del área de datos. Opcionalmente, también puede especificar una expresión IN LIBRARY para indicar la biblioteca OS/400 en la que existe el área de datos. Si no se especifica la expresión IN LIBRARY, la biblioteca toma por omisión el valor *LIBL.

Cuando utiliza la instrucción DISPLAY para grabar datos en un área de datos que haya creado, el sistema la bloquea con un bloqueo LEAR (Bloqueo exclusivo permitir lectura) antes de grabar en el área de datos. Si existe algún otro bloqueo en el área de datos, no se aplica el bloqueo LEAR y no se graba en el área de datos. Si especifica la expresión WITH LOCK, podrá mantener el área de datos bloqueada una vez se complete la operación DISPLAY. Si utiliza la instrucción ACCEPT para recuperar datos de un área de datos que haya creado, el sistema aplica un bloqueo LSRD (Bloqueo compartido para lectura) para evitar que se modifique el área de datos mientras se lee. Cuando se completa la lectura, se elimina el bloqueo LSRD y, si se ha especificado una expresión WITH LOCK, se mantiene un bloqueo LEAR sobre el área de datos.

| Con las instrucciones ACCEPT y DISPLAY, si no se ha especificado una expresión WITH LOCK, se eliminarán todos los bloqueos LEAR realizados con anterioridad a la instrucción.

| En ILE COBOL/400 sólo se colocará un bloqueo LEAR en el área de datos mientras esté activa la unidad de ejecución de COBOL (grupo de activación). Si hay áreas de datos bloqueadas cuando finaliza un grupo de activación, se eliminarán los bloqueos.

| Una condición ON EXCEPTION puede deberse a diferentes causas:

- El área de datos especificada para la palabra FOR:
  - No se encuentra
  - El usuario no tiene autorización sobre el área de datos
  - El área de datos está bloqueada por un grupo de activación anterior o por otro trabajo
- La posición AT:
  - Era inferior a 1 o superior a la longitud del área de datos

| Los elementos de datos de coma flotante internos y externos pueden transferirse utilizando un área de datos. Los números de coma flotante internos que se graban en el área de datos utilizando la instrucción DISPLAY se convierten a números de coma flotante externos.

* ILE COBOL/400 da soporte a áreas de datos decimales (*DEC), de caracteres (*CHAR), lógicas (*LGL) y DDM (*DDM). Con independencia del tipo de área de datos, la información que se mueve a un área de datos y desde la misma se justifica por la izquierda. Cuando se hace referencia a un área de datos decimal o a un área de datos lógica, si se especifica la posición AT ésta debe ser 1.

* Los datos se mueven a o desde un área de datos decimal en formato empaquetado. Un área de datos decimal se crea con un número de dígitos totales y de dígitos decimales determinado. Este mismo número de dígitos debe declararse en un programa ILE COBOL/400 que acceda al área de datos decimal. Por ejemplo:

* - Mandato CL para crear el área de datos:

```
CRTDTAARA DTAARA(QGPL/DECDATA) TYPE(*DEC) LEN(5 2)
```

* - Programa ILE COBOL/400 parcial para acceder al área de datos:

```
*       WORKING-STORAGE SECTION.
* 01 data-value.
*    05 returned-packed1  pic s9(3)v9(2) packed-decimal.

*  PROCEDURE DIVISION.
*       move 345.67 to returned-packed1.

*       DISPLAY  data-value UPON data-area
*          FOR "DECDATA" LIBRARY "QGPL".

*       ACCEPT   data-value FROM data-area
*          FOR "DECDATA" LIBRARY "QGPL".
```

Utilización del área de datos de parámetros de inicialización de programa (PIP)

El área de datos PIP la utiliza un trabajo de prearranque. Generalmente, un trabajo de prearranque es un trabajo de un sistema remoto bajo ICF que se arranca y está preparado para ejecutarse hasta que se le llama.

Si utiliza un trabajo de prearranque, no tiene que esperar a que el programa que llama pase por el proceso de iniciación de trabajo. La iniciación de trabajo se realiza antes de que un programa pueda realmente arrancar. Como la iniciación del trabajo ya se ha realizado, un trabajo de prearranque permite al programa arrancar más rápidamente una vez recibida la petición de arranque del programa.

Un programa ILE COBOL/400 puede acceder al área de datos PIP para su trabajo con la instrucción ACCEPT, utilizando un nombre nemotécnico asociado con el nombre de función en PIP-DATA.

El área de datos PIP es un elemento alfanumérico de 2.000 bytes y contiene parámetros recibidos desde un programa de llamada. Proporciona los parámetros de inicialización del programa que, en trabajos de no prearranque, se proporcionan a través de parámetros COBOL estándar.

Utilice una instrucción ACCEPT de Formato 5 para acceder al área de datos PIP, similar a la forma de utilizar una instrucción ACCEPT de Formato 4 para leer el área de datos local. Tenga en cuenta que no puede actualizar el área de datos PIP utilizando ILE COBOL/400. Vea la *ILE COBOL/400 Reference* para obtener información sobre sintaxis detallada.

Para obtener más información relacionada con los trabajos de prearranque y el área de datos PIP, consulte el manual *Gestión de Trabajos* y el *CL Programación*.

Efecto de EXIT PROGRAM, STOP RUN, GOBACK y CANCEL en archivos internos

Las instrucciones siguientes afectan el estado de un archivo de forma distinta:

- Una instrucción EXIT PROGRAM no cambia el estado de ninguno de los archivos de una unidad de ejecución a menos que:
  - El programa ILE COBOL/400 que emite EXIT PROGRAM tenga el atributo INITIAL. Si lo tiene, se cerrarán todos los archivos internos definidos en ese programa.
  - Haya emitido una instrucción EXIT PROGRAM con la expresión CONTINUE en el programa principal de un grupo de activación *NEW. En este caso, se devuelve el control del programa principal al de llamada y ello provoca a su vez la finalización del grupo de activación *NEW y el cierre de todos los archivos del ámbito del grupo de activación.
- Una instrucción STOP RUN, realizada en un límite de control fijo, cierra todos los archivos de una unidad de ejecución COBOL.
- Una instrucción GOBACK emitida desde un programa principal cierra todos los archivos de una unidad de ejecución. Una instrucción GOBACK emitida desde un programa principal actúa de la misma forma que la instrucción STOP RUN. Una instrucción GOBACK emitida desde un subprograma no cambia el estado de ninguno de los archivos de una unidad de ejecución, a menos que el pro-

grama ILE COBOL/400 que emite la instrucción GOBACK tenga el atributo INITIAL. Si lo tiene, éste cerrará todos los archivos internos definidos en ese programa. Una instrucción GOBACK emitida desde un subprograma actúa de la misma forma que la instrucción EXIT PROGRAM.

- Una instrucción CANCEL restablece el almacenamiento que contiene información sobre el archivo interno. Si el programa tiene archivos internos que estén abiertos cuando se procesa la instrucción CANCEL, estos archivos internos se cerrarán cuando se cancele el programa. El programa ya no puede utilizar los archivos a menos que los vuelva a abrir. Si el programa cancelado se llama de nuevo, el programa considerará que el archivo está cerrados. Si el programa abre el archivo, se establecerá un nuevo enlace con el archivo.

Cancelación de un programa ILE COBOL/400

Cuando un subprograma finaliza con EXIT PROGRAM o GOBACK, se queda en el estado en el que se utilizó por última vez, a menos que tenga el atributo INITIAL. Un subprograma que utilice la instrucción EXIT PROGRAM con la expresión AND CONTINUE RUN UNIT también se deja en el estado en que se utilizó por última vez. La próxima vez que se llame en la unidad de ejecución, sus valores internos estarán igual que se dejaron, excepto para las instrucciones PERFORM, que se restablecen.

Para restablecer los valores internos de un subprograma a su estado inicial antes de volver a llamarlo, debe cancelar el subprograma. La cancelación del subprograma asegura que la próxima vez que se llame, éste estará en su estado inicial.

Cancelación desde otro programa ILE COBOL/400

En ILE COBOL/400, utilice la instrucción CANCEL para cancelar un subprograma. El subprograma debe estar en el mismo grupo de activación que el programa que lo está cancelando para que funcione la instrucción CANCEL.

Después de ejecutar una instrucción CANCEL para un subprograma llamado, ese subprograma ya no tiene ninguna conexión lógica con el programa. El contenido de los elementos de datos en registros de datos EXTERNAL y archivos EXTERNAL descritos por el subprograma no cambia cuando se cancela un subprograma. Si posteriormente se ejecuta una instrucción CALL en la unidad de ejecución que hace referencia al mismo subprograma, al entrar al subprograma éste se mostrará en su estado inicial.

Los subprogramas llamados pueden contener instrucciones CANCEL; sin embargo, un subprograma llamado no debe contener una instrucción CANCEL que cancele, directa o indirectamente, su programa de llamada o cualquier otro programa superior en la jerarquía de llamadas. Si un subprograma llamado intenta cancelar el programa de llamada, no se tendrá en cuenta la instrucción CANCEL del subprograma.

Un programa nombrado en una instrucción CANCEL no debe hacer referencia a ningún programa que se haya llamado y que aún no haya devuelto el control al programa de llamada. Un programa puede cancelar los programas a los que se haya llamado pero que ya hayan devuelto el control siempre que estén en el mismo grupo de activación. Por ejemplo:

A llama a B y B llama a C	Cuando A recibe el control, puede cancelar C.
A llama a B y A llama a C	Cuando C recibe el control, puede cancelar B.

Nota: Al cancelar un objeto de programa que contiene programas ILE COBOL/400 múltiples, sólo se cancela el programa ILE COBOL/400 asociado con el PEP del objeto de programa.

Consulte la *ILE COBOL/400 Reference* para obtener una descripción completa de la instrucción CANCEL.

Cancelación desde otro lenguaje

Puede cancelar un programa ILE COBOL/400 principal desde ILE RPG/400, ILE C/400 y ILE CL, llamando a su procedimiento de cancelación mediante una llamada estática de procedimientos. El nombre del procedimiento de cancelación se forma tomando el nombre del programa ILE COBOL/400 más externo y añadiéndole el sufijo `_reset`.

No puede cancelar un programa ILE COBOL/400 desde un programa OPM COBOL/400 ni desde un programa OPM RPG/400.

No utilice el mandato de CL Reclamar recursos (RCLSRC) para cancelar un programa ILE COBOL/400. Si se emite RCLRSC dentro de un grupo de activación ILE, provocará una excepción. Para obtener más información sobre el mandato RCLRSC, consulte el manual *CL Reference*.

Capítulo 10. Llamada de datos y compartimento de datos con otros lenguajes

ILE COBOL/400 puede llamar a otros lenguajes ILE, OPM y EPM o ser llamado por ellos.

En este capítulo se describe:

- cómo llamar y cómo pasar datos a otro lenguaje desde ILE COBOL/400
- cómo se devuelve el control a ILE COBOL/400 desde otro lenguaje
- cómo emitir un mandato CL desde un programa ILE COBOL/400
- cómo incluir sentencias SQL (Lenguaje de Consulta Estructurada) en el programa ILE COBOL/400

En general:

- Si el programa ILE COBOL/400 **llama** a otro lenguaje, utilice una instrucción CALL con la expresión USING que señale a los elementos que constituirán la lista de parámetros. El control se devuelve al programa en la siguiente instrucción después de la instrucción CALL (a menos que el programa al que se llama o cualquier programa al que éste llame termine la unidad de ejecución).
- Si el programa ILE COBOL/400 **es llamado** con parámetros por otro lenguaje, necesitará una expresión USING en la PROCEDURE DIVISION y un apartado Linkage Section en el que se describan los parámetros que se han de recibir. El programa ILE COBOL/400 puede devolver el control al programa que le llama con una instrucción GOBACK o una instrucción EXIT PROGRAM. El programa ILE COBOL/400 puede terminar la unidad de ejecución con una instrucción STOP RUN o GOBACK siempre y cuando el límite de control más próximo sea fijo; la unidad de ejecución no terminará si dicho límite es variable.

  Para obtener una descripción completa de la manera de llamar a un programa ILE COBOL/400 desde otro lenguaje, consulte la guía de programación de dicho lenguaje.

Una cuestión que debe considerar a la hora de pasar o recibir datos con programas no ILE COBOL/400 es la coincidencia de las listas de parámetros. Si el programa ILE COBOL/400 llama a un programa no ILE COBOL/400, ha de tener claro que es lo que espera como entrada y configurar los datos en consecuencia. Si el programa es llamado, debe saber qué es lo que se pasará y configurar el apartado Linkage Section para aceptarlo.

Otra cuestión a tener en cuenta es el tratamiento del registro especial RETURN-CODE . Dicho registro no puede establecerlo un programa no ILE COBOL/400. Cuando el registro especial RETURN-CODE contiene un valor incorrecto después de haberse devuelto el control desde un programa al que se ha llamado, establezca el registro especial RETURN-CODE con un valor que tenga significado antes de que el programa ILE COBOL/400 devuelva el control al programa de llamada. Si no lo hace así, se pasará un código de retorno incorrecto al programa que realiza la llamada.

Llamada a programas y procedimientos ILE C/400 y VisualAge C++ para OS/400

Nota: Todas las referencias realizadas a ILE C/400 es este apartado son pertinentes también para VisualAge C++ para OS/400.

Un programa ILE COBOL/400 puede llamar a programas y procedimientos ILE C/400* utilizando llamadas dinámicas a programas o llamadas estáticas a procedimientos.

Cuando se utiliza una llamada dinámica a programas para llamar a un programa ILE C/400, el programa ILE C/400 debe compilarse y enlazarse como un objeto de programa separado. Cuando se utiliza una llamada estática a procedimientos para llamar a un procedimiento ILE C/400, el procedimiento ILE C/400 debe compilarse primero en un objeto de módulo y enlazarse después al programa ILE COBOL/400 de llamada. En ILE C/400, un procedimiento ILE corresponde a una función ILE C/400. Consulte la publicación *ILE C/400 Programmer's Guide* para obtener una descripción de la compilación y vinculación de programas y procedimientos ILE C/400.

Para llamar a un programa o procedimiento ILE C/400 desde un programa ILE COBOL/400, utilice la instrucción CALL *literal* (donde *literal* es el nombre del programa o procedimiento ILE C/400). Para llamar al programa o procedimiento ILE C/400, escriba la instrucción CALL *literal* de la misma manera que lo haría si llamase a otro subprograma ILE COBOL/400. Consulte el apartado "Utilización de llamadas estáticas de procedimiento y llamadas dinámicas de programa" en la página 186 para obtener información detallada sobre la manera de escribir la instrucción CALL en el programa ILE COBOL/400 para llamar a un programa ILE C/400 con llamadas dinámicas a programas o llamadas estáticas a procedimientos.

También puede llamar a un programa ILE C/400 desde un programa ILE COBOL/400 utilizando la instrucción CALL *identificador*. Consulte el apartado "Utilización de CALL identificador" en la página 189 para obtener más información sobre CALL *identificador*.

Como alternativa, puede utilizar CALL *puntero de procedimiento* para llamada a un programa o procedimiento ILE C/400 desde un programa ILE COBOL/400. Consulte el apartado "Utilización de CALL puntero de procedimiento" en la página 190 para obtener más información sobre CALL *puntero de procedimiento*. Un puntero de procedimiento en ILE COBOL/400 es similar a un puntero de función en ILE C/400. Puede pasar un puntero de procedimiento como argumento en la instrucción CALL desde ILE COBOL/400 a una función ILE C/400 y hacer que la función ILE C/400 defina su parámetro como un puntero de función.

Sólo puede llamar a una función ILE C/400 que devuelva un valor si se ha especificado la expresión RETURNING de la instrucción ILE COBOL/400 CALL.

Dos o más programas ILE C/400 del mismo grupo de activación pueden interactuar con los recursos de ejecución de los demás. Consulte la publicación *ILE C/400 Programmer's Guide* para obtener una descripción de cómo se consigue esto. Así pues, debe asegurarse de que los programas ILE C/400 a los que llama desde el programa ILE COBOL/400 están diseñados para trabajar conjuntamente en el mismo grupo de activación. Dos programas ILE C/400 del mismo grupo de activación pueden compartir cosas tales como `errno`, vectores de señal y agru-

paciones de almacenamiento. Si el programa ILE COBOL/400 necesita llamar a más de un programa ILE C/400 que no están diseñados para compartir la misma ejecución, especifique un nombre diferente para el grupo de activación en el se ejecutará el programa ILE C/400.

ILE C/400 permite la recursividad, pero ILE COBOL/400 no. Si una función ILE C/400 llama a un programa ILE COBOL/400 de forma recursiva, desde el programa ILE COBOL/400 se generará un mensaje de error en ejecución.

Para llamar a una función ILE C/400 desde un programa ILE COBOL/400, es posible que el nombre de la función ILE C/400 a la que se llama deba ser sensible a las mayúsculas y minúsculas, tener una longitud superior a 10 caracteres (hasta 256) y contener ciertos caracteres especiales. En tal caso, utilice una llamada estática a procedimientos y compile el programa ILE COBOL/400 con el valor *NOMONOPRC del parámetro OPTION de los mandatos CRTCBLMOD o CRTBNDCBL.

Cuando se llama a un procedimiento VisualAge C++ for OS/400 desde ILE COBOL/400, las palabras clave extern "COBOL" o extern "C" deben colocarse en el prototipo de función ILE C++ para impedir la mutilación del nombre de función ILE C++. Utilice extern "C" si ILE COBOL/400 pasa argumentos BY VALUE a ILE C++.

Transferencia de datos a un programa o procedimiento ILE C/400

Puede pasar datos a un programa o procedimiento ILE C/400 al que se haya llamado utilizando CALL...BY REFERENCE, CALL...BY VALUE o CALL...BY CONTENT. Consulte el apartado “Transferencia de datos utilizando CALL...BY REFERENCE, BY VALUE o BY CONTENT” en la página 199 para obtener una descripción de la utilización de CALL...BY REFERENCE, CALL...BY VALUE o CALL...BY CONTENT.

Cuando se pasan datos al programa ILE C/400 con CALL...BY REFERENCE, en la lista de argumentos aceptada por el programa ILE C/400 se coloca un puntero que señala al dato.

Cuando se pasan datos al programa ILE C/400 con CALL...BY CONTENT, el valor del elemento de datos se copia en una ubicación temporal y, a continuación, en la lista de argumentos aceptada por el programa ILE C/400 se coloca un puntero que contiene la dirección de la ubicación temporal de la copia.

Para CALL...BY VALUE, el valor del dato se coloca en la lista de argumentos aceptada por el programa ILE C/400. CALL...BY VALUE puede utilizarse para llamar a procedimientos ILE C/400, pero no para llamar a objetos de programa ILE C/400.

En el programa ILE COBOL/400, describa los argumentos que desea pasar al programa o procedimiento ILE C/400 en el apartado Data Division de la misma forma en que se describen otros datos en Data Division. Consulte el apartado “Transferencia y compartimiento de datos entre programas” en la página 198 para obtener una descripción de la manera de describir los argumentos que desee pasar.

Cuando el objeto de programa ILE C/400 llamado empiece a ejecutarse, se llamará de forma automática a la función `main`. Cada objeto de programa ILE C/400 debe tener una función denominada `main`. Al pasar parámetros al objeto de programa

ILE C/400, debe declarar dos parámetros con la función `main`. Aunque puede darse cualquier nombre a dichos parámetros, normalmente se hace referencia a ellos como *argc* y *argv*. El primer parámetro, *argc* (recuento de argumentos), tiene el tipo `int` e indica cuántos argumentos se han facilitado en la instrucción CALL que ha llamado al objeto de programa ILE C/400. El segundo parámetro, *argv* (vector de argumentos), tiene la matriz tipo de puntero que señala a los objetos de matriz `char`.

El valor de *argc* indica el número de punteros de la matriz *argv*. Si hay disponible un nombre de programa, el primer elemento de *argv* señala a la matriz de caracteres que contiene el nombre de invocación del programa ILE C/400 que se está ejecutando.Los elementos restantes de *argv* contienen punteros que señalan a los parámetros que se están pasando al programa ILE C/400 al que se llama. El último elemento, *argv[argc]*, siempre contiene `NULL`.

Consulte la publicación *ILE C/400 Programmer's Guide* para obtener más información sobre la descripción de parámetros en el programa o procedimiento ILE C/400 al que se llama.

Compatibilidad de tipos de datos entre ILE C/400 e ILE COBOL/400

ILE C/400 e ILE COBOL/400 tienen tipos de datos diferentes.Cuando desee pasar datos entre programas escritos en ILE C/400 e ILE COBOL/400, debe tener presentes dichas diferencias. Algunos tipos de datos de ILE C/400 e ILE COBOL/400 carecen de equivalente directo en el otro lenguaje.

La Tabla 12 muestra la compatibilidad de tipos de datos entre ILE COBOL/400 e ILE C/400.

Tabla 12 (Página 1 de 2). Compatibilidad de tipos de datos ILE COBOL/400 con ILE C/400

ILE COBOL/400	Declaración en prototipo de ILE C/400	Longitud	Descripción
PIC X(n).	char[n] o char *	n	Campo de tipo carácter en el que n=1 a 3 000 000.
PIC X(6).	char[6]	6	Campo de fecha.
PIC X(5).	char[5]	5	Campo de día.
PIC X.	char	1	Campo de día de la semana.
PIC X(8).	char[8]	8	Campo de hora.
PIC X(n).	char[n]	26	Campo de indicación de la hora.
PIC G(n)	char[2n]	2n	Campo gráfico.
PIC 1 INDIC ..	char	1	Indicador.
PIC S9(n) DISPLAY	char[n]	n	Decimal con zona.
PIC S9(n-p)V9(p) COMP-3	decimal(n,p)	n/2+1	Decimal empaquetado.
PIC S9(n-p)V9(p) PACKED-DECIMAL.	decimal(n,p)	n/2+1	Decimal empaquetado.
PIC S9(4) COMP-4.	short int	2	Entero con signo de 2 bytes con un rango de -9999 a +9999.

Tabla 12 (Página 2 de 2). Compatibilidad de tipos de datos ILE COBOL/400 con ILE C/400

ILE COBOL/400	Declaración en prototipo de ILE C/400	Longitud	Descripción
PIC S9(4) BINARY.	short int	2	Entero con signo de 2 bytes con un rango de -9999 a +9999.
PIC S9(9) COMP-4.	int	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.
PIC S9(9) BINARY.	int	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.
PIC S9(9) COMP-4.	long int	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.
PIC S9(9) BINARY.	long int	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.
PIC S9(18) COMP-4.	No está soportado.	8	Entero de 8 bytes.
PIC S9(18) BINARY.	No está soportado.	8	Entero de 8 bytes.
05 VL-FIELD. 10 i PIC S9(4) COMP-4. 10 data PIC X(n).	_Packed struct {short i; char[n]}	n+2	Campo de longitud variable en el que i es la longitud prevista y n la longitud máxima.
05 n PIC 9(9) COMP-4. 05 x redefines n PIC X(4).	struct {unsigned int : n}x;	4	Los campos de bit pueden manipularse con literales hexadecimales.
01 record 05 field1... 05 field2...	struct	n	Estructura. Utilice el calificador _Packed en la estructura. Las estructuras deben pasarse como puntero a la estructura si desea cambiar el contenido de la estructura.
USAGE IS POINTER	*	16	Puntero.
PROCEDURE-POINTER	pointer to function	16	Puntero de 16 bytes que señala a un procedimiento.
USAGE IS INDEX	int	4	Entero de 4 bytes.
REDEFINES	union.element	n	Elemento de una unión.
OCCURS	data_type[n]	n*(longitud del tipo de datos)	Matriz.
USAGE IS COMP-1	float	4	Coma flotante de 4 bytes.
USAGE IS COMP-2	double	8	Coma flotante de 8 bytes.
No está soportado.	long double	8	Doble largo de 8 bytes.
No está soportado.	enum	1, 2, 4	Enumeración.

Nota: Para todos los elementos de datos COMP-4 y BINARY, las limitaciones de rango indicadas son pertinentes sólo cuando se especifica el valor *STDTRUNC para el parámetro OPTION de los mandatos CRTCBLMOD o CRTBNDCBL. Si se utiliza *NOSTDTRUNC, no es necesario observar las restricciones de rango.

Compartir datos externos con un programa o procedimiento ILE C/400

Los datos externos pueden compartirse entre un programa ILE COBOL/400 y otro ILE C/400. Para que un dato pueda compartirse, debe estar definido con el mismo nombre y descripción en el programa ILE COBOL/400 y el ILE C/400. Si los datos externos que se han de compartir entre el programa ILE C/400 y el ILE COBOL/400 están definidos con un tamaño diferente en los programas, el tamaño de los datos externos se ajustará por fuerza al definido con la palabra clave `extern` en el programa ILE C/400.

El programa ILE COBOL/400 y el ILE C/400 deben estar vinculados estáticamente entre sí para que los datos externos puedan compartirse.

En el programa ILE COBOL/400, el dato debe estar descrito con la cláusula EXTERNAL en el apartado Working Storage Section. Consulte el apartado “Compartimento de datos EXTERNAL” en la página 203 o el apartado referente a la cláusula EXTERNAL de ILE C/400 para obtener una descripción más amplia de la forma en que se utilizan los datos externos en un programa ILE COBOL/400.

En el programa ILE C/400, el dato debe declararse utilizando la palabra clave `extern`. Consulte la publicación *ILE C/400 Programmer's Guide* para obtener una descripción detallada de la palabra clave `extern`.

Devolución del control desde un programa o procedimiento ILE C/400

La palabra clave `return` en ILE C/400 hace que se devuelva el control al programa que llama. Si la palabra clave ILE C/400 `return` devuelve otra cosa que no sea una anulación (VOID), la instrucción ILE COBOL/400 CALL debe tener una expresión de retorno. Además, el tipo de datos y la longitud del elemento devuelto desde ILE C/400 debe coincidir con el tipo de datos y la longitud del identificador de la sentencia RETURNING de la instrucción de llamada COBOL.

Cuando se emite `return` desde un programa ILE C/400, es posible que provoque la finalización del grupo de activación ILE en el que se está ejecutando el programa ILE C/400 al que se ha llamado. Si se ha definido el programa ILE C/400 para que se ejecute en un grupo de activación *NEW, al emitir `return` cerca de un límite de control fijo finalizará el grupo de activación en el que se estaba ejecutando el programa ILE C/400. Si se ha definido el programa ILE C/400 para que se ejecute en un grupo de activación *CALLER o en un grupo de activación con nombre, al emitir `return` el grupo de activación en el que se estaba ejecutando el programa ILE C/400 permanece activo. Una llamada posterior al programa ILE C/400 en este grupo de activación hallará al programa ILE C/400 en el estado en que se haya utilizado por última vez.

La función `exit(n)` puede hacer que se devuelva el control al límite de control más cercano. Una condición de excepción puede provocar la llamada a un manejador de excepciones o la devolución del control al límite de control más próximo.

Cuando el programa ILE C/400 se ejecuta en un grupo de activación con un nombre distinto del que tiene el grupo de activación del programa ILE COBOL/400 que llama, `exit(n)` o una excepción sin manejar hacen que ocurra lo siguiente. Cuando `exit(n)` o la excepción sin manejar se producen cerca de un límite de control fijo, finaliza el grupo de activación en el que se ejecuta el programa ILE C/400. Si se producen cerca de un límite de control variable, el grupo de activación permanece activo. Si una excepción sin manejar finaliza el grupo de activación en

el que se está ejecutando el programa ILE C/400, se activa el mensaje de escape CEE9901 en el grupo de activación del programa ILE COBOL/400 que llama.

Cuando el programa ILE C/400 y el programa ILE COBOL/400 que llama se ejecutan en el mismo grupo de activación, `exit(n)` o una excepción sin manejar hacen que ocurra lo siguiente. Si `exit(n)` o la excepción sin manejar se producen cerca de un límite de control fijo, finaliza el grupo de activación, incluido el programa ILE COBOL/400. Si una excepción sin manejar finaliza el grupo de activación en el que se está ejecutando el programa ILE C/400 y el programa ILE COBOL/400, el programa anterior al límite de control fijo recibe el mensaje de escape CEE9901. Si `exit(n)` o la excepción sin manejar se producen cerca de un límite de control variable, finalizan todos los programas y procedimientos, incluido el programa ILE COBOL/400, existentes entre el programa ILE C/400 desde el que se haya realizado `exit(n)` y el programa del límite de control variable.

Se devuelve el control al programa ILE COBOL/400 en la instrucción siguiente después de la instrucción CALL si el programa al que se ha llamado finaliza sin ninguna excepción.Si dicho programa finaliza con una excepción, puede que se llame a un procedimiento de manejo de excepciones identificado en el programa ILE COBOL/400. Consulte el Capítulo 12, “Manejo de errores y excepciones ILE COBOL/400” en la página 261 para obtener una descripción completa de la transferencia del control a un procedimiento de manejo de excepciones.

El programa al que se llama también puede enviar un mensaje de escape después del programa ILE COBOL/400 que llama saltándoselo. En tal caso, se cancela la invocación del programa ILE COBOL/400. El resultado de cancelar la invocación es similar a si volviera del programa ILE COBOL/400.

Llamada a programas y procedimientos ILE RPG/400

Un programa ILE COBOL/400 puede llamar a programas y procedimientos ILE RPG/400* utilizando llamadas dinámicas a programa o llamadas estáticas a procedimientos.

Cuando se utiliza una llamada dinámica a programa para llamar a un programa ILE RPG/400, el programa ILE RPG/400 debe compilarse y vincularse como un objeto de programa separado.Cuando se utiliza una llamada estática a procedimiento para llamar a un procedimiento ILE RPG/400, el procedimiento ILE RPG/400 debe compilarse primero en un objeto de módulo y vincularse después al programa ILE COBOL/400 que llama. Consulte la publicación *ILE RPG/400 Guía del programador* para obtener una descripción de la compilación y vinculación de los programas y procedimientos ILE RPG/400.

Para llamar a un programa o procedimiento ILE RPG/400 desde un programa ILE COBOL/400, utilice la instrucción CALL *literal* (donde *literal* es el nombre del programa o procedimiento ILE RPG/400). Para llamar al programa o procedimiento ILE RPG/400, escriba la instrucción CALL *literal* de la misma manera que lo haría si llamase a otro subprograma ILE COBOL/400. Consulte el apartado “Utilización de llamadas estáticas de procedimiento y llamadas dinámicas de programa” en la página 186 para obtener información detallada sobre la manera de escribir la instrucción CALL en el programa ILE COBOL/400 para llamar a un programa ILE RPG/400 con llamadas dinámicas a programas o llamadas estáticas a procedimientos.

También puede llamar a un programa ILE RPG/400 desde un programa ILE COBOL/400 utilizando la instrucción CALL *identificador*. Consulte el apartado “Utilización de CALL identificador” en la página 189 para obtener más información sobre CALL *identificador*.

Transferencia de datos a un programa o procedimiento ILE RPG/400

Puede pasar datos a un programa o procedimiento ILE RPG/400 al que se haya llamado utilizando CALL...BY REFERENCE, CALL...BY VALUE o CALL...BY CONTENT. Consulte el apartado “Transferencia de datos utilizando CALL...BY REFERENCE, BY VALUE o BY CONTENT” en la página 199 para obtener una descripción de la utilización de CALL...BY REFERENCE, CALL...BY VALUE o CALL...BY CONTENT.

Cuando se pasan datos al programa ILE RPG/400 con CALL...BY REFERENCE, en la lista de argumentos aceptada por el programa ILE RPG/400 se coloca un puntero que señala al elemento de datos.
Cuando se pasan datos al programa ILE RPG/400 con CALL...BY CONTENT, el valor del dato se copia en una ubicación temporal y, a continuación, en la lista de argumentos aceptada por el programa ILE RPG/400 se coloca un puntero que contiene la dirección de la ubicación temporal de la copia. En CALL...BY VALUE, el valor del elemento se coloca en la lista de argumentos aceptada por el programa ILE RPG/400. CALL...BY VALUE puede utilizarse para llamar a procedimientos ILE RPG/400, pero no para llamar a objetos de programa ILE RPG/400.

En el programa ILE COBOL/400, describa los argumentos que desea pasar al programa o procedimiento ILE RPG/400 en el apartado Data Division de la misma forma en que se describen otros datos en Data Division. Consulte el apartado “Transferencia y compartimiento de datos entre programas” en la página 198 para obtener una descripción de la manera de describir los argumentos que desee pasar.

En el programa ILE RPG/400 al que se llama, describa los parámetros que desea recibir del programa ILE COBOL/400 utilizando la operación `PARM`. Cada parámetro a recibir se define en una operación `PARM` por separado. Especifique el nombre del parámetro en el campo Result. Las entradas Factor 1 y Factor 2 son opcionales e indican variables o literales. El valor del campo Factor 1 se transfiere de la entrada de campo Result cuando se produce la llamada. El valor del campo Factor 2 se coloca en la entrada de campo Result al volver.

Otro método de definir parámetros en un programa ILE RPG/400 es especificarlos en un prototipo. Cada parámetro está definido en una especificación de definición por separado. Para los parámetros pasados utilizando BY REFERENCE, no es necesaria ninguna palabra clave especial. Para los parámetros pasados utilizando BY VALUE se utiliza la palabra clave `VALUE`. Consulte la publicación *ILE RPG/400 Guía del programador* para obtener más información sobre la forma de describir los argumentos en un programa ILE RPG/400.

Compatibilidad de los tipos de datos entre ILE RPG/400 e ILE COBOL/400

ILE RPG/400 e ILE COBOL/400 tienen tipos de datos diferentes.Cuando desee pasar datos entre programas escritos en ILE RPG/400 e ILE COBOL/400, debe tener presentes dichas diferencias. Algunos tipos de datos de ILE RPG/400 e ILE COBOL/400 carecen de equivalente directo en el otro lenguaje.

La Tabla 13 en la página 225 muestra la compatibilidad de los tipos de datos de ILE COBOL/400 con ILE RPG/400.

Tabla 13 (Página 1 de 3). Compatibilidad de tipos de datos ILE COBOL/400 con ILE RPG/400

ILE COBOL/400	**Especificaciones I, D o C ILE RPG/400**	**Longitud**	**Descripción**
PIC X(n).	Blanco o A en la columna de tipo de datos, n en la columna de longitud y blanco en la columna de posición decimal	n	Campo de tipo carácter en el que n=1 a 32 766 .
PIC 1 INDIC ..	*INxxxx	1	Indicador.
PIC S9(n) DISPLAY	S en la columna de tipo de datos o blanco en la columna de tipo de datos, n en la columna de longitud y 0 en la columna de posición decimal	n	Decimal con zona.
PIC S9(n-p)V9(p) COMP-3	P en la columna de tipo de datos, n en la columna de longitud y p en la columna de posición decimal	n/2 + 1	Decimal empaquetado.
PIC S9(n-p)V9(p) PACKED-DECIMAL.	P en la columna de tipo de datos, n en la columna de longitud y p en la columna de posición decimal	n/2 + 1	Decimal empaquetado.
PIC S9(4) COMP-4.	B en la columna de tipo de datos, 4 en la columna de longitud y 0 en la columna de posición decimal	2	Entero con signo de 2 bytes con un rango de -9999 a +9999.
PIC S9(4) BINARY.	B en la columna de tipo de datos, 4 en la columna de longitud y 0 en la columna de posición decimal	2	Entero con signo de 2 bytes con un rango de -9999 a +9999.
PIC S9(4) BINARY. Opción *NOSTDTRUNC	I en la columna de tipo de datos, 5 en la columna de longitud y 0 en la columna de posición decimal	2	Entero con signo de 2 bytes con un rango de -32768 a 32767
No está soportado	U en la columna de tipo de datos, 5 en la columna de longitud y 0 en la columna de posición decimal	2	Entero con signo de 2 bytes con un rango de 0 a 65535
PIC S9(9) COMP-4.	B en la columna de tipo de datos, 9 en la columna de longitud y 0 en la columna de posición decimal	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.
PIC S9(9) BINARY.	B en la columna de tipo de datos, 9 en la columna de longitud y 0 en la columna de posición decimal	4	Entero con signo de 4 bytes con un rango de -999999999 a +999999999.

Tabla 13 (Página 2 de 3). Compatibilidad de tipos de datos ILE COBOL/400 con ILE RPG/400

ILE COBOL/400	Especificaciones I, D o C ILE RPG/400	Longitud	Descripción
PIC S9(9) BINARY. Opción *NOSTDTRUNC	I en la columna de tipo de datos, 10 en la columna de longitud y 0 en la columna de posición decimal	4	Entero con signo de 4 bytes con un rango de -2147483648 a 2147483647
No está soportado	U en la columna de tipo de datos, 10 en la columna de longitud y 0 en la columna de posición decimal	4	Entero sin signo de 4 bytes con un rango de 0 a 4294967295
PIC S9(18) COMP-4.	No está soportado	8	Entero de 8 bytes.
PIC S9(18) BINARY.	No está soportado	8	Entero de 8 bytes.
USAGE IS COMP-1	F en la columna de tipo de datos, 4 en la columna de longitud	4	Campo de coma flotante interno de 4 bytes.
USAGE IS COMP-2	F en la columna de tipo de datos, 8 en la columna de longitud.	8	Campo de coma flotante interno de 8 bytes.
05 VL-FIELD. 10 i PIC S9(4) COMP-4. 10 data PIC X(n).	No está soportado	n+2	Campo de longitud variable en el que i es la longitud prevista y n la longitud máxima.
05 n PIC 9(9) COMP-4. 05 x redefines n PIC X(4).	No está soportado	4	Los campos de bit pueden manipularse con literales hexadecimales.
01 record 05 field1... 05 field2...	estructura de datos	n	Estructura. Las estructuras deben pasarse como puntero a la estructura si desea cambiar el contenido de la estructura.
USAGE IS POINTER	* en la columna de tipo de datos	16	Puntero.
PROCEDURE-POINTER	* en la columna de tipo de datos y la palabra clave PROCPTR	16	Puntero de 16 bytes que señala a un procedimiento.
USAGE IS INDEX	I en la columna de tipo de datos, la longitud es 10, 0 en la columna de posición decimal	4	Entero de 4 bytes.
REDEFINES	subcampo de estructura de datos	n	Elemento de una unión.
OCCURS	Palabra clave OCCURS o palabra clave DIM	n*(longitud del tipo de datos)	Matriz.
PIC X(n)	D en la columna de tipo de datos	n	Tipo de datos de fecha.
PIC X(n)	T en la columna de tipo de datos	n	Tipo de datos de hora.
PIC X(n)	Z en la columna de tipo de datos	n	Tipo de datos de indicación de la hora.

Tabla 13 (Página 3 de 3). Compatibilidad de tipos de datos ILE COBOL/400 con ILE RPG/400			
ILE COBOL/400	**Especificaciones I, D o C ILE RPG/400**	**Longitud**	**Descripción**
PIC G(n)	G en la columna de tipo de datos	n*2	Tipo de datos gráficos (datos de doble byte).

Devolución del control desde un programa o procedimiento ILE RPG/400

Al volver de un procedimiento ILE RPG/400 principal, el código de operación RETURN hace que el control vuelva al programa que llama. Si antes de ejecutar el código de operación RETURN, se utiliza el código de operación SETON para establecer el indicador LR, el programa ILE RPG/400 al que se llama se restablece a su estado inicial al volver al programa que llama. De lo contrario, el programa ILE RPG/400 al que se llama queda en el estado utilizado por última vez.

Al volver de un subprocedimiento ILE RPG/400, el código de operación RETURN hace que se devuelva el control al programa que llama. Si el procedimiento devuelve un valor, el valor devuelto se especifica en el factor 2 ampliado de la operación RETURN. Si el subprocedimiento devuelve un valor, la instrucción CALL de COBOL debe tener una expresión RETURNING.

Nota: El indicador LR carece de significado al volver de un subprocedimiento

Se devuelve el control al programa ILE COBOL/400 en la instrucción siguiente después de la instrucción CALL si el programa al que se ha llamado finaliza sin ninguna excepción.Si dicho programa finaliza con una excepción, se devuelve el control al procedimiento de manejo de excepciones identificado en el programa ILE COBOL/400. Consulte el Capítulo 12, “Manejo de errores y excepciones ILE COBOL/400” en la página 261 para obtener una descripción completa de la transferencia del control a un procedimiento de manejo de excepciones.

El programa al que se llama también puede enviar un mensaje de escape después del programa ILE COBOL/400 que llama saltándoselo. En tal caso, se cancela la invocación del programa ILE COBOL/400. El resultado de cancelar la invocación es similar a si volviera del programa ILE COBOL/400.

Llamada a programas y procedimientos ILE CL

Un programa ILE COBOL/400 puede llamar a programas y procedimientos ILE CL* utilizando llamadas dinámicas a programa o llamadas estáticas a procedimientos.

Cuando se utiliza una llamada dinámica a programa para llamar a un programa ILE CL, el programa ILE CL debe compilarse y enlazarse como un objeto de programa por separado.Cuando se utiliza una llamada estática a procedimiento para llamar a un procedimiento ILE CL, el procedimiento ILE CL debe compilarse primero en un objeto de módulo y enlazarse después al programa ILE COBOL/400 que llama. Consulte la publicación *CL Programación* para obtener una descripción de la compilación y vinculación de los programas y procedimientos ILE CL.

Para llamar a un programa o procedimiento ILE CL desde un programa ILE COBOL/400, utilice la instrucción CALL *literal* (donde *literal* es el nombre del pro-

grama o procedimiento ILE CL). Para llamar al programa o procedimiento ILE CL, escriba la instrucción CALL *literal* de la misma manera que lo haría si llamase a otro subprograma ILE COBOL/400. Consulte el apartado “Utilización de llamadas estáticas de procedimiento y llamadas dinámicas de programa” en la página 186 para obtener información detallada sobre la manera de escribir la instrucción CALL en el programa ILE COBOL/400 para llamar a un programa ILE CL con llamadas dinámicas a programa o llamadas estáticas a procedimiento.

También puede llamar a un programa ILE CL desde un programa ILE COBOL/400 utilizando la instrucción *identificador*. Consulte el apartado “Utilización de CALL identificador” en la página 189 para obtener más información sobre CALL *identificador*.

Transferencia de datos a un programa o procedimiento ILE CL

Puede pasar datos a un programa o procedimiento ILE CL al que se haya llamado utilizando CALL...BY REFERENCE o CALL...BY VALUE o CALL...BY CONTENT. Consulte el apartado “Transferencia de datos utilizando CALL...BY REFERENCE, BY VALUE o BY CONTENT” en la página 199 para obtener una descripción de la utilización de CALL...BY REFERENCE o CALL...BY VALUE o CALL...BY CONTENT.

Cuando se pasan datos al programa ILE CL con CALL...BY REFERENCE, en la lista de argumentos aceptada por el programa ILE CL se coloca un puntero que señala al dato.Cuando se pasan datos al programa ILE CL con CALL...BY CONTENT, el valor del dato se copia en una ubicación temporal y, a continuación, en la lista de argumentos aceptada por el programa ILE CL se coloca un puntero que contiene la dirección de la ubicación temporal de la copia.

En el programa ILE COBOL/400, describa los argumentos que desea pasar al programa o procedimiento ILE CL en el apartado Data Division de la misma forma en que se describen otros datos en Data Division. Consulte el apartado “Transferencia y compartimiento de datos entre programas” en la página 198 para obtener una descripción de la manera de describir los argumentos que desee pasar.

En el programa ILE CL al que se llama, describa los parámetros que desea recibir del programa ILE COBOL/400 en el parámetro PARM de la instrucción PGM. El orden en que figuran en el parámetro PARM los parámetros a recibir debe ser el mismo que el orden en el que figuran en la instrucción CALL del programa ILE COBOL/400. Además de a la posición de los parámetros, debe prestar especial atención a la longitud y tipo de los mismos.
Los parámetros que figuran en el programa ILE CL al que se llama deben declararse como de la misma longitud y tipo que tienen en el programa ILE COBOL/400 que llama.

Para describir en el programa ILE CL al que se llama los parámetros a recibir, utilice instrucciones DCL.
El orden de las instrucciones DCL no es importante.
Sólo el orden en el que se especifican los parámetros en la instrucción PGM determina qué variables se reciben.
El ejemplo siguiente muestra cómo se describen los parámetros en el programa ILE CL al que se llama.

```
PGM PARM(&P1 &P2)
DCL VAR(&P1) TYPE(*CHAR) LEN(32)
DCL VAR(&P2) TYPE(*DEC) LEN(15 5)
 .
 .
 .
RETURN
ENDPGM
```

Consulte la publicación *CL Programación* para obtener una descripción de la manera de describir parámetros en un programa ILE CL.

Compatibilidad de los tipos de datos entre ILE CL e ILE COBOL/400

ILE CL e ILE COBOL/400 tienen tipos de datos diferentes.Cuando desee pasar datos entre programas escritos en ILE CL e ILE COBOL/400, debe tener presentes dichas diferencias. Algunos tipos de datos de ILE CL e ILE COBOL/400 carecen de equivalente directo en el otro lenguaje.

La Tabla 14 muestra la compatibilidad de los tipos de datos de ILE COBOL/400 con ILE CL.

Tabla 14. Compatibilidad de tipos de datos ILE COBOL/400 con ILE CL

ILE COBOL/400	ILE CL	Longitud	Descripción
PIC X(n).	TYPE(*CHAR) LEN(n)	n	Campo de tipo carácter en el que n=1 a 32 766.
01 flag PIC 1. 88 flag-on VALUE B'1'. 88 flag-off VALUE B'0'.	TYPE(*LGL)	1	Contiene '1' ó '0'.
PIC S9(n-p)V9(p) COMP-3.	TYPE(*DEC) LEN(n p)	n/2+1	Decimal empaquetado. Valor máximo de n=15. Valor máximo de p=9.
PIC S9(n-p)V9(p) PACKED-DECIMAL.	TYPE(*DEC) LEN(n p)	n/2+1	Decimal empaquetado. Valor máximo de n=15. Valor máximo de p=9.
USAGE IS COMP-1	No está soportado.	4	Coma flotante interno de 4 bytes.
USAGE IS COMP-2	No está soportado.	8	Coma flotante interno de 8 bytes.

Devolución del control desde un programa o procedimiento ILE CL

El mandato RETURN en ILE CL hace que se devuelva e control al programa que llama.

Se devuelve el control al programa ILE COBOL/400 en la instrucción siguiente después de la instrucción CALL si el programa al que se ha llamado finaliza sin ninguna excepción.Si dicho programa finaliza con una excepción, se devuelve el control al procedimiento de manejo de excepciones identificado en el programa ILE COBOL/400. Consulte el Capítulo 12, “Manejo de errores y excepciones ILE COBOL/400” en la página 261 para obtener una descripción completa de la transferencia del control a un procedimiento de manejo de excepciones.

El programa al que se llama también puede enviar un mensaje de escape después del programa ILE COBOL/400 que llama saltándoselo. En tal caso, se cancela la invocación del programa ILE COBOL/400. El resultado de cancelar la invocación es similar a si volviera del programa ILE COBOL/400.

Llamada a lenguajes OPM

Los programas escritos en lenguajes OPM, como por ejemplo OPM COBOL/400 o OPM RPG/400, sólo pueden llamarse desde ILE COBOL/400 con llamadas dinámicas a programa. Los programas OPM no pueden enlazarse de forma estática a los programas ILE COBOL/400. Si intenta llamar a un programa OPM con una llamada estática a procedimientos, recibirá un mensaje de error. En el momento de la vinculación, recibirá un mensaje de aviso del enlazador por una referencia no resuelta a la llamada estática a procedimientos. Si no hace caso del mensaje de aviso y crea el objeto de programa ILE, obtendrá una excepción cuando intente realizar la llamada estática a procedimientos en la ejecución.

Para llamar a un programa OPM desde un programa ILE COBOL/400, utilice la instrucción CALL *literal* (donde *literal* es el nombre del programa OPM). Para llamar al programa OPM, escriba la instrucción CALL *literal* de la misma manera que lo haría si llamase a otro subprograma ILE COBOL/400 con una llamada dinámica a programa. Consulte el apartado “Realización de llamadas dinámicas de programa utilizando CALL literal” en la página 188 para obtener información detallada sobre la manera de escribir la instrucción CALL en el programa ILE COBOL/400 para llamar a un programa OPM con llamadas dinámicas a programas.

También puede llamar a un programa OPM desde un programa ILE COBOL/400 utilizando la instrucción CALL *identificador*. Consulte el apartado “Utilización de CALL identificador” en la página 189 para obtener más información sobre CALL *identificador*.

Los programas escritos en lenguajes OPM sólo se pueden ejecutar en el Grupo de activación por omisión (*DFTACTGRP).

Puede llamar a un programa ILE COBOL/400 desde un programa OPM con la misma semántica que utilizaría para llamar a otro programa OPM.

Los datos externos no pueden compartirse entre programas OPM y programas ILE COBOL/400.

Llamada a programas OPM COBOL/400

Los programas OPM COBOL/400 sólo pueden ejecutarse en el Grupo de activación por omisión (*DFTACTGRP). Los programas ILE COBOL/400 pueden ejecutarse en el Grupo de activación por omisión (*DFTACTGRP), en grupos de activación ILE *NEW y en grupos de activación ILE con nombre.

Nota: CRTPGM no permite que se especifique *DFTACTGRP de forma explícita en el parámetro ACTGRP, pero sí permite que se especifique *CALLER en el parámetro ACTGRP. La especificación de *CALLER en el parámetro ACTGRP permite la ejecución en el Grupo de activación por omisión de un programa ILE COBOL/400 llamado desde un programa OPM COBOL/400 (o desde cualquier programa OPM). Esta es la única forma en que puede ejecutarse un programa ILE COBOL/400 en el grupo de activación por

omisión. Un programa ILE COBOL/400 no puede ser el límite de control fijo del grupo de activación por omisión.

Cuando se ejecuta una aplicación mixta con programas en OPM COBOL/400 y ILE COBOL/400, debe respetarse el siguiente marco para poder reproducir con la mayor fidelidad posible una unidad de ejecución OPM COBOL/400:

- Todos los programas participantes deben ejecutarse en el grupo de activación por omisión (*DFTACTGRP).
- El primer programa COBOL a activar en el grupo de activación debe ser un programa OPM COBOL/400.
- Para finalizar la unidad de ejecución, un programa OPM COBOL/400 debe emitir STOP RUN o un programa principal OPM COBOL/400 debe emitir GOBACK.
- Si se produjera una excepción que provocase un STOP RUN implícito, debe manejarse de tal manera que el STOP RUN implícito lo desencadene un OPM COBOL/400.

En el caso de un aplicación mixta con programas en OPM COBOL/400 e ILE COBOL/400 en ejecución en el grupo de activación por omisión, cada programa ILE COBOL/400 está considerado como programa no COBOL por parte de los programas OPM COBOL/400 y cada programa OPM COBOL/400 está considerado como programa no COBOL por parte de los programas ILE COBOL/400. Además, cada programa ILE COBOL/400 llamado por un programa OPM COBOL/400 genera un límite de control variable por el que se vincula el ámbito del STOP RUN emitido por ILE COBOL/400. Cuando STOP RUN lo emite el programa ILE COBOL/400, se devuelve el control al programa OPM COBOL/400 sin renovar el estado del programa ILE COBOL/400 y la unidad de ejecución OPM COBOL/400 no finaliza. Cuando STOP RUN se emite desde un programa OPM COBOL/400, se devuelve el control al emisor de la llamada del programa OPM COBOL/400 principal actual y finaliza la unidad de ejecución OPM COBOL/400.

En el caso de una aplicación mixta con programas en OPM COBOL/400 y ILE COBOL/400 en la que un programa ILE COBOL/400 se ejecuta en un grupo de activación ILE *NEW o con nombre y el programa OPM COBOL/400 se ejecute en el grupo de activación por omisión, el efecto del STOP RUN emitido por el programa ILE COBOL/400 depende de si el límite de control más cercano es fijo o variable.Si es fijo, se devuelve el control al emisor de la llamada del límite de control fijo y su grupo de activación *NEW o ILE con nombre finaliza. Si se trata de un límite de control variable, se devuelve el control al emisor de la llamada del límite de control variable, pero el grupo de activación ILE *NEW o con nombre del programa ILE COBOL/400 permanece activo.

Nota: Este marco no se ajusta a una unidad de ejecución OPM COBOL/400.

Llamada a lenguajes EPM

Los programas escritos en lenguajes EPM, como por ejemplo EPM C/400, Pascal y FORTRAN, pueden ser llamados desde un programa ILE COBOL/400 mediante una llamada CALL a `QPXXCALL`.

En el ejemplo siguiente, un programa ILE COBOL/400 utiliza `QPXXCALL` para llamar a un procedimiento Pascal.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/COBTOPAS         AS400SYS   96/07/04 10:36:45         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID. COBTOPAS.
    3     000030 ENVIRONMENT DIVISION.
    4     000040 CONFIGURATION SECTION.
    5     000050   SOURCE-COMPUTER. IBM-AS400.
    6     000060   OBJECT-COMPUTER. IBM-AS400.
    7     000070 DATA DIVISION.
    8     000080 WORKING-STORAGE SECTION.
    9     000090 01  PARAMETER-LIST.
   10     000100     05  ENTRY-NAME   PIC X(100)         VALUE "SQUARE".
   11     000110     05  ENTRY-ID     PIC X(10)          VALUE "*MAIN".
   12     000120     05  PROG-NAME    PIC X(20)          VALUE "MATH".
   13     000130     05  A-REAL       PIC S9(9)  COMP-4  VALUE 0.
   14     000140     05  CLEAN        PIC S9(9)  COMP-4  VALUE 0.
   15     000150     05  INPT         PIC S99            VALUE 0.
   16     000160 PROCEDURE DIVISION.
          000170 MAINLINE.
   17     000180     DISPLAY "ENTRE EL NÚMERO DE ÁREA:".
   18     000190     ACCEPT INPT.
   19     000200     MOVE INPT TO A-REAL.
   20     000210     CALL "QPXXCALL" USING ENTRY-NAME
          000220                           ENTRY-ID
          000230                           PROG-NAME
          000240                           A-REAL.
   21     000250     DISPLAY A-REAL.
   22     000260     CALL "QPXXDLTE" USING CLEAN.
   23     000270     STOP RUN.

                   * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 57. Llamada a un procedimiento Pascal desde un programa ILE COBOL/400.

```
segment MATH;
procedure SQUARE ( var X : integer ) ; external ;
procedure SQUARE;
   begin
    X := X * X
end; .
```

Figura 58. Punto de entrada Pascal al que ha de llamarse desde un programa ILE COBOL/400.

Pascal permite que un programa ILE COBOL/400 llame a un procedimiento Pascal como subprograma. Para hacerlo, especifique el procedimiento Pascal con la directriz `EXTERNAL` (consulte la Figura 58). En el ejemplo anterior, la primera invocación del procedimiento con `*MAIN` para el parámetro `ENTRY-ID` de `QPXXCALL` establecerá un Entorno principal Pascal. Puede utilizar `QPXXDLTE` para borrar los entornos principal y reentrante Pascal.Pasar cero a `QPXXDLTE` en el parámetro `CLEAN` hace que se borre el entorno principal Pascal actual.

Puede llamar a un programa ILE COBOL/400 desde un programa EPM con la misma semática que utilizaría para llamar a otro programa EPM.

Los datos externos no pueden compartirse entre programas EPM y programas ILE COBOL/400.

Emitir un mandato CL desde un programa ILE COBOL/400

Puede emitir un mandato CL desde un programa ILE COBOL/400 por medio de una llamada dinámica a programa a QCMDEXC.

En el ejemplo siguiente, la llamada CALL a QCMDEXC (en el número de secuencia 000160) da como resultado el proceso del mandato CL Añadir Entrada de Lista de Bibliotecas (ADDLIBLE) en el número de secuencia 000110. La realización satisfactoria del mandato CL da como resultado la adición de la biblioteca, COBOLTEST, a la lista de bibliotecas.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CMDXMPLE         AS400SYS   96/07/04 09:54:12         Página     2

                                   F u e n t e

 INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID. CMDXMPLE.
    3     000030 ENVIRONMENT DIVISION.
    4     000040 CONFIGURATION SECTION.
    5     000050   SOURCE-COMPUTER. IBM-AS400.
    6     000060   OBJECT-COMPUTER. IBM-AS400.
    7     000070 DATA DIVISION.
    8     000080 WORKING-STORAGE SECTION.
    9     000090 01  PROGRAM-VARIABLES.
   10     000100     05  CL-CMD       PIC X(33)
          000110                       VALUE "ADDLIBLE COBOLTEST".
   11     000120     05  PACK-VAL     PIC 9(10)V9(5) COMP-3
          000130                       VALUE 18.
   12     000140 PROCEDURE DIVISION.
          000150 MAINLINE.
   13     000160     CALL "QCMDEXC" USING CL-CMD PACK-VAL.
   14     000170     STOP RUN.

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

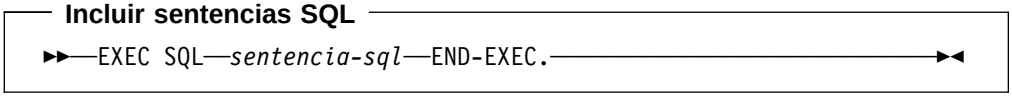
Figura 59. Emisión de un mandato CL desde un programa ILE COBOL/400.

Para obtener más información sobre QCMDEXC, consulte la publicación *CL Programación*.

Inclusión de sentencias SQL (Lenguaje de Consulta Estructurada) en el programa ILE COBOL/400

La sintaxis de las sentencias SQL incluidas en un programa fuente ILE COBOL/400 es:

Incluir sentencias SQL



Si el tipo de miembro del programa fuente es SQLCBLLE, cuando el corrector sintáctico de COBOL encuentre una sentencia SQL, ésta se pasará al corrector sintáctico de SQL. Si se detecta un error, se devuelve un mensaje.

Si se encuentra una instrucción SQL y el tipo de miembro no es SQLCBLLE, se devuelve un mensaje que indica que una instrucción ILE COBOL/400 tiene un error.

Si hay errores en la sentencia SQL incluida además de errores en las instrucciones ILE COBOL/400 precedentes, el mensaje de error de SQL se visualizará después de que se hayan corregido los errores COBOL precedentes.

Para obtener más información sobre sentencias SQL, consulte la publicación *DB2/400 Manual de Consulta SQL*.

Puede crear programas SQL para su utilización con programas ILE COBOL/400. Un cursor SQL utilizado en el programa ILE COBOL/400 puede tener por ámbito el objeto de módulo o el grupo de activación. Para especificar el ámbito del cursor SQL, utilice el parámetro CLOSQLCSR de los mandatos Crear programa SQL (CRTSQLxxx). Para obtener más información sobre los cursores SQL, consulte la publicación *DB/2 for OS/400 SQL Programming*.

Llamada a un API ILE para recuperar el siglo actual

El ejemplo siguiente Figura 60 muestra cómo recuperar un año de cuatro dígitos utilizando la API enlazable ILE, Obtener hora local actual (CEELOCT). Esta API recupera la hora local actual con tres formatos.El tercer formato es el gregoriano, cuyos primeros cuatro caracteres son el año.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. DATE1.
      * Programa de ejemplo para obtener el año de 4 dígitos en ILE COBOL/400
       ENVIRONMENT DIVISION.
         CONFIGURATION SECTION.
        SOURCE-COMPUTER. IBM-AS400.
        OBJECT-COMPUTER. IBM-AS400.
      DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 date-vars.
          05 lilian            pic 9(4) usage binary.
          05 lilian-time-stamp pic x(8).
          05 gregorian-date.
             10 greg-year      pic x(4).
             10 greg-month     pic x(2).
             10 greg-day       pic x(2).
             10 greg-time      pic x(9).
             10 filler         pic x(6).
PROCEDURE DIVISION.
      TEST-PARA.
          call procedure "CEELOCT" using
              lilian lilian-time-stamp
              gregorian-date.
          display "la fecha es " gregorian-date.
          display "año " greg-year.
          display "mes " greg-month.
              STOP RUN.
```

Figura 60. Ejemplo de recuperación del siglo actual.

Capítulo 11. Utilización de punteros en un programa ILE COBOL/400

Puede utilizar un **puntero** (un elemento de datos en el cual pueden almacenarse valores de direcciones) dentro de un programa ILE COBOL/400 cuando desee transferir y recibir direcciones de elementos de datos, procedimientos ILE u objetos programa.

Este capítulo describe:

- cómo definir y redefinir punteros
- cómo inicializar punteros
- cómo leer y grabar punteros
- cómo manipular datos utilizando punteros.

Definición de punteros

Puede definir los punteros de dos formas:

- Un puntero para un elemento de datos. Este puntero se define con la cláusula USAGE POINTER. El elemento de datos resultante se denomina un **elemento de datos de puntero**.
- Un puntero para un programa ILE COBOL/400, un procedimiento ILE o un objeto de programa.Este puntero se define con la cláusula USAGE PROCEDURE-POINTER. El elemento de datos resultante se denomina un **elemento de datos de puntero de procedimientos**.

En el sistema AS/400, los punteros tienen 16 bytes de longitud.

Los elementos de datos de puntero ILE COBOL/400 son **punteros de espacio** del AS/400, ya que señalan objetos de espacio del sistema. Una parte del puntero describe sus atributos, como a qué objeto de espacio del AS/400 está señalando. Otra parte del puntero contiene el desplazamiento del objeto de espacio en el sistema AS/400.

Los elementos de datos del puntero de procedimientos ILE COBOL/400 son **punteros abiertos** del AS/400. Los punteros abiertos tienen la posibilidad de utilizarse como otros tipos de punteros del AS/400. En concreto, cuando un elemento de datos de puntero de procedimientos ILE COBOL/400 se coloca en un objeto de programa, el puntero abierto contendrá un puntero del sistema AS/400. Cuando un elemento de datos de puntero de procedimientos ILE COBOL/400 se coloca en un procedimiento ILE, el puntero abierto contendrá un puntero de procedimientos del AS/400.

Un puntero elemental de un elemento de 16 bytes puede compararse por igualdad o utilizarse para establecer el valor de otros elementos de puntero.

Un elemento de datos de puntero sólo puede utilizarse en:

- Una instrucción SET (sólo formatos 5 y 7)
- Una condición relacional

- La expresión USING de una instrucción CALL o en la cabecera Procedure Division.
- El operando para los registros especiales LENGTH OF y ADDRESS OF.

Un elemento de datos de puntero de procedimientos sólo puede utilizarse en:

- Una instrucción SET (sólo Formato 6)
- Una condición relacional
- La expresión USING de una instrucción CALL o en la cabecera Procedure Division.
- El operando para los registros especiales LENGTH OF y ADDRESS OF.
- La instrucción CALL como destino

Si los punteros se utilizan en una condición relacional, los únicos operadores válidos son igual a y no igual a.

Como los elementos de datos de puntero no son simplemente números binarios del sistema AS/400, no funciona la manipulación de punteros como enteros.

Los elementos de datos de puntero se definen explícitamente con la cláusula USAGE IS POINTER y están implícitos al utilizar un registro especial ADDRESS OF o la ADDRESS OF de un elemento.

Si un elemento de grupo se describe con la cláusula USAGE IS POINTER, los elementos iniciales dentro del elemento de grupo son elementos de datos de puntero. El grupo en sí no es un elemento de datos de puntero y no puede utilizarse en la sintaxis donde se permite un elemento de datos de puntero. Si un elemento de grupo se describe con la cláusula USAGE PROCEDURE-POINTER, se aplican las mismas normas. La cláusula USAGE de un elemento elemental no puede contradecir la cláusula USAGE de un grupo al que pertenece el elemento.

Los punteros pueden ser parte de un grupo al que se hace referencia en una instrucción MOVE o en una instrucción de entrada/salida; sin embargo, si un puntero forma parte de un grupo, no existe ninguna conversión de valores de puntero a otra forma de representación interna al ejecutar la instrucción.

Alineación de puntero

Los punteros pueden definirse en cualquier nivel (excepto el 88) de las secciones File, Working-Storage o Linkage Section.

Cuando se hace referencia a un puntero en un sistema AS/400, debe ser en un límite de almacenamiento de 16 bytes. La **alineación de puntero** hace referencia al proceso de colocar elementos de puntero del compilador ILE COBOL/400 en desplazamientos múltiples de 16 bytes desde el principio del registro. Si un elemento de puntero no se encuentra en un límite de 16 bytes, se envía una excepción de alineación de puntero al programa ILE COBOL/400. En general, las excepciones de alineación de puntero se producen en la Linkage Section, donde es responsabilidad del usuario alinear dichos elementos.

En las File Section y Working-Storage Section, el compilador asegura que esta excepción no se produzca al añadir elementos FILLER implícitos. Cada fez que el compilador añade un elemento FILLER implícito, se emite un aviso. En la Linkage Section, el compilador no añade ningún elemento FILLER implícito; sin embargo,

los avisos se emiten indicando cuantos bytes hubiera añadido FILLER si el elemento de grupo hubiera aparecido en las File Section o Working-Storage Section.

Puede definir un elemento de datos como un puntero especificando la cláusula USAGE IS POINTER o USAGE IS PROCEDURE-POINTER, tal como se muestra en el ejemplo siguiente:

```
 ID DIVISION.
 PROGRAM-ID. PROGA.

 WORKING-STORAGE SECTION.
  77  APTR  USAGE POINTER.
  77  APROC-PTR USAGE PROCEDURE-POINTER.
  01  AB.
      05 BPTR  USAGE POINTER.
      05  BVAR  PIC S9(3) PACKED-DECIMAL.

       LINKAGE SECTION.
  01  AVAR.
      05 CVAR  PIC X(30).

 PROCEDURE DIVISION.
      SET APTR TO ADDRESS OF AVAR.
      SET APROC-PTR TO ENTRY "PROGA".
```

En el ejemplo anterior, AVAR es un elemento de datos de nivel 01, así que ADDRESS OF AVAR es el registro especial ADDRESS OF. Debido a que el registro especial es una área de almacenamiento real, la instrucción SET traslada el contenido de ADDRESS OF AVAR al elemento de datos de puntero APTR.

En el ejemplo anterior, si la instrucción SET utilizara ADDRESS OF CVAR, no existiría ningún registro especial. En su lugar, el elemento de datos de puntero APTR se asigna a la dirección calculada de CVAR.

En el ejemplo anterior, la segunda instrucción SET posiciona al elemento de datos de puntero de procedimientos APROC-PTR en el primer programa ILE COBOL/400 "PROGA".

Escritura de la File Section y de la Working Storage Section para la alineación del puntero

En la File Section y en la Working-Storage Section, todos los elementos del nivel 01 y los del nivel 77 (y algunos del nivel 66) se sitúan en límites de 16 bytes.

Dentro de una estructura de grupo, los punteros también deben situarse en un límite de 16 bytes. Para asegurarlo, el compilador ILE COBOL/400 añade elementos FILLER justo delante de los punteros. Para evitar estos elementos FILLER, coloque los punteros al principio de un elemento de grupo.

Si el puntero forma parte de una tabla, el primer elemento de la tabla se coloca en un límite de 16 bytes. Para asegurarse de que todas las apariciones siguientes del puntero se encuentran en un límite de 16 bytes, se añade en caso necesario un elemento FILLER al final de la tabla.

A continuación encontrará un ejemplo de alineación de puntero:

```
WORKING-STORAGE SECTION.
 77  APTR  USAGE POINTER.
 01  AB.
     05 ALPHA-NUM PIC X(10).
     05  BPTR  USAGE PROCEDURE-POINTER.
 01  EF.
     05 ARRAY-1 OCCURS 3 TIMES.
        10 ALPHA-NUM-TWO PIC X(14).
        10 CPTR  USAGE POINTER.
        10 ALPHA-NUM-THREE PIC X(5).
```

En el ejemplo anterior, APTR es un elemento de datos de puntero. Por consiguiente, el elemento de nivel 77 se coloca en un límite de 16 bytes. El elemento de grupo AB es un elemento de nivel 01 y se coloca automáticamente en un límite de 16 bytes. Dentro del elemento de grupo AB, BPTR no está en un límite de 16 bytes. Para alinearlo correctamente, el compilador inserta un elemento FILLER de 6 bytes después de ALPHA-NUM. Finalmente, CPTR necesita un FILLER de 2 bytes para alinear su primera aparición. Debido a que ALPHA-NUM-THREE sólo tiene 5 bytes de longitud, debe añadirse otro FILLER de 11 bytes al final de ARRAY-1 para alinear todas las apariciones siguientes de CPTR.

Cuando un puntero se define en la Sección de archivos y un fichero no tiene activado el bloqueo, cada elemento de nivel 01 se colocará en un límite de 16 bytes. Si un archivo tiene el bloqueo activado, sólo se garantiza que el primer registro de un bloque está en un límite de 16 bytes. Es por eso que los punteros no deben definirse para archivos con bloqueo activado. Para obtener más información sobre el bloqueo, consulte “Desbloqueo de registros de entrada y bloqueo de registros de salida” en la página 308.

Redefinición de punteros

Un elemento de datos de puntero o un elemento de datos de puntero de procedimientos pueden ser el sujeto o el objeto de una cláusula REDEFINES.

Cuando un puntero es el sujeto de una cláusula REDEFINES, el elemento de datos de objeto debe estar en un límite de 16 bytes. Por ejemplo:

```
WORKING-STORAGE SECTION.
 01  AB.
     05 ALPHA-NUM PIC X(16).
     05 APTR REDEFINES ALPHA-NUM USAGE POINTER.
     05 BPTR  USAGE POINTER.
     05 CPTR REDEFINES BPTR USAGE POINTER.
```

En el ejemplo anterior, tanto APTR como CPTR son elementos de datos de puntero que definen elementos alineados de 16 bytes. En el ejemplo siguiente, el elemento redefinido daría como resultado un error de compilador grave:

```
WORKING-STORAGE SECTION.
 01  EF.
     05 ALPHA-NUM PIC X(5).
     05 HI.
        10 ALPHA-NUM-TWO PIC X(11).
        10 APTR  USAGE POINTER.
     05 BPTR REDEFINES HI USAGE POINTER.
```

En el ejemplo anterior, APTR está alineado en un límite de 16 bytes. Es decir, el compilador ILE COBOL/400 no tuvo que añadir elementos FILLER para alinear APTR. El elemento de grupo HI no se encuentra en un límite de 16 bytes y entonces tampoco lo está el elemento de datos de puntero BPTR. Como el compilador ILE COBOL/400 no puede añadir elementos FILLER para situar BPTR en un límite de 16 bytes, se producirá un error grave. En el ejemplo siguiente, similar al anterior, el compilador ILE COBOL/400 puede situar al elemento de datos de puntero en un límite de 16 bytes:

```
WORKING-STORAGE SECTION.
 01  EF.
     05 ALPHA-NUM PIC X(5).
     05 HI.
        10 ALPHA-NUM-TWO PIC X(11).
        10 APTR  USAGE POINTER.
        10 ALPHA-NUM-THREE PIC X(5).
     05 KL REDEFINES HI.
        10 BPTR USAGE POINTER.
```

En el ejemplo anterior, el elemento de grupo KL no se encuentra en un límite de 16 bytes; sin embargo, el compilador añade un FILLER de 11 bytes delante del elemento de datos de puntero BPTR para asegurarse de que se encontrará en un límite de 16 bytes.

Inicialización de punteros utilizando la constante figurativa NULL

La constante figurativa NULL representa un valor utilizado para indicar que los elementos de datos definidos con USAGE IS POINTER, USAGE IS PROCEDURE-POINTER, ADDRESS OF o el registro especial ADDRESS OF no contienen una dirección válida. Por ejemplo:

```
WORKING-STORAGE SECTION.
 77 APTR  USAGE POINTER VALUE NULL.
PROCEDURE DIVISION.
    IF APTR = NULL THEN
      DISPLAY 'APTR IS NULL'
    END-IF.
```

En el ejemplo anterior, el puntero APTR está establecido en NULL en la Working-Storage Section. La comparación en la Procedure Division será verdadera y se ejecutará la instrucción de visualización.

En el sistema AS/400, el valor inicial de un elemento de datos de puntero o de un elemento de datos de puntero de procedimientos, con o sin una cláusula VALUE de NULL, es igual a NULL.

Lectura y grabación de punteros

Los punteros pueden definirse en la File Section y pueden establecerse y utilizarse como cualquier otro elemento de datos del puntero de la Working-Storage. Sin embargo, existen algunas restricciones:

- Si un archivo tiene el bloqueo activado, sólo se garantiza que el primer registro de un bloque esté en un límite de 16 bytes. Es por eso que los punteros no deben definirse para archivos con bloqueo activado.

- Un registro que contenga punteros puede grabarse en un archivo, sin embargo, en lecturas posteriores de ese registro, los elementos de datos de puntero y los elementos de datos de puntero de procedimientos son igual a NULL.

Utilización del registro especial LENGTH OF con punteros

El registro especial LENGTH OF contiene el número de bytes utilizado por un identificador. Da como resultado un valor de 16 para un elemento de datos de puntero o un elemento de datos de puntero de procedimientos.

Puede utilizar LENGTH OF en la Procedure Division, en cualquier lugar donde se utilice un elemento de datos numérico que tenga la misma definición que la definición implícita del registro especial LENGTH OF; sin embargo, LENGTH OF no puede utilizarse como un subíndice ni como un elemento de datos de recepción. LENGTH OF tiene la definición implícita:

```
USAGE IS BINARY, PICTURE 9(9)
```

El ejemplo siguiente muestra cómo puede utilizar LENGTH OF con punteros:

```
 WORKING-STORAGE SECTION.
  77  APTR  USAGE POINTER.
  01  AB.
      05  BPTR  USAGE PROCEDURE-POINTER.
      05  BVAR  PIC S9(3) PACKED-DECIMAL.
      05  CVAR  PIC S9(3) PACKED-DECIMAL.
 PROCEDURE DIVISION.
      MOVE LENGTH OF AB TO BVAR.
      MOVE LENGTH OF BPTR TO CVAR.
```

En el ejemplo anterior, la longitud del elemento de grupo AB se traslada a la variable BVAR. BVAR tiene un valor de 20 debido a que BPTR tiene 16 bytes de longitud y ambas variables, BVAR y CVAR tienen 2 bytes de longitud. CVAR recibe un valor de 16.

También puede utilizar el registro especial LENGTH OF para configurar estructuras de datos dentro de espacios de usuarios o para aumentar las direcciones recibidas desde otro programa. Para ver un ejemplo de un programa que utilice el registro especial LENGTH OF para definir estructuras de datos dentro de espacios de usuarios, consulte la Figura 62 en la página 245.

Establecimiento de la dirección de elementos de la Linkage Section

Generalmente, cuando un programa ILE COBOL/400 llama a otro programa, los operandos se transfieren desde el programa de llamada ILE COBOL/400 al programa llamado ILE COBOL/400 de la siguiente forma:

- el programa de llamada utiliza la instrucción CALL USING para transferir los operandos al programa llamado y,
- el programa llamado especifica la expresión USING en la cabecera Procedure Division.

Para cada operando listado en la instrucción CALL USING del programa de llamada ILE, debe haber el operando correspondiente especificado por la expresión USING de la Procedure Division del programa llamado.

Al utilizar el registro especial ADDRESS OF, ya no necesita asegurar una correlación de uno a uno entre las expresiones USING de ambos programas. Para aquellos elementos de datos de la Linkage Section que no están especificados en la expresión USING de la cabecera Procedure Division, puede utilizar una instrucción SET para especificar la dirección inicial de la estructura de datos. Una vez la instrucción SET se ejecuta puede hacerse referencia libremente al elemento de datos, ya que la dirección del elemento de datos ya está establecida. Para obtener un ejemplo de una instrucción SET utilizada de esta forma, consulte la Figura 63 en la página 246. En la Figura 63 en la página 246, 15 y 16 muestran cómo se utiliza la instrucción SET para establecer la dirección inicial de las estructuras de datos *ls-header-record* y *ls-user-space* al principio del espacio de usuario.

Utilización de ADDRESS OF y del registro especial ADDRESS OF

Al especificar ADDRESS OF en un programa ILE COBOL/400, el compilador determina si debe utilizarse la dirección calculada de un elemento de datos, denominada ADDRESS OF, o el registro especial ADDRESS OF. El registro especial ADDRESS OF es la dirección inicial de la estructura de datos desde la cual se determinan todas las direcciones calculadas. Debido a que el registro especial ADDRESS OF es la dirección inicial de una estructura, debe ser un elemento de datos de nivel 01 ó 77. Si modifica por referencia este elemento de datos, ya no será la dirección inicial de la estructura de datos. Es una dirección calculada o ADDRESS OF. Si toma la ADDRESS OF de un elemento inicial y la ADDRESS OF del elemento de nivel 01 se ha establecido en NULL, aparecerá una excepción de puntero (MCH3601).

No puede utilizar la ADDRESS OF calculada donde un elemento pueda cambiarse. Sólo puede cambiarse el registro especial ADDRESS OF. Por ejemplo, en la Figura 63 en la página 246, la instrucción SET del número 17 utiliza el registro especial ADDRESS OF porque es un elemento de nivel 01. En el número 18 , ADDRESS OF se utiliza porque, aunque es un elemento de nivel 01, se modifica por referencia.

Utilización de punteros en una instrucción MOVE

Los punteros elementales no pueden trasladarse utilizando la instrucción MOVE; debe utilizarse una instrucción SET. Sin embargo, los punteros se trasladan implícitamente cuando forman parte de un elemento de grupo.

Al compilar una instrucción MOVE, el compilador ILE COBOL/400 genera un código para mantener (un puntero MOVE) o no mantener (un puntero no MOVE) punteros dentro de un elemento de grupo.

Un puntero MOVE se realiza cuando se cumplen todas la condiciones siguientes:

1. El origen o el receptor de una instrucción MOVE contienen un puntero
2. Ambos elementos tienen como mínimo 16 bytes de longitud
3. Los elementos de datos están alineados correctamente
4. Los elementos de datos son elementos alfanuméricos o de grupo.

De las condiciones listadas anteriormente, determinar si dos elementos de datos están alineados correctamente puede ser la más difícil.

Nota: Un puntero MOVE es más lento que utilizar la instrucción SET para trasladar un puntero.

Si los elementos trasladados son elementos de nivel 01 o forman parte de un elemento de nivel 01, deben estar en el mismo desplazamiento relativo a un límite de 16 bytes para que se produzca un puntero MOVE. (Se emite un aviso si esto no es verdadero.) Los ejemplos siguientes muestran tres estructuras de datos y los resultados al emitir una instrucción MOVE:

```
WORKING-STORAGE SECTION.
   01  A.
       05  B      PIC X(10).
       05  C.
           10  D      PIC X(6).
           10  E      POINTER.

   01  A2.
       05  B2     PIC X(6).
       05  C2.
           10  D2     PIC X(10).
           10  E2     POINTER.

   01  A3.
       05  B3     PIC X(22).
       05  C3.
           10  D3     PIC X(10).
           10  E3     POINTER.

PROCEDURE DIVISION.
MOVE  A  a A2.  1
MOVE  A  a A3.  1
MOVE  C  a C2.  2
MOVE  C2 a C3.  3
```

Figura 61. Utilización de punteros en una instrucción MOVE

1 El resultado es un puntero MOVE porque el desplazamiento de cada elemento de grupo a trasladar es cero. Se mantiene la integridad del puntero.

2 El resultado es un puntero no MOVE porque los desplazamientos no coinciden. El desplazamiento del elemento de grupo C es 10 y el del elemento de grupo C2 es 6. No se mantiene la integridad del puntero.

3 El resultado es un puntero MOVE porque el desplazamiento del elemento de grupo C2 es 6 y el de C3 relativo a un límite de 16 bytes es también 6. (Cuando el desplazamiento es superior a 16, el desplazamiento relativo a un límite de 16 bytes se calcula dividiendo el desplazamiento por 16. El resto es el desplazamiento relativo. En este caso, el desplazamiento era 22, lo que, al dividir por 16, deja un resto o un desplazamiento de 6). Se mantiene la integridad del puntero.

Si un elemento de grupo contiene un puntero y el compilador ILE COBOL/400 no puede determinar el desplazamiento relativo a un límite de 16 bytes, el compilador ILE COBOL/400 emite un mensaje de aviso y se intenta el traslado del puntero. Sin embargo, la integridad del

puntero puede que no se mantenga. El compilador ILE COBOL/400 no puede determinar el desplazamiento si el elemento se define en la Linkage Section o si el elemento se modifica por referencia con una posición inicial desconocida. Debe asegurarse de que se mantiene la alineación del puntero o puede producirse un error de comprobación de máquina.

El compilador ILE COBOL/400 coloca todos los elementos de nivel 01 y de nivel 77 en un límite de 16 bytes tanto si contienen punteros como si no.

Utilización de punteros en una instrucción CALL

Cuando se transfiere un elemento de datos de puntero o elemento de datos de puntero de procedimientos en una instrucción CALL, el elemento se trata como todos los demás elementos USING. En otras palabras, cuando un elemento de datos de puntero se transfiere BY REFERENCE (o BY CONTENT), se transfiere al programa llamado un puntero del elemento de datos de puntero (o una copia del elemento de datos de puntero). Cuando un elemento de datos de puntero se transfiere BY VALUE, el contenido del elemento de datos de puntero se coloca en la pila de llamadas. Los elementos de datos del puntero de procedimientos se transfieren de forma similar.

Deben proporcionarse consideraciones especiales cuando una instrucción CALL con la expresión BY CONTENT se utiliza para transferir punteros y elementos de grupo que contengan punteros. Es similar al caso de una sentencia MOVE. Para una instrucción CALL...BY CONTENT, se realiza un MOVE implícito de un elemento para crearlo en una área temporal. Para asegurar la alineación del puntero en este puntero MOVE, el compilador ILE COBOL/400 o la ejecución deben determinar el desplazamiento del elemento BY CONTENT relativo al límite de 16 bytes. Para obtener el mejor rendimiento, el elemento BY CONTENT debe codificarse de forma que el compilador ILE COBOL/400 pueda determinar este desplazamiento.

La ejecución ILE COBOL/400 tiene que determinar el desplazamiento de un elemento relativo a un límite de 16 bytes cuando el elemento BY CONTENT:

- Se modifica por referencia con una posición inicial desconocida o
- Se define en la Linkage Section.

Cuando un operando se modifica por referencia, el desplazamiento es la posición inicial de modificación de referencia menos uno, más el desplazamiento del operando dentro de la estructura de datos. Cuando un operando se encuentra en la Linkage Section, su desplazamiento puede determinarse desde el programa de llamada.

Para evitar problemas de alineación de puntero, transfiera los elementos utilizando BY REFERENCE.

Ajuste del valor de los punteros

El ejemplo siguiente muestra cómo se puede ajustar el valor de un puntero aumentándolo (UP BY) o disminuyéndolo (DOWN BY) un valor entero. Este método de modificar el valor de un puntero puede ser útil al acceder a elementos de una tabla a los que haga referencia un elemento de datos de puntero.

```
    LINKAGE SECTION.
01  USER-INFO.
    05  USER-NAME PIC X(10).
    05  USER-ID   PIC 9(7).
WORKING-STORAGE SECTION.
01  A.
    05 ARRAY-USER-INFO occurs
          300 to 2000 times.
       10  USER-NAME PIC X(10).
       10  USER-ID   PIC 9(7).

PROCEDURE DIVISION.

     SET ADDRESS OF USER-INFO TO
               ADDRESS OF ARRAY-USER-INFO(200).[1]

     SET ADDRESS OF USER-INFO UP BY LENGTH OF USER-INFO.[2]

     MOVE "NEW NAME" TO USER-NAME OF USER-INFO.[3]
```

Notas:

1. La primera instrucción SET proporciona al elemento de datos USER-INFO la misma dirección que el elemento 200 de la matriz ARRAY-USER-INFO.
2. La segunda instrucción SET proporciona al elemento de datos USER-INFO la misma dirección que el elemento 201 de la matriz ARRAY-USER-INFO (en otras palabras, un elemento más que la primera instrucción SET).
3. Esta instrucción es la misma que:

   ```
   MOVE "NEW NAME" to USER-NAME OF ARRAY-USER-INFO (201).
   ```

Para obtener una definición completa de la instrucción SET, consulte el manual *ILE COBOL/400 Reference*.

Acceso a espacios de usuario utilizando punteros y API

El ejemplo siguiente muestra cómo utilizar punteros para acceder a los espacios de usuario y encadenar registros juntos.

POINTA es un programa que lee los nombres y las direcciones de clientes en un espacio de usuario y luego visualiza la información en una lista. El programa asume que la información de cliente existe en un archivo denominado POINTACU.

El campo de dirección de cliente es un campo de longitud variable para permitir las direcciones largas.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ESTE ES EL ARCHIVO DE INFORMACIÓN DE CLIENTE - POINTACUST
     A
     A
     A          R FSCUST                    TEXT('REGISTRO MAESTRO DE CLIENTE')
     A            FS_CUST_NO     8S00       TEXT('NÚMERO DE CLIENTE')
     A                                       ALIAS(NÚMERO_CLIENTE_FS)
     A            FS_CUST_NM     20         TEXT('NOMBRE CLIENTE')
     A                                       ALIAS(NOMBRE_CLIENTE_FS)
     A            FS_CUST_AD     100        TEXT('DIRECCIÓN CLIENTE')
     A                                       ALIAS(DIRECCIÓN-CLIENTE_FS)
     A                                       VARLEN
```

Figura 62. Ejemplo de la utilización de punteros para acceder a espacios de usuario -- DDS

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     2
                                    F u e n t e
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         000010 PROCESS varchar  1
    1    000020 ID DIVISION.
         000030* Este programa lee un archivo de registros de longitud
         000040*  variable en un espacio del usuario. Entonces muestra
         000050*  los registros en la pantalla.
    2    000060 PROGRAM-ID. pointa.
    3    000070 ENVIRONMENT DIVISION.
    4    000080 CONFIGURATION SECTION.
    5    000090 SPECIAL-NAMES. CONSOLE IS CRT,
    7    000100                CRT STATUS IS ws-crt-status.  2
    8    000110 INPUT-OUTPUT SECTION.
    9    000120 FILE-CONTROL.
   10    000130     SELECT cust-file ASSIGN TO DATABASE-pointacu
   12    000140            ORGANIZATION IS SEQUENTIAL
   13    000150            FILE STATUS IS ws-file-status.
   14    000160 DATA DIVISION.
   15    000170 FILE SECTION.
   16    000180 FD  cust-file.
   17    000190 01  fs-cust-record.
         000200* copia los nombres de campo convirtiendo el subrayado a
         000210*  guiones y utilizando nombres de alias
         000220 COPY DDR-ALL-FORMATS-I OF pointacu.
   18   +000001       05  POINTACU-RECORD PIC X(130).                                    <-ALL-FMTS
        +000002*    FORMATO E-S:FSCUST    DE ARCHIVO POINTACU    DE BIBLIOTECA TESTLIB   <-ALL-FMTS
        +000003*                          REGISTRO MAESTRO CLIENTE                       <-ALL-FMTS
        +000004      05  FSCUST        REDEFINES POINTACU-RECORD.                      <-ALL-FMTS
        +000005          06 FS-CUST-NUMBER        PICS9(8).                             <-ALL-FMTS
        +000006*                 NÚMERO DE CLIENTE                                     <-ALL-FMTS
        +000007          06 FS-CUST-NAME                                                <-ALL-FMTS
        +000008*                 NOMBRE DE CLIENTE                                     <-ALL-FMTS
        +000009          06 FS-CUST-ADDRESS.  3                                  <-ALL-FMTS
        +000010*              (Campo de longitud variable)                              <-ALL-FMTS
        +000011              49 FS-CUST-ADDRESS-LENGTH                                 <-ALL-FMTS
        +000012                                 PIC S9(4) COMP-4.                      <-ALL-FMTS
        +000013              49 FS-CUST-ADDRESS-DATA                                   <-ALL-FMTS
        +000014                                 PIC X(100).                            <-ALL-FMTS
        +000015*                 DIRECCIÓN DE CLIENTE                                  <-ALL-FMTS
   18    000230 WORKING-STORAGE SECTION.
   19    000240 01  ws-file-status.
   20    000250     05 ws-file-status-1 PIC X.
   21    000260        88  ws-file-stat-good   VALUE "0".
   22    000270        88  ws-file-stat-at-end VALUE "1".
   23    000280     05 ws-file-status-2 PIC X.
   24    000290 01  ws-crt-status.  4
   25    000300     05 ws-status-1           PIC 9(2).
   26    000310        88 ws-status-1-ok       VALUE 0.
   27    000320        88 ws-status-1-func-key VALUE 1.
   28    000330        88 ws-status-1-error    VALUE 9.
   29    000340     05 ws-status-2           PIC 9(2).
   30    000350        88 ws-func-03           VALUE 3.
   31    000360        88 ws-func-07           VALUE 7.
   32    000370        88 ws-func-08           VALUE 8.
   33    000380     05 ws-status-3           PIC 9(2).
   34    000390 01  ws-params.  5
   35    000400     05  ws-space-ptr         POINTER.  6
   36    000410     05  ws-space.
   37    000420       10  ws-space-name      PIC X(10) VALUE "MYSPACE".
   38    000430       10  ws-space-lib       PIC X(10) VALUE "QTEMP".
   39    000440     05  ws-attr              PIC X(10) VALUE "PF".
   40    000450     05  ws-init-size         PIC S9(5) VALUE 32000 BINARY.
   41    000460     05  ws-init-char         PIC X     VALUE SPACE.
   42    000470     05  ws-auth              PIC X(10) VALUE "*ALL".
   43    000480     05  ws-text              PIC X(50) VALUE
         000490                               "Registros de información del cliente".
   44    000500     05  ws-replace           PIC X(10) VALUE "*YES".
   45    000510     05  ws-err-data.  7
   46    000520         10 ws-input-l        PIC S9(6) BINARY VALUE 16.
   47    000530         10 ws-output-l       PIC S9(6) BINARY.
   48    000540         10 ws-exception-id   PIC X(7).
   49    000550         10 ws-reserved       PIC X(1).
```

Figura 63 (Parte 1 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     3
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         000560
   50    000570 77  ws-accept-data           PIC X   VALUE SPACE.
   51    000580   88 ws-acc-blank             VALUE SPACE.
   52    000590   88 ws-acc-create-space      VALUE "Y", "y".
   53    000600   88 ws-acc-use-prv-space     VALUE "N", "n".
   54    000610   88 ws-acc-delete-space      VALUE "Y", "y".
   55    000620   88 ws-acc-save-space        VALUE "N", "n".
         000630
   56    000640 77  ws-prog-indicator        PIC X   VALUE "G".
   57    000650   88  ws-prog-continue       VALUE "G".
   58    000660   88  ws-prog-end            VALUE "C".
   59    000670   88  ws-prog-loop           VALUE "L".
         000680
   60    000690 77  ws-line                  PIC 99.
         000700* línea de mensaje de error
   61    000710 77  ws-error-msg             PIC X(50) VALUE SPACES.
         000720* indicador de información de más direcciones
   62    000730 77  ws-plus                  PIC X.
         000740* longitud de la información de dirección a visualizar
   63    000750 77  ws-temp-size            PIC 9(2).
         000760
   64    000770 77  ws-current-rec           PIC S9(4) VALUE 1.
   65    000780 77  ws-old-rec               PIC S9(4) VALUE 1.
   66    000790 77  ws-old-space-ptr         POINTER.
         000800* número máximo de líneas a visualizar
   67    000810 77  ws-displayed-lines       PIC S99 VALUE 20.
         000820* línea en la que se empiezan a visualizar los registros
   68    000830 77  ws-start-line            PIC S99 VALUE 5.
         000840* variables para crear nuevo registro en espacio
   69    000850 77  ws-addr-inc              PIC S9(4) PACKED-DECIMAL.
   70    000860 77  ws-temp                  PIC S9(4) PACKED-DECIMAL.
   71    000870 77  ws-temp-2                PIC S9(4) PACKED-DECIMAL.
         000880* puntero de registro anterior
   72    000890 77  ws-cust-prev-ptr         POINTER VALUE NULL.
   73    000900 LINKAGE SECTION.
   74    000910 01 ls-header-record.  8
   75    000920    05  ls-hdr-cust-ptr            USAGE POINTER.
         000930* número de registros leídos del archivo
   76    000940    05  ls-record-counter          PIC S9(3) BINARY.
   77    000950    05  FILLER                     PIC X(14).  9
   78    000960 01 ls-user-space.  1
   79    000970     05  ls-customer-rec.
         000980* puntero a registro del cliente anterior
   80    000990         10 ls-cust-prev-ptr       USAGE POINTER.
   81    001000         10 ls-cust-rec-length     PIC S9(4) BINARY.
   82    001010         10 ls-cust-name           PIC X(20).
   83    001020         10 ls-cust-number         PIC S9(8).
         001030* longitud total del registro incluyendo bytes de relleno
         001040*  para asegurarse límite de 16 bytes en próximo registro
   84    001050         10 ls-cust-address-length  PIC S9(4) BINARY.
   85    001060     05  ls-cust-address-data      PIC X(116).
         001070
         001080* El tamaño de ls-user-space sobrepasa en 16 lo
         001090* necesario. Ello permite el establecimiento de la dirección
         001100* de inicio del siguiente registro sin superar el tamaño
         001110* declarado. El tamaño es 16 bytes mayor para permitir la
         001120* alineación de punteros.
```

Figura 63 (Parte 2 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     4
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
   86     001130 PROCEDURE DIVISION.
          001140* no es necesario para la entrada "USING" en PROC... DIV.
   87     001150 DECLARATIVES.
          001160 cust-file-para SECTION.
          001170     USE AFTER ERROR PROCEDURE ON cust-file.
          001180 cust-file-para-2.
   88     001190     MOVE "Error XX en puntero archivo" TO ws-error-msg.
   89     001200     MOVE ws-file-status TO ws-error-msg(7:2).
          001210 END DECLARATIVES.
          001220
          001230 main-program section.
          001240 mainline.
          001250* lee pantalla inicial hasta entrada correcta de datos
   90     001260     SET ws-prog-loop TO TRUE.
   91     001270     PERFORM initial-display THRU read-initial-display
          001280         UNTIL NOT ws-prog-loop.
          001290* si desea continuar con el programa y desea crear área
          001300*   de información del cliente, rellene el espacio con
          001310*   registros del archivo cliente
   92     001320     IF ws-prog-continue AND
          001330        ws-acc-create-space THEN
   93     001340       PERFORM read-customer-file
   94     001350       MOVE 1 TO ws-current-rec
          001360* establece ptr al registro de cabecera
   95     001370       SET ADDRESS OF ls-header-record TO ws-space-ptr
          001380* establece en espacio el primer registro del cliente
   96     001390       SET ADDRESS OF ls-user-space TO ls-hdr-cust-ptr
          001400     END-IF.
   97     001410     IF ws-prog-continue THEN
   98     001420       PERFORM main-loop UNTIL ws-prog-end
          001430     END-IF.
          001440 end-program.
   99     001450     PERFORM clean-up.
  100     001460     STOP RUN.
          001470
          001480 initial-display.  11
  101     001490     DISPLAY "Crear área de información del cliente" AT 0118 WITH
          001500                BLANK SCREEN REVERSE-VIDEO
          001510             "Crear área de información del cliente (Y/N)=>   <="
          001520                AT 1015
          001530             "F3=Salir" AT 2202.
  102     001540     IF ws-error-msg NOT = SPACES THEN
  103     001550       DISPLAY ws-error-msg at 2302 with beep highlight
  104     001560       MOVE SPACES TO ws-error-msg
          001570     END-IF.
          001580
          001590 read-initial-display.  12
  105     001600     ACCEPT ws-accept-data AT 1056 WITH REVERSE-VIDEO
          001610       ON EXCEPTION
  106     001620         IF ws-status-1-func-key THEN
  107     001630           IF ws-func-03 THEN
  108     001640             SET ws-prog-end TO TRUE
          001650           ELSE
  109     001660             MOVE "Tecla de función no válida" TO ws-error-msg
          001670           END-IF
          001680         ELSE
  110     001690           MOVE "Error desconocido" TO ws-error-msg
```

Figura 63 (Parte 3 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     5
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         001700         END-IF
         001710       NOT ON EXCEPTION
  111    001720         IF ws-acc-create-space THEN
  112    001730           PERFORM create-space THRU set-space-ptrs
  113    001740           SET ws-prog-continue TO TRUE
         001750         ELSE
  114    001760           IF ws-acc-use-prv-space THEN
  115    001770             PERFORM get-space
  116    001780             IF ws-space-ptr = NULL
  117    001790               MOVE "No existe área información cliente" TO ws-error-msg
         001800             ELSE
  118    001810               PERFORM set-space-ptrs
  119    001820               SET ws-prog-continue TO TRUE
         001830             END-IF
         001840           ELSE
  120    001850             MOVE "Entrado carácter no válido" TO ws-error-msg
         001860           END-IF
         001870         END-IF
         001880     END-ACCEPT.
         001890
         001900 create-space.
  121    001910     CALL "QUSCRTUS" USING ws-space, ws-attr, ws-init-size,  13
         001920                           ws-init-char, ws-auth, ws-text,
         001930                           ws-replace, ws-err-data.
         001940
         001950* comprobación de errores en creación de espacio podría añadirse aquí
         001960
         001970 get-space.
  122    001980     CALL "QUSPTRUS" USING ws-space, ws-space-ptr, ws-err-data.  14
         001990
         002000 set-space-ptrs.
         002010* establece el registro de cabecera al principio del espacio
  123    002020     SET ADDRESS OF ls-header-record  15
         002030         ADDRESS OF ls-user-space  16
         002040         TO ws-space-ptr.
         002050* establece el registro del primer cliente después del de cabecera
  124    002060     SET ADDRESS OF ls-user-space TO  17
         002070         ADDRESS OF ls-user-space(LENGTH OF ls-header-record  18
         002080                                  + 1:1).
         002090* guarda ptr en primer registro en registro de cabecera
  125    002100     SET ls-hdr-cust-ptr TO ADDRESS OF ls-user-space.
         002110
         002120 delete-space.
  126    002130     CALL "QUSDLTUS" USING ws-space, ws-err-data.  19
         002140
         002150 read-customer-file.
         002160* lee todos los registro del archivo de cliente y mueve a espacio
  127    002170     OPEN INPUT cust-file.
  128    002180     IF ws-file-stat-good THEN
  129    002190       READ cust-file AT END CONTINUE
         002200       END-READ
  131    002210       PERFORM VARYING ls-record-counter FROM 1 BY 1
         002220               UNTIL not ws-file-stat-good
  132    002230         SET  ls-cust-prev-ptr  TO ws-cust-prev-ptr
         002240*  Mueve la información del archivo al espacio
  133    002250         MOVE fs-cust-name      TO ls-cust-name
  134    002260         MOVE fs-cust-number    TO ls-cust-number
```

Figura 63 (Parte 4 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     6
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
  135     002270         MOVE fs-cust-address-length  TO ls-cust-address-length
  136     002280         MOVE fs-cust-address-data(1:fs-cust-address-length)
          002290              TO ls-cust-address-data(1:ls-cust-address-length)
          002300*  Guarda ptr en registro actual
  137     002310         SET  ws-cust-prev-ptr  TO ADDRESS OF ls-user-space
          002320*  Asegúrese de que registro siguiente cumple límite 16 bytes
  138     002330         ADD LENGTH OF ls-customer-rec  2
          002340             ls-cust-address-length TO 1 GIVING ws-addr-inc
  139     002350         DIVIDE ws-addr-inc BY 16 GIVING ws-temp
          002360             REMAINDER ws-temp-2
  140     002370         SUBTRACT ws-temp-2 FROM 16 GIVING ws-temp
          002380*  Guarda longitud total de registro en espacio de usuario
  141     002390         ADD ws-addr-inc TO ws-temp GIVING ls-cust-rec-length
  142     002400         SET ADDRESS OF ls-user-space
          002410          TO ADDRESS OF ls-user-space(ls-cust-rec-length + 1:1)
          002420*  Obtiene el registro siguiente del archivo
  143     002430         READ cust-file AT END CONTINUE
          002440         END-READ
          002450       END-PERFORM
          002460*  Al final del bucle tiene un registro más que los reales
          002470*
  145     002480       SUBTRACT 1 FROM ls-record-counter
          002490     END-IF.
  146     002500     CLOSE cust-file.
          002510
          002520 main-loop.  21
          002530* escribe los registro en pantalla hasta que se entre F3
  147     002540     DISPLAY "Información del cliente" AT 0124 WITH
          002550                BLANK SCREEN REVERSE-VIDEO
          002560             "Clien     Nombre del cliente   Cliente"
          002570                AT 0305
          002580             " Dirección"
          002590             "Número"   AT 0405
          002600             "F3=Salir" AT 2202.
          002610* si había un error pendiente lo visualiza en pantalla
  148     002620     IF ws-error-msg NOT = SPACES THEN
  149     002630       DISPLAY ws-error-msg at 2302 with beep highlight
  150     002640       MOVE SPACES TO ws-error-msg
          002650     END-IF.
          002660* si puede retroceder en la lista aparece F7 en la pantalla
  151     002670     IF ws-current-rec > 1 THEN  22
  152     002680       DISPLAY "F7=Retroceso" AT 2240
          002690     END-IF.
          002700* guarda el registro actual
  153     002710     MOVE ws-current-rec TO ws-old-rec.
  154     002720     SET  ws-old-space-ptr TO ADDRESS OF ls-user-space.  23
          002730* mueve cada registro a la pantalla
  155     002740     PERFORM VARYING ws-line FROM  ws-start-line BY 1
          002750         UNTIL ws-line > ws-displayed-lines or
          002760               ws-current-rec > ls-record-counter
          002770* si la dirección es mayor que anchura de pantalla muestra "+"
  156     002780         IF ls-cust-address-length > 40 THEN
  157     002790           MOVE "+" TO ws-plus
  158     002800           MOVE 40 TO ws-temp-size
          002810         ELSE
  159     002820           MOVE ls-cust-address-length TO ws-temp-size
  160     002830           MOVE SPACE TO ws-plus
```

Figura 63 (Parte 5 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     7
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         002840         END-IF
  161    002850         DISPLAY ls-cust-number at line ws-line column 5
         002860                 ls-cust-name ls-cust-address-data with
         002870                   size ws-temp-size ws-plus at line
         002880                   ws-line column 78
         002890* coloca registro siguiente en el espacio
  162    002900         ADD 1 TO ws-current-rec
  163    002910         SET ADDRESS OF ls-user-space
         002920          TO ADDRESS OF ls-user-space
         002930            (ls-cust-rec-length + 1:1)
         002940     END-PERFORM.
         002950* si puede avanzar coloca F8 en la pantalla
  164    002960     IF ws-current-rec < ls-record-counter THEN  22
  165    002970       DISPLAY "F8=Avance" AT 2250
         002980     END-IF.
         002990* comprueba si debe continuar, salir u obtener registro
         003000*   siguiente o registro anterior
  166    003010     SET ws-acc-blank to TRUE.
  167    003020     ACCEPT ws-accept-data WITH SECURE  24
         003030       ON EXCEPTION
  168    003040         IF ws-status-1-func-key THEN
  169    003050           IF ws-func-03 THEN
  170    003060             SET ws-prog-end TO TRUE
         003070           ELSE
  171    003080            IF ws-func-07 THEN
  172    003090              PERFORM back-screen
         003100         ELSE
  173    003110            IF ws-func-08 THEN
  174    003120              PERFORM forward-screen
         003130            ELSE
  175    003140              MOVE "Tecla de función no válida" TO ws-error-msg
  176    003150              MOVE ws-old-rec TO ws-current-rec
  177    003160              SET ADDRESS OF ls-user-space TO ws-old-space-ptr
         003170            END-IF
         003180           END-IF
         003190         ELSE
  178    003200           MOVE "Error desconocido" TO ws-error-msg
  179    003210           MOVE ws-old-rec TO ws-current-rec
  180    003220           SET ADDRESS OF ls-user-space TO ws-old-space-ptr
         003230         END-IF
         003240       NOT ON EXCEPTION
  181    003250         MOVE ws-old-rec TO ws-current-rec
  182    003260         SET ADDRESS OF ls-user-space TO ws-old-space-ptr
         003270     END-ACCEPT.
         003280
         003290 clean-up.
         003300* realiza un borrado del programa
         003310* continúa leyendo pantalla final hasta entrada correcta de datos
  183    003320     SET ws-prog-loop to TRUE.
  184    003330     SET ws-acc-blank to TRUE.
  185    003340     PERFORM final-display THRU read-final-display  25
         003350         UNTIL NOT ws-prog-loop.
         003360
         003370 final-display.
  186    003380     DISPLAY "Suprimir área de información cliente" AT 0118 WITH  26
         003390                BLANK SCREEN REVERSE-VIDEO
         003400             "Suprimir área información cliente (Y/N)=>   <="
```

Figura 63 (Parte 6 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     8
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         003410                AT 1015
         003420             "F3=Salir" AT 2202.
  187    003430     IF ws-error-msg NOT = SPACES THEN
  188    003440       DISPLAY ws-error-msg at 2302 with beep highlight
  189    003450       MOVE SPACES TO ws-error-msg
         003460     END-IF.
         003470
         003480 read-final-display.
  190    003490     ACCEPT ws-accept-data AT 1056 WITH REVERSE-VIDEO
         003500       ON EXCEPTION
  191    003510         IF ws-status-1-func-key THEN
  192    003520           IF ws-func-03 THEN
  193    003530             SET ws-prog-end TO TRUE
         003540         ELSE
  194    003550             MOVE "Tecla de función no válida" TO ws-error-msg
         003560           END-IF
         003570           ELSE
  195    003580           MOVE "Error desconocido" TO ws-error-msg
         003590           END-IF
         003600       NOT ON EXCEPTION
  196    003610         IF ws-acc-delete-space THEN
  197    003620           PERFORM delete-space
  198    003630           SET ws-prog-continue TO TRUE
         003640         ELSE
  199    003650           IF ws-acc-save-space THEN
  200    003660             SET ws-prog-continue TO TRUE
         003670           ELSE
  201    003680             MOVE "Entrado carácter no válido" TO ws-error-msg
         003690           END-IF
         003700         END-IF
         003710     END-ACCEPT.
         003720
         003730 back-screen.  27
  202    003740     IF ws-old-rec <= 1 THEN
  203    003750       MOVE "Principio de registro del cliente" TO ws-error-msg
  204    003760       MOVE ws-old-rec TO ws-current-rec  28
  205    003770       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
         003780     ELSE
  206    003790       MOVE ws-old-rec TO ws-current-rec  28
  207    003800       SET ADDRESS OF ls-user-space TO ws-old-space-ptr
  208    003810       PERFORM VARYING ws-line FROM ws-start-line BY 1
         003820         UNTIL ws-line > ws-displayed-lines or
         003830               ws-current-rec <= 1
         003840* Retrocede un registro, de uno en uno
  209    003850         SET ws-cust-prev-ptr TO ls-cust-prev-ptr  29
  210    003860         SET ADDRESS OF ls-user-space TO ws-cust-prev-ptr
  211    003870         SUBTRACT 1 FROM ws-current-rec
         003880       END-PERFORM
         003890     END-IF.
         003900
         003910 forward-screen.  3
         003920* si el registro actual es mayor o igual al máximo de registros
         003930*   de error de impresión, se ha alcanzado el máximo de registros
  212    003940     IF ws-current-rec >= ls-record-counter
  213    003950       MOVE "No hay más registros del cliente" TO ws-error-msg
  214    003960       MOVE ws-old-rec TO ws-current-rec
  215    003970       SET ADDRESS OF ls-user-space TO ws-old-space-ptr

5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/POINTA          TORAS015   96/07/26 15:31:31         Página     9
 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB
         003980     ELSE
  216    003990       MOVE ws-current-rec TO ws-old-rec
  217    004000       SET ws-old-space-ptr TO ADDRESS OF ls-user-space
         004010     END-IF.
                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 63 (Parte 7 de 7). Ejemplo de la utilización de punteros para acceder a espacios de usuario

2 CRT STATUS IS especifica un nombre de datos en el que se incluye un valor de estado después del final de una instrucción ACCEPT extendida. En este ejemplo, el valor de tecla STATUS se utiliza para determinar qué tecla de función se ha pulsado.

3 *fs-cust-address* es un campo de longitud variable. Para ver aquí nombres con sentido en lugar de FILLER, especifique *VARCHAR para el parámetro CVTOPT del mandato CRTCBLMOD o CRTBNDCBL o VARCHAR en la instrucción PROCESS, tal como se muestra en 1 . Para obtener más información sobre los campos de longitud variable, consulte “Declaración de elementos de datos utilizando tipos de datos SAA” en la página 332.

4 CRT STATUS tal como se menciona en 2 se define aquí.

5 La estructura *ws-params* contiene los parámetros utilizados al llamar a las API para acceder a los espacios de usuario.

6 *ws-space-ptr* define un elemento de datos de puntero establecido por la API QUSPTRUS. Esto señala el principio del espacio de usuario y se utiliza para establecer las direcciones de elementos en la Linkage Section.

7 *ws-err-data* es la estructura para el parámetro de error de las API de espacio de usuario. Observe que *ws-input-l* es cero, lo que significa que cualquier excepción se señala al programa y no se transfiere en el parámetro de código de error. Para obtener más información sobre los parámetros de código de error, consulte System API Reference.

8 La primera estructura de datos (*ls-header-record*) a definir en el espacio de usuario.

9 FILLER se utiliza para mantener la alineación de puntero porque hace que *ls-header-record* sea un múltiplo de 16 bytes de longitud.

1 La segunda estructura de datos (*ls-user-space*) a definir en el espacio de usuario.

11 *initial-display* muestra la pantalla Create Customer Information Area.

12 *read-initial-display* lee la primera pantalla y determina si el usuario elige continuar o finalizar el programa. Si el usuario continúa el programa pulsando Intro, éste comprobará *ws-accept-data* para ver si se ha de crear el área de información del cliente.

13 QUSCRTUS es una API utilizada para crear espacios de usuario.

14 QUSPTRUS es una API utilizada para devolver un puntero al principio de un espacio de usuario.

15 Correlaciona la primera estructura de datos (*ls-header-record*) por encima del principio del espacio de usuario.

16 Correlaciona la segunda estructura de datos (*ls-user-space*) por encima del principio del espacio de usuario.

17 Utiliza el registro especial ADDRESS OF

18 Utiliza ADDRESS OF, no el registro especial ADDRESS OF, ya que se modifica por referencia.

19 QUSDLTUS es una API utilizada para eliminar un espacio de usuario.

2 Las cuatro sentencias aritméticas siguientes calculan la longitud total de cada registro y se aseguran de que cada registro sea un múltiplo de 16 bytes de longitud.

21 *main-loop* presenta la pantalla Customer Information.

22 Esta instrucciones determinan si el programa debe visualizar las teclas de función F7 y F8.

23 Salva un puntero en el primer registro de cliente de la pantalla.

24 Esta instrucción ACCEPT espera la entrada desde la pantalla Customer Information. Basándose en la tecla de función pulsada, llama al párrafo apropiado para visualizar el siguiente conjunto de registros (*forward-screen*) o el conjunto de registros anterior, (*back-screen*) o establece un indicador para finalizar la rutina si se pulsa F3.

25 La rutina de borrado visualiza la pantalla Delete Customer Information Area hasta que se pulsa la tecla apropiada.

26 Esta instrucción presenta la pantalla Delete Customer Information Area.

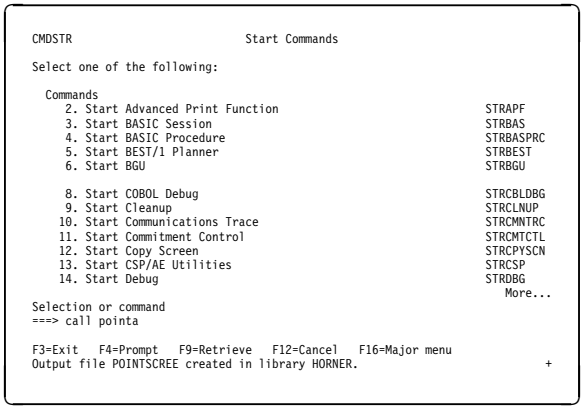
27 Cada registro contiene un puntero del registro de clientes anterior. El registro especial ADDRESS OF señala el registro de clientes actual. Cambiando el registro especial ADDRESS OF, se cambia el registro de clientes actual.

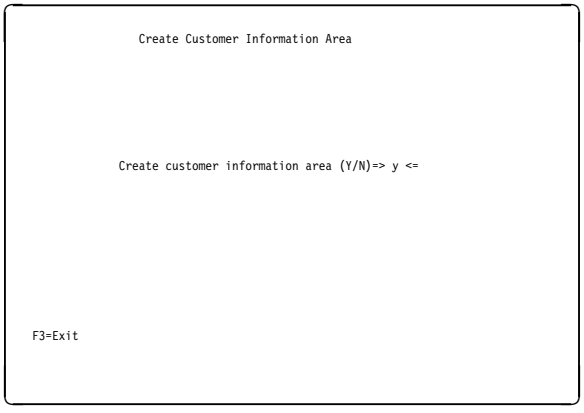
back-screen desplaza el puntero de registro actual hacia el registro anterior de uno en uno 29 , trasladando el puntero del registro de clientes anterior al puntero del registro de clientes actual (ADDRESS OF). Antes de trasladarse hacia el registro anterior de uno en uno, el programa establece el registro de clientes actual al primer registro actualmente visualizado 28 .

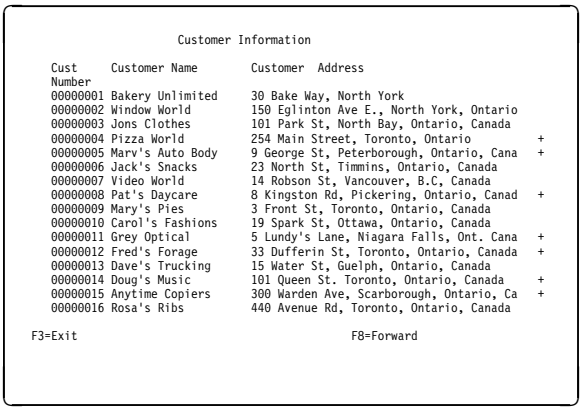
3 *forward-screen* establece *ws-old-space-ptr* (que señala el primer registro de la pantalla) para señalar el registro actual (que se encuentra después del último registro visualizado).

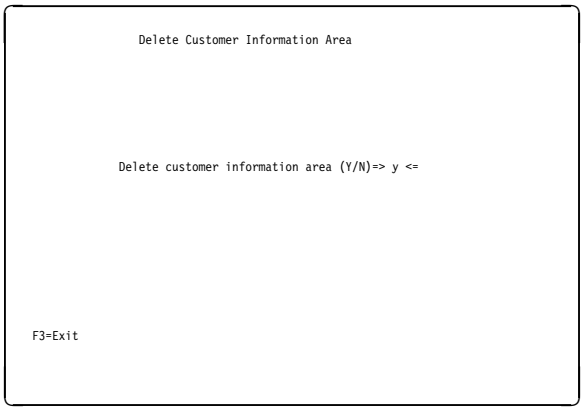
Un espacio de usuario siempre empieza en un límite de 16 bytes, así que el método mostrado aquí asegura que **todos** los registros estén alineados. *ls-cust-rec-length* también se utiliza para encadenar los registros juntos.

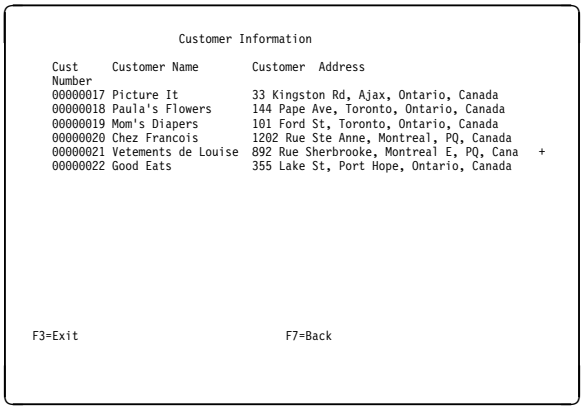
Al ejecutar POINTA, verá las pantallas siguientes:

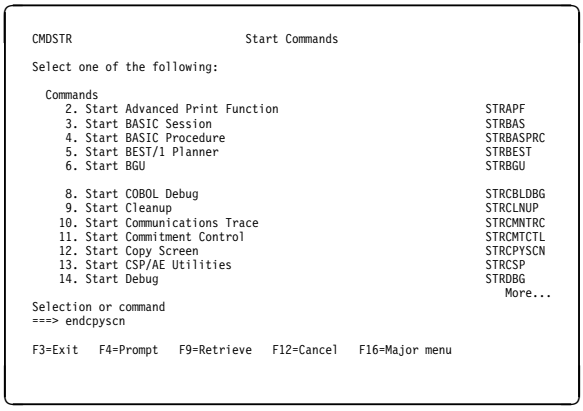


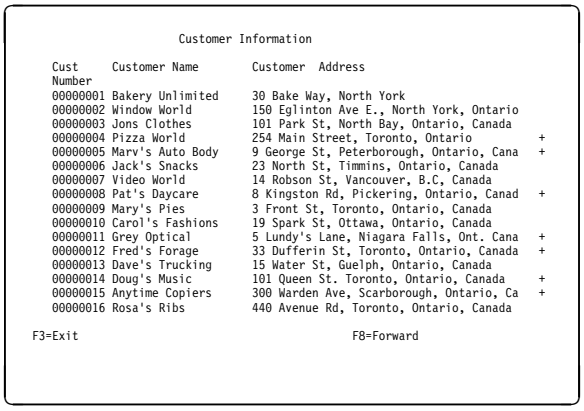












Proceso de una lista encadenada utilizando punteros

Una aplicación típica de la utilización de elementos de datos de puntero es en el proceso de una lista encadenada (una serie de registros donde cada uno señala al siguiente).

Para este ejemplo, imagine una lista de datos encadenada que se componga de registros de salarios individuales. La Figura 64 en la página 256 muestra una forma de visualizar como están enlazados estos registros en el almacenamiento:

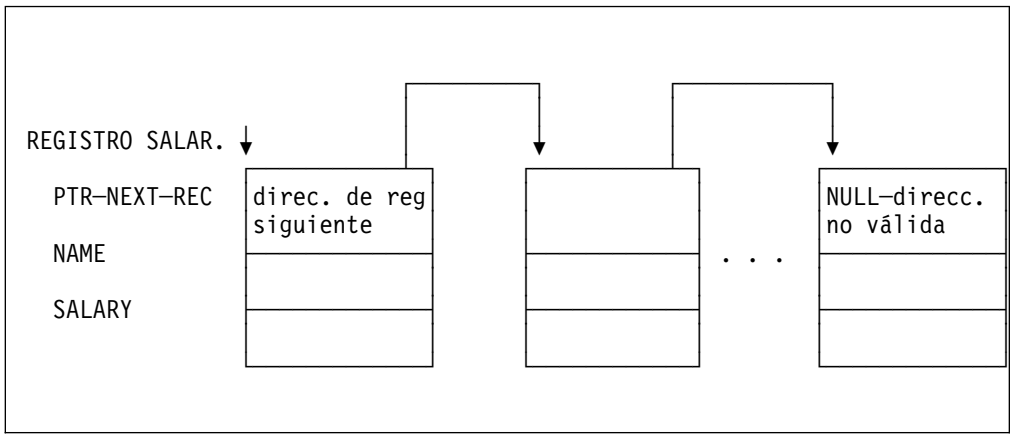


Figura 64. Representación de una lista encadenada que finaliza con NULL

El primer elemento de cada registro (excepto el último) señala al registro siguiente. El primer elemento del último registro, para indicar que es el último registro, contiene un valor nulo en lugar de una dirección.

La lógica de alto nivel de una aplicación que procese estos registros tiene un aspecto similar a este:

```
OBTAIN ADDRESS OF FIRST RECORD IN CHAINED LIST FROM ROUTINE
CHECK FOR END OF THE CHAINED LIST
DO UNTIL END OF THE CHAINED LIST
   PROCESS RECORD
   GO ON TO THE NEXT RECORD
END
```

La Figura 65 en la página 257 contiene un perfil del programa de proceso, CHAINLST, utilizado en este ejemplo de proceso de una lista encadenada.

```
5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400      TESTLIB/CHAINLST        AS400SYS   96/07/04 11:34:45        Página     2

                                 F u e n t e

 INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. CHAINLST.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 DATA DIVISION.
          000500*
    5     000600 WORKING-STORAGE SECTION.
    6     000700 77  PTR-FIRST         POINTER  VALUE IS NULL.                                  96/11/08
    7     000800 77  DEPT-TOTAL        PIC 9(4) VALUE IS 0.                                     96/11/08
          000900*
    8     001000 LINKAGE SECTION.
    9     001100 01  SALARY-REC.
   10     001200   05  PTR-NEXT-REC    POINTER.                                                 96/11/08
   11     001300   05  NAME            PIC X(20).                                               96/11/08
   12     001400   05  DEPT            PIC 9(4).                                                96/11/08
   13     001500   05  SALARY          PIC 9(6).                                                96/11/08
   14     001600 01  DEPT-X            PIC 9(4).                                                96/11/08
          001700*
   15     001800 PROCEDURE DIVISION USING DEPT-X.
          001900 CHAINLST-PROGRAM SECTION.                                                       96/11/08
          002000 MAINLINE.                                                                      96/11/08
          002100*
          002200* PARA TODO EL PERSONAL DEL DEPARTAMENTO RECIBIDO COMO DEPT-X,
          002300* PASAR POR TODOS LOS REGISTROS DE LA LISTA ENCADENADA EN BASE
          002400* A LA DIRECCIÓN OBTENIDA DE "CHAINANC" Y ACUMULAR LOS                          96/11/08
          002500* SALARIOS.
          002600* EN CADA REGISTRO, PTR-NEXT-REC ES UN PUNTERO AL
          002700* REGISTRO SIGUIENTE DE LA LISTA. EN EL ÚLTIMO REGISTRO,
          002800* PTR-NEXT-REC ES NULO. VISUALIZAR EL TOTAL.
          002900*
   16     003000     CALL "CHAINANC" USING PTR-FIRST                                            96/11/08
   17     003100     SET ADDRESS OF SALARY-REC TO PTR-FIRST
          003200*
   18     003300     PERFORM WITH TEST BEFORE UNTIL ADDRESS OF SALARY-REC = NULL
   19     003400        IF DEPT = DEPT-X THEN
   20     003500           ADD SALARY TO DEPT-TOTAL
          003600        END-IF
   21     003700        SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
          003800     END-PERFORM
          003900*
   22     004000     DISPLAY DEPT-TOTAL
   23     004100     GOBACK.

                      * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 65. Programa ILE COBOL/400 para procesar una lista encadenada

Transferencia de punteros entre programas y procedimientos

Para obtener la dirección de la primera área de registro SALARY-REC, el programa CHAINLST llama al programa CHAINANC:

```
CALL "CHAINANC" USING PTR-FIRST
```

PTR-FIRST se define en la WORKING-STORAGE en el programa de llamada (CHAINLST) como un elemento de datos de puntero:

```
WORKING-STORAGE SECTION.
77  PTR-FIRST         POINTER  VALUE IS NULL.
```

Cuando vuelve de la llamada a CHAINANC, PTR-FIRST contiene la dirección del primer registro de la lista encadenada.

PTR-FIRST se define inicialmente con un valor nulo como comprobación lógica. Si se produce un error con la llamada y PTR-FIRST nunca recibe el valor de la direc-

ción del primer registro de la cadena, un valor nulo permanecerá en PTR-FIRST y, según la lógica del programa, el registro no se procesará.

NULL es una constante figurativa utilizada para asignar el valor de una dirección no válida a elementos de puntero. Puede utilizarse en la cláusula VALUE IS NULL, en la instrucción SET y como un operando en una condición de relación con un puntero.

La Linkage Section del programa de llamada contiene la descripción de los registros de la lista encadenada. También contiene la descripción del código de departamento transferido a través de la expresión USING de la instrucción CALL.

```
LINKAGE SECTION.
01  SALARY-REC.
  05  PTR-NEXT-REC    POINTER.
  05  NAME            PIC X(20).
  05  DEPT            PIC 9(4).
  05  SALARY          PIC 9(6).
01  DEPT-X            PIC 9(4).
```

Para basar la descripción de registro SALARY-REC en la dirección contenida en PTR-FIRST, utilice una instrucción SET:

```
CALL "CHAINANC" USING PTR-FIRST
SET ADDRESS OF SALARY-REC TO PTR-FIRST
```

Comprobación del final de la lista encadenada

La lista encadenada de este ejemplo está configurada de manera que el último registro contenga una dirección no válida. Para hacerlo, al elemento de datos de puntero del último registro se le asigna el valor NULL.

A un elemento de datos de puntero se le puede asignar el valor NULL de tres formas:

- Un elemento de datos de puntero puede definirse con una cláusula VALUE IS NULL en su definición de datos.
- NULL puede ser el campo de envío en una instrucción SET.
- El valor inicial de un elemento de datos de puntero, con o sin una cláusula VALUE de NULL, es igual a NULL.

En el caso de una lista encadenada en la que el puntero del último registro contenga un valor nulo, el código para comprobar el final de la lista sería:

```
IF PTR-NEXT-REC = NULL
           ⋮

(lógica para final de cadena)
```

Si no ha llegado al final de la lista, procese el registro y vaya al siguiente registro.

En el programa CHAINLST, esta prueba para el final de la lista encadenada se consigue con una estructura “hacer mientras”:

```
PERFORM WITH TEST BEFORE UNTIL ADDRESS OF SALARY-REC = NULL
 IF DEPT = DEPT-X
   THEN ADD SALARY TO DEPT-TOTAL
   ELSE CONTINUE
 END-IF
SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
END-PERFORM
```

Proceso del registro siguiente

Para trasladarse al registro siguiente, establezca la dirección del registro en la Linkage Section para que sea igual a la dirección del registro siguiente. Esto se consigue mediante el elemento de datos de puntero que se encuentra en el primer campo en SALARY-REC:

```
SET ADDRESS OF SALARY-REC TO PTR-NEXT-REC
```

Luego repita la rutina de proceso de registros, que procesará el registro siguiente de la lista encadenada.

Aumento de las direcciones recibidas desde otro programa

Los datos transferidos desde un programa de llamada pueden contener información de cabecera que desee ignorar (por ejemplo, en datos recibidos desde una aplicación CICS no migrada al nivel de mandatos).

Debido a que los elementos de datos de puntero no son numéricos, no puede realizar operaciones aritmética sobre ellos. Sin embargo, puede utilizar el verbo SET para aumentar la dirección transferida con el fin de ignorar la información de cabecera.

Puede configurar la Linkage Section de la siguiente forma:

```
LINKAGE SECTION.
01  RECORD-A.
  05  HEADER          PIC X(16).
  05  REAL-SALARY-REC PIC X(30).
        ⋮
01  SALARY-REC.
  05  PTR-NEXT-REC    POINTER.
  05  NAME            PIC X(20).
  05  DEPT            PIC 9(4).
  05  SALARY          PIC 9(6).
```

Dentro de la Procedure division, base la dirección de SALARY-REC en la dirección de REAL-SALARY-REC:

```
SET ADDRESS OF SALARY-REC TO ADDRESS OF REAL-SALARY-REC
```

SALARY-REC se basa ahora en la dirección de RECORD-A + 16.

Transferencia de las direcciones de punto de entrada con punteros de procedimientos

Puede utilizar elementos de datos de puntero de procedimientos, definidos con la cláusula USAGE IS PROCEDURE-POINTER, para transferir la dirección de entrada de un programa en un formato requerido por ciertos servicios invocables ILE.

Por ejemplo, para que una rutina de manejo de errores escritos por usuario tome el control cuando se produce una condición de excepción durante la ejecución de un programa, primero debe transferir la dirección de entrada de un procedimiento ILE, como por ejemplo un programa ILE COBOL/400 principal, a CEEHDLR, un servicio invocable ILE de gestión de condiciones para registrarlo.

Los elementos de datos de puntero de procedimientos pueden establecerse para contener la dirección de entrada para los tipos de programas siguientes:

- Un primer programa ILE COBOL/400
- Un procedimiento ILE escrito en otro lenguaje ILE.
- Un objeto de programa ILE o un objeto de programa OPM.

Nota: Un elemento de datos de puntero de procedimientos no puede establecerse en la dirección de un programa ILE COBOL/400 anidado.

Un elemento de datos de puntero de procedimientos sólo puede establecerse utilizando el Formato 6 de la instrucción SET.

Para obtener una definición completa de la cláusula USAGE IS PROCEDURE-POINTER y de la instrucción SET, consulte *ILE COBOL/400 Reference*.

Capítulo 12. Manejo de errores y excepciones ILE COBOL/400

ILE COBOL/400 contiene elementos especiales para ayudarle a anticiparse y corregir condiciones de error que pueden producirse cuando se ejecuta un programa. Incluso si el código no tiene defectos, los errores pueden producirse en los recursos del sistema que el programa utiliza.

Puede anticiparse a posibles condiciones de error colocando código en el programa para que los maneje.Si el programa no tiene código de manejo de errores, podría comportarse de forma imprevisible, los archivos se dañarían y podría producir resultados incorrectos.Sin el código de manejo de errores, puede que incluso ni se dé cuenta de que existe un problema.

La acción tomada por el código de manejo de errores puede variar desde intentar hacer frente a la situación y continuar, a emitir un mensaje o detener el programa.Como mínimo, codificar un mensaje de error para identificar la condición de un error es una buena idea.

Al ejecutar un programa ILE COBOL/400, pueden producirse varios tipos de error. La instrucción ILE COBOL/400 activa en el momento de un error determinado provoca que ciertas cláusulas o expresiones ILE COBOL/400 se ejecuten.

Este capítulo trata cómo:

- utilizar las API de enlace de manejo de errores
- iniciar vuelcos intencionados
- manejar errores en operaciones de serie
- manejar errores en operaciones aritméticas
- manejar errores en operaciones de entrada-salida
- manejar errores en operaciones de ordenar/fusionar
- manejar excepciones en la instrucción CALL
- crear rutinas de manejo de errores escritas por usuarios.

Manejo de condiciones ILE

En el sistema AS/400, existen varias formas en que los programas pueden comunicar el estado a otro programa.Uno de los principales métodos es enviar un mensaje OS/400.

Existen varios tipos de mensajes OS/400. Estos incluyen de consulta, de información, de conclusión, de escape y de notificación. Por ejemplo, el mensaje final enviado por el compilador ILE COBOL/400 cuando una compilación es satisfactoria es LNC0901,

```
Se ha creado el programa nombre-de-programa en la biblioteca
nombre-de-biblioteca en fecha a las
hora.
```

El mensaje LNC0901 es un mensaje de conclusión. Si una compilación falla, recibirá el mensaje LNC9001,

```
Ha fallado la compilación. No se ha creado nombre-de-programa.
```

El mensaje LNC9001 es un mensaje de escape.

Una condición ILE y un mensaje OS/400 son bastante similares. Cualquier mensaje de escape, estado, notificación o comprobación de función es una condición y cada condición ILE tiene un mensaje OS/400 asociado.

Como los mensajes OS/400, que pueden manejarse declarando y habilitando un supervisor de mensajes, una condición ILE puede manejarse registrando un **manejador de condiciones ILE**. Un manejador de condiciones ILE le permite registrar un procedimiento de manejo de excepciones en tiempo de ejecución al que se da el control cuando se produce una excepción. Para registrar un manejador de excepciones, utilice la API de enlace Registrar un manejador de condiciones escritas por usuario (CEEHDLR).

Cuando se llama a un objeto de programa o a un procedimiento ILE, se crea una nueva entrada de pila de llamadas.Hay una cola de mensajes de llamada asociada a cada entrada en la pila de llamadas.Esta cola de mensajes de llamada es una cola de mensajes de programas, si se llama a un objeto de programa o, una cola de mensajes de procedimientos, si se llama a un procedimiento ILE. En ILE, se puede enviar un mensaje a un objeto de programa o a un procedimiento ILE, enviando un mensaje a su entrada de pila de llamadas.De forma similar, puede señalar una condición de un objeto de programa o procedimiento ILE, señalando una condición en su entrada de pila de llamadas.Puede señalar una condición a un objeto de programa utilizando las API de enlace ILE. Consulte la sección sobre las API de enlace ILE en *ILE Conceptos* para obtener una lista de las API de enlace de gestión de condiciones.

Cada entrada de la pila de llamadas tiene varios manejadores de condiciones ILE registrados.Cuando hay múltiples manejadores de condiciones ILE registrados para la misma entrada de la pila de llamadas, el sistema llama a estos manejadores en orden último en entrar, primero en salir (LIFO).Estos manejadores de condiciones ILE también pueden registrarse con distintos niveles de prioridad.Sólo unas cuantas de estas prioridades están disponibles para ILE COBOL/400.Aproximadamente, existen diez prioridades distintas entre el rango 85 a 225. Los manejadores de condiciones ILE se llaman en orden de prioridad ascendente.

En ILE, si una condición de excepción no se maneja en una entrada de la pila de llamada concreta, el mensaje de excepción no manejado se filtra en la cola de mensajes de entrada de la pila de llamadas anterior. Cuando esto sucede, el proceso de excepción continúa en la entrada de la pila de llamadas anterior.El filtrado de una condición de excepción no manejada continúa hasta que se llega al límite de control o hasta que se maneja el mensaje de excepción.Un mensaje de excepción no manejado se convierte en una comprobación de función cuando se filtra en el límite de control.

El mensaje de excepción de comprobación de función puede entonces ser manejado por la entrada de pila de llamadas que emitió la condición de excepción original o se filtra hacia el límite de control.Si se maneja la comprobación de función, el proceso normal continúa y el proceso de excepción finaliza. Si la comprobación de función se filtra hacia el límite de control, ILE considera que la aplicación ha finalizado con un error no esperado. El mensaje de excepción de anomalía genérica, CEE9901, es enviado por ILE al llamador del límite de control.

Cuando se produce una condición de excepción en un objeto de programa o en un procedimiento ILE, primero la maneja el manejador de condición ILE registrado

para la entrada de la pila de llamadas del objeto de programa o del procedimiento ILE. Si no existe ningún manejador de condición ILE registrado para la entrada de la pila de llamadas, entonces la condición de excepción la manejan los manejadores de errores específicos de HLL. Los manejadores de errores específicos de HLL son funciones del lenguaje definidas para manejar errores. El manejo de errores específicos de HLL en ILE COBOL/400 incluye la instrucción declarativa USE para manejar errores de E/S y las imperativas en las expresiones condicionales en el ámbito de instrucciones, como por ejemplo ON SIZE ERROR e INVALID KEY. Si la condición de excepción no la maneja el manejador de errores específicos de HLL, entonces la condición no manejada se filtra hacia la cola de mensajes de entrada de la pila de llamadas anterior, tal como se describe anteriormente.

Para obtener más información sobre el manejo de condiciones ILE, consulte los apartados sobre manejo de errores y gestión de excepciones y condiciones en el manual *ILE Conceptos*.

Finalización de un programa ILE COBOL/400

Un programa ILE COBOL/400 puede finalizarse realizando las acciones siguientes:

- Una instrucción ILE COBOL/400 (EXIT PROGRAM, STOP RUN o GOBACK)
- Una respuesta a un mensaje de consulta
- Una instrucción STOP RUN o EXIT PROGRAM implícita
- Otro equivalente del lenguaje ILE de la instrucción STOP RUN de ILE COBOL/400. Por ejemplo la función `exit()` de ILE C/400.
- Otro equivalente del lenguaje ILE de la instrucción anormal STOP RUN de ILE COBOL/400. Por ejemplo, la función `abort()` ILE C/400.
- Un mensaje de escape enviado después del programa de llamada ILE COBOL/400 por el procedimiento ILE o el objeto de programa llamado
- La finalización, por parte del procedimiento ILE o del objeto de programa llamado, del grupo de activación en el cual se ejecuta el programa de llamada ILE COBOL/400.

Existe una instrucción STOP RUN implícita cuando un programa principal ILE COBOL/400 no tiene ninguna instrucción ejecutable a continuación (EXIT PROGRAM implícito para un subprograma ILE COBOL/400), es decir, cuando el proceso se desactiva mediante la última instrucción de un programa.

Los mensajes de consulta pueden emitirse en respuesta a una instrucción ILE COBOL/400 (es decir, un literal STOP), pero generalmente se emiten cuando se produce un error grave en un programa o cuando una operación ILE COBOL/400 no finaliza satisfactoriamente. (LNR7205, LNR7207 y LNR7208 son ejemplos de ello). Los mensajes de consulta le permiten determinar qué acción tomar cuando se produce un error de excepción.

Existen cuatro respuestas comunes a un mensaje de consulta COBOL: C, D, F, y G (cancelar, cancelar y volcar, cancelar y volcar totalmente, continuar). Las tres primeras provocan (como sus pasos finales) una instrucción STOP RUN implícita anormal.

Una instrucción STOP RUN implícita o explícita o una instrucción GOBACK en el programa principal ILE COBOL/400, provocan la señalización de la condición de finalización inminente en el límite de control más cercano. La condición de finalización inminente puede manejarse de dos formas:

- a través de un manejador de errores registrado antes de que alcance el límite de control o

  Nota: Para registrar un manejador de excepciones, utilice la API de enlace Registrar un manejador de condiciones escritas por usuario (CEEHDLR). Consulte la publicación *ILE Conceptos* para obtener más información sobre los manejadores de excepción.

- si ha alcanzado el límite de control, entonces todos los programas situados detrás del límite de control finalizarán y el control volverá al programa situado antes del límite de control.

Si este límite de control es un límite de control fijo, entonces el grupo de activación (unidad de ejecución) finalizará.

Si la instrucción STOP RUN es anormal y alcanza un límite de control fijo, el programa situado antes del límite de control recibirá el mensaje de escape CEE9901.

Utilización de Interfaces de programación de aplicaciones (API) de enlace de manejo de errores

Existen dos niveles en los cuales los errores pueden manejarse en ILE COBOL/400. Primero, los manejadores de condiciones registrados en cada nivel de prioridad tienen la ocasión de manejar la condición. Si la condición permanece sin manejar cuando se alcanza el límite de control, se envía una condición de comprobación de función.Cada procedimiento ILE COBOL/400 tiene un manejador de condición ILE, registrado con el nivel de prioridad 205, para manejar una comprobación de función. Este manejador de condiciones de comprobación de función emitirá un mensaje de consulta COBOL, a no ser que la manejen las API de enlace siguientes:

- Recuperar manejador de errores COBOL (QlnRtvCobolErrorHandler)

  La API Recuperar manejador de errores COBOL (QlnRtvCobolErrorHandler) le permite recuperar el nombre del procedimiento de manejo de errores ILE COBOL/400 actual para el grupo de activación desde el que se llama a la API.

- Establecer manejador de errores COBOL (QlnSetCobolErrorHandler)

  La API Establecer manejador de errores COBOL (QlnSetCobolErrorHandler) le permite especificar la identidad de un procedimiento de manejo de errores ILE COBOL/400 para el grupo de activación desde el cual se llama a la API.

Estas API sólo afectan al manejo de excepciones dentro de programas ILE COBOL/400. Para obtener información detallada sobre todas estas API, consulte el apartado sobre API de COBOL en el manual *System API Reference*.

Nota: El valor *NOMONOPRC debe especificarse en el parámetro OPTION de los mandatos CRTCBLMOD o CRTBNDCBL para utilizar estas API.

Inicialización de vuelcos intencionados

Puede utilizar la API de enlace Vuelco de COBOL (QlnDumpCobol) para provocar intencionadamente un vuelco con formato de un programa ILE COBOL/400. La API QlnDumpCobol acepta seis parámetros, que definen:

- el nombre del objeto de programa
- el nombre de la biblioteca
- el nombre del objeto de módulo
- el tipo de objeto de programa
- el tipo de vuelco
- el código de error

A continuación encontrará algunos ejemplos de como llamar a la API QlnDumpCobol API y las operaciones resultantes:

```
WORKING-STORAGE SECTION.
01  ERROR-PARMS.
    05  BYTES-PROVIDED     PIC S9(6)  BINARY VALUE ZERO.
    05  BYTES-AVAILABLE    PIC S9(6)  BINARY VALUE ZERO.
    05  EXCEPTION-ID       PIC X(7.
    05  RESERVED-X         PIC X.
    05  EXCEPTION-DATA     PIC X(64).
01  PROGRAM-NAME           PIC X(10).
01  LIBRARY-NAME           PIC X(10).
01  MODULE-NAME            PIC X(10).
01  PROGRAM-TYPE           PIC X(10).
01  DUMP-TYPE              PIC X.

PROCEDURE DIVISION.
    MOVE LENGTH OF ERROR-PARMS TO BYTES-PROVIDED.
    MOVE "MYPROGRAM"           TO PROGRAM-NAME.
    MOVE "TESTLIB"             TO LIBRARY-NAME.
    MOVE "MYMOD1"              TO MODULE-NAME.
    MOVE "*PGM"                TO PROGRAM-TYPE.
    MOVE "D"                   TO DUMP-TYPE.
    CALL "QlnDumpCobol"  USING PROGRAM-NAME, LIBRARY-NAME,
                               MODULE-NAME, PROGRAM-TYPE,
                               DUMP-TYPE, ERROR-CODE.
```

Esto proporcionaría un vuelco con formato de identificadores COBOL (opción D) para el objeto de módulo MYMOD1 en el objeto de programa MYPROGRAM de la biblioteca TESTLIB.

```
WORKING-STORAGE SECTION.
01  ERROR-PARMS.
    05  BYTES-PROVIDED     PIC S9(6)  BINARY VALUE ZERO.
    05  BYTES-AVAILABLE    PIC S9(6)  BINARY VALUE ZERO.
    05  EXCEPTION-ID       PIC X(7.
    05  RESERVED-X         PIC X.
    05  EXCEPTION-DATA     PIC X(64).
01  PROGRAM-NAME           PIC X(10).
01  LIBRARY-NAME           PIC X(10).
01  MODULE-NAME            PIC X(10).
01  PROGRAM-TYPE           PIC X(10).
01  DUMP-TYPE              PIC X.

PROCEDURE DIVISION.
    MOVE LENGTH OF ERROR-PARMS TO BYTES-PROVIDED.
    MOVE "*SRVPGM"             TO PROGRAM-TYPE.
    MOVE "F"                   TO DUMP-TYPE.
    CALL "QlnDumpCobol"  USING OMITTED, OMITTED,
                               OMITTED, PROGRAM-TYPE,
                               DUMP-TYPE, ERROR-CODE.
```

Esto provocaría un vuelco con formato de identificadores COBOL y de información relacionada con archivos (opción F) para el programa de servicio que llamó a la API QlnDumpCobol.

Si alguno de los parámetros de entrada a la API QlnDumpCobol contiene datos no válidos, el vuelco no se realiza y se genera un mensaje de error, o se devuelven los datos de excepción. Se genera un mensaje de error si el campo BYTES-PROVIDED contiene el valor cero.Si el campo BYTES-PROVIDED contiene un valor distinto de cero, entonces se devuelven los datos de excepción del parámetro ERROR-CODE y no se genera ningún mensaje de error.

Para obtener información detallada sobre la API QInDumpCobol, consulte el apartado sobre API de COBOL en el manual *System API Reference*.

Manejo de errores en operaciones de serie

Al concatener series o descomponer una serie en subserie puede producirse un error. Las instrucciones STRING y UNSTRING proporcionan una expresión ON OVERFLOW para manejar las condiciones típicas de errores de desbordamiento de serie. Para la instrucción STRING, la expresión ON OVERFLOW se ejecutará cuando el valor del puntero implícito o explícito sea:

- inferior a 1
- superior a la longitud del campo de recepción.

Para la instrucción UNSTRING, se ejecutará la expresión ON OVERFLOW cuando:

- el valor implícito o explícito del puntero sea inferior a 1
- el valor implícito o explícito del puntero sea mayor que la longitud del campo de envío
- se haya actuado en todos los campos de recepción y el campo de envío aún contenga caracteres sin examinar.

Cualquier otra condición no manejada por la expresión ON OVERFLOW provocará generalmente mensajes MCH. Dichos mensajes los maneja normalmente el manejador de condiciones de comprobación de función. Para evitar que se llame al

manejador de condiciones de comprobación de función, puede registrar su propio manejador de condiciones utilizando la API CEEHDLR para captar los mensajes MCH.

Utilice la expresión ON OVERFLOW de la instrucción STRING o UNSTRING para identificar los pasos de manejo de errores que desee realizar cuando se produzca una condición de desbordamiento. Si no tiene una cláusula ON OVERFLOW en la instrucción STRING o UNSTRING, el control se transferirá a la siguiente instrucción secuencial y no se le notificará la operación incompleta.

Consulte las instrucciones STRING y UNSTRING en la publicación *ILE COBOL/400 Reference* para obtener más información sobre la expresión ON OVERFLOW.

Manejo de errores en operaciones aritméticas

Las operaciones aritméticas pueden provocar ciertos errores típicos. Estos errores típicos generalmente dan como resultado mensajes MCH.

La expresión ON SIZE ERROR

La expresión ON SIZE ERROR de las instrucciones ADD, SUBTRACT, MULTIPLY, DIVIDE y COMPUTE:

- permitirá emitir mensajes de desbordamiento binarios y decimales. El mensaje de desbordamiento binario y decimal es MCH1210. El mensaje de división decimal por cero es MCH1211.
- registrará un manejador de condiciones para captar los mensajes de desbordamiento binarios, decimales y de coma flotante, así como otros mensajes MCH aritméticos. Los mensajes de desbordamiento de coma flotante son el MCH1206 (desbordamiento) y el MCH1207 (subdesbordamiento).

  A diferencia de lo que ocurre con los mensajes de desbordamiento binarios y decimales, no se habilita el desbordamiento de coma flotante con la expresión ON SIZE ERROR. El desbordamiento de coma flotante se habilita o inhabilita a nivel de trabajo. Por omisión, siempre se emiten mensajes de desbordamiento de coma flotante. Por este motivo, ILE COBOL/400 hará caso omiso de estos mensajes a menos que se haya codificado la expresión ON SIZE ERROR. Para habilitar o inhabilitar el desbordamiento de coma flotante, consulte el apartado “Manejo de errores en cálculos de coma flotante” en la página 268.

ILE COBOL/400 registra el manejador de condiciones mencionado anteriormente con el nivel de prioridad 85.Un manejador de condiciones de usuario, registrado con el nivel de prioridad 165, sólo recibirá el control si el manejador de condiciones mencionado anteriormente no maneja la excepción.

Si no se codifica la expresión ON SIZE ERROR, no se emitirán los mensajes de desbordamiento binarios y decimales y se hará caso omiso de los mensajes de desbordamiento de coma flotante. El manejador de condiciones de comprobación de función manejará todos los demás mensajes MCH aritméticos de forma normal, a menos que se haya registrado un manejador de condiciones de usuario utilizando la API CEEHDLR.

En las situaciones siguientes se produce una condición de error de tamaño:

- El resultado de la operación aritmética es superior al campo de punto fijo que tiene que contenerlo.
- División por cero
- Cero elevado a la potencia de cero
- Cero elevado a un número negativo
- Un número negativo elevado a una potencia fraccionada.
- Desbordamiento y subdesbordamiento de coma flotante

Durante operaciones aritméticas, los errores más comunes que se producen son errores de tamaño (MCH1210) y errores de datos decimales (MCH1202). ILE COBOL/400 no es capaz de detectar la mayoría de estos errores, sino que éstos son detectados por el sistema operativa y dan lugar a mensajes del sistema. Entonces ILE COBOL/400 supervisa estos mensajes, estableciendo bits internos que determinan si se debe ejecutar una instrucción imperativa SIZE ERROR o emitir un mensaje de ejecución (LNR7200) para finalizar el programa.

Para impedir el envío del mensaje LNR7200, puede registrarse un manejador de condiciones de usuario utilizando la API CEEHDLR para manejar los mensajes MCH, o puede codificarse un manejador de errores ILE COBOL/400 utilizando las API de enlace COBOL para manejar los mensajes de consulta LNR72xx.

ILE COBOL/400 detecta errores que son el resultado de una división por cero durante una operación aritmética. Si ILE COBOL/400 los detecta, estos errores hacen que se ejecute la instrucción SIZE ERROR imperativa.

El mensaje del sistema MCH1210 suele producirse cuando se mueve un campo binario o decimal a otro y el receptor es demasiado pequeño. ILE COBOL/400 supervisa este error, que también da como resultado la ejecución de la instrucción imperativa SIZE ERROR.

LNR7200 es un mensaje de ejecución que generalmente se emite cuando se produce un error grave no controlado en el programa ILE COBOL/400.

El mensaje de sistema MCH1202 es un ejemplo típico de un error grave no controlado. Este tipo de error da como resultado el mensaje de ejecución ILE COBOL/400 LNR7200 (o LNR7204 si el error se produce en un programa llamado por un programa ILE COBOL/400). Los mensajes de sistema MCH3601 y MCH0601 son otros ejemplos de errores graves no supervisados.

Manejo de errores en cálculos de coma flotante

OS/400 proporciona un conjunto de instrucciones MI con Atributos computacionales (CA) para recuperar información acerca de operaciones de coma flotante y para modificar el comportamiento de las operaciones de coma flotante. Por ejemplo, la instrucción MI SETCA (Establecer atributos computacionales) puede evitar que se produzcan ciertas excepciones de coma flotante además de indicar si se realiza o no se realiza el redondeo. Por omisión, el resultado de una operación de coma flotante se redondea *siempre* y se señalan todas las excepciones, salvo el Operando no válido.

Las excepciones que la coma flotante puede evitar son las siguientes:

1. Desbordamiento
2. Subdesbordamiento

| 3. División por cero
| 4. Resultado inexacto
| 5. Operando no válido

| Para el manejo de la expresión ON SIZE ERROR, ILE COBOL/400 requiere que se
| señalen las 3 primeras excepciones.

| ILE COBOL también requiere que se lleve a cabo el redondeo a la posición
| decimal más cercana, lo que significa que se ha utilizado las instrucciones MI CA
| para impedir el redondeo, se eliminarán los dígitos adicionales con lo que el resul-
| tado sería inexacto.

Manejo de errores en operaciones de entrada-salida

El manejo de errores le ayuda durante el proceso de instrucciones de entrada-salida captando errores severos que, de lo contrario, pasarían inadvertidos.Para las operaciones de entrada-salida, existen varias expresiones y cláusulas de manejo de errores importantes. Estas son las siguientes:

- la expresión AT END
- la expresión INVALID KEY
- la expresión NO DATA
- el procedimiento declarativo USE AFTER EXCEPTION/ERROR
- la cláusula FILE STATUS.

Durante las operaciones de entrada-salida, el sistema detecta errores y envía mensajes; estos mensajes los supervisa ILE COBOL/400. Además, ILE COBOL/400 detectará algunos de los errores durante una operación de entrada-salida sin el soporte del sistema. Con independencia de la forma en que se detecte un error durante una operación de entrada-salida, el resultado siempre será un estado de archivo interno distinto de cero, un mensaje de ejecución o ambas cosas. Una característica importante del manejo de errores es la emisión de un mensaje de ejecución cuando se produce un error durante el proceso de una instrucción de entrada-salida, si no existe la expresión AT END/INVALID KEY en la instrucción de entrada-salida, el procedimiento USE AFTER EXCEPTION/ERROR o la cláusula FILE STATUS en la instrucción SELECT para el archivo.

Un punto a recordar sobre los errores de entrada-salida es que el usuario elige si el programa debe seguir ejecutándose o no después de que se produzca un error de entrada-salida grave.ILE COBOL/400 no realiza una acción correctora.Si elige que el programa continúe (incorporando código de manejo de errores en su diseño), también deberá codificar el procedimiento apropiado de recuperación de errores.

Junto a las expresiones y cláusulas de manejo de errores relacionadas específicamente con las instrucciones de entrada-salida, los manejadores de condiciones ILE definidos por usuario y las API de manejo de errores ILE COBOL/400 también pueden utilizarse para manejar errores de E/S.

Para cada instrucción de E/S, ILE COBOL/400 registra un manejador de condiciones para obtener las distintas condiciones relacionadas de E/S.Estos manejadores de condiciones se registran con el nivel de prioridad 185, lo que permite a los manejadores de condiciones definidos por el usuario recibir primero el control.

Tal como se ha mencionado anteriormente, se emite un mensaje de ejecución ILE COBOL/400 cuando se produce un error y no existe ninguna expresión AT END, INVALID KEY, ningún procedimiento USE ni ninguna cláusula FILE STATUS para un archivo. El mensaje, LNR7057, es un mensaje de escape. Este mensaje puede manejarlo un manejador de condiciones definido por el usuario.Si ningún manejador de condiciones puede manejar este mensaje, se volverá a enviar el mensaje LNR7057 como una comprobación de función.

ILE COBOL/400 tiene un manejador de condiciones de comprobación de función que al final emitirá un mensaje de consulta LNR7207 a menos que se haya definido una API de manejo de errores ILE COBOL/400.

Proceso de verbos de entrada-salida

El diagrama siguiente muestra cuando se ejecuta el procedimiento USE y las instrucciones imperativas (NOT) AT END, (NOT) INVALID KEY y NO DATA.

El estado de archivo mostrado aquí hace referencia al estado de archivo interno.

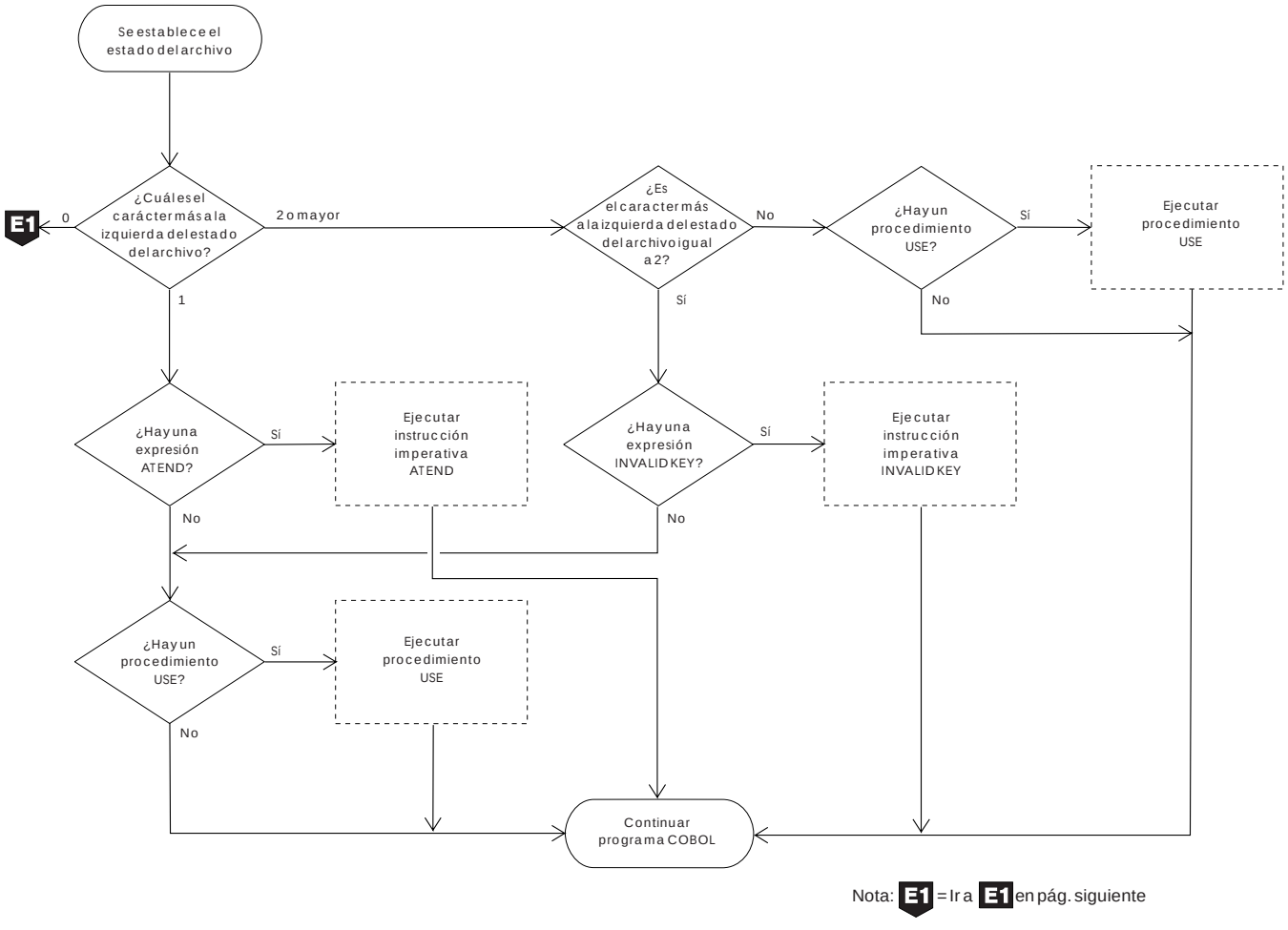


Figura 66 (Parte 1 de 2). Proceso de verbos de E/S

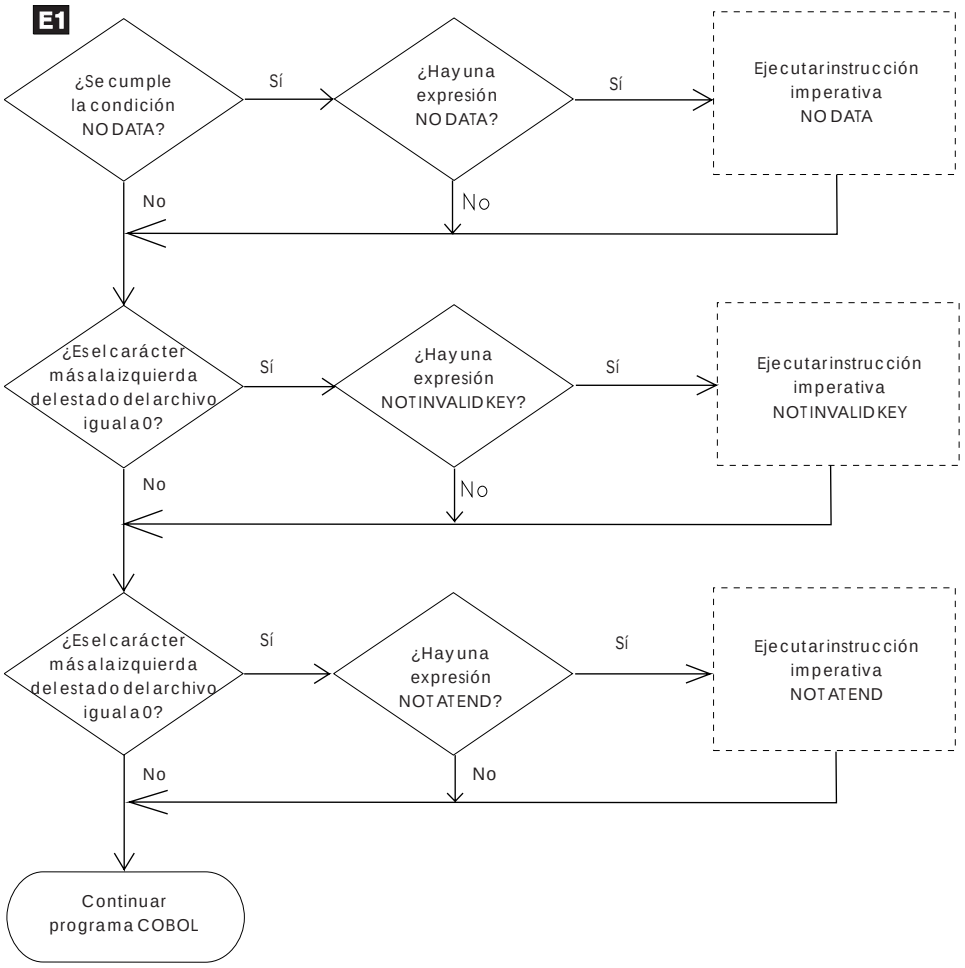


Figura 66 (Parte 2 de 2). Proceso de verbos de E/S

Nota: Siga las partes del diagrama que sean aplicables a sus instrucciones.

Detección de condiciones de fin de archivo (expresión AT END)

Una condición de fin de archivo puede representar un error o no. En muchos diseños, la lectura secuencial hasta el final del archivo se realiza intencionadamente y se espera la condición AT END.

En muchos casos, sin embargo, la condición de fin de archivo reflejará un error.Codifique la expresión AT END de la instrucción READ para manejar cada caso, según el diseño del programa.

Si codifica una expresión AT END, se ejecuta la instrucción imperativa identificada por la expresión cuando se produzca una condición de fin de archivo. Si no codifica la expresión AT END, se ejecutará la declarativa USE AFTER EXCEPTION/ERROR asociada.

Cualquier expresión NOT AT END que codifique se ejecutará sólo si la instrucción READ ha finalizado satisfactoriamente.Si la operación READ falla por culpa de cualquier condición distinta de la de fin de archivo, no se ejecutará la expresión AT END ni NOT AT END.En su lugar, el control pasa al final de la instrucción READ después de ejecutar el procedimiento declarativo USE AFTER EXCEPTION/ERROR asociado.

Si no ha codificado una expresión AT END ni un procedimiento declarativo USE AFTER EXCEPTION/ERROR pero ha codificado la cláusula STATUS KEY para el archivo, el control pasa a la siguiente instrucción secuencial después de la instrucción de entrada-salida que detectó la condición de fin de archivo.En este punto, la codificación debería observar la clave de estado y realizar la acción apropiada para manejar el error.

Detección de condiciones de clave no válida (expresión INVALID KEY)

La instrucción imperativa identificada por la expresión INVALID KEY tomará el control en el caso de que se produzca un error de entrada-salida debido a una clave de índice errónea o a una clave relativa. Puede incluir expresiones INVALID KEY en las instrucciones READ, START, WRITE, REWRITE y DELETE de archivos indexados o relativos.

Las expresiones INVALID KEY se diferencian de las declarativas USE AFTER EXCEPTION/ERROR en las siguientes premisas:

- Las expresiones INVALID KEY funcionan sólo para tipos de errores limitados, mientras que las declarativas USE AFTER EXCEPTION/ERROR abarcan la mayoría de formas de errores.
- Las expresiones INVALID KEY se codifican directamente en el verbo de entrada-salida, mientras que las declarativas USE AFTER EXCEPTION/ERROR se codifican por separado.
- Las expresiones INVALID KEY son específicas para una sóla operación de entrada, mientras que las declarativas USE AFTER EXCEPTION/ERROR son más generales.

Si especifica la expresión INVALID KEY en una instrucción de entrada-salida que causa una condición de clave no válida, el control se transfiere a la instrucción imperativa identificada por la expresión INVALID KEY. En este caso, cualquier declarativa USE AFTER EXCEPTION/ERROR codificada no se ejecutará.

Cualquier expresión NOT INVALID KEY que especifique se ejecuta sólo si la instrucción finaliza satisfactoriamente.Si la operación falla por culpa de cualquier condición distinta a la clave no válida, no se ejecutará la expresión INVALID KEY ni la expresión NOT INVALID KEY. En su lugar, el control pasa al final de la instrucción de entrada-salida después de ejecutar cualquier declarativa USE AFTER EXCEPTION/ERROR asociada.

Utilice la cláusula FILE STATUS junto con la expresión INVALID KEY para evaluar la clave de estado y determine la condición de clave no válida específica.

Por ejemplo, supongamos que tiene un archivo que contiene registros maestros de clientes y necesita actualizar algunos de estos registros con información de un archivo de actualización de transacción.Leerá cada registro de transacción buscará el correspondiente registro en el archivo maestro y realizará las actualizaciones necesarias.Los registros de ambos archivos contienen cada uno un campo para número de cliente y cada registro del archivo maestro tiene un número de cliente exclusivo.

La entrada FILE-CONTROL para el archivo maestro de registros a modificar incluye instrucciones que definen la organización indexada, el acceso aleatorio, MASTER-COMMUTER-NUMBER como la clave de registro y

COMMUTER-FILE-STATUS como la clave de estado de archivos.El ejemplo siguiente muestra cómo puede utilizar la cláusula FILE STATUS junto con la expresión INVALID KEY para determinar de forma más específica la causa de una anomalía de instrucción de entrada-salida.

```
.
. (leer el registro de actualización de transacción)
.
MOVE "TRUE" TO TRANSACTION-MATCH
MOVE UPDATE-COMMUTER-NUMBER TO MASTER-COMMUTER-NUMBER
READ MASTER-COMMUTER-FILE INTO WS-CUSTOMER-RECORD
   INVALID KEY
      DISPLAY "REGISTRO MAESTRO DE CLIENTE NO ENCONTRADO"
      DISPLAY "CÓDIGO FILE STATUS:" COMMUTER-FILE-STATUS
      MOVE "FALSE" TO TRANSACTION-MATCH
END-READ
```

Utilización de procedimientos declarativos EXCEPTION/ERROR (Instrucción USE)

Puede codificar uno o más procedimientos declarativos USE AFTER EXCEPTION/ERROR en el programa ILE COBOL/400 al que se dará el control si se produce un error de entrada-salida.Puede tener:

- procedimientos individuales para cada modalidad de apertura de archivo (tanto INPUT, OUTPUT, I-O o EXTEND)
- procedimientos individuales para cada archivo particular.
- procedimientos individuales para grupos de archivos.

Coloque cada procedimiento en la Declaratives Section de la Procedure División del programa.Consulte la *ILE COBOL/400 Reference* para obtener detalles sobre cómo grabar un procedimiento declarativo.

En el procedimiento, puede escoger si desea intentar una acción correctora, reintentar la operación, continuar o finalizar el programa.Puede utilizar el procedimiento declarativo USE AFTER EXCEPTION/ERROR en combinación con las claves de estado si desea un análisis más profundo del error.

Para archivos GLOBAL, cada programa ILE COBOL/400 puede tener su propio procedimiento declarativo USE AFTER EXCEPTION/ERROR.

El procedimiento declarativo USE AFTER EXCEPTION/ERROR puede declararse a sí mismo GLOBAL.Se siguen unas normas especiales de prioridad cuando se pueden ejecutar procedimientos declarativos múltiples en un error de E/S.Al aplicar estas normas, sólo se seleccionará el primer procedimiento declarativo calificado para la ejecución.El procedimiento declarativo seleccionado debe satisfacer las normas para la ejecución de dicho procedimiento declarativo.El orden de prioridad para seleccionar un procedimiento declarativo es:

1. Un procedimiento declarativo específico de archivo (uno de la forma USE AFTER ERROR ON *file-name-1*) dentro del programa que contiene la instrucción que causó la condición calificada
2. Un procedimiento declarativo específico de modalidad (uno de la forma USE AFTER ERROR ON INPUT) dentro del programa que contiene la instrucción que causó la condición calificada

3. Un procedimiento declarativo específico de archivo que especifica la expresión GLOBAL y se encuentra dentro del programa que contiene directamente el último programa examinado para la condición calificada
4. Un proceso declarativo específico de modalidad que incluye la expresión GLOBAL y se encuentra dentro del programa que contiene directamente el último programa en el que se examinó la condición calificada.
5. Las normas 3 y 4 se aplican, de forma recursiva, a través de los padres en la jerarquía de programas.

Escriba un procedimiento declarativo USE AFTER EXCEPTION/ERROR si desea devolver el control al programa después de que se produzca un error.Si no escribe dicho procedimiento, el trabajo puede cancelarse o finalizar anormalmente después de que se produzca un error.

Cada procedimiento declarativo USE AFTER EXCEPTION/ERROR se ejecuta como una invocación diferente de las llamadas de otros procedimientos declarativos y de la parte no declarativa del mismo programa ILE COBOL/400. Así, si llama a la API CEEHDLR para registrar un manejador de condiciones ILE desde un procedimiento declarativo, ese manejador de condiciones ILE sólo se llama en excepciones que se producen en el procedimiento declarativo USE AFTER EXCEPTION/ERROR y no en excepciones de cualquier otra parte del programa ILE COBOL/400.

Determinación del tipo de error mediante la clave de estado de archivo

La clave de estado de archivos se actualiza después de cada operación de entrada-salida en un archivo, colocando valores en los dos dígitos de la clave de estado de archivo. En general, un cero en el primer dígito indica una operación satisfactoria y un cero en ambos dígitos significa "nada anómalo sobre lo que informar".

Debe proporcionar una entrada FILE-CONTROL para especificar la organización y el método de acceso para cada archivo utilizado por el programa ILE COBOL/400. También puede codificar una cláusula
FILE STATUS en esta entrada.

La cláusula FILE STATUS designa uno o dos elementos de datos (codificados en la sección WORKING-STORAGE) para guardar una copia del resultado de una operación de E/S. La copia del primero de estos elementos se denomina estado de archivo externo. Si utiliza un archivo TRANSACTION, tiene un registro más del resultado denominado código de retorno externo, que consta de los códigos de retorno principal y secundario externos.

ILE COBOL/400 mantiene la información correspondiente a estos dos elementos de datos en el Descriptor de campos de archivos (FFD) ILE COBOL/400. Las copias ILE COBOL/400 de estos dos elementos de datos se denominan estado de archivo interno y código de retorno interno. En este capítulo, el *estado de archivo* y el *código de retorno* (*principal*/*secundario*) hacen referencia a las copias de ILE COBOL/400, a menos que se especifique lo contrario.

Durante el proceso de una instrucción de E/S, el estado de archivo puede actualizarse de una de las tres formas descritas más abajo. El contenido del estado de archivo determina qué procedimientos de manejo de errores ejecutar.

Los procedimientos de manejo de errores toman el control después de una operación de entrada o salida no satisfactoria, indicado por un estado de archivo distinto de cero. Antes de ejecutar cualquiera de estos procedimientos, el estado de archivos se copia en el estado de archivo externo.

El estado de archivo se establece de una de estas tres formas:

- Método A (todos los archivos):

  ILE COBOL/400 comprueba el contenido de las variables de los bloques de control de archivos Si el contenido no es lo que se espera, se establece un estado de archivo distinto de cero. La mayoría de estados de archivos establecidos de esta forma son el resultado de comprobar el Descriptor de campos de archivo (FFD) ILE COBOL/400 y el Bloque de control de archivos de usuario (UFCB) del sistema.

- Método B (archivos de transacciones):

  ILE COBOL/400 comprueba los códigos de retorno principal y secundario del sistema. Si el código de retorno principal no es cero, el código de retorno (formado por códigos de retorno principal y secundario) se traslada al estado de archivo. Si el código de retorno principal es cero, el estado de archivo se habrá establecido mediante el Método A o C.

  Para operaciones READ, WRITE y REWRITE de subarchivos, sólo son aplicables los Métodos A y C.

  Para obtener una lista de códigos de retorno y sus estados de archivo correspondientes, consulte el apartado “File Structure Support Summary and Status Key Values” en la publicación *ILE COBOL/400 Reference*.

- Método C (todos los archivos):

  El sistema envía un mensaje cuando ILE COBOL/400 llama a la gestión de datos para realizar una operación de E/S. Entonces ILE COBOL/400 supervisa estos mensajes y establece un estado de archivo según corresponda. Cada operación de E/S ILE COBOL/400 la maneja una rutina dentro de un programa de servicio, que se proporciona con el compilador ILE COBOL/400. Esta rutina llama a la gestión de datos para realizar la operación de E/S. En la mayoría de casos, se habilita un único supervisor de mensajes alrededor de esta llamada a la rutina del programa de servicio.

  El supervisor de mensajes para cada operación de E/S maneja excepciones de E/S típicas que dan como resultado mensajes CPF que empiezan con “CPF4” o “CPF5”. El supervisor de mensajes establece el estado de archivo basándose en el mensaje CPF que recibe. Para obtener una lista de mensajes que maneje el supervisor de mensajes, vea “File Structure Support Summary and Status Key Values” en la *ILE COBOL/400 Reference*.

  A través de la utilización de supervisores de mensajes de esta forma, el estado de archivo se establece de forma coherente para cada tipo de operación de E/S, independientemente de qué otros tipos de operaciones de E/S tenga en su programa. Consulte “Manejo de mensajes mediante manejadores de condiciones” en la página 278 para obtener más información sobre supervisores de mensajes.

Cómo se establece el estado del archivo

001

– Iniciar la operación de E/S.
– Restablecer el estado de archivo interno.
– Método A: Comprobar el contenido de las variables de los bloques de control de archivos.

(Comprobar, por ejemplo, que el archivo se ha abierto correctamente.)

¿Están las variables de los bloques de control de archivos establecidas como se esperaba?

Sí **No**

002

– Establecer el estado de archivo interno para indicar que se ha producido un error.
– Continuar con el Paso 006

003

– Llamar a la gestión de datos para realizar la operación de E/S.

¿Devuelve la gestión de datos una excepción?

Sí **No**

004

– Método A: Comprobar el contenido de las variables de los bloques de control de archivos.

¿Están las variables de los bloques de control de archivos establecidas como se esperaba?

Sí **No**

005

– Establecer el estado de archivo interno para indicar que se ha producido un error.
– Continuar con el Paso 006

006

– Trasladar el estado de archivo interno al estado de archivo externo (especificado en la cláusula de estado de archivo).
– Basándose en el estado de archivo interno, ejecutar el código de manejo de errores.

007

(Paso **007** continúa)

007 (continuación)
¿Es el archivo un archivo de transacción?
Sí **No**

008

– Método C: Establecer el estado de archivo interno según el mensaje CPF enviado por la gestión de datos.
– Continuar con el Paso 004 en la página 276

009

¿Están los códigos de retorno principal y secundario disponibles en el sistema?
Sí **No**

010

– Método C: Establecer el estado de archivo interno según el mensaje CPF enviado por la gestión de datos.
– Continuar con el Paso 004 en la página 276

011

– Método B: Establecer el estado de archivo interno basándose en los códigos de retorno principal y secundario disponibles en el sistema.
– Continuar con el Paso 004 en la página 276

Interpretación de códigos de retorno principal y secundario

Al especificar un archivo TRANSACTION en el programa, la cláusula FILE STATUS de la instrucción SELECT puede contener dos nombres de datos: el estado de archivo externo y el código de retorno externo (principal y secundario). Tal como se describe en el apartado “Determinación del tipo de error mediante la clave de estado de archivo” en la página 274, un estado de archivo puede establecerse de tres formas; sin embargo, los códigos de retorno los establece el sistema después de cualquier transacción de E/S que llame a la gestión de datos. Como consecuencia, la mayoría de condiciones de error que tienen como resultado un mensaje del sistema, también tienen código de retorno asociado.

Los códigos de retorno son similares a valores de estado de archivos. Es decir, los mensajes CPF enviados por el sistema son agrupados por el manejador de excepciones de ejecución ILE COBOL/400 y cada grupo de mensajes CPF se utiliza para establecer uno o más estados de archivo. De forma similar, cada código de retorno principal es también generado por un grupo de mensajes CPF. (El código de retorno secundario no es necesariamente el mismo). La principal diferencia entre los estados de archivos y los códigos de retorno es que la agrupación de los mensajes CPF es distinta.

Aunque ILE COBOL/400 sólo establece códigos de retorno para archivos TRANSACTION, otros tipos de archivos (como los archivos de impresora) también establecen códigos de retorno. Puede acceder a los códigos de retorno para estos archivos mediante un ACCEPT desde la operación I-O-FEEDBACK.

Manejo de mensajes mediante manejadores de condiciones

Un manejador de condiciones proporciona una forma para que un objeto de programa o procedimiento ILE maneje mensajes enviados por el sistema o por otro objeto de programa o procedimiento ILE. Un manejador de condiciones puede manejar uno o más mensajes.

En algunos aspectos, un manejador de condiciones se parece a un procedimiento USE. De forma similar a la manera en que un proceso USE especifica las acciones que deben realizarse como respuesta a un error de E/S, un manejador de condiciones especifica una acción que debe realizarse cuando se produce un error durante el proceso de una instrucción de interfaz de máquina (MI). Un mensaje del sistema señala un error de instrucción MI y cada instrucción ILE COBOL/400 está compuesta por una o más instrucciones MI.

Existen dos tipos de manejadores de condiciones:

- Un tipo de manejadores de condiciones es el que está activo para el todo el programa. Estos manejadores de condiciones están diseñados para manejar condiciones de error genéricas.
- El otro tipo de manejadores de condiciones está activo instrucción a instrucción.Una utilización típica de estos manejadores de condiciones es la supervisión de operaciones de E/S. Estos manejadores de condiciones establecen los estados de archivo e indican condiciones SIZE ERROR, END-OF-PAGE y OVERFLOW.

Manejo de errores en operaciones de Ordenar/Fusionar

Utilice el registro especial SORT-RETURN para detectar errores en las operaciones SORT o MERGE. El registro especial SORT-RETURN contiene código de retorno que indica el éxito o el fallo de una operación SORT o MERGE.El registro especial SORT-RETURN contiene código de retorno de 0 si la operación ha sido satisfactoria o de 16 si no lo ha sido.

Puede establecer el registro especial SORT-RETURN a 16 en un procedimiento declarativo de error o de entrada/salida para finalizar una operación SORT/MERGE antes de que se hayan procesado todos los registros. La operación finaliza antes de que se devuelva o libere un registro.

El registro especial SORT-RETURN tiene la definición implícita:

```
01     SORT-RETURN  GLOBAL   PIC S9(4) USAGE BINARY VALUE ZERO.
```

Cuando se utiliza en programas anidados, el registro especial SORT-RETURN se define implícitamente como GLOBAL en el primer programa ILE COBOL/400.

Consulte las instrucciones SORT y MERGE en la publicación *ILE COBOL/400 Reference* para obtener más información sobre el registro especial SORT-RETURN.

Manejo de excepciones en la instrucción CALL

En una instrucción CALL se produce una condición de excepción cuando se produce una anomalía en la operación CALL. Por ejemplo, puede que el sistema no tenga almacenamiento o no pueda ubicar el programa llamado. En este caso, si no tiene una cláusula ON EXCEPTION o ON OVERFLOW en la instrucción CALL, la aplicación puede finalizar de forma anómala.Utilice la cláusula ON EXCEPTION o ON OVERFLOW para detectar la condición de excepción, evitar la finalización anómala y realizar su propia rutina de manejo de errores.Por ejemplo:

```
CALL "REPORTA"
   IN LIBRARY "MYLIB"
      ON EXCEPTION
      DISPLAY "Programa REPORTA no disponible."
END-CALL
```

Si el programa REPORTA no está disponible o no se encuentra en la biblioteca MYLIB, el control continuará con la cláusula ON EXCEPTION.

Las expresiones ON EXCEPTION y ON OVERFLOW sólo manejan las excepciones resultantes de la anomalía de la operación CALL.

Las condiciones ON EXCEPTION señaladas por la operación CALL se manejan mediante un manejador de condiciones registrado con la prioridad 130. Con este nivel de prioridad, sólo se manejan las condiciones señaladas en la entrada de la pila de llamadas específica donde se encuentra la instrucción CALL. Con este nivel de prioridad los manejadores de condiciones escritas por usuarios puede que no tengan la oportunidad de ver determinadas condiciones.

Si no tiene las expresiones ON EXCEPTION ni ON OVERFLOW en las instrucciones CALL de la aplicación y se produce una anomalía en la instrucción CALL, la excepción la manejará el manejador de condiciones ILE. Consulte el apartado “Manejo de condiciones ILE” en la página 261 para obtener una visión general del manejo de condiciones ILE.

Rutinas de manejo de errores escritas por el usuario

Puede manejar la mayoría de condiciones de error que se produzcan cuando un programa se esté ejecutando utilizando la expresión ON EXCEPTION, la expresión ON SIZE ERROR y otra semántica del lenguaje ILE COBOL/400. Pero en el caso de una condición de error extraordinaria como una comprobación de máquina, ILE COBOL/400 emitirá un mensaje de consulta para permitirle determinar qué acción debe seguir después de que se haya producido un error grave. Sin embargo, ILE COBOL/400 con ILE proporcionan un mecanismo, a través de los manejadores de condiciones ILE escritos por usuario, donde pueden manejarse condiciones de error extraordinarias antes de emitir un mensaje de consulta. El manejo de condiciones ILE le proporciona la oportunidad de escribir sus propias rutinas de manejo de errores para manejar condiciones de errores que permitan al programa continuar ejecutándose.

Los manejadores de condiciones escritos por usuario tienen el nivel de prioridad 165. Este nivel de prioridad otorga a los manejadores de condiciones escritos por el usuario la oportunidad de ver las condiciones señaladas antes de que lo hagan los manejadores de condiciones de entrada-salida o de depurador ILE.

Para que ILE transfiera el control a su propia rutina de manejo de errores escrita por usuario, primero debe identificar y registrar su punto de entrada a ILE. Para registrar un manejador de excepciones, transfiera un puntero de procedimiento a la API de enlace Registrar un manejador de condiciones escritas por usuario (CEEHDLR). Si desea utilizar un programa ILE COBOL/400 como un manejador de excepciones, sólo podrá registrar el programa ILE COBOL/400 más externo. Como ILE COBOL/400 no permite la recursión, si registra un programa ILE COBOL/400 como un manejador de excepción, debe asegurarse de que sólo puede llamarse una vez en un grupo de activación.

Consulte la publicación *ILE Conceptos* para obtener más información sobre los manejadores de excepción. Los elementos de datos de puntero de procedimiento le permiten pasar la dirección de entrada de los puntos de entrada de procedimiento a los servicios ILE. Para obtener más información sobre los elementos de datos de puntero de procedimiento, vea el apartado “Transferencia de las direcciones de punto de entrada con punteros de procedimientos” en la página 260. Puede registrarse cualquier número de manejadores de condiciones escritos por el usuario. Si se registran más de un manejador de condiciones escrito por usuario, a los manejadores se les da otorga el control en el orden último en entrar primero en salir (LIFO).

Los manejadores de condiciones escritos por usuario también se pueden borrar del registro utilizando la API Eliminar de registro manejador de condiciones escritas por usuario (CEEHDLU).

Excepciones comunes y algunas de sus causas

Error de datos decimal MCH1202:

- Se ha utilizado un elemento numérico inicial como fuente cuando no se ha almacenado ningún dato válido previamente en él. El elemento debería tener una cláusula VALUE o debería utilizarse una instrucción MOVE para inicializar su valor.
- Se ha intentado situar un dato no numérico en un elemento numérico.
- Al principio del programa, se han escrito datos incorrectos en un subarchivo. Los datos del subarchivo no se validarán hasta que se graben en la pantalla, por eso, un error 1202 puede producirse en la instrucción WRITE de un registro de control de subarchivo pero los datos incorrectos se han puesto realmente en el subarchivo antes.

Excepciones de puntero MCH0601:

- Parte de un elemento de la Linkage Section se ha extendido más allá del espacio asignado.

  Por ejemplo, si establece la dirección de un elemento de la Linkage section, y uno o más de sus elementos de datos iniciales se extienden más allá del espacio con una instrucción MOVE del elemento de datos inicial, se emitirá MCH0601.

  Para obtener más información sobre la utilización de punteros, consulte Capítulo 11, “Utilización de punteros en un programa ILE COBOL/400” en la página 235.

Alineación de punteros MCH0602:

- La alineación del puntero en la Working Storage Section del programa de llamada no coincide con la alineación de la Linkage Section del programa llamado. La alineación debe estar en un límite de 16 bytes.

  Para obtener más información sobre la utilización de punteros, consulte Capítulo 11, “Utilización de punteros en un programa ILE COBOL/400” en la página 235.

Error de comprobación de rango MCH0603:

- El valor de subíndice es inferior al límite inferior de la matriz o mayor que el límite superior de la matriz, o el operando compuesto ha definido una serie de caracteres fuera de los límites de la serie de caracteres base.

Error de puntero MCH3601:

- Se ha realizado una referencia a un registro o a un campo de un registro y el archivo asociado se ha cerrado o no se ha abierto nunca.

  Por ejemplo, la instrucción OPEN para el archivo no fue satisfactoria y se ha intentado el proceso con cualquier otra instrucción de E/S para ese archivo. El estado del archivo debe comprobarse antes de intentar ninguna otra E/S.

Fin de consultas CPF2415:

- Se ha intentado aceptar la entrada de la corriente de entrada de trabajos mientras el sistema se está ejecutando en modalidad por lotes y no hay ninguna entrada disponible.

Recuperación después de una anomalía

Puede llevarse a cabo alguna recuperación después de una anomalía. Las dos áreas donde puede producirse dicha recuperación son:

- recuperación de archivos con control de compromiso
- recuperación de archivos TRANSACTION.

Recuperación de archivos con control de compromiso

Cuando se rearranca el sistema después de una anomalía, los archivos bajo control de compromiso se restauran automáticamente a sus estados en el último límite de compromiso. Para obtener información adicional sobre el control de compromiso, vea “Utilización del control de compromiso” en la página 310.

El control de compromiso puede encontrarse en el ámbito de dos niveles, el nivel de grupo de activación y el nivel de trabajo. Consulte la sección “Commitment Control Scoping” en *ILE Conceptos* para obtener más información.

Si un trabajo o grupo de activación finaliza de forma anómala (debido a un error del usuario o del sistema), los archivos bajo el control de compromiso se restauran como parte de la terminación de trabajo o del grupo de activación al estado en que se encontraban los archivos durante el último límite de compromiso. El límite de control de compromiso se determina mediante el ámbito de control de compromiso elegido para el programa.

Como los archivos bajo el control de compromiso se retrotraen después de una anomalía del sistema o del proceso, esta característica puede utilizarse para ayudar a rearrancar. Puede crear un registro separado para almacenar datos que serán útiles en el caso de que sea necesario rearrancar un trabajo. Estos datos de rearranque pueden incluir elementos, como por ejemplo totales, contadores, valores de clave de archivo, valores de clave relativa y otra información relevante del proceso de una aplicación.

Si guarda los datos de rearranque mencionados anteriormente en un archivo bajo control de compromiso, los datos de rearranque también se almacenarán permanentemente en la base de datos cuando se emita una instrucción COMMIT. Cuando se produce un ROLLBACK después de una anomalía del trabajo o del proceso, se puede recuperar un registro procesado satisfactoriamente antes de la anomalía. Tenga en cuenta que el método anterior es sólo una técnica de programación sugerida y no siempre será el adecuado, depende de la aplicación.

Recuperación de archivos TRANSACTION

En algunos casos, se pueden recuperar errores de E/S en archivos TRANSACTION sin la intervención del operador o la desactivación/activación de estaciones de trabajo o dispositivos de comunicaciones.

Para errores de E/S potencialmente recuperables en archivos TRANSACTION, el sistema inicia la acción además de los pasos que deben realizarse en el programa de aplicación para intentar la recuperación de errores. Para obtener más información sobre la acción realizada por el sistema, vea la *Gestión de Comunicaciones*.

Examinando el estado del archivo después de una operación de E/S, el programa de aplicación puede determinar si es posible una recuperación de un error de E/S en el archivo TRANSACTION. Si la clave de estado de archivo tiene un valor de 9N, el programa debe ser capaz de recuperarse de un error de E/S. Un procedimiento de recuperación debe codificarse como parte del programa de aplicación y varía dependiendo si un sólo dispositivo se ha adquirido mediante el archivo TRANSACTION o si se han conectado dispositivos múltiples.

Para un archivo con un dispositivo adquirido:

1. Cierre el archivo TRANSACTION con el error de E/S.
2. Vuelva a abrir el archivo.
3. Procese los pasos necesarios para recuperar la operación de E/S anómala. Esto puede implicar un número de pasos, según el tipo de dispositivo de programa utilizado. (Por ejemplo, si la última operación de E/S fue READ, deberá repetir una o más instrucciones WRITE, las cuales se procesaron antes de la instrucción READ). Para obtener más información sobre los procedimientos de recuperación, vea el manual *ICF Programming*.

Para un archivo de pantalla con dispositivos múltiples adquiridos:

1. DEJAR el dispositivo de programa que causó el error de E/S en el archivo TRANSACTION.
2. ADQUIERA el mismo dispositivo de programa.
3. Vea el paso 3 superior.

Para un archivo ICF con dispositivos múltiples adquiridos:

1. ADQUIERA el mismo dispositivo de programa.
2. Vea el paso 3 superior.

Para un archivo de pantalla con dispositivos múltiples adquiridos:

Los intentos de recuperación del programa de aplicación deben normalmente intentarse una sola vez.

Si el intento de recuperación falla:

- Si el archivo sólo tiene conectado un dispositivo de programa, termine el programa procesando las instrucciones STOP RUN, EXIT PROGRAM o GOBACK e intente localizar el origen del error.
- Si el archivo tiene múltiples dispositivos de programa adquiridos, puede realizar una de las opciones siguientes:
  - Continuar procesando sin el dispositivo de programa que causó el error de E/S en el archivo TRANSACTION y volver a adquirir el dispositivo posteriormente.
  - Finalizar el programa.

Para obtener una descripción de los códigos de retorno principales y secundarios que puedan ayudar en el diagnóstico de errores de E/S en el archivo TRANSACTION, vea el manual *ICF Programming* o el manual *Gestión de datos*.

La Figura 68 en la página 284 proporciona un ejemplo de un procedimiento de recuperación de errores.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A*  ARCHIVO DE PANTALLA PARA EJEMPLO DE RECUPERACIÓN DE ERRORES
     A*
     A                                      INDARA
     A          R FORMAT1                   CF01(01 'FIN DEL PROGRAMA')
     A*
     A                                  12 28'ENTRAR ENTRADA '
     A            INPUTFLD       5   I 12 42
     A                                  20 26'F1 - TERMINAR'
```

Figura 67. Ejemplo del procedimiento de recuperación de errores -- DDS

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RECOVERY         AS400SYS   96/07/04 09:39:09         Página     2

                                  F u e n t e

 INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  RECOVERY.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500 SOURCE-COMPUTER. IBM-AS400.
    6     000600 OBJECT-COMPUTER. IBM-AS400.
    7     000700 INPUT-OUTPUT SECTION.
    8     000800 FILE-CONTROL.
    9     000900     SELECT RECOVFILE
   10     001000         ASSIGN TO WORKSTATION-RECVFILE-SI
   11     001100         ORGANIZATION IS TRANSACTION
   12     001200         ACCESS MODE IS SEQUENTIAL
   13     001300         FILE STATUS IS STATUS-FLD, STATUS-FLD-2
   14     001400         CONTROL-AREA IS CONTROL-FLD.
   15     001500     SELECT PRINTER-FILE
   16     001600         ASSIGN TO PRINTER-QPRINT.
          001700
   17     001800 DATA DIVISION.
   18     001900 FILE SECTION.
   19     002000 FD  RECOVFILE.
   20     002100 01  RECOV-REC.
          002200     COPY DDS-ALL-FORMATS OF RECVFILE.
   21    +000001       05  RECVFILE-RECORD PIC X(5).                                    <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1    DE ARCHIVO RECVFILE   DE BIBLIOTECA TESTLIB     <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   22    +000004       05  FORMAT1-I     REDEFINES RECVFILE-RECORD.                   <-ALL-FMTS
   23    +000005           06 INPUTFLD              PIC X(5).                         <-ALL-FMTS
         +000006* FORMATO SALIDA:FORMAT1    DE ARCHIVO RECVFILE   DE BIBLIOTECA TESTLIB      <-ALL-FMTS
         +000007*                                                                      <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINE RECVFILE-RECORD.                    <-ALL-FMTS
          002300
   24     002400 FD  PRINTER-FILE.
   25     002500 01  PRINTER-REC.
   26     002600    05  PRINTER-RECORD             PIC X(132).
          002700
   27     002800 WORKING-STORAGE SECTION.
          002900
   28     003000 01  I-O-VERB                     PIC X(10).
   29     003100 01  STATUS-FLD                   PIC X(2).
   30     003200    88  NO-ERROR                    VALUE "00".
   31     003300    88  ACQUIRE-FAILED              VALUE "9H".
   32     003400    88  TEMPORARY-ERROR             VALUE "9N".
   33     003500 01  STATUS-FLD-2                 PIC X(4).
   34     003600 01  CONTROL-FLD.
   35     003700    05  FUNCTION-KEY               PIC X(2).
   36     003800    05  PGM-DEVICE-NAME            PIC X(10).
   37     003900    05  RECORD-FORMAT              PIC X(10).
   38     004000 01  END-INDICATOR                PIC 1   INDICATOR 1
          004100                                    VALUE B"0".
   39     004200    88  END-NOT-REQUESTED           VALUE B"0".
   40     004300    88  END-REQUESTED               VALUE B"1".
   41     004400 01  USE-PROC-FLAG                PIC 1
          004500                                    VALUE B"1".
   42     004600    88  USE-PROC-NOT-EXECUTED       VALUE B"0".
   43     004700    88  USE-PROC-EXECUTED           VALUE B"1".
```

Figura 68 (Parte 1 de 4). Ejemplo del procedimiento de recuperación de errores

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RECOVERY         AS400SYS   96/07/04 09:39:09         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   44     004800 01  RECOVERY-FLAG                PIC 1
          004900                                     VALUE B"0".
   45     005000    88  NO-RECOVERY-DONE            VALUE B"0".
   46     005100    88  RECOVERY-DONE               VALUE B"1".
   47     005200 01  HEADER-LINE.
   48     005300    05  FILLER                     PIC X(60)
          005400                                     VALUE SPACES.
   49     005500    05  FILLER                     PIC X(72)
          005600                                     VALUE "ERROR REPORT".
   50     005700 01  DETAIL-LINE.
   51     005800    05  FILLER                     PIC X(15)
          005900                                     VALUE SPACES.
   52     006000    05  DESCRIPTION                PIC X(25)
          006100                                     VALUE SPACES.
   53     006200    05  DETAIL-VALUE               PIC X(92)
          006300                                     VALUE SPACES.
   54     006400 01  MESSAGE-LINE.
   55     006500    05  FILLER                     PIC X(15)
          006600                                     VALUE SPACES.
   56     006700    05  DESCRIPTION                PIC X(117)
          006800                                     VALUE SPACES.
   57     006900 PROCEDURE DIVISION.
   58     007000 DECLARATIVES.
          007100 HANDLE-ERRORS SECTION.
          007200     USE AFTER STANDARD ERROR PROCEDURE ON RECOVFILE.  1
          007300 DISPLAY-ERROR.
   59     007400     SET USE-PROC-EXECUTED TO TRUE.
   60     007500     WRITE PRINTER-REC FROM HEADER-LINE
          007600          AFTER ADVANCING PAGE
          007700     END-WRITE
   61     007800     MOVE "ERROR OCCURED IN" TO DESCRIPTION OF DETAIL-LINE.
   62     007900     MOVE I-O-VERB TO DETAIL-VALUE OF DETAIL-LINE.
   63     008000     WRITE PRINTER-REC FROM DETAIL-LINE
          008100          AFTER ADVANCING 5 LINES
          008200     END-WRITE
   64     008300     MOVE "FILE STATUS =" TO DESCRIPTION OF DETAIL-LINE.
   65     008400     MOVE STATUS-FLD TO DETAIL-VALUE OF DETAIL-LINE.  2
   66     008500     WRITE PRINTER-REC FROM DETAIL-LINE
          008600          AFTER ADVANCING 2 LINES
          008700     END-WRITE
   67     008800     MOVE "EXTENDED FILE STATUS =" TO DESCRIPTION OF DETAIL-LINE.
   68     008900     MOVE STATUS-FLD-2 TO DETAIL-VALUE OF DETAIL-LINE.
   69     009000     WRITE PRINTER-REC FROM DETAIL-LINE
          009100          AFTER ADVANCING 2 LINES
          009200     END-WRITE
   70     009300     MOVE "CONTROL-AREA =" TO DESCRIPTION OF DETAIL-LINE.
   71     009400     MOVE CONTROL-FLD TO DETAIL-VALUE OF DETAIL-LINE.
   72     009500     WRITE PRINTER-REC FROM DETAIL-LINE
          009600          AFTER ADVANCING 2 LINES
          009700     END-WRITE.                                                                    96/07/04
          009800 CHECK-ERROR.
   73     009900     IF TEMPORARY-ERROR AND NO-RECOVERY-DONE THEN
   74     010000         MOVE "***ERROR RECOVERY BEING ATTEMPTED***"  3
          010100             TO  DESCRIPTION OF MESSAGE-LINE
   75     010200         WRITE PRINTER-REC FROM MESSAGE-LINE
          010300             AFTER ADVANCING 3 LINES
          010400         END-WRITE
```

Figura 68 (Parte 2 de 4). Ejemplo del procedimiento de recuperación de errores

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RECOVERY         AS400SYS   96/07/04 09:39:09         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   76     010500         SET RECOVERY-DONE TO TRUE
   77     010600         DROP PGM-DEVICE-NAME FROM RECOVFILE
   78     010700         ACQUIRE PGM-DEVICE-NAME FOR RECOVFILE  4
          010800     ELSE
   79     010900         IF RECOVERY-DONE THEN  5
   80     011000             MOVE "***ERROR AROSE FROM RETRY AFTER RECOVERY***"
          011100                 TO  DESCRIPTION OF MESSAGE-LINE
   81     011200             WRITE PRINTER-REC FROM MESSAGE-LINE
          011300                 AFTER ADVANCING 3 LINES
          011400             END-WRITE
   82     011500             MOVE "***PROGRAM ENDED***"
          011600                 TO  DESCRIPTION OF MESSAGE-LINE
   83     011700             WRITE PRINTER-REC FROM MESSAGE-LINE
          011800                 AFTER ADVANCING 2 LINES
          011900             END-WRITE
   84     012000                 CLOSE RECOVFILE
          012100                       PRINTER-FILE
   85     012200                 STOP RUN
          012300         ELSE
   86     012400             SET NO-RECOVERY-DONE TO TRUE
          012500         END-IF
          012600     END-IF
   87     012700     MOVE "***EXECUTION CONTINUES***"
          012800         TO DESCRIPTION OF MESSAGE-LINE.
   88     012900     WRITE PRINTER-REC FROM MESSAGE-LINE
          013000         AFTER ADVANCING 2 LINES
          013100     END-WRITE.                                                                    96/07/04
          013200 END DECLARATIVES.
          013300
          013400 MAIN-PROGRAM SECTION.
          013500 MAINLINE.
   89     013600     MOVE "OPEN" TO I-O-VERB.
   90     013700     OPEN I-O     RECOVFILE
          013800          OUTPUT  PRINTER-FILE.
   91     013900     PERFORM I-O-PARAGRAPH UNTIL END-REQUESTED.  6
   92     014000     CLOSE RECOVFILE
          014100           PRINTER-FILE.
   93     014200     STOP RUN.
          014300
          014400 I-O-PARAGRAPH.
   94     014500     PERFORM UNTIL USE-PROC-NOT-EXECUTED OR NO-RECOVERY-DONE  7
   95     014600          MOVE "WRITE" TO I-O-VERB
   96     014700          SET USE-PROC-NOT-EXECUTED TO TRUE
   97     014800          WRITE RECOV-REC FORMAT IS "FORMAT1"
          014900               INDICATOR IS END-INDICATOR
          015000          END-WRITE
          015100     END-PERFORM
   98     015200     MOVE "READ" TO I-O-VERB.
   99     015300     SET USE-PROC-NOT-EXECUTED TO TRUE.
  100     015400     SET NO-RECOVERY-DONE TO TRUE.
  101     015500     READ RECOVFILE FORMAT IS "FORMAT1"
          015600         INDICATOR IS END-INDICATOR  8
          015700     END-READ
  102     015800     IF NO-ERROR THEN
  103     015900         PERFORM SOME-PROCESSING
          016000     END-IF.                                                                       96/07/04
          016100
```

Figura 68 (Parte 3 de 4). Ejemplo del procedimiento de recuperación de errores

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RECOVERY         AS400SYS   96/07/04 09:39:09         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


          016200 SOME-PROCESSING.
          016300*    (Inserte el proceso de alguna base de datos por ejemplo).

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 68 (Parte 4 de 4). Ejemplo del procedimiento de recuperación de errores

1 Define el proceso que se lleva a cabo cuando se produce un error de E/S en RECOVFILE.

2 Imprime información para ayudar a diagnosticar el problema.

3 Si el estado del archivo es igual a 9N (error temporal) y no se ha intentado ninguna recuperación de error previa para esta operación de E/S, ahora se intentará la recuperación de error.

4 La recuperación consiste en dejar y luego volver a adquirir el dispositivo de programa en el cual se ha producido el error de E/S.

5 Para evitar la repetición en bucle del programa, si la recuperación se ha intentado antes, no se intentará ahora.

6 La línea principal del programa consiste en grabar en y leer de un dispositivo hasta que el usuario señale una finalización del programa pulsando F1.

7 Si la operación WRITE ha fallado pero se ha realizado la recuperación, WRITE se intenta de nuevo.

8 Si la operación READ ha fallado, el proceso continuará grabando en el dispositivo de nuevo y luego, intentando la operación READ otra vez.

Consideraciones sobre entrada y salida en ILE COBOL/400

Capítulo 13. Definición de archivos

Este capítulo describe cómo:

- definir archivos descritos por programa
- definir archivos descritos externamente
- describir archivos utilizando las Especificaciones de descripción de datos (DDS)
- utilizar archivos descritos externamente en un programa ILE COBOL/400.

Tipos de descripciones de archivos

El elemento clave para todas las operaciones de E/S del sistema AS/400 es el archivo. El sistema operativo mantiene una descripción de cada archivo utilizado por un programa. La descripción del archivo del sistema operativo incluye información sobre el tipo de archivo, como por ejemplo la base de datos o un dispositivo, la longitud de los registros del archivo y una descripción de cada campo y sus atributos. El archivo se describe al nivel del campo del sistema operativo a través de IDDU, de los mandatos SQL/400 o de DDS. Si crea un archivo (por ejemplo, utilizando el mandato CRTPF) sin especificar DDS para él, el archivo aún tiene una descripción de campo. El campo único tiene el mismo nombre que el archivo y tiene la longitud de registro especificada en el mandato de creación.

Puede definir un archivo de dos formas:

- Un **archivo descrito por programa** lo describe el programador en el nivel de campo de la DATA DIVISION dentro del programa ILE COBOL/400.
- Para un **archivo descrito externamente**, el compilador ILE COBOL/400 utiliza la descripción del archivo del sistema para generar las instrucciones fuente ILE COBOL/400 de la DATA DIVISION, que describen el archivo en el nivel de campo dentro del programa ILE COBOL/400. El archivo debe crearse antes de compilar el programa.

Tanto los archivos descritos externamente como los descritos por programa deben definirse en el programa ILE COBOL/400 dentro de la INPUT-OUTPUT SECTION y la FILE SECTION. Las descripciones registradas de la FILE SECTION para archivos descritos externamente se definen con la instrucción COPY de Formato 2. Sólo se extraen las descripciones de nivel de campo. Cuando se especifica EXTERNALLY-DESCRIBED-KEY como RECORD KEY, también se extraen los campos que componen RECORD KEY desde DDS. Para obtener más información sobre la instrucción COPY de Formato 2, vea la *ILE COBOL/400 Reference*.

El proceso de archivos real dentro de la Procedure Division es la misma si el archivo se describe externamente o lo describe el programa.

Definición de archivos descritos por el programa

Los registros y los campos para un archivo descrito por programa se describen codificando las descripciones de registro directamente en la FILE SECTION del programa ILE COBOL/400, en lugar de utilizar la instrucción COPY de Formato 2.

El archivo debe existir en el sistema antes de que pueda ejecutarse el programa. La única excepción es cuando se utiliza la creación dinámica de archivos, especificando OPTION(*CRTF) en el mandato CRTCBLMOD/CRTBNDCBL. Para más información, consulte la descripción del parámetro OPTION en “Parámetros del mandato CRTCBLMOD” en la página 33.

Para crear un archivo, utilice uno de los mandatos de Crear archivo, que puede encontrar en la publicación *CL Reference*. Las DDS puede utilizarse con los mandatos Crear archivo. Para un archivo indexado ILE COBOL/400, debe crearse una vía de acceso por clave. Especifique una clave en las DDS al crear el archivo. La clave del registro del programa ILE COBOL/400 debe coincidir con la clave definida al crear el archivo. Si estos valores clave no coinciden, aún puede realizarse la operación sobre el archivo pero se le pasa al sistema una clave de archivo incorrecta. Si la clave de registro incorrecta contiene por casualidad un valor de clave aparentemente correcta, la operación de entrada/salida se realizará satisfactoriamente, pero sobre datos incorrectos. Por ello, la integridad de los datos puede comprometerse. Para evitar que ocurra este problema, debería utilizar archivos descritos externamente siempre que sea posible.

Definición de archivos descritos externamente

La descripción externa para un archivo incluye:

- Las especificaciones de formato de registro que contienen una descripción de los campos en un registro
- Especificaciones de vía de acceso que describen cómo se recuperarán los registros.

Estas especificaciones provienen de la descripción de archivos externa y del mandato OS/400 que utilice para crear el archivo.

Los archivos descritos externamente ofrecen las siguientes ventajas sobre los archivos descritos por programa:

- Menos codificación en programas ILE COBOL/400. Si el mismo archivo lo utilizan varios programas, los campos pueden definirse una vez para el sistema operativo y luego los utilizarán todos los programas. Esto elimina la necesidad de codificar una descripción de registro separada para cada programa que utilice el archivo.
- Reduce la posibilidad de un error de programación. Puede actualizar programas a menudo cambiando el formato del registro del archivo y luego recompilando los programas que utilizan el archivo sin cambiar ninguna codificación del programa.
- Comprobación de nivel de la descripción de archivos. Se realiza una comprobación de nivel de la descripción del archivo en el programa ILE COBOL/400 y del archivo real del sistema cuando éste se abre (a menos que se especifique LVLCHK(*NO) en el mandato de crear archivo o de alteración temporal). Si la

descripción del archivo en el programa no coincide con el archivo real, la operación de apertura fallará con un estado de archivo de 39.

- Para archivos indexados, si se especifica EXTERNALLY-DESCRIBED-KEY en la cláusula RECORD KEY, puede asegurar que la clave de archivo ocupa la misma posición en el archivo real que el la descripción del archivo del programa ILE COBOL/400. Además, puede utilizar claves no continuas, lo que no es posible con archivos descritos por programa.
- Documentación mejorada. Los programas que utilizan los mismos archivos utilizan nombres de formatos de registro y de campos coherentes.
- Cualquier edición a procesar en archivos de salida descritos externamente pueden especificarse en DDS.

Antes de que pueda utilizar un archivo descrito externamente en su programa, debe crear DDS para describir el archivo y crear el archivo real.

Descripción de archivos utilizando especificaciones de descripción de datos (DDS)

Puede utilizar Especificaciones de descripción de datos (DDS) para describir archivos a nivel de campo en el sistema operativo. En DDS, cada formato de registro de un archivo descrito externamente se identifica mediante un nombre de formato de registro exclusivo.

Las especificaciones de formato de registro describen los campos de un registro y su ubicación. Los campos se sitúan en el registro en el orden especificado en las DDS. La descripción de campos incluye generalmente el nombre del campo, el tipo de campo (carácter, binario, decimal externo, decimal interno, coma flotante interno) y la longitud de campo (incluyendo el número de posiciones decimales de un campo numérico). En lugar de especificarse en el formato de registro para un archivo físico o lógico, los atributos de campo pueden definirse en un archivo de referencia de campos. (Vea la Figura 69 en la página 294.)

Las claves para un formato de registro se especifican en las DDS. Al utilizar una instrucción COPY de Formato 2, se genera una tabla de comentarios en el listado de programas fuente mostrando como se definen las claves para el formato en las DDS.

Además, pueden utilizarse palabras clave de DDS para:

- Especificar códigos de edición para un campo (EDTCDE)
- Especificar palabras de edición para un campo (EDTWRD)
- Especificar que los valores de clave duplicados no se permiten para el archivo (UNIQUE)
- Especificar una descripción de texto para un formato de registro o un un campo (TEXT).

Vea la publicación *DDS Reference* para obtener una lista completa de las palabras clave de DDS válidas para un archivo de base de datos.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A**FLDREF    DSTREF     DISTRIBUTION APPLICATION FIELDS REFERENCE
     A          R DSTREF                 TEXT('REF. CAMPO DISTRIBUCIÓN')
     A* COMMON FIELDS USED AS REFERENCE
   1 A            BASDAT         6  0     EDTCDE(Y)
     A                                    TEXT('CAMPO FECHA BASE')
     A* FIELDS USED BY CUSTOMER MASTER FILE
   2 A            CUST           5        CHECK(MF)
     A                                    COLHDG('NÚMERO' 'CLIENTE')
     A            NAME          20        COLHDG('NOMBRE DEL CLIENTE')
   3 A            ADDR       R            REFFLD(NAME)
     A                                    COLHDG('DIRECCIÓN DEL CLIENTE')
     A            CITY       R            REFFLD(NAME)
     A                                    COLHDG('CIUDAD DEL CLIENTE')
   2 A            STATE          2        CHECK(MF)
     A                                    COLHDG('PROVINCIA')
     A            SRHCOD         6        CHECK(MF)
     A                                    COLHDG('CÓDIGO' 'BÚSQUEDA')
     A                                    TEXT('CÓDIGO BÚSQUEDA NÚMERO CLIENTE')
   2 A            ZIP            5  0     CHECK(MF)
     A                                    COLHDG('CÓDIGO' 'POSTAL')
   4 A            CUSTYP         1  0     RANGE(1 5)
     A                                    COLHDG('TIPO' 'CLIENTE')
     A                                    TEXT('TIPO DE CLIENTE 1=GOV 2=SCH 3=B+
     A                                    US 4=PT 5=OTH')
   5 A            ARBAL          8  2     COLHDG('SALDO' 'REG.CTAS.')
     A                                    EDTCDE(J)
   6 A            ORDBAL     R            REFFLD(ARBAL)
     A                                    COLHDG('TOTAL CC' 'EN ARCHIVO PEDIDOS')
     A            LSTAMT     R            REFFLD(ARBAL)
     A                                    COLHDG('ÚLTIMA' 'CANT.' 'PAGADA')
   7 A                                     TEXT('ÚLTIMA CANT. PAGADA EN CC')
     A            LSTDAT     R            REFFLD(ARBAL)
     A                                    COLHDG('ÚLTIMA' 'FECHA' 'PAGADA  ')
     A                                    TEXT('ÚLTIMA FECHA PAGADA EN CC')
     A            CRDLMT         8  2     COLHDG('LÍMITE' 'CRÉDITO' 'CLIENTE')
     A                                    EDTCDE(J)
     A            SLSYR         10  2     COLHDG('VENTAS' 'CLIENTE' 'AÑO ACTUAL')
     A                                    EDTCDE(J)
     A            SLSLYR        10  2     COLHDG('VENTAS' 'CLIENTE' 'AÑO ANTERIOR')
     A                                    EDTCDE(J)
```

Figura 69. Ejemplo de un archivo de referencia de campos

Este ejemplo de un archivo de referencia de campos (Figura 69) muestra las definiciones de los campos utilizados por el archivo CUSMSTL (lógico maestro de clientes), que se muestra en la Figura 70 en la página 295. El archivo de referencia de campos contiene normalmente las definiciones de campos que utilizan otros archivos. El texto siguiente describe algunas de las entradas para este archivo de referencia de campos.

1 El códico de edición Y edita el campo BASDAT, tal como lo indica la palabra clave EDTCDE (Y). Si este campo se utiliza en un archivo de salida descrito externamente para un programa ILE COBOL/400, el campo generado por COBOL es compatible con el tipo de datos especificado en las DDS. El campo se edita cuando se escribe el registro. Si el campo se utiliza en un archivo de salida descrito por programa, la compatibilidad con los campos DDS del archivo es responsabilidad del usuario. Si DDS no se utiliza para crear el archivo, la edición apropiada del campo en el programa ILE COBOL/400 también es responsabilidad del usuario.

2 La entrada CHECK(MF) especifica que el campo es un campo de relleno obligatorio cuando se entra desde una estación de trabajo de pantalla. De relleno obligatorio significa que todos los caracteres para el campo deben introducirse desde la estación de trabajo de la pantalla.

3 Los campos ADDR y CITY comparten los mismos atributos especificados para el campo NAME, tal como lo indica la palabra clave REFFLD.

4 La palabra clave RANGE, especificada para el campo CUSTYP, asegura que los únicos valores válidos que pueden introducirse en este campo desde una estación de trabajo de pantalla son del 1 al 5.

5 La palabra clave COLHDG proporciona una cabecera de columna para el campo si la utilizan las Herramientas para el desarrollo de aplicaciones/400.

6 El código de edición J edita el campo ARBAL, tal como lo indica la palabra clave EDTCDE(J).

7 Para algunos campos se proporciona una descripción de texto (palabra clave TEXT). La palabra clave TEXT se utiliza por motivos de documentación y aparece en varios listados.

Utilización de archivos descritos externamente en un programa ILE COBOL/400

Puede incorporar la descripción de archivos en el programa codificando una instrucción COPY de Formato 2. Entonces la información de la descripción externa la recupera el compilador ILE COBOL/400 y se genera una estructura de datos ILE COBOL/400.

Las páginas siguientes proporcionan ejemplos de la utilización de DDS y del código ILE COBOL/400 que resultaría de la utilización de una instrucción COPY de Formato 2. (Vea la *ILE COBOL/400 Reference* para obtener una descripción detallada de la instrucción COPY de Formato 2.)

- La Figura 70 muestra las DDS para un archivo lógico y la Figura 71 en la página 296 muestra el código ILE COBOL/400 generado.
- La Figura 72 en la página 297 describe el mismo archivo pero incluye la palabra clave ALIAS y la Figura 73 en la página 297 muestra el código ILE COBOL/400 generado.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
   1 A** LOGICAL  CUSMSTL   CUSTOMER MASTER FILE
     A                                     2 UNIQUE
     A        3 R CUSREC                    PFILE(CUSMSTP)
     A                                      TEXT('CUSTOMER MASTER RECORD')
     A            CUST
     A            NAME
     A            ADDR
     A            CITY
     A            STATE
     A            ZIP
     A            SRHCOD
     A            CUSTYP
     A            ARBAL
     A            ORDBAL
     A            LSTAMT
     A            LSTDAT
     A            CRDLMT
     A            SLSYR       5
     A            SLSLYR
     A        4 K CUST
```

Figura 70. Ejemplo de Especificaciones de descripción de datos para un archivo lógico

1 Se define un archivo lógico para procesar el archivo físico maestro de clientes (CUSMSTP) y se denomina CUSMSTL.

2 La palabra clave UNIQUE indica que no están permitidos los valores de clave duplicados.

3 Se define un formato de registro (CUSREC) para el archivo CUSMSTL, el cual debe basarse en el archivo físico CUSMSTP.

4 El campo CUST se identifica como el campo de clave para este archivo.

5 Si no se especifican los atributos de campo (como por ejemplo, longitud, tipo de datos y posiciones decimales) en las DDS para un archivo lógico, los atributos se obtienen del campo correspondiente en el archivo físico. Cualquier atributo de campo especificado en las DDS para el archivo lógico altera temporalmente los atributos para el campo correspondiente del archivo físico. La definición de los campos del archivo físico puede hacer referencia a un archivo de referencia de campos. Un archivo de referencia de campos es un archivo de descripción de datos formado por nombres de campos y sus funciones, como por ejemplo el tamaño y el tipo. Cuando se utiliza un archivo de referencia de campos, los mismos campos utilizados en formatos de registro múltiples deben definirse sólo una vez en el archivo de referencia de campos. Para obtener más información sobre un archivo de referencia de campos, vea el manual *DB2/400 Programación de la base de datos*.

La Figura 69 en la página 294 muestra un ejemplo de un archivo de referencia de campos que define los atributos de los campos utilizados en el archivo de base de datos. Vea el manual *DB2/400 Programación de la base de datos* para obtener más información relacionada con los archivos de referencia de campos.

```
                            F u e n t e

INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


  18     000200 01  CUSTOMER-INVOICE-RECORD.
         000210     COPY DDS-CUSREC OF CUSMSTL.
        +000001*    FORMATO E-S:CUSREC     DE ARCHIVO CUSMSTL    DE BIBLIOTECA TESTLIB           CUSREC
        +000002*                         REGISTRO MAESTRO DEL CLIENTE                       CUSREC
        +000003*         CLAVE SUMINISTRADA POR USUARIO MEDIANTE CLÁUSULA RECORD KEY        CUSREC
  19    +000004       05  CUSREC.                                                          CUSREC
  20    +000005           06 CUST                  PIC X(5).                              CUSREC
        +000006*                 NÚMERO DE CLIENTE                                        CUSREC
  21    +000007           06 NAME                  PIC X(25).                             CUSREC
        +000008*                 NOMBRE DE CLIENTE                                        CUSREC
  22    +000009           06 ADDR                  PIC X(20).                             CUSREC
        +000010*                 DIRECCIÓN DE CLIENTE                                     CUSREC
  23    +000011           06 CITY                  PIC X(20).                             CUSREC
        +000012*                 CIUDAD DEL CLIENTE                                       CUSREC
  24    +000013           06 STATE                 PIC X(2).                              CUSREC
        +000014*                 PROVINCIA                                                CUSREC
  25    +000015           06 ZIP                   PIC S9(5)       COMP-3.                CUSREC
        +000016*                 CÓDIGO POSTAL                                            CUSREC
  26    +000017           06 SRHCOD                PIC X(6).                              CUSREC
        +000018*                 CÓDIGO BÚSQUEDA NÚMERO CLIENTE                           CUSREC
  27    +000019           06 CUSTYP                PIC S9(1)       COMP-3.                CUSREC
        +000020*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT             CUSREC
  28    +000021           06 ARBAL                 PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000022*                 SALDO REGISTRO CUENTAS                                   CUSREC
  29    +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000024*                 TOTAL CC EN ARCHIVO PEDIDOS                              CUSREC
  30    +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000026*                 ÚLTIMA CANT. PAGADA EN CC                                CUSREC
  31    +000027           06 LSTDAT                PIC S9(6)       COMP-3.                CUSREC
        +000028*                 ÚLTIMA FECHA PAGADA EN CC                                CUSREC
  32    +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000030*                 LÍMITE CRÉDITO CLIENTE                                   CUSREC
  33    +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                CUSREC
        +000032*                 VENTAS CLIENTE AÑO ACTUAL                                CUSREC
  34    +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                CUSREC
        +000034*                 VENTAS CLIENTE AÑO ANTERIOR                              CUSREC
```

Figura 71. Ejemplo de los resultados de la instrucción COPY de formato 2 (DDS)

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A** ARCHIVO MAESTRO DE CLIENTES CUSMSTL
     A                                     UNIQUE
     A          R CUSREC                   PFILE(CUSMSTP)
     A                                     TEXT('REGISTRO MAESTRO CLIENTE')
     A            CUST                     ALIAS(NÚMERO_CLIENTE)
     A            NAME                   1 ALIAS(NOMBRE_CLIENTE)
     A            ADDR                     ALIAS(ADDRESS)
     A            CITY
     A            STATE
     A            ZIP
     A            SRHCOD                   ALIAS(CÓDIGO_BÚSQUEDA)
     A            CUSTYP                   ALIAS(TIPO_CLIENTE)
     A            ARBAL                    ALIAS(SALDO_REG_CUENTAS)
     A            ORDBAL
     A            LSTAMT
     A            LSTDAT
     A            CRDLMT
     A            SLSYR
     A            SLSLYR
     A          K CUST
```

Figura 72. Ejemplo de Especificaciones de descripción de datos con ALIAS

1 Este es el nombre asociado con la palabra clave ALIAS, que se incluirá en el programa. Disponible a través de la opción DDS ALIAS, un alias es un pseudónimo alternativo que permite incluir un nombre de datos de hasta 30 caracteres en un programa ILE COBOL/400.

```
                              F u e n t e

INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


  18     002000 01  CUSTOMER-INVOICE-RECORD.
         002100     COPY DDS-CUSREC OF CUSMSTL ALIAS.
        +000001*    FORMATO E-S:CUSREC    DE ARCHIVO CUSMSTL    DE BIBLIOTECA TESTLIB  CUSREC
        +000002*                          REGISTRO MAESTRO DEL CLIENTE                       CUSREC
        +000003*          CLAVE PROPORCIONADA POR USUARIO MEDIANTE CLÁUSULA RECORD KEY      CUSREC
  19    +000004       05  CUSREC.                                                          CUSREC
  20    +000005           06 CUSTOMER-NUMBER       PIC X(5).                              CUSREC
        +000006*                 NÚMERO DEL CLIENTE CUSREC
  21    +000007           06 CUSTOMER-NAME         PIC X(25).                             CUSREC
        +000008*                 NOMBRE DEL CLIENTE                                      CUSREC
  22    +000009           06 ADDRESS-DDS           PIC X(20).                             CUSREC
        +000010*                 DIRECCIÓN DEL CLIENTE                                   CUSREC
  23    +000011           06 CITY                  PIC X(20).                             CUSREC
        +000012*                 CIUDAD DEL CLIENTE                                      CUSREC
  24    +000013           06 STATE                 PIC X(2).                              CUSREC
        +000014*                 PROVINCIA                                               CUSREC
  25    +000015           06 ZIP                   PIC S9(5)       COMP-3. CUSREC
        +000016*                 CÓDIGO POSTAL                                           CUSREC
  26    +000017           06 SEARCH-CODE           PIC X(6).                              CUSREC
        +000018*                 CÓDIGO DE BÚSQUEDA DE NÚMERO CLIENTE                    CUSREC
  27    +000019           06 CUSTOMER-TYPE         PIC S9(1)       COMP-3.                CUSREC
        +000020*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT            CUSREC
  28    +000021           06 ACCT-REC-BALANCE      PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000022*                 SALDO DE REGISTROS DE CUENTA                            CUSREC
  29    +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000024*                 TOTAL CC EN ARCHIVO PEDIDOS                             CUSREC
  30    +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000026*                 ÚLTIMA CANTIDAD PAGADA EN CC                            CUSREC
  31    +000027           06 LSTDAT                PIC S9(6)       COMP-3.                CUSREC
        +000028*                 ÚLTIMA FECHA PAGADA EN CC                               CUSREC
  32    +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                CUSREC
        +000030*                 LÍMITE CRÉDITO CLIENTE                                  CUSREC
  33    +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                CUSREC
        +000032*                 VENTAS CLIENTE AÑO ACTUAL                               CUSREC
  34    +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                CUSREC
        +000034*                 VENTAS CLIENTE AÑO ANTERIOR                             CUSREC
```

Figura 73. Ejemplo de los resultados de la instrucción COPY de formato 2 (DD) con la palabra clave ALIAS

Además de colocar la descripción externa del archivo en el programa mediante la utilización de la instrucción COPY de Formato 2, también puede utilizar la definición y redefinición de registro estándar para describir archivos externos o proporcionar una definición de grupo para una serie de campos. Es responsabilidad del programador asegurarse de que las definiciones descritas por programa son compatibles con las defincciones externas del archivo.

La Figura 74 muestra cómo los programas ILE COBOL/400 pueden estar relacionados con archivos en el sistema AS/400, utilizando descripciones de archivo externas de las DDS.

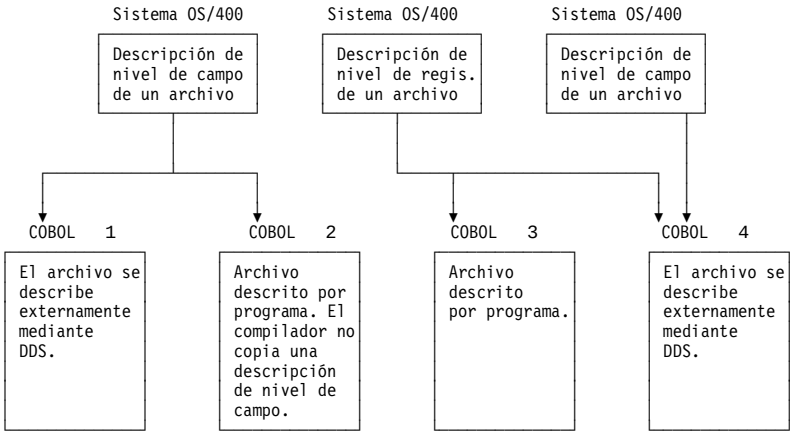


Figura 74. Ejemplo que muestra cómo ILE COBOL/400 puede relacionarse con archivos AS/400

1 El programa ILE COBOL/400 utiliza la descripción de nivel de campo de un archivo definido para el sistema operativo. Codifique una instrucción COPY de Formato 2 para la descripción del registro. En la compilación, el compilador copia la descripción de nivel de campo externa y la convierte en una descripción de registro ILE COBOL/400 sintácticamente correcta. El archivo debe existir en el momento de la compilación.

2 Se utiliza un archivo descrito externamente como un archivo descrito por programa en el programa ILE COBOL/400. La descripción de registro completa para el archivo se codifica en el programa ILE COBOL/400. Este archivo no tiene que existir en el momento de la compilación.

3 Se describe un archivo al sistema operativo aunque sólo sea el nivel de registro. La descripción de registro completa debe codificarse en el programa ILE COBOL/400. Este archivo no tiene que existir en el momento de la compilación.

4 Puede especificarse un nombre de archivo en la compilación y, en la ejecución, puede especificarse un nombre de archivo distinto. Una instrucción ILE COBOL/400 COPY de Formato 2 genera la descripción de registro para el archivo en la compilación. En la ejecución, puede utilizarse una lista de bibliotecas distinta o un mandato de alteración temporal de archivos de manera que el programa acceda a un archivo distinto. La descripción de archivo copiada en la compilación se utiliza para describir los registros de entrada utilizados en la ejecución.

Nota: Para los archivos descritos externamente, los dos formatos de archivo deben ser los mismos. De lo contrario, se producirá un error de comprobación de nivel.

Especificación de recuperación de registros por clave y de registros que no son por clave

La descripción de un archivo descrito externamente contiene la vía de acceso que describe cómo se recuperan los registros del archivo. Los registros pueden recuperarse basándose en una vía de acceso (sin clave) en secuencia de llegada o en una vía de acceso en secuencia por clave. Vea *DB2/400 Programación de la base de datos* para obtener una descripción completa de las vías de acceso para un archivo de base de datos descrito externamente.

La **vía de acceso en secuencia de llegada** se basa en el orden en que los registros se almacenan en el archivo. Los registros sólo se añaden al final del archivo.

Para la **vía de acceso en secuencia por clave**, la secuencia en que se recuperan los registros desde el archivo se basa en el contenido de los campos clave definidos en las DDS para el archivo. Por ejemplo, en las DDS mostradas en la Figura 70 en la página 295, CUST se define como el campo clave. La vía de acceso en secuencia por clave se actualiza siempre que se añaden o eliminan registros o cuando cambia el contenido de un campo clave. Para una vía de acceso en secuencia por clave, pueden definirse uno o más campos en las DDS para utilizarlos como los campos clave para un formato de registro. No todos los formatos de registro de un archivo tienen que tener los mismos campos clave. Por ejemplo, un registro de cabecera de orden puede tener el campo ORDER definido como el campo clave y los registros de detalle de orden pueden tener los campos ORDER y LINE definidos como los campos clave.

Si no especifica un formato en la operación de E/S, entonces la clave para un archivo la determinan las claves válidas para los formatos de registro de ese archivo. La clave del archivo se determina de la siguiente forma:

- Si todos los formatos de registro de un archivo tienen el mismo número de campos clave definidos en DDS y son idénticos en atributos, la *clave para el archivo* consta de todos los campos de la clave para los formatos de registro. (Los campos correspondientes no tienen el mismo nombre). Por ejemplo, si el archivo tiene tres formatos de registro y la clave para cada formato de registro está formada por los campos A, B y C, la clave del archivo está formada por los campos A, B y C. Es decir, la clave del archivo es la misma que la clave del registro.
- Si todos los formatos de registro del archivo no tienen los mismos campos clave, la clave para el archivo está formada por los campos clave *comunes* a todos los formatos de registro. Por ejemplo, un archivo tiene tres formatos de registro y los campos clave se definen tal como se muestra a continuación:
  - REC1 contiene el campo clave A.
  - REC2 contiene los campos clave A y B.
  - REC3 contiene los campos clave A, B y C.

  Entonces la clave del archivo será A, el campo clave común de todos los formatos de registro.
- Si no hay ningún campo clave común a todos los formatos de registro, cualquier referencia por clave al archivo dará siempre como resultado el primer registro del archivo.

En ILE COBOL/400, debe especificar RECORD KEY para un archivo indexado para identificar el registro que desee procesar. ILE COBOL/400 compara el valor clave con la clave del archivo o del registro y procesa la operación especificada en el registro, la clave de la cual coincide con el valor RECORD KEY.

Cuando se especifique RECORD KEY IS EXTERNALLY-DESCRIBED-KEY:

- Si se especifica la expresión FORMAT, el compilador crea el argumento de búsqueda a partir de los campos clave del área de registro para el formato especificado.
- Si no se especifica la expresión FORMAT, el compilador crea el argumento de búsqueda a partir de los campos clave en el área de registro para el primer formato de registro definido en el programa para ese archivo.

Nota: Para un archivo que contenga múltiples campos clave a procesar en ILE COBOL/400, los campos clave deben ser continuos en el formato de registro utilizado por el programa ILE COBOL/400, excepto cuando se especifique RECORD KEY IS EXTERNALLY-DESCRIBED-KEY.

Comprobación del nivel de los archivos descritos externamente

Cuando un programa ILE COBOL/400 utiliza un archivo descrito externamente, el sistema operativo proporciona una función de comprobación de nivel (LVLCHK). Esta función asegura que los formatos del archivo no se han cambiado desde la compilación.

El compilador siempre proporciona la información requerida por la comprobación de nivel cuando se utiliza un archivo descrito externamente (es decir, cuando se ha definido una descripción de registro para el archivo utilizando la instrucción COPY de Formato 2). Sólo a aquellos formatos que se copiaron mediante la instrucción COPY de Formato 2 bajo FD para un archivo se les comprueba el nivel. La función de comprobación de nivel se iniciará en la ejecución basándose en la selección realizada en los mandatos de creación, cambio o alteración temporal de un archivo. El valor por omisión del mandato de creación de archivo es para solicitar la comprobación de nivel. Si se solicita, la comprobación de nivel se produce según el formato de registro cuando se abre el archivo. Si se produce un error de comprobación de nivel, ILE COBOL/400 establece un estado de archivo de 39.

Cuando se especifica LVLCHK(*NO) en los mandatos CRTxxxF, CHGxxxF o OVRxxxF CL y el archivo se vuelve a crear utilizando un formato existente, los programas ILE COBOL/400 existentes que utilicen dicho formato puede que no funcionen sin recompilación, según los cambios realizados al formato.

Tenga mucha precaución al utilizar archivos en programas ILE COBOL/400 sin comprobación de nivel. Se arriesga a que se produzca un fallo en el programa y a dañar los datos si utiliza programas ILE COBOL/400 sin comprobación de nivel ni recompilación.

Nota: El compilador ILE COBOL/400 no proporciona comprobación de nivel para archivos descritos por programa.

Para obtener más información sobre la comprobación de nivel, vea el manual *Gestión de datos*.

Capítulo 14. Proceso de archivos

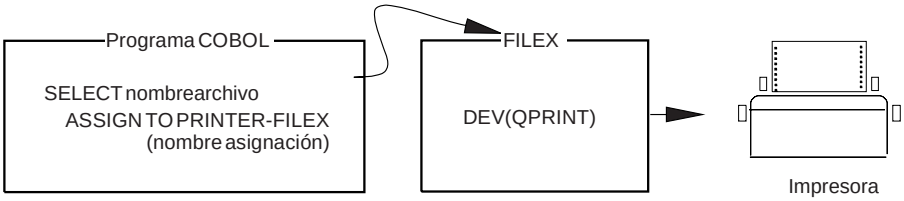
ILE COBOL/400 utiliza y procesa archivos de varias formas en el AS/400. Este capítulo describe cómo:

- asociar archivos con dispositivos de entrada-salida
- especificar spooling de entrada y salida
- cambiar atributos de archivos
- redirigir la entrada y salida de los archivos
- bloquear y liberar archivos
- desbloquear registros de entrada y bloquear registros de salida
- compartir una vía de datos abierta para acceder a un archivo
- utilizar el estado de archivo y las áreas de realimentación
- utilizar el control de compromiso
- ordenar y fusionar archivos
- declarar elementos de datos utilizando tipos de datos CVTOPT

Asociación de archivos con dispositivos de entrada-salida

Los archivos sirven de enlace de conexión entre un programa y el dispositivo utilizado para la entrada y salida. La asociación de dispositivos real se realiza en el momento en que se abre el archivo. En algunos casos, este tipo de control de E/S permite al usuario cambiar los atributos del archivo (y, en algunos casos, cambiar el dispositivo) utilizado en un programa sin necesidad de cambiar el programa.

En el lenguaje ILE COBOL/400, el nombre de archivo especificado en la entrada ASSIGNMENT-NAME de la cláusula ASSIGN de la entrada de control de archivos se utiliza para señalar al archivo. Este nombre de archivo señala a la descripción de archivo del sistema:

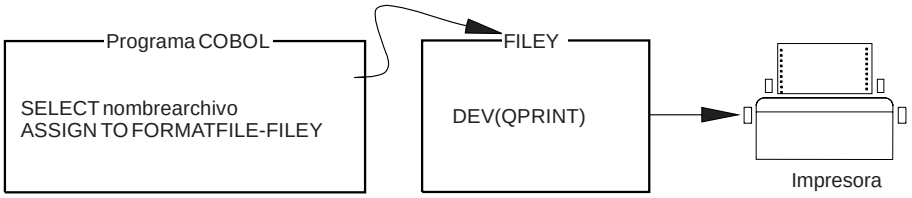


El nombre de dispositivo ILE COBOL/400 de la cláusula ASSIGN define las funciones ILE COBOL/400 que pueden procesarse en el archivo seleccionado. En la compilación, ciertas funciones ILE COBOL/400 son válidas sólo para un tipo de dispositivo ILE COBOL/400 específico; por lo que a esto se refiere, ILE COBOL/400 depende del dispositivo. A continuación encontrará ejemplos de dependencia de dispositivos:

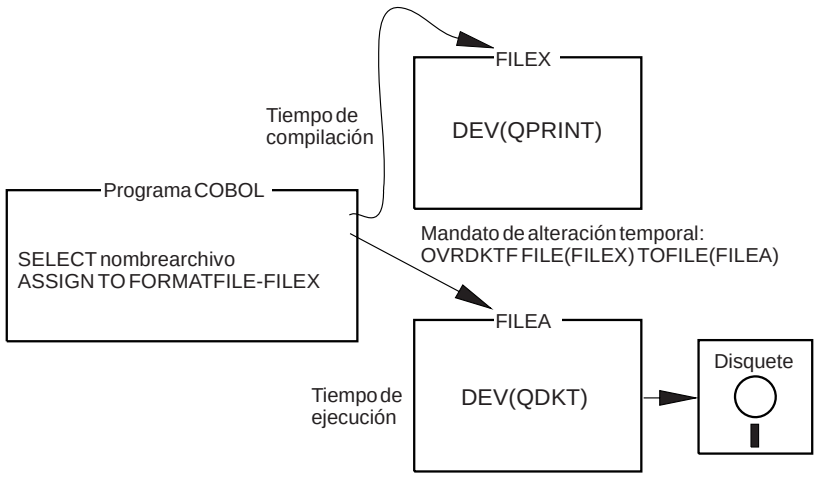
- Las operaciones SUBFILE sólo son válidas para un dispositivo WORKSTATION.
- Los indicadores sólo son válidos para dispositivos WORKSTATION o FORMATFILE.

- LINAGE sólo es válido para un dispositivo PRINTER.
- OPEN INPUT WITH NO REWIND sólo es válido para un dispositivo TAPEFILE.

Por ejemplo, supongamos que el nombre de archivo FILEY está asociado, en el programa ILE COBOL/400, con el dispositivo FORMATFILE. El dispositivo FORMATFILE es un tipo de dispositivo independiente. Por consiguiente, ninguna de las especificaciones de control de línea o de página es válida, por lo que la expresión ADVANCING no puede especificarse en la instrucción WRITE para un archivo FORMATFILE. Cuando se ejecuta el programa, el dispositivo de E/S real se especifica en la descripción de FILEY. Por ejemplo, el dispositivo puede ser una impresora; sólo se utilizarán el control de línea o de página por omisión o aquellos definidos en las DDS:



Los mandatos CL pueden utilizarse para alterar temporalmente un parámetro en la descripción de archivos especificada o para redirigir un archivo en la compilación o en la ejecución. La redirección de archivos permite al usuario especificar un archivo en la compilación y otro en la ejecución:



En el ejemplo anterior, el mandato Alterar temporalmente en archivo de disquete (OVRDKTF) permite al programa ejecutarse con un archivo de dispositivos completamente distinto al especificado en la compilación.

Nota: Los dispositivos FORMATFILE no pueden utilizarse para entrada. Alterar temporalmente una entrada/salida de un dispositivo que permite la entrada, como por ejemplo un dispositivo DISKETTE, a un dispositivo FORMATFILE puede provocar resultados impredecibles si se intenta una operación de entrada.

No todas las alteraciones temporales de archivos son válidas. En la ejecución se realiza una comprobación para asegurar que las especificaciones del programa ILE COBOL/400 son válidas para el archivo que se procesa. Si las especificaciones

que el programa ILE COBOL/400 transfiere en el bloque de control de archivos y la petición de E/S son incorrectas, se producirá una anomalía en la operación de E/S. El sistema operativo OS/400 permite realizar algunas redirecciones de archivos aunque el programa contenga los específicos de dispositivos. Por ejemplo, si el nombre de dispositivo ILE COBOL/400 es PRINTER y el archivo real que utiliza el programa no es una impresora, el sistema operativo ignorará las especificaciones de espaciado y salto de impresora ILE COBOL/400.

Existen otras redirecciones de archivo que el sistema operativo no permite y que pueden inutilizar el archivo. Por ejemplo, si el nombre de dispositivo ILE COBOL/400 es DATABASE o DISK y se especifica una operación READ por clave en el programa, el archivo se vuelve inutilizable si el archivo real que el programa utiliza no es un archivo de disco o de base de datos.

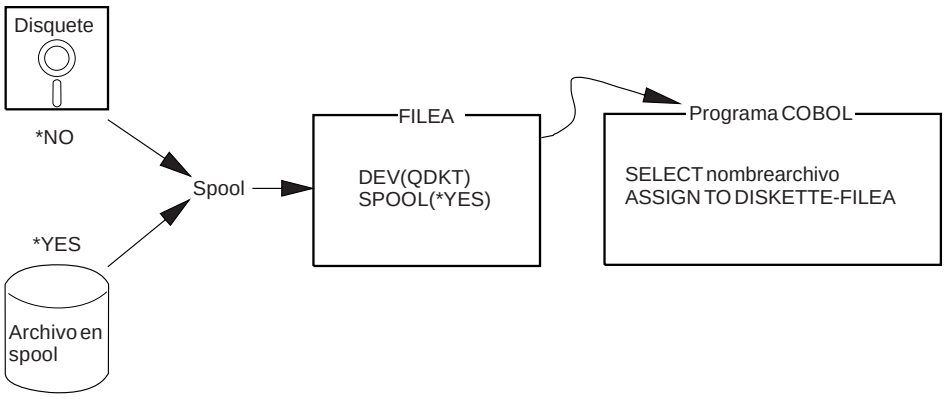
Especificación de spooling de entrada y salida

El sistema AS/400 proporciona funciones de spooling de entrada y salida. Cada descripción de archivo AS/400 contiene un atributo de spool que determina si se utiliza el spooling para el archivo en la ejecución. El programa ILE COBOL/400 no es consciente de que se está utilizando el spooling. El dispositivo físico real desde el que se lee un archivo o al que se graba un archivo está determinado por el lector o el transcriptor de spool. Vea *Gestión de datos* para obtener más información detallada sobre el spooling.

Spooling de entrada

El spooling de entrada sólo es válido para archivos de datos incorporados en trabajos por lotes. Si los datos de entrada leídos por ILE COBOL/400 provienen de un archivo en spool, ILE COBOL/400 no sabe desde qué dispositivo los datos se han puesto en spool.

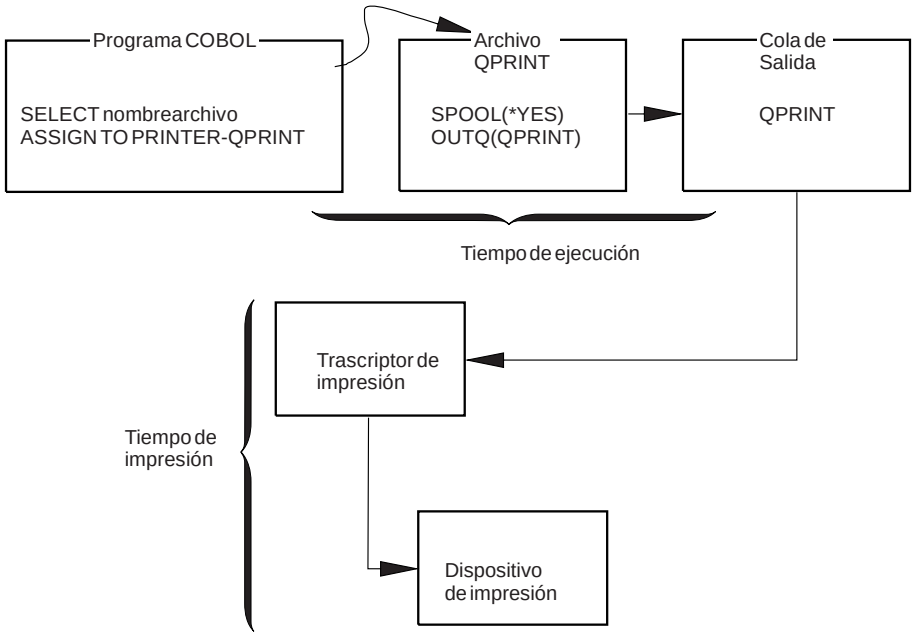
Los datos se leen desde un archivo incorporado en spool:



Vea *Gestión de datos* para obtener más información sobre los archivos de datos incorporados.

Spooling de salida

El spooling de salida es válido para trabajos por lotes y trabajos interactivos. La descripción del archivo especificado en ILE COBOL/400 por el nombre de sistema contiene la especificación para el spooling, tal como se muestra en el ejemplo siguiente:



Los mandatos de alteración temporal de archivos pueden utilizarse en la ejecución para alterar temporalmente las opciones de spooling especificadas en la descripción de archivos, como por ejemplo el número de copias a imprimir. Además, el soporte de spooling del AS/400 le permite redirigir un archivo después de que se haya ejecutado el programa. Por ejemplo, puede dirigir la salida de la impresora a un dispositivo distinto, por ejemplo un disquete.

Alteración temporal de atributos de archivos

Debe especificar cualquier alteración temporal antes de que el programa ILE COBOL/400 abra el archivo. El sistema utiliza el mandato de alteración temporal de archivos para determinar que se abra el archivo y los atributos del archivo. El ámbito de las alteraciones temporales de archivos es el nivel de llamada, el nivel de grupo de activación o el nivel de trabajo.

Para el ámbito del **nivel de llamada**, una alteración temporal emitida a un nivel de llamada concreto es efectiva para cualquier llamada situada después del nivel de llamada, independientemente de en qué grupo de activación se encuentren las llamadas, y su efecto finaliza cuando se devuelve el control para el nivel de llamada en el que se ha emitido la alteración temporal.

Para el ámbito del **nivel de grupo de activación**, la alteración temporal es aplicable a todos los objetos de programa que se ejecuten en ese grupo de activación y permanecerá activa hasta que el grupo de activación finalice o se elimine la alteración explícitamente.

Nota: En el Grupo de activación por omisión (*DFTACTGRP), cuando se especifica el nivel de grupo de activación, el ámbito real de la alteración temporal es el nivel de llamada.

Para el ámbito del **nivel de trabajo**, la alteración temporal es aplicable a todos los objetos de programa dentro del trabajo y permanece activa hasta que el trabajo finaliza o la alteración se elimina explícitamente.

Utilice el parámetro OVRSCOPE de cualquier mandato CL de alteración temporal para especificar el ámbito de la alteración.Si no especifica explícitamente el ámbito, el ámbito por omisión de la alteración temporal dependerá de dónde se emita la alteración. Si se emite desde el grupo de activación por omisión, el ámbito se encontrará en el nivel de llamada.Si se emite desde cualquier otro grupo de activación, el ámbito será el grupo de activación.

La forma más simple de alterar temporalmente un archivo es alterar algunos atributos del archivo. Por ejemplo, FILE(OUTPUT) con COPIES(2) se especifica cuando se crea un archivo de impresora. Entonces, antes de ejecutar el programa ILE COBOL/400, el número de copias impresas de la salida puede cambiarse a 3. El mandato de alteración temporal es el siguiente:

```
OVRPRTF FILE(OUTPUT) COPIES(3)
```

Redirección de entrada y salida de archivos

Otra forma de alterar temporalmente archivos es redirigir el programa ILE COBOL/400 para que acceda a un archivo distinto. Cuando la alteración redirige el programa a un archivo del *mismo* tipo (como por ejemplo un archivo de impresora a otro archivo de impresora), el archivo se procesa de la misma forma que el archivo original.

Cuando la alteración temporal redirige el programa a un archivo de tipo *distinto*, el archivo de alteración se procesa de la misma forma que se procesaría el archivo original. El sistema ignora las especificaciones dependientes de dispositivos en el programa ILE COBOL/400 que no son aplicables al dispositivo de alteración temporal.

No todas las redirecciones de archivo son válidas. Por ejemplo, un archivo indexado para un programa ILE COBOL/400 sólo puede alterarse temporalmente con otro archivo indexado con una vía de acceso por clave.

Para un archivo de base de datos, el proceso de miembros múltiples puede realizarse alterando temporalmente un archivo de base de datos para procesar todos los miembros. Tenga en cuenta las excepciones siguientes:

- Un archivo fuente de base de datos utilizado en la compilación de un programa ILE COBOL/400 no puede alterarse temporalmente para procesar todos los miembros. La especificación OVRDBF MBR(*ALL) provocará la terminación de la compilación.
- Un archivo de base de datos utilizado para una instrucción COPY no puede alterarse temporalmente para procesar todos los miembros. Especificar OVRDBF MBR(*ALL) hará que se ignore la instrucción COPY.

Debe asegurarse de que las alteraciones temporales de archivos se aplican correctamente. Para obtener más información sobre redirecciones de archivos válidas,

las características dependientes de dispositivos ignoradas y los valores por omisión asumidos, vea el manual *Gestión de datos*.

Bloqueo y liberación de archivos

El sistema operativo permite colocar un estado de bloqueo (exclusivo, lectura permitida exclusiva, compartido para actualización, compartido sin actualización o compartido para lectura) en un archivo utilizado durante un paso de trabajo. Puede colocar el archivo en un estado de bloqueo con el mandato Asignar objeto (ALCOBJ).

Por omisión, el sistema operativo coloca los siguientes estados de bloqueo en archivos de bases de datos cuando los archivos los abren programas ILE COBOL/400:

Tipo de APERTURA	Estado de bloqueo
INPUT	Compartido para lectura
I-O	Compartido para actualización
EXTEND	Compartido para actualización
OUTPUT	Compartido para actualización

El estado compartido para lectura permite a otro usuario abrir el archivo con un estado de bloqueo compartido para lectura, compartido para actualización, compartido para no actualización o lectura permitida exclusiva, pero el usuario no puede especificar la utilización exclusiva del archivo. El estado de bloqueo compartido para actualización permite a otro usuario abrir el archivo con un estado de bloqueo compartido para lectura o compartido para actualización.

El sistema operativo coloca el bloqueo compartido para lectura en el archivo de dispositivos y un estado de bloqueo de lectura permitida exclusiva en el dispositivo. Otro usuario puede abrir el archivo pero no puede utilizar el mismo dispositivo.

Nota: Cuando un programa ILE COBOL/400 abre un archivo físico para OUTPUT, ese archivo estará sujeto a un bloqueo exclusivo durante el período de tiempo necesario para borrar el miembro.

Para obtener más información sobre la asignación de recursos y estados de bloqueo, vea el manual *Gestión de datos*.

Bloqueo y liberación de registros

Cuando un programa ILE COBOL/400 realiza una operación READ en un archivo de base de datos y el archivo se abre para E-S, se coloca un bloqueo en ese registro para que ningún otro programa pueda actualizarlo. Es decir, otro programa puede leer el registro si abre un archivo para entrada, pero no si lo abre para E-S. De forma similar, después de una operación START satisfactoria para un archivo abierto en modalidad de E-S, se colocará un bloqueo en el registro en que se encuentra el archivo.

Para obtener información sobre la duración del bloqueo de registro con o sin control de compromiso, consulte la Tabla 15 en la página 312.

Para evitar que las instrucciones READ o START bloqueen los registros de los archivos abiertos en modalidad de E-S (actualización), puede utilizar la expresión NO LOCK. La instrucción READ WITH NO LOCK desbloquea los registros bloqueados por una instrucción READ o START anterior. Además, el registro leído por la instrucción READ WITH NO LOCK no se bloquea. La instrucción START WITH NO LOCK desbloquea los registros bloqueados por una instrucción START o READ anterior. Para obtener más información sobre esta expresión, consulte el apartado sobre las instrucciones READ y START en la *ILE COBOL/400 Reference*.

Para un archivo lógico basado en un archivo físico, el bloqueo se coloca en el registro del archivo físico. Si un archivo lógico está basado en más de un archivo físico, el bloqueo se coloca en un registro de cada archivo físico.

Este bloqueo es aplicable no sólo a otros programas, sino también al programa original si intenta actualizar el mismo registro físico subyacente a través de un segundo archivo.

Nota: Cuando se abre un archivo con organización indexada o relativa para E-S utilizando acceso aleatorio o dinámico, una operación de E/S errónea en cualquiera de los verbos de E/S excepto WRITE, también desbloquea el registro. Una operación WRITE no se considera una operación de actualización; por consiguiente, el bloqueo del registro no se libera.

Para obtener más información sobre la liberación de registros de bases de datos leídos para actualizar, vea el manual *Gestión de datos*.

Compartimiento de una vía de acceso de datos abierta para acceder a un archivo

Si ya ha abierto un archivo mediante otro programa en su paso de direccionamiento, el programa ILE COBOL/400 puede utilizar la misma Vía de datos abierta (ODP) para acceder al archivo.

Nota: Los pasos de direccionamiento se describen en la Gestión de Trabajos; generalmente un trabajo sólo contiene un paso de direccionamiento.

Las normas siguientes son aplicables a las ODP compartidas:

1. Debe especificar SHARE(*YES) en el mandato que crea el archivo (CRTxxxF), en un mandato de cambio (CHGxxxF) o en un mandato de alteración temporal (OVRxxxF) del archivo.
2. Después de que un programa abre por primera vez un archivo con una ODP compartida y permanece abierto, las siguientes operaciones OPEN compartidas dentro del mismo paso de direccionamiento se ejecutan más rápido que las operaciones OPEN estándares. No afecta a la velocidad de otras operaciones de E/S.
3. La utilización del archivo dentro de los distintos programas debe ser coherente. Otros programas que utilizan el mismo archivo compartido afectarán a la posición del archivo actual cuando realicen operaciones de E/S en el archivo.

Desbloqueo de registros de entrada y bloqueo de registros de salida

Un **bloque** contiene más de un registro. El desbloqueo de los registros de entrada y el bloqueo de los registros de salida se produce bajo las condiciones siguientes:

1. Se especifica *NOBLK en el parámetro OPTION de los mandatos CRTCBLMOD o CRTBNDCBL (con o sin una cláusula BLOCK CONTAINS) y se cumplen **todas** las condiciones siguientes:

   a. Se especifica ACCESS IS SEQUENTIAL para el archivo.

   b. El archivo se abre sólo para INPUT o OUTPUT en ese programa.

   c. El archivo se asigna a DISK, DATABASE, DISKETTE o TAPEFILE.

   d. No se especifica ninguna instrucción START para el archivo.

   Para una organización RELATIVE, no se realiza el bloqueo para OPEN OUTPUT.

   Si especifica BLOCK CONTAINS, se ignorará. El sistema determina el número de registros por bloque.

2. Se especifica *BLK en el parámetro OPTION de los mandatos CRTCBLMOD o CRTBNDCBL con BLOCK CONTAINS y se cumplen **todas** las condiciones siguientes:

   a. Se especifica ACCESS IS SEQUENTIAL o ACCESS IS DYNAMIC para el archivo.

   b. El archivo se abre sólo para INPUT o OUTPUT en ese programa.

   c. El archivo se asigna a DISK, DATABASE, DISKETTE o TAPEFILE.

   Para una organización RELATIVE, no se realiza el bloqueo para OPEN OUTPUT.

   La cláusula BLOCK CONTAINS controla el número de registros por bloque. En el caso de archivos DISKETTE, el sistema siempre determina el número de registros por bloque.

Incluso cuando se cumplen todas las condiciones anteriores, ciertas restricciones del OS/400 pueden hacer que el bloqueo y el desbloqueo no surtan efecto. En estos casos, las mejoras de rendimiento no se notarán.

Si está utilizando archivos de índices accedidos dinámicamente, puede utilizar READ PRIOR y READ NEXT para realizar el bloqueo. Utilizando READ PRIOR y READ NEXT para realizar el bloque, no puede cambiar de dirección mientras queden registros en el bloque. Para eliminar los registros de un bloque, especifique una operación aleatoria, como por ejemplo READ o START aleatorios o utilice una operación READ FIRST o READ LAST secuencial.

Si se produce un cambio de dirección no permitido, el estado del archivo será 9U. No se permitirá ninguna E/S más hasta que se cierre el archivo y se vuelva abrir.

Puede alterar temporalmente el bloqueo en la ejecución especificando SEQONLY(*NO) para el mandato OVRDBF.

Para archivos de discos y de bases de datos, cuando utilice BLOCK CONTAINS y el factor de bloqueo especificado o calculado es cero, el sistema determinará el factor de bloqueo.

En ciertos casos el factor de bloqueo especificado puede cambiar. Vea el manual *DB2/400 Programación de la base de datos* para obtener más información sobre estas situaciones.

Cuando se graba o lee un bloque de registros, el área de realimentación de E-S contiene el número de registros de ese bloque. El área I-O-FEEDBACK no se actualiza después de cada lectura o grabación de archivos donde ILE COBOL/400 bloquea y desbloquea múltiples registros. Se actualiza al leer o grabar el próximo bloque.

Para los archivos de bases de datos con bloqueo activado, puede que no vea todos los cambios mientras se producen si estos se realizan en distintos programas. Para obtener una descripción del efecto del bloqueo en los cambios realizados en los archivos de bases de datos, vea el tema sobre el proceso sólo secuencial en el manual *DB2/400 Programación de la base de datos*.

Utilización del estado de archivo y de áreas de realimentación

Para transferir datos de las áreas OPEN-FEEDBACK y I-O-FEEDBACK asociadas con un archivo abierto a un identificador, utilice la instrucción ACCEPT de Formato 3. Consulte el apartado “Instrucción ACCEPT” de la publicación *ILE COBOL/400 Reference* para obtener más información sobre la especificación de esta instrucción.

FILE STATUS

Cuando se especifica la cláusula FILE STATUS, el sistema traslada un valor al elemento de datos de clave de estado después de cada petición de entrada/salida que haga referencia, explícita o implícitamente, a este archivo. Este valor de 2 caracteres indica el estado de ejecución de la instrucción. Cuando los registros de entrada se desbloquean y los registros de salida se bloquean, los valores de estado de archivo causados por las excepciones OS/400 sólo se establecen cuando se procesa un bloque. Para obtener más información sobre el bloqueo de registros, consulte “Desbloqueo de registros de entrada y bloqueo de registros de salida” en la página 308.

Área OPEN-FEEDBACK

El área OPEN-FEEDBACK forma parte de la vía de datos abiertos (DDP) que contiene información sobre la operación OPEN.Esta información se establece durante el proceso OPEN y está disponible siempre que el archivo esté abierto.

Esta área proporciona información sobre el archivo que el programa está utilizando.Contiene:

- Información sobre el archivo abierto actualmente, como por ejemplo el nombre de archivo y el tipo de archivo
- Información que depende del tipo de archivo abierto, como por ejemplo el tamaño de la impresora, el tamaño de la pantalla, las etiquetas de disquetes o las etiquetas de cintas.

Nota: Los archivos OPTIONAL INPUT abiertos satisfactoriamente no tendrá ninguna información sobre el área OPEN-FEEDBACK.

Área I-O-FEEDBACK

El sistema actualiza el área I-O-FEEDBACK cada vez que se transfiere un bloque entre el sistema operativo y el programa.Un bloque puede contener uno o más registros.

El área I-O-FEEDBACK no se actualiza después de cada operación de lectura o grabación para archivos en los cuales COBOL bloquea y desbloquea múltiples registros.Si la información de I-O-FEEDBACK es necesaria después de cada operación de lectura o grabación del programa, el usuario puede realizar una de las operaciones siguientes:

- Evitar que el compilador genere código de bloqueo y desbloqueo, no cumpliendo una las condiciones listadas en “Desbloqueo de registros de entrada y bloqueo de registros de salida” en la página 308.
- Especificar SEQONLY(*NO) en el mandato CL Alterar temporalmente archivo de base de datos (OVRDBF).

Evitar que el compilador genere código de bloqueo y desbloqueo es más eficaz que especificar SEQONLY(*NO).

Incluso cuando el compilador genera código de bloqueo y desbloqueo, ciertas restricciones OS/400 pueden provocar que no se realice el bloqueo ni el desbloqueo.En estos casos, no se notará una mejora de rendimiento. Sin embargo, el área I-O-FEEDBACK se actualizará después de cada operación de lectura o grabación.

El área I-O-FEEDBACK contiene información sobre la última operación de E-S satisfactoria, como por ejemplo: nombre del dispositivo, tipo de dispositivo, carácter AID e información de error para algunos dispositivos. Esta área está formada por una área común y una área dependiente de dispositivo. El área dependiente de dispositivo varía en longitud y en contenido, según el tipo de dispositivo al que esté asociado el archivo. Esta área sigue a la área común I-O-FEEDBACK y puede obtenerse especificando el identificador de recepción con suficiente capacidad para incluir el área común y el área dependiente de dispositivo apropiada.

Vea el manual *Gestión de datos* para obtener el diseño y la descripción de las áreas de datos contenidas en las áreas OPEN-FEEDBACK y I-O-FEEDBACK.

Utilización del control de compromiso

El control de compromiso es una función que permite:

- La sincronización de los cambios realizados a archivos de bases de datos dentro del mismo trabajo.
- La cancelación de cambios que no deben introducirse permanentemente en la base de datos.
- El bloqueo de registros que se estén cambiando hasta que se hayan finalizado los cambios.
- Las técnicas para recuperarse de una anomalía de trabajo o de sistema.

En algunas aplicaciones, es conveniente sincronizar los cambios realizados en los registros de bases de datos. Si el programa determina que los cambios son válidos, entonces los cambios se hacen permanentes a la base de datos (se procesa una instrucción COMMIT). Si los cambios no son válidos o si se produce

un problema durante el proceso, pueden cancelarse (se procesa una instrucción ROLLBACK). (Cuando se elimina un archivo después de haber sido abierto para OUTPUT, el proceso de una instrucción ROLLBACK no restaura los registros eliminados en el archivo). Los cambios realizados en registros de un archivo que *no* estén bajo control de compromiso son siempre permanentes. Dichos cambios nunca se ven afectados por las instrucciones COMMIT o ROLLBACK siguientes.

Cada punto donde se procesa satisfactoriamente una instrucción COMMIT o ROLLBACK es un límite de compromiso. (Si aún no se ha emitido ninguna instrucción COMMIT o ROLLBACK en un programa, se creará un límite de control la primera vez que se abra un archivo bajo el control de compromiso). El compromiso o la retrotracción de cambios sólo afecta a los cambios realizados desde el límite de compromiso anterior.

La sincronización de cambios en los límites de compromiso facilita los procedimientos de reinicio o recuperación después de una anomalía. Para obtener más información, vea “Recuperación después de una anomalía” en la página 281.

Cuando el control de compromiso se utiliza para archivos de bases de datos, los registros en dichos archivos están sujetos a uno de los siguientes niveles de bloqueo:

- **nivel de bloqueo alto**

  Un nivel de bloqueo alto lo especifica el parámetro LCKLVL(*ALL) del mandato CL Iniciar control de compromiso (STRCMTCTL). Con un nivel de bloqueo alto (*ALL), *todos* los registros a los que los archivos han accedido bajo el control de compromiso, tanto para entrada como para salida, se bloquean hasta que se procesa una instrucción COMMIT o ROLLBACK satisfactoria.

- **nivel de bloqueo de estabilidad de cursor**

  Un nivel de bloqueo de estabilidad del cursor se especifica mediante el parámetro LCKLVL(*CS) del mandato CL Iniciar control de compromiso (STRCMTCTL). Con un nivel de bloqueo de estabilidad de cursor (*CS), se bloquea cada registro al que han accedido los archivos abiertos bajo el control de compromiso.Un registro que se haya leído, no cambiado ni eliminado, se desbloquea cuando se lee un registro distinto.Los registros cambiados, añadidos o eliminados se bloquean hasta que se procesa una instrucción COMMIT o ROLLBACK satisfactoriamente.

- **nivel de bloqueo bajo**

  Un nivel de bloqueo bajo lo especifica el parámetro LCKLVL(*CHG) del mandato CL Iniciar control de compromiso (STRCMTCTL). Con un nivel de bloqueo bajo (*CHG), se bloquea cada lectura de registro para actualización (para un archivo abierto bajo el control de compromiso).Si se cambia, añade o elimina un registro, éste permanece bloqueado hasta que se procesa una instrucción COMMIT o ROLLBACK satisfactoria. Los registros a los que se a accedido para operaciones de actualización, pero que se liberan sin haberlos cambiado, se desbloquean.

Un registro bloqueado sólo puede modificarse dentro del mismo trabajo y a través del mismo archivo físico o lógico.

El nivel de bloqueo también rige si los registros bloqueados pueden leerse. Con un nivel de bloqueo alto (*ALL), no puede leer los registros bloqueados en un archivo de base de datos. Con un nivel de bloqueo bajo (*CHG), puede leer los

registros bloqueados en un archivo de base de datos, siempre que el archivo esté abierto como INPUT en su trabajo o como I-O y se utilice READ WITH NO LOCK.

Otros trabajos, donde los archivos *no* se encuentran bajo el control de compromiso, siempre pueden leer registros bloqueados, independientemente del nivel de bloqueo utilizado, siempre que los archivos se abran como INPUT. Como en algunos casos es posible que otros trabajos lean los registros bloqueados, se puede acceder a los datos *antes de que se comprometan permanentemente a una base de datos*. Si se procesa una instrucción ROLLBACK *después* de que otro trabajo haya leído registros bloqueados, los datos a los que se ha accedido no reflejarán el contenido de la base de datos.

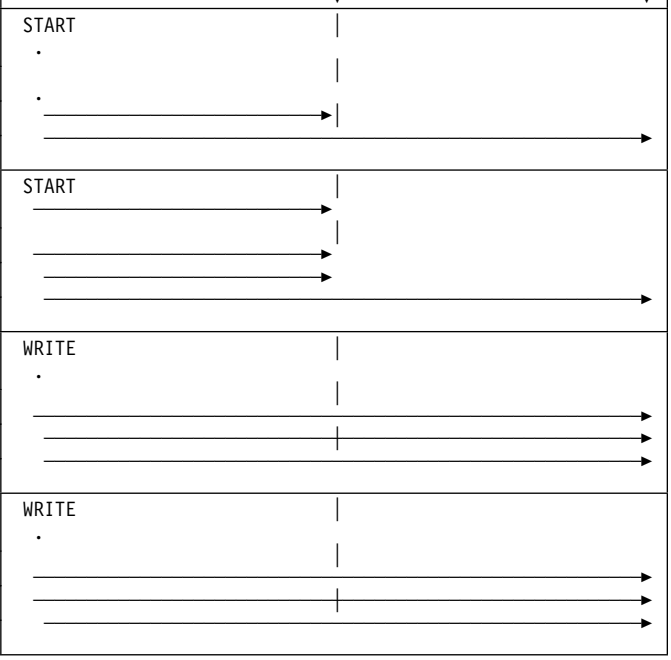
La Tabla 15 muestra las consideraciones sobre bloqueo de registros para archivos con y sin control de compromiso.

Tabla 15 (Página 1 de 2). Consideraciones sobre el bloqueo de registros con y sin control de compromiso

VERBO	MODALIDAD APERTURA	NIVEL BLOQUEO		DURACIÓN DE BLOQUEO REGISTROS
				Próxima operación / COMMIT o de actualización / ROLLBACK
DELETE	I-O	Sin control de compromiso		DELETE
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
READ	INPUT	Sin control de compromiso		READ
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
READ WITH NO LOCK	I-O	Sin control de compromiso		READ
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
READ	I-O	Sin control de compromiso		READ
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
REWRITE	I-O	Sin control de compromiso		REWRITE
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
START	INPUT	Sin control de compromiso		START
		Con control de compromiso	*CHG	
			*CS	
			*ALL	

Tabla 15 (Página 2 de 2). Consideraciones sobre el bloqueo de registros con y sin control de compromiso

VERBO	MODALIDAD APERTURA	NIVEL BLOQUEO		DURACIÓN DE BLOQUEO REGISTROS
				Próxima operación / COMMIT o de actualización / ROLLBACK
START WITH NO LOCK	I-O	Sin control de compromiso		START
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
START	I-O	Sin control de compromiso		START
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
WRITE	I-O	Sin control de compromiso		WRITE
		Con control de compromiso	*CHG	
			*CS	
			*ALL	
WRITE	OUTPUT	Sin control de compromiso		WRITE
		Con control de compromiso	*CHG	
			*CS	
			*ALL	



Nota: Las operaciones de actualización incluyen la realización de operaciones START, READ, REWRITE o DELETE para el mismo archivo (independientemente de si es satisfactoria o no) y la finalización del archivo.Una operación WRITE no se considera una operación de actualización; por lo tanto, no se establecerá un bloqueo mediante la operación WRITE.

Un archivo bajo el control de compromiso puede cerrarse o abrirse sin que afecte al estado de los cambios realizados desde el último límite de compromiso. Aún debe emitirse una instrucción COMMIT para hacer que los cambios sean permanentes o una instrucción ROLLBACK para cancelar los cambios. Una instrucción COMMIT, al procesarla, deja los archivos en el mismo estado abierto o cerrado en que estaban antes del proceso.

Todos los archivos bajo el control de compromiso dentro del mismo trabajo deben registrarse por diario en el mismo diario. Para obtener más información sobre la gestión de diarios y sus funciones relacionadas, y sobre el control de compromiso, consulte el manual *Backup and Recovery – Advanced*.

El control de compromiso también debe especificarse fuera de ILE COBOL/400 mediante el lenguaje de control OS/400 (CL). El mandato Iniciar control de compromiso (STRCMTCTL) establece la posibilidad del control de compromiso y el nivel de bloqueo de registro en nivel alto (*ALL), nivel de estabilidad de cursor (*CS) o nivel bajo (*CHG).

Cuando se arranca el control de compromiso utilizando el mandato STRCMTCTL, el sistema crea una **definición de compromiso**. Cada definición de compromiso sólo la conoce el trabajo o el grupo de activación del trabajo que emitió el mandato STRCMTCTL, según el ámbito del control de compromiso. La definición de com-

promiso contiene información perteneciente a los recursos cambiados bajo el control de compromiso de este trabajo o grupo de activación dentro del trabajo. El sistema mantiene la información sobre el control de compromiso de la definición de compromiso a medida que cambian los recursos de compromiso.

El mandato STRCMTCTL no inicia automáticamente el control de compromiso para un archivo. Dicho archivo también debe especificarse en la cláusula COMMITMENT CONTROL del párrafo I-O-CONTROL dentro del programa ILE COBOL/400. El entorno del control de compromiso normalmente se finaliza utilizando el mandato Finalizar control de compromiso (ENDCMTCTL). Esto cancela los cambios no comprometidos para los archivos de bases de datos bajo el control de compromiso. (Se procesa una operación ROLLBACK implícita). Consulte la publicación *CL Reference* para obtener más información sobre los mandatos STRCMTCTL y ENDCMTCTL.

Para obtener más información sobre el control de compromiso, vea el manual *Backup and Recovery – Advanced*.

Nota: La capacidad de evitar la lectura de datos no comprometidos que se han cambiado es una función del control de compromiso y sólo está disponible si está ejecutando bajo el control de compromiso. El soporte de base de datos normal (no comprometido) no lo cambia la extensión de control de compromiso, y permite la lectura de registros bloqueados al leer un archivo abierto sólo para entrada. Intente utilizar los archivos coherentemente. Normalmente, los archivos deben ejecutarse siempre bajo el control de compromiso o no deben ejecutarse nunca bajo dicho control.

Ámbito de control de compromiso

Los objetos de programa que se ejecutan dentro de un trabajo pueden iniciar y utilizar las definiciones de compromiso múltiples. Cada definición de compromiso para un trabajo identifica una transacción separada con recursos asociados. Estos recursos puede comprometerse o retrotraerse independientemente de todas las otras definiciones de compromiso iniciadas para el trabajo.

El **ámbito** para una definición de compromiso indica qué programas ejecutados en el trabajo utilizan esa definición de compromiso. Las definiciones de compromiso tienen dos ámbitos:

- a nivel de grupo de activación
- a nivel de trabajo

Especifique el ámbito para una definición de compromiso en el parámetro CMTSCOPE del mandato STRCMTCTL.

El ámbito por omisión para una definición de compromiso es para el grupo de activación del programa que emite el mandato STRCMTCTL. Sólo los objetos de programa que se ejecutan en ese grupo de activación utilizarán esa definición de compromiso.La definición de compromiso iniciada en el nivel de grupo de activación para el grupo de activación por omisión OPM se conoce como la definición de compromiso del grupo de activación por omisión (*DFTACTGRP). Cada grupo de activación puede tener su propia definición de compromiso.

Una definición de compromiso también puede estar en el ámbito del trabajo. Cualquier objeto de programa que se ejecute en un grupo de activación que no haya iniciado una definición de compromiso en el nivel de grupo de activación, utiliza la

definición de compromiso del nivel de trabajo.Esto sucede si la definición de compromiso del nivel de trabajo ya ha sido iniciado por otro objeto de programa para un trabajo.Sólo puede iniciarse una única definición de compromiso de nivel de trabajo para un trabajo.

Para un grupo de activación dado, los objetos de programa que se ejecutan en ese grupo de activación sólo pueden utilizar una única definición de compromiso.Los objetos de programa que se ejecutan dentro de un grupo de activación pueden utilizar la definición de compromiso en el nivel de trabajo o en el nivel de grupo de activación.Sin embargo, no pueden utilizar las definiciones de compromiso al mismo tiempo.

Cuando un programa ILE COBOL/400 realiza una operación de control de compromiso, no indica directamente qué definición de compromiso debe utilizarse para la petición.En su lugar, el sistema determina qué definición de compromiso debe utilizar basándose en el grupo de activación en que se está ejecutando el objeto de programa de petición.

Los archivos asociados con una definición de compromiso en el ámbito de un grupo de activación ILE se cerrarán y se comprometerán implícitamente cuando el grupo de activación finalice normalmente.Cuando un grupo de activación finalice de forma anómala, los archivos asociados con una definición de compromiso en el ámbito del grupo de activación se retrotraerán y se cerrarán.

Consulte el manual *ILE Conceptos* para obtener más información sobre el ámbito del control de compromiso.

Ejemplo de utilización de control de compromiso

La Figura 77 en la página 317 muestra una posible utilización del control de compromiso en un entorno bancario. El programa procesa transacciones para transferir fondos de una cuenta a otra. Si no se produce ningún problema durante la transacción, los cambios se comprometerán en el archivo de base de datos. Si no puede realizarse la transferencia debido a un número de cuenta incorrecto o a fondos insuficientes, se emitirá una operación ROLLBACK para cancelar los cambios.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ACCOUNT MASTER PHYSICAL FILE -- ACCTMST
     A
     A                                     UNIQUE
     A          R ACCNTREC
     A            ACCNTKEY       5S
     A            NAME          20
     A            ADDR          20
     A            CITY          20
     A            STATE          2
     A            ZIP            5S
     A            BALANCE       10S 2
     A          K ACCNTKEY
```

Figura 75. Ejemplo de utilización del control de compromiso -- DDS de archivo maestro de cuentas

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* PROMPT SCREEN FILE NAME 'ACCTFMTS'
     A*
     A                                     1 INDARA
     A          R ACCTPMT
     A                                       TEXT('SOLICITUD CUENTA CLIENTE')
     A
     A                                       CA01(15 'FIN DE PROGRAMA')
     A                                       PUTRETAIN OVERLAY
     A                                   1  3'ACTUALIZAR MAESTRO DE CUENTAS'
     A                                   3  3'NÚMERO DE CUENTA ORIGEN'
     A            ACCTFROM       5Y 0I  3 23CHECK(ME)
     A  99                                   ERRMSG('NÚMERO DE CUENTA ORIGEN +
     A                                       NO VÁLIDO' 99)
     A  98                                   ERRMSG('FONDOS INSUFICIENTES EN +
     A                                       CUENTA ORIGEN' 98)
     A                                   4  3'NÚMERO DE CUENTA DESTINO'
     A            ACCTTO         5Y 0I  4 23CHECK(ME)
     A  97                                   ERRMSG('NÚMERO DE CUENTA DESTINO +
     A                                       NO VÁLIDO' 97)
     A                                   5  3'CANTIDAD TRANSFERIDA'
     A            TRANSAMT      10Y02I  5 23
     A          R ERRFMT
     A  96                               6  5'FILE STATUS NO VÁLIDO'
     A  95                               7  5'CLAVE NO VÁLIDA EN REWRITE'
     A  94                               8  5'FIN DE ARCHIVO EN READ'
```

Figura 76. Ejemplo de utilización del control de compromiso -- DDS de pantalla de solicitud

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ACCOUNT          AS400SYS   96/07/04 11:17:32        Página     2

                                 F u e n t e

 INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     ACCOUNT.
    3     000300 ENVIRONMENT DIVISION.
    4     000400 CONFIGURATION SECTION.
    5     000500 SOURCE-COMPUTER. IBM-AS400.
    6     000600 OBJECT-COMPUTER. IBM-AS400.
    7     000700 INPUT-OUTPUT SECTION.
    8     000800 FILE-CONTROL.
    9     000900     SELECT ACCOUNT-FILE  ASSIGN TO DATABASE-ACCTMST
   11     001000         ORGANIZATION IS INDEXED
   12     001100         ACCESS IS DYNAMIC
   13     001200         RECORD IS EXTERNALLY-DESCRIBED-KEY
   14     001300         FILE STATUS IS ACCOUNT-FILE-STATUS.
   15     001400     SELECT DISPLAY-FILE ASSIGN TO WORKSTATION-ACCTFMTS-SI  1
   17     001500         ORGANIZATION IS TRANSACTION.
          001600*********************************************************************
   18     001700 I-O-CONTROL.
   19     001800     COMMITMENT CONTROL FOR ACCOUNT-FILE.  2
          001900*********************************************************************
   20     002000 DATA DIVISION.
   21     002100 FILE SECTION.
   22     002200 FD  ACCOUNT-FILE.
   23     002300 01  ACCOUNT-RECORD.
          002400     COPY DDS-ALL-FORMATS OF ACCTMST.
   24    +000001       05  ACCTMST-RECORD PIC X(82).                                <-ALL-FMTS
         +000002*    FORMATO E-S:ACCNTREC   DE ARCHIVO ACCTMST    DE BIBLIOTECA TESTLIB      <-ALL-FMTS
         +000003*                                                                   <-ALL-FMTS
         +000004*LAS DEFINICIONES CLAVE PARA FORMATO DE REGISTRO ACCNTREC                 <-ALL-FMTS
         +000005*  NÚMERO              NOMBRE              RECUPERACIÓN    SECALT      <-ALL-FMTS
         +000006*   0001   ACCNTKEY                       ASCENDENTE       NO         <-ALL-FMTS
   25    +000007       05  ACCNTREC      REDEFINES ACCTMST-RECORD.                   <-ALL-FMTS
   26    +000008           06 ACCNTKEY              PIC S9(5).                       <-ALL-FMTS
   27    +000009           06 NAME                  PIC X(20).                       <-ALL-FMTS
   28    +000010           06 ADDR                  PIC X(20).                       <-ALL-FMTS
   29    +000011           06 CITY                  PIC X(20).                       <-ALL-FMTS
   30    +000012           06 STATE                 PIC X(2).                        <-ALL-FMTS
   31    +000013           06 ZIP                   PIC S9(5).                       <-ALL-FMTS
   32    +000014           06 BALANCE               PIC S9(8)V9(2).                  <-ALL-FMTS
          002500
   33     002600 FD  DISPLAY-FILE.
   34     002700 01  DISPLAY-REC.
          002800     COPY DDS-ALL-FORMATS OF ACCTFMTS.
   35    +000001       05  ACCTFMTS-RECORD PIC X(20).                               <-ALL-FMTS
         +000002*  FORMATO ENTRADA:ACCTPMT    DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB    <-ALL-FMTS
         +000003*                    SOLICITUD DE CUENTA DEL CLIENTE                 <-ALL-FMTS
   36    +000004       05  ACCTPMT-I     REDEFINES ACCTFMTS-RECORD.                 <-ALL-FMTS
   37    +000005           06 ACCTFROM              PIC S9(5).                       <-ALL-FMTS
   38    +000006           06 ACCTTO                PIC S9(5).                       <-ALL-FMTS
   39    +000007           06 TRANSAMT              PIC S9(8)V9(2).                  <-ALL-FMTS
         +000008* FORMATO SALIDA:ACCTPMT    DE ARCHIVO ACCTFMTS    DE BIBLIOTECA TESTLIB    <-ALL-FMTS
         +000009*                         SOLICITUD DE CUENTA DEL CLIENTE           <-ALL-FMTS
         +000010*      05  ACCTPMT-O     REDEFINE REGISTRO ACCTFMTS.                <-ALL-FMTS
         +000011*  FORMATO ENTRADA:ERRFMT   DE ARCHIVO ACCTFMTS    DE BIBLIOTECA TESTLIB    <-ALL-FMTS
         +000012*                                                                   <-ALL-FMTS
         +000013*      05  ERRFMT-I      REDEFINE REGISTRO ACCTFMTS.                <-ALL-FMTS
```

Figura 77 (Parte 1 de 4). Ejemplo de utilización de control de compromiso

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ACCOUNT          AS400SYS   96/07/04 11:17:32         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


        +000014* FORMATO SALIDA:ERRFMT   DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB        <-ALL-FMTS
        +000015*                                                                          <-ALL-FMTS
        +000016*      05  ERRFMT-O      REDEFINE REGISTRO ACCTFMTS.                       <-ALL-FMTS
         002900
   40    003000 WORKING-STORAGE SECTION.
   41    003100 77  ACCOUNT-FILE-STATUS           PIC X(2).
   42    003200 77  IND-ON                       PIC 1    VALUE B"1".
   43    003300 77  IND-OFF                      PIC 1    VALUE B"0".
   44    003400 01  DISPFILE-INDICS.
         003500     COPY DDS-ALL-FORMATS-INDIC OF ACCTFMTS.  3
   45   +000001       05  ACCTFMTS-RECORD.                                              <-ALL-FMTS
        +000002*  FORMATO ENTRADA:ACCTPMT    DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB     <-ALL-FMTS
        +000003*                    SOLICITUD DE CUENTA DEL CLIENTE                       <-ALL-FMTS
   46   +000004           06 ACCTPMT-I-INDIC.                                            <-ALL-FMTS
   47   +000005                07 IN15             PIC 1  INDIC 15.                      <-ALL-FMTS
        +000006*                           FIN DEL PROGRAMA                              <-ALL-FMTS
   48   +000007                07 IN97             PIC 1  INDIC 97.                      <-ALL-FMTS
        +000008*                           NÚMERO DE CUENTA ORIGEN NO VÁLIDO             <-ALL-FMTS
   49   +000009                07 IN98             PIC 1  INDIC 98.                      <-ALL-FMTS
        +000010*                           FONDOS INSUFICIENTES EN CUENTA ORIGEN         <-ALL-FMTS
   50   +000011                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
        +000012*                           NÚMERO DE CUENTA DESTINO NO VÁLIDO            <-ALL-FMTS
        +000013* FORMATO SALIDA:ACCTPMT    DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB   <-ALL-FMTS
        +000014*                           SOLICITUD DE CUENTA DE CLIENTE                <-ALL-FMTS
   51   +000015           06 ACCTPMT-O-INDIC.                                            <-ALL-FMTS
   52   +000016                07 IN97             PIC 1  INDIC 97.                      <-ALL-FMTS
        +000017*                           NÚMERO DE CUENTA ORIGEN NO VÁLIDO             <-ALL-FMTS
   53   +000018                07 IN98             PIC 1  INDIC 98.                      <-ALL-FMTS
        +000019*                           FONDOS INSUFICIENTES EN CUENTA ORIGEN         <-ALL-FMTS
   54   +000020                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
        +000021*                           NÚMERO DE CUENTA DESTINO NO VÁLIDO            <-ALL-FMTS
        +000022*  FORMATO ENTRADA:ERRFMT     DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB     <-ALL-FMTS
        +000023*                                                                          <-ALL-FMTS
        +000024*          06 ERRFMT-I-INDIC.                                            <-ALL-FMTS
        +000025* FORMATO SALIDA:ERRFMT     DE ARCHIVO ACCTFMTS   DE BIBLIOTECA TESTLIB   <-ALL-FMTS
        +000026*                                                                          <-ALL-FMTS
   55   +000027           06 ERRFMT-O-INDIC.                                            <-ALL-FMTS
   56   +000028                07 IN94             PIC 1  INDIC 94.                      <-ALL-FMTS
   57   +000029                07 IN95             PIC 1  INDIC 95.                      <-ALL-FMTS
   58   +000030                07 IN96             PIC 1  INDIC 96.                      <-ALL-FMTS
         003600
   59    003700 PROCEDURE DIVISION.
   60    003800 DECLARATIVES.
         003900 ACCOUNT-ERR-SECTION SECTION.
         004000     USE AFTER STANDARD EXCEPTION PROCEDURE ON ACCOUNT-FILE.
         004100 ACCOUNT-ERR-PARAGRAPH.
   61    004200     IF ACCOUNT-FILE-STATUS IS NOT EQUAL "23" THEN
   62    004300         MOVE IND-ON TO IN96 OF ERRFMT-O-INDIC  4
         004400     ELSE
   63    004500         MOVE IND-ON TO IN95 OF ERRFMT-O-INDIC  5
         004600     END-IF
   64    004700     WRITE DISPLAY-REC FORMAT IS "ERRFMT"
         004800          INDICATORS ARE ERRFMT-O-INDIC
         004900     END-WRITE
         005000                                                                                  94/03/03
   65    005100     CLOSE DISPLAY-FILE
         005200           ACCOUNT-FILE.
```

Figura 77 (Parte 2 de 4). Ejemplo de utilización de control de compromiso

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ACCOUNT          AS400SYS   96/07/04 11:17:32         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   66     005300     STOP RUN.
          005400
          005500 DISPLAY-ERR-SECTION SECTION.
          005600     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPLAY-FILE.
          005700 DISPLAY-ERR-PARAGRAPH.
   67     005800     MOVE IND-ON TO IN94 OF ERRFMT-O-INDIC
   68     005900     WRITE DISPLAY-REC FORMAT IS "ERRFMT"
          006000          INDICATORS ARE ERRFMT-O-INDIC
          006100     END-WRITE
   69     006200     CLOSE DISPLAY-FILE
          006300           ACCOUNT-FILE.
   70     006400     STOP RUN.
          006500 END DECLARATIVES.
          006600
          006700 MAIN-PROGRAM SECTION.
          006800 MAINLINE.
   71     006900     OPEN I-O DISPLAY-FILE
          007000          I-O ACCOUNT-FILE.
   72     007100     MOVE ZEROS TO ACCTPMT-I-INDIC
          007200                   ACCTPMT-O-INDIC.
   73     007300     PERFORM WRITE-READ-DISPLAY.
   74     007400     PERFORM VERIFY-ACCOUNT-NO UNTIL IN15 EQUAL IND-ON.
   75     007500     CLOSE DISPLAY-FILE
          007600           ACCOUNT-FILE.
   76     007700     STOP RUN.
          007800
          007900 VERIFY-ACCOUNT-NO.
   77     008000     PERFORM VERIFY-TO-ACCOUNT.
   78     008100     IF IN97 OF ACCTPMT-O-INDIC EQUAL IND-OFF THEN
   79     008200         PERFORM VERIFY-FROM-ACCOUNT.
   80     008300     PERFORM WRITE-READ-DISPLAY.
          008400
          008500 VERIFY-FROM-ACCOUNT.
   81     008600     MOVE ACCTFROM TO ACCNTKEY.
   82     008700     READ ACCOUNT-FILE
   83     008800         INVALID KEY MOVE IND-ON TO IN99 OF ACCTPMT-O-INDIC
          008900     END-READ
   84     009000     IF IN99 OF ACCTPMT-O-INDIC EQUAL IND-ON THEN  6
          009100*
   85     009200         ROLLBACK
          009300*
          009400     ELSE
   86     009500         PERFORM UPDATE-FROM-ACCOUNT
          009600     END-IF.
          009700
          009800 VERIFY-TO-ACCOUNT.
   87     009900     MOVE ACCTTO TO ACCNTKEY.
   88     010000     READ ACCOUNT-FILE
   89     010100         INVALID KEY MOVE IND-ON TO IN97 OF ACCTPMT-O-INDIC  7
          010200     END-READ
   90     010300     IF IN97 OF ACCTPMT-O-INDIC EQUAL IND-ON THEN
          010400*
   91     010500         ROLLBACK  8
          010600*
          010700     ELSE
   92     010800         PERFORM UPDATE-TO-ACCOUNT
          010900     END-IF.
```

Figura 77 (Parte 3 de 4). Ejemplo de utilización de control de compromiso

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ACCOUNT          AS400SYS   96/07/04 11:17:32         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         011000
         011100 UPDATE-TO-ACCOUNT.
  93     011200     ADD TRANSAMT TO BALANCE.
  94     011300     REWRITE ACCOUNT-RECORD.
         011400
         011500 UPDATE-FROM-ACCOUNT.
  95     011600     SUBTRACT TRANSAMT FROM BALANCE.
  96     011700     REWRITE ACCOUNT-RECORD.
  97     011800     IF BALANCE IS LESS THAN 0 THEN
  98     011900         MOVE IND-ON TO IN98 OF ACCTPMT-O-INDIC
         012000*
  99     012100         ROLLBACK  9
         012200*
         012300     ELSE
         012400*
 100     012500         COMMIT  1
         012600*
         012700     END-IF.
         012800
         012900 WRITE-READ-DISPLAY.
 101     013000     WRITE DISPLAY-REC FORMAT IS "ACCTPMT"
         013100         INDICATORS ARE ACCTPMT-O-INDIC  11
         013200     END-WRITE
 102     013300     MOVE ZEROS TO ACCTPMT-I-INDIC
         013400                   ACCTPMT-O-INDIC.
 103     013500     READ DISPLAY-FILE RECORD
         013600         INDICATORS ARE ACCTPMT-I-INDIC
         013700     END-READ.
         013800

                      * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 77 (Parte 4 de 4). Ejemplo de utilización de control de compromiso

1 Se proporciona una área de indicadores separada para el programa.

2 La cláusula COMMITMENT CONTROL especifica los archivos a colocar bajo el control de compromiso. Los verbos COMMIT y ROLLBACK afectan a los nombres nombrados en esta cláusula.

3 La instrucción COPY de Formato 2 con el atributo de indicador INDIC define las entradas de descripción de datos en WORKING-STORAGE para los indicadores a utilizar en el programa.

4 IN96 se establece si existe un estado de archivo no válido.

5 IN95 se establece si existe una condición INVALID KEY en la operación REWRITE.

6 IN99 se establece si el número de cuenta introducido no es válido para la cuenta desde la cual se transfiere el dinero.

7 IN97 se establece si el número de cuenta introducido no es válido para la cuenta a la que se transfiere el dinero.

8 Si se produce una condición INVALID KEY en la operación READ, se utiliza ROLLBACK y se libera el bloqueo de registro del registro situado detrás del primer READ.

9 Si no se permite la transferencia de datos (se ha establecido un indicador), se procesará la instrucción ROLLBACK. Todos los cambios realizados en los archivos de bases de datos bajo el control de compromiso se cancelarán.

1 Si la transferencia de fondos es válida (no se ha establecido ningún indicador), se procesará la instrucción COMMIT y todos los cambios realizados en los archivos de bases de datos bajo el control de compromiso se convertirán en permanentes.

11 La expresión INDICATORS es necesaria para opciones de la pantalla de la estación de trabajo controladas por indicadores.

Clasificación y fusión de archivos

La organización de registros en una secuencia concreta es un requisito común en un proceso de datos.Dicha secuencia de registros puede conseguirse utilizando las operaciones de clasificación y fusión.

- La operación de **clasificación** acepta entradas no clasificadas y produce salidas en una secuencia especificada.
- La operación de **fusión** compara dos o más archivos clasificados y los combina en orden secuencial.

Para clasificar o fusionar archivos, deberá realizar lo siguiente:

1. Describa los archivos de entrada y salida, si existen, para clasificar o fusionar.
   - Esto se realiza seleccionando los archivos en el párrafo FILE-CONTROL de la INPUT-OUTPUT SECTION y describiendo el archivo, utilizando entradas FD (Descripción de archivos) en la FILE SECTION de DATA DIVISION.
2. Describa los archivos de clasificación y los archivos de fusión.
   - Esto se realiza seleccionando los archivos de clasificiación y fusión en el párrafo FILE-CONTROL de la INPUT-OUTPUT SECTION y describiendo el archivo, utilizando entradas SD (Descripción de clasificación) en la FILE SECTION de DATA DIVISION.
3. Especifique la operación de clasificación y fusión
   - Esto se realiza ejecutando las instrucciones SORT o MERGE en PROCEDURE DIVISION.

Descripción de archivos

Los archivos de clasificación y los de fusión deben describirse con instrucciones SELECT en la ENVIRONMENT DIVISION y con entradas SD (Descripción de clasificación) en la DATA DIVISION. Por ejemplo, vea la Figura 78 en la página 322. El archivo de clasificación o el de fusión descritos en una entrada SD es el archivo de trabajo utilizado durante la operación de clasificación o fusión.No puede ejecutar ninguna instrucción de entrada/salida para este archivo.

Para describir archivos utilizados como entrada o salida de una operación de clasificación o fusión, especifique entradas FD (Descripción de archivos) en la DATA DIVISION. También puede clasificar o fusionar registros definidos solamente en la Working-Storage Section o en la Linkage Section. Si sólo está clasificando o fusionando elementos de datos desde la Working-Storage Section o la Linkage Section y no está utilizando archivos como entrada o salida de una operación de clasificación o fusión, aún necesitará entradas SD y FILE-CONTROL para el archivo de clasificación o de fusión.

Cada entrada SD debe contener una descripción de registro, por ejemplo:

```
SD  SORT-WORK-1.
01  SORT-WORK-1-AREA.
    05  SORT-KEY-1      PIC X(10).
    05  SORT-KEY-2      PIC X(10).
    05  FILLER          PIC X(80).
```

```
5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400      TESTLIB/SMPLSORT        AS400SYS   96/07/04 09:52:26        Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  SMPLSORT.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
          000100*
          000110*    El nombre en la cláusula ASSIGN para un archivo de clasificación
          000120*    se trata como documentación.
    9     000130     SELECT SORT-WORK-1
   10     000140            ASSIGN TO DISK-SORTFILE1.
   11     000150     SELECT SORT-WORK-2
   12     000160            ASSIGN TO DISK-SORTFILE1.
   13     000170     SELECT INPUT-FILE
   14     000180            ASSIGN TO DISK-INFILE.
          000190
   15     000200 DATA DIVISION.
   16     000210 FILE SECTION.
   17     000220 SD  SORT-WORK-1.
   18     000230 01  SORT-WORK-1-AREA.
   19     000240     05  SORT-KEY-1             PIC X(10).
   20     000250     05  SORT-KEY-2             PIC X(10).
   21     000260     05  FILLER                 PIC X(80).
          000270
   22     000280 SD  SORT-WORK-2.
   23     000290 01  SORT-WORK-2-AREA.
   24     000300     05  SORT-KEY               PIC X(5).
   25     000310     05  FILLER                 PIC X(25).
          000320
   26     000330 FD INPUT-FILE.
   27     000340 01  INPUT-RECORD                PIC X(100).
          000350
          000360*    .
          000370*    .
          000380*    .
          000390
   28     000400 WORKING-STORAGE SECTION.
   29     000410 01  EOS-SW                      PIC X.
   30     000420 01  FILLER.
   31     000430     05  TABLE-ENTRY  OCCURS 100 TIMES
          000440                      INDEXED BY X1        PIC X(30).
          000450*    .
          000460*    .
          000470*    .
```

Figura 78. Entradas de ENVIRONMENT DIVISION y DATA DIVISION para un programa de ordenación

Los archivos de clasificación y fusión se procesan con instrucciones SORT o MERGE en la PROCEDURE DIVISION.La instrucción especifica el campo o campos clave dentro del registro sobre el cual debe realizarse la clasificación o la fusión. Puede especificar una clave o claves como ascendente o descendente o, cuando especifique más de una clave, como una mezcla de los dos.

Puede mezclar instrucciones SORT y MERGE en el mismo programa ILE COBOL/400. Un programa ILE COBOL/400 puede contener cualquier número de operaciones de clasificación y fusión, cada uno con su propio procedimiento de entrada o salida independiente.

Puede realizar más de una operación de clasificación o fusión en el programa ILE COBOL/400, incluyendo:

- Múltiples llamadas de la misma operación de clasificación o fusión
- Múltiples operaciones de clasificación o fusión diferentes

Sin embargo, debe finalizarse una operación antes de empezar la siguiente.

Clasificación de archivos

La operación de **clasificación** acepta entradas no clasificadas y produce salidas en una secuencia especificada.

Puede especificar procedimientos de **entrada** para que se ejecuten en los registros de clasificación **antes** de que se clasifiquen utilizando la instrucción SORT...INPUT PROCEDURE.

Puede especificar procedimientos de **salida** para que se ejecuten en los registros de clasificación **después** de que se clasifiquen utilizando la instrucción SORT...OUTPUT PROCEDURE.

Utilice procedimientos de entrada o salida para añadir, eliminar, alterar, editar o cualquier otra modificación de los registros.

Puede utilizar la instrucción SORT para:

- Clasificar elementos de datos (incluyendo tablas) en la Working-Storage Section o en la Linkage Section.
- Leer registros directamente en el archivo nuevo sin ningún proceso preliminar, utilizando la instrucción SORT...USING
- Transferir registros clasificados directamente a un archivo sin más proceso, utilizando la instrucción SORT...GIVING.

Un programa ILE COBOL/400 que contenga una operación de clasificación, se organiza generalmente de manera que un procedimiento de entrada lea y opere en uno o más archivos de entrada.Dentro del procedimiento de entrada, una instrucción RELEASE coloca un registro en el archivo de clasificación. Si no desea modificar ni procesar los registros antes de que empiece la operación de clasificación, la expresión USING de la instrucción SORT transfiere los registros sin modificar desde los archivos de entrada especificados al archivo nuevo.

Una vez finalizada la operación de clasificación, los registros clasificados pueden hacerse disponibles, de uno en uno, mediante una instrucción RETURN para modificarlos en un procedimiento de salida.Si no desea modificar ni procesar los registros clasificados, la opción GIVING de la instrucción SORT nombra al archivo de salida y graba los registros clasificados en el archivo de salida.

Consulte la *ILE COBOL/400 Reference* para obtener más información sobre las instrucciones SORT, RELEASE y RETURN.

Fusión de archivos

La operación de **fusión** compara dos o más archivos clasificados y los combina en orden secuencial.

Tiene acceso a procedimientos de salida, utilizados después de la fusión, que pueden modificar los registros de salida utilizando la instrucción MERGE...OUTPUT PROCEDURE.

A diferencia de la instrucción SORT, no puede especificar un procedimiento de entrada en una instrucción MERGE; debe utilizar la instrucción MERGE...USING.

No es necesario clasificar archivos de entrada antes de una operación de fusión. La operación de fusión los clasifica y combina en un archivo clasificado.

Cuando la instrucción MERGE se encuentra en la Procedure Division, empieza el proceso de fusión.Esta operación de fusión compara claves de los registros de los archivos de entrada y pasa los registros clasificados, de uno en uno, a la instrucción RETURN de un procedimiento de salida o al archivo nombrado en la expresión GIVING.

Si desea procesar los registros fusionados, pueden hacerse disponibles al programa ILE COBOL/400, de uno en uno, a través de la instrucción RETURN en un procedimiento de salida.Si no desea modificar ni procesar los registros fusionados, la expresión GIVING de la instrucción MERGE nombra al archivo fusionado de salida en el que se grabarán los registros fusionados.

Especificación del criterio de clasificación

En la instrucción SORT, especifique la clave sobre la que se clasificarán los registros. La clave debe estar definida en la descripción del registro de los registros que deben clasificarse. En el ejemplo siguiente, observe que SORT-GRID-LOCATION y SORT-SHIFT están definidas en la Data Division antes de que se utilicen en la instrucción SORT.

```
DATA DIVISION.
       ⋮
  SD  SORT-FILE.
  01  SORT-RECORD.
      05  SORT-KEY.
          10  SORT-SHIFT               PIC X(1).
          10  SORT-GRID-LOCATION       PIC X(2).
          10  SORT-REPORT              PIC X(3).
      05  SORT-EXT-RECORD.
          10  SORT-EXT-EMPLOYEE-NUM    PIC X(6).
          10  SORT-EXT-NAME            PIC X(30).
          10  FILLER                   PIC X(73).

PROCEDURE DIVISION.
       ⋮
      SORT SORT-FILE
           ON ASCENDING KEY SORT-GRID-LOCATION SORT-SHIFT
           INPUT PROCEDURE 600-SORT3-INPUT
           OUTPUT PROCEDURE 700-SORT3-OUTPUT.
       ⋮
```

Para clasificar sobre más de una clave, como se muestra en el ejemplo anterior, liste las claves en orden descendiente de importancia.El ejemplo también muestra la utilización de un procedimiento de entrada y de uno de salida. Utilice un procedimiento de entrada si desea procesar los registros antes de clasificarlos, y utilice un procedimiento de salida si desea seguir procesando los registros después de clasificarlos.

Restricciones sobre la longitud de la clave de clasificación

No existe ningún número máximo de claves, siempre y cuando la longitud total de las claves no exceda los 2.000 bytes.

Consideraciones de coma flotante

Los elementos de datos de clave puden ser de coma flotante para la operación SORT (y para la operación MERGE). Si la clave es un elemento de coma flotante externo, éste se trata como datos datos de tipo carácter, lo cual significa que el orden en que se clasificarán los registros dependerá del orden de clasificación que se utilice. Si la clave es un elemento de coma flotante interno, la secuencia aparecerá en orden numérico.

Órdenes de clasificación alternativos

Puede clasificar registros en EBCDIC, ASCII o en otro orden de clasificación. El orden de clasificación por omisión es EBCDIC o el orden de PROGRAM COLLATING SEQUENCE especificado en la Configuration Section. Puede alterar temporalmente el orden de clasificación nombrado en PROGRAM COLLATING SEQUENCE, utilizando la expresión COLLATING SEQUENCE de la instrucción SORT. Como consecuencia, puede utilizar distintos órdenes de clasificación para realizar múltiples clasificaciones en el programa.

También puede especificar el orden de clasificación que un programa utilizará cuando se ejecute en el momento que compile el programa fuente ILE COBOL/400. Puede especificar el orden de clasificación a utilizar, mediante los parámetros SRTSEQ y LANGID de los mandatos CRTCBLMOD y CRTBNDCBL. Consulte el apartado “Especificación de la secuencia de ordenación de idioma en CRTCBLMOD” en la página 46 para obtener una descripción de cómo especificar el orden de clasificación en la compilación. Puede alterar temporalmente el orden de clasificación especificado en la compilación, especificando la cláusula PROGRAM COLLATING SEQUENCE en el párrafo OBJECT-COMPUTER o utilizando la expresión COLLATING SEQUENCE de la instrucción SORT.

Al clasificar un archivo ASCII, tiene que solicitar el orden de clasificación ASCII.Para hacerlo, utilice la expresión COLLATING SEQUENCE *nombre-alfabeto* de la instrucción SORT, donde *nombre-alfabeto* se ha definido en el párrafo SPECIAL-NAMES como STANDARD-1. También puede especificarlo en la cláusula PROGRAM COLLATING SEQUENCE del párrafo OBJECT-COMPUTER si no se especifica la expresión COLLATING SEQUENCE en la instrucción SORT o MERGE que la altera temporalmente.

Grabación del proceso de entrada

Utilice SORT...USING si no tiene que procesar los registros en un archivo o archivos de entrada antes de que se entreguen al programa de clasificación. Con SORT...USING *nombre-de-archivo*, el compilador ILE COBOL/400 genera un procedimiento de entrada para abrir el archivo, leer los registros, transferir los registros al programa de clasificación y cerrar el archivo.

El archivo de entrada no debe estar abierto cuando se ejecute la instrucción SORT. Si desea procesar los registros del archivo de entrada antes de que se transfieran al programa de clasificación, utilice la opción INPUT PROCEDURE de la instrucción SORT.

Cada procedimiento de entrada debe estar representado como un párrafo o una sección. Por ejemplo, para transferir registros de una tabla del Almacenamiento de trabajo a un archivo de clasificación, utilice lo siguiente:

```
PROCEDURE DIVISION.
       ⋮
      SORT SORT-FILE
           ON ASCENDING KEY SORT-KEY
           INPUT PROCEDURE 600-SORT3-INPUT-PROC
       ⋮
600-SORT3-INPUT-PROC SECTION.
      PERFORM WITH TEST AFTER
              VARYING X1 FROM 1 BY 1 UNTIL X1 = 100
           RELEASE SORT-RECORD  FROM  TABLE-ENTRY(X1)
      END-PERFORM.
```

Un procedimiento de entrada contiene código para procesar registros y transferirlos a la operación de clasificación.Quizás desee utilizar un procedimiento de entrada para:

- Transferir elementos de datos del Almacenamiento de trabajo al archivo de clasificación
- Transferir registros que ya se han leído en algún otro lugar del programa
- Leer registros desde un archivo de entrada, seleccionarlos o procesarlos, y transferiros al archivo de clasificación.

Para transferir registros al archivo de clasificación, todos los procedimientos de entrada deben contener como mínimo una instrucción RELEASE o RELEASE FROM.

Grabación del proceso de salida

Utilice SORT...GIVING si desea transferir los registros clasificados directamente desde el archivo de clasificación a otro archivo sin realizar ningún otro proceso. Con SORT...GIVING *nombre-de-archivo*, el compilador ILE COBOL/400 genera un procedimiento de salida para abrir el archivo, devolver los registros, grabar los registros y cerrar el archivo. En el momento de la ejecución de la instrucción SORT, el archivo nombrado con la expresión GIVING no debe estar abierto.

Si desea seleccionar, editar o de lo contrario modificar los registros clasificados antes de grabarlos del archivo de trabajo de clasificación a otro archivo, utilice la expresión OUTPUT PROCEDURE de la instrucción SORT.

En el procedimiento de salida, debe utilizar la instrucción RETURN para que cada registro clasificado esté disponible para el procedimiento de salida. Entonces el procedimiento de salida puede contener cualquier instrucción necesaria para procesar, uno a uno, los registros que la instrucción RETURN hace disponibles.

Puede utilizar RETURN INTO, en lugar de RETURN, para devolver y procesar los registros en el Almacenamiento de trabajo o en una área de salida.También puede utilizar la expresión AT END con la instrucción RETURN. Las instrucciones impe-

rativas de la expresión AT END se ejecutan después de que se hayan devuelto todos los registros del archivo de clasificación.

Cada procedimiento de salida debe incluir como mínimo una instrucción RETURN o RETURN INTO.Además, cada procedimiento de salida debe estar representado como una sección o un párrafo.

Restricciones en los procedimientos de entrada y en los procedimientos de salida

Las restricciones siguientes son aplicables a las instrucciones de los procedimientos de entrada y de salida:

- Los procedimientos de entrada y los procedimientos de salida no deben contener ninguna instrucción SORT o MERGE.
- Los procedimientos de entrada y los procedimientos de salida no deben contener ninguna instrucción STOP RUN, EXIT PROGRAM o GOBACK.
- Se permite una instrucción CALL a otro programa. El programa llamado no puede ejecutar una instrucción SORT o MERGE.
- Puede utilizar instrucciones ALTER, GO TO y PERFORM en los procedimientos de entrada y en los de salida para hacer referencia a nombres de procedimientos fuera del procedimiento de entrada o del de salida; sin embargo, debe volver al procedimiento de entrada o al de salida después de una instrucción GO TO o PERFORM. Los procedimientos COBOL ejecutados como resultado de la instrucción GO TO o PERFORM no deben contener ninguna instrucción SORT o MERGE.
- El resto de la PROCEDURE DIVISION no debe contener ninguna transferencia de control al procedimiento de entrada o al de salida (con la excepción del retorno del control desde una sección declarativa).
- Durante una operación de clasificación o fusión, se utiliza el elemento de datos SD. No debería utilizarlo en el procedimiento de salida antes de que se ejecute una instrucción RETURN.

Determinación de si la clasificación o fusión ha sido satisfactoria

Una vez finalizada una operación de clasificación o fusión, se almacena código de retorno o código de terminación en el registro especial SORT-RETURN. El registro especial SORT-RETURN contiene código de retorno de 0 si la operación de clasificación o fusión ha sido satisfactoria o contiene código de 16 si la operación no ha sido satisfctoria.

El contenido del registro especial SORT-RETURN cambia después de la ejecución de cada instrucción SORT o MERGE.Deberá comprobar si ha finalizado satisfactoriamente después de cada instrucción SORT o MERGE. Por ejemplo:

```
PROCEDURE DIVISION.
       ⋮
      SORT SORT-WORK-2
           ON ASCENDING KEY SORT-KEY
           INPUT PROCEDURE 600-SORT3-INPUT-PROC
           OUTPUT PROCEDURE 700-SORT3-OUTPUT-PROC.
      IF SORT-RETURN NOT EQUAL TO 0
           DISPLAY "FINAL. ANORMAL CLASIFICACIÓN. SORT-RETURN = " SORT-RETURN
       ⋮
600-SORT3-INPUT-PROC SECTION.
       ⋮
700-SORT3-OUTPUT-PROC SECTION.
       ⋮
```

Finalización prematura de una operación de clasificación o fusión

Puede utilizar el registro especial SORT-RETURN para finalizar una operación de clasificación o fusión antes de que se haya completado. Establezca el registro especial SORT-RETURN a 16 en un procedimiento declarativo de error o de entrada/salida, para finalizar la operación de clasificación o fusión antes de que se hayan procesado todos los registros.La operación de clasificación o fusión finalizará antes de devolver o transferir un registro.Entonces el control vuelve a la instrucción que sigue a la instrucción SORT o MERGE.

Clasificación de registros de longitud variable

Los archivos con registros de longitud variable tienen una longitud de registro mínima y una longitud de registro máxima, en lugar de una sola longitud de registro.

Si se están clasificando o fusionando registros de longitud variable, todos los elementos de datos referenciados por nombres de datos clave deben estar en las primeras *n* posiciones de caracteres del registro, donde *n* es igual al tamaño de registro mínimo especificado para el archivo.

Al procesar la instrucción SORT, el compilador ILE COBOL/400 emitirá un mensaje de error si alguna KEY especificada en la instrucción SORT tiene una longitud de registro superior al tamaño de registro mínimo.

Los registros clasificados se truncarán cuando:

- la longitud de registro máxima del archivo de entrada sea superior a la longitud de registro máxima del archivo de clasificación
- la longitud de registro máxima del archivo de clasificación sea superior a la longitud de registro máxima del archivo de salida

Cuando vaya a producirse un truncamiento, se emitirá un mensaje de error de tiempo de compilación; en la ejecución se emitirá un mensaje de diagnóstico.

Los registros de clasificación se rellenarán con espacios en blanco cuando:

- la longitud de registro mínima del archivo de entrada sea inferior a la longitud del registro mínimo del archivo de clasificación
- la longitud de registro mínima del archivo de clasificación sea inferior a la longitud de registro mínima del archivo de salida

Durante la compilación, se emite un mensaje informativo cuando los registros se rellenan con espacios en blanco; no se emite ningún mensaje durante la ejecución.

Ejemplo de clasificación y fusión de archivos

La Figura 79 en la página 330 muestra la creación de archivos clasificados de ventas actuales y de ventas del año a la fecha.

Primero, se ejecuta la instrucción SORT para las ventas actuales. El procedimiento de entrada para esta operación de clasificación es SCREEN-DEPT. Los registros se almacenan en orden ascendente de departamento y, dentro de cada departamento, en orden descendente de ventas netas. Entonces se imprime la salida para esta clasificación.

Una vez finalizada la operación de clasificación, los registros de ventas actuales se fusionan con los registros de ventas del año a la fecha. Los registros de este archivo se fusionan en orden ascendente de número de departamento y, dentro de cada departamento, en orden ascendente de números de empleado, y, para cada empleado, en orden ascendente de meses para crear un archivo maestro actualizado del año a la fecha.

Cuando el proceso de fusión finaliza, se imprime el archivo maestro del año a la fecha actualizado.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SORTMERG         AS400SYS   96/07/04 11:23:54         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  SORTMERGE.
          000030*********************************************************
          000040* ESTE ES UN EJEMPLO DE SORT/MERGE UTILIZANDO UN        *
                  PROCEDIMIENTO DE ENTRADA                             *
          000050*********************************************************
    3     000060 ENVIRONMENT DIVISION.
    4     000070 CONFIGURATION SECTION.
    5     000080 SOURCE-COMPUTER. IBM-AS400.
    6     000090 OBJECT-COMPUTER. IBM-AS400.
    7     000100 INPUT-OUTPUT SECTION.
    8     000110 FILE-CONTROL.
    9     000120     SELECT WORK-FILE
   10     000130            ASSIGN TO DISK-WRK.
   11     000140     SELECT CURRENT-SALES-FILE-IN
   12     000150            ASSIGN TO DISK-CURRIN.
   13     000160     SELECT CURRENT-SALES-FILE-OUT
   14     000170            ASSIGN TO DISK-CURROUT.
   15     000180     SELECT YTD-SALES-FILE-IN
   16     000190            ASSIGN TO DISK-YTDIN.
   17     000200     SELECT YTD-SALES-FILE-OUT
   18     000210            ASSIGN TO DISK-YTDOUT.
   19     000220     SELECT PRINTER-OUT
   20     000230            ASSIGN TO PRINTER-PRTSUMM.
          000240
   21     000250 DATA DIVISION.
   22     000260 FILE SECTION.
   23     000270 SD  WORK-FILE.
   24     000280 01  SALES-RECORD.
   25     000290   05 EMPL-NO                PIC 9(6).
   26     000300   05 DEPT                   PIC 9(2).
   27     000310   05 SALES                  PIC 9(7)V99.
   28     000320   05 NAME-ADDR              PIC X(61).
   29     000330   05 MONTH                  PIC X(2).
   30     000340 FD CURRENT-SALES-FILE-IN.
   31     000350 01 CURRENT-SALES-IN.
   32     000360   05 EMPL-NO                PIC 9(6).
   33     000370   05 DEPT                   PIC 9(2).
   34     000380      88 ON-SITE-EMPLOYEE    VALUES 0 THRU 6, 8.
   35     000390   05 SALES                  PIC 9(7)V99.
   36     000400   05 NAME-ADDR              PIC X(61).
   37     000410   05 MONTH                  PIC X(2).
   38     000420 FD CURRENT-SALES-FILE-OUT.
   39     000430 01 CURRENT-SALES-OUT.
   40     000440   05 EMPL-NO                PIC 9(6).
   41     000450   05 DEPT                   PIC 9(2).
   42     000460   05 SALES                  PIC 9(7)V99.
   43     000470   05 NAME-ADDR              PIC X(61).
   44     000480   05 MONTH                  PIC X(2).
   45     000490 FD YTD-SALES-FILE-IN.
   46     000500 01 YTD-SALES-IN.
   47     000510   05 EMPL-NO                PIC 9(6).
   48     000520   05 DEPT                   PIC 9(2).
   49     000530   05 SALES                  PIC 9(7)V99.
   50     000540   05 NAME-ADDR              PIC X(61).
   51     000550   05 MONTH                  PIC X(2).
```

Figura 79 (Parte 1 de 3). Ejemplo de utilización de SORT/MERGE

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SORTMERG         AS400SYS   96/07/04 11:23:54         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   52     000560 FD YTD-SALES-FILE-OUT.
   53     000570 01 YTD-SALES-OUT.
   54     000580   05 EMPL-NO                PIC 9(6).
   55     000590   05 DEPT                   PIC 9(2).
   56     000600   05 SALES                  PIC 9(7)V99.
   57     000610   05 NAME-ADDR              PIC X(61).
   58     000620   05 MONTH                  PIC X(2).
   59     000630 FD PRINTER-OUT.
   60     000640 01 PRINT-LINE.
   61     000650   05 RECORD-LABEL           PIC X(25).
   62     000660   05 DISK-RECORD-DISPLAY    PIC X(80).
          000670
   63     000680 WORKING-STORAGE SECTION.
   64     000690 01 SALES-FILE-IN-EOF-STATUS  PIC X        VALUE "F".
   65     000700    88 SALES-FILE-IN-END-OF-FILE          VALUE "T".
   66     000710 01 SALES-FILE-OUT-EOF-STATUS PIC X        VALUE "F".
   67     000720    88 SALES-FILE-OUT-END-OF-FILE         VALUE "T".
   68     000730 01 YTD-SALES-OUT-EOF-STATUS  PIC X        VALUE "F".
   69     000740    88 YTD-SALES-OUT-END-OF-FILE          VALUE "T".
          000750
   70     000760 PROCEDURE DIVISION.
          000770 MAIN-PROGRAM SECTION.
          000780 MAINLINE.
          000790
   71     000800     OPEN INPUT CURRENT-SALES-FILE-IN
          000810                CURRENT-SALES-FILE-OUT
          000820                YTD-SALES-FILE-OUT
          000830           OUTPUT PRINTER-OUT.
          000840*
          000850*    Clasificación ventas actuales
          000860*
   72     000870     SORT WORK-FILE
          000880          ON ASCENDING KEY DEPT OF SALES-RECORD
          000890          ON DESCENDING KEY SALES OF SALES-RECORD
          000900          INPUT PROCEDURE SCREEN-DEPT
          000910          GIVING CURRENT-SALES-FILE-OUT.
   73     000920     READ CURRENT-SALES-FILE-OUT
   74     000930         AT END SET SALES-FILE-OUT-END-OF-FILE TO TRUE
          000940     END-READ.
   75     000950     PERFORM UNTIL SALES-FILE-OUT-END-OF-FILE
   76     000960         MOVE "SORTED CURRENT SALES "
          000970             TO RECORD-LABEL OF PRINT-LINE
   77     000980         MOVE CURRENT-SALES-OUT TO DISK-RECORD-DISPLAY
   78     000990         WRITE PRINT-LINE
   79     001000         READ CURRENT-SALES-FILE-OUT
   80     001010             AT END SET SALES-FILE-OUT-END-OF-FILE TO TRUE
          001020         END-READ
          001030     END-PERFORM.
          001040*
          001050*    Actualización informe anual
          001060*
   81     001070     MERGE WORK-FILE
          001080          ON ASCENDING KEY DEPT OF SALES-RECORD
          001090          ON ASCENDING KEY EMPL-NO OF SALES-RECORD
          001100          ON ASCENDING KEY MONTH OF SALES-RECORD
          001110          USING YTD-SALES-FILE-IN
          001120                CURRENT-SALES-FILE-IN
```

Figura 79 (Parte 2 de 3). Ejemplo de utilización de SORT/MERGE

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SORTMERG         AS400SYS   96/07/04 11:23:54         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         001130          GIVING YTD-SALES-FILE-OUT.
         001140*
         001150*    Impresión informe anual
         001160*
   82    001170     READ YTD-SALES-FILE-OUT
   83    001180         AT END SET YTD-SALES-OUT-END-OF-FILE TO TRUE
         001190     END-READ.
   84    001200     PERFORM UNTIL YTD-SALES-OUT-END-OF-FILE
   85    001210         MOVE "MERGED YTD SALES ",
         001220             TO RECORD-LABEL OF PRINT-LINE
   86    001230         MOVE YTD-SALES-OUT TO DISK-RECORD-DISPLAY
   87    001240         WRITE PRINT-LINE
   88    001250         READ YTD-SALES-FILE-OUT
   89    001260             AT END SET YTD-SALES-OUT-END-OF-FILE TO TRUE
         001270         END-READ
         001280     END-PERFORM.
         001290
   90    001300     CLOSE CURRENT-SALES-FILE-IN
         001310           CURRENT-SALES-FILE-OUT
         001320           YTD-SALES-FILE-OUT
         001330           PRINTER-OUT.
   91    001340     STOP RUN.
         001350
         001360 SCREEN-DEPT SECTION.
         001370 SCREEN-DEPT-PROCEDURE.
         001380
   92    001390     READ CURRENT-SALES-FILE-IN
   93    001400         AT END SET SALES-FILE-IN-END-OF-FILE TO TRUE
         001410     END-READ.
   94    001420     PERFORM UNTIL SALES-FILE-IN-END-OF-FILE
   95    001430         MOVE "UNSORTED CURRENT SALES ",
         001440             TO RECORD-LABEL OF PRINT-LINE
   96    001450         MOVE CURRENT-SALES-IN TO DISK-RECORD-DISPLAY
   97    001460         WRITE PRINT-LINE
   98    001470         IF ON-SITE-EMPLOYEE
   99    001480             MOVE CURRENT-SALES-IN TO SALES-RECORD
  100    001490             RELEASE SALES-RECORD
         001500         END-IF
  101    001510         READ CURRENT-SALES-FILE-IN
  102    001520             AT END SET SALES-FILE-IN-END-OF-FILE TO TRUE
         001530         END-READ
         001540     END-PERFORM.

                     * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 79 (Parte 3 de 3). Ejemplo de utilización de SORT/MERGE

Declaración de elementos de datos utilizando tipos de datos SAA

El compilador ILE COBOL/400 le permite convertir campos de longitud variable, desde archivos descritos externamente y tipos de datos de base de datos SAA, en elementos de datos COBOL estándares. Los tipos de datos SAA que puede convertir son campos de longitud variable, campos de fecha, hora o indicación de la hora y campos gráficos DBCS. ILE COBOL/400 proporciona soporte limitado para estos tipos de datos.

Campos de longitud variable

Puede trasladar un campo de longitud variable al programa si especifica *VARCHAR en el parámetro CVTOPT de los mandatos CRTCBLMOD o CRTBNDCBL, o la opción VARCHAR de la instrucción PROCESS. Cuando se especifica *VARCHAR, el programa ILE COBOL/400 convierte un campo de lon-

gitud variable de un archivo descrito externamente en un elemento de grupo ILE COBOL/400.

Un ejemplo de dicho elemento de grupo es:

```
06  ITEM1.
    49  ITEM1-LENGTH     PIC S9(4) COMP-4.
    49  ITEM1-DATA       PIC X(n).
```

donde `n` representa la longitud máxima del campo de longitud variable. Dentro del programa, `PIC S9(4) COMP-4` se trata como cualquier otra declaración de este tipo y `PIC X(n)` se trata como alfanumérica estándar.

Cuando no se especifica *VARCHAR, los campos de longitud variable se ignoran y se declaran como campos FILLER en programas ILE COBOL/400. Si se especifica *NOVARCHAR, el elemento se declara como sigue:

```
06  FILLER     PIC x(n+2).
```

Para obtener información sobre sintaxis, vea el parámetro CVTOPT de la página 38.

El programa puede realizar cualquier operación de caracteres válida en la parte de datos generados; sin embargo, debido a la estructura del campo, la parte de la longitud debe tener un dato binario válido. Estos datos no son válidos si son negativos o superiores a la longitud de campo máxima.

Si los dos primeros bytes del campo no contienen un número binario válido, se producirá un error si intenta grabar (WRITE) o volver a grabar (REWRITE) un registro que contenga el campo y dará como resultado el estado del archivo 90.

Las condiciones siguientes son aplicables cuando especifique campos de longitud variable:

- Si se encuentra un campo de longitud variable al utilizar una instrucción COPY de Formato 2 en la DATA DIVISION, se declara en un programa ILE COBOL/400 como un campo de caracteres de longitud fija.
- Para campos de caracteres de un sólo byte, la longitud del campo ILE COBOL/400 declarado es el número de caracteres de un sólo byte en el campo DDS más 2 bytes.
- Para campos de datos gráficos DBCS, la longitud del campo ILE COBOL/400 declarado es dos veces el número de caracteres gráficos DBCS en el campo DDS más 2 bytes. Para obtener más información sobre los tipos de datos gráficos, vea “Campos gráficos DBCS” en la página 335. Los dos bytes extra del campo ILE COBOL/400 contienen un número binario que representa la longitud actual del campo de longitud variable. La Figura 80 en la página 334 muestra la longitud del campo ILE COBOL/400 de campos de longitud variable.

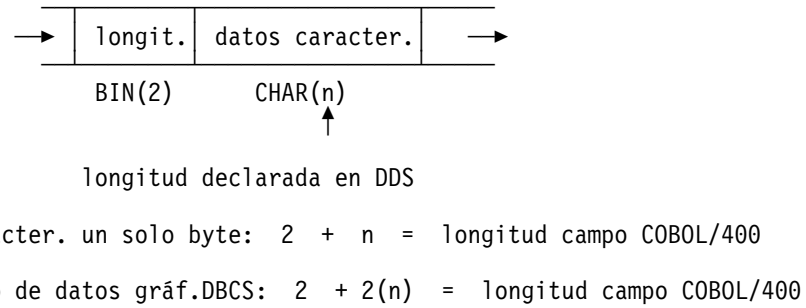


Figura 80. Longitud de campo COBOL/400 de un campo de longitud variable

- El programa ILE COBOL/400 puede realizar cualquier operación válida de manipulación de caracteres en el campo de longitud fija declarado. Sin embargo, debido a la estructura del campo, los dos primeros bytes del campo deben contener datos binarios válidos (los datos de longitud de campo actuales no válidos son inferiores a 0 o superiores a la longitud de campo DDS). Se producirá un error para una operación de entrada o salida si los dos primeros bytes del campo contienen datos de longitud de campo no válidos; dará el estado de archivo 90.
- Si no especifica *VARCHAR, puede encontrarse con problemas al realizar operaciones WRITE en campos de longitud variable debido a que el usuario no puede asignar un valor a FILLER. El campo de dos bytes puede que tenga un valor (por ejemplo `X'4040'`) que dé como resultado una longitud superior al rango permitido por el campo. Esto originará un error de E/S.
- Los campos de longitud variable no pueden utilizarse en una clave SORT/MERGE como un campo de longitud variable.Si el campo de longitud variable se utiliza en una clave SORT/MERGE, entonces la estructura entera se comparará como un elemento de datos alfanumérico.

Para ver un ejemplo de un programa utilizando campos de longitud variable, consulte “Ejemplos de utilización de campos gráficos DBCS de longitud variable” en la página 337.

Campos de fecha, hora e indicación de la hora

Los campos de fecha, hora e indicación de la hora se trasladan al programa sólo si se especifica la opción *DATETIME del parámetro CVTOPT de CRTCBLMOD o CRTBNDCBL o la opción DATETIME de la instrucción PROCESS. Para obtener una descripción y la sintaxis del parámetro CVTOPT, vea la página 38. Si *DATETIME no se especifica, los campos de fecha, hora e indicación de la hora se ignorarán y se declararán campos FILLER en el programa ILE COBOL/400.

Los campos de fecha, hora e indicación de la hora se trasladan a un programa ILE COBOL/400 como campos de caracteres de longitud fija. El programa ILE COBOL/400 puede realizar cualquier operación de caracteres válida en los campos de longitud fija. Estas operaciones seguirán las normas COBOL estándares para elementos de datos alfanuméricos.

Los tipos de datos de fecha, hora e indicación de la hora tienen cada uno su propio formato.

Si el programa actualiza un campo que contiene información sobre la fecha, hora o indicación de la hora y la información actualizada se ha de volver a pasar a la base de datos, el formato del campo debe ser exactamente el mismo que cuando se recuperó el campo de la base de datos. Si no utiliza el mismo formato, se producirá un error. Para obtener información sobre los formatos válidos para cada tipo de datos, vea la *DDS Reference*.

Si intenta GRABAR (WRITE) un registro antes de trasladar un valor apropiado a un campo de fecha, hora o indicación de la hora, la operación WRITE fallará y devolverá el estado de archivo 90.

Si declara los elementos de fecha, hora o indicación de la hora en el programa como FILLER, no intente grabar (WRITE) registros que contengan estos campos, ya que no podrá establecerlos en valores que el sistema acepte.

Los campos de fecha, hora e indicación de la hora no pueden utilizarse como parte de una clave SORT/MERGE como campos de fecha, hora e indicación de la hora.En ILE COBOL/400, los campos de fecha, hora e indicación de la hora se convierten a elementos de datos alfanuméricos.Se comparan como elementos de datos alfanuméricos cuando se utilizan en una clave SORT/MERGE.

Campos con posibilidad de nulos

Aunque el programa puede procesar campos con posibilidad de nulos, los valores nulos no se soportan. Pueden realizarse las operaciones READ, SORT y MERGE en campos con posibilidad de nulo, pero si los campos contienen realmente valores nulos, se producirá un error.

Campos gráficos DBCS

El tipo de datos gráficos DBCS es una serie de caracteres en la que cada carácter se representa mediante 2 bytes. El tipo de datos gráficos DBCS no contiene caracteres de desplazamiento a teclado ideográfico (SO) ni caracteres de desplazamiento a teclado estándar (SI). La diferencia entre datos de un sólo byte y datos gráficos DBCS se muestra en la figura siguiente:

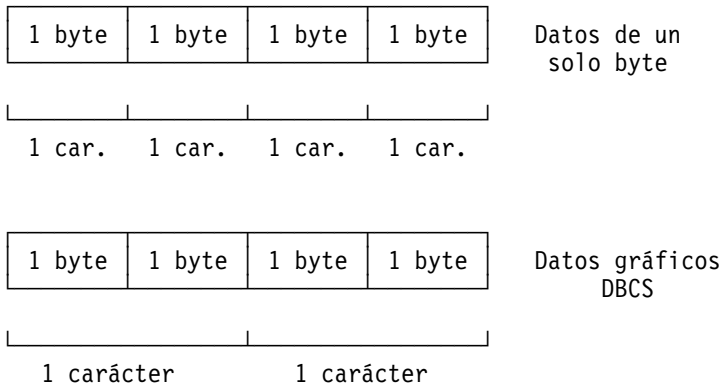


Figura 81. Comparación de datos de un sólo byte y datos gráficos

Los datos gráficos DBCS se trasladan al programa ILE COBOL/400 sólo si especifica el valor *PICXGRAPHIC o *PICGGRAPHIC en el parámetro CVTOPT de los mandatos CRTCBLMOD o CRTBNDCBL, o la opción CVTPICXGRAPHIC o CVTPICGGRAPHIC de la instrucción PROCESS. Si no lo hace, los datos gráficos se ignorarán y se declararán como campos FILLER en el programa ILE COBOL/400. Para obtener una descripción y la sintaxis del parámetro CVTOPT, vea “Parámetros del mandato CRTCBLMOD” en la página 33.

Las condiciones siguientes son aplicables cuando se especifican datos gráficos DBCS:

- Los datos gráficos DBCS se copian en un programa ILE COBOL/400 como un campo alfanumérico o DBCS de longitud fija.
- Cada *carácter* de datos gráfico DBCS tiene una longitud de 2 bytes.
- Cuando se especifica *PICXGRAPHIC, cada *campo* de datos gráficos DBCS de longitud fija (por ejemplo, especificado como 2G) tiene una longitud del número de bytes en el campo (una longitud de 4 bytes en el caso de 2G, por ejemplo). Cuando se especifica *PICGGRAPHIC, cada campo de datos gráficos DBCS de longitud fija tiene una longitud del número de caracteres de doble byte (2 caracteres como se muestra en la Figura 81 en la página 335). Para obtener una descripción de la longitud de campo de campos de datos gráficos de longitud variable, vea “Campos de longitud variable” en la página 332.
- El programa ILE COBOL/400 puede realizar cualquier operación de caracteres válida en los campos de longitud fija.

Campos gráficos DBCS de longitud variable

Puede utilizar campos de longitud variable con tipos de datos gráficos DBCS para especificar datos gráficos DBCS de longitud variable. Para especificar datos gráficos DBCS de longitud variable, especifique *VARCHAR y *PICXGRAPHIC para el parámetro CVTOPT de los mandatos CRTCBLMOD o CRTBNDCBL o las opciones VARCHAR y CVTPICXGRAPHIC para la instrucción PROCESS.

Si especifica una de las opciones siguientes: CVTOPT(*NOVARCHAR *NOPICGGRAPHIC), CVTOPT(*NOVARCHAR *PICGGRAPHIC), CVTOPT(*NOVARCHAR *NOPICXGRAPHIC) o CVTOPT(*NOVARCHAR *PICXGRAPHIC) y el compilador ILE COBOL/400 se encuentra con un elemento de datos gráficos DBCS de longitud variable, el programa resultante contiene lo siguiente:

```
             06 FILLER            PIC X(2n+2).
      *           (Campo de longitud variable)
```

donde `n` es el número de caracteres en el campo DDS.

Si especifica CVTOPT(*VARCHAR *NOPICGGRAPHIC) o CVTOPT(*VARCHAR *NOPICXGRAPHIC), y el compilador ILE COBOL/400 encuentra un elemento de datos gráficos DBCS de longitud variable, el programa resultante contiene lo siguiente:

```
           06 NAME
      *         (Campo de longitud variable)
              49 NAME-LENGTH        PIC S9(4) COMP-4.
      *                     (Número de caracteres de 2 bytes)
              49 FILLER             PIC X(2n).
```

donde n es el número de caracteres DBCS en el campo DDS.

Si especifica CVTOPT(*VARCHAR *PICXGRAPHIC) y el compilador ILE COBOL/400 encuentra un elemento de datos gráficos DBCS de longitud variable, el programa resultante contiene lo siguiente:

```
           06 NAME
      *         (Campo de longitud variable)
              49 NAME-LENGTH        PIC S9(4) COMP-4.
      *                     (Número de caracteres de 2 bytes)
              49 NAME-DATA          PIC X(2n).
```

donde n es el número de caracteres DBCS en el campo DDS.

Si especifica CVTOPT(*VARCHAR *PICGGRAPHIC) y el compilador ILE COBOL/400 encuentra un elemento de datos gráficos DBCS de longitud variable, el programa resultante contiene lo siguiente:

```
           06 NAME
      *         (Campo de longitud variable)
              49 NAME-LENGTH        PIC S9(4) COMP-4.
      *                     (Número de caracteres de 2 bytes)
              49 NAME-DATA          PIC G(n) DISPLAY-1.
```

donde n es el número de caracteres DBCS en el campo DDS.

Ejemplos de utilización de campos gráficos DBCS de longitud variable

La Figura 82 muestra un ejemplo de un archivo DDS que define un elemento de datos gráficos DBCS de longitud variable. La Figura 83 en la página 338 muestra el programa ILE COBOL/400 utilizando una instrucción COPY de Formato 2 con *PICXGRAPHIC y el listado resultante al compilar el programa. La Figura 84 en la página 339 muestra el programa ILE COBOL/400 utilizando elementos de datos gráficos DBCS de longitud variable con *PICGGRAPHIC.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R SAMPLEFILE
     A*
     A            VARITEM      100          VARLEN
     A*
     A            TIMEITEM       T          TIMFMT(*HMS)
     A            DATEITEM       L          DATFMT(*YMD)
     A            TIMESTAMP      Z
     A*
     A            GRAPHITEM    100G
     A            VGRAPHITEM   100G          VARLEN
```

Figura 82. Archivo DDS definiendo un campo de datos gráficos de longitud variable

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PGM1            AS400SYS   96/07/04 10:56:39         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


         000010 process varchar datetime cvtpicxgraphic
    1    000020 Identification division.
    2    000030   Program-id.  pgm1.
         000040
    3    000050 Environment division.
    4    000060 Configuration section.
    5    000070   Source-computer.  ibm-as400.
    6    000080   Object-computer.  ibm-as400.
    7    000090 Input-output section.
    8    000100   File-control.
    9    000110     Select file1
   10    000120       assign to database-samplefile
   11    000130       organization is sequential
   12    000140       access is sequential
   13    000150       file status is fs1.
         000160
   14    000170 Data division.
   15    000180 File section.
   16    000190 fd  file1.
   17    000200 01  record1.
         000210 copy dds-all-formats of samplefile.
   18   +000001       05  SAMPLEFILE-RECORD PIC X(546).                                  <-ALL-FMTS
        +000002*    FORMATO E-S:SAMPLEFILE DEL ARCH. SAMPLEFILE DE LA BIBL. TESTLIB    <-ALL-FMTS
        +000003*                                                                       <-ALL-FMTS
   19   +000004       05  SAMPLEFILE    REDEFINES SAMPLEFILE-RECORD.                  <-ALL-FMTS
   20   +000005           06 VARITEM.                                                 <-ALL-FMTS
        +000006*              (Campo de longitud variable)                             <-ALL-FMTS
   21   +000007               49 VARITEM-LENGTH   PIC S9(4) COMP-4.                   <-ALL-FMTS
   22   +000008               49 VARITEM-DATA     PIC X(100).                         <-ALL-FMTS
   23   +000009           06 TIMEITEM             PIC X(8).                           <-ALL-FMTS
        +000010*                 (Campo de hora)                                       <-ALL-FMTS
   24   +000011           06 DATEITEM             PIC X(8).                           <-ALL-FMTS
        +000012*                 (Campo de fecha)                                      <-ALL-FMTS
   25   +000013           06 TIMESTAMP            PIC X(26).                          <-ALL-FMTS
        +000014*                 (Campo de indicación de la hora)                      <-ALL-FMTS
   26   +000015           06 GRAPHITEM            PIC X(200).                         <-ALL-FMTS
   27   +000017           06 VGRAPHITEM.                                              <-ALL-FMTS
        +000018*              (Campo de longitud variable)                             <-ALL-FMTS
   28   +000019               49 VGRAPHITEM-LENGTH                                    <-ALL-FMTS
        +000020                                  PIC S9(4) COMP-4.                    <-ALL-FMTS
        +000021*                 (Número de caracteres de 2 bytes)                     <-ALL-FMTS
   29   +000022               49 VGRAPHITEM-DATA  PIC X(200).                         <-ALL-FMTS
   30    000220 Working-Storage section.
   31    000230 77  fs1      pic x(2).
         000240
   32    000250 Procedure division.
         000260 Mainline.
   33    000270     stop run.

                      * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 83. Programa ILE COBOL/400 utilizando elementos de datos gráficos DBCS de longitud variable

```
5716CB1 V3R7M0  961108 LN       IBM ILE COBOL/400       TESTLIB/PGM1            AS400SYS   96/07/04 10:56:39        Página     2

                                 F u e n t e

 INST PL NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


        000010 process varchar datetime cvtpicggraphic
   1    000020 Identification division.
   2    000030   Program-id.  pgm1.
        000040
   3    000050 Environment division.
   4    000060 Configuration section.
   5    000070   Source-computer.  ibm-as400.
   6    000080   Object-computer.  ibm-as400.
   7    000090 Input-output section.
   8    000100   File-control.
   9    000110     Select file1
  10    000120       assign to database-samplefile
  11    000130       organization is sequential
  12    000140       access is sequential
  13    000150       file status is fs1.
        000160
  14    000170 Data division.
  15    000180 File section.
  16    000190 fd  file1.
  17    000200 01  record1.
        000210 copy dds-all-formats of samplefile.
  18   +000001       05  SAMPLEFILE-RECORD PIC X(546).                                  <-ALL-FMTS
       +000002*    FORMATO E-S:SAMPLEFILE DEL ARCH. SAMPLEFILE DE LA BIBL. TESTLIB      <-ALL-FMTS
       +000003*                                                                        <-ALL-FMTS
  19   +000004       05  SAMPLEFILE    REDEFINES SAMPLEFILE-RECORD.                    <-ALL-FMTS
  20   +000005           06 VARITEM.                                                   <-ALL-FMTS
       +000006*              (Campo de longitud variable)                              <-ALL-FMTS
  21   +000007               49 VARITEM-LENGTH   PIC S9(4) COMP-4.                     <-ALL-FMTS
  22   +000008               49 VARITEM-DATA     PIC X(100).                           <-ALL-FMTS
  23   +000009           06 TIMEITEM             PIC X(8).                             <-ALL-FMTS
       +000010*                 (Campo de hora)                                        <-ALL-FMTS
  24   +000011           06 DATEITEM             PIC X(8).                             <-ALL-FMTS
       +000012*                 (Campo de fecha)                                       <-ALL-FMTS
  25   +000013           06 TIMESTAMP            PIC X(26).                            <-ALL-FMTS
       +000014*                 (Campo de indicación de la hora)                       <-ALL-FMTS
  26   +000015           06 GRAPHITEM            PIC G(100) DISPLAY-1.                 <-ALL-FMTS
       +000016*                 (Campo gráfico)                                        <-ALL-FMTS
  27   +000017           06 VGRAPHITEM.                                                <-ALL-FMTS
       +000018*              (Campo de longitud variable)                              <-ALL-FMTS
  28   +000019               49 VGRAPHITEM-LENGTH                                      <-ALL-FMTS
       +000020                                  PIC S9(4) COMP-4.                     <-ALL-FMTS
       +000021*                 (Número de caracteres de 2 bytes)                      <-ALL-FMTS
  29   +000022               49 VGRAPHITEM-DATA  PIC G(100) DISPLAY-1.                 <-ALL-FMTS
       +000023*                 (Campo gráfico)                                        <-ALL-FMTS
  30    000220 Working-Storage section.
  31    000230 77  fs1      pic x(2).
        000240
  32    000250 Procedure division.
        000260 Mainline.
  33    000270     stop run.

                      * * * * *  E N D  O F  S O U R C E  * * * * *
```

*Figura 84. Programa ILE COBOL/400 utilizando elementos de datos DBCS de longitud variable y *CVTPICGGRAPHIC*

Campos de coma flotante

Puede trasladar un campo de coma flotante interno al programa si especifica *FLOAT en el parámetro CVTOPT de los mandatos CRTCBLMOD o CRTBNDCBL, o la opción FLOAT de la instrucción PROCESS.

Cuando se especifica *FLOAT, los tipos de datos de coma flotante se transfieren al programa con los nombres de DDS y un USAGE de COMP-1 (precisión simple) o

COMP-2 (precisión doble). Si no se especifica *FLOAT, los tipos de datos de coma flotante se declaran como campos FILLER con un USAGE binario.

Por ejemplo, si especifica *FLOAT para un campo de coma flotante de precisión simple con la siguiente DDS:

```
COMP1          9F         FLTPCN(*SINGLE)
```

el elemento de datos que se transfiere al programa es el siguiente:

```
06  COMP1               COMP-1.
```

Si no especifica *FLOAT (o especifica *NOFLOAT) para la DDS anterior, el campo DDS se generará de la siguiente forma:

```
06  FILLER      PIC 9(5)        COMP-4.
```

En general, los elementos de datos de coma flotante pueden utilizarse en cualquier lugar en que se utilicen elementos de datos decimales numéricos.

Capítulo 15. Acceso a dispositivos conectados externamente

Este capítulo describe cómo ILE COBOL/400 interactúa con dispositivos conectados externamente.Estos dispositivos son hardware conectado externamente, como por ejemplo impresoras, unidades de cinta, unidades de disquetes, estaciones de pantalla y otros sistemas.

Puede acceder a dispositivos conectados externamente desde ILE COBOL/400, utilizando archivos de dispositivos. Los **Archivos de dispositivo** son archivos que proporcionan acceso a dispositivos conectados externamente, como por ejemplo pantallas, impresoras, cintas, disquetes y otros sistemas conectados mediante una línea de comunicaciones.

Tipos de archivos de dispositivos

Antes de que el programa ILE COBOL/400 pueda leer o grabar en los dispositivos del sistema, debe crearse, al configurarse un dispositivo, una descripción del mismo que identifique las posibilidades de hardware del dispositivo al sistema operativo.Un archivo de dispositivos especifica cómo puede utilizarse un dispositivo. Haciendo referencia a un archivo de dispositivos específico, el programa ILE COBOL/400 utiliza el dispositivo de la forma descrita al sistema. El archivo de dispositivos da formato a los datos de salida del programa ILE COBOL/400 para presentarlos al dispositivo y da formato a los datos de entrada para presentarlos al programa ILE COBOL/400.

Utilice los archivos de dispositivos listados en Tabla 16 en la página 342 para acceder a los dispositivos asociados conectados externamente:

Tabla 16. Archivos de dispositivos y sus dispositivos asociados conectados externamente

Archivo de dispositivos	Dispositivos asociados conectados externamente	Mandatos CL	Nombre de dispositivo ILE COBOL/400	Nombre de archivo por omisión ILE COBOL/400
Archivos de impresora	Proporciona acceso a dispositivos de impresora y describe el formato de la salida impresa.	CRTPRTF CHGPRTF OVRPRTF	PRINTER FORMATFILE	QPRINT
Archivos de cinta	Proporciona acceso a archivos de cinta almacenados en dispositivos de cinta.	CRTTAPF CHGTAPF OVRTAPF	TAPEFILE	QTAPE
Archivos de disquete	Proporciona acceso a archivos de datos almacenados en dispositivos de disquete.	CRTDKTF CHGDKTF OVRDKTF	DISKETTE	QDKT
Archivos de pantalla	Proporciona acceso a dispositivos de pantalla.	CRTDSPF CHGDSPF OVRDSPF	WORKSTATION	
Archivos ICF	Permite a un programa de un sistema comunicarse con un programa de otro sistema.	CRTICFF CHGICFF OVRICFF	WORKSTATION	

El archivo de dispositivo contiene la descripción de archivo que identifica el dispositivo a utilizar; no contiene datos.

Acceso a dispositivos de impresora

Puede crear una salida impresa en un dispositivo de impresora desde un programa ILE COBOL/400, emitiendo una instrucción WRITE a uno o más archivos de impresora. Puede utilizar uno de los archivos de impresora suministrados por IBM, como por ejemplo QPRINT, o puede crear sus propios archivos de impresora utilizando el mandato Crear archivo de impresión (CRTPRTF). Vea la publicación *CL Reference* para obtener más información sobre el mandato CRTPRTF.

Para utilizar un archivo de impresora en un programa ILE COBOL/400, debe:

- denominar el archivo de impresora mediante una entrada de control de archivos en el párrafo FILE-CONTROL de la Environment Division
- describir el archivo de impresora mediante una entrada de descripción de archivos en la DATA DIVISION

Las operaciones válidas de archivos para un archivo de impresora son WRITE, OPEN y CLOSE.

Denominación de archivos de impresora

Para utilizar un archivo de impresora en un programa ILE COBOL/400, debe denominar el archivo de impresora mediante una entrada de control de archivos en el párrafo FILE-CONTROL de la ENVIRONMENT DIVISION. Vea la publicación *ILE COBOL/400 Reference* para obtener una descripción completa del párrafo FILE-CONTROL. Puede utilizar más de un archivo de impresora en un programa ILE COBOL/400, pero cada archivo de impresora debe tener un nombre exclusivo.

Los archivos de impresora pueden ser archivos descritos por programa o archivos descritos externamente.

Denomine un archivo de impresora descrito por programa en el párrafo FILE-CONTROL tal como sigue:

```
FILE-CONTROL.
    SELECT nombre-archivo-impresora
        ASSIGN TO PRINTER-nombre_dispositivo_impresora
        ORGANIZATION IS SEQUENTIAL.
```

Denomine un archivo de impresora descrito externamente en el párrafo FILE-CONTROL tal como sigue:

```
FILE-CONTROL.
    SELECT nombre-archivo-impresora
        ASSIGN TO FORMATFILE-nombre_dispositivo_impresora
        ORGANIZATION IS SEQUENTIAL.
```

Utilice la cláusula SELECT para elegir un archivo. Este archivo debe identificarse mediante una entrada FD en la DATA DIVISION.

Utilice la cláusula ASSIGN para asociar el archivo de impresora con un dispositivo de impresora. Debe especificar un tipo de dispositivo PRINTER en la cláusula ASSIGN para utilizar un archivo de impresora descrito por programa.Para utilizar un archivo de impresora descrito externamente, debe especificar un tipo de dispositivo de FORMATFILE en la cláusula ASSIGN.

Utilice ORGANIZATION IS SEQUENTIAL en la entrada de control de archivos al denominar un archivo de impresora.

Descripción de archivos de impresora

Una vez denominado el archivo de impresora en la Environment Division debe describir el archivo de impresora mediante una entrada de descripción de archivos en la DATA DIVISION. Vea la publicación *ILE COBOL/400 Reference* para obtener una descripción completa de la Entrada de descripción de archivos.Utilice la Entrada de descripción de archivos de Formato 4 para describir un archivo de impresora.

Los archivos de impresora pueden ser descritos por programa o descritos externamente. Los archivos de impresora descritos por programa se asignan a un dispositivo de PRINTER. Los archivos de impresora descritos externamente se asignan a un dispositivo de FORMATFILE. Utilizar FORMATFILE le permite aprovechar la función AS/400 al máximo, y utilizar PRINTER permite una mayor portabilidad del programa.

La utilización de archivos de impresora descritos externamente tiene las siguientes ventajas sobre los archivos de impresora descritos por programa:

- Mediante una instrucción WRITE pueden imprimirse múltiples líneas. Cuando se imprimen múltiples líneas mediante una instrucción WRITE y se llega a la condición END-OF-PAGE, la instrucción imperativa END-OF-PAGE se procesa después de que se hayan impreso todas las líneas. Es posible imprimir líneas en el área de desbordamiento y en la página siguiente, antes de que se procese la instrucción imperativa END-OF-PAGE.

  La Figura 87 en la página 348 muestra un ejemplo de una aparición de la condición END-OF-PAGE mediante FORMATFILE.

- Es posible la impresión opcional de campos basados en valores de indicador.
- La edición de valores de campo se define fácilmente.
- El mantenimiento de formatos de impresión, especialmente los utilizados por programas múltiples, es más fácil.

La utilización de la expresión ADVANCING para archivos FORMATFILE causa la emisión de un error de compilación. El avance de las líneas se controla en un archivo FORMATFILE mediante palabras clave DDS, como por ejemplo SKIPA y SKIPB, y mediante la utilización de números de línea

Descripción de archivos de impresora descritos por programa

Los archivos de impresora descritos por programa deben asignarse a un dispositivo de PRINTER.Una entrada de descripción de archivos simple en la Data Division, que describe un archivo de impresora descrito por programa, tiene la siguiente apariencia:

```
FD  archivo-impresión.
01  registro-impresión         PIC X(132).
```

Utilización de la cláusula LINAGE para manejar los controles de espaciado y paginación: Puede solicitar que todos los controles de espaciado y paginación los maneje internamente el código generado por compilador, especificando la cláusula LINAGE en la entrada de descripción de archivos de un archivo de impresora descrito por programa.

```
FD  archivo-impresión
    LINAGE IS entero-1 LINES
           WITH FOOTING AT entero-2
           LINES AT TOP entero-3
           LINES AT BOTTOM entero-4.
01  registro-impresión         PIC X(132).
```

La colocación del papel sólo se realiza cuando se ejecuta la instrucción WRITE. El papel de la impresora se coloca en una nueva página física y el LINAGE-COUNTER se establece en 1. Cuando el archivo de impresora se comparte y otros programas han grabado registros en el archivo, la instrucción WRITE de ILE COBOL/400 aún se considera la primera instrucción WRITE. La colocación del papel la maneja el compilador ILE COBOL/400, aunque no sea la primera instrucción WRITE de ese archivo.

Todo el espaciado y la paginación para las instrucciones WRITE se controla internamente. El tamaño físico de la página se ignora cuando la posición del papel no está definida correctamente para el compilador ILE COBOL/400. Para un archivo

que tenga una cláusula LINAGE y esté asignado a PRINTER, la paginación está formada por el espacio hasta el final de la página lógica (cuerpo de página) y el espacio pasado los márgenes superior e inferior.

La utilización de la cláusula LINAGE degrada el rendimiento. La cláusula LINAGE debe utilizarse sólo cuando sea necesario. Si la paginación física es aceptable, la cláusula LINAGE no será necesaria.

Descripción de archivos de impresora descritos externamente (FORMATFILE)

Los archivos de impresora descritos externamente se asignan a un dispositivo de FORMATFILE. El término FORMATFILE se utiliza porque la expresión FORMAT es válida en las instrucciones WRITE para el archivo y el formato de los datos se especifica en las DDS para el archivo. Una entrada de descripción de archivos en la DATA DIVISION, que describe un archivo de impresora descrito externamente, tiene la siguiente apariencia:

```
FD  archivo-impresión.
01  registro-impresión.
    COPY DDS-ALL-FORMATS-O OF dds-archivo-impresión.
```

Cree las DDS para el archivo FORMATFILE que desee utilizar.Consulte la publicación *DDS Reference* para saber cómo crear las DDS.

Una vez creadas las DDS para el archivo FORMATFILE, utilice la instrucción COPY de Formato 2 para describir el diseño del registro de datos de archivos de impresora. Al compilar el programa ILE COBOL/400, COPY de Formato 2 creará las instrucciones de DATA DIVISION para describir el archivo de impresora. Utilice la opción DDS-ALL-FORMATS-O de la instrucción COPY de Formato 2 para generar una área de almacenamiento para todos los formatos.

Cuando haya especificado un dispositivo de FORMATFILE, puede obtener el formato de la salida impresa de dos formas:

1. Elija los formatos a imprimir y su orden, utilizando valores apropiados en las expresiones FORMAT especificadas para las instrucciones WRITE. Por ejemplo, utilice un formato una vez por página para producir una cabecera y utilice otro formato para producir las líneas de detalle de la página.
2. Elija las opciones apropiadas cuando se imprima cada formato, estableciendo los valores de indicador y pasando estos indicadores mediante la expresión INDICATOR de la instrucción WRITE. Por ejemplo, los campos pueden estar subrayados, las líneas en blanco pueden producirse antes o después de imprimir el formato, o puede ignorarse la impresión de cierto tipo de campos.

La cláusula LINAGE no debería utilizarse para archivos asignados a FORMATFILE. Si se utiliza, se emitirá un mensaje de error de tiempo de compilación indicando que la cláusula LINAGE se ha ignorado.

Grabación en archivos de impresora

Antes de poder grabar en un archivo de impresora, primero debe abrir el archivo. Utilice la instrucción OPEN de Formato 1 para abrir un archivo de impresora. Un archivo de impresora debe abrirse para OUTPUT.

```
OPEN OUTPUT nombre-archivo-impresora.
```

Utilice la instrucción WRITE para enviar la salida a un archivo de impresora. Utilice la instrucción WRITE de Formato 1 cuando grabe en un archivo de impresora descrito por programa.Utilice la instrucción WRITE de Formato 3 cuando grabe en un archivo de impresora descrito externamente.

Cuando se especifica un nombre mnemotécnico asociado con el nombre de función CSP en la expresión ADVANCING de una instrucción WRITE para un archivo de impresora, tiene el mismo efecto que especificar ADVANCING 0 LINES. Cuando se especifica un nombre mnemotécnico asociado con el nombre de función C01 en la expresión ADVANCING de una instrucción WRITE para un archivo de impresora, tiene el mismo efecto que especificar ADVANCING PAGE.

La expresión ADVANCING no puede especificarse en instrucciones WRITE para archivos con un ASSIGN en el tipo de dispositivo FORMATFILE.

Cuando termine de utilizar un archivo de impresora, debe cerrarlo.Utilice la instrucción CLOSE de Formato 1 para cerrar el archivo de impresora. Cuando cierre el archivo, no podrá procesarlo de nuevo hasta que lo vuelva a abrir.

```
CLOSE nombre-archivo-impresora.
```

Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400

Este programa imprime registros detallados de empleados para todos los empleados varones de un archivo de personal.Los registros de entrada se organizan en orden ascendente de número de empleado.El archivo de entrada y el de salida se describen externamente.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* DDS DE ARCHIVO FÍSICO PARA ARCHIVO DE EMPLEADOS COMO EJEMPLO DE ARCHIVO FORMATFILE
     A
     A                                     UNIQUE
     A          R PERSREC
     A            EMPLNO         6S
     A            NAME          30
     A            ADDRESS1      35
     A            ADDRESS2      20
     A            BIRTHDATE      6
     A            MARSTAT        1
     A            SPOUSENAME    30
     A            NUMCHILD       2S
     A          K EMPLNO
```

Figura 85. Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400 -- DDS de archivos físicos

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* DDS DE ARCHIVO DE IMPRESORA PARA EJEMPLO DE ARCHIVO FORMATFILE
     A*
     A                                     1 INDARA REF(PERSFILE)
     A          R HEADING 2                  SKIPB(1) SPACE(3)  3
     A                                     15'LISTADO DE EMPLEADOS'
     A                                       UNDERLINE
     A                                     33'- ORDENADO POR'
     A            ORDERTYPE     15         46
     A                                     80DATE EDTCDE(Y)
     A                                     93TIME   4
     A                                    115'PÁGINA:'
     A                                     +1PAGNBR EDTCDE(3)
     A*
     A          R DETAIL 5                   SPACEA(3) 6
     A*    LINE 1
     A                                      1'NOMBRE:'
     A            NAME      R             11UNDERLINE
     A                                     55'NÚMERO DE EMPLEADO:'
     A            EMPLNO    R             73
     A                                     87'FECHA DE NACIMIENTO:'
     A            BIRTHDATE R            103SPACEA(1) 7
     A*    LINE 2
     A                                      1'DIRECCIÓN:'
     A            ADDRESS1  R             11
     A                                     55'ESTADO CIVIL:'
     A            MARSTAT   R             73
     A  01                                 87'NOMBRE CÓNYUJE:'
     A  01  8     SPOUSENAMER            103
     A*    LINE 3
     A            ADDRESS2  R             11SPACEB(1)
     A                                     55'HIJOS:'
     A            NUMCHILD  R             73EDTCDE(3) 9
```

Figura 86. Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400 -- DDS de archivos de impresora

1 INDARA especifica que se utilizará una área de indicador separada para el archivo.

2 HEADING es el nombre de formato que proporciona cabeceras para cada página.

3 SKIPB(1) y SPACEA(3) se utilizan para:

1. Saltar a la línea 1 de la página siguiente antes de que se imprima el formato HEADING.
2. Dejar 3 líneas en blanco después de imprimir el formato HEADING.

4 DATE, TIME y PAGNBR se utilizan para imprimir la fecha actual, la hora actual y el número de página cuando se imprima el formato HEADING.

5 DETAIL es el nombre de formato utilizado para imprimir la línea de datos para cada empleado del archivo de personal.

6 SPACEA(3) deja tres líneas en blanco después de cada línea de datos de los empleados.

7 SPACEA(1) imprime una línea en blanco después de imprimir el campo BIRTHDATE. Como resultado, los siguientes campos del mismo formato se imprimen en una nueva línea.

8 01 significa que estos campos se imprimen sólo si el programa ILE COBOL/400 activa el indicador 01 y lo pasa al imprimir el formato DETAIL.

9 EDTCDE(3) se utiliza para eliminar ceros iniciales al imprimir este campo numérico.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/FRMTFILE         AS400SYS   96/07/04 11:56:45         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     FRMTFILE.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-AS400.
    6     000700   OBJECT-COMPUTER. IBM-AS400.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT PERSREPT ASSIGN TO FORMATFILE-PERSREPT-SI  1
   11     001100         ORGANIZATION IS SEQUENTIAL.
   12     001200     SELECT PERSFILE ASSIGN TO DATABASE-PERSFILE
   14     001300         ORGANIZATION IS INDEXED
   15     001400         ACCESS MODE IS SEQUENTIAL
   16     001500         RECORD IS EXTERNALLY-DESCRIBED-KEY.
          001600
   17     001700 DATA DIVISION.
   18     001800 FILE SECTION.
   19     001900 FD  PERSREPT.
   20     002000 01  PERSREPT-REC.
          002100     COPY DDS-ALL-FORMATS-O OF PERSREPT.  2
   21    +000001       05  PERSREPT-RECORD PIC X(130).                                  <-ALL-FMTS
         +000002* FORMATO SALIDA:HEADING    DE ARCHIVO PERSREPT   DE BIBLIOTECA TESTLIB  <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   22    +000004       05  HEADING-O     REDEFINES PERSREPT-RECORD.                    <-ALL-FMTS
   23    +000005           06 ORDERTYPE             PIC X(15).                        <-ALL-FMTS
         +000006* FORMATO SALIDA:DETAIL   DE ARCHIVO PERSREPT  DE BIBLIOTECA TESTLIB    <-ALL-FMTS
         +000007*                                                                      <-ALL-FMTS
   24    +000008       05  DETAIL-O      REDEFINES PERSREPT-RECORD.  3                 <-ALL-FMTS
   25    +000009           06 NAME                 PIC X(30).                        <-ALL-FMTS
   26    +000010           06 EMPLNO                PIC S9(6).                        <-ALL-FMTS
   27    +000011           06 BIRTHDATE             PIC X(6).                         <-ALL-FMTS
   28    +000012           06 ADDRESS1              PIC X(35).                        <-ALL-FMTS
   29    +000013           06 MARSTAT               PIC X(1).                         <-ALL-FMTS
   30    +000014           06 SPOUSENAME            PIC X(30).                        <-ALL-FMTS
   31    +000015           06 ADDRESS2              PIC X(20).                        <-ALL-FMTS
   32    +000016           06 NUMCHILD              PIC S9(2).                        <-ALL-FMTS
   33     002200 FD  PERSFILE.
   34     002300 01  PERSFILE-REC.
          002400     COPY DDS-ALL-FORMATS-O OF PERSFILE.
   35    +000001       05  PERSFILE-RECORD PIC X(130).                                  <-ALL-FMTS
         +000002*    FORMATO E-S:PERSREC    DE ARCHIVO PERSFILE   DE BIBLIOTECA TESTLIB  <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
         +000004*LAS DEFINICIONES CLAVE PARA FORMATO DE REGISTRO  PERSREC                    <-ALL-FMTS
         +000005*  NÚMERO              NOMBRE              RECUPERACIÓN    ALTSEQ        <-ALL-FMTS
         +000006*   0001   EMPLNO                         ASCENDIENTE       NO          <-ALL-FMTS
   36    +000007       05  PERSREC       REDEFINES PERSFILE-RECORD.                    <-ALL-FMTS
   37    +000008           06 EMPLNO                PIC S9(6).                        <-ALL-FMTS
   38    +000009           06 NAME                  PIC X(30).                        <-ALL-FMTS
   39    +000010           06 ADDRESS1              PIC X(35).                        <-ALL-FMTS
   40    +000011           06 ADDRESS2              PIC X(20).                        <-ALL-FMTS
   41    +000012           06 BIRTHDATE             PIC X(6).                         <-ALL-FMTS
   42    +000013           06 MARSTAT               PIC X(1).                         <-ALL-FMTS
   43    +000014           06 SPOUSENAME            PIC X(30).                        <-ALL-FMTS
   44    +000015           06 NUMCHILD              PIC S9(2).                        <-ALL-FMTS
```

Figura 87 (Parte 1 de 3). Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/FRMTFILE         AS400SYS   96/07/04 11:56:45         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


        002500
  45    002600 WORKING-STORAGE SECTION.
  46    002700 77  HEAD-ORDER                   PIC X(15)
        002800                                    VALUE "NÚMERO DE EMPLEADO".
  47    002900 01  PERSREPT-INDICS.
        003000     COPY DDS-ALL-FORMATS-O-INDIC OF PERSREPT.  4
  48   +000001       05  PERSREPT-RECORD.                                          <-ALL-FMTS
       +000002* FORAMTO SALIDA:HEADING    DE ARCHIVO PERSREPT   DE BIBLIOTECA TESTLIB       <-ALL-FMTS
       +000003*                                                                   <-ALL-FMTS
       +000004*          06 HEADING-O-INDIC.                                      <-ALL-FMTS
       +000005* FORMATO SALIDA:DETAIL     DE ARCHIVO PERSREPT   DE BIBLIOTECA TESTLIB       <-ALL-FMTS
       +000006*                                                                   <-ALL-FMTS
  49   +000007           06 DETAIL-O-INDIC.                                       <-ALL-FMTS
  50   +000008                07 IN01             PIC 1  INDIC 01.              <-ALL-FMTS
        003100
  51    003200 77  EOF-FLAG                      PIC X(1)
        003300                                    VALUE "0".
  52    003400    88  NOT-END-OF-FILE            VALUE "0".
  53    003500    88  END-OF-FILE                VALUE "1".
  54    003600 77  MARRIED                      PIC X(1)
        003700                                    VALUE "M".
        003800
  55    003900 PROCEDURE DIVISION.
        004000 MAIN-PROGRAM SECTION.
        004100 MAINLINE.
  56    004200     OPEN INPUT PERSFILE
        004300          OUTPUT PERSREPT.
  57    004400     PERFORM HEADING-LINE.
  58    004500     PERFORM UNTIL END-OF-FILE
  59    004600         READ PERSFILE
  60    004700             AT END SET END-OF-FILE TO TRUE
  61    004800             NOT AT END PERFORM PRINT-RECORD  5
        004900         END-READ
        005000     END-PERFORM
  62    005100     CLOSE PERSFILE
        005200           PERSREPT.
  63    005300     STOP RUN.
        005400
        005500 PRINT-RECORD.
  64    005600     MOVE CORR PERSREC TO DETAIL-O.  6
              *** CORRESPONDING items for statement 005600:
              ***       EMPLNO
              ***       NAME
              ***       ADDRESS1
              ***       ADDRESS2
              ***       BIRTHDATE
              ***       MARSTAT
              ***       SPOUSENAME
              ***       NUMCHILD
              *** End of CORRESPONDING items for statement 005600
  65    005700     IF MARSTAT IN PERSFILE-REC IS EQUAL MARRIED THEN  7
  66    005800         MOVE B"1" TO IN01 IN DETAIL-O-INDIC
        005900     ELSE
  67    006000         MOVE B"0" TO IN01 IN DETAIL-O-INDIC
        006100     END-IF
  68    006200     WRITE PERSREPT-REC FORMAT IS "DETAIL"  8
        006300         INDICATORS ARE DETAIL-O-INDIC
```

Figura 87 (Parte 2 de 3). Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/FRMTFILE         AS400SYS   96/07/04 11:56:45         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   69     006400         AT EOP PERFORM HEADING-LINE  9
          006500     END-WRITE.                                                                  96/11/08
          006600
          006700 HEADING-LINE.
   70     006800     MOVE HEAD-ORDER TO ORDERTYPE
   71     006900     WRITE PERSREPT-REC FORMAT IS "HEADING"
          007000     END-WRITE.                                                                  96/11/08
          007100

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 87 (Parte 3 de 3). Ejemplo de utilización de archivos FORMATFILE en un programa ILE COBOL/400

1 El archivo de impresora descrito externamente se asigna al dispositivo FORMATFILE. SI indica que se ha especificado una área de indicador separada en las DDS.

2 La instrucción COPY de Formato 2 se utiliza para copiar los campos para el archivo de impresora en el programa.

3 Observe que, aunque los campos del formato DETAIL se imprimirán en tres líneas separadas, se definen en un registro.

4 La instrucción COPY de Formato 2 se utiliza para copiar los indicadores utilizados en el archivo de impresora del programa.

5 El párrafo PROCESS-RECORD procesa PRINT-RECORD para cada registro de empleado.

6 Todos los campos del registro de empleados se trasladan al registro para el formato DETAIL.

7 Si el empleado está casado, el indicador 01 se activará; si no, el indicador se desactivará, evitando que se imprima el campo de nombre de la esposa en DETAIL.

8 El formato DETAIL se imprime con el indicador 01 pasado para controlar la impresión.

9 Si se ha excedido el número de líneas por página, se producirá un END-OF-PAGE. Se imprimirá el formato HEADING en una nueva página.

Acceso a archivos almacenados en dispositivos de cinta

Utilice **archivos de cinta** para leer y grabar registros en un dispositivo de cinta.Los archivos almacenados en dispositivos de cinta pueden dividirse en las dos categorías siguientes:

- **Volumen único secuencial:** Un archivo secuencial contenido completamente en un volumen.Este volumen puede contener más de un archivo.
- **Multivolumen secuencial:** Un archivo secuencial contenido en más de un volumen.

Puede crear sus propios archivos de cinta utilizando el mandato Create Tape File (CRTTAPF). Vea la publicación *CL Reference* para obtener más información sobre el mandato CRTTAPF. Alternativamente, puede utilizar el archivo de cinta suminis-

trado por IBM QTAPE. El archivo de cinta identifica el dispositivo de cinta a utilizar.

Para utilizar un archivo almacenado en un dispositivo de cinta, en el programa ILE COBOL/400 debe:

- denominar el archivo mediante una entrada de control de archivos del párrafo FILE-CONTROL de la ENVIRONMENT DIVISION
- describir el archivo mediante una entrada de descripción de archivos en la DATA DIVISION

Sólo puede almacenar un archivo secuencial en un dispositivo de cinta, debido a que a los dispositivos de cinta sólo puede accederse secuencialmente. Los archivos almacenados en un dispositivo de cinta pueden tener registros de longitud fija o variable.

Las operaciones válidas de archivo para un dispositivo de cinta son OPEN, CLOSE, READ y WRITE.

Denominación de archivos almacenados en dispositivos de cinta

Para utilizar un archivo secuencial almacenado en un dispositivo de cinta, en el programa ILE COBOL/400, debe denominar el archivo mediante una entrada de control de archivos en el párrafo FILE-CONTROL de la ENVIRONMENT DIVISION. Vea la *ILE COBOL/400 Reference* para obtener una descripción completa del párrafo FILE-CONTROL.

Denomine el archivo del párrafo FILE-CONTROL de la forma siguiente:

```
FILE-CONTROL.
    SELECT nombre-archivo-secuencial
        ASSIGN TO TAPEFILE-nombre_dispositivo_cinta
        ORGANIZATION IS SEQUENTIAL.
```

Utilice la cláusula SELECT para elegir un archivo. Este archivo debe identificarse mediante una entrada FD en la DATA DIVISION.

Utilice la cláusula ASSIGN para asociar el archivo con un dispositivo de cinta. Debe especificar un tipo de dispositivo TAPEFILE en la cláusula ASSIGN para utilizar un archivo de cinta.

Utilice ORGANIZATION IS SEQUENTIAL en la entrada de control de archivos, al denominar un archivo al que accederá mediante un archivo de cinta.

Descripción de archivos almacenados en dispositivos de cinta

Cuando ya haya nombrado el archivo secuencial en la Environment Division, entonces debe describir el archivo mediante una entrada de descripción de archivos en la DATA DIVISION. Vea la publicación *ILE COBOL/400 Reference* para obtener una descripción completa de la Entrada de descripción de archivos.Utilice la entrada de descripción de archivos de Formato 3 para describir un archivo secuencial al que se acceda mediante un archivo de cinta.

Los archivos de cinta no tienen especificaciones de descripción de datos (DDS). Un archivo secuencial que se almacena en un dispositivo de cinta debe ser un archivo descrito por programa.El programa ILE COBOL/400 debe describir los

campos del formato de registro, de manera que el programa pueda organizar los datos recibidos o enviados al dispositivo de cinta de la forma especificada por la descripción de archivos de cinta.

Una entrada de descripción de archivos simple en la Data Division, que describe un archivo secuencial al que se accede mediante un archivo de cinta, tiene la apariencia siguiente:

```
FD  nombre-archivo-secuencial.
01  registro-archivo-secuencial.
    05  elemento-registro-1   PIC ... .
    05  elemento-registro-2   PIC ... .
    05  elemento-registro-3   PIC ... .
              .
              .
              .
```

Descripción de archivos de cinta con registros de longitud variable

Puede almacenar archivos que tengan registros de longitud variable en un dispositivo de cinta. Especifique la cláusula RECORD de Formato 3 con la entrada FD del archivo para definir las longitudes de registro máximas o mínimas para el archivo.

Una entrada de descripción de archivos simple en la Data Division, que describe un archivo secuencial con registros de longitud variable, tiene la apariencia siguiente:

```
FILE SECTION.
FD  nombre-archivo-secuencial
    RECORD IS VARYING IN SIZE
              FROM entero-6 TO entero-7
              DEPENDING ON nombre-datos-1.
01  registro-tamaño-mínimo.
    05  elemento-tamaño-mínimo     PIC X(entero-6).
01  registro-tamaño-máximo.
    05  elemento-tamaño-máximo    PIC X(entero-7).
              .
              .
              .

WORKING-STORAGE SECTION.
77  nombre-datos-1                PIC 9(5).
              .
              .
              .
```

El tamaño de registro mínimo de cualquier registro del archivo lo define el *entero-6*. El tamaño de registro máximo de cualquier registro del archivo lo define el *entero-7*. No cree descripciones de registro para el archivo que contenga una longitud de registro inferior a la especificada en *entero-6* ni superior a la especificada en *entero-7*. Si cualquier descripción de registro rompe esta norma, el compilador ILE COBOL/400 emitirá un mensaje de error de tiempo de compilación. Entonces el compilador ILE COBOL/400 utilizará los límites implícitos en la descripción de registros. El compilador ILE COBOL/400 también emitirá un mensaje de error de

tiempo de compilación cuando ninguna de las descripciones de registro tenga una longitud de registro tan larga como el *entero-7*.

Cuando se realiza una instrucción READ o WRITE en un registro de longitud variable, el tamaño de ese registro lo define el contenido de *nombre-datos-1*.

Consulte la cláusula RECORD de Formato 3 en la publicación *ILE COBOL/400 Reference* para obtener más descripciones de cómo se manejan los registros de longitud variable.

Lectura y grabación de archivos almacenados en dispositivos de cinta

Antes de poder leer o grabar en un archivo almacenado en un dispositivo de cinta, primero debe abrirlo. Utilice la instrucción OPEN de Formato 1 para abrir el archivo. Para leer un archivo almacenado en un dispositivo de cinta, debe abrirlo en la modalidad INPUT. Para grabar en un archivo almacenado en un dispositivo de cinta, debe abrirlo en modalidad OUTPUT o EXTEND. Un archivo almacenado en un dispositivo de cinta **no** puede abrirse en modalidad de E-S. A continuación encontrará ejemplos de la instrucción OPEN.

```
OPEN INPUT nombre-archivo-secuencial.

OPEN OUTPUT nombre-archivo-secuencial.

OPEN EXTEND nombre-archivo-secuencial.
```

Utilice la instrucción READ de Formato 1 para leer un registro de un archivo secuencial almacenado en un dispositivo de cinta.La instrucción READ hace que el próximo registro lógico del archivo esté disponible para el programa ILE COBOL/400. Para un archivo multivolumen secuencial, si el final del volumen se reconoce durante el proceso de la instrucción READ y no se ha llegado al final lógico del archivo, se realizarán las acciones siguientes en el orden indicado:

1. Se procesa el procedimiento de etiqueta de volumen final estándar.
2. Se produce una conmutación de volumen.
3. Se ejecuta el procedimiento de etiqueta de volumen inicial estándar.
4. Queda disponible el primer registro de datos del próximo volumen.

El programa ILE COBOL/400 no recibirá ninguna indicación de que las secciones anteriores se han producido durante la operación de lectura.

Utilice la instrucción WRITE de Formato 1 para grabar un registro en un archivo secuencial almacenado en un dispositivo de cinta. Para un archivo de multivolumen secuencial, si el final del volumen se reconoce durante el proceso de la instrucción WRITE, se realizarán las acciones siguientes en el orden indicado:

1. Se ejecuta el procedimiento de etiqueta de volumen final estándar.
2. Se produce una conmutación de volumen.
3. Se ejecuta el procedimiento de etiqueta de volumen inicial estándar.
4. El registro de datos se graba en el próximo volumen.

No se devuelve ninguna indicación al programa ILE COBOL/400 conforme se ha producido una condición de final de volumen.

Cuando termine de utilizar un archivo almacenado en un dispositivo de cinta, debe cerrarlo.Utilice la instrucción CLOSE de Formato 1 para cerrar el archivo. Cuando cierre el archivo, no podrá seguir procesándolo hasta que vuelva a abrirlo.

```
CLOSE nombre-archivo-secuencial.
```

La instrucción CLOSE también le proporciona la oportunidad de rebobinar y descargar el volumen.

Normalmente, cuando se ejecuta la instrucción CLOSE en un archivo de cinta, el volumen se rebobina.Sin embargo, si desea que el volumen actual se quede en su posición actual después de cerrar el archivo, especifique la expresión NO REWIND en la instrucción CLOSE. Cuando se especifica NO REWIND, el carrete no se rebobinará.

Para archivos de cinta de multivolumen secuencial, la expresión REEL/UNIT FOR REMOVAL hace que se rebobine y descargue el volumen actual.Entonces se notifica al sistema que se ha extraído el volumen.

Para obtener más detalles sobre rebobinar y descargar volúmenes, consulte la información sobre la instrucción CLOSE de Formato 1 en la publicación *ILE COBOL/400 Reference*.

Lectura y grabación de archivos de cinta con registros de longitud variable

Al leer o grabar registros de longitud variable en un archivo de cinta, asegúrese de que el registro de longitud variable máxima es inferior o igual a la longitud de registro máxima para la cinta. La longitud de registro máxima para la cinta se determina en el momento de abrirla para OUTPUT. Si la longitud de registro máxima en la cinta es inferior a cualquiera de los registros de longitud variable grabados en ella, estos registros se truncarán a la longitud de registro máxima para la cinta.

Utilice la instrucción READ de Formato 1 para leer un registro de un archivo secuencial almacenado en un dispositivo de cinta.La instrucción READ hace que el próximo registro lógico del archivo esté disponible para el programa ILE COBOL/400.

Si la operación READ es satisfactoria entonces el *nombre-datos-1*, si se ha especificado, tendrá el número de posiciones de caracteres del registro leído en ese momento. Si la operación READ no es satisfactoria, entonces el *nombre-datos- 1* tendrá el valor que tenía antes de que se intentara la operación READ.

Al especificar la expresión INTO en la instrucción READ, el número de posiciones de caracteres del registro actual, que participa como elemento de envío en la instrucción MOVE implícita, lo determina

- el contenido de *nombre-datos-1*, si *nombre-datos-1* se especifica, o
- el número de posiciones de caracteres del registro leído en ese momento, si no se especifica el *nombre-datos-1*.

Cuando se realiza la instrucción READ, si el número de posiciones de caracteres del registro leído es inferior a la longitud de registro mínima especificada por las entradas de descripción de registros para el archivo, la parte del área de registro situada a la derecha del último caracter válido leído se rellena con espacios en blanco. Si el número de posiciones de caracteres del registro leído es superior a la longitud de registro máxima especificada por las entradas de descripción de registro para el archivo, el registro se truncará a la derecha del tamaño de registro máximo especificado en las entradas de descripción de registro. Da un estado de

archivo de 04 cuando se lee un registro, la longitud del cual no se encuentra dentro del rango de longitudes de registro mínima o máxima definidas en las entradas de descripción de archivos para el archivo.

Utilice la instrucción WRITE de Formato 1 para grabar un registro de longitud variable en un archivo secuencial almacenado en un dispositivo de cinta. Especifique la longitud del registro a grabar en el *nombre-datos-1*. Si no especifica el *nombre-datos-1*, la longitud del registro a grabar se determina de la siguiente forma:

- si el registro contiene un elemento OCCURS...DEPENDING ON, mediante la suma de la parte fija y la parte de la tabla descrita, por el número de apariciones en el momento en que se ejecuta la instrucción WRITE
- si el registro no contiene un elemento OCCURS...DEPENDING ON, por el número de posiciones de caracteres en la definición del registro.

Acceso a archivos almacenados en dispositivos de disquete

Utilice **archivos de disquete** para leer y grabar registros en disquetes que estén en el dispositivo de disquete y que hayan sido inicializados en el formato de intercambio básico, H o I. Los archivos almacenados en dispositivos de disquete pueden dividirse en las dos categorías siguientes:

- **Volumen único secuencial:** Un disquete que contiene un archivo secuencial completo.Este disquete puede contener más de un archivo.
- **Multivolumen secuencial:** Un archivo secuencial contenido en más de un disquete.

Puede crear sus propios archivos de disquetes utilizando el mandato Crear archivo de disquete (CRTDKTF). Vea la publicación *CL Reference* para obtener más información sobre el mandato CRTDKTF. Alternativamente, puede utilizar el archivo de disquete suministrado por IBM por omisión QDKT. El archivo de disquete identifica el dispositivo de disquete a utilizar.

Para utilizar un archivo almacenado en un dispositivo de disquete, en el programa ILE COBOL/400 debe:

- denominar el archivo mediante una entrada de control de archivos del párrafo FILE-CONTROL de la ENVIRONMENT DIVISION
- describir el archivo mediante una entrada de descripción de archivos en la DATA DIVISION

Sólo puede almacenar un archivo secuencial en un dispositivo de disquete, debido a que a los dispositivos de disquete sólo puede accederse secuencialmente. Las operaciones de archivos válidas para un dispositivo de disquete son OPEN, CLOSE, READ y WRITE.

Denominación de archivos almacenados en dispositivos de disquete

Para utilizar un archivo secuencial almacenado en un dispositivo de disquete en el programa ILE COBOL/400, debe denominar el archivo mediante una entrada de control de archivo en el párrafo FILE-CONTROL de la ENVIRONMENT DIVISION. Vea la publicación *ILE COBOL/400 Reference* para obtener una descripción del párrafo FILE-CONTROL.

Denomine el archivo del párrafo FILE-CONTROL de la forma siguiente:

```
FILE-CONTROL.
    SELECT nombre-archivo-secuencial
        ASSIGN TO DISKETTE-nombre_dispositivo_disquete
        ORGANIZATION IS SEQUENTIAL.
```

Utilice la cláusula SELECT para elegir un archivo. Este archivo debe identificarse mediante una entrada FD en la DATA DIVISION.

Utilice la cláusula ASSIGN para asociar el archivo con un dispositivo de disquete. Debe especificar un tipo de dispositivo de DISKETTE en la cláusula ASSIGN para utilizar un archivo de disquete.

Utilice ORGANIZATION IS SEQUENTIAL en la entrada de control de archivos, cuando denomine un archivo al que se accederá mediante un archivo de disquete.

Descripción de archivos almacenados en dispositivos de disquete

Cuando ya haya nombrado el archivo secuencial en la Environment Division entonces debe describir el archivo mediante una entrada de descripción de archivos en la DATA DIVISION. Vea la *ILE COBOL/400 Reference* para obtener una descripción completa de la Entrada de descripción de archivos.Utilice la Entrada de descripción de archivos de Formato 2 para describir un archivo secuencial al que se accede mediante un archivo de disquete.

Los archivos de disquetes no tienen especificaciones de descripción de datos (DDS). Un archivo secuencial que se almacena en un dispositivo de disquete debe ser un archivo descrito por programa.El programa ILE COBOL/400 debe describir los campos en el formato de registro, de manera que el programa pueda organizar los datos recibidos desde, o enviados al dispositivo de disquete de la forma especificada por la descripción de archivos de disquete.

Una entrada de descripción de archivos simple en la Data Division, que describe un archivo secuencial al que se accede mediante un archivo de disquete, tiene la apariencia siguiente:

```
FD  nombre-archivo-secuencial.
01  registro-archivo-secuencial.
    05  elemento-registro-1   PIC ... .
    05  elemento-registro-2   PIC ... .
    05  elemento-registro-3   PIC ... .
              .
              .
              .
```

Lectura y grabación de archivos almacenados en dispositivos de disquete

Antes de poder leer o grabar en un archivo almacenado en un dispositivo de disquete, primero debe abrirlo. Utilice la instrucción OPEN de Formato 1 para abrir el archivo. Para leer un archivo almacenado en un dispositivo de disquetes, debe abrirlo en la modalidad INPUT. Para grabar en un archivo almacenado en un dispositivo de disquetes, debe abrirlo en modalidad OUTPUT o EXTEND. Un archivo almacenado en un dispositivo de disquete **no** puede abrirse en modalidad de E-S. A continuación encontrará ejemplos de la instrucción OPEN.

```
OPEN INPUT nombre-archivo-secuencial.

OPEN OUTPUT nombre-archivo-secuencial.

OPEN EXTEND nombre-archivo-secuencial.
```

Utilice la instrucción READ de Formato 1 para leer un registro de un archivo secuencial almacenado en un dispositivo de disquete.La instrucción READ hace que el próximo registro lógico del archivo esté disponible para el programa ILE COBOL/400.

Al leer registros de un archivo de entrada, la longitud de registro que especifique en el programa COBOL debe ser la misma que la longitud de registro que se encuentra en la etiqueta del archivo de datos del disquete. Si la longitud del registro especificada en el programa COBOL no es igual a la longitud de los registros del archivo de datos, los archivos se rellenarán o truncarán en la longitud especificada en el programa.

Para un archivo multivolumen secuencial, si el final del volumen se reconoce durante el proceso de la instrucción READ y no se ha llegado al final lógico del archivo, se realizarán las acciones siguientes en el orden indicado:

1. Se procesa el procedimiento de etiqueta de volumen final estándar.
2. Se produce una conmutación de volumen.
3. Se ejecuta el procedimiento de etiqueta de volumen inicial estándar.
4. Queda disponible el primer registro de datos del próximo volumen.

El programa ILE COBOL/400 no recibirá ninguna indicación de que las secciones anteriores se han producido durante la operación de lectura.

Utilice la instrucción WRITE de Formato 1 para grabar un registro en un archivo secuencial almacenado en un dispositivo de disquetes.

Al grabar registros en el archivo de salida, debe especificar la longitud de registro en el programa COBOL.Si la longitud de registro especificada en el programa excede la longitud para la cual está formateado el disquete, se enviará un mensaje de diagnóstico al programa y se truncarán los registros.
Las longitudes de registro máximas soportadas para dispositivos de disquetes, mediante tipo de intercambio, son las siguientes:

Tipo de intercambio	*Longitud de registro máxima soportada*
Intercambio básico	128 bytes
Intercambio H	256 bytes
Intercambio I	4096 bytes

Para un archivo de multivolumen secuencial, si el final del volumen se reconoce durante el proceso de la instrucción WRITE, se realizarán las acciones siguientes en el orden indicado:

1. Se ejecuta el procedimiento de etiqueta de volumen final estándar.
2. Se produce una conmutación de volumen.
3. Se ejecuta el procedimiento de etiqueta de volumen inicial estándar.
4. El registro de datos se graba en el próximo volumen.

No se devuelve al programa COBOL ninguna indicación de que se ha producido una condición de final de volumen.

Cuando termine de utilizar un archivo almacenado en un dispositivo de disquete, debe cerrarlo.Utilice la instrucción CLOSE de Formato 1 para cerrar el archivo. Cuando cierre el archivo, no podrá procesarlo de nuevo hasta que lo vuelva a abrir.

```
CLOSE nombre-archivo-secuencial.
```

Acceso a dispositivos de pantalla y archivos ICF

Utilice los archivos de pantalla para intercambiar información entre el programa ILE COBOL/400 y un dispositivo de pantalla, como por ejemplo una estación de trabajo.Un archivo de pantalla se utiliza para definir el formato de la información que se presentará en una pantalla y cómo el sistema procesará dicha información de camino a y desde la pantalla. ILE COBOL/400 utiliza archivos TRANSACTION para comunicarse interactivamente con un dispositivo de pantalla.

Utilice los archivos de Función de comunicaciones intersistemas (ICF) para permitir que un programa en un sistema se comunique con un programa situado en el mismo sistema o en un sistema remoto. ILE COBOL/400 utiliza los archivos TRANSACTION para comunicación de intersistemas.

Vea Capítulo 17, “Utilización de archivos de transacción” en la página 395 para obtener información sobre cómo utilizar los archivos TRANSACTION con dispositivos de pantalla y archivos ICF.

Capítulo 16. Utilización de archivos DISK y DATABASE

Los archivos de base de datos, asociados con los dispositivos ILE COBOL/400 DATABASE y DISK, pueden ser:

- Archivos descritos externamente, cuyos campos se describen al OS/400 mediante DDS
- Archivos descritos por programa, cuyos campos se describen en el programa que utiliza el archivo.

Los archivos de base de datos se crean utilizando los mandatos de CL Crear archivo físico (CRTPF) o Crear archivo lógico (CRTLF). Consulte la publicación *CL Reference* para obtener una descripción de estos mandatos.

Este capítulo describe:

- las diferencias entre archivos DISK y DATABASE
- las formas en que se organizan los archivos DISK y DATABASE
- los distintos métodos de procesar archivos DISK y DATABASE

Diferencias entre archivos DISK y DATABASE

Utilice el tipo de dispositivo DISK para asociar un archivo del programa ILE COBOL/400 con un archivo de base de datos físico o con un archivo de base de datos lógico de formato único.Cuando elija DISK como el tipo de dispositivo, no podrá utilizar ninguna extensión de base de datos ILE COBOL/400. El tipo de dispositivo DISK soporta la creación dinámica de archivos (excepto para archivos indexados) y registros de longitud variable.

Utilice el tipo de dispositivo DATABASE para asociar un archivo del programa ILE COBOL/400 con cualquier archivo de base de datos o DDM.Escoger DATABASE como tipo de dispositivo le permite utilizar cualquier extensión de base de datos ILE COBOL/400. Estas extensiones de base de datos incluyen lo siguiente:

- control de compromiso
- claves de archivo duplicadas
- formatos de registro
- archivos descritos externamente

Sin embargo, el tipo de dispositivo DATABASE no soporta la creación dinámica de archivos ni los registros de longitud variable.

Organización de archivos y vías de acceso a archivos AS/400

Existen dos tipos de vías de acceso para acceder a los registros de un archivo:

- vía de acceso en secuencia de clave
- vía de acceso en secuencia de llegada.

Un archivo con una **vía de acceso en secuencia de clave** puede procesarse en ILE COBOL/400 como un archivo con organización SEQUENTIAL, RELATIVE o INDEXED.

Para procesar un archivo en secuencia de clave como un archivo relativo en ILE COBOL/400, debe ser un archivo físico o un archivo lógico, cuyos miembros se basen en un miembro del archivo físico. Para procesar un archivo en secuencia de clave como un archivo secuencial en ILE COBOL/400, debe ser un archivo físico o un archivo lógico basado en un miembro del archivo físico y que no contenga lógica de selección/omisión.

Un archivo con una **vía de acceso en secuencia de llegada** puede procesarse en ILE COBOL/400 como un archivo con organización RELATIVE o SEQUENTIAL. El archivo debe ser un archivo físico o un archivo lógico donde cada miembro del archivo lógico se basa sólo en un miembro del archivo físico.

Cuando se especifica el acceso secuencial para un archivo lógico, se accede a los registros del archivo mediante la vía de acceso por omisión para el archivo.

Métodos de proceso de archivos para archivos DISK y DATABASE

Los archivos DISK y DATABASE pueden tener la organización siguiente:

- SEQUENTIAL
- RELATIVE
- INDEXED

Cada tipo de organización de archivo utiliza métodos de proceso de archivos exclusivos.

Proceso de archivos secuenciales

Un archivo secuencial ILE COBOL/400 es un archivo en el cual se procesan los registros en el orden en que se colocaron en el archivo, es decir, en secuencia de llegada. Por ejemplo, el décimo registro del archivo ocupa la décima posición de registro y es el décimo registro a procesar. Para procesar un archivo como un archivo secuencial, debe especificar ORGANIZATION IS SEQUENTIAL en la cláusula SELECT u omitir la cláusula ORGANIZATION. A un archivo secuencial sólo puede accederse secuencialmente.

| Para escribir programas en norma COBOL que accedan a un archivo secuencial, debe crear el archivo con ciertas características. La Tabla 17 en la página 361 lista estas características y qué las controla.

Tabla 17. Características de archivos secuenciales accesibles a programas de norma COBOL

Característica	Control
El archivo debe ser un archivo físico.	Cree el archivo utilizando el mandato CL CRTPF.
El archivo no puede ser un archivo compartido.	Especifique SHARE(*NO) en el mandato CL CRTPF.
No puede especificarse ninguna clave para el archivo.	No incluya ninguna línea con K en la posición 17 en las Especificaciones de descripción de datos (DDS) del archivo.
El archivo debe tener un tipo de archivo de DATA.	Especifique FILETYPE(*DATA) en el mandato CL CRTPF.
No puede utilizarse la edición de archivos.	No especifique las palabras clave EDTCDE y EDTWRD en las DDS del archivo.
No puede especificarse la información de línea y posición.	Deje espacios en blanco en las posiciones 39 a 44 de las descripciones de campo de las DDS del archivo.
No pueden especificarse las palabras clave de espaciado y salto.	No especifique las palabras clave SPACEA, SPACEB, SKIPA y SKIPB en las DDS del archivo.
No pueden utilizarse indicadores.	Deje espacios en blanco en las posiciones 9 a 16 de todas las líneas de las DDS del archivo.
No pueden utilizarse las funciones proporcionadas por el sistema, como por ejemplo fecha, hora y número de página.	No especifique las palabras clave DATE, TIME ni PAGNBR en las DDS del archivo.
No pueden utilizarse las palabras clave de selección/omisión para el archivo.	No incluya ninguna línea con S u O en la posición 17 de las DDS del archivo. No especifique las palabras clave COMP, RANGE, VALUES o ALL.
Los registros del archivo no pueden volverse a utilizarse.	Especifique REUSEDLT(*NO) en el mandato CL CRTPF.
Los registros del archivo no pueden contener campos NULL.	No especifique la palabra clave ALWNULL en las DDS del archivo.

Las instrucciones OPEN, READ, WRITE, REWRITE y CLOSE se utilizan para acceder a datos almacenados en un archivo secuencial. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción de cada una de estas instrucciones.

Todos los archivos de base de datos físicos con organización SEQUENTIAL abiertos para OUTPUT se borran.

Para mantener la secuencia de registros en un archivo abierto en modalidad E-S (actualización), no cree ni cambie el archivo de manera que pueda volver a utilizar los registros que hay en él. Es decir, no utilice un mandato CL Cambiar archivo físico (CHGPF) con la opción REUSEDLT.

Nota: El compilador ILE COBOL/400 no comprueba si el dispositivo asociado con el archivo externo es del tipo especificado en la parte de dispositivo de nombre-asignación. El dispositivo especificado en nombre-asignación debe

coincidir con el dispositivo real al que se asigna el archivo. Vea la sección "ASSIGN Clause" de la publicación *ILE COBOL/400 Reference* para obtener más información.

Proceso de archivos relativos

Un archivo relativo ILE COBOL/400 es un archivo que se procesará mediante un número relativo de registro. Para procesar un archivo mediante un número relativo de registro, debe especificar ORGANIZATION IS RELATIVE en la instrucción SELECT para el archivo. A un archivo relativo se puede acceder secuencialmente, aleatoriamente por número de registro o dinámicamente. Un archivo relativo ILE COBOL/400 no puede tener una vía de acceso en clave.

Para escribir programas de norma COBOL que accedan a un archivo relativo, debe crear el archivo con ciertas características. .* The following table La Tabla 18 lista estas características y qué las controla.

Tabla 18. Características de archivos relativos accesibles a programas de norma COBOL

Característica	Control
El archivo debe ser un archivo físico.[1]	Cree el archivo utilizando el mandato CL CRTPF.
El archivo no puede ser un archivo compartido.	Especifique SHARE(*NO) en el mandato CL CRTPF.
No puede especificarse ninguna clave para el archivo.	No incluya ninguna línea con K en la posición 17 en las Especificaciones de descripción de datos (DDS) del archivo.
No puede especificarse una posición inicial para recuperar registros.	No emita el mandato CL OVRDBF con el parámetro POSITION.
No pueden utilizarse las palabras clave de selección/omisión para el archivo.	No incluya ninguna línea con S u O en la posición 17 de las DDS del archivo. No especifique las palabras clave COMP, RANGE, VALUES o ALL.
Los registros del archivo no pueden volverse a usar.	Especifique REUSEDLT(*NO) en el mandato CL CRTPF.
Los registros del archivo no pueden contener campos NULL.	No especifique la palabra clave ALWNULL en las DDS del archivo.

Nota:

[1] Un archivo lógico, cuyos miembros se basen en un archivo físico, puede utilizarse como un archivo relativo ILE COBOL/400.

Las instrucciones OPEN, READ, WRITE, START, REWRITE, DELETE y CLOSE se utilizan para acceder a datos almacenados en un archivo relativo. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción de cada una de estas instrucciones. La instrucción START sólo es aplicable a archivos abiertos para INPUT o I-O y a los que se accede secuencial o dinámicamente.

Para archivos relativos a los que se accede secuencialmente, se ignora la expresión KEY de la cláusula SELECT excepto para la instrucción START. Si no se especifica la expresión KEY en la instrucción START, se utilizará la expresión RELATIVE KEY de la cláusula SELECT y se asumirá KEY IS EQUAL.

Para archivos relativos a los que se accede aleatoria o dinámicamente, se utiliza la expresión RELATIVE KEY de la cláusula SELECT.

La expresión NEXT sólo puede especificarse para la instrucción READ de un archivo con modalidad de acceso SEQUENTIAL o DYNAMIC. Si se especifica NEXT, la expresión KEY de la cláusula SELECT se ignorará. El elemento de datos RELATIVE KEY se actualiza con el número relativo de registro para archivos con acceso secuencial en operaciones READ.

Se borran todos los archivos de base de datos físicos abiertos para OUTPUT. Los archivos de base de datos con organización RELATIVE, y con modalidad de acceso dinámico o aleatorio, también se inicializan con registros borrados. Los retardos prolongados en el proceso OPEN OUTPUT son normales para los archivos relativos extremadamente largos (más de 1.000.000 de registros) a los que se accede en modalidad de acceso dinámico o aleatorio, ya que los archivos se inicializan con registros borrados. El tiempo que se tarda en abrir un archivo con inicialización depende del número de registros del archivo.

Cuando la primera instrucción OPEN para el archivo no es OPEN OUTPUT, los archivos relativos deben borrarse e inicializarse con registros borrados antes de ser utilizados. Vea la sección sobre los mandatos CLRPFM y INZPFM en la publicación *CL Reference* para obtener más información. El parámetro RECORDS del mandato INZPFM debe especificar *DLT. Las alteraciones temporales se aplican cuando las operaciones de borrar y eliminar son procesadas por ILE COBOL/400, pero no cuando son procesadas con mandatos CL.

Los archivos relativos nuevos abiertos para OUTPUT en modalidad de acceso secuencial se tratan de forma diferente. La Tabla 19 en la página 364 resumen las condiciones que los afectan.

Tabla 19. Inicialización de archivos de salida relativos

Especificaciones de acceso a archivos y CL	Condiciones en la apertura	Condiciones en el cierre	Límite de archivo
Secuencial *INZDLT		Se inicializan los registros no grabados.[1]	Todos los aumentos.
Secuencial *INZDLT tamaño *NOMAX		CLOSE satisfactorio.[1] Estado archivo es 0Q.[2]	Hasta el límite de registros grabados.
Secuencial *NOINZDLT			Hasta el límite de registros grabados.
Aleatorio o dinámico	Se inicializan los registros. Se abre el archivo.		Todos los aumentos.
Aleatorio o dinámico tamaño *NOMAX	OPEN falla. Estado archivo es 9Q.[3]		El archivo está vacío.

Nota:

[1] Los retardos prolongados son normales cuando hay un gran número de registros (más de 1.000.000) a inicializar a registros borrados cuando se ejecuta la instrucción CLOSE.

[2] Para ampliar un límite de archivo más allá del número actual de registros, pero respetando el tamaño del archivo, utilice el mandato INZPFM para añadir registros borrados antes de procesar el archivo. Tiene que hacerlo si recibe un estado de archivo de 0Q y aún desea añadir más registros al archivo. Cualquier intento de ampliar un archivo relativo más allá de su tamaño real provoca una violación del límite.

[3] Para recuperarse de un estado de archivo de 9Q, utilice el mandato CHGPF tal como se describe en el texto de mensaje de ejecución asociado.

Para un archivo ILE COBOL/400 con una organización RELATIVE, el mandato CL Reorganizar miembro de archivo físico (RGZPFM) puede:

- Eliminar todos los registros borrados del archivo. Como ILE COBOL/400 inicializa todos los registros de archivos relativos para registros borrados, cualquier registro que no haya sido grabado explícitamente será eliminado del archivo. Los números relativos de registro de todos los registros después del primer registro borrado del archivo cambiarán.
- Cambie los números relativos de registro si el archivo tiene una clave y la secuencia de llegada se cambia para coincidir con una secuencia por clave (con el parámetro KEYFILE).

Además, un mandato CL Cambiar archivo físico (CHGPF) con la opción REUSEDLT puede cambiar el orden de registros recuperados o grabados cuando en el archivo se trabaja de forma secuencial, ya que permite la reutilización de registros borrados.

Proceso de archivos indexados

Un archivo indexado es un archivo cuya vía de acceso por omisión se crea sobre valores clave. Una forma de crear una vía de acceso por clave para un archivo indexado es utilizando DDS.

Un archivo indexado se identifica mediante la cláusula ORGANIZATION IS INDEXED de la instrucción SELECT.

Los campos clave identifican los registros en un archivo indexado. El usuario especifica el campo clave en la cláusula RECORD KEY de la instrucción SELECT. El elemento de datos RECORD KEY debe definirse en una descripción de registro para el archivo indexado. Si existen múltiples descripciones de registro para el archivo, sólo es necesario que una contenga el nombre de dato RECORD KEY. Sin embargo, en las otras descripciones de registro, se accede a las mismas posiciones de la descripción de registro que contiene el elemento de datos RECORD KEY como el valor KEY para cualquier referencia a las otras descripciones de registro para ese archivo.

A un archivo indexado puede accederse secuencialmente, aleatoriamente por clave o dinámicamente.

| Para escribir programas de norma COBOL que accedan a un archivo indexado, debe crear el archivo con ciertas características. La Tabla 20 en la página 366 lista estas características y qué las controla.

Tabla 20. Características de archivos indexados accesibles a programas de norma COBOL

Característica	Control
El archivo debe ser un archivo físico.	Cree el archivo utilizando el mandato CL CRTPF.
El archivo no puede ser un archivo compartido.	Especifique SHARE(*NO) en el mandato CL CRTPF.
El archivo no puede tener registros con valores clave duplicados.	Especifique la palabra clave UNIQUE en las DDS del archivo.
Las claves deben tener una secuencia ascendente.	No especifique la palabra clave DESCEND en las DDS del archivo.
Las claves deben ser continuas en el registro.	Especifique un solo campo clave en las DDS del archivo o especifique campos clave que se sigan de forma inmediata, en orden descendente según la importancia de la clave.
Debe definirse una clave para el archivo.	Defina por lo menos un campo clave en las Especificaciones de descripción de datos (DDS) del archivo utilizando una línea con K en la posición 17.
Los campos clave deben ser alfanuméricos. No pueden ser numéricos.	Especifique A o H en la posición 35 al definir un campo que deba utilizarse como campo clave de DDS.
El valor de la clave utilizada para la secuencia debe incluir todos los 8 bits de cada byte.	Especifique campos clave alfanuméricos.
No puede especificarse una posición inicial para recuperar registros.	No emita el mandato CL OVRDBF con el parámetro POSITION.
No pueden utilizarse las palabras clave de selección/omisión para el archivo.	No incluya ninguna línea con S u O en la posición 17 de las DDS del archivo. No especifique las palabras clave COMP, RANGE, VALUES o ALL.
Los registros del archivo no pueden contener campos NULL.	No especifique la palabra clave ALWNULL en las DDS del archivo.

Las instrucciones OPEN, READ, WRITE, START, REWRITE, DELETE y CLOSE se utilizan para acceder a datos almacenados en un archivo indexado. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción de cada una de estas instrucciones. Al acceder a archivos indexados, la expresión FORMAT es opcional para los archivos DATABASE y no está permitida para los archivos DISK. Si no se especifica la expresión FORMAT, se utiliza el nombre de formato por omisión del archivo. El nombre de formato por omisión del archivo es el primer nombre de formato definido en el archivo. El registro especial DB-FORMAT-NAME puede utilizarse para recuperar el nombre de formato utilizado en la última operación satisfactoria de E/S.

Al leer registros secuencialmente desde un archivo indexado, los registros se devolverán en secuencia de llegada o en secuencia en clave, según cómo se haya descrito el archivo en el programa ILE COBOL/400. Para recuperar los registros en secuencia de llegada, utilice

```
ORGANIZATION IS SEQUENTIAL
ACCESS IS SEQUENTIAL
```

con la instrucción SELECT para el archivo indexado. Para recuperar los registros en secuencia por clave (normalmente en orden ascendente), utilice

```
ORGANIZATION IS INDEXED
ACCESS IS SEQUENTIAL
```

con la instrucción SELECT para el archivo indexado.

Para archivos indexados a los que se accede secuencialmente, se ignora la expresión KEY de la cláusula SELECT, excepto para la instrucción START. Si no se especifica la expresión KEY en la instrucción START, se utilizará la expresión RECORD KEY de la cláusula SELECT y se asumirá KEY IS EQUAL.

Para archivos indexados a los que se accede aleatoriamente o dinámicamente, se utiliza la expresión KEY de la cláusula SELECT, excepto para la instrucción START. Si no se especifica la expresión KEY en la instrucción START, se utilizará la expresión RECORD KEY de la cláusula SELECT y se asumirá KEY IS EQUAL.

NEXT, PRIOR, FIRST o LAST pueden especificarse para la instrucción READ en archivos DATABASE con acceso DYNAMIC. NEXT también puede especificarse para la instrucción READ para los archivos DATABASE con acceso SEQUENTIAL. Si se especifica NEXT, PRIOR, FIRST o LAST, se ignorará la expresión KEY de la cláusula SELECT.

Todos los archivos de base de datos físicos con organización INDEXED abiertos para OUTPUT se borran.

RECORD KEY válidas

Las DDS para el archivo especifican los campos a utilizar como el campo clave. Si el archivo tiene múltiples campos clave, los campos clave deben ser contiguos en cada registro, a menos que se especifique RECORD KEY IS EXTERNALLY-DESCRIBED-KEY.

Cuando las DDS especifican sólo un campo clave para el archivo, RECORD KEY debe ser un sólo campo de la misma longitud que el campo clave definido en las DDS.

Si se especifica una instrucción COPY de Formato 2 para el archivo, en la cláusula RECORD KEY debe especificar uno de los nombres siguientes:

- El nombre utilizado en las DDS para el campo clave, añadiendo -DDS al final si el nombre es una palabra reservada COBOL.
- El nombre de datos definido en una descripción de registro descrito por programa para el archivo, con la misma longitud y la misma ubicación que el campo clave definido en las DDS.
- EXTERNALLY-DESCRIBED-KEY. Esta palabra clave especifica que las claves definidas en las DDS para cada formato de registro deben utilizarse para acceder al archivo. Estas claves pueden ser no contiguas. Pueden definirse en distintas posiciones dentro un formato de registro.

Cuando las DDS especifican múltiples campos de claves contiguas, el nombre de datos RECORD KEY debe ser un único campo con su longitud igual a la suma de las longitudes de múltiples campos clave en las DDS. Si se especifica una instruc-

ción COPY de Formato 2 para el archivo, también tiene que haber una descripción de registro descrita por programa para el archivo que define el nombre de datos RECORD KEY con la longitud y la posición adecuadas del registro.

Los **elementos contiguos** son elementos elementales o de grupo consecutivos en la Data Division contenidos en una única jerarquía de datos.

Referencia a una clave parcial

Una instrucción START permite la utilización de una clave parcial. Es necesaria la expresión KEY IS.

Consulte el apartado “Instrucción START” en la publicación *ILE COBOL/400 Reference* para obtener información sobre las normas para especificar un argumento de búsqueda que haga referencia a una clave parcial.

La Figura 88 en la página 369 muestra un ejemplo de las instrucciones START utilizando un archivo descrito por programa.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/STRTPGMD         AS400SYS   96/07/04 11:14:42         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. STRTPGMD.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-AS400.
    6     000700   OBJECT-COMPUTER. IBM-AS400.
    7     000800 INPUT-OUTPUT SECTION.                                                          96/11/08
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DISK-FILE1
   11     001100     ACCESS IS DYNAMIC RECORD KEY IS FULL-NAME IN FILE-1
   13     001200     ORGANIZATION IS INDEXED.
          001300
   14     001400 DATA DIVISION.
   15     001500 FILE SECTION.
   16     001600 FD  FILE-1.
   17     001700 01  RECORD-DESCRIPTION.
   18     001800     03 FULL-NAME.
   19     001900        05 LAST-AND-FIRST-NAMES.
   20     002000           07 LAST-NAME            PIC X(20).
   21     002100           07 FIRST-NAME           PIC X(20).
   22     002200        05 MIDDLE-NAME             PIC X(20).
   23     002300     03 LAST-FIRST-MIDDLE-INITIAL-NAME REDEFINES FULL-NAME
          002400                                   PIC X(41).
   24     002500     03 REST-OF-RECORD            PIC X(50).
          002600
   25     002700 PROCEDURE DIVISION.
          002800 MAIN-PROGRAM SECTION.
          002900 MAINLINE.
   26     003000     OPEN INPUT FILE-1.
          003100*
          003200* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
          003300* "SMITH"
   27     003400     MOVE "SMITH" TO LAST-NAME.
   28     003500     START FILE-1 KEY IS EQUAL TO LAST-NAME
   29     003600           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR " LAST-NAME
   30     003700                       GO TO ERROR-ROUTINE                                       96/11/08
          003800     END-START.                                                                  96/11/08
          003900*          .
          004000*          .
          004100*          .
          004200*
          004300* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
          004400* "SMITH" AND A FIRST NAME OF "ROBERT"
   31     004500     MOVE "SMITH" TO LAST-NAME.
   32     004600     MOVE "ROBERT" TO FIRST-NAME.
   33     004700     START FILE-1 KEY IS EQUAL TO LAST-AND-FIRST-NAMES
   34     004800           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
          004900                               LAST-AND-FIRST-NAMES
   35     005000                       GO TO ERROR-ROUTINE                                       96/11/08
          005100     END-START.                                                                  96/11/08
          005200*          .
          005300*          .
          005400*          .
          005500*
```

Figura 88 (Parte 1 de 2). Instrucciones START utilizando un archivo descrito por programa

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/STRTPGMD         AS400SYS   96/07/04 11:14:42         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         005600* POSITION THE FILE STARTING WITH RECORDS THAT HAVE A LAST NAME OF
         005700* "SMITH", A FIRST NAME OF "ROBERT", AND A MIDDLE INITIAL OF "M"
         005800
   36    005900     MOVE "SMITH" TO LAST-NAME.
   37    006000     MOVE "ROBERT" TO FIRST-NAME.
   38    006100     MOVE "M" TO MIDDLE-NAME.
   39    006200     START FILE-1 KEY IS EQUAL TO LAST-FIRST-MIDDLE-INITIAL-NAME                        96/11/08
   40    006300           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         006400                               LAST-FIRST-MIDDLE-INITIAL-NAME
   41    006500                       GO TO ERROR-ROUTINE                                            96/11/08
         006600     END-START.                                                                       96/11/08
         006700
         006800
         006900 ERROR-ROUTINE.
   42    007000     STOP RUN.                                                                        96/11/08

                     * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 88 (Parte 2 de 2). Instrucciones START utilizando un archivo descrito por programa

La Figura 89 y la Figura 90 en la página 371 muestran un ejemplo de las instrucciones START utilizando un archivo descrito externamente.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                      UNIQUE
     A          R RDE                       TEXT('DESCRIPCIÓN DE REGISTRO')
     A            FNAME         20          TEXT('NOMBRE DE PILA')
     A            MINAME         1          TEXT('INICIAL DE SEGUNDO NOMBRE')
     A            MNAME         19          TEXT('PRIMER APELLIDO')
     A            LNAME         20          TEXT('SEGUNDO APELLIDO')
     A            PHONE         10  0       TEXT('NÚMERO DE TELÉFONO')
     A            DATA          40          TEXT('RESTO DE DATOS')
     A          K LNAME
     A          K FNAME
     A          K MINAME
     A          K MNAME
```

Figura 89. Instrucciones START utilizando un archivo descrito externamente -- DDS

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/STRTEXTD         AS400SYS   96/07/04 11:23:45         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID. STRTEXTD.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-AS400.
    6     000700   OBJECT-COMPUTER. IBM-AS400.
    7     000800 INPUT-OUTPUT SECTION.                                                          96/11/08
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DATABASE-NAMES
   11     001100     ACCESS IS DYNAMIC RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
   13     001200     ORGANIZATION IS INDEXED.
          001300
   14     001400 DATA DIVISION.
   15     001500 FILE SECTION.
   16     001600 FD  FILE-1.
   17     001700 01  RECORD-DESCRIPTION.                                                       96/11/08
          001800     COPY DDS-RDE OF NAMES.                                                  96/11/08
         +000001*    FORMATO E-S:RDE       DE ARCHIVO NAMES      DE BIBLIOTECA TESTLIB  RDE
         +000002*                          DESCRIPCIÓN DE REGISTRO                        RDE
         +000003*LAS DEFINICIONES CLAVE PARA FORMATO DE REGISTRO  RDE                       RDE
         +000004*  NÚMERO              NOMBRE              RECUPERACIÓN    SECALT         RDE
         +000005*   0001   LNAME                          ASCENDENTE        NO           RDE
         +000006*   0002   FNAME                          ASCENDENTE        NO           RDE
         +000007*   0003   MINAME                         ASCENDENTE        NO           RDE
         +000008*   0004   MNAME                          ASCENDENTE        NO           RDE
   18    +000009       05  RDE.                                                           RDE
   19    +000010           06 FNAME                PIC X(20).                            RDE
         +000011*                NOMBRE DE PILA                                         RDE
   20    +000012           06 MINAME               PIC X(1).                             RDE
         +000013*                INICIAL SEGUNDO NOMBRE                                 RDE
   21    +000014           06 MNAME                PIC X(19).                            RDE
         +000015*                PRIMER APELLIDO                                        RDE
   22    +000016           06 LNAME                PIC X(20).                            RDE
         +000017*                SEGUNDO APELLIDO                                       RDE
   23    +000018           06 PHONE                PIC S9(10)      COMP-3.               RDE
         +000019*                NÚMERO DE TELÉFONO                                     RDE
   24    +000020           06 DATA-DDS             PIC X(40).                            RDE
         +000021*                RESTO DE DATOS                                         RDE
   25     001900 66  MIDDLE-NAME RENAMES MINAME THRU MNAME.
          002000
   26     002100 PROCEDURE DIVISION.
          002200 MAIN-PROGRAM SECTION.
          002300 MAINLINE.
   27     002400     OPEN INPUT FILE-1.
          002500*
          002600* COLOCAR EL ARCHIVO COMENZANDO CON LOS REGISTRO QUE TENGAN UN SEGUNDO APELLIDO
          002700* DE "SMITH"
   28     002800     MOVE "SMITH" TO LNAME.
   29     002900     START FILE-1 KEY IS EQUAL TO LNAME
   30     003000           INVALID KEY DISPLAY "SIN DATOS EN EL SISTEMA PARA " LNAME
   31     003100                       GO TO ERROR-ROUTINE
          003200     END-START.                                                            96/11/08
          003300*          .
          003400*          .
```

Figura 90 (Parte 1 de 2). Instrucciones START utilizando un archivo descrito externamente

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/STRTEXTD         AS400SYS   96/07/04 11:23:45         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         003500*          .
         003600*
         003700* COLOCAR EL ARCHIVO EMPEZANDO POR LOS REGISTROS QUE TIENEN UN SEGUNDO APELLIDO
         003800* DE "SMITH" Y UN NOMBRE DE PILA DE "ROBERT"
   32    003900     MOVE "SMITH" TO LNAME.
   33    004000     MOVE "ROBERT" TO FNAME.
   34    004100     START FILE-1 KEY IS EQUAL TO LNAME, FNAME
   35    004200           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         004300                               LNAME " " FNAME
   36    004400                       GO TO ERROR-ROUTINE
         004500     END-START.                                                          96/11/08
         004600*          .
         004700*          .
         004800*          .
         004900*
         005000* COLOCAR EL ARCHIVO EMPEZANDO CON LOS REGISTROS QUE TIENEN UN SEGUNDO APELLIDO
         005100* "SMITH", UN NOMBRE DE PILA "ROBERT", Y UNA INICIAL DE SEGUNDO NOMBRE "M"
   37    005200     MOVE "SMITH" TO LNAME.
   38    005300     MOVE "ROBERT" TO FNAME.
   39    005400     MOVE "M" TO MINAME.
   40    005500     START FILE-1 KEY IS EQUAL TO LNAME, FNAME, MINAME
   41    005600           INVALID KEY DISPLAY "NO DATA IN SYSTEM FOR "
         005700                               LNAME SPACE FNAME SPACE MINAME
   42    005800                       GO TO ERROR-ROUTINE
         005900     END-START.                                                          96/11/08
         006000
         006100
         006200 ERROR-ROUTINE.
   43    006300     STOP RUN.                                                         96/11/08

                     * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 90 (Parte 2 de 2). Instrucciones START utilizando un archivo descrito externamente

Proceso de un archivo lógico como archivos indexados

Cuando se procesa un archivo lógico con formatos de registro múltiples, cada uno de ellos con campos clave asociados, como un archivo indexado en ILE COBOL/400, se aplican las siguientes restricciones y consideraciones:

- La expresión FORMAT debe especificarse en todas las instrucciones WRITE para el archivo, a menos que exista un programa selector de formato de registro especificado en el parámetro FMTSLR del mandato Crear archivo lógico (CRTLF), el mandato Cambiar archivo lógico (CHGLF) o el mandato Alterar temporalmente archivo de base de datos (OVRDBF). Para obtener información sobre la utilización de los programas selectores de formato, consulte el manual *DB2/400 Programación de la base de datos*.
- Si la modalidad de acceso es RANDOM o DYNAMIC, y no se especifica la expresión DUPLICATES para el archivo, debe especificarse la expresión FORMAT en todas las instrucciones DELETE y REWRITE.
- Cuando no se especifica la expresión FORMAT, el sistema sólo utiliza la parte del elemento de datos RECORD KEY común a todos los formatos de registro para el archivo como la clave para la instrucción de E/S. Cuando se especifica la expresión FORMAT, el sistema sólo utiliza la parte del elemento de datos RECORD KEY definida para el formato de registro especificado como la clave. Vea el manual *DB2/400 Programación de la base de datos* para obtener más información sobre el proceso de archivos lógicos.

- Cuando se especifica *NONE como el primer campo clave para cualquier formato de un archivo, a los registros sólo puede accederse secuencialmente. Cuando un archivo se lee de forma aleatoria:
  - Si se especifica un nombre de formato, devuelve el primer registro con el formato especificado.
  - Si no se especifica, devuelve el primer registro del archivo.

  En ambos casos, se ignora el valor del elemento de datos RECORD KEY.
- Para un campo clave definido por programa:
  - Los campos clave de cada formato de registro deben ser contiguos.
  - El primer campo clave para cada formato de registro debe empezar en la misma posición relativa dentro de cada registro.
  - La longitud del elemento de datos RECORD KEY debe ser igual a la longitud de la clave más larga para cualquier formato del archivo.
- Para una EXTERNALLY-DESCRIBED-KEY:
  - Los campos clave de cada formato pueden no ser contiguos.
  - Los campos clave pueden definirse en distintas posiciones dentro de un formato de registro.

La Figura 91 y la Figura 92 en la página 374 muestran ejemplos de cómo utilizar DDS para describir la vía de acceso para los archivos indexados.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R FORMATA                   PFILE(ORDDTLP)
     A                                      TEXT('VÍA DE ACCESO PARA ARCHIVO INDEXADO')
     A            FLDA          14
     A            ORDERN         5S 0
     A            FLDB         101
     A          K ORDERN
```

Figura 91. Utilización de las Especificaciones de descripción de datos para definir la vía de acceso para un archivo indexado

Las especificaciones de descripción de datos pueden utilizarse para crear la vía de acceso para un archivo indexado descrito por programa.

En las DDS mostradas en la Figura 91, para el formato de registro FORMATA para el archivo lógico ORDDTLL, el campo ORDERN, de cinco dígitos de longitud, se define como un campo clave. La definición de ORDERN como el campo clave establece la vía de acceso por clave para este archivo. Otros dos campos, FLDA y FLDB, describen las restantes posiciones de este registro como campos de caracteres.

El archivo de entrada descrito por programa ORDDTLL se describe en la sección FILE-CONTROL de la cláusula SELECT como un archivo indexado.

Las descripciones ILE COBOL/400 de cada campo de la entrada FD deben coincidir con la descripción correspondiente del archivo DDS. El elemento de datos RECORD KEY debe definirse como un entero numérico de cinco dígitos en la posición 15 del registro.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R FORMATA                   PFILE(ORDDTLP)
     A                                      TEXT('VÍA DE ACCESO PARA ARCHIVO INDEXADO')
     A            FLDA          14
     A            ORDERN         5S 0
     A            ITEM           5
     A            FLDB          96
     A          K ORDERN
     A          K ITEM
```

Figura 92. Especificaciones de descripción de datos para definir la vía a acceso (una clave compuesta) de un archivo indexado

En este ejemplo, las DDS mostradas en la Figura 92 definen dos campos clave para el formato de registro FORMATA en el archivo lógico ORDDTLL. Para utilizar los dos campos como una clave compuesta para un archivo indexado descrito por programa, los campos clave deben ser contiguos en el registro.

La descripción ILE COBOL/400 de cada campo debe coincidir con la descripción correspondiente del archivo DDS. Debe definirse un elemento de 10 caracteres empezando en la posición 15 del registro en la cláusula RECORD KEY de la entrada de control de archivos. Las descripciones ILE COBOL/400 de los campos de DDS ORDERN y ITEM serían subordinados del elemento de 10 caracteres definido en la cláusula RECORD KEY.

Proceso de archivos con secuencias por clave descendentes

Los archivos creados con una secuencia por clave descendente (en DDS) hacen que las expresiones NEXT, PRIOR, FIRST y LAST de la instrucción READ funcionen de manera exactamente opuesta a como lo harían en un archivo con una secuencia por clave ascendente. Puede especificar una secuencia por clave descendente en las DDS con la palabra clave DESCEND en las posiciones 45 a 80 al lado de un campo clave. En la **secuencia por clave descendente**, los datos se organizan en orden desde el valor superior al valor inferior del campo clave.

Por ejemplo, READ FIRST recupera el registro con el valor clave más alto y READ LAST recupera el registro con el valor clave más bajo. READ NEXT recupera el registro con el próximo valor clave inferior. Los archivos con una secuencia por clave descendente también hacen que los calificadores START funcionen de manera opuesta. Por ejemplo, START GREATER THAN posiciona el puntero de registro actual en un registro con una clave inferior a la clave actual.

Proceso de archivos con registros de longitud variable

Los registros de longitud variable sólo se soportan para archivos de base de datos asociados con tipos de dispositivos DISK.

Descripción de archivos DISK con registros de longitud variable

Especifique la cláusula RECORD de Formato 2 con la entrada FD del archivo para definir las longitudes de registro máximas o mínimas para el archivo.

Una entrada de descripción de archivos simple en la DATA DIVISION que describe un archivo secuencial con registros de longitud variable tiene la apariencia siguiente:

```
FILE SECTION.
FD  nombre-archivo-secuencial
    RECORD IS VARYING IN SIZE
              FROM entero-6 TO entero-7
              DEPENDING ON nombre-datos-1.
01  registro-tamaño-mínimo.
    05  elemento-tamaño-mínimo     PIC X(entero-6).
01  registro-tamaño-máximo.
    05  elemento-tamaño-máximo    PIC X(entero-7).
              .
              .
              .

WORKING-STORAGE SECTION.
77  nombre-datos-1                PIC 9(5).
              .
              .
              .
```

El tamaño de registro mínimo de cualquier registro del archivo lo define el *entero-6*. El tamaño de registro máximo de cualquier registro del archivo lo define el *entero-7*. No cree descripciones de registro para el archivo que contenga una longitud de registro inferior a la especificada en *entero-6* ni superior a la especificada en *entero-7*. Si cualquier descripción de registro rompe esta norma, el compilador ILE COBOL/400 emitirá un mensaje de error de tiempo de compilación. El compilador ILE COBOL/400 también emitirá un mensaje de error de tiempo de compilación cuando ninguna de las descripciones de registro contenga una longitud de registro tan larga como *entero-7*.

Para **archivos indexados** que contienen registros de longitud variable, la primera clave de registro debe estar dentro de las primeras `'n'` posiciones de caracteres del registro, donde `'n'` es el tamaño de registro mínimo especificado para el archivo.Al procesar la entrada FD, el compilador ILE COBOL/400 comprobará que todos los RECORD KEY estén dentro de la parte fija del registro.Si alguna clave viola esta norma, se emitirá un mensaje de error.

Apertura de archivos DISK con registros de longitud variable

Las condiciones siguientes deben cumplirse para que la instrucción OPEN se procese correctamente en un archivo de base de datos con registros de longitud variable:

- los formatos abiertos en el archivo deben tener un campo de longitud variable al final del formato
- la suma de los campos de longitud fija en todos los formatos abiertos debe ser la misma
- la longitud de registro mínima debe ser superior o igual a la suma de los campos de longitud fija para todos los formatos e inferior o igual a la longitud de registro máxima para el archivo
- si el archivo se abre para procesarlo en secuencia por clave, la clave no debe contener ningún campo de longitud variable

Si no se cumple alguna de las condiciones anteriores, se generará un mensaje de error, dará como resultado el estado de archivo 39 y fallará la operación de apertura.

Si se intenta una operación de apertura en un archivo de base de datos con SHARE(*YES) ya abierto, pero con una longitud de registro distinta a la operación de apertura actual, se generará un mensaje de error y dará como resultado el estado de archivo 90.

Lectura y grabación de archivos DISK con registros de longitud variable

Cuando de realiza una instrucción READ, WRITE o REWRITE en un registro de longitud variable, el contenido de *nombre-datos-1* define el tamaño de este registro.

Consulte el formato 2 de la cláusula RECORD en la publicación *ILE COBOL/400 Reference* para obtener más información sobre cómo se manejan los registros de longitud variable.

Utilice la instrucción READ para leer un registro de longitud variable de un archivo de base de datos. Si la operación READ es satisfactoria entonces el *nombre-datos-1*, si se ha especificado, tendrá el número de posiciones de caracteres del registro leído en ese momento. Si la operación READ no es satisfactoria, entonces el *nombre-datos- 1* tendrá el valor que tenía antes de que se intentara la operación READ.

Al especificar la expresión INTO en la instrucción READ, el número de posiciones de caracteres del registro actual que participa como elemento de envío en la instrucción MOVE implícita lo determina

- el contenido de *nombre-datos-1* si *nombre-datos-1* se especifica o
- el número de posiciones de caracteres del registro leído en ese momento si no se especifica el *nombre-datos-1*.

Cuando se realiza la instrucción READ, si el número de posiciones de caracteres del registro leído es inferior a la longitud de registro mínima especificada por las entradas de descripción de registros para el archivo, la parte del área de registro situada a la derecha del último carácter válido leído se rellena con espacios en blanco. Si el número de posiciones de caracteres del registro leído es superior a la longitud de registro máxima especificada por las entradas de descripción de registro para el archivo, el registro se truncará a la derecha del tamaño de registro máximo especificado en las entradas de descripción de registro. Se devuelve un estado de archivo de 04 cuando se lee un registro, la longitud del cual no se encuentra dentro del rango de longitudes de registro mínima o máxima definidas en la cláusula RECORD de la entrada de descripción de archivo para el archivo.

Utilice la instrucción WRITE o REWRITE para grabar un registro de longitud variable en un archivo de base de datos. Especifique la longitud del registro a grabar en el *nombre-datos-1*. Si no especifica el *nombre-datos-1*, la longitud del registro a grabar se determina de la siguiente forma:

- si el registro contiene un elemento OCCURS...DEPENDING ON, mediante la suma de la parte fija y la parte de la tabla descrita por el número de apariciones en el momento en que se ejecuta la instrucción WRITE
- si el registro no contiene un elemento OCCURS...DEPENDING ON, por el número de posiciones de caracteres en la definición del registro.

Ejemplos de proceso de archivos DISK y DATABASE

Los siguientes programas de ejemplo muestran las técnicas fundamentales de programación asociadas con cada tipo de organización de archivo AS/400. Estos ejemplos están pensados para utilizarlos sólo como guía, y para mostrar las instrucciones de entrada/salida necesarias para ciertos métodos de acceso. Otras características ILE COBOL/400 (por ejemplo, la utilización de la instrucción PERFORM) se utilizan sólo de paso. Los programas muestran:

- Creación de archivos secuenciales
- Actualización y ampliación de archivos secuenciales
- Creación de archivos relativos
- Actualización de archivos relativos
- Recuperación de archivos relativos
- Creación de archivos indexados
- Actualización de archivos indexados

Creación de archivos secuenciales

Este programa crea un archivo secuencial de registros de salarios de empleados. Los registros de entrada se organizan en orden ascendente de número de empleado. El archivo de salida tiene el orden idéntico.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTSEQ          AS400SYS   96/07/04 11:12:49        Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  CRTSEQ.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT INPUT-FILE ASSIGN TO DISK-FILEA
   11     000110         FILE STATUS IS INPUT-FILE-STATUS.
   12     000120     SELECT OUTPUT-FILE ASSIGN TO DISK-FILEB
   14     000130         FILE STATUS IS OUTPUT-FILE-STATUS.
   15     000140 DATA DIVISION.
   16     000150 FILE SECTION.
   17     000160 FD  INPUT-FILE.
   18     000170 01  INPUT-RECORD.
   19     000180     05  INPUT-EMPLOYEE-NUMBER      PICTURE 9(6).
   20     000190     05  INPUT-EMPLOYEE-NAME        PICTURE X(28).
   21     000200     05  INPUT-EMPLOYEE-CODE        PICTURE 9.
   22     000210     05  INPUT-EMPLOYEE-SALARY      PICTURE 9(6)V99.
   23     000220 FD  OUTPUT-FILE.
   24     000230 01  OUTPUT-RECORD.
   25     000240     05  OUTPUT-EMPLOYEE-NUMBER     PICTURE 9(6).
   26     000250     05  OUTPUT-EMPLOYEE-NAME       PICTURE X(28).
   27     000260     05  OUTPUT-EMPLOYEE-CODE       PICTURE 9.
   28     000270     05  OUTPUT-EMPLOYEE-SALARY     PICTURE 9(6)V99.
          000280
   29     000290 WORKING-STORAGE SECTION.
   30     000300 77  INPUT-FILE-STATUS              PICTURE XX.
   31     000310 77  OUTPUT-FILE-STATUS             PICTURE XX.
   32     000320 77  OP-NAME                        PICTURE X(7).
   33     000330 01  INPUT-END                      PICTURE X VALUE SPACE.
   34     000340     88  THE-END-OF-INPUT           VALUE "E".
          000350
   35     000360 PROCEDURE DIVISION.
   36     000370 DECLARATIVES.
          000380 INPUT-ERROR SECTION.
          000390      USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          000400 INPUT-ERROR-PARA.
   37     000410      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR INPUT-FILE".
   38     000420      DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   39     000430      DISPLAY "PROCESSING ENDED".
   40     000440      STOP RUN.
          000450
          000460 OUTPUT-ERROR SECTION.
          000470      USE AFTER STANDARD ERROR PROCEDURE ON OUTPUT-FILE.
          000480 OUTPUT-ERROR-PARA.
   41     000490      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR OUTPUT-FILE".
   42     000500      DISPLAY "FILE STATUS IS ", OUTPUT-FILE-STATUS.
   43     000510      DISPLAY "PROCESSING ENDED".
   44     000520      STOP RUN.
          000530 END DECLARATIVES.
          000540
          000550 MAIN-PROGRAM SECTION.
```

Figura 93 (Parte 1 de 2). Ejemplo de un archivo secuencial de registros de salarios de empleados

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTSEQ           AS400SYS   96/07/04 11:12:49         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         000560 MAINLINE.
   45    000570     MOVE "OPEN" TO OP-NAME.
   46    000580     OPEN INPUT INPUT-FILE
         000590          OUTPUT OUTPUT-FILE.
         000600
   47    000610     MOVE "READ" TO OP-NAME.
   48    000620     READ INPUT-FILE INTO OUTPUT-RECORD
   49    000630         AT END SET THE-END-OF-INPUT TO TRUE
         000640     END-READ.
         000650
   50    000660     PERFORM UNTIL THE-END-OF-INPUT
   51    000670         MOVE "WRITE" TO OP-NAME
   52    000680         WRITE OUTPUT-RECORD
   53    000690         MOVE "READ" TO OP-NAME
   54    000700         READ INPUT-FILE INTO OUTPUT-RECORD
   55    000710             AT END SET THE-END-OF-INPUT TO TRUE
         000720         END-READ
         000730     END-PERFORM.
         000740
   56    000750     MOVE "CLOSE" TO OP-NAME.
   57    000760     CLOSE INPUT-FILE
         000770           OUTPUT-FILE.
   58    000780     STOP RUN.

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 93 (Parte 2 de 2). Ejemplo de un archivo secuencial de registros de salarios de empleados

Actualización y ampliación de archivos secuenciales

Este programa actualiza y amplía el archivo creado por el programa `CRTSEQ`. Se lee `INPUT-FILE` y `MASTER-FILE`. Cuando se encuentra una coincidencia entre `INPUT-EMPLOYEE-NUMBER` y `MST-EMPLOYEE-NUMBER`, el registro de entrada sustituye al registro original. Después de procesar `MASTER-FILE`, se añaden nuevos registros de empleados al final del archivo.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTSEQ          AS400SYS   96/07/04 11:26:35         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  UPDTSEQ.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT INPUT-FILE ASSIGN TO DISK-FILES
   11     000110         FILE STATUS IS INPUT-FILE-STATUS.
   12     000120     SELECT MASTER-FILE ASSIGN TO DISK-MSTFILEB
   14     000130         FILE STATUS IS MASTER-FILE-STATUS.
          000140
   15     000150 DATA DIVISION.
   16     000160 FILE SECTION.
   17     000170 FD  INPUT-FILE.
   18     000180 01  INPUT-RECORD.
   19     000190     05  INPUT-EMPLOYEE-NUMBER      PICTURE 9(6).
   20     000200     05  INPUT-EMPLOYEE-NAME        PICTURE X(28).
   21     000210     05  INPUT-EMPLOYEE-CODE        PICTURE 9.
   22     000220     05  INPUT-EMPLOYEE-SALARY      PICTURE 9(6)V99.
   23     000230 FD  MASTER-FILE.
   24     000240 01  MASTER-RECORD.
   25     000250     05 MST-EMPLOYEE-NUMBER         PICTURE 9(6).
   26     000260     05 MST-EMPLOYEE-NAME           PICTURE X(28).
   27     000270     05 MST-EMPLOYEE-CODE           PICTURE 9.
   28     000280     05 MST-EMPLOYEE-SALARY         PICTURE 9(6)V99.
   29     000290 WORKING-STORAGE SECTION.
   30     000300 77  INPUT-FILE-STATUS             PICTURE XX.
   31     000310 77  MASTER-FILE-STATUS            PICTURE XX.
   32     000320 77  OP-NAME                       PICTURE X(12).
   33     000330 01  INPUT-END                     PICTURE X VALUE SPACE.
   34     000340     88  THE-END-OF-INPUT          VALUE "E".
   35     000350 01  MASTER-END                    PICTURE X VALUE SPACE.
   36     000360     88 THE-END-OF-MASTER          VALUE "E".
   37     000370 PROCEDURE DIVISION.
   38     000380 DECLARATIVES.
          000390 INPUT-ERROR SECTION.
          000400      USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          000410 INPUT-ERROR-PARA.
   39     000420      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR INPUT-FILE".
   40     000430      DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   41     000440      DISPLAY "PROCESSING ENDED".
   42     000450      STOP RUN.
          000460
          000470 I-O-ERROR SECTION.
          000480      USE AFTER STANDARD ERROR PROCEDURE ON MASTER-FILE.
          000490 I-O-ERROR-PARA.
   43     000500      DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, "FOR MASTER-FILE".
   44     000510      DISPLAY "FILE STATUS IS ", MASTER-FILE-STATUS.
   45     000520      DISPLAY "PROCESSING ENDED".
   46     000530      STOP RUN.
          000540 END DECLARATIVES.
          000550
```

Figura 94 (Parte 1 de 2). Ejemplo de un programa de actualización de archivos secuenciales

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTSEQ          AS400SYS   96/07/04 11:26:35         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         000560 MAIN-PROGRAM SECTION.
         000570 MAINLINE.
   47    000580        MOVE "OPEN" TO OP-NAME.
   48    000590        OPEN INPUT INPUT-FILE
         000600             I-O   MASTER-FILE.
         000610
   49    000620       PERFORM READ-INPUT-FILE.
   50    000630       PERFORM READ-MASTER-FILE.
   51    000640       PERFORM PROCESS-FILES UNTIL THE-END-OF-INPUT.
         000650
   52    000660       MOVE "CLOSE" TO OP-NAME.
   53    000670       CLOSE MASTER-FILE
         000680             INPUT-FILE.
   54    000690       STOP RUN.
         000700
         000710 READ-INPUT-FILE.
   55    000720       MOVE "READ" TO OP-NAME.
   56    000730       READ INPUT-FILE
   57    000740           AT END SET THE-END-OF-INPUT TO TRUE
         000750       END-READ.
         000760
         000770 READ-MASTER-FILE.
   58    000780       MOVE "READ" TO OP-NAME.
   59    000790       READ MASTER-FILE
         000800           AT END
   60    000810              SET THE-END-OF-MASTER TO TRUE
   61    000820              MOVE "AT END CLOSE" TO OP-NAME
   62    000830              CLOSE MASTER-FILE
   63    000840              MOVE "OPEN EXTEND" TO OP-NAME
   64    000850              OPEN EXTEND MASTER-FILE
         000860       END-READ.
         000870
         000880 PROCESS-FILES.
   65    000890     IF THE-END-OF-MASTER THEN
   66    000900         WRITE MASTER-RECORD FROM INPUT-RECORD
   67    000910         PERFORM READ-INPUT-FILE
         000920     ELSE
   68    000930         IF MST-EMPLOYEE-NUMBER
         000940            LESS THAN INPUT-EMPLOYEE-NUMBER THEN
   69    000950              PERFORM READ-MASTER-FILE
         000960        ELSE
   70    000970           IF MST-EMPLOYEE-NUMBER EQUAL INPUT-EMPLOYEE-NUMBER THEN
   71    000980              MOVE "REWRITE" TO OP-NAME
   72    000990              REWRITE MASTER-RECORD FROM INPUT-RECORD
   73    001000              PERFORM READ-INPUT-FILE
   74    001010              PERFORM READ-MASTER-FILE
         001020           ELSE
   75    001030               DISPLAY "ERROR RECORD -> ", INPUT-EMPLOYEE-NUMBER
   76    001040               PERFORM READ-INPUT-FILE
         001050           END-IF
         001060        END-IF
         001070     END-IF.

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 94 (Parte 2 de 2). Ejemplo de un programa de actualización de archivos secuenciales

Creación de archivos relativos

Este programa crea un archivo relativo de registros de ventas de resumen utilizando el acceso secuencial. Cada registro contiene un resumen de cinco años de ventas por unidad y por dólares para una semana del año; existen 52 registros dentro del archivo, representando cada uno a una semana.

Cada registro de entrada representa las ventas de resumen de una semana de un año. Los registros para la primera semana de los últimos cinco años (en orden ascendente) son los cinco primeros registros de entrada. Los registros para la

segunda semana de los últimos cinco años son los siguientes cinco registros de entrada y así sucesivamente. Así, cinco registros de entrada rellenan un registro de salida.

La `RELATIVE KEY` para `RELATIVE-FILE` no se especifica porqué no es necesaria para el acceso secuencial, a menos que se utilice la instrucción START. (Sin embargo, para actualizar, la clave es `INPUT-WEEK`.)

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTREL           AS400SYS   96/07/04 11:14:29        Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.   CRTREL.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
   11     000110         ORGANIZATION IS RELATIVE
   12     000120         ACCESS IS SEQUENTIAL
   13     000130         FILE STATUS RELATIVE-FILE-STATUS.
   14     000140     SELECT INPUT-FILE ASSIGN TO DISK-FILEC
   16     000150         ORGANIZATION IS SEQUENTIAL
   17     000160         ACCESS IS SEQUENTIAL
   18     000170         FILE STATUS INPUT-FILE-STATUS.
          000180
   19     000190 DATA DIVISION.
   20     000200 FILE SECTION.
   21     000210 FD  RELATIVE-FILE.
   22     000220 01  RELATIVE-RECORD-01.
   23     000230     05  RELATIVE-RECORD OCCURS 5 TIMES INDEXED BY REL-INDEX.
   24     000240         10  RELATIVE-YEAR          PICTURE 99.
   25     000250         10  RELATIVE-WEEK          PICTURE 99.
   26     000260         10  RELATIVE-UNIT-SALES    PICTURE S9(6).
   27     000270         10  RELATIVE-DOLLAR-SALES  PICTURE S9(9)V99.
   28     000280 FD  INPUT-FILE.
   29     000290 01  INPUT-RECORD.
   30     000300     05  INPUT-YEAR                PICTURE 99.
   31     000310     05  INPUT-WEEK                PICTURE 99.
   32     000320     05  INPUT-UNIT-SALES          PICTURE S9(6).
   33     000330     05  INPUT-DOLLAR-SALES        PICTURE S9(9)V99.
          000340
   34     000350 WORKING-STORAGE SECTION.
   35     000360 77  RELATIVE-FILE-STATUS          PICTURE XX.
   36     000370 77  INPUT-FILE-STATUS             PICTURE XX.
   37     000380 77  OP-NAME                       PICTURE X(5).
   38     000390 01  WORK-RECORD.
   39     000400     05  WORK-YEAR                 PICTURE 99 VALUE 00.
   40     000410     05  WORK-WEEK                 PICTURE 99.
   41     000420     05  WORK-UNIT-SALES           PICTURE S9(6).
   42     000430     05  WORK-DOLLAR-SALES         PICTURE S9(9)V99.
   43     000440 01  INPUT-END                     PICTURE X VALUE SPACE.
   44     000450     88  THE-END-OF-INPUT          VALUE "E".
          000460
   45     000470 PROCEDURE DIVISION.
   46     000480 DECLARATIVES.
          000490 INPUT-ERROR SECTION.
          000500       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          000510 INPUT-ERROR-PARA.
   47     000520     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   48     000530     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   49     000540     DISPLAY "PROCESSING ENDED"
   50     000550     STOP RUN.
```

Figura 95 (Parte 1 de 2). Ejemplo de un programa de archivos relativos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTREL           AS400SYS   96/07/04 11:14:29         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         000560
         000570 OUTPUT-ERROR SECTION.
         000580       USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
         000590 OUTPUT-ERROR-PARA.
   51    000600     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   52    000610     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   53    000620     DISPLAY "PROCESSING ENDED"
   54    000630     STOP RUN.
         000640 END DECLARATIVES.
         000650
         000660 MAIN-PROGRAM SECTION.
         000670 MAINLINE.
   55    000680     MOVE "OPEN" TO OP-NAME.
   56    000690     OPEN INPUT INPUT-FILE
         000700          OUTPUT RELATIVE-FILE.
         000710
   57    000720     SET REL-INDEX TO 1.
   58    000730     MOVE "READ" TO OP-NAME.
   59    000740     READ INPUT-FILE
   60    000750         AT END SET THE-END-OF-INPUT TO TRUE
         000760     END-READ.
         000770
   61    000780     PERFORM UNTIL THE-END-OF-INPUT
   62    000790         MOVE INPUT-RECORD TO RELATIVE-RECORD (REL-INDEX)
   63    000800         IF REL-INDEX NOT = 5
   64    000810             SET REL-INDEX UP BY 1
         000820         ELSE
   65    000830             SET REL-INDEX TO 1
   66    000840             MOVE "WRITE" TO OP-NAME
   67    000850             WRITE RELATIVE-RECORD-01
         000860         END-IF
         000870
   68    000880         MOVE "READ" TO OP-NAME
   69    000890         READ INPUT-FILE
   70    000900             AT END SET THE-END-OF-INPUT TO TRUE
         000910         END-READ
         000920     END-PERFORM.
         000930
   71    000940     CLOSE RELATIVE-FILE
         000950           INPUT-FILE.
   72    000960     STOP RUN.
         000970

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 95 (Parte 2 de 2). Ejemplo de un programa de archivos relativos

Actualización de archivos relativos

Este programa utiliza acceso secuencial para actualizar el archivo de registros de ventas de resumen creado en el programa `CRTREL`. El programa de actualización añade un registro por el nuevo año y elimina los registros de los años más antiguos de `RELATIVE-FILE`.

El registro de entrada representa el registro de ventas de resumen para una semana del año anterior. La RELATIVE KEY para el `RELATIVE-FILE` se encuentra en el registro de entrada como `INPUT-WEEK`.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTREL          AS400SYS   96/07/04 11:20:19         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  UPDTREL.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
   11     000110         ORGANIZATION IS RELATIVE
   12     000120         ACCESS IS SEQUENTIAL
   13     000130         RELATIVE KEY INPUT-WEEK
   14     000140         FILE STATUS RELATIVE-FILE-STATUS.
   15     000150     SELECT INPUT-FILE ASSIGN TO DISK-FILES2
   17     000160         ORGANIZATION IS SEQUENTIAL
   18     000170         ACCESS IS SEQUENTIAL
   19     000180         FILE STATUS INPUT-FILE-STATUS.
          000190
   20     000200 DATA DIVISION.
   21     000210 FILE SECTION.
   22     000220 FD  RELATIVE-FILE.
   23     000230 01  RELATIVE-RECORD               PICTURE X(105).
   24     000240 FD  INPUT-FILE.
   25     000250 01  INPUT-RECORD.
   26     000260     05  INPUT-YEAR                PICTURE 99.
   27     000270     05  INPUT-WEEK                PICTURE 99.
   28     000280     05  INPUT-UNIT-SALES          PICTURE S9(6).
   29     000290     05  INPUT-DOLLAR-SALES        PICTURE S9(9)V99.
          000300
   30     000310 WORKING-STORAGE SECTION.
   31     000320 77  RELATIVE-FILE-STATUS          PICTURE XX.
   32     000330 77  INPUT-FILE-STATUS             PICTURE XX.
   33     000340 77  OP-NAME                       PICTURE X(7).
   34     000350 01  WORK-RECORD.
   35     000360     05  FILLER                    PICTURE X(21).
   36     000370     05  CURRENT-WORK-YEARS        PICTURE X(84).
   37     000380     05  NEW-WORK-YEAR.
   38     000390     10  WORK-YEAR                 PICTURE 99.
   39     000400     10  WORK-WEEK                 PICTURE 99.
   40     000410     10  WORK-UNIT-SALES           PICTURE S9(6).
   41     000420     10  WORK-DOLLAR-SALES         PICTURE S9(9)V99.
   42     000430 66  WORK-OUT-RECORD RENAMES
          000440     CURRENT-WORK-YEARS THROUGH NEW-WORK-YEAR.
   43     000450 01  INPUT-END                     PICTURE X VALUE SPACE.
   44     000460     88  THE-END-OF-INPUT          VALUE "E".
          000470
   45     000480 PROCEDURE DIVISION.
   46     000490 DECLARATIVES.
          000500 INPUT-ERROR SECTION.
          000510       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          000520 INPUT-ERROR-PARA.
   47     000530     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   48     000540     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   49     000550     DISPLAY "PROCESSING ENDED"
```

Figura 96 (Parte 1 de 2). Ejemplo de un programa de actualización de archivos relativos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTREL          AS400SYS   96/07/04 11:20:19         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   50     000560     STOP RUN.
          000570
          000580 I-O-ERROR SECTION.
          000590       USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
          000600 I-O-ERROR-PARA.
   51     000610     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   52     000620     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   53     000630     DISPLAY "PROCESSING ENDED"
   54     000640     STOP RUN.
          000650 END DECLARATIVES.
          000660
          000670 MAIN-PROGRAM SECTION.
          000680 MAINLINE.
   55     000690     MOVE "OPEN" TO OP-NAME.
   56     000700     OPEN INPUT INPUT-FILE
          000710          I-O RELATIVE-FILE.
          000720
   57     000730     MOVE "READ" TO OP-NAME.
   58     000740     READ RELATIVE-FILE INTO WORK-RECORD
   59     000750         AT END SET THE-END-OF-INPUT TO TRUE
          000760     END-READ.
   60     000770     READ INPUT-FILE INTO NEW-WORK-YEAR
   61     000780         AT END SET THE-END-OF-INPUT TO TRUE
          000790     END-READ.
          000800
   62     000810     PERFORM UNTIL THE-END-OF-INPUT
   63     000820        MOVE "REWRITE" TO OP-NAME
   64     000830        REWRITE RELATIVE-RECORD FROM WORK-OUT-RECORD
          000840
   65     000850        MOVE "READ" TO OP-NAME
   66     000860        READ RELATIVE-FILE INTO WORK-RECORD
   67     000870            AT END SET THE-END-OF-INPUT TO TRUE
          000880        END-READ
   68     000890        READ INPUT-FILE INTO NEW-WORK-YEAR
   69     000900            AT END SET THE-END-OF-INPUT TO TRUE
          000910         END-READ
          000920     END-PERFORM.
          000930
   70     000940     MOVE "CLOSE" TO OP-NAME.
   71     000950     CLOSE INPUT-FILE
          000960           RELATIVE-FILE.
   72     000970     STOP RUN.
          000980

                         * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 96 (Parte 2 de 2). Ejemplo de un programa de actualización de archivos relativos

Recuperación de archivos relativos

Este programa, utilizando acceso dinámico, recupera el archivo de resumen creado por el programa CRTREL.

Los registros de INPUT-FILE contienen un campo necesario (INPUT-WEEK), que es la RELATIVE KEY para RELATIVE-FILE y un campo opcional (END-WEEK). Un registro de entrada que contiene datos en INPUT-WEEK y espacios en END-WEEK solicita una salida impresa para el RELATIVE-RECORD específico; el registro se recupera mediante el acceso aleatorio. Un registro de entrada que contiene datos en INPUT-WEEK y END-WEEK solicita una salida impresa de todos los registros RELATIVE-FILE dentro del rango RELATIVE KEY de INPUT-WEEK a END-WEEK inclusive. Estos registros se recuperan mediante el acceso secuencial.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RTRVREL          AS400SYS   96/07/04 11:41:56         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  RTRVREL.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT RELATIVE-FILE ASSIGN TO DISK-FILED
   11     000110         ORGANIZATION IS RELATIVE
   12     000120         ACCESS IS DYNAMIC
   13     000130         RELATIVE KEY INPUT-WEEK
   14     000140         FILE STATUS IS RELATIVE-FILE-STATUS.
   15     000150     SELECT INPUT-FILE ASSIGN TO DISK-FILEF
   17     000160         FILE STATUS IS INPUT-FILE-STATUS.
   18     000170     SELECT PRINT-FILE ASSIGN TO PRINTER-QSYSPRT
   20     000180         FILE STATUS IS PRINT-FILE-STATUS.
          000190
   21     000200 DATA DIVISION.
   22     000210 FILE SECTION.
   23     000220 FD  RELATIVE-FILE.
   24     000230 01  RELATIVE-RECORD-01.
   25     000240     05  RELATIVE-RECORD OCCURS 5 TIMES INDEXED BY REL-INDEX.
   26     000250         10  RELATIVE-YEAR         PICTURE 99.
   27     000260         10  RELATIVE-WEEK         PICTURE 99.
   28     000270         10  RELATIVE-UNIT-SALES   PICTURE S9(6).
   29     000280         10  RELATIVE-DOLLAR-SALES PICTURE S9(9)V99.
   30     000290 FD  INPUT-FILE.
   31     000300 01  INPUT-RECORD.
   32     000310     05 INPUT-WEEK                 PICTURE 99.
   33     000320     05  END-WEEK                  PICTURE 99.
   34     000330 FD  PRINT-FILE.
   35     000340 01  PRINT-RECORD.
   36     000350     05 PRINT-WEEK                 PICTURE 99.
   37     000360     05 FILLER                     PICTURE X(5).
   38     000370     05 PRINT-YEAR                 PICTURE 99.
   39     000380     05 FILLER                     PICTURE X(5).
   40     000390     05 PRINT-UNIT-SALES           PICTURE ZZZ,ZZ9.
   41     000400     05 FILLER                     PICTURE X(5).
   42     000410     05 PRINT-DOLLAR-SALES         PICTURE $$$$,$$$,$$$.99.
          000420
   43     000430 WORKING-STORAGE SECTION.
   44     000440 77  RELATIVE-FILE-STATUS          PICTURE XX.
   45     000450 77  INPUT-FILE-STATUS             PICTURE XX.
   46     000460 77  PRINT-FILE-STATUS             PICTURE XX.
   47     000470 77  HIGH-WEEK                     PICTURE 99 VALUE 53.
   48     000480 77  OP-NAME                       PICTURE X(9).
   49     000490 01  INPUT-END                     PICTURE X(9).
   50     000500     88  THE-END-OF-INPUT          VALUE "E".
          000510
   51     000520 PROCEDURE DIVISION.
   52     000530 DECLARATIVES.
          000540 RELATIVE-FILE-ERROR SECTION.
          000550          USE AFTER STANDARD ERROR PROCEDURE ON RELATIVE-FILE.
```

Figura 97 (Parte 1 de 3). Ejemplo de un programa de recuperación de archivos relativos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RTRVREL          AS400SYS   96/07/04 11:41:56         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         000560 RELATIVE-ERROR-PARA.
   53    000570     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR RELATIVE-FILE".
   54    000580     DISPLAY "FILE STATUS IS ", RELATIVE-FILE-STATUS.
   55    000590     DISPLAY "PROCESSING ENDED"
   56    000600     STOP RUN.
         000610
         000620 INPUT-FILE-ERROR SECTION.
         000630          USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
         000640 INPUT-ERROR-PARA.
   57    000650     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   58    000660     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   59    000670     DISPLAY "PROCESSING ENDED"
   60    000680     STOP RUN.
         000690
         000700 PRINT-FILE-ERROR SECTION.
         000710          USE AFTER STANDARD ERROR PROCEDURE ON PRINT-FILE.
         000720 PRINT-ERROR-MSG.
   61    000730     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR PRINT-FILE ".
   62    000740     DISPLAY "FILE STATUS IS ", PRINT-FILE-STATUS.
   63    000750     DISPLAY "PROCESSING ENDED"
   64    000760     STOP RUN.
         000770 END DECLARATIVES.
         000780
         000790 MAIN-PROGRAM SECTION.
         000800 MAINLINE.
   65    000810     MOVE "OPEN" TO OP-NAME.
   66    000820     OPEN INPUT INPUT-FILE
         000830                RELATIVE-FILE
         000840          OUTPUT PRINT-FILE.
         000850
   67    000860     MOVE SPACES TO PRINT-RECORD.
   68    000870     PERFORM READ-INPUT-FILE.
   69    000880     PERFORM CONTROL-PROCESS THRU READ-INPUT-FILE
         000890                             UNTIL THE-END-OF-INPUT.
         000900
   70    000910     MOVE "CLOSE" TO OP-NAME.
   71    000920     CLOSE RELATIVE-FILE
         000930           INPUT-FILE
         000940           PRINT-FILE.
   72    000950     STOP RUN.
         000960
         000970 CONTROL-PROCESS.
   73    000980     IF (END-WEEK = SPACES OR END-WEEK = 00)
   74    000990         MOVE "READ" TO OP-NAME
   75    001000         READ RELATIVE-FILE
   76    001010         PERFORM PRINT-SUMMARY VARYING REL-INDEX FROM 1 BY 1
         001020                               UNTIL REL-INDEX > 5
         001030     ELSE
   77    001040         MOVE "READ" TO OP-NAME
   78    001050         READ RELATIVE-FILE
   79    001060         PERFORM READ-REL-SEQ
         001070              UNTIL RELATIVE-WEEK(1) GREATER THAN END-WEEK
         001080         END-IF.
         001090
         001100 READ-INPUT-FILE.
   80    001110     MOVE "READ" TO OP-NAME.
   81    001120     READ INPUT-FILE
```

Figura 97 (Parte 2 de 3). Ejemplo de un programa de recuperación de archivos relativos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/RTRVREL          AS400SYS   96/07/04 11:41:56         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   82     001130         AT END SET THE-END-OF-INPUT TO TRUE
          001140     END-READ.
          001150
          001160 READ-REL-SEQ.
   83     001170     PERFORM PRINT-SUMMARY VARYING REL-INDEX FROM 1 BY 1
          001180                           UNTIL REL-INDEX > 5.
   84     001190     MOVE "READ NEXT" TO OP-NAME.
   85     001200     READ RELATIVE-FILE NEXT RECORD
   86     001210         AT END MOVE HIGH-WEEK TO RELATIVE-WEEK(1)
          001220     END-READ.
          001230
          001240 PRINT-SUMMARY.
   87     001250     MOVE RELATIVE-YEAR (REL-INDEX) TO PRINT-YEAR.
   88     001260     MOVE RELATIVE-WEEK (REL-INDEX) TO PRINT-WEEK.
   89     001270     MOVE RELATIVE-UNIT-SALES (REL-INDEX) TO PRINT-UNIT-SALES.
   90     001280     MOVE RELATIVE-DOLLAR-SALES(REL-INDEX) TO PRINT-DOLLAR-SALES.
   91     001290     MOVE "WRITE" TO OP-NAME.
   92     001300     WRITE PRINT-RECORD AFTER ADVANCING 2 LINES
          001310     END-WRITE.

                         * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 97 (Parte 3 de 3). Ejemplo de un programa de recuperación de archivos relativos

Creación de archivos indexados

Este programa crea un archivo indexado de registros de resumen para depositantes bancarios. La clave de cada registro de archivo indexado es INDEX-KEY (el número de la cuenta del depositante); los registros de entrada se ordenan en secuencia ascendente en esta clave. Se leen los registros del archivo de entrada y se transfieren al área de registro de archivo indexado. Entonces se graba el registro de archivo indexado.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTIND          AS400SYS   96/07/04 11:45:34        Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020 PROGRAM-ID.  CRTIND.
          000030
    3     000040 ENVIRONMENT DIVISION.
    4     000050 CONFIGURATION SECTION.
    5     000060 SOURCE-COMPUTER. IBM-AS400.
    6     000070 OBJECT-COMPUTER. IBM-AS400.
    7     000080 INPUT-OUTPUT SECTION.
    8     000090 FILE-CONTROL.
    9     000100     SELECT INDEXED-FILE ASSIGN TO DISK-INDEXFILE
   11     000110         ORGANIZATION IS INDEXED
   12     000120         ACCESS IS SEQUENTIAL
   13     000130         RECORD KEY IS INDEX-KEY
   14     000140         FILE STATUS IS INDEXED-FILE-STATUS.
   15     000150     SELECT INPUT-FILE ASSIGN TO DISK-FILEG
   17     000160         FILE STATUS IS INPUT-FILE-STATUS.
          000170
   18     000180 DATA DIVISION.
   19     000190 FILE SECTION.
   20     000200 FD  INDEXED-FILE.
   21     000210 01  INDEX-RECORD.
   22     000220     05  INDEX-KEY                   PICTURE X(10).
   23     000230     05  INDEX-FLD1                  PICTURE X(10).
   24     000240     05  INDEX-NAME                  PICTURE X(20).
   25     000250     05  INDEX-BAL                   PICTURE S9(5)V99.
   26     000260 FD  INPUT-FILE.
   27     000270 01  INPUT-RECORD.
   28     000280     05  INPUT-KEY                   PICTURE X(10).
   29     000290     05  INPUT-NAME                  PICTURE X(20).
   30     000300     05  INPUT-BAL                   PICTURE S9(5)V99.
   31     000310 WORKING-STORAGE SECTION.
   32     000320 77  INDEXED-FILE-STATUS             PICTURE XX.
   33     000330 77  INPUT-FILE-STATUS               PICTURE XX.
   34     000340 77  OP-NAME                         PICTURE X(7).
   35     000350 01  INPUT-END                       PICTURE X VALUE SPACES.
   36     000360     88  THE-END-OF-INPUT            VALUE "E".
          000370
   37     000380 PROCEDURE DIVISION.
   38     000390 DECLARATIVES.
          000400 INPUT-ERROR SECTION.
          000410       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          000420 INPUT-ERROR-PARA.
   39     000430     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   40     000440     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   41     000450     DISPLAY "PROCESSING ENDED"
   42     000460     STOP RUN.
          000470
          000480 OUTPUT-ERROR SECTION.
          000490       USE AFTER STANDARD ERROR PROCEDURE ON INDEXED-FILE.
          000500 OUTPUT-ERROR-PARA.
   43     000510     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INDEXED-FILE ".
   44     000520     DISPLAY "FILE STATUS IS ", INDEXED-FILE-STATUS.
   45     000530     DISPLAY "PROCESSING ENDED"
   46     000540     STOP RUN.
          000550 END DECLARATIVES.
```

Figura 98 (Parte 1 de 2). Ejemplo de un programa de archivos indexados

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/CRTIND          AS400SYS   96/07/04 11:45:34         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         000560
         000570 MAIN-PROGRAM SECTION.
         000580 MAINLINE.
   47    000590     MOVE "OPEN" TO OP-NAME.
   48    000600     OPEN INPUT INPUT-FILE
         000610         OUTPUT INDEXED-FILE.
         000620
   49    000630     MOVE "READ" TO OP-NAME.
   50    000640     READ INPUT-FILE
   51    000650          AT END SET THE-END-OF-INPUT TO TRUE
         000660     END-READ.
         000670
   52    000680     PERFORM UNTIL THE-END-OF-INPUT
   53    000690         MOVE INPUT-KEY TO INDEX-KEY
   54    000700         MOVE INPUT-NAME TO INDEX-NAME
   55    000710         MOVE INPUT-BAL TO INDEX-BAL
   56    000720         MOVE SPACES TO INDEX-FLD1
   57    000730         MOVE "WRITE" TO OP-NAME
   58    000740         WRITE INDEX-RECORD
         000750
   59    000760         MOVE "READ" TO OP-NAME
   60    000770         READ INPUT-FILE
   61    000780              AT END SET THE-END-OF-INPUT TO TRUE
         000790         END-READ
         000800     END-PERFORM.
         000810
   62    000820     MOVE "CLOSE" TO OP-NAME.
   63    000830     CLOSE INPUT-FILE
         000840           INDEXED-FILE.
   64    000850     STOP RUN.
         000860

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 98 (Parte 2 de 2). Ejemplo de un programa de archivos indexados

Actualización de archivos indexados

Este programa, utilizando acceso dinámico, actualiza el archivo indexado creado en el programa `CRTIND`.

Los registros de entrada contienen la clave para el registro, el nombre del depositante y el importe de la transacción.

Cuando se lee el registro de entrada, el programa determina si es:

- un registro de transacción (en cuyo caso, se rellenan todos los campos del registro)
- un registro que solicita una recuperación secuencial de una clase genérica específica (en cuyo caso, sólo el campo `INPUT-GEN-FLD` del registro de entrada contiene datos).

El acceso aleatorio se utiliza para actualizar e imprimir los registros de transacción. El acceso secuencial se utiliza para recuperar e imprimir todos los registros de una clase genérica.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTIND          AS400SYS   96/07/04 11:32:12         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.  UPDTIND.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-AS400.
    6     000700   OBJECT-COMPUTER. IBM-AS400.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT INDEXED-FILE ASSIGN TO DISK-INDXFILE
   11     001100         ORGANIZATION IS INDEXED
   12     001200         ACCESS IS DYNAMIC
   13     001300         RECORD KEY IS INDEX-KEY
   14     001400         FILE STATUS IS INDEXED-FILE-STATUS.
   15     001500     SELECT INPUT-FILE ASSIGN TO DISK-FILEH
   17     001600         FILE STATUS IS INPUT-FILE-STATUS.
   18     001700     SELECT PRINT-FILE ASSIGN TO PRINTER-OSYSPRT
   20     001800         FILE STATUS IS PRINT-FILE-STATUS.
          001900
   21     002000 DATA DIVISION.
   22     002100 FILE SECTION.
   23     002200 FD  INDEXED-FILE.
   24     002300 01  INDEX-RECORD.
   25     002400     05  INDEX-KEY.
   26     002500       10    INDEX-GEN-FLD    PICTURE X(5).
   27     002600       10    INDEX-DET-FLD    PICTURE X(5).
   28     002700     05  INDEX-FLD1           PICTURE X(10).
   29     002800     05  INDEX-NAME           PICTURE X(20).
   30     002900     05  INDEX-BAL            PICTURE S9(5)V99.
   31     003000 FD  INPUT-FILE.
   32     003100 01  INPUT-REC.
   33     003200     05  INPUT-KEY.
   34     003300       10  INPUT-GEN-FLD      PICTURE X(5).
   35     003400       10  INPUT-DET-FLD      PICTURE X(5).
   36     003500     05  INPUT-NAME           PICTURE X(20).
   37     003600     05  INPUT-AMT            PICTURE S9(5)V99.
   38     003700 FD  PRINT-FILE                                                                  96/11/08
          003800     LINAGE 12 LINES FOOTING AT 9.
   39     003900 01  PRINT-RECORD-1.
   40     004000     05  PRINT-KEY            PICTURE X(10).
   41     004100     05  FILLER               PICTURE X(5).
   42     004200     05  PRINT-NAME           PICTURE X(20).
   43     004300     05  FILLER               PICTURE X(5).
   44     004400     05  PRINT-BAL            PICTURE $$$,$$9.99-.
   45     004500     05  FILLER               PICTURE X(7).
   46     004600     05  PRINT-AMT            PICTURE $$$,$$9.99-.
   47     004700     05  FILLER               PICTURE X(5).
   48     004800     05  PRINT-NEW-BAL        PICTURE $$$,$$9.99-.
   49     004900 01  PRINT-RECORD-2           PICTURE X(89).
          005000
   50     005100 WORKING-STORAGE SECTION.
   51     005200 77  INDEXED-FILE-STATUS      PICTURE XX.
   52     005300 77  INPUT-FILE-STATUS        PICTURE XX.
   53     005400 77  PRINT-FILE-STATUS        PICTURE XX.
   54     005500 77  OP-NAME                  PICTURE X(9).
```

Figura 99 (Parte 1 de 4). Ejemplo de un programa de actualización de archivos indexados

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTIND          AS400SYS   96/07/04 11:32:12         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   55     005600 77  LINES-TO-FOOT            PICTURE 99.
   56     005700 01  PAGE-HEAD.
   57     005800     05  FILLER               PICTURE X(38) VALUE SPACES.
   58     005900     05  FILLER               PICTURE X(13) VALUE "UPDATE REPORT".
   59     006000     05  FILLER               PICTURE X(38) VALUE SPACES.
   60     006100 01  COLUMN-HEAD.
   61     006200     05  FILLER               PICTURE X(6)  VALUE "KEY ID".
   62     006300     05  FILLER               PICTURE X(9)  VALUE SPACES.
   63     006400     05  FILLER               PICTURE X(4)  VALUE "NAME".
   64     006500     05  FILLER               PICTURE X(21) VALUE SPACES.
   65     006600     05  FILLER               PICTURE X(11) VALUE "CUR BALANCE".
   66     006700     05  FILLER               PICTURE X(6)  VALUE SPACES.
   67     006800     05  FILLER               PICTURE X(13) VALUE "UPDATE AMOUNT".
   68     006900     05  FILLER               PICTURE X(4)  VALUE SPACES.
   69     007000     05  FILLER               PICTURE X(11) VALUE "NEW BALANCE".
   70     007100     05  FILLER               PICTURE X(4)  VALUE SPACES.
   71     007200 01  PAGE-FOOT.
   72     007300     05  FILLER               PICTURE X(81)  VALUE SPACES.
   73     007400     05  FILLER               PICTURE A(6)   VALUE "PAGE  ".
   74     007500     05  PG-NUMBER            PICTURE 99     VALUE 00.
          007600
   75     007700 01  INPUT-END               PICTURE X VALUE SPACE.
   76     007800     88  THE-END-OF-INPUT              VALUE "E".
          007900
   77     008000 PROCEDURE DIVISION.
   78     008100 DECLARATIVES.
          008200 INPUT-ERROR SECTION.
          008300       USE AFTER STANDARD ERROR PROCEDURE ON INPUT-FILE.
          008400 INPUT-ERROR-PARA.
   79     008500     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INPUT-FILE ".
   80     008600     DISPLAY "FILE STATUS IS ", INPUT-FILE-STATUS.
   81     008700     DISPLAY "PROCESSING ENDED"
   82     008800     STOP RUN.
          008900
          009000 I-O-ERROR SECTION.
          009100       USE AFTER STANDARD ERROR PROCEDURE ON INDEXED-FILE.
          009200 I-O-ERROR-PARA.
   83     009300     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR INDEXED-FILE ".
   84     009400     DISPLAY "FILE STATUS IS ", INDEXED-FILE-STATUS.
   85     009500     DISPLAY "PROCESSING ENDED"
   86     009600     STOP RUN.
          009700
          009800 OUTPUT-ERROR SECTION.
          009900       USE AFTER STANDARD ERROR PROCEDURE ON PRINT-FILE.
          010000 OUTPUT-ERROR-PARA.
   87     010100     DISPLAY "UNEXPECTED ERROR ON ", OP-NAME, " FOR PRINT-FILE ".
   88     010200     DISPLAY "FILE STATUS IS ", PRINT-FILE-STATUS.
   89     010300     DISPLAY "PROCESSING ENDED"
   90     010400     STOP RUN.
          010500 END DECLARATIVES.
          010600
          010700 MAIN-PROGRAM SECTION.
          010800 MAINLINE.
   91     010900     MOVE "OPEN" TO OP-NAME.
   92     011000     OPEN INPUT INPUT-FILE
          011100         I-O INDEXED-FILE
          011200         OUTPUT PRINT-FILE.
```

Figura 99 (Parte 2 de 4). Ejemplo de un programa de actualización de archivos indexados

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTIND          AS400SYS   96/07/04 11:32:12         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         011300
   93    011400     PERFORM PAGE-START.
   94    011500     PERFORM READ-INPUT-FILE.
   95    011600     PERFORM PROCESS-DATA THRU READ-INPUT-FILE
         011700                          UNTIL THE-END-OF-INPUT.
   96    011800     PERFORM PAGE-END.
         011900
   97    012000     MOVE "CLOSE" TO OP-NAME.
   98    012100     CLOSE INPUT-FILE
         012200           INDEXED-FILE
         012300           PRINT-FILE.
   99    012400     STOP RUN.
         012500
         012600 PROCESS-DATA.
  100    012700     IF INPUT-DET-FLD EQUAL SPACES
  101    012800         MOVE INPUT-GEN-FLD TO INDEX-GEN-FLD
  102    012900         MOVE "START" TO OP-NAME
  103    013000         START INDEXED-FILE
         013100               KEY IS NOT LESS THAN INDEX-GEN-FLD
         013200         END-START
  104    013300         PERFORM SEQUENTIAL-PROCESS
         013400             UNTIL INPUT-GEN-FLD NOT EQUAL INDEX-GEN-FLD
         013500     ELSE
  105    013600         MOVE INPUT-KEY TO INDEX-KEY
  106    013700         MOVE "READ" TO OP-NAME
  107    013800         READ INDEXED-FILE
  108    013900         IF INPUT-GEN-FLD EQUAL INDEX-GEN-FLD THEN
  109    014000             MOVE INDEX-KEY TO PRINT-KEY
  110    014100             MOVE INDEX-NAME TO PRINT-NAME
  111    014200             MOVE INDEX-BAL TO PRINT-BAL
  112    014300             MOVE INPUT-AMT TO PRINT-AMT
  113    014400             ADD INPUT-AMT TO INDEX-BAL
  114    014500             MOVE INDEX-BAL TO PRINT-NEW-BAL
  115    014600             PERFORM PRINT-DETAIL
  116    014700             MOVE "REWRITE" TO OP-NAME
  117    014800             REWRITE INDEX-RECORD
         014900         END-IF
         015000     END-IF.
         015100
         015200 READ-INPUT-FILE.
  118    015300     MOVE "READ" TO OP-NAME.
  119    015400     READ INPUT-FILE
  120    015500         AT END SET THE-END-OF-INPUT TO TRUE
         015600     END-READ.
         015700
         015800 SEQUENTIAL-PROCESS.
  121    015900     MOVE "READ NEXT" TO OP-NAME.
  122    016000     READ INDEXED-FILE NEXT RECORD
  123    016100         AT END MOVE HIGH-VALUE TO INDEX-GEN-FLD
         016200     END-READ.
  124    016300     IF INPUT-GEN-FLD EQUAL INDEX-GEN-FLD THEN
  125    016400         MOVE INDEX-KEY TO PRINT-KEY
  126    016500         MOVE INDEX-NAME TO PRINT-NAME
  127    016600         MOVE INDEX-BAL TO PRINT-NEW-BAL
  128    016700         PERFORM PRINT-DETAIL
         016800     END-IF.
         016900
```

Figura 99 (Parte 3 de 4). Ejemplo de un programa de actualización de archivos indexados

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/UPDTIND          AS400SYS   96/07/04 11:32:12         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         017000 PRINT-DETAIL.
  129    017100     MOVE "WRITE" TO OP-NAME.
  130    017200     WRITE PRINT-RECORD-1
         017300          AT END-OF-PAGE
  131    017400          PERFORM PAGE-END THROUGH PAGE-START
         017500     END-WRITE.
  132    017600     MOVE SPACES TO PRINT-RECORD-1.
         017700
         017800 PAGE-END.
  133    017900     MOVE "WRITE" TO OP-NAME.
  134    018000     ADD 1 TO PG-NUMBER.
  135    018100     SUBTRACT LINAGE-COUNTER OF PRINT-FILE FROM 12
         018200          GIVING LINES-TO-FOOT.
  136    018300     MOVE SPACES TO PRINT-RECORD-1.
  137    018400     WRITE PRINT-RECORD-1
         018500          AFTER ADVANCING LINES-TO-FOOT
         018600     END-WRITE.
  138    018700     WRITE PRINT-RECORD-2 FROM PAGE-FOOT
         018800          BEFORE ADVANCING PAGE
         018900     END-WRITE.
         019000
         019100 PAGE-START.
  139    019200     WRITE PRINT-RECORD-2 FROM PAGE-HEAD
         019300          AFTER ADVANCING 1 LINE                                                  96/11/08
         019400     END-WRITE.
  140    019500     MOVE SPACES TO PRINT-RECORD-2.
  141    019600     WRITE PRINT-RECORD-2 FROM COLUMN-HEAD
         019700          AFTER ADVANCING 1 LINE
         019800     END-WRITE.
  142    019900     MOVE SPACES TO PRINT-RECORD-2.

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 99 (Parte 4 de 4). Ejemplo de un programa de actualización de archivos indexados

Capítulo 17. Utilización de archivos de transacción

En este capítulo se describen las ampliaciones de lenguaje ILE COBOL/400 que dan soporte a estaciones de trabajo y a las comunicaciones de programa a programa.

La organización de archivos TRANSACTION permite que un programa ILE COBOL/400 se comunique de forma interactiva con:

- Una o más estaciones de trabajo
- Uno o más programas en un sistema remoto
- Uno o más dispositivos en un sistema remoto.

El sistema AS/400 le permite comunicarse con un programa o dispositivo (como por ejemplo los del tipo comunicación asíncrona) en un sistema remoto. Para obtener una explicación detallada de tales dispositivos, consulte la publicación *ICF Programming*.

Normalmente, los archivos TRANSACTION ILE COBOL/400 están descritos externamente. Si están descritos por el programa, sólo se puede realizar un formato de pantalla simple. Consulte la publicación *Gestión de datos* para obtener más información sobre la utilización de archivos de pantalla descritos por programa. Un archivo TRANSACTION ILE COBOL/400 utiliza por lo general un archivo descrito externamente que contiene la información de archivo y una descripción de los campos en los registros. Los registros de este archivo pueden describirse en un programa ILE COBOL/400 utilizando la instrucción COPY de formato 2. Consulte la publicación *ILE COBOL/400 Reference* para obtener más información sobre la instrucción COPY de formato 2.

No envíe datos empaquetados ni binarios (COMP, COMP-3 o COMP-4) a una estación de pantalla como datos de salida. Tales datos pueden contener caracteres de control de estación de pantalla que pueden provocar resultados imprevisibles.

Definición de archivos de transacción con especificaciones de descripción de datos

Para describir un archivo TRANSACTION descrito externamente, puede utilizar las especificaciones de descripción de datos (DDS).

Además de las descripciones de campo (tales como atributos y nombres de campo), las especificaciones de descripción de datos (DDS) de un archivo de dispositivo de pantalla hacen lo siguiente:

- Especifican las entradas de número de línea y número de posición de cada campo y constante para dar formato a la colocación del registro en la pantalla.
- Especifican las funciones de atención, como por ejemplo el subrayado y resaltado de campos, el contraste invertido o un cursor parpadeante.
- Especifican la comprobación de la validez de los datos entrados en la estación de pantalla.
- Controlan las funciones de gestión de pantalla, como por ejemplo cuándo se han de borrar, superponer o retener los campos al visualizar datos nuevos.

- Asocian los indicadores que van del 01 al 99 con teclas de funciones designadas con el tipo `CA` o `CF`. Si una tecla de función está designada como `CF`, al programa se devuelve tanto el registro de datos modificado como el indicador de respuesta. Si una tecla de función está designada como `CA`, al programa se devuelve el indicador de respuesta, pero el registro de datos suele contener valores por omisión para los campos de sólo entrada y valores escritos en el formato para los campos de entrada/salida ocultos. Para obtener más información sobre las teclas de función de tipo `CF` y `CA`, consulte la publicación *DDS Reference*.
- Asignan un código de edición (palabra clave EDTCDE) o palabra de edición (palabra clave EDTWRD) a un campo, para especificar la forma de visualización de los valores del campo.
- Especifican subarchivos.

Los **datos de formato de pantalla** definen o describen una pantalla. Un formato de registro de dispositivo de pantalla contiene tres tipos de campos:

- *Campos de entrada*: Los campos de entrada pasan del dispositivo al programa cuando éste lee un registro. Pueden inicializarse con un valor por omisión; si éste no cambia, pasa al programa. Los campos de entrada no inicializados se visualizan como blancos allí donde el usuario de la estación de trabajo puede entrar datos.
- *Campos de salida*: Los campos de salida pasan del programa al dispositivo cuando el programa escribe un registro en una pantalla. El programa o el formato de registro del archivo de dispositivo puede proporcionar campos de salida.
- *Campos de entrada/salida (bivalentes)*: Un campo de entrada/salida es un campo de salida que puede cambiar para convertirse en un campo de entrada. Los campos de entrada/salida pasan *desde el* programa cuando éste escribe un registro en una pantalla y *al* programa cuando éste lee un registro de la pantalla. Se utilizan cuando el usuario ha de cambiar o actualizar los datos escritos en la pantalla desde el programa.

Para obtener una descripción detallada de un archivo de comunicaciones de datos, consulte la publicación *ICF Programming*. Si desea obtener más información sobre los archivos de pantalla definidos externamente, consulte la publicación *Gestión de datos*. Para obtener una lista de las palabras clave de especificaciones de descripción de datos (DDS) válidas, consulte la publicación *DDS Reference*.

La Figura 100 muestra un ejemplo de las DDS para un archivo de dispositivo de pantalla:

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ARCHIVO DE CONSULTA DEL MAESTRO DE CLIENTES -- CUSMINQ
     A*
     A                                      REF(CUSMSTP)  1
     A          R CUSPMT                    TEXT('SOLICITUD CLIENTE')
     A                                      CA01(15 'FIN DE PROGRAMA')  2
     A                                    1  3'CONSULTA MAESTRO CLIENTES'
     A                                    3  3'NÚMERO CLIENTE'
     A            CUST      R        I  3 20
     A  99                                  ERRMSG('NOMBRE CLIENTE NO ENCONTRADO +  3
     A                                      PULSE RESTABLECER Y ENTRE UN NÚMERO +
     A                                      VÁLIDO' 99)
     A                                    5  3'USE CF1 PARA FINALIZAR PROGRAMA, USE +
     A                                      INTRO PARA VOLVER A PANT.SOLICITUD'
     A          R CUSFLDS                   TEXT('PANTALLA CLIENTE')
     A                                      CA01(15 'FIN DE PROGRAMA')
     A                                      OVERLAY  4
     A                                    8  3'NOMBRE'
     A            NAME      R           8 11
     A                                    9  3'DIRECCIÓN'
     A            ADDR      R           9 11
     A                                   10  3'CIUDAD'  5
     A            CITY      R          10 11
     A                              6    11  3'ESTADO'
     A            STATE     R          11 11
     A                                  11 21'CÓD. POSTAL'
     A            ZIP       R          11 31
     A                                  12  3'SALDO C/C'
     A            ARBAL     R          12 17
```

Figura 100. Ejemplo de especificaciones de descripción de datos para un archivo de dispositivo de pantalla

Este archivo de dispositivo de pantalla contiene dos formatos de registro: CUSPMT y CUSFLDS.

1 Los atributos de los campos de este archivo están definidos en el archivo de referencia de campos CUSMSTP. Por ejemplo, EDTCDE(J) está definido en CUSMSTP para el campo ARBAL.

2 La tecla F1 está asociada con el indicador 15, con el que el usuario finaliza el programa.

3 La palabra clave ERRMSG identifica el mensaje de error que se visualiza si el indicador 99 está activado en el programa que utiliza este formato de registro.

4 La palabra clave OVERLAY se utiliza para el formato de registro CUSFLDS de manera que el registro CUSPMT de la pantalla no se borre cuando se escriba el registro CUSFLDS en la pantalla.

5 Las constantes, como por ejemplo ‘Nombre’, ‘Dirección’ y ‘Ciudad’, describen los campos que escribe el programa.

6 Las entradas de línea y posición identifican el punto en el que se escriben los campos o las constantes en la pantalla.

Proceso de un archivo de transacción descrito externamente

Cuando se procesa un archivo TRANSACTION descrito externamente, el sistema operativo transforma los datos del programa ILE COBOL/400 en el formato especificado para el archivo y visualiza los datos. Cuando los datos pasan al programa ILE COBOL/400, se transforman en el formato utilizado por el programa ILE COBOL/400.

El sistema operativo facilita la información de control de dispositivo para realizar las operaciones de entrada/salida para el dispositivo. Cuando el programa ILE

COBOL/400 solicita un registro de entrada del dispositivo, el sistema operativo emite la petición y elimina la información de control de dispositivo de los datos antes de pasarlos al programa. Además, el sistema operativo puede pasar indicadores al programa ILE COBOL/400 que indiquen qué campos del registro han cambiado, si es que alguno lo ha hecho.

Cuando el programa ILE COBOL/400 solicita una operación de salida, pasa el registro de salida al sistema operativo. El sistema operativo facilita la información de control de dispositivo necesaria para visualizar el registro. También añade la información sobre constantes especificada para el formato de registro al visualizarse el registro.

Cuando un registro pasa al programa ILE COBOL/400, los campos se disponen en el orden en que están especificados en la DDS. El orden en que se visualizan los campos está basado en las posiciones de pantalla (números de línea y posiciones) asignadas a los campos en las DDS. Por lo tanto, el orden en que los campos están especificados en las DDS y el orden en el que aparecen en la pantalla no es necesariamente el mismo.

Escritura de programas que utilicen archivos de transacción

Por lo general, los archivos TRANSACTION se utilizan para leer un registro de una pantalla o para escribir un registro en ella. Para utilizar un archivo TRANSACTION en un programa ILE COBOL/400, debe:

- darle nombre al archivo por medio de una entrada de control de archivo en el párrafo FILE-CONTROL de Environment Division
- describir el archivo mediante una entrada de descripción de archivo en Data Division
- utilizar ampliaciones a las instrucciones de Procedure Division que den soporte al proceso de transacciones.

Nota: No se recomienda utilizar instrucciones ACCEPT/DISPLAY ampliadas y archivos TRANSACTION en el mismo programa. Si se utilizan instrucciones ACCEPT/DISPLAY ampliadas en el mismo programa que los archivos TRANSACTION, el archivo TRANSACTION debe cerrarse cuando se realicen las instrucciones ACCEPT/DISPLAY ampliadas. Si una instrucción ACCEPT/DISPLAY ampliada se realiza cuando hay un archivo TRANSACTION abierto, se producirán resultados imprevistos. Es posible que se genere un error grave o que los datos de la estación de trabajo se superpongan o se entremezclen.

Denominación de un archivo de transacción

Para utilizar un archivo TRANSACTION en un programa ILE COBOL/400, debe dar nombre al archivo por medio de una entrada de control de archivo en el párrafo FILE-CONTROL. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción completa del párrafo FILE-CONTROL.

Dé nombre al archivo TRANSACTION en el párrafo FILE-CONTROL de la forma siguiente:

```
FILE-CONTROL.
    SELECT nombre-archivo-transacción
        ASSIGN TO WORKSTATION-nombre_archivo_pantalla
        ORGANIZATION IS TRANSACTION
        ACCESS MODE IS SEQUENTIAL
        CONTROL AREA IS dato-área-control.
```

Utilice la cláusula SELECT para elegir un archivo. Dicho archivo debe estar identificado con una entrada FD en Data Division.

Utilice la cláusula ASSIGN para asociar el archivo TRANSACTION con un archivo de pantalla o un archivo ICF.Debe especificar el tipo de dispositivo WORKSTATION en la cláusula ASSIGN para utilizar archivos TRANSACTION. Si desea utilizar un área de indicadores separada para este archivo TRANSACTION, será necesario que incluya el atributo -SI con la cláusula ASSIGN. Consulte el apartado “Utilización de indicadores con archivos de transacción” en la página 411 para obtener información más detallada sobre la utilización del área de indicadores por separado.

Debe especificar ORGANIZATION IS TRANSACTION en la entrada de control de archivo para poder utilizar un archivo TRANSACTION. Esta cláusula le indica al programa ILE COBOL/400 que interactuará con un usuario de estación de trabajo o con otro sistema.

A un archivo TRANSACTION se accede de forma secuencial. La cláusula ACCESS MODE de la entrada de control de archivo sirve para explicarle al programa ILE COBOL/400 la forma de acceder al archivo TRANSACTION. Especifique ACCESS MODE IS SEQUENTIAL para leer o escribir en el archivo TRANSACTION en orden secuencial.Si no especifica la cláusula ACCESS MODE, se supone que el acceso es secuencial

Si desea información de retorno sobre el estado de una petición de entrada-salida que haga referencia a un archivo TRANSACTION, defina un dato de clave de estado en la entrada de control de archivo con la cláusula FILE STATUS. Cuando se especifica la cláusula FILE STATUS, el sistema mueve un valor en el dato de clave de estado después de cada petición de entrada-salida que haga referencia de forma explícita o implícita al archivo TRANSACTION. El valor indica el estado de la ejecución de la instrucción de E-S.

Puede obtener información específica, dependiente del dispositivo y dependiente del sistema que se utiliza para controlar la operaciones de entrada-salida de los archivos TRANSACTION, identificando un dato de área de control con la cláusula CONTROL-AREA. Puede definir el dato especificado por la cláusula CONTROL-AREA en LINKAGE SECTION o WORKING-STORAGE SECTION con el formato siguiente:

```
01  dato-área-control.
    05  tecla-función         PIC X(2).
    05  nombre-dispositivo    PIC X(10).
    05  formato-registro      PIC X(10).
```

El área de control puede tener una longitud de 2, 12 ó 22 bytes. Así pues, sólo puede especificar el primer elemento de nivel 05, los dos primeros elementos de nivel 05 o los tres elementos de nivel 05, dependiendo del tipo de información que busque.

El dato de área de control le permitirá identificar:

- la tecla de función que pulsa el operador para iniciar una transacción
- el nombre del dispositivo de programa utilizado
- el nombre del formato de registro de DDS al que ha hecho referencia en la última instrucción de E-S.

Descripción de un archivo de transacción

Para utilizar un archivo TRANSACTION en un programa ILE COBOL/400, debe describir el archivo por medio de una entrada de descripción de archivo en Data Division. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción completa de la entrada de descripción de archivo.Para describir un archivo TRANSACTION, utilice la entrada de descripción de archivo de formato 6.

Una entrada de descripción de archivo de Data Division que describe a un archivo TRANSACTION es así:

```
FD  CUST-DISPLAY.
01  DISP-REC.
    COPY DDS-ALL-FORMATS OF CUSMINQ.
```

En ILE COBOL/400, los archivos TRANSACTION suelen estar descritos externamente.Cree una DDS para el archivo TRANSACTION que desee utilizar.Consulte el apartado “Definición de archivos de transacción con especificaciones de descripción de datos” en la página 395 para saber cómo crear una DDS o bien la publicación *DDS Reference*. A continuación, cree el archivo TRANSACTION.

Una vez haya creado la DDS para el archivo TRANSACTION así como éste, utilice la instrucción COPY de formato 2 para describir el diseño del registro de datos del archivo TRANSACTION. Al compilar el programa ILE COBOL/400, COPY de formato 2 creará las instrucciones de Data Division para describir el archivo TRANSACTION. Para generar un área de almacenamiento para todos los formatos, utilice la opción DDS-ALL-FORMATS de la instrucción COPY de formato 2.

Proceso de un archivo de transacción

La lista que sigue a continuación muestra todas las instrucciones de Procedure Division que tienen ampliaciones específicamente para procesar archivos TRANSACTION en un programa ILE COBOL/400. Consulte la publicación *ILE COBOL/400 Reference* si desea una explicación detallada de cada una de estas sentencias.

- Instrucción ACCEPT - formato 6
- Instrucción ACQUIRE
- Instrucción CLOSE - formato 1
- Instrucción DROP
- Instrucción OPEN - formato 3
- Instrucción READ - formato 4 (no subarchivo)
- Instrucción WRITE - formato 4 (no subarchivo)

Apertura de un archivo de transacción

Para procesar un archivo TRANSACTION en Procedure Division, primero debe abrir el archivo.Para abrir un archivo TRANSACTION, utilice la sentencia OPEN de formato 3.Un archivo TRANSACTION debe abrirse en modalidad de E-S.

```
OPEN I-O nombre-archivo.
```

Adquisición de dispositivos de programa

Debe adquirir un dispositivo de programa para el archivo TRANSACTION. Una vez adquirido, estará disponible para operaciones de entrada y salida.Puede adquirirlo de forma implícita o explícita.

Un dispositivo de programa se adquiere implícitamente al abrir el archivo TRANSACTION.Si el archivo es un archivo ICF, el dispositivo de archivo adquirido implícitamente está determinado por el parámetro ACQPGMDEV del mandato CRTICFF. Si es un archivo de pantalla, el dispositivo está determinado por la primera entrada del parámetro DEV del mandato CRTDSPF. Los dispositivos de programa adicionales deben adquirirse explícitamente.

Un dispositivo de programa se adquiere explícitamente con la instrucción ACQUIRE. Para un archivo ICF, el dispositivo debe habérsele definido al archivo con el mandato ADDICFDEVE o OVRICFDEVE CL antes de abrir el archivo.Para los archivos de pantalla no existe tal requisito. Es decir, el dispositivo mencionado en la instrucción ACQUIRE no ha de estar especificado en el parámetro DEV de los mandatos CRTDSPF, CHGDSPF o OVRDSPF. Sin embargo, al crear el archivo de pantalla, debe especificar el número de dispositivos que pueden adquirirse (el valor por omisión es uno). Para un archivo de pantalla, el nombre de dispositivo de programa debe coincidir con el dispositivo de pantalla.

```
ACQUIRE nombre-dispositivo-programa FOR nombre-archivo-transacción.
```

Escritura en un archivo de transacción

Una vez abierto el archivo TRANSACTION y adquirido un dispositivo de programa, ya está preparado para realizar operaciones de entrada y salida en él.

La primera operación de entrada/salida que suele realizarse en un archivo TRANSACTION es escribir un registro en la pantalla. Este registro se utiliza para pedir al usuario que entre una respuesta o datos.

Para escribir un registro lógico en el archivo TRANSACTION, utilice la instrucción WRITE de formato 4.El código de la sentencia WRITE es el siguiente:

```
WRITE nombre-registro FORMAT IS nombre-formato.
```

En determinadas situaciones, es posible que tenga varios registros de datos, cada uno de ellos con un formato diferente, que desee tener activos para un archivo TRANSACTION. En tal caso, debe utilizar la expresión FORMAT de la instrucción WRITE de formato 4 para especificar el formato del registro de datos de salida que desea escribir en el archivo TRANSACTION.

Si ha adquirido de forma explícita varios dispositivos de programa para el archivo TRANSACTION, debe utilizar la expresión TERMINAL de la instrucción WRITE de formato 4 para especificar el dispositivo de programa al que desea que se envíe el registro de salida.

Puede controlar el número de línea de la pantalla en el que la instrucción WRITE grabará el registro de salida especificando la expresión STARTING y ROLLING de la instrucción WRITE de formato 4. La expresión STARTING especifica el número de línea inicial de los formatos de registros que utilizan la palabra clave de línea inicial de registro variable.La expresión ROLLING le permite mover las líneas visualizadas en la pantalla de la estación de trabajo.Se pueden girar hacia arriba o hacia abajo todas las líneas de la pantalla o parte de ellas.

```
WRITE nombre-registro FORMAT IS nombre-formato
                  TERMINAL IS nombre-dispositivo-programa
                  STARTING AT LINE n -línea-inicial
                  AFTER ROLLING LINES n -primera-línea THRU n -última-línea
                                DOWN n -de-líneas LINES
END-WRITE.
```

Lectura en un archivo de transacción

Para leer un registro lógico en el archivo TRANSACTION, utilice la instrucción READ de formato 4.Si hay datos disponibles al ejecutar la instrucción READ, se devuelven en el área de registro. Los nombres del formato de registro y del dispositivo de programa se devuelven en el área I-O-FEEDBACK y CONTROL-AREA.

Antes de utilizar la instrucción READ, debe haber adquirido al menos un dispositivo de programa para el archivo TRANSACTION. Si se realiza una instrucción READ y no hay dispositivos de programa adquiridos, se notifica un error de lógica estableciendo el estado de archivo en 92.

La forma más simple de la instrucción READ es la siguiente:

```
READ nombre-registro RECORD.
```

Si sólo a adquirido un dispositivo de programa, la forma simple de la instrucción READ esperará siempre a que los datos estén disponibles. Incluso si el trabajo recibe una cancelación controlada, o espera el espacio de tiempo especificado en el parámetro WAITRCD al archivo de pantalla o al archivo ICF, el programa no recuperará nunca el control desde la instrucción READ.

Si ha adquirido varios dispositivos de programa, la forma simple de la instrucción READ recibirá los datos del primer dispositivo de programa invitado que tenga datos disponibles.Cuando se han adquirido varios dispositivos de programa, la forma simple de la instrucción READ puede finalizar sin devolver ningún dato si no hay dispositivos invitados y no se ha especificado ningún tiempo de espera, si se produce una cancelación controlada del trabajo o si el tiempo de espera especificado expira.

Para obtener una explicación detallada de cómo se realiza la instrucción READ, consulte el apartado dedicado a la instrucción en la publicación *ILE COBOL/400 Reference*.

En aquellos casos en los que haya adquirido múltiples dispositivos de programa, puede especificar explícitamente el dispositivo de programa del que lee datos, identificándolo en la expresión TERMINAL de la instrucción READ.

En aquellos casos en los que desee recibir los datos con un formato específico, puede identificarlo en la expresión FORMAT de la instrucción READ. Si los datos disponibles no coinciden con el formato de registro solicitado, se establece el estado de archivo 9K.

Los ejemplos que siguen a continuación son de la instrucción READ con las expresiones TERMINAL y FORMAT especificadas.

```
READ nombre-registro RECORD
     FORMAT IS formato-registro
END-READ

READ nombre-registro RECORD
     TERMINAL IS nombre-dispositivo-programa
END-READ

READ nombre-registro RECORD
     FORMAT IS formato-registro
     TERMINAL IS nombre-dispositivo-programa
END-READ
```

Al realizar la instrucción READ, pueden producirse las condiciones siguientes:

1. Los datos están disponibles de forma inmediata y la condición AT END no existe.La condición AT END se produce cuando no hay dispositivos de programa invitados y no se ha especificado ningún tiempo de espera.
2. Los datos no están disponibles de forma inmediata.
3. Existe la condición AT END.

Puede transferir el control a diversas instrucciones del programa ILE COBOL/400 desde la instrucción READ tomando como base la condición que resulta de realizar instrucción READ especificando la expresión NO DATA, AT END o NOT AT END.

Para realizar un grupo de instrucciones cuando la instrucción READ finaliza satisfactoriamente, especifique la expresión NOT AT END de la instrucción READ.

Para realizar un grupo de instrucciones cuando los datos no están disponibles de forma inmediata, especifique la expresión NO DATA de la instrucción READ.La expresión NO DATA impide que la instrucción READ espere a que los datos están disponibles.

Para realizar un grupo de instrucciones cuando existe la condición AT END, especifique la expresión AT END de la instrucción READ.

Los ejemplos que siguen a continuación son de la instrucción READ con las expresiones NO DATA, NOT AT END y AT END especificadas.

```
READ nombre-registro RECORD
     TERMINAL IS nombre-dispositivo-programa
     NO DATA instrucción-imperativa-1
END-READ

READ nombre-registro RECORD
     TERMINAL IS nombre-dispositivo-programa
     AT END instrucción-imperativa-2
     NOT AT END instrucción-imperativa-3
END-READ
```

Liberación de dispositivos de programa

Una vez que haya terminado de utilizar un dispositivo de programa previamente adquirido para un archivo TRANSACTION, debe liberarlo. Liberar un dispositivo de programa significa que ya no estará disponible para operaciones de entrada o salida mediante el archivo TRANSACTION. Liberando un dispositivo de programa éste queda disponible para otras aplicaciones. Un dispositivo de programa puede liberarse de forma implícita o explícita.

Los dispositivos de programa conectados a un archivo TRANSACTION se liberan implícitamente al cerrar el archivo.

Un dispositivo de programa se libera explícitamente indicándolo en la instrucción DROP. El dispositivo, una vez liberado, puede volver a adquirirse otra vez si fuese necesario.

```
DROP nombre-dispositivo-programa FROM nombre-archivo-transacción.
```

Cierre de un archivo TRANSACTION

Cuando termine de utilizar un archivo TRANSACTION, debe cerrarlo.Para cerrar un archivo TRANSACTION, utilice la sentencia CLOSE de formato 1. Una vez cerrado el archivo, no puede procesarse de nuevo hasta que vuelva a abrirse.

```
CLOSE nombre-archivo-transacción.
```

Ejemplo de un programa de consulta básica con archivos de transacción

La Figura 101 muestra las DDS asociadas para un programa de consulta básica que utiliza el archivo TRANSACTION ILE COBOL/400.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ARCHIVO DE CONSULTA DEL MAESTRO DE CLIENTES -- CUSMINQ
     A*
     A                                      REF(CUSMSTP)
     A          R CUSPMT                    TEXT('SOLICITUD CLIENTE')
     A                                      CA01(15 'FIN DE PROGRAMA')
     A                                    1  3'CONSULTA MAESTRO CLIENTES'
     A                                    3  3'NÚMERO CLIENTE'
     A            CUST      R        I  3 20
     A  99                                  ERRMSG('NÚMERO CLIENTE NO ENCONTRADO +
     A                                      PULSE RESTABLECER Y ENTRE UN NÚMERO +
     A                                      VÁLIDO' 99)
     A  98                                  ERRMSG('CONDICIÓN FIN ARCHIVO EN +
     A                                      READ, FINALIZÓ PROGRAMA' 98)
     A                                    5  3'USE F1 PARA FINALIZAR PROGRAM, USE+
     A                                       INTRO PARA VOLVER A PANT.SOLICITUD'
     A          R CUSFLDS                   TEXT('PANTALLA CLIENTE')
     A                                      CA01(15 'FIN DE PROGRAMA')
     A                                      OVERLAY
     A                                    8  3'NOMBRE'
     A            NAME      R          8 11
     A                                    9  3'DIRECCIÓN'
     A            ADDR      R          9 11
     A                                   10  3'CIUDAD'
     A            CITY      R         10 11
     A                                   11  3'ESTADO'
     A            STATE     R         11 11
     A                                   11 21'CÓD. POSTAL'
     A            ZIP       R         11 31
     A                                   12  3'SALDO C/C'
     A            ARBAL     R         12 17
```

Figura 101. Ejemplo de un programa de consulta TRANSACTION con un solo dispositivo de pantalla

Las especificaciones de descripción de datos (DDS) del archivo de dispositivo de pantalla (CUSMINQ) que ha de utilizar el programa describen dos formatos de registro: CUSPMT y CUSFLDS.

El formato de registro CUSPMT contiene la constante ‘Consulta maestro de clientes’, que identifica la pantalla. También contiene la solicitud ‘Número de cliente’ y el campo de entrada (CUST) en el que se puede entrar el número de cliente. Debajo del campo de entrada CUST aparecen cinco caracteres de subrayado en la pantalla en el punto en el que se ha de entrar el número de cliente. También se incluye el mensaje de error:

```
Número de cliente no encontrado
```

en este formato de registro. Este mensaje se visualiza si el indicador 99 está establecido como **ON** por el programa. Además, este formato de registro define una tecla de función que puede pulsarse para finalizar el programa. Al pulsar la tecla de función F1, el indicador 15 se establece como **ON** en el programa ILE COBOL/400. Este indicador se utilizará entonces para finalizar el programa.

El registro de formato CUSFLDS contiene las constantes siguientes:

- Nombre
- Dirección
- Ciudad
- Provincia
- Código postal
- Saldos de cuentas a cobrar (Saldo C/C).

Estas constantes identifican los campos que han de escribirse desde el programa.Este formato de registro también describe los campos que corresponden a estas constantes.Todos estos campos están descritos como campos de salida (blanco en la posición 38) porque los rellena el programa; el usuario no entra ningún dato en ellos.Para entrar otro número de cliente, pulse Intro como respuesta a este registro. Observará que el registro CUSFLDS se superpone al registro CUSPMT. Por tanto, cuando se escriba el registro CUSFLDS en la pantalla, el registro CUSPMT permanecerá en la pantalla.

Además de describir las constantes, los campos y los atributos de la pantalla, los formatos de registro también definen los números de línea y las posiciones horizontales en los que se visualizarán las constantes y los campos.

Nota: Los atributos de campo se definen en un archivo físico (CUSMSTP) utilizado con fines de referencia de campo en lugar de las DDS del archivo de pantalla.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* ESTE ES EL ARCHIVO MAESTRO DE CLIENTES -- CUSMSTP
     A
     A
     A                                     UNIQUE
     A          R CUSMST                   TEXT('CUSTOMER MASTER RECORD')
     A            CUST           5         TEXT('CUSTOMER NUMBER')
     A            NAME          25         TEXT('CUSTOMER NAME')
     A            ADDR          20         TEXT('CUSTOMER ADDRESS')
     A            CITY          20         TEXT('CUSTOMER CITY')
     A            STATE          2         TEXT('STATE')
     A            ZIP            5 00      TEXT('ZIP CODE')
     A            SRHCOD         6         TEXT('CUSTOMER NUMBER SEARCH CODE')
     A            CUSTYP         1 00      TEXT('CUSTOMER TYPE 1=GOV 2=SCH +
     A                                     3=BUS 4=PVT 5=OT')
     A            ARBAL          8 02      TEXT('ACCOUNTS REC. BALANCE')
     A            ORDBAL         8 02      TEXT('A/R AMT. IN ORDER FILE')
     A            LSTAMT         8 02      TEXT('LAST AMT. PAID IN A/R')
     A            LSTDAT         6 00      TEXT('LAST DATE PAID IN A/R')
     A            CRDLMT         8 02      TEXT('CUSTOMER CREDIT LIMIT')
     A            SLSYR         10 02      TEXT('CUSTOMER SALES THIS YEAR')
     A            SLSLYR        10 02      TEXT('CUSTOMER SALES LAST YEAR')
                K CUST
```

Figura 102. Especificación de descripción de datos del formato de registro CUSMST.

Las especificaciones de descripción de datos (DDS) del archivo de base de datos que utiliza este programa describen un formato de registro: CUSMST. Cada campo del formato de registro está descrito y el campo CUST está identificado como campo de clave del formato de registro.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INQUIRY          AS400SYS   96/07/04 11:34:12        Página   2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INQUIRY.
          000300*    PROGRAMA DE CONSULTA DE TRANSACCION DE EJEMPLO CON 1 DISPOSITIVO DE PANTALLA
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600 CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-AS400.
    6     000800 OBJECT-COMPUTER. IBM-AS400.
    7     000900 INPUT-OUTPUT SECTION.
    8     001000 FILE-CONTROL.
    9     001100     SELECT CUST-DISPLAY
   10     001200         ASSIGN TO WORKSTATION-CUSMINQ
   11     001300         ORGANIZATION IS TRANSACTION
   12     001400         CONTROL-AREA IS WS-CONTROL.
   13     001500     SELECT CUST-MASTER
   14     001600         ASSIGN TO DATABASE-CUSMSTP
   15     001700         ORGANIZATION IS INDEXED
   16     001800         ACCESS IS RANDOM
   17     001900             RECORD KEY IS CUST OF CUSMST
   18     002000         FILE STATUS IS CM-STATUS.
          002100
   19     002200 DATA DIVISION.
   20     002300 FILE SECTION.
   21     002400 FD  CUST-DISPLAY.
   22     002500 01  DISP-REC.
          002600     COPY DDS-ALL-FORMATS OF CUSMINQ.
   23    +000001       05  CUSMINQ-RECORD PIC X(80).                                  <-ALL-FMTS
         +000002*  FORMATO ENTRADA:CUSPMT     DEL ARCH. CUSMINQ    DE LA BIBL. TESTLIB       <-ALL-FMTS
         +000003*                           SOLICITUD DE CLIENTE                      <-ALL-FMTS
   24    +000004       05  CUSPMT-I      REDEFINES CUSMINQ-RECORD.                   <-ALL-FMTS
   25    +000005           06 CUSPMT-I-INDIC.                                         <-ALL-FMTS
   26    +000006                07 IN15            PIC 1  INDIC 15.                  <-ALL-FMTS
         +000007*                           FIN DEL PROGRAMA                          <-ALL-FMTS
   27    +000008                07 IN99            PIC 1  INDIC 99.                  <-ALL-FMTS
         +000009*                           N  CLIENTE NO ENCONTRADO, RESTABLECER E INTRO     <-ALL-FMTS
   28    +000010                07 IN98            PIC 1  INDIC 98.                  <-ALL-FMTS
         +000011*                           CONDICION EOF EN READ, PROGRAMA FINALIZADO        <-ALL-FMTS
   29    +000012           06 CUST                 PIC X(5).                         <-ALL-FMTS
         +000013*                 NUMERO DE CLIENTE                                 <-ALL-FMTS
         +000014*  FORMATO SALIDA:CUSPMT     DEL ARCH. CUSMINQ DE LA BIBL. TESTLIB     <-ALL-FMTS
         +000015*                           SOLICITUD DE CLIENTE                      <-ALL-FMTS
   30    +000016       05  CUSPMT-O      REDEFINES CUSMINQ-RECORD.                   <-ALL-FMTS
   31    +000017           06 CUSPMT-O-INDIC.                                         <-ALL-FMTS
   32    +000018                07 IN99            PIC 1  INDIC 99.                  <-ALL-FMTS
         +000019*                           N  CLIENTE NO ENCONTRADO, RESTABLECER E INTRO     <-ALL-FMTS
   33    +000020                07 IN98            PIC 1  INDIC 98.                  <-ALL-FMTS
         +000021*                           CONDICION EOF EN READ, PROGRAMA FINALIZADO        <-ALL-FMTS
         +000022*  FORMATO ENTRADA:CUSFLDS     DEL ARCH. CUSMINQ    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000023*                           PANTALLA DE CLIENTE                       <-ALL-FMTS
   34    +000024       05  CUSFLDS-I     REDEFINES CUSMINQ-RECORD.                   <-ALL-FMTS
   35    +000025           06 CUSFLDS-I-INDIC.                                        <-ALL-FMTS
   36    +000026                07 IN15            PIC 1  INDIC 15.                  <-ALL-FMTS
         +000027*                           FIN DEL PROGRAMA                          <-ALL-FMTS
         +000028*  FORMATO SALIDA:CUSFLDS     DEL ARCH. CUSMINQ    DE LA BIBL. TESTLIB       <-ALL-FMTS
         +000029*                           PANTALLA DE CLIENTE                       <-ALL-FMTS
```

Figura 103 (Parte 1 de 4). Listado fuente de un programa de consulta TRANSACTION con un solo dispositivo de pantalla.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INQUIRY          AS400SYS   96/07/04 11:34:12         Página   3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


  37    +000030       05  CUSFLDS-O     REDEFINES CUSMINQ-RECORD.                          <-ALL-FMTS
  38    +000031           06 NAME                  PIC X(25).                              <-ALL-FMTS
        +000032*                 NOMBRE DEL CLIENTE                                        <-ALL-FMTS
  39    +000033           06 ADDR                  PIC X(20).                              <-ALL-FMTS
        +000034*                 DIRECCION DEL CLIENTE                                     <-ALL-FMTS
  40    +000035           06 CITY                  PIC X(20).                              <-ALL-FMTS
        +000036*                 LOCALIDAD DEL CLIENTE                                     <-ALL-FMTS
  41    +000037           06 STATE                 PIC X(2).                               <-ALL-FMTS
        +000038*                 PROVINCIA                                                 <-ALL-FMTS
  42    +000039           06 ZIP                   PIC S9(5).                              <-ALL-FMTS
        +000040*                 CODIGO POSTAL                                             <-ALL-FMTS
  43    +000041           06 ARBAL                 PIC S9(6)V9(2).                         <-ALL-FMTS
        +000042*                 SALDO DE CUENTAS POR COBRAR                               <-ALL-FMTS
         002700
  44     002800 FD  CUST-MASTER.
  45     002900 01  CUST-REC.
         003000     COPY DDS-CUSMST OF CUSMSTP.
        +000001*    FORMATO E-S:CUSMST     DEL ARCH. CUSMSTP    DE LA BIBL. TESTLIB          CUSMST
        +000002*                        REGISTRO MAESTRO DE CLIENTES                         CUSMST
        +000003*          CLAVE DADA POR USUARIO CON CLAUSULA RECORD KEY                      CUSMST
  46    +000004       05  CUSMST.                                                           CUSMST
  47    +000005           06 CUST                  PIC X(5).                               CUSMST
        +000006*                 NUMERO DE CLIENTE                                         CUSMST
  48    +000007           06 NAME                  PIC X(25).                              CUSMST
        +000008*                 NOMBRE DEL CLIENTE                                        CUSMST
  49    +000009           06 ADDR                  PIC X(20).                              CUSMST
        +000010*                 DIRECCION DEL CLIENTE                                     CUSMST
  50    +000011           06 CITY                  PIC X(20).                              CUSMST
        +000012*                 LOCALIDAD DEL CLIENTE                                     CUSMST
  51    +000013           06 STATE                 PIC X(2).                               CUSMST
        +000014*                 PROVINCIA                                                 CUSMST
  52    +000015           06 ZIP                   PIC S9(5)       COMP-3.                 CUSMST
        +000016*                 CODIGO POSTAL                                             CUSMST
  53    +000017           06 SRHCOD                PIC X(6).                               CUSMST
        +000018*                 CODIGO DE BUSQUEDA DE NUMERO DE CLIENTE                   CUSMST
  54    +000019           06 CUSTYP                PIC S9(1)       COMP-3.                 CUSMST
        +000020*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT              CUSMST
  55    +000021           06 ARBAL                 PIC S9(6)V9(2)  COMP-3.                 CUSMST
        +000022*                 SALDO DE CUENTAS POR COBRAR                               CUSMST
  56    +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                 CUSMST
        +000024*                 IMPORTE DE CC EN ARCHIVO DE PEDIDOS                       CUSMST
  57    +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
        +000026*                 ULTIMO IMPORTE PAGADO EN CC                               CUSMST
  58    +000027           06 LSTDAT                PIC S9(6)       COMP-3.                 CUSMST
        +000028*                 ULTIMA FECHA DE PAGO EN CC                                CUSMST
  59    +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
        +000030*                 LIMITE DE CREDITO DEL CLIENTE                             CUSMST
  60    +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                 CUSMST
        +000032*                 VENTAS DEL CLIENTE EN EL AÑO ACTUAL                       CUSMST
  61    +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                 CUSMST
        +000034*                 VENTAS DEL CLIENTE EL AÑO PASADO                          CUSMST
         003100
  62     003200 WORKING-STORAGE SECTION.
  63     003300 01  ONE                              PIC 1 VALUE B"1".
  64     003400 01  CM-STATUS                       PIC X(2).
  65     003500 01  WS-CONTROL.
  66     003600     02  WS-IND                       PIC X(2).
```

Figura 103 (Parte 2 de 4). Listado fuente de un programa de consulta TRANSACTION con un solo dispositivo de pantalla.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INQUIRY          AS400SYS   96/07/04 11:34:12         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   67     003700     02  WS-FORMAT                      PIC X(10).
          003800
   68     003900 PROCEDURE DIVISION.
   69     004000 DECLARATIVES.
          004100 DISPLAY-ERR-SECTION SECTION.
          004200     USE AFTER STANDARD EXCEPTION PROCEDURE ON CUST-DISPLAY.
          004300 DISPLAY-ERR-PARAGRAPH.
   70     004400     MOVE ONE TO IN98 OF CUSPMT-O
   71     004500     WRITE DISP-REC FORMAT IS "CUSPMT"
          004600     END-WRITE
   72     004700     CLOSE CUST-MASTER
          004800           CUST-DISPLAY.
   73     004900     STOP RUN.
          005000 END DECLARATIVES.
          005100
          005200 MAIN-PROGRAM SECTION.
          005300 MAINLINE.
   74     005400     OPEN INPUT CUST-MASTER
          005500          I-O CUST-DISPLAY.
          005600
   75     005700     MOVE ZERO TO IN99 OF CUSPMT-O
   76     005800     WRITE DISP-REC FORMAT IS "CUSPMT"  1
          005900     END-WRITE
   77     006000     READ CUST-DISPLAY RECORD
          006100     END-READ
          006200
   78     006300     PERFORM UNTIL IN15 OF CUSPMT-I IS EQUAL TO ONE
          006400
   79     006500         MOVE CUST OF CUSPMT-I TO CUST OF CUSMST
   80     006600         READ CUST-MASTER RECORD  2
          006700             INVALID KEY  3
   81     006800                 MOVE ONE TO IN99 OF CUSPMT-O
   82     006900                 WRITE DISP-REC FORMAT IS "CUSPMT"
          007000                 END-WRITE
   83     007100                 READ CUST-DISPLAY RECORD
          007200                 END-READ
          007300             NOT INVALID KEY
   84     007400                 MOVE CORRESPONDING CUSMST TO CUSFLDS-O                                  96/11/08
                *** Elementos CORRESPONDING para la instrucción 007400:
                ***       NAME
                ***       ADDR
                ***       CITY
                ***       STATE
                ***       ZIP
                ***       ARBAL
                *** Fin de elementos CORRESPONDING para la instrucción 007400
   85     007500                 WRITE DISP-REC FORMAT IS "CUSFLDS"
          007600                 END-WRITE
   86     007700                 READ CUST-DISPLAY RECORD
          007800                 END-READ
   87     007900                 IF  IN15 OF CUSPMT-I IS NOT EQUAL TO ONE
   88     008000                     MOVE ZERO TO IN99 OF CUSPMT-O
   89     008100                     WRITE DISP-REC FORMAT IS "CUSPMT"
          008200                     END-WRITE
   90     008300                     READ CUST-DISPLAY RECORD
          008400                     END-READ
          008500                 END-IF
```

Figura 103 (Parte 3 de 4). Listado fuente de un programa de consulta TRANSACTION con un solo dispositivo de pantalla.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INQUIRY          AS400SYS   96/07/04 11:34:12         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


          008600         END-READ
          008700
          008800     END-PERFORM
          008900
   91     009000     CLOSE CUST-MASTER
          009100           CUST-DISPLAY.
   92     009200     GOBACK.

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 103 (Parte 4 de 4). Listado fuente de un programa de consulta TRANSACTION con un solo dispositivo de pantalla.

El listado fuente completo de este ejemplo de programa es el mostrado más arriba.En concreto, fíjese en las entradas FILE-CONTROL y FD y las estructuras de datos generadas por las instrucciones COPY de formato 2.

La operación WRITE de 1 escribe el formato CUSPMT en la pantalla. Este registro pide que se entre un número de cliente.Si entra un número de cliente y pulsa Intro, la siguiente operación READ lee el registro al programa.

La operación READ de 2 utiliza el campo de número de cliente (CUST) para recuperar el registro CUSMST correspondiente del archivo CUSMSTP. Si no se encuentra ningún archivo en el archivo CUSMSTP, se realizan las instrucciones imperativas INVALID KEY de 3 . Se activa el indicador 99 y se visualiza el mensaje:

```
Número de cliente no encontrado
```

al escribir el formato. El mensaje está condicionado por el indicador 99 de las DDS de archivo. Al recibirse este mensaje, se bloquea el teclado. Debe pulsar la tecla Restablecer como respuesta al mensaje para desbloquear el teclado.A continuación, podrá entrar otro número de cliente.

Si la operación READ recupera un registro del archivo CUSMSTP, la operación WRITE escribe el registro CUSFLDS en la estación de pantalla.Este registro contiene el nombre, la dirección y el saldo de cuentas por cobrar del cliente.

Pulse Intro y el programa volverá al principio. A continuación, podrá entrar otro número de cliente o finalizar el programa. Para finalizar el programa, pulse F1, que activa el indicador 15 en el programa.

Cuando está activo el indicador 15, el programa cierra todos los archivos y procesa la instrucción GOBACK. El programa devuelve el control a la persona que ha llamado al programa ILE COBOL/400.

Esta es la pantalla inicial escrita por la operación WRITE en 1 :

```
Consulta maestro de cliente

Número de cliente  _______

Utilice F3 para finalizar el programa, Intro para volver a la
pantalla de solicitud
```

Si se encuentra un registro en el archivo CUSMSTP para el número de cliente entrado como respuesta a la primera pantalla, aparece esta otra:

```
Consulta maestro de cliente

Número de cliente 1000

Utilice F3 para finalizar el programa, Intro para volver a la
pantalla de solicitud


Nombre    EXAMPLE WHOLESALERS LTD.
Dirección ANYWHERE STREET
Ciudad    ACITY
Provincia IL        Código postal 12345
Saldo C/C           137.02
```

Si el archivo CUSMSTP no contiene ningún registro para el número de cliente entrado como respuesta a la primera pantalla, aparece esta otra:

```
Consulta maestro de cliente

Número de cliente

Utilice F3 para finalizar el programa, Intro para volver a la
pantalla de solicitud




Número de cliente no encontrado, pulse restablecer y entre un número válido
```

Utilización de indicadores con archivos de transacción

Los indicadores son datos booleanos que pueden tener el valor B"0" o B"1".

Al definir un formato de registro para un archivo utilizando DDS, puede condicionar las opciones con indicadores; los indicadores también pueden utilizarse para

reflejar respuestas concretas. Estos indicadores se conocen con el nombre de OPTION y RESPONSE, respectivamente.

Los indicadores de opción dan opciones tales como espaciado, subrayado y permitir o solicitar la transferencia de datos del programa ILE COBOL/400 a una impresora o dispositivo de pantalla. Los indicadores de respuesta facilitan información de respuesta al programa ILE COBOL/400 procedente de un dispositivo, como por ejemplo las teclas de función pulsadas por un usuario de estación de trabajo y cuando se han entrado datos .

Los indicadores pueden pasarse con registros de datos en un área de registro o desde fuera del área de registro en un área de indicadores por separado.

Traspaso de indicadores en un área de indicadores por separado

Si especifica la palabra clave a nivel de archivo INDARA en las DDS, se pasan todos los indicadores definidos en el formato o formatos de registro de dicho archivo al programa ILE COBOL/400, y desde él, en un área de indicadores por separado, no en el área de registro. Para obtener información sobre la manera de especificar la palabra clave INDARA, consulte la publicación *DDS Reference*.

La entrada de control de archivo de un archivo que tenga especificada INDARA en las DDS debe tener un atributo de área de indicadores por separado, SI, que forme parte del nombre de asignación. Por ejemplo, la asignación de un archivo denominado DSPFILE es la siguiente:

```
FILE-CONTROL.
    SELECT DISPFILE
           ASSIGN TO WORKSTATION-DSPFILE-SI
           ORGANIZATION IS TRANSACTION
           ACCESS IS SEQUENTIAL.
```

Las ventajas de utilizar un área de indicadores por separado son las siguientes:

- El número y el orden de los indicadores utilizados en una instrucción de E/S de cualquier formato de registro de un archivo no ha de coincidir necesariamente con el número y el orden de los indicadores especificados en las DDS de dicho formato de registro.
- El programa asocia el número de indicador en una entrada de descripción de datos con el indicador pertinente.

Traspaso de indicadores en el área de registro

Si no se utiliza la palabra clave INDARA en las DDS del archivo, los indicadores se crean en el área de registro. Cuando los indicadores estás definidos en un formato de registro de un archivo, se leen, reescriben y escriben con los datos del área de registro.

El número y el orden de los indicadores definidos en las DDS de un formato de registro de un archivo determina el número y el orden en el que se deben codificar en el programa ILE COBOL/400 las entradas de descripción de datos de los indicadores del formato de registro.

La entrada de control de archivo de un archivo que no tenga especificada INDARA en sus DDS asociadas *no* debe tener el atributo de área de indicadores por separado, SI, formando parte del nombre de asignación.

Si utiliza una instrucción COPY de formato 2 para copiar los indicadores en el programa ILE COBOL/400, los indicadores se definirán en el orden en el que están especificados en las DDS del archivo.

Ejemplos de utilización de indicadores en programas ILE COBOL/400

Este apartado contiene ejemplos de programas ILE COBOL/400 que ilustran la utilización de los indicadores en un área de registro o en un área de indicadores por separado.

Todos los programas ILE COBOL/400 hacen lo siguiente:

1. Determinan la fecha actual.
2. Si se trata del primer día del mes, activan un indicador de opción que provoca la aparición de un campo de salida parpadeante.
3. Permiten que se pulsen teclas de función para terminar el programa o activan indicadores de respuesta y llaman a programas para escribir informes diarios o mensuales.

La Figura 105 en la página 415 muestra un programa ILE COBOL/400 que utiliza los indicadores en el área de registro, pero que no utiliza la expresión INDICATORS en ninguna instrucción de E/S. La Figura 104 muestra las DDS asociadas del archivo.

La Figura 106 en la página 418 muestra un programa ILE COBOL/400 que utiliza los indicadores en el área de registro y la expresión INDICATORS en las instrucciones de E/S. Las DDS asociadas de la Figura 106 están en la Figura 104.

La Figura 108 en la página 421 muestra un programa ILE COBOL/400 que utiliza los indicadores en un área de indicadores por separado, definida en WORKING-STORAGE SECTION con la instrucción COPY de formato 2. La Figura 107 en la página 420 muestra las DDS asociadas del archivo.

La Figura 109 en la página 424 muestra un programa ILE COBOL/400 que utiliza los indicadores en un área de indicadores por separado, definida en una tabla en WORKING-STORAGE SECTION. Las DDS asociadas del archivo son las mismas que las de la Figura 107 en la página 420.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* DDS DE ARCHIVO DE PANTALLA PARA EJEMPLOS DE INDICADORES - INDICADORES EN AREA DE REGISTRO
     A* DSPFILEX                              1
     A      2   R FORMAT1                  3 CF01(99 'FIN DE PROGRAMA')
     A                                      CF05(51 'INFORME DIARIO')
     A                                      CF09(52 'INFORME MENSUAL')
     A*
     A                              4    10 10'NÚM. DEPARTAMENTO:'
     A            DEPTNO          5   I 10 32
     A   5 01                             20 26'GENERAR INFORMES MENSUALES'
     A                                        DSPATR(BL)
     A*
     A                                6   24 01'F5 = INFORME DIARIO'
     A                                   24 26'F9 = INFORME MENSUAL'
     A                                   24 53'F1 = FINALIZAR'
     A          R ERRFMT
     A     98                              6  5'ERROR ENTRADA-SALIDA'
```

Figura 104. Ejemplo de un programa con indicadores en el área de registro sin utilizar la expresión INDICATORS en la instrucción E/S -- DDS

1 No se utiliza la palabra clave INDARA; los indicadores se almacenan en el área de registro con los campos de datos.

2 Se especifica el formato FORMAT1.

3 Se asocian tres indicadores con tres teclas de función. El indicador 99 se activará al pulsar F1 y así sucesivamente.

4 Se define un campo para entrada.

5 Se define el indicador 01 para hacer que el campo de constante asociado parpadee si el indicador está activo.

6 Las definiciones de tecla de función (F) estás documentadas en la pantalla de la estación de trabajo.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC1           AS400SYS   96/07/04 11:56:12         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020   PROGRAM-ID.     INDIC1.
          000030*    PROGRAMA DE EJEMPLO CON INDICADORES EN EL AREA DE REGISTRO.
          000040
    3     000050 ENVIRONMENT DIVISION.
    4     000060   CONFIGURATION SECTION.
    5     000070   SOURCE-COMPUTER. IBM-AS400.
    6     000080   OBJECT-COMPUTER. IBM-AS400.
    7     000090   INPUT-OUTPUT SECTION.
    8     000100   FILE-CONTROL.
    9     000110     SELECT DISPFILE
   10     000120       ASSIGN TO WORKSTATION-DSPFILEX  1
   11     000130       ORGANIZATION IS TRANSACTION
   12     000140       ACCESS       IS SEQUENTIAL.
          000150
   13     000160 DATA DIVISION.
   14     000170   FILE SECTION.
   15     000180   FD  DISPFILE.
   16     000190   01  DISP-REC.
          000200       COPY DDS-ALL-FORMATS OF DSPFILEX.  2
   17    +000001       05  DSPFILEX-RECORD PIC X(8).                                    <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   19    +000005           06 FORMAT1-I-INDIC.                                        <-ALL-FMTS
   20    +000006                07 IN99             PIC 1  INDIC 99.  3               <-ALL-FMTS
         +000007*                          FIN DEL PROGRAMA                          <-ALL-FMTS
   21    +000008                07 IN51             PIC 1  INDIC 51.                  <-ALL-FMTS
         +000009*                          INFORME DIARIO                            <-ALL-FMTS
   22    +000010                07 IN52             PIC 1  INDIC 52.                  <-ALL-FMTS
         +000011*                          INFORME MENSUAL                           <-ALL-FMTS
   23    +000012           06 DEPTNO               PIC X(5).                         <-ALL-FMTS
         +000013*  FORMATO SALIDA:FORMAT1     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000014*                                                                      <-ALL-FMTS
   24    +000015       05  FORMAT1-O     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   25    +000016           06 FORMAT1-O-INDIC.                                        <-ALL-FMTS
   26    +000017                07 IN01             PIC 1  INDIC 01.                  <-ALL-FMTS
         +000018*  FORMATO ENTRADA:ERRFMT     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000019*                                                                      <-ALL-FMTS
         +000020*      05  ERRFMT-I      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
         +000021*  FORMATO SALIDA:ERRFMT     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000022*                                                                      <-ALL-FMTS
   27    +000023       05  ERRFMT-O      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   28    +000024           06 ERRFMT-O-INDIC.                                         <-ALL-FMTS
   29    +000025                07 IN98             PIC 1  INDIC 98.                  <-ALL-FMTS
          000210
   30     000220 WORKING-STORAGE SECTION.
   31     000230 01  CURRENT-DATE.
   32     000240    05  CURR-YEAR                 PIC 9(2).
   33     000250    05  CURR-MONTH                PIC 9(2).
   34     000260    05  CURR-DAY                  PIC 9(2).
   35     000270 01  INDIC-AREA.  4
   36     000280    05  IN01                      PIC 1.
   37     000290       88  NEW-MONTH              VALUE B"1".  5
   38     000300    05  IN51                      PIC 1.
```

Figura 105 (Parte 1 de 3). Ejemplo de un programa con indicadores en el área de registro sin utilizar la expresión INDICATORS en la instrucción E/S–Programa fuente COBOL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC1          AS400SYS   96/07/04 11:56:12         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   39     000310       88  WANT-DAILY              VALUE B"1".
   40     000320    05  IN52                       PIC 1.
   41     000330       88  WANT-MONTHLY            VALUE B"1".
   42     000340    05  IN98                       PIC 1.
   43     000350       88  IO-ERROR                VALUE B"1".
   44     000360    05  IN99                       PIC 1.
   45     000370       88  NOT-END-OF-JOB          VALUE B"0".
   46     000380       88  END-OF-JOB              VALUE B"1".
          000390
   47     000400 PROCEDURE DIVISION.
   48     000410 DECLARATIVES.
          000420 DISPLAY-ERR-SECTION SECTION.
          000430     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
          000440 DISPLAY-ERR-PARAGRAPH.
   49     000450     SET IO-ERROR TO TRUE
   50     000460     MOVE CORR INDIC-AREA TO ERRFMT-O-INDIC
                 *** Elementos CORRESPONDING para la instrucción 000460:
                 ***       IN98
                 *** Fin de elementos CORRESPONDING para la instrucción 000460
   51     000470     WRITE DISP-REC FORMAT IS "ERRFMT"
          000480     END-WRITE
   52     000490     CLOSE DISPFILE.
   53     000500     STOP RUN.
          000510 END DECLARATIVES.
          000520
          000530 MAIN-PROGRAM SECTION.
          000540 MAINLINE.
   54     000550     OPEN I-O DISPFILE.
   55     000560     ACCEPT CURRENT-DATE FROM DATE.
   56     000570     SET NOT-END-OF-JOB TO TRUE.
   57     000580     PERFORM UNTIL END-OF-JOB
          000590
   58     000600         MOVE ZEROS TO INDIC-AREA  6
   59     000610         IF CURR-DAY = 01 THEN
   60     000620            SET NEW-MONTH TO TRUE  7
          000630         END-IF
   61     000640         MOVE CORR INDIC-AREA TO FORMAT1-O-INDIC  8
                 *** Elementos CORRESPONDING para la instrucción 000640:
                 ***       IN01
                 *** Fin de elementos CORRESPONDING para la instrucción 000640
   62     000650         WRITE DISP-REC FORMAT IS "FORMAT1"  9
          000660         END-WRITE
          000670
   63     000680         MOVE ZEROS TO INDIC-AREA
   64     000690         READ DISPFILE FORMAT IS "FORMAT1"  1
          000700         END-READ
   65     000710         MOVE CORR FORMAT1-I-INDIC TO INDIC-AREA  11
                 *** Elementos CORRESPONDING para la instrucción 000710:
                 ***       IN99
                 ***       IN51
                 ***       IN52
                 *** Fin de elementos CORRESPONDING para la instrucción 000710
   66     000720         IF WANT-DAILY THEN
   67     000730            CALL "DAILY" USING DEPTNO  12
          000740         ELSE
   68     000750            IF WANT-MONTHLY THEN
   69     000760               CALL "MONTHLY" USING DEPTNO
```

Figura 105 (Parte 2 de 3). Ejemplo de un programa con indicadores en el área de registro sin utilizar la expresión INDICATORS en la instrucción E/S–Programa fuente COBOL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC1          AS400SYS   96/07/04 11:56:12         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


          000770            END-IF
          000780         END-IF
          000790
          000800     END-PERFORM.
   70     000810     CLOSE DISPFILE.
   71     000820     STOP RUN.

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 105 (Parte 3 de 3). Ejemplo de un programa con indicadores en el área de registro sin utilizar la expresión INDICATORS en la instrucción E/S–Programa fuente COBOL

1 El atributo de área de indicadores por separado, SI, no está codificado en la cláusula ASSIGN. Como resultado, los indicadores forman parte del área de registro.

2 La instrucción COPY de formato 2 define los campos de datos y los indicadores en el área de registro.

3 Dado que los indicadores de archivo forman parte del área de registro, los indicadores de opción y de respuesta están definidos en el orden en que se utilizan en las DDS y los números de indicador se tratan como documentación.

4 Todos los indicadores utilizados por el programa está definidos con nombres con significado en las entradas de descripción de datos de WORKING-STORAGE SECTION. Los números de indicador se omiten porque no tienen ningún efecto.

5 Para cada indicador, se asocia un nombre de condición de nivel 88 con significado a un valor para dicho indicador.

6 Inicializar el nivel de grupo a ceros.

7 SE activa IN01 en WORKING-STORAGE SECTION si es el primer día del mes.

8 Los indicadores pertinentes a la salida de FORMAT1 se copian en el área de registro.

9 Se escribe FORMAT1 en la pantalla de la estación de trabajo con los datos y los valores de indicador del área de registro.

La expresión INDICATORS no es necesaria porque no hay ninguna área de indicadores por separado y los valores de indicador se han establecido en el área de registro mediante la instrucción MOVE CORRESPONDING anterior.

1 FORMAT1, incluido los datos y los indicadores, se lee de la pantalla.

11 Los indicadores de respuesta de FORMAT1 se copian del área de registro a las entradas de descripción de datos de WORKING-STORAGE SECTION.

12 Si se ha pulsado F5, se procesa una llamada a programa.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC2          AS400SYS   96/07/04 10:32:12        Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000010 IDENTIFICATION DIVISION.
    2     000020   PROGRAM-ID.     INDIC2.
          000030*    PROGRAMA DE EJEMPLO - ARCHIVO CON INDICADORES EN AREA DE REGISTRO
          000040
    3     000050 ENVIRONMENT DIVISION.
    4     000060   CONFIGURATION SECTION.
    5     000070   SOURCE-COMPUTER. IBM-AS400.
    6     000080   OBJECT-COMPUTER. IBM-AS400.
    7     000090   INPUT-OUTPUT SECTION.
    8     000100   FILE-CONTROL.
    9     000110     SELECT DISPFILE
   10     000120       ASSIGN TO WORKSTATION-DSPFILEX  1
   11     000130       ORGANIZATION IS TRANSACTION
   12     000140       ACCESS       IS SEQUENTIAL.
          000150
   13     000160 DATA DIVISION.
   14     000170   FILE SECTION.
   15     000180   FD  DISPFILE.
   16     000190   01  DISP-REC.
          000200       COPY DDS-ALL-FORMATS OF DSPFILEX.  2
   17    +000001       05  DSPFILEX-RECORD PIC X(8).                                    <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   19    +000005           06 FORMAT1-I-INDIC.                                        <-ALL-FMTS
   20    +000006                07 IN99             PIC 1  INDIC 99.  3              <-ALL-FMTS
         +000007*                          FIN DEL PROGRAMA                          <-ALL-FMTS
   21    +000008                07 IN51             PIC 1  INDIC 51.                 <-ALL-FMTS
         +000009*                          INFORME DIARIO                            <-ALL-FMTS
   22    +000010                07 IN52             PIC 1  INDIC 52.                 <-ALL-FMTS
         +000011*                          INFORME MENSUAL                           <-ALL-FMTS
   23    +000012           06 DEPTNO               PIC X(5).                         <-ALL-FMTS
         +000013*  FORMATO SALIDA:FORMAT1     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000014*                                                                      <-ALL-FMTS
   24    +000015       05  FORMAT1-O     REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   25    +000016           06 FORMAT1-O-INDIC.                                        <-ALL-FMTS
   26    +000017                07 IN01             PIC 1  INDIC 01.                 <-ALL-FMTS
         +000018*  FORMATO ENTRADA:ERRFMT     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000019*                                                                      <-ALL-FMTS
         +000020*      05  ERRFMT-I      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
         +000021*  FORMATO SALIDA:ERRFMT     DEL ARCH. DSPFILEX    DE LA BIBL. TESTLIB      <-ALL-FMTS
         +000022*                                                                      <-ALL-FMTS
   27    +000023       05  ERRFMT-O      REDEFINES DSPFILEX-RECORD.                   <-ALL-FMTS
   28    +000024           06 ERRFMT-O-INDIC.                                         <-ALL-FMTS
   29    +000025                07 IN98             PIC 1  INDIC 98.                 <-ALL-FMTS
          000210
   30     000220 WORKING-STORAGE SECTION.
   31     000230 01  CURRENT-DATE.
   32     000240    05  CURR-YEAR                 PIC 9(2).
   33     000250    05  CURR-MONTH                PIC 9(2).
   34     000260    05  CURR-DAY                  PIC 9(2).
          000270
   35     000280 77  IND-OFF                     PIC 1      VALUE B"0".
   36     000290 77  IND-ON                      PIC 1      VALUE B"1".
          000300
```

Figura 106 (Parte 1 de 2). Ejemplo de programa con indicadores en el área de registro y la expresión INDICATORS en las instrucciones de E/S–programa fuente COBOL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC2          AS400SYS   96/07/04 10:32:12        Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   37     000310 01  RESPONSE-INDICS.
   38     000320     05  END-OF-PROGRAM            PIC 1.  4
   39     000330     05  DAILY-REPORT              PIC 1.
   40     000340     05  MONTHLY-REPORT            PIC 1.
   41     000350 01  OPTION-INDICS.
   42     000360     05  NEW-MONTH                 PIC 1.
   43     000370 01  ERROR-INDICS.
   44     000380     05  IO-ERROR                  PIC 1.
          000390
   45     000400 PROCEDURE DIVISION.
   46     000410 DECLARATIVES.
          000420 DISPLAY-ERR-SECTION SECTION.
          000430     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
          000440 DISPLAY-ERR-PARAGRAPH.
   47     000450     MOVE IND-ON TO IO-ERROR
   48     000460     WRITE DISP-REC FORMAT IS "ERRFMT"
          000470                    INDICATORS ARE ERROR-INDICS
          000480     END-WRITE
   49     000490     CLOSE DISPFILE.
   50     000500     STOP RUN.
          000510 END DECLARATIVES.
          000520
          000530 MAIN-PROGRAM SECTION.
          000540 MAINLINE.
   51     000550     OPEN I-O DISPFILE.
   52     000560     ACCEPT CURRENT-DATE FROM DATE.
   53     000570     MOVE IND-OFF TO END-OF-PROGRAM.
   54     000580     PERFORM UNTIL END-OF-PROGRAM = IND-ON
   55     000590         MOVE ZEROS TO OPTION-INDICS
   56     000600         IF CURR-DAY = 01 THEN  5
   57     000610            MOVE IND-ON TO NEW-MONTH
          000620         END-IF
   58     000630         WRITE DISP-REC FORMAT IS "FORMAT1"  6
          000640                        INDICATORS ARE OPTION-INDICS
          000650         END-WRITE
          000660
   59     000670         MOVE ZEROS TO RESPONSE-INDICS
   60     000680         READ DISPFILE FORMAT IS "FORMAT1"  7
          000690                       INDICATORS ARE RESPONSE-INDICS  8
          000700         END-READ
   61     000710         IF DAILY-REPORT = IND-ON THEN
   62     000720            CALL "DAILY" USING DEPTNO  9
          000730         ELSE
   63     000740            IF MONTHLY-REPORT = IND-ON THEN
   64     000750               CALL "MONTHLY" USING DEPTNO
          000760            END-IF
          000770         END-IF
          000780
          000790     END-PERFORM
   65     000800     CLOSE DISPFILE.
   66     000810     STOP RUN.
          000820

                      * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 106 (Parte 2 de 2). Ejemplo de programa con indicadores en el área de registro y la expresión INDICATORS en las instrucciones de E/S–programa fuente COBOL

1 El atributo de área de indicadores por separado, SI, no está codificado en la cláusula ASSIGN.

2 La instrucción COPY de formato 2 define los campos de datos y los indicadores en el área de registro.

3 Dado que el archivo carece de área de indicadores por separado, los indicadores de respuesta y de opción están definidos en el orden en que se utilizan en las DDS y los números de indicador se tratan como documentación.

4 Todos los indicadores utilizados por el programa está definidos con nombres con significado en las entradas de descripción de datos de WORKING-STORAGE SECTION. Los números de indicador se omiten porque no tienen ningún efecto. Los indicadores deben definirse en el orden en que los necesita el archivo de pantalla.

5 SE activa IN01 en WORKING-STORAGE SECTION si es el primer día del mes.

6 Se escribe FORMAT1 en la pantalla de la estación de trabajo:

- La expresión INDICATORS hace que el contenido de la variable OPTION-INDICS se copie al principio del área de registro.
- Los datos y los valores de indicador se escriben en la pantalla de la estación de trabajo.

7 FORMAT1, incluido los datos y los indicadores, se lee de la pantalla de la estación de trabajo.

8 La expresión INDICATORS hace que los bytes se copien desde el principio del área de registro en RESPONSE-INDICS.

9 Si se ha pulsado F5, se procesa una llamada a programa.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* ARCHIVO DE PANTALLA PARA EJEMPLOS DE INDICADORES - INDICADORES EN AREA SI
     A* DSPFILE
     A                                      INDARA  1
     A          R FORMAT1                   CF01(99 'FIN DE PROGRAMA')
     A                                      CF05(51 'INFORME DIARIO')
     A                                      CF09(52 'INFORME MENSUAL')
     A*
     A                                   10 10'NÚM. DEPARTAMENTO:'
     A            DEPTNO         5   I 10 32
     A     01                            20 26'GENERAR INFORMES MENSUALES'
     A                                        DSPATR(BL)
     A*
     A                                   24 01'F5 = INFORME DIARIO'
     A                                   24 26'F9 = INFORME MENSUAL'
     A                                   24 53'F1 = FINALIZAR'
     A          R ERRFMT
     A     98                             6  5'ERROR ENTRADA-SALIDA'
```

Figura 107. Ejemplo de un programa con indicadores en un área de indicadores separada definida en WORKING-STORAGE con la instrucción COPY -- DDS

1 Se ha especificado la palabra clave INDARA; los indicadores se almacenan en un área de indicadores por separado, no en el área de registro. Con la excepción de esta especificación, las DDS de este archivo son las mismas que las mostradas en la Figura 104 en la página 413.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC3           AS400SYS   96/07/04 09:58:23         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INDIC3.
          000300*    PROGRAMA DE EJEMPLO - ARCHIVO CON AREA DE INDICADORES POR SEPARADO
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600 CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-AS400.
    6     000800 OBJECT-COMPUTER. IBM-AS400.
    7     000900 INPUT-OUTPUT SECTION.
    8     001000 FILE-CONTROL.
    9     001100     SELECT DISPFILE
   10     001200         ASSIGN TO WORKSTATION-DSPFILE-SI  1
   11     001300         ORGANIZATION IS TRANSACTION
   12     001400         ACCESS IS SEQUENTIAL.
          001500
   13     001600 DATA DIVISION.
   14     001700 FILE SECTION.
   15     001800 FD  DISPFILE.
   16     001900 01  DISP-REC.
          002000     COPY DDS-ALL-FORMATS OF DSPFILE.  2
   17    +000001       05  DSPFILE-RECORD PIC X(5).                                    <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILE-RECORD.                    <-ALL-FMTS
   19    +000005           06 DEPTNO               PIC X(5).                         <-ALL-FMTS
         +000006*  FORMATO SALIDA:FORMAT1     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000007*                                                                      <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES DSPFILE-RECORD.                    <-ALL-FMTS
         +000009*  FORMATO ENTRADA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000010*                                                                      <-ALL-FMTS
         +000011*      05  ERRFMT-I      REDEFINES DSPFILE-RECORD.                    <-ALL-FMTS
         +000012*  FORMATO SALIDA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000013*                                                                      <-ALL-FMTS
         +000014*      05  ERRFMT-O      REDEFINES DSPFILE-RECORD.                    <-ALL-FMTS
          002100
   20     002200 WORKING-STORAGE SECTION.
   21     002300 01  CURRENT-DATE.
   22     002400    05  CURR-YEAR                 PIC 9(2).
   23     002500    05  CURR-MONTH                PIC 9(2).
   24     002600    05  CURR-DAY                  PIC 9(2).
          002700
   25     002800    77 IND-OFF                    PIC 1 VALUE B"0".
   26     002900    77 IND-ON                     PIC 1 VALUE B"1".
          003000
   27     003100 01  DISPFILE-INDICS.
          003200     COPY DDS-ALL-FORMATS-INDIC OF DSPFILE.  3                                  96/11/08
   28    +000001       05  DSPFILE-RECORD.                                             <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000003*                                                                      <-ALL-FMTS
   29    +000004           06 FORMAT1-I-INDIC.                                        <-ALL-FMTS
   30    +000005                07 IN51            PIC 1  INDIC 51.  4                <-ALL-FMTS
         +000006*                          INFORME DIARIO                            <-ALL-FMTS
   31    +000007                07 IN52            PIC 1  INDIC 52.                   <-ALL-FMTS
         +000008*                          INFORME MENSUAL                           <-ALL-FMTS
   32    +000009                07 IN99            PIC 1  INDIC 99.                   <-ALL-FMTS
```

Figura 108 (Parte 1 de 2). Listado COBOL con indicadores en un área de indicadores por separado

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC3           AS400SYS   96/07/04 09:58:23         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


        +000010*                          FIN DEL PROGRAMA                                    <-ALL-FMTS
        +000011* FORMATO SALIDA:FORMAT1     DEL ARCH. DSPFILE DE LA BIBL. TESTLIB             <-ALL-FMTS
        +000012*                                                                              <-ALL-FMTS
   33   +000013           06 FORMAT1-O-INDIC.                                                 <-ALL-FMTS
   34   +000014                07 IN01            PIC 1  INDIC 01.                          <-ALL-FMTS
        +000015*  FORMATO ENTRADA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB         <-ALL-FMTS
        +000016*                                                                              <-ALL-FMTS
        +000017*          06 ERRFMT-I-INDIC.                                                  <-ALL-FMTS
        +000018* FORMATO SALIDA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB          <-ALL-FMTS
        +000019*                                                                              <-ALL-FMTS
   35   +000020           06 ERRFMT-O-INDIC.                                                  <-ALL-FMTS
   36   +000021                07 IN98            PIC 1  INDIC 98.                          <-ALL-FMTS
         003300
   37    003400 PROCEDURE DIVISION.
   38    003500 DECLARATIVES.
         003600 DISPLAY-ERR-SECTION SECTION.
         003700     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
         003800 DISPLAY-ERR-PARAGRAPH.
   39    003900     MOVE IND-ON TO IN98 IN ERRFMT-O-INDIC
   40    004000     WRITE DISP-REC FORMAT IS "ERRFMT"
         004100                    INDICATORS ARE ERRFMT-O-INDIC
         004200     END-WRITE
   41    004300     CLOSE DISPFILE.
   42    004400     STOP RUN.
         004500 END DECLARATIVES.
         004600
         004700 MAIN-PROGRAM SECTION.
         004800 MAINLINE.
         004900
   43    005000     OPEN I-O DISPFILE.
   44    005100     ACCEPT CURRENT-DATE FROM DATE.
   45    005200     MOVE IND-OFF TO IN99 IN FORMAT1-I-INDIC.
   46    005300     PERFORM UNTIL IN99 IN FORMAT1-I-INDIC = IND-ON
         005400
   47    005500         MOVE ZEROS TO FORMAT1-O-INDIC
   48    005600         IF CURR-DAY = 01 THEN
   49    005700            MOVE IND-ON TO IN01 IN FORMAT1-O-INDIC  5
         005800         END-IF
   50    005900         WRITE DISP-REC FORMAT IS "FORMAT1"
         006000                        INDICATORS ARE FORMAT1-O-INDIC  6
         006100         END-WRITE
         006200
   51    006300         MOVE ZEROS TO FORMAT1-I-INDIC
   52    006400         READ DISPFILE FORMAT IS "FORMAT1"
         006500                       INDICATORS ARE FORMAT1-I-INDIC  7
         006600         END-READ
   53    006700         IF IN51 IN FORMAT1-I-INDIC = IND-ON THEN
   54    006800             CALL "DAILY" USING DEPTNO  8
         006900         ELSE
   55    007000             IF IN52 IN FORMAT1-I-INDIC = IND-ON THEN
   56    007100                 CALL "MONTHLY" USING DEPTNO
         007200             END-IF
         007300         END-IF
         007400
         007500     END-PERFORM
   57    007600     CLOSE DISPFILE.
   58    007700     STOP RUN.
         007800

                       * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 108 (Parte 2 de 2). Listado COBOL con indicadores en un área de indicadores por separado

1 El atributo de área de indicadores por separado, SI, está especificado en la cláusula ASSIGN.

2 La instrucción COPY de formato 2 genera descripciones de datos en el área de registro sólo para los campos de datos. No se generan entradas de descripción de datos para los indicadores porque se ha especificado un área de indicadores por separado para el archivos.

3 La instrucción COPY de formato 2, con el atributo INDICATOR, INDIC, define las entradas de descripción de datos en WORKING-STORAGE SECTION para todos los indicadores utilizados en las DDS del formato de registro del archivo.

4 Dado que el archivo tiene un área de indicadores por separado, los números de indicador utilizados en las entradas de descripción de datos no se tratan como documentación.

5 Se activa IN01 en el área de indicadores por separado de FORMAT1 si es el primer día del mes.

6 Se requiere la expresión INDICATORS para enviar los valores de indicador a la pantalla de la estación de trabajo.

7 Se requiere la expresión INDICATORS para recibir valores de indicador de la pantalla de la estación de trabajo. Si ha pulsado F5, se activa IN51.

8 Si se ha activado IN51, se procesa una llamada a programa.

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC4          AS400SYS   96/07/04 11:23:53         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     INDIC4.
          000300*    PROGRAMA DE EJEMPLO
          000400*    ARCHIVO CON AREA DE INDICADORES POR SEPARADO EN WORKING STORAGE
          000500
    3     000600 ENVIRONMENT DIVISION.
    4     000700 CONFIGURATION SECTION.
    5     000800 SOURCE-COMPUTER. IBM-AS400.
    6     000900 OBJECT-COMPUTER. IBM-AS400.
    7     001000 INPUT-OUTPUT SECTION.
    8     001100 FILE-CONTROL.
    9     001200     SELECT DISPFILE
   10     001300         ASSIGN TO WORKSTATION-DSPFILE-SI  1
   11     001400         ORGANIZATION IS TRANSACTION
   12     001500         ACCESS IS SEQUENTIAL.
          001600
   13     001700 DATA DIVISION.
   14     001800 FILE SECTION.
   15     001900 FD  DISPFILE.
   16     002000 01  DISP-REC.
          002100     COPY DDS-ALL-FORMATS OF DSPFILE.  2
   17    +000001       05  DSPFILE-RECORD PIC X(5).                                    <-ALL-FMTS
         +000002*  FORMATO ENTRADA:FORMAT1     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000003*                                                                     <-ALL-FMTS
   18    +000004       05  FORMAT1-I     REDEFINES DSPFILE-RECORD.                   <-ALL-FMTS
   19    +000005           06 DEPTNO               PIC X(5).                         <-ALL-FMTS
         +000006*  FORMATO SALIDA:FORMAT1     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000007*                                                                     <-ALL-FMTS
         +000008*      05  FORMAT1-O     REDEFINES DSPFILE-RECORD.                   <-ALL-FMTS
         +000009*  FORMATO ENTRADA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000010*                                                                     <-ALL-FMTS
         +000011*      05  ERRFMT-I      REDEFINES DSPFILE-RECORD.                   <-ALL-FMTS
         +000012*  FORMATO SALIDA:ERRFMT     DEL ARCH. DSPFILE    DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000013*                                                                     <-ALL-FMTS
         +000014*      05  ERRFMT-O      REDEFINES DSPFILE-RECORD.                   <-ALL-FMTS
          002200
   20     002300 WORKING-STORAGE SECTION.
   21     002400 01  CURRENT-DATE.
   22     002500    05  CURR-YEAR                 PIC 9(2).
   23     002600    05  CURR-MONTH                PIC 9(2).
   24     002700    05  CURR-DAY                  PIC 9(2).
          002800
   25     002900 01  INDIC-AREA.
   26     003000    05 INDIC-TABLE  OCCURS 99      PIC 1    INDICATOR 1.  3
   27     003100      88  IND-OFF                          VALUE B"0".
   28     003200      88  IND-ON                           VALUE B"1".
          003300
   29     003400 01  DISPFILE-INDIC-USAGE.
   30     003500    05  IND-NEW-MONTH              PIC 9(2) VALUE 01.
   31     003600    05  IND-DAILY                  PIC 9(2) VALUE 51.  4
   32     003700    05  IND-MONTHLY                PIC 9(2) VALUE 52.
   33     003800    05  IND-IO-ERROR               PIC 9(2) VALUE 98.
   34     003900    05  IND-EOJ                    PIC 9(2) VALUE 99.
          004000
   35     004100 PROCEDURE DIVISION.
```

Figura 109 (Parte 1 de 2). Ejemplo de programa con indicadores en un área de indicadores por separado, definida en una tabla en WORKING-STORAGE

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/INDIC4           AS400SYS   96/07/04 11:23:53         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   36     004200 DECLARATIVES.
          004300 DISPLAY-ERR-SECTION SECTION.
          004400     USE AFTER STANDARD EXCEPTION PROCEDURE ON DISPFILE.
          004500 DISPLAY-ERR-PARAGRAPH.
   37     004600     SET IND-ON (IND-IO-ERROR) TO TRUE
   38     004700     WRITE DISP-REC FORMAT IS "ERRFMT"
          004800                     INDICATORS ARE INDIC-AREA                                        96/11/08
          004900     END-WRITE
   39     005000     CLOSE DISPFILE.
   40     005100     STOP RUN.
          005200 END DECLARATIVES.
          005300
          005400 MAIN-PROGRAM SECTION.
          005500 MAINLINE.
   41     005600     OPEN I-O DISPFILE.
   42     005700     ACCEPT CURRENT-DATE FROM DATE.
   43     005800     SET IND-OFF (IND-EOJ) TO TRUE.
   44     005900     PERFORM UNTIL IND-ON (IND-EOJ)
          006000
   45     006100         MOVE ZEROS TO INDIC-AREA
   46     006200         IF CURR-DAY = 01 THEN
   47     006300            SET IND-ON (IND-NEW-MONTH) TO TRUE  5
          006400         END-IF
   48     006500         WRITE DISP-REC FORMAT IS "FORMAT1"
          006600                        INDICATORS ARE INDIC-AREA  6                                   96/11/08
          006700         END-WRITE
          006800
   49     006900         READ DISPFILE FORMAT IS "FORMAT1"
          007000                        INDICATORS ARE INDIC-AREA  7                                   96/11/08
          007100         END-READ
   50     007200         IF IND-ON (IND-DAILY) THEN
   51     007300             CALL "DAILY" USING DEPTNO  8
          007400         ELSE
   52     007500             IF IND-ON (IND-MONTHLY) THEN
   53     007600                 CALL "MONTHLY" USING DEPTNO
          007700             END-IF
          007800         END-IF
          007900
          008000     END-PERFORM
   54     008100     CLOSE DISPFILE.
   55     008200     STOP RUN.
          008300

                      * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 109 (Parte 2 de 2). Ejemplo de programa con indicadores en un área de indicadores por separado, definida en una tabla en WORKING-STORAGE

1 El atributo de área de indicadores por separado, SI, está especificado en la cláusula ASSIGN.

2 La instrucción COPY de formato 2 genera campos en el área de registro sólo para los campos de datos.

3 Se define una tabla de 99 datos booleanos en WORKING-STORAGE SECTION. La cláusula INDICATOR de esta entrada de descripción de datos hace que estos datos se asocien con los indicadores de 1 al 99, respectivamente. El uso de una tabla así puede dar como resultado un mejor rendimiento en comparación con el uso de un grupo con varias entradas subordinadas para indicadores individuales.

4 Se define una serie de datos en WORKING-STORAGE SECTION para dar nombres de subíndice con significado con los que hacer referencia a la tabla de indicadores. No se requiere el uso de tales datos.

5 Se activa INDIC-TABLE (01) en el área de indicadores por separado de FORMAT1 si es el primer día del mes.

6 Se requiere la expresión INDICATOR para enviar los valores de indicador a la pantalla de la estación de trabajo.

7 Se requiere la expresión INDICATOR para recibir valores de indicador de la pantalla de la estación de trabajo. Si se ha pulsado F5, se activará INDIC-TABLE (51).

8 Si se ha activado INDIC-TABLE (51), se llama al programa DAILY.

Utilización de archivos de transacción de subarchivo

Un **subarchivo** es un grupo de registros leídos de un dispositivo de pantalla o escritos en él. El programa procesa los registros de uno en uno, pero el sistema operativo y la estación de trabajo envían y reciben bloques de registros. Si se transmiten más registros de los que se pueden mostrar en la pantalla a la vez, el operador de la estación de trabajo puede avanzar o retroceder páginas por los bloques de registro sin devolver el control al programa

Los subarchivos constituyen una forma cómoda de leer y escribir un número elevado de registros similares a y desde las pantallas. Son archivos de pantalla a cuyos registros pueden accederse de forma secuencial o aleatoria por valor de clave relativa.

Supongamos, por ejemplo, que desea visualizar todos los clientes que el año pasado han realizado un gasto superior a 500.000 pts en la compañía. Puede hacer una consulta a la base de datos y obtener los nombres de todos estos clientes y colocarlos en un archivo especial (el subarchivo) realizando operaciones WRITE SUBFILE en el subarchivo. Cuando lo haya hecho, puede escribir todo el contenido del subarchivo en la pantalla realizando una operación WRITE en el registro de control de subarchivo. A continuación, puede leer la lista de clientes modificada por el usuario utilizando una operación READ en el registro de control de subarchivo y recuperar posteriormente los registros individuales del subarchivo con operaciones READ SUBFILE.

Los subarchivos pueden especificarse en las DDS de un archivo de pantalla para que el usuario pueda manejar múltiples registros del mismo tipo en una pantalla. Vea la Figura 110 en la página 427 para tener un ejemplo de una pantalla de subarchivo.

Los formatos de registro a incluir en un subarchivo se especifican en la DDS del archivo. El número de registros que puede contener un subarchivo también debe especificarse en las DDS. Un archivo puede contener más de un subarchivo; sin embargo, sólo puede haber activos de forma concurrente 12 subarchivos para un dispositivo.

Definición de un subarchivo con especificaciones de descripción de datos

Las DDS de un subarchivo constan de dos formatos de registro: un formato de registro de subarchivo y un formato de registro de control de subarchivo.

El formato de registro de subarchivo contiene las descripciones de campo de los registros del subarchivo. Las especificaciones del formato de registro de subar-

chivo en READ, WRITE o REWRITE hacen que se procese el registro de subarchivo especificado, pero no afectan directamente a los datos visualizados.

La especificación del formato de registro de control de subarchivo en la instrucción READ o WRITE hacen que las operaciones físicas de lectura, escritura o configuración de un subarchivo tengan lugar.La Figura 111 en la página 429 muestra un ejemplo de las DDS de un formato de registro de subarchivo y de un formato de registro de control de subarchivo.

Para obtener una descripción de cómo pueden visualizarse los registros de un subarchivo así como una descripción de las palabras clave que pueden especificarse para un subarchivo, consulte las publicaciones *Gestión de datos* y *DDS Reference*.

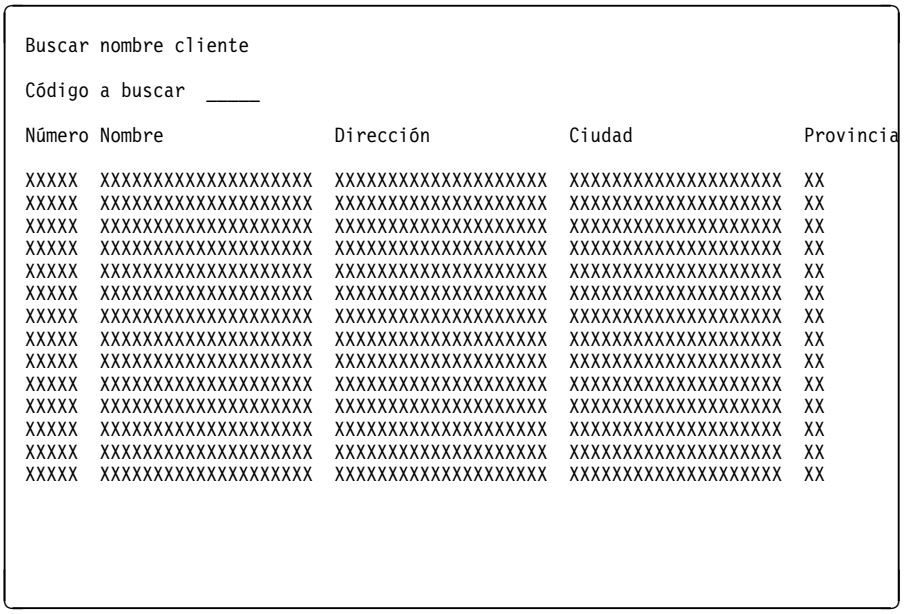


Figura 110. Pantalla de subarchivo

Utilización de subarchivos en un archivo de pantalla

Para utilizar un subarchivo en un archivo de pantalla de un programa ILE COBOL/400, debe especificar la expresión SUBFILE con la operación de entrada/salida. Las operaciones de subarchivo válidas son:

- READ SUBFILE nombre-archivo RECORD
- WRITE SUBFILE nombre-registro
- REWRITE SUBFILE nombre-registro.

Los subarchivos pueden procesarse secuencialmente con la instrucción READ SUBFILE NEXT MODIFIED o aleatoriamente especificando un valor de clave relativa. Un valor de clave relativa es un número sin signo que puede utilizar directamente el sistema para localizar un registro en un archivo.

El archivo TRANSACTION debe ser un archivo descrito externamente. En ILE COBOL/400, el acceso al subarchivo se realiza con un número relativo de registro, excepto cuando de utiliza READ SUBFILE NEXT MODIFIED. Si se utilizan las expresiones SUBFILE con un archivo TRANSACTION, la instrucción SELECT de

Environment Division debe incluir ACCESS MODE IS DYNAMIC y debe especificar RELATIVE KEY.

Si determina que un archivo de pantalla adquiere más de un dispositivo de pantalla, habrá un subarchivo separado para cada dispositivo de pantalla individual. Si se ha creado un subarchivo para un dispositivo de pantalla en concreto adquirido por un archivo TRANSACTION, todas las operaciones de entrada que hagan referencia a un formato de registro del subarchivo se realizan en el subarchivo perteneciente a dicho dispositivo. Las operaciones que hagan referencia a un nombre de formato de registro que no esté designado como subarchivo se procesan como operaciones de entrada/salida directamente en el dispositivo de pantalla.

Los usos más habituales de los subarchivos están resumidos en la Tabla 21.

Tabla 21. Utilizaciones de los subarchivos

Uso	**Significado**
Sólo visualización	El usuario de la estación de trabajo revisa la pantalla.
Visualización con selección	El usuario solicita más información sobre uno de los elementos de la pantalla.
Modificación	El usuario modifica uno o más de los registros.
Sólo entrada (sin comprobación de validez)	Se utiliza un subarchivo para una función de entrada de datos.
Sólo entrada (con comprobación de validez)	Se utiliza un subarchivo para una función de entrada de datos y, además, se comprueban los registros.
Combinación de tareas	Se puede utilizar un subarchivo como pantalla con modificación.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* ESTE ES EL ARCHIVO DE DISPOSITIVO DE PANTALLA DE PAYUPDT -- PAYUPDTD
     A* ACTUALIZACION INTERACTIVA DE PAGOS DE CUENTAS POR COBRAR
     A*
     A
     A          R SUBFILE1                  SFL  1
     A                                      TEXT('SUBARCHIVO DE PAGOS CLIENTE'
     A*
     A            ACPPMT         4A  I  5  4TEXT('ACEPTAR PAGO')
     A                                    2   VALUES('*YES' '*NO')  3
     A  51                                  DSPATR(RI MDT)
     A N51                                  DSPATR(ND PR)
     A*
     A            CUST           5   B  5 15TEXT('NÚMERO CLIENTE')
     A  52                          4       DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            AMPAID         8 02B  5 24TEXT('IMPORTE PAGADO')
     A                                      CHECK(FE)  5
     A                                      AUTO(RAB)  6
     A                                      CMP(GT 0)  7
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            ECPMSG        31A  O  5 37TEXT('MENSAJE DE EXCEPCIÓN')
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(BL)
     A*
     A            OVRPMT         8Y 2O  5 70TEXT('DINERO PAGADO DE MÁS')
     A                                      EDTCDE(1)  8
     A  55                                  DSPATR(BL)  9
     A N56                                  DSPATR(ND)
     A*
     A            STSCDE         1A  H      TEXT('CÓDIGO DE ESTADO')
     A          R CONTROL1                  TEXT('SUBARCHIVO DE CONTROL')
     A                                      SFLCTL(SUBFILE1)  1
     A                                      SFLSIZ(17)  11
     A                                      SFLPAG(17)  12
     A  61                                  SFLCLR  13
     A  62                                  SFLDSP  14
     A  62                                  SFLDSPCTL  15
     A                                      OVERLAY
     A                                      LOCK  16
     A*
     A                                      HELP(99 'TECLA AYUDA')  17
     A                                      CA12(98 'FIN ACTUALIZACIÓN PAGOS')
     A                                      CA11(97 'IGNORAR ENTRADA')
     A*                                      18
     A  99                                  SFLMSG(' F11 - IGNORAR ENTRADA NO +
     A                                                 VÁLIDA F12 - FIN DE +
     A                                      ACTUALIZACIÓN PAGOS')
     A*
     A                                   1  2'SOLICITUD ACTUALIZACIÓN PAGOS CLIENTES'
     A                                   1 65'FECHA'
     A                                   1 71DATE EDTCDE(Y)
     A  63                               3  2'ACEPTAR'
     A  63                               4  2'PAGO'
     A                                   3 14'CLIENTE'
     A                                   3 26'IMPORTE'
     A  64                               3 37'MENSAJE DE EXCEPCIÓN'
     A*
     A          R MESSAGE1                  TEXT('REGISTRO DE MENSAJES')
     A                                      OVERLAY
     A                                      LOCK
     A*
     A  71                              24  2' VALORES DE ACEPTAR PAGO: (*NO *YES)
                                            DSPATR(RI)
```

Figura 111. Especificaciones de descripción de datos para un formato de registro de subarchivo

Las especificaciones de descripción de datos (DDS) de un formato de registro de subarchivo describen los registros del subarchivo:

1 La palabra clave SFL identifica el formato de registro como subarchivo.

2 Las entradas de línea y posición identifican la ubicación de los campos en la pantalla.

3 La palabra clave VALUES especifica que el usuario sólo puede especificar *YES o *NO como valores para el campo ACPPMT.

4 Las entradas de utilización definen si el campo mencionado ha de ser de salida (O), de entrada (I), de salida/entrada (B) u oculto (H).

5 La entrada CHECK(FE) especifica que el usuario no puede saltar al siguiente campo de entrada sin pulsar una de las teclas de salida de campo.

6 La entrada AUTO(RAB) especifica que los datos entrados en el campo AMPAID han de justificarse por la derecha de forma automática y que los caracteres iniciales de relleno son blancos.

7 La entrada CMP(GT 0) especifica que los datos entrados para el campo AMPAID han de compararse con cero para asegurarse que el valor es mayor que cero.

8 La palabra clave EDTCDE especifica la edición deseada para el campo de salida OVRPMT. EDTCDE(1) indica que el campo OVRPMT ha de imprimirse con comas, coma decimal y sin signo. Además, se imprimirá un saldo cero y se suprimirán los ceros iniciales.

9 Se utiliza la palabra clave DSPATR para especificar los atributos de visualización del campo mencionado cuando el estado de indicador correspondiente sea cierto. Los atributos especificados son:

- BL (parpadeo)
- RI (contraste invertido)
- PR (protegido)
- MDT (establecer el identificador de datos modificados)
- ND (no visualización).

El formato de registro de control de subarchivo define los atributos del subarchivo, el campo de entrada de búsqueda, las constantes y las teclas de mandatos. Las palabras clave utilizadas indican lo siguiente:

1 SFLCTL identifica este registro como registro de control de subarchivo y menciona el registro de subarchivo asociado (SUBFILE1).

11 SFLSIZ indica el número total de registros a incluir en el subarchivo (17).

12 SFLPAG indica el número total de registros de una página (17).

13 SFLCLR indica cuándo se ha de borrar el subarchivo (cuando esté activado el indicador 61).

14 SFLDSP indica cuándo se ha de visualizar el subarchivo (cuando esté activado el indicador 62).

15 SFLDSPCTL indica cuándo se ha de visualizar el registro de control de subarchivo (cuando esté activado el indicador 62).

16 La palabra clave LOCK impide que el usuario de la estación de trabajo utilice el teclado cuando se visualiza inicialmente el formato de registro CONTROL1.

17 HELP permite al usuario pulsar la tecla Ayuda y activa el indicador 99.

18 SFLMSG identifica la constante como un mensaje que se visualiza si se activa el indicador 99.

Además de la información de control, el formato de registro de control de subarchivo define las constantes que han de utilizarse como cabeceras de columna para el formato de registro de subarchivo. Vea la Figura 111 en la página 429 para tener un ejemplo del formato de registro de control de subarchivo.

Acceso a archivos de un solo dispositivo y a archivos de múltiples dispositivos

Un **archivo de un solo dispositivo** es un archivo de dispositivo creado con un solo dispositivo de programa definido para él. Los archivos de impresora, de disquete y de cinta son archivos de un solo dispositivo. Los archivos de pantalla y los archivos ICF (función de comunicaciones intersistemas) creados con un máximo de un programa de dispositivo también son archivos de un solo dispositivo.

Un **archivo de múltiples dispositivos** es un archivo pantalla o un archivo ICF (función de comunicaciones intersistemas). Un archivo de múltiples dispositivos puede adquirir más de un dispositivo de programa. Si desea tener un ejemplo del uso de los archivos de múltiples dispositivos, vea la Figura 115 en la página 433.

Un archivo de pantalla puede tener múltiples dispositivos de programa si el parámetro MAXDEV del mandato CRTDSPF es mayor que 1. Si especifica *NONE para el parámetro DEV de este mandato, debe proporcionar el nombre de un dispositivo de pantalla *antes de* utilizar cualquiera de los campos relacionados con el archivo.

Para obtener más información sobre la manera de crear y de utilizar un archivo de pantalla, consulte la publicación *Gestión de datos*.

Los archivos ICF pueden tener múltiples dispositivos de programa si el parámetro MAXPGMDEV del mandato CRTICFF es mayor que 1. Para obtener más información sobre la manera de crear y de utilizar archivos ICF, consulte la publicación *ICF Programming*.

ILE COBOL/400 determina durante la ejecución si un archivo es de un solo dispositivo o de múltiples tomando como base si el archivo es *capaz* de tener múltiples dispositivos. El número real de dispositivos adquiridos no afecta a si un archivo está considerado de un solo dispositivo o de múltiples. La determinación de si un archivo es de un solo dispositivo o de múltiples *no* se realiza durante la compilación; dicha determinación está basada en la descripción actual del archivo ICF o de pantalla.

En el caso de los archivos de múltiples dispositivos, si se ha de utilizar un dispositivo de programa concreto en una instrucción de E/S, dicho dispositivo lo especifica la expresión TERMINAL. La expresión TERMINAL también puede especificarse para un archivo de un solo dispositivo.

Las siguientes páginas contienen un ejemplo que ilustra el uso de archivos de múltiples dispositivos. El programa utiliza un archivo de pantalla y está pensado para ejecutarse en modalidad por lotes. Adquiere terminales e invita a los mismos con una pantalla de inicio de sesión. Una vez se ha invitado a los terminales, se les sondea. Si nadie inicia una sesión antes de que expire el tiempo de espera, el programa finaliza. Si entra una contraseña válida, se le permite actualizar un archivo de empleados llamando a otro programa ILE COBOL/400. Una vez finalizada la actualización, se invita de nuevo al dispositivo y se sondean otra vez los terminales.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A*            ESTE ES EL ARCHIVO DE PANTALLA DE VARIOS DISPOSITIVOS
     A*
     A*                  DDS PARA EL ARCHIVO MIXTO MULT
     A*
     A
     A          R SIGNON                    INVITE  1
     A                                 O  5 20'                    '
     A                                        DSPATR(RI)
     A                                 O  6 20'  '
     A                                        DSPATR(RI)
     A                                 O  6 38'  '
     A                                        DSPATR(RI)
     A                                 O  7 20'  '
     A                                        DSPATR(RI)
     A                                 O  7 27'M D F'
     A                                        DSPATR(HI BL)
     A                                 O  7 38'  '
     A                                        DSPATR(RI)
     A                                 O  9 20'                    '
     A                                        DSPATR(RI)
     A                                 O 20 20'PLEASE LOGON'
     A                                        DSPATR(HI)
     A            PASSWORD      10A  I 20 43DSPATR(PC ND)
     A            WRONG         20A  O 21 43
     A
     A          R UPDATE
     A                                 O  3  5'ACTUALIZACIÓN ARCHIVO EMPLEADOS'
     A                                        DSPATR(BL)
     A                                 O  7  5'ENTRE EL NÚMERO DE EMPLEADO A +
     A                                          ACTUALIZAR'
     A            NUM            7A  I  7 44DSPATR(RI PC)
     A
     A          R EMPLOYEE
     A                                 O  3  5'NÚMERO EMPLEADO'
     A            NUM            7A  B  3 25DSPATR(PC)
     A                                 O  5  5'NOMBRE EMPLEADO'
     A            NAME          30A  B  5 25DSPATR(PC)
     A                                 O  7  5'DIRECCIÓN EMPLEADO'
     A                                 O  9  5'CALLE'
     A            STREET        30A  B  9 25DSPATR(PC)
     A                                 O 11  5'PISO'
     A            APTNO          5A  B 11 25DSPATR(PC)
     A                                 O 13  5'CIUDAD'
     A            CITY          20A  B 13 25DSPATR(PC)
     A                                 O 15  5'PROVINCIA'
     A            PROV          20A  B 15 25DSPATR(PC)
     A
     A          R RECOVERY
     A                                 O  3  5'EL NÚMERO DE EMPLEADO ENTRADO NO
     A                                          ES VÁLIDO'
     A                                 O  6  5'ENTRE Y PARA VOLVER A INTENTARLO'
     A                                 O  8  5'ENTRE N PARA SALIR'
     A            ANSWER         1X  I 10  5DSPATR(RI PC)
     A                                       VALUES('Y' 'N')
```

Figura 112. Ejemplo del uso de archivos de múltiples dispositivos -- archivo de pantalla

1 El formato SIGNON tiene asociada la palabra clave INVITE. Esto significa que, si se utiliza el formato SIGNON en una instrucción WRITE, se invitará al dispositivo al que esté escribiendo.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A** DDS PARA EL ARCHIVO FISICO PASSWORD
     A*
     A*
     A                                       UNIQUE
     A          R PASSWORDS
     A            PASSKEY       10
     A            PASSWORD      10
     A          K PASSKEY
     A
```

Figura 113. Ejemplo del uso de archivos de múltiples dispositivos -- archivo físico PASSWORD

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A*  DDS PARA EL ARCHIVO FISICO TERM
     A*  QUE CONTIENE LA LISTA DE TERMINALES
     A*
     A
     A          R TERM
     A            TERM          10
```

Figura 114. Ejemplo del uso de archivos de múltiples dispositivos -- archivo físico TERM

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPMDF          AS400SYS   96/07/04 11:38:49         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     SAMPMDF.
          000300
          000400*****************************************************************
          000500* EL SIGUIENTE PROGRAMA ES UNA DEMOSTRACION DE ALGUNAS FUNCIONES *
          000600* DISPONIBLES CON SOPORTE DE ARCHIVOS DE VARIOS DISPOSITIVOS.    *
          000700*****************************************************************
          000800
    3     000900 ENVIRONMENT DIVISION.
    4     001000 CONFIGURATION SECTION.
    5     001100 SOURCE-COMPUTER. IBM-AS400.
    6     001200 OBJECT-COMPUTER. IBM-AS400.
    7     001300 SPECIAL-NAMES.  ATTRIBUTE-DATA IS ATTR.  1
    9     001400 INPUT-OUTPUT SECTION.
   10     001500 FILE-CONTROL.
   11     001600     SELECT MULTIPLE-FILE
   12     001700            ASSIGN TO WORKSTATION-MULT
   13     001800            ORGANIZATION IS TRANSACTION  2
   14     001900            ACCESS MODE IS SEQUENTIAL
   15     002000            FILE STATUS IS MULTIPLE-FS1, MULTIPLE-FS2  3
   16     002100            CONTROL-AREA IS MULTIPLE-CONTROL-AREA.  4
          002200
   17     002300     SELECT TERMINAL-FILE
   18     002400            ASSIGN TO DATABASE-TERM
   19     002500            ORGANIZATION IS SEQUENTIAL
   20     002600            ACCESS IS SEQUENTIAL
   21     002700            FILE STATUS IS TERMINAL-FS1.
          002800
   22     002900     SELECT PASSWORD-FILE
   23     003000            ASSIGN TO DATABASE-PASSWORD
   24     003100            ORGANIZATION IS INDEXED
   25     003200            RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
   26     003300            ACCESS MODE IS RANDOM
   27     003400            FILE STATUS IS PASSWORD-FS1.
          003500
   28     003600     SELECT PRINTER-FILE
   29     003700            ASSIGN TO PRINTER-QPRINT.
          003800
   30     003900 DATA DIVISION.
   31     004000 FILE SECTION.
   32     004100 FD  MULTIPLE-FILE.
   33     004200 01  MULTIPLE-REC.
          004200                  COPY DDS-SIGNON OF MULT.  5
   34    +000001       05  MULT-RECORD PIC X(20).                                    SIGNON
         +000002*  FORMATO ENTRADA:SIGNON     DEL ARCH. MULT    DE LA BIBL. TESTLIB  SIGNON
         +000003*                                                                   SIGNON
   35    +000004       05  SIGNON-I      REDEFINES MULT-RECORD.                     SIGNON
   36    +000005           06 PASSWORD              PIC X(10).  6                   SIGNON
         +000006* FORMATO SALIDA:SIGNON     DEL ARCH. MULT    DE LA BIBL. TESTLIB  SIGNON
         +000007*                                                                   SIGNON
   37    +000008       05  SIGNON-O      REDEFINES MULT-RECORD.                     SIGNON
   38    +000009           06 WRONG                 PIC X(20).                     SIGNON
          004300
   39     004400 FD  TERMINAL-FILE.
   40     004500 01  TERMINAL-REC.
```

Figura 115 (Parte 1 de 5). Listado fuente ILE COBOL/400 para soporte de archivos de múltiples dispositivos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPMDF          AS400SYS   96/07/04 11:38:49         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         004500                  COPY DDS-ALL-FORMATS OF TERM.
   41    +000001       05  TERM-RECORD PIC X(10).                                    <-ALL-FMTS
         +000002*    FORMATO E-S:TERM       DEL ARCH. TERM       DE LA BIBL. TESTLIB  <-ALL-FMTS
         +000003*                                                                    <-ALL-FMTS
   42    +000004       05  TERM          REDEFINES TERM-RECORD.                     <-ALL-FMTS
   43    +000005           06 TERM                  PIC X(10).                      <-ALL-FMTS
         004600
   44    004700 FD  PASSWORD-FILE.
   45    004800 01  PASSWORD-REC.
         004800                  COPY DDS-ALL-FORMATS OF PASSWORD.
   46    +000001       05  PASSWORD-RECORD PIC X(20).                               <-ALL-FMTS
         +000002*    FORMATO E-S:PASSWORDS       DEL ARCH. PASSWORD       DE LA BIBL. TESTLIB <-ALL-FMTS
         +000003*                                                                    <-ALL-FMTS
         +000004*DEFINICIONES DE CLAVE DEL FORMATO DE REGISTROS  PASSWORDS          <-ALL-FMTS
         +000005*  NUMERO              NOMBRE              RECUPERACION    ALTSEQ   <-ALL-FMTS
         +000006*   0001   PASSKEY                        ASCENDING         NO     <-ALL-FMTS
   47    +000007       05  PASSWORDS     REDEFINES PASSWORD-RECORD.                 <-ALL-FMTS
   48    +000008           06 PASSKEY               PIC X(10).                      <-ALL-FMTS
   49    +000009           06 PASSWORD              PIC X(10).                      <-ALL-FMTS
         004900
   50    005000 FD  PRINTER-FILE.
   51    005100 01  PRINTER-REC.
   52    005200    05 PRINTER-RECORD        PIC X(132).
         005300
   53    005400 WORKING-STORAGE SECTION.
         005500
         005600*************************************************************
         005700*       DECLARAR EL ESTADO DE ARCHIVO PARA CADA ARCHIVO     *
         005800*************************************************************
         005900
   54    006000 01  MULTIPLE-FS1           PIC X(2)    VALUE SPACES.
   55    006100 01  MULTIPLE-FS2. 7
   56    006200    05  MULTIPLE-MAJOR      PIC X(2)    VALUE SPACES.
   57    006300    05  MULTIPLE-MINOR      PIC X(2)    VALUE SPACES.
   58    006400 01  TERMINAL-FS1           PIC X(2)    VALUE SPACES.
   59    006500 01  PASSWORD-FS1           PIC X(2)    VALUE SPACES.
         006600
         006700*************************************************************
         006800*   DECLARAR ESTRUCTURA PARA CONTENER ATRIBUTOS DE ARCHIVO   *
         006900*************************************************************
         007000
   60    007100 01  STATION-ATTR.
   61    007200    05 STATION-TYPE          PIC X(1). 8
   62    007300    05 STATION-SIZE          PIC X(1).
   63    007400    05 STATION-LOC           PIC X(1).
   64    007500    05 FILLER                PIC X(1).
   65    007600    05 STATION-ACQUIRE       PIC X(1).
   66    007700    05 STATION-INVITE        PIC X(1).
   67    007800    05 STATION-DATA          PIC X(1).
   68    007900    05 STATION-STATUS        PIC X(1).
   69    008000    05 STATION-DISPLAY       PIC X(1).
   70    008100    05 STATION-KEYBOARD      PIC X(1).
   71    008200    05 STATION-SIGNON        PIC X(1).
   72    008300    05 FILLER                PIC X(5).
         008400
         008500*************************************************************
         008600*       DECLARAR AREA DE CONTROL PARA ARCHIVO-MÚLTIPLE      *
```

Figura 115 (Parte 2 de 5). Listado fuente ILE COBOL/400 para soporte de archivos de múltiples dispositivos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPMDF          AS400SYS   96/07/04 11:38:49         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


        008700*****************************************************************
        008800
  73    008900 01  MULTIPLE-CONTROL-AREA.
  74    009000    05 MULTIPLE-KEY-FEEDBACK PIC X(2)    VALUE SPACES.
  75    009100    05 MULTIPLE-DEVICE-NAME  PIC X(10)   VALUE SPACES.
  76    009200    05 MULTIPLE-FORMAT-NAME  PIC X(10)   VALUE SPACES.
        009300
        009400*****************************************************************
        009500*      DECLARAR VARIABLES DE NOTIFICACION DE ERRORES         *
        009600*****************************************************************
        009700
  77    009800 01  HEADER-LINE.
  78    009900    05 FILLER                PIC X(60)   VALUE SPACES.
  79    010000    05 FILLER                PIC X(72)
        010100                               VALUE "MDF ERROR REPORT".
  80    010200 01  DETAIL-LINE.
  81    010300    05 FILLER                PIC X(15)   VALUE SPACES.
  82    010400    05 DESCRIPTION           PIC X(25)   VALUE SPACES.
  83    010500    05 DETAIL-VALUE          PIC X(92)   VALUE SPACES.
        010600
        010700*****************************************************************
        010800* DECLARAR CONTADORES, INDICADORES Y VARIABLES DE ALMACENAMIENTO *
        010900*****************************************************************
        011000
  84    011100 01  CURRENT-TERMINAL        PIC X(10)   VALUE SPACES.
  85    011200 01  TERMINAL-ARRAY.
  86    011300    05 LIST-OF-TERMINALS OCCURS 250 TIMES.
  87    011400      07 DEVICE-NAME         PIC X(10).
  88    011500 01  COUNTER                 PIC 9(3)    VALUE IS 1.
  89    011600 01  NO-OF-TERMINALS         PIC 9(3)    VALUE IS 1.
  90    011700 01  TERMINAL-LIST-FLAG      PIC 1.
  91    011800    88 END-OF-TERMINAL-LIST              VALUE IS B"1".
  92    011900    88 NOT-END-OF-TERMINAL-LIST          VALUE IS B"0".
  93    012000 01  NO-DATA-FLAG            PIC 1.
  94    012100    88 NO-DATA-AVAILABLE                 VALUE IS B"1".
  95    012200    88 DATA-AVAILABLE                    VALUE IS B"0".
        012300
  96    012400 PROCEDURE DIVISION.
        012500
  97    012600 DECLARATIVES.
        012700
        012800 MULTIPLE-SECTION SECTION.
        012900     USE AFTER STANDARD EXCEPTION PROCEDURE ON MULTIPLE-FILE.
        013000
        013100 MULTIPLE-PARAGRAPH.
  98    013200     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
  99    013300     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
 100    013400     MOVE "MULTIPLE FILE" TO DETAIL-VALUE OF DETAIL-LINE.
 101    013500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
 102    013600     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
 103    013700     MOVE MULTIPLE-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
 104    013800     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
 105    013900     MOVE "EXTENDED STATUS IS:" TO DESCRIPTION OF DETAIL-LINE. 9
 106    014000     MOVE MULTIPLE-FS2 TO DETAIL-VALUE OF DETAIL-LINE.
 107    014100     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
 108    014200     ACCEPT STATION-ATTR FROM ATTR. 1
 109    014300     MOVE "FILE ATTRIBUTES ARE:" TO DESCRIPTION OF DETAIL-LINE.
```

Figura 115 (Parte 3 de 5). Listado fuente ILE COBOL/400 para soporte de archivos de múltiples dispositivos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPMDF          AS400SYS   96/07/04 11:38:49         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


  110     014400     MOVE STATION-ATTR TO DETAIL-VALUE OF DETAIL-LINE.
  111     014500     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  112     014600     STOP RUN.
          014700
          014800 TERMINAL-SECTION SECTION.
          014900     USE AFTER STANDARD EXCEPTION PROCEDURE ON TERMINAL-FILE.
          015000 TERMINAL-PARAGRAPH.
  113     015100     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
  114     015200     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
  115     015300     MOVE "TERMINAL FILE" TO DETAIL-VALUE OF DETAIL-LINE.
  116     015400     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
  117     015500     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
  118     015600     MOVE TERMINAL-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
  119     015700     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  120     015800     STOP RUN.
          015900
          016000 PASSWORD-SECTION SECTION.
          016100     USE AFTER STANDARD EXCEPTION PROCEDURE ON PASSWORD-FILE.
          016200 PASSWORD-PARAGRAPH.
  121     016300     WRITE PRINTER-REC FROM HEADER-LINE AFTER ADVANCING PAGE.
  122     016400     MOVE "FILE NAME IS:" TO DESCRIPTION OF DETAIL-LINE.
  123     016500     MOVE "PASSWORD FILE" TO DETAIL-VALUE OF DETAIL-LINE.
  124     016600     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 5 LINES.
  125     016700     MOVE "FILE STATUS IS:" TO DESCRIPTION OF DETAIL-LINE.
  126     016800     MOVE PASSWORD-FS1 TO DETAIL-VALUE OF DETAIL-LINE.
  127     016900     WRITE PRINTER-REC FROM DETAIL-LINE AFTER ADVANCING 2 LINES.
  128     017000     STOP RUN.
          017100
          017200 END DECLARATIVES.
          017300
          017400*************************************************************
          017500*     AQUI EMPIEZA LA LOGICA DEL PROGRAMA PRINCIPAL         *
          017600*************************************************************
          017700
          017800 MAIN-PROGRAM SECTION.
          017900 MAINLINE.
  129     018000     OPEN I-O    MULTIPLE-FILE 11
          018100          INPUT  TERMINAL-FILE
          018200          I-O    PASSWORD-FILE
          018300          OUTPUT PRINTER-FILE.
          018400
  130     018500       MOVE 1 TO COUNTER.
  131     018600       SET NOT-END-OF-TERMINAL-LIST TO TRUE.
          018700******
          018800*      Traer lista de terminales
          018900******
  132     019000       PERFORM UNTIL END-OF-TERMINAL-LIST
  133     019100          READ TERMINAL-FILE RECORD
          019200             INTO LIST-OF-TERMINALS(COUNTER)
          019300             AT END
  134     019400                SET END-OF-TERMINAL-LIST TO TRUE
  135     019500                SUBTRACT 1 FROM COUNTER
  136     019600                MOVE COUNTER TO NO-OF-TERMINALS
          019700          END-READ
  137     019800          ADD 1 TO COUNTER
          019900       END-PERFORM.
          020000******
```

Figura 115 (Parte 4 de 5). Listado fuente ILE COBOL/400 para soporte de archivos de múltiples dispositivos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPMDF          AS400SYS   96/07/04 11:38:49         Página     6

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


         020100*      Adquirir terminales e invitarlos
         020200******
  138    020300       PERFORM VARYING COUNTER FROM 1 BY 1
         020400               UNTIL COUNTER GREATER THAN NO-OF-TERMINALS
  139    020500          ACQUIRE LIST-OF-TERMINALS(COUNTER) FOR MULTIPLE-FILE 12
  140    020600          WRITE MULTIPLE-REC 13
         020700             FORMAT IS "SIGNON"
         020800             TERMINAL IS LIST-OF-TERMINALS(COUNTER)
         020900          END-WRITE
         021000       END-PERFORM.
         021100
  141    021200       MOVE 1 TO COUNTER.
  142    021300       SET DATA-AVAILABLE TO TRUE.
         021400******
         021500*      Sondear a los terminales
         021600******
  143    021700       PERFORM UNTIL NO-DATA-AVAILABLE
  144    021800          READ MULTIPLE-FILE RECORD 14
  145    021900          IF MULTIPLE-FS2 EQUAL "310" THEN
  146    022000             SET NO-DATA-AVAILABLE TO TRUE  15
         022100          END-IF
  147    022200          IF DATA-AVAILABLE THEN
  148    022300             MOVE MULTIPLE-DEVICE-NAME TO CURRENT-TERMINAL
         022400******
         022500*            Validar la contraseña  16
         022600******
  149    022700             MOVE CURRENT-TERMINAL TO PASSKEY OF PASSWORD-REC
  150    022800             READ PASSWORD-FILE RECORD
  151    022900             IF PASSWORD OF SIGNON-I EQUAL
         023000                PASSWORD OF PASSWORD-REC THEN
  152    023100                   CALL "UPDT" USING CURRENT-TERMINAL
  153    023200                   MOVE SPACES TO WRONG OF SIGNON-O
         023300             ELSE
  154    023400                   MOVE "INVALID PASSWORD" TO WRONG OF SIGNON-O
         023500             END-IF
  155    023600             WRITE MULTIPLE-REC
         023700                FORMAT IS "SIGNON"
         023800                TERMINAL IS CURRENT-TERMINAL
         023900             END-WRITE
         024000          END-IF
         024100       END-PERFORM.
         024200******
         024300*      Liberar los terminales
         024400******
  156    024500       PERFORM VARYING COUNTER FROM 1 BY 1
         024600               UNTIL COUNTER GREATER THAN NO-OF-TERMINALS
  157    024700          DROP LIST-OF-TERMINALS(COUNTER) FROM MULTIPLE-FILE  17
         024800       END-PERFORM.
         024900
  158    025000       CLOSE   MULTIPLE-FILE
         025100               TERMINAL-FILE
         025200               PASSWORD-FILE
         025300               PRINTER-FILE.
  159    025400       STOP RUN.
         025500

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 115 (Parte 5 de 5). Listado fuente ILE COBOL/400 para soporte de archivos de múltiples dispositivos

1 ATTR es un nombre mnemotécnico asociado con nombre-función ATTRIBUTE-DATA. ATTR se utiliza en la instrucción ACCEPT para obtener los datos de atributos para el archivo TRANSACTION MULTIPLE-FILE. Véase el punto 1 .

2 El archivo MULT debe haberse creado con el mandato CRTDSPF, en el que el parámetro DEV tiene el valor *NONE y el parámetro MAXDEV tiene un valor mayor que 1. El parámetro WAITRCD especifica el tiempo de espera

para las operaciones READ en el archivo. El parámetro WAITRCD debe tener un valor mayor que 0.

3 MULTIPLE-FS2 es el estado de archivo ampliado del archivo TRANSACTION MULTIPLE-FILE. Esta variable se ha declarado en el apartado WORKING-STORAGE del programa. Véase el punto 7 .

4 MULTIPLE-CONTROL-AREA es el área de control del archivo TRANSACTION MULTIPLE-FILE. Esta variable se utiliza para determinar qué dispositivo de programa se ha utilizado para iniciar la sesión. Véase el punto 15 .

5 La descripción de datos de MULTIPLE-REC se ha definido con la instrucción COPY DDS.

Nota: Sólo los campos que se copian son campos con nombre. Consulte las DDS de este ejemplo para ver los comentarios referentes a las DDS utilizadas.

6 SIGNON es el formato con la palabra clave INVITE. Es el formato que se utilizará para invitar a los dispositivos por medio de la instrucción WRITE.

7 Es la declaración del estado de archivo ampliado MULTIPLE-FS2. Es un campo de 4 bytes que está subdividido en un código de retorno principal (los 2 primeros bytes) y un código de retorno secundario (los 2 últimos).

8 STATION-ATTR es el lugar en el que la instrucción ACCEPT almacena los datos de atributo del archivo TRANSACTION MULTIPLE-FILE. Véase el punto 1 .

9 En esta instrucción, se escribe el estado de archivo ampliado MULTIPLE-FS2.

1 Esta instrucción acepta los datos de atributo del archivo TRANSACTION MULTIPLE-FILE. Dado que no se ha especificado la expresión FOR con la instrucción ACCEPT, se utiliza el último dispositivo de programa.

11 Esta instrucción abre al archivo TRANSACTION MULTIPLE-FILE. Dado que el parámetro ACQPGMDEV del mandato CRTDSPF tiene el valor *NONE, no se adquiere ningún dispositivo de programa de forma implícita al abrir el archivo.

12 Esta instrucción adquiere el dispositivo de programa contenido en la variable LIST-OF-TERMINALS (COUNTER), para el archivo TRANSACTION MULTIPLE-FILE.

13 Esta instrucción WRITE invita al dispositivo de programa especificado en la expresión TERMINAL. El formato SIGNON tiene asociada la palabra clave de DDS INVITE. Véase el punto 14 .

14 Esta instrucción READ leerá en cualquier dispositivo de programa invitado. Véase el punto 13 . Si el tiempo de espera expira antes de que alguien entre algo en los dispositivos invitados, el estado de archivo ampliado quedará establecido como “0310” y el proceso continuará. Véase el punto 15 .

15 En esta instrucción, se comprueba el estado de archivo ampliado de MULTIPLE-FILE para ver si ha expirado el tiempo de espera.

16 Se utiliza el nombre de dispositivo de programa almacenado en el área de control para determinar qué dispositivo de programa se ha utilizado al iniciar la sesión. Véase el punto 4 .

17 Esta instrucción DROP desconecta el dispositivo de programa contenido en la variable LIST-OF-TERMINALS del archivo TRANSACTION MULTIPLE-FILE.

Escritura de programas que utilicen archivos de transacción de subarchivo

Por lo general, los archivos TRANSACTION de subarchivo se utilizan para leer un grupo de registros de un dispositivo de pantalla o para escribir un grupo de registros en él. Para utilizar un archivo TRANSACTION de subarchivo en un programa ILE COBOL/400, debe:

- darle nombre al archivo por medio de una entrada de control de archivo en el párrafo FILE-CONTROL de Environment Division
- describir el archivo mediante una entrada de descripción de archivo en Data Division
- utilizar ampliaciones a las instrucciones de Procedure Division que den soporte al proceso de transacciones.

Denominación de un archivo de transacción de subarchivo

Para utilizar un archivo TRANSACTION de un subarchivo en un programa ILE COBOL/400, debe dar nombre al archivo por medio de una entrada de control de archivo en el párrafo FILE-CONTROL. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción completa del párrafo FILE-CONTROL.

De nombre al archivo TRANSACTION en el párrafo FILE-CONTROL de la forma siguiente:

```
FILE-CONTROL.
    SELECT nombre-archivo-transacción
        ASSIGN TO WORKSTATION-nombre_archivo_pantalla
        ORGANIZATION IS TRANSACTION
        ACCESS MODE IS DYNAMIC
               RELATIVE KEY IS dato-clave-relativa
        CONTROL AREA IS dato-área-control.
```

Utilice la cláusula SELECT para elegir un archivo. Dicho archivo debe estar identificado con una entrada FD en Data Division.

Utilice la cláusula ASSIGN para asociar el archivo TRANSACTION con un archivo de pantalla.Debe especificar el tipo de dispositivo WORKSTATION en la cláusula ASSIGN para utilizar archivos TRANSACTION. Si desea utilizar un área de indicadores por separado para este archivo TRANSACTION, será necesario que incluya el atributo -SI con la cláusula ASSIGN. Consulte el apartado “Utilización de indicadores con archivos de transacción” en la página 411 para obtener información más detallada sobre la utilización del área de indicadores por separado.

Debe especificar ORGANIZATION IS TRANSACTION en la entrada de control de archivo para poder utilizar un archivo TRANSACTION. Esta cláusula le indica al programa ILE COBOL/400 que interactuará con un usuario de estación de trabajo o con otro sistema.

A un archivo TRANSACTION de subarchivo se accede de forma dinámica. El acceso dinámico le permite leer o grabar registros en el archivo de forma secuencial o aleatoria, dependiendo de la forma de la petición específica de entrada-

salida. Los subarchivos son los únicos archivos TRANSACTION a los que puede accederse de forma aleatoria. La cláusula ACCESS MODE de la entrada de control de archivo sirve para explicarle al programa ILE COBOL/400 la forma de acceder al archivo TRANSACTION. Debe especificar ACCESS MODE IS DYNAMIC para leer o escribir en el archivo TRANSACTION de subarchivo.

Al utilizar subarchivos, debe proporcionar una clave relativa.Para identificar el dato de clave relativa, utilice la cláusula RELATIVE KEY. El dato de clave relativa especifica el número relativo de registro de un registro específico de un subarchivo.

Si desea información de retorno sobre el estado de una petición de entrada/salida que haga referencia a un archivo TRANSACTION, defina un dato de clave de estado en la entrada de control de archivo con la cláusula FILE STATUS. Cuando se especifica la cláusula FILE STATUS, el sistema mueve un valor en el dato de clave de estado después de cada petición de entrada-salida que haga referencia de forma explícita o implícita al archivo TRANSACTION. El valor indica el estado de la ejecución de la instrucción de E-S.

Puede obtener información específica, dependiente del dispositivo y dependiente del sistema que se utiliza para controlar la operaciones de entrada-salida de los archivos TRANSACTION, identificando un dato de área de control con la cláusula CONTROL-AREA. Puede definir el dato especificado por la cláusula CONTROL-AREA en LINKAGE SECTION o WORKING-STORAGE SECTION con el formato siguiente:

```
01  dato-área-control.
    05  tecla-función         PIC X(2).
    05  nombre-dispositivo    PIC X(10).
    05  formato-registro      PIC X(10).
```

El área de control puede tener una longitud de 2, 12 ó 22 bytes. Así pues, sólo puede especificar el primer elemento de nivel 05, los dos primeros elementos de nivel 05 o los tres elementos de nivel 05, dependiendo del tipo de información que busque.

El dato de área de control le permitirá identificar:

- la tecla de función que pulsa el operador para iniciar una transacción
- el nombre del dispositivo de programa utilizado
- el nombre del formato de registro de DDS al que ha hecho referencia en la última instrucción de E-S.

Descripción de un archivo de transacción de subarchivo

Para utilizar un archivo TRANSACTION en un programa ILE COBOL/400, debe describir el archivo por medio de una entrada de descripción de archivo en Data Division. Consulte la publicación *ILE COBOL/400 Reference* para obtener una descripción completa de la entrada de descripción de archivo.Para describir un archivo TRANSACTION, utilice la entrada de descripción de archivo de formato 6.

Una entrada de descripción de archivo de Data Division que describe a un archivo TRANSACTION es así:

```
FD  CUST-DISPLAY.
01  DISP-REC.
    COPY DDS-ALL-FORMATS OF CUSMINQ.
```

En ILE COBOL/400, los archivos TRANSACTION suelen estar descritos externamente.Cree una DDS para el archivo TRANSACTION que desee utilizar.Consulte el apartado “Definición de archivos de transacción con especificaciones de descripción de datos” en la página 395 para saber cómo crear una DDS o bien la publicación *DDS Reference*. A continuación, cree el archivo TRANSACTION.

Una vez haya creado la DDS para el archivo TRANSACTION así como éste, utilice la instrucción COPY de formato 2 para describir el diseño del registro de datos del archivo TRANSACTION. Al compilar el programa ILE COBOL/400, COPY de formato 2 creará las instrucciones de Data Division para describir el archivo TRANSACTION. Para generar un área de almacenamiento para todos los formatos, utilice la opción DDS-ALL-FORMATS de la instrucción COPY de formato 2.

Proceso de un archivo de transacción de subarchivo

La lista que sigue a continuación muestra todas las instrucciones de Procedure Division que tienen ampliaciones específicamente para procesar archivos TRANSACTION en un programa ILE COBOL/400. Consulte la publicación *ILE COBOL/400 Reference* si desea una explicación detallada de cada una de estas sentencias.

- Instrucción ACCEPT - formato 6
- Instrucción ACQUIRE
- Instrucción CLOSE - formato 1
- Instrucción DROP
- Instrucción OPEN - formato 3
- Instrucción READ - formato 5 (subarchivo)
- Instrucción REWRITE - formato 2 (subarchivo)
- Instrucción WRITE - formato 5 (subarchivo)

Apertura de un archivo de transacción de subarchivo

Para procesar un archivo TRANSACTION en Procedure Division, primero debe abrir el archivo.Para abrir un archivo TRANSACTION, utilice la sentencia OPEN de formato 3.Un archivo TRANSACTION debe abrirse en modalidad de E-S.

```
OPEN I-O nombre-archivo.
```

Adquisición de dispositivos de programa

Debe adquirir un dispositivo de programa para el archivo TRANSACTION. Una vez adquirido, estará disponible para operaciones de entrada y salida.Puede adquirirlo de forma implícita o explícita.

Un dispositivo de programa se adquiere implícitamente al abrir el archivo TRANSACTION. Si es un archivo de pantalla, el dispositivo está determinado por la primera entrada del parámetro DEV del mandato CRTDSPF. Los dispositivos de programa adicionales deben adquirirse explícitamente.

Un dispositivo de programa se adquiere explícitamente con la instrucción ACQUIRE. Para los archivos de pantalla, el dispositivo mencionado en la instrucción ACQUIRE no ha de estar especificado en el parámetro DEV del mandato CRTDSPF, CHGDSPF o OVRDSPF. Sin embargo, al crear el archivo de pantalla, debe especificar el número de dispositivos que pueden adquirirse (el valor por

omisión es uno). Para un archivo de pantalla, el nombre de dispositivo de programa debe coincidir con el dispositivo de pantalla.

```
ACQUIRE nombre-dispositivo-programa FOR nombre-archivo-transacción.
```

Grabación de un archivo de transacción de subarchivo

Una vez abierto el archivo TRANSACTION y adquirido un dispositivo de programa, ya está preparado para realizar operaciones de entrada y salida en él.

La primera operación de entrada/salida que suele realizarse en un archivo TRANSACTION es escribir un registro en la pantalla. Este registro se utiliza para pedir al usuario que entre una respuesta o datos.

Para escribir un registro lógico en el archivo TRANSACTION de subarchivo, utilice la instrucción WRITE de formato 5.El código de la sentencia WRITE es el siguiente:

```
WRITE SUBFILE nombre-registro FORMAT IS nombre-formato.
```

En determinadas situaciones, es posible que tenga varios registros de datos, cada uno de ellos con un formato diferente, que desee tener activos para un archivo TRANSACTION. En tal caso, debe utilizar la expresión FORMAT de la instrucción WRITE de formato 5 para especificar el formato del registro de datos de salida que desea escribir en el archivo TRANSACTION.

Si ha adquirido de forma explícita varios dispositivos de programa para el archivo TRANSACTION, debe utilizar la expresión TERMINAL de la instrucción WRITE de formato 5 para especificar el subarchivo del dispositivo de programa al que desea que se envíe el registro de salida.

```
WRITE SUBFILE nombre-registro
      FORMAT IS nombre-formato
      TERMINAL IS nombre-dispositivo-programa
END-WRITE.
```

Antes o después de rellenar el archivo TRANSACTION de subarchivo con registros utilizando la instrucción WRITE de formato 5, puede escribir el registro de control de subarchivo en el dispositivo de programa con la instrucción WRITE de formato 4.Consulte el apartado “Escritura en un archivo de transacción” en la página 401 para obtener una descripción de la manera de utilizar la instrucción WRITE de formato para escribir en un archivo TRANSACTION. Escribir el registro de control de subarchivo podría provocar la visualización del registro de control de subarchivo o de los registros de subarchivo, o bien la visualización tanto del registro de control de subarchivo como de los registros de subarchivo.

Lectura de un archivo de transacción de subarchivo

Para leer un registro de control de subarchivo, utilice la instrucción READ de formato 4. Consulte el apartado “Lectura en un archivo de transacción” en la página 402 para obtener una descripción de la manera de utilizar la instrucción READ de formato 4 para leer en un archivo TRANSACTION. La lectura del registro de control de subarchivo transfiere físicamente los registros del dispositivo de programa de manera que puedan quedar disponibles para el subarchivo.

Una vez los registros están disponibles para el subarchivo, utilice la instrucción READ de formato 5 para leer un registro específico del archivo TRANSACTION de subarchivo. La instrucción READ de formato 5 sólo puede utilizarse para leer un

formato que sea un registro de subarchivo; no se puede utilizar para dispositivos de comunicaciones.

Antes de utilizar la instrucción READ, debe haber adquirido al menos un dispositivo de programa para el archivo TRANSACTION. Si se realiza una instrucción READ y no hay dispositivos de programa adquiridos, se notifica un error de lógica estableciendo el estado de archivo en 92.

Los subarchivos pueden leerse de forma secuencial o aleatoria.

Para leer un subarchivo de forma secuencial, debe especificar la expresión NEXT MODIFIED en la instrucción READ de formato 5. Cuando se especifica la expresión NEXT MODIFIED, el registro que queda disponible es el primer registro del subarchivo que se ha modificado.Para obtener información sobre cómo se marca un registro de subarchivo como modificado, consulte la publicación *Gestión de datos*. Si no hay disponible ningún registro siguiente de subarchivo modificado, existe la condición AT END, el estado de archivo se establece como 12 y el valor del elemento de datos RELATIVE KEY se establece como la clave del último registro del subarchivo.

Al leer un subarchivo de forma secuencial, debe especificar también la expresión AT END en la instrucción READ de formato 5.La expresión AT END le permite especificar una instrucción imperativa para que se ejecute cuando se produzca la condición AT END.

```
READ SUBFILE nombre-subarchivo NEXT MODIFIED RECORD
     AT END instrucción-imperativa
END-READ
```

Para leer un subarchivo de forma aleatoria, debe especificar, en el elemento de datos RELATIVE KEY, el número relativo de registro del registro de subarchivo que desea leer y no debe especificar la expresión NEXT MODIFIED en la instrucción READ de formato 5. Cuando no se especifica la expresión NEXT MODIFIED, el registro que queda disponible es el registro del subarchivo con un número relativo de registro que corresponde al valor del elemento de datos RELATIVE KEY. Si el elemento de datos RELATIVE KEY, en el momento de realizarse la instrucción READ, contiene un valor que no corresponde a un número relativo de registro del subarchivo se produce una condición INVALID KEY.

Al leer un subarchivo de forma aleatoria, debe especificar también la expresión INVALID KEY en la instrucción READ de formato 5. La expresión INVALID KEY le permite especificar una instrucción imperativa para que se ejecute cuando se produzca la condición INVALID KEY.

```
READ SUBFILE nombre-subarchivo RECORD
     INVALID KEY instrucción-imperativa
END-READ
```

Para obtener una explicación detallada de cómo se realiza la instrucción READ, consulte el apartado dedicado a la instrucción en la publicación *ILE COBOL/400 Reference*.

En aquellos casos en los que haya adquirido varios dispositivos de programa, puede especificar explícitamente el dispositivo de programa del que lee datos, identificándolo en la expresión TERMINAL de la instrucción READ.

En aquellos casos en los que desee recibir los datos con un formato específico, puede identificarlo en la expresión FORMAT de la instrucción READ. Si los datos disponibles no coinciden con el formato de registro solicitado, se establece el estado de archivo 9K.

Los ejemplos que siguen a continuación son de la instrucción READ con las expresiones TERMINAL y FORMAT especificadas.

```
READ SUBFILE nombre-subarchivo RECORD
     FORMAT IS formato-registro
END-READ

READ SUBFILE nombre-subarchivo RECORD
     TERMINAL IS nombre-dispositivo-programa
END-READ

READ SUBFILE nombre-subarchivo RECORD
     FORMAT IS formato-registro
     TERMINAL IS nombre-dispositivo-programa
END-READ
```

Sustitución (regrabación) de un registro de subarchivo

Una vez haya leído y modificado un registro de subarchivo, puede reemplazarlo en el subarchivo con la instrucción REWRITE.

```
REWRITE SUBFILE nombre-registro
     FORMAT IS formato-registro
     TERMINAL IS nombre-dispositivo-programa
END-REWRITE
```

El registro reemplazado en el subarchivo es el registro del subarchivo al que ha accedido la operación READ satisfactoria anterior.

Liberación de dispositivos de programa

Una vez que haya terminado de utilizar un dispositivo de programa previamente adquirido para un archivo TRANSACTION, debe liberarlo. Liberar un dispositivo de programa significa que ya no estará disponible para operaciones de entrada o salida mediante el archivo TRANSACTION. Liberando un dispositivo de programa éste queda disponible para otras aplicaciones. Un dispositivo de programa puede liberarse de forma implícita o explícita.

Los dispositivos de programa conectados a un archivo TRANSACTION se liberan implícitamente al cerrar el archivo.

Un dispositivo de programa se libera explícitamente indicándolo en la instrucción DROP. El dispositivo, una vez liberado, puede volver a adquirirse otra vez si fuese necesario.

```
DROP nombre-dispositivo-programa FROM nombre-archivo-transacción.
```

Cierre de un archivo de transacción de subarchivo

Cuando termine de utilizar un archivo TRANSACTION de subarchivo, debe cerrarlo. Para cerrar el archivo TRANSACTION, utilice la sentencia CLOSE de formato 1. Una vez cerrado el archivo, no puede procesarse de nuevo hasta que vuelva a abrirse.

```
CLOSE nombre-archivo-transacción.
```

Ejemplo de utilización de WRITE SUBFILE en un programa de consulta de pedidos

La Figura 119 en la página 449 muestra un ejemplo de programa de consulta de pedidos, ORDINQ, que utiliza subarchivos. También se muestran las DDS asociadas, excepto las DDS del archivo maestro de clientes, CUSMSTP. Vea la Figura 102 en la página 406 para tener las DDS de CUSMSTP.

ORDINQ visualiza todos los registros de detalles del pedido correspondientes al número de pedido solicitado.El programa pide al usuario que entre el número de pedido que ha de revisarse.El número de pedido se coteja con el archivo de cabecera de pedido, ORDHDRP. Si existe el número de pedido, se coteja el número de cliente al que se accede desde el archivo de cabecera de pedido con el archivo maestro de clientes, CUSMSTP. Se leen todos los registros de detalles del pedido que hay en ORDDTLP para el pedido solicitado y se graban en el subarchivo.Se procesa una operación de escritura para el formato de registro de control de subarchivo y se visualizan los registros de detalles del pedido para que el usuario los revise. Para finalizar el programa, hay que pulsar F12.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A** ORDDTLP FISICO     ARCHIVO DE DETALLES DE PEDIDO
     A
     A                                      UNIQUE
     A*
     A          R ORDDTL                    TEXT('ORDER DETAIL RECORD')
     A*
     A            CUST           5          CHECK(MF)
     A                                      COLHDG('CUSTOMER' 'NUMBER')
     A*
     A            ORDERN         5  0       COLHDG('ORDER' 'NUMBER')
     A*
     A            LINNUM         3  0
     A                                      COLHDG('LINE' 'NO')
     A                                      TEXT('LINE NUMBER OF LINE IN ORDER'
     A                                      )
     A*
     A            ITEM           5  0       CHECK(M10)
     A                                      COLHDG('ITEM' 'NUMBER')
     A            QTYORD         3  0
     A                                      COLHDG('QUANTITY' 'ORDERED')
     A                                      TEXT('QUANTITY ORDERED')
     A*
     A            DESCRP        30          COLHDG('ITEM' 'DESCRIPTION')
     A*
     A            PRICE          6  2       CMP(GT 0)
     A                                      COLHDG('PRICE')
     A                                      TEXT('SELLING PRICE')
     A                                      EDTCDE(J)
     A            EXTENS         8  2       COLHDG('EXTENSION')
     A                                      TEXT('EXTENSION AMOUNT OF QTYORD X
     A                                      PRICE')
     A*
     A            WHSLOC         3          CHECK(MF)
     A                                      COLHDG('BIN' 'NO.')
     A*
     A            ORDDAT         6  0       TEXT('DATE ORDER WAS ENTERED')
     A*
     A            CUSTYP         1  0       RANGE(1 5)
     A                                      COLHDG('CUST' 'TYPE')
     A                                      TEXT('CUSTOMER TYPE 1=GOV 2=SCH   +
     A                                      3=BUS 4=PVT 5=OT')
     A*
     A            STATE          2          CHECK(MF)
     A                                      COLHDG('STATE')
     A*
     A            ACTMTH         2  0       COLHDG('ACCT' 'MTH')
     A                                      TEXT('ACCOUNTING MONTH OF SALE')
     A*
     A            ACTYR          2  0       COLHDG('ACCT' 'YEAR')
     A                                      TEXT('ACCOUNTING YEAR OF SALE')
     A
     A          K ORDERN
     A          K LINNUM
```

Figura 116. Especificaciones de descripción de datos para un programa de consulta de pedidos - archivo de detalles de pedido

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A*   ORDINQD            ARCHIVO DE PANTALLA DE REVISION DE PEDIDOS EXISTENTES
     A
     A*
     A          R SUB1                      SFL
     A            ITEM           5  0  10  2TEXT('NÚMERO DE ARTÍCULO')
     A            QTYORD         3  0  10  9TEXT('CANTIDAD PEDIDA')
     A            DESCRP        30     10 14TEXT('DESCRIPCIÓN DEL ARTÍCULO')
     A            PRICE          6  2  10 46TEXT('PRECIO DE VENTA')
     A            EXTENS         8  2  10 56EDTCDE(J)
     A                                      TEXT('IMPORTE AMPLIADO DE QTYORD +
     A                                      X PRICE')
     A
     A          R SUBCTL1                   SFLCTL(SUB1)
     A  58                                  SFLCLR
     A  57                                  SFLDSP
     A N58                                  SFLDSPCTL
     A                                      SFLSIZ(57)
     A                                      SFLPAG(14)
     A  57                                  SFLEND
     A                                      OVERLAY
     A                                      LOCK
     A N45
     AON47                                  ROLLUP(97 'CONTINUAR PANTALLA')
     A                                      CA12(98 'FIN DEL PROGRAMA')
     A                                      SETOFF(57 'SUBARCHIVO DE PANTALLA')
     A                                      SETOFF(58 'OFF = VISUALIZAR SUBCTL1 O+
     A                                      N = BORRAR SUBARCHIVO')
     A                                    1  2'CONSULTA DE PEDIDOS EXISTENTES'
     A                                    3  2'PEDIDO'
     A            ORDERN         5Y 0B  3  8TEXT('NÚMERO DE PEDIDO')
     A  61                                  ERRMSG('NÚMERO DE PEDIDO NO ENCONTRADO' 61)
     A  47                                  ERRMSG('NO HAY LÍNEAS PARA ESTE PEDIDO' 47)
     A  62                                  ERRMSG('NO HAY REGISTROS DE CLIENTE' 62)
     A                                    4  2'FECHA'
     A            ORDDAT         6  0   4  7TEXT('SE ENTRÓ ORDEN DE FECHA')
     A                                    5  2'N  CLI'
     A            CUST           5       5  9TEXT('NÚMERO DEL CLIENTE')
     A            NAME          25       3 16TEXT('NOMBRE DEL CLIENTE')
     A            ADDR          20       4 16TEXT('DIRECCIÓN DEL CLIENTE')
     A            CITY          20       5 16TEXT('CIUDAD DEL CLIENTE')
     A            STATE          2       6 16TEXT('PROVINCIA DEL CLIENTE')
     A            ZIP            5  0    6 31TEXT('CÓD. ESTADO')
     A                                    1 44'TOTAL'
     A            ORDAMT         8  2   1 51TEXT('IMPORTE TOTAL DEL PEDIDO')
     A                                    2 44'ESTADO'
     A            STSORD        12       2 51
     A                                    3 44'SITUACIÓN'
     A            STSOPN        12       3 51
     A                                    4 44'PEDIDO DEL CLIENTE'
     A            CUSORD        15       4 59TEXT('NÚMERO DE ORDEN DE COMPRA +
     A                                       DEL CLIENTE')
     A                                    5 44'INSTRUCC. DE ENVÍO'
     A            SHPVIA        15       5 59TEXT('INSTRUCCIONES DE ENVIÓ')
     A                                    6 44'FECHA DE IMPRESIÓN'
     A            PRTDAT         6  0    6 57TEXT('FECHA DE IMPRESIÓN DEL PEDIDO')
     A                                    7 29'FACTURA'
     A            INVNUM         5  0    7 38TEXT('NÚMERO DE FACTURA')
     A                                    7 64'MVC'
     A            ACTMTH         2  0    7 68TEXT('MES CONTABLE DE LA VENTA')
     A                                    7 72'AÑO'
     A            ACTYR          2  0    7 77TEXT('EJERCICIO CONTABLE DE LA VENTA')
     A                                    8  2'ARTÍCULO'
     A                                    8  8'CDAD.'
     A                                    8 14'DESCRIPCIÓN'
     A                                    8 46'PRECIO'
     A                                    8 55'EXTENSIÓN'
```

Figura 117. Especificaciones de descripción de datos para un programa de consulta de pedidos - archivo de revisión de pedido

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* ESTE ES EL ARCHIVO DE CABECERA DE PEDIDO -- ORDHDRP
     A
     A
     A                                     UNIQUE
     A          R ORDHDR                   TEXT('ORDER HEADER RECORD')
     A            CUST           5         TEXT('CUSTOMER NUMBER')
     A            ORDERN         5 00      TEXT('ORDER NUMBER')
     A            ORDDAT         6 00      TEXT('DATE ORDER ENTERED')
     A            CUSORD        15         TEXT('CUSTOMER PURCHASE ORDER +
     A                                     NUMBER')
     A            SHPVIA        15         TEXT('SHIPPING INSTRUCTIONS')
     A            ORDSTS         1 00      TEXT('ORDER SATAUS 1PCS 2CNT  +
                                           3CHK 4RDY 5PRT 6PCK')
     A            OPRNAM        10         TEXT('OPERATOR WHO ENTERED ORD')
     A            ORDAMT         8 02      TEXT('DOLLAR AMOUNT OF ORDER')
     A            CUSTYP         1 00      TEXT('CUSTOMER TYPE 1=GOV 2=SCH +
     A                                     3=BUS 4=PVT 5=OT')
     A            INVNUM         5 00      TEXT('INVOICE NUMBER')
     A            PRTDAT         6 00      TEXT('DATE ORDER WAS PRINTED')
     A            OPNSTS         1 00      TEXT('ORDER OPEN STATUS 1=OPEN +
                                           2= CLOSE 3=CANCEL')
     A            TOTLIN         3 00      TEXT('TOTAL LINE ITEMS IN ORDER')
     A            ACTMTH         2 00      TEXT('ACCOUNTING MONTH OF SALE')
     A            ACTYR          2 00      TEXT('ACCOUNTING YEAR OF SALE')
     A            STATE          2         TEXT('STATE')
     A            AMPAID         8 02      TEXT('AMOUNT PAID')
                K ORDERN
```

Figura 118. Especificaciones de descripción de datos para un programa de consulta de pedidos - archivo de cabecera de pedido

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     2

                                  F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     ORDINQ.
          000300*    PROGRAMA DE CONSULTA DE PEDIDOS DE EJEMPLO
          000400
    3     000500 ENVIRONMENT DIVISION.
    4     000600 CONFIGURATION SECTION.
    5     000700 SOURCE-COMPUTER. IBM-AS400.
    6     000800 OBJECT-COMPUTER. IBM-AS400.
    7     000900 INPUT-OUTPUT SECTION.
    8     001000 FILE-CONTROL.
    9     001100     SELECT ORDER-HEADER-FILE
   10     001200         ASSIGN TO DATABASE-ORDHDRP
   11     001300         ORGANIZATION IS INDEXED
   12     001400         ACCESS MODE IS RANDOM
   13     001500         RECORD KEY IS ORDERN OF ORDER-HEADER-RECORD.
   14     001600     SELECT ORDER-DETAIL-FILE
   15     001700         ASSIGN TO DATABASE-ORDDTLP
   16     001800         ORGANIZATION IS INDEXED
   17     001900         ACCESS IS DYNAMIC
   18     002000         RECORD KEY IS ORDER-DETAIL-RECORD-KEY.
   19     002100     SELECT CUSTOMER-MASTER-FILE
   20     002200         ASSIGN TO DATABASE-CUSMSTP
   21     002300         ORGANIZATION IS INDEXED
   22     002400         ACCESS IS RANDOM
   23     002500         RECORD KEY IS CUST OF CUSTOMER-MASTER-RECORD.
   24     002600     SELECT EXISTING-ORDER-DISPLAY-FILE
   25     002700         ASSIGN TO WORKSTATION-ORDINQD
   26     002800         ORGANIZATION IS TRANSACTION
   27     002900         ACCESS IS DYNAMIC
   28     003000         RELATIVE KEY IS SUBFILE-RECORD-NUMBER
   29     003100         FILE STATUS IS STATUS-CODE-ONE.
          003200
   30     003300 DATA DIVISION.
   31     003400 FILE SECTION.
   32     003500 FD  ORDER-HEADER-FILE.
   33     003600 01  ORDER-HEADER-RECORD.
          003700     COPY DDS-ORDHDR OF ORDHDRP.
         +000001*    FORMATO E-S:ORDHDR     DEL ARCH. ORDHDRP    DE LA BIBL. TESTLIB          ORDHDR
         +000002*                         REGISTRO DE CABECERA DE PEDIDO                     ORDHDR
         +000003*          CLAVE DADA POR USUARIO CON CLAUSULA RECORD KEY                     ORDHDR
   34    +000004       05  ORDHDR.                                                           ORDHDR
   35    +000005           06 CUST                 PIC X(5).                                ORDHDR
         +000006*                 NUMERO DE CLIENTE                                         ORDHDR
   36    +000007           06 ORDERN                PIC S9(5)       COMP-3.                  ORDHDR
         +000008*                 NUMERO DE PEDIDO                                          ORDHDR
   37    +000009           06 ORDDAT                PIC S9(6)       COMP-3.                  ORDHDR
         +000010*                 FECHA DE ENTRADA DEL PEDIDO                               ORDHDR
   38    +000011           06 CUSORD                PIC X(15).                               ORDHDR
         +000012*                 NUMERO DE ORDEN DE COMPRA DEL CLIENTE                     ORDHDR
   39    +000013           06 SHPVIA                PIC X(15).                               ORDHDR
         +000014*                 INSTRUCCIONES DE ENVIO                                    ORDHDR
   40    +000015           06 ORDSTS                PIC S9(1)       COMP-3.                  ORDHDR
         +000016*                 ESTADO DEL PEDIDO 1PCS 2CNT  3CHK 4RDY 5PRT 6PCK          ORDHDR
   41    +000017           06 OPRNAM                PIC X(10).                               ORDHDR
         +000018*                 OPERADOR QUE HA ENTRADO EL PEDIDO                         ORDHDR
```

Figura 119 (Parte 1 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


   42    +000019           06 ORDAMT                PIC S9(6)V9(2)  COMP-3.                   ORDHDR
         +000020*                 IMPORTE DEL PEDIDO                                          ORDHDR
   43    +000021           06 CUSTYP                PIC S9(1)       COMP-3.                   ORDHDR
         +000022*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                ORDHDR
   44    +000023           06 INVNUM                PIC S9(5)       COMP-3.                   ORDHDR
         +000024*                 NUMERO DE FACTURA                                           ORDHDR
   45    +000025           06 PRTDAT                PIC S9(6)       COMP-3.                   ORDHDR
         +000026*                 FECHA DE IMPRESION DEL PEDIDO                               ORDHDR
   46    +000027           06 OPNSTS                PIC S9(1)       COMP-3.                   ORDHDR
         +000028*                 ESTADO DEL PEDIDO 1=OPEN 2= CLOSE 3=CANCEL                  ORDHDR
   47    +000029           06 TOTLIN                PIC S9(3)       COMP-3.                   ORDHDR
         +000030*                 LINEAS DE DETALLE EN TOTAL DEL PEDIDO                       ORDHDR
   48    +000031           06 ACTMTH                PIC S9(2)       COMP-3.                   ORDHDR
         +000032*                 MES CONTABLE DE LA VENTA                                    ORDHDR
   49    +000033           06 ACTYR                 PIC S9(2)       COMP-3.                   ORDHDR
         +000034*                 EJERCICIO CONTABLE DE LA VENTA                              ORDHDR
   50    +000035           06 STATE                 PIC X(2).                                 ORDHDR
         +000036*                 PROVINCIA                                                   ORDHDR
   51    +000037           06 AMPAID                PIC S9(6)V9(2)  COMP-3.                   ORDHDR
         +000038*                 IMPORTE PAGADO                                              ORDHDR
          003800
   52     003900 FD  ORDER-DETAIL-FILE.
   53     004000 01  ORDER-DETAIL-RECORD.
          004100     COPY DDS-ORDDTL OF ORDDTLP.
         +000001*    FORMATO E-S:ORDDTL     DEL ARCH. ORDDTLP    DE LA BIBL. TESTLIB          ORDDTL
         +000002*                          REGISTRO DE DETALLES DEL PEDIDO                     ORDDTL
         +000003*          CLAVE DADA POR USUARIO CON CLAUSULA RECORD KEY                       ORDDTL
   54    +000004       05  ORDDTL.                                                            ORDDTL
   55    +000005           06 CUST                  PIC X(5).                                 ORDDTL
         +000006*                 NUMERO DE CLIENTE                                           ORDDTL
   56    +000007           06 ORDERN                PIC S9(5)       COMP-3.                   ORDDTL
         +000008*                 NUMERO DE PEDIDO                                            ORDDTL
   57    +000009           06 LINNUM                PIC S9(3)       COMP-3.                   ORDDTL
         +000010*                 NUMERO DE LINEA DEL PEDIDO                                  ORDDTL
   58    +000011           06 ITEM                  PIC S9(5)       COMP-3.                   ORDDTL
         +000012*                 NUMERO DE ARTICULO                                          ORDDTL
   59    +000013           06 QTYORD                PIC S9(3)       COMP-3.                   ORDDTL
         +000014*                 CANTIDAD PEDIDA                                             ORDDTL
   60    +000015           06 DESCRP                PIC X(30).                                ORDDTL
         +000016*                 DESCRIPCION DEL ARTICULO                                    ORDDTL
   61    +000017           06 PRICE                 PIC S9(4)V9(2)  COMP-3.                   ORDDTL
         +000018*                 PRECIO DE VENTA                                             ORDDTL
   62    +000019           06 EXTENS                PIC S9(6)V9(2)  COMP-3.                   ORDDTL
         +000020*                 IMPORTE AMPLIADO DE QTYORD X PRICE                          ORDDTL
   63    +000021           06 WHSLOC                PIC X(3).                                 ORDDTL
         +000022*                 NUMERO DE DEPOSITO                                          ORDDTL
   64    +000023           06 ORDDAT                PIC S9(6)       COMP-3.                   ORDDTL
         +000024*                 FECHA DE ENTRADA DEL PEDIDO                                 ORDDTL
   65    +000025           06 CUSTYP                PIC S9(1)       COMP-3.                   ORDDTL
         +000026*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT                ORDDTL
   66    +000027           06 STATE                 PIC X(2).                                 ORDDTL
         +000028*                 PROVINCIA                                                   ORDDTL
   67    +000029           06 ACTMTH                PIC S9(2)       COMP-3.                   ORDDTL
         +000030*                 MES CONTABLE DE LA VENTA                                    ORDDTL
   68    +000031           06 ACTYR                 PIC S9(2)       COMP-3.                   ORDDTL
         +000032*                 EJERCICIO CONTABLE DE LA VENTA                              ORDDTL
   69     004200 66  ORDER-DETAIL-RECORD-KEY RENAMES ORDERN THRU LINNUM.
```

Figura 119 (Parte 2 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


        004300
  70    004400 FD  CUSTOMER-MASTER-FILE.
  71    004500 01  CUSTOMER-MASTER-RECORD.
        004600     COPY DDS-CUSMST OF CUSMSTP.
       +000001*    FORMATO E-S:CUSMST     DEL ARCH. CUSMSTP    DE LA BIBL. TESTLIB          CUSMST
       +000002*                          REGISTRO MAESTRO DE CLIENTES                       CUSMST
       +000003*          CLAVE DADA POR USUARIO CON CLAUSULA RECORD KEY                      CUSMST
  72   +000004       05  CUSMST.                                                           CUSMST
  73   +000005           06 CUST                  PIC X(5).                               CUSMST
       +000006*                 NUMERO DE CLIENTE                                         CUSMST
  74   +000007           06 NAME                  PIC X(25).                              CUSMST
       +000008*                 NOMBRE DEL CLIENTE                                        CUSMST
  75   +000009           06 ADDR                  PIC X(20).                              CUSMST
       +000010*                 DIRECCION DEL CLIENTE                                     CUSMST
  76   +000011           06 CITY                  PIC X(20).                              CUSMST
       +000012*                 LOCALIDAD DEL CLIENTE                                     CUSMST
  77   +000013           06 STATE                 PIC X(2).                               CUSMST
       +000014*                 PROVINCIA                                                 CUSMST
  78   +000015           06 ZIP                   PIC S9(5)       COMP-3.                 CUSMST
       +000016*                 CODIGO POSTAL                                             CUSMST
  79   +000017           06 SRHCOD                PIC X(6).                               CUSMST
       +000018*                 CODIGO DE BUSQUEDA DE NUMERO DE CLIENTE                   CUSMST
  80   +000019           06 CUSTYP                PIC S9(1)       COMP-3.                 CUSMST
       +000020*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT              CUSMST
  81   +000021           06 ARBAL                 PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000022*                 SALDO DE CUENTAS POR COBRAR                               CUSMST
  82   +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000024*                 IMPORTE DE CC EN ARCHIVO DE PEDIDOS                       CUSMST
  83   +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000026*                 ULTIMO IMPORTE PAGADO EN CC                               CUSMST
  84   +000027           06 LSTDAT                PIC S9(6)       COMP-3.                 CUSMST
       +000028*                 ULTIMA FECHA DE PAGO EN CC                                CUSMST
  85   +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000030*                 LIMITE DE CREDITO DEL CLIENTE                             CUSMST
  86   +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                 CUSMST
       +000032*                 VENTAS DEL CLIENTE EN EL AÑO ACTUAL                       CUSMST
  87   +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                 CUSMST
       +000034*                 VENTAS DEL CLIENTE EL AÑO PASADO                          CUSMST
        004700
  88    004800 FD  EXISTING-ORDER-DISPLAY-FILE.
  89    004900 01  EXISTING-ORDER-DISPLAY-RECORD.
        005000     COPY DDS-ALL-FORMATS OF ORDINQD.
  90   +000001       05  ORDINQD-RECORD PIC X(171).                                        <-ALL-FMTS
       +000002*    FORMATO E-S:SUB1       DEL ARCH. ORDINQD       DE LA BIBL. TESTLIB     <-ALL-FMTS
       +000003*                                                                          <-ALL-FMTS
  91   +000004       05  SUB1          REDEFINES ORDINQD-RECORD.                        <-ALL-FMTS
  92   +000005           06 ITEM                  PIC S9(5).                              <-ALL-FMTS
       +000006*                 NUMERO DE ARTICULO                                        <-ALL-FMTS
  93   +000007           06 QTYORD                PIC S9(3).                              <-ALL-FMTS
       +000008*                 CANTIDAD PEDIDA                                           <-ALL-FMTS
  94   +000009           06 DESCRP                PIC X(30).                              <-ALL-FMTS
       +000010*                 DESCRIPCION DEL ARTICULO                                  <-ALL-FMTS
  95   +000011           06 PRICE                 PIC S9(4)V9(2).                         <-ALL-FMTS
       +000012*                 PRECIO DE VENTA                                           <-ALL-FMTS
  96   +000013           06 EXTENS                PIC S9(6)V9(2).                         <-ALL-FMTS
       +000014*                 IMPORTE AMPLIADO DE QTYORD X PRICE                        <-ALL-FMTS
       +000015*  FORMATO ENTRADA:SUBCTL1     DEL ARCH. ORDINQD    DE LA BIBL. TESTLIB     <-ALL-FMTS
```

Figura 119 (Parte 3 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


       +000016*                                                                                <-ALL-FMTS
 97    +000017       05  SUBCTL1-I     REDEFINES ORDINQD-RECORD.                              <-ALL-FMTS
 98    +000018           06 SUBCTL1-I-INDIC.                                                   <-ALL-FMTS
 99    +000019                07 IN97             PIC 1  INDIC 97.                          <-ALL-FMTS
       +000020*                            CONTINUAR PANTALLA                                 <-ALL-FMTS
100    +000021                07 IN98             PIC 1  INDIC 98.                          <-ALL-FMTS
       +000022*                            FIN DEL PROGRAMA                                   <-ALL-FMTS
101    +000023                07 IN57             PIC 1  INDIC 57.                          <-ALL-FMTS
       +000024*                            SUBARCHIVO DE PANTALLA                             <-ALL-FMTS
102    +000025                07 IN58             PIC 1  INDIC 58.                          <-ALL-FMTS
       +000026*                            OFF = VISUALIZAR SUBCTL1 ON = BORRAR SUBARCHIVO    <-ALL-FMTS
103    +000027                07 IN61             PIC 1  INDIC 61.                          <-ALL-FMTS
       +000028*                            NUMERO DE PEDIDO NO ENCONTRADO                     <-ALL-FMTS
104    +000029                07 IN47             PIC 1  INDIC 47.                          <-ALL-FMTS
       +000030*                            NO HAY LINEAS PARA ESTE PEDIDO                     <-ALL-FMTS
105    +000031                07 IN62             PIC 1  INDIC 62.                          <-ALL-FMTS
       +000032*                            NO HAY REGISTROS DE CLIENTE                        <-ALL-FMTS
106    +000033           06 ORDERN                PIC S9(5).                                <-ALL-FMTS
       +000034*                 NUMERO DE PEDIDO                                              <-ALL-FMTS
       +000035* FORMATO SALIDA:SUBCTL1    DEL ARCH. ORDINQD    DE LA BIBL. TESTLIB           <-ALL-FMTS
       +000036*                                                                               <-ALL-FMTS
107    +000037       05  SUBCTL1-O     REDEFINES ORDINQD-RECORD.                              <-ALL-FMTS
108    +000038           06 SUBCTL1-O-INDIC.                                                   <-ALL-FMTS
109    +000039                07 IN58             PIC 1  INDIC 58.                          <-ALL-FMTS
       +000040*                            OFF = VISUALIZAR SUBCTL1 ON = BORRAR SUBARCHIVO    <-ALL-FMTS
110    +000041                07 IN57             PIC 1  INDIC 57.                          <-ALL-FMTS
       +000042*                            SUBARCHIVO DE PANTALLA                             <-ALL-FMTS
111    +000043                07 IN45             PIC 1  INDIC 45.                          <-ALL-FMTS
112    +000044                07 IN47             PIC 1  INDIC 47.                          <-ALL-FMTS
       +000045*                            NO HAY LINEAS PARA ESTE PEDIDO                     <-ALL-FMTS
113    +000046                07 IN61             PIC 1  INDIC 61.                          <-ALL-FMTS
       +000047*                            NUMERO DE PEDIDO NO ENCONTRADO                     <-ALL-FMTS
114    +000048                07 IN62             PIC 1  INDIC 62.                          <-ALL-FMTS
       +000049*                            NO HAY REGISTROS DE CLIENTE                        <-ALL-FMTS
115    +000050           06 ORDERN                PIC S9(5).                                <-ALL-FMTS
       +000051*                 NUMERO DE PEDIDO                                              <-ALL-FMTS
116    +000052           06 ORDDAT                PIC S9(6).                                <-ALL-FMTS
       +000053*                 FECHA DE ENTRADA DEL PEDIDO                                   <-ALL-FMTS
117    +000054           06 CUST                  PIC X(5).                                 <-ALL-FMTS
       +000055*                 NUMERO DE CLIENTE                                             <-ALL-FMTS
118    +000056           06 NAME                  PIC X(25).                                <-ALL-FMTS
       +000057*                 NOMBRE DEL CLIENTE                                            <-ALL-FMTS
119    +000058           06 ADDR                  PIC X(20).                                <-ALL-FMTS
       +000059*                 DIRECCION DEL CLIENTE                                         <-ALL-FMTS
120    +000060           06 CITY                  PIC X(20).                                <-ALL-FMTS
       +000061*                 LOCALIDAD DEL CLIENTE                                         <-ALL-FMTS
121    +000062           06 STATE                 PIC X(2).                                 <-ALL-FMTS
       +000063*                 PROVINCIA DEL CLIENTE                                         <-ALL-FMTS
122    +000064           06 ZIP                   PIC S9(5).                                <-ALL-FMTS
       +000065*                 CODIGO POSTAL                                                 <-ALL-FMTS
123    +000066           06 ORDAMT                PIC S9(6)V9(2).                           <-ALL-FMTS
       +000067*                 IMPORTE TOTAL DEL PEDIDO                                      <-ALL-FMTS
124    +000068           06 STSORD                PIC X(12).                                <-ALL-FMTS
125    +000069           06 STSOPN                PIC X(12).                                <-ALL-FMTS
126    +000070           06 CUSORD                PIC X(15).                                <-ALL-FMTS
       +000071*                 NUMERO DE ORDEN DE COMPRA DEL CLIENTE                         <-ALL-FMTS
127    +000072           06 SHPVIA                PIC X(15).                                <-ALL-FMTS
```

Figura 119 (Parte 4 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     6

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


        +000073*                 INSTRUCCIONES DE ENVIO                                    <-ALL-FMTS
 128    +000074           06 PRTDAT                PIC S9(6).                               <-ALL-FMTS
        +000075*                 FECHA DE IMPRESION DEL PEDIDO                             <-ALL-FMTS
 129    +000076           06 INVNUM                PIC S9(5).                               <-ALL-FMTS
        +000077*                 NUMERO DE FACTURA                                         <-ALL-FMTS
 130    +000078           06 ACTMTH                PIC S9(2).                               <-ALL-FMTS
        +000079*                 MES CONTABLE DE LA VENTA                                  <-ALL-FMTS
 131    +000080           06 ACTYR                 PIC S9(2).                               <-ALL-FMTS
        +000081*                 EJERCICIO CONTABLE DE LA VENTA                            <-ALL-FMTS
        005100
 132    005200 WORKING-STORAGE SECTION.
 133    005300 01  EXISTING-ORDER-DISPLAY-KEY.
 134    005400    05  SUBFILE-RECORD-NUMBER               PIC 9(2)
        005500                                         VALUE ZERO.
        005600
 135    005700 01  ORDER-STATUS-COMMENT-VALUES.
 136    005800    05  FILLER                              PIC X(12)
        005900                                         VALUE "1-IN PROCESS".
 137    006000    05  FILLER                              PIC X(12)
        006100                                         VALUE "2-CONTINUED ".
 138    006200    05  FILLER                              PIC X(12)
        006300                                         VALUE "3-CREDIT CHK".
 139    006400    05  FILLER                              PIC X(12)
        006500                                         VALUE "4-READY PRT ".
 140    006600    05  FILLER                              PIC X(12)
        006700                                         VALUE "5-PRINTED   ".
 141    006800    05  FILLER                              PIC X(12)
        006900                                         VALUE "6-PICKED    ".
 142    007000    05  FILLER                              PIC X(12)
        007100                                         VALUE "7-INVOICED  ".
 143    007200    05  FILLER                              PIC X(12)
        007300                                         VALUE "8-INVALID   ".
 144    007400    05  FILLER                              PIC X(12)
        007500                                         VALUE "9-CANCELED  ".
        007600
 145    007700 01  ORDER-STATUS-COMMENT-TABLE
        007800         REDEFINES ORDER-STATUS-COMMENT-VALUES.
 146    007900    05  ORDER-STATUS OCCURS 9 TIMES.
 147    008000       10 ORDER-STATUS-COMMENT               PIC X(12).
        008100
 148    008200 01  OPEN-STATUS-COMMENT-VALUES.
 149    008300    05  FILLER                              PIC X(12)
        008400                                         VALUE "1-OPEN      ".
 150    008500    05  FILLER                              PIC X(12)
        008600                                         VALUE "2-CLOSED    ".
 151    008700    05  FILLER                              PIC X(12)
        008800                                         VALUE "3-CANCELED  ".
        008900
 152    009000 01  OPEN-STATUS-COMMENT-TABLE
        009100         REDEFINES OPEN-STATUS-COMMENT-VALUES.
 153    009200    05  OPEN-STATUS OCCURS 3 TIMES.
 154    009300       10  OPEN-STATUS-COMMENT              PIC X(12).
        009400
 155    009500 01  ERRHDL-PARAMETERS.
 156    009600    05  STATUS-CODE-ONE                     PIC X(2).
 157    009700       88  SUBFILE-IS-FULL               VALUE "0M".
        009800
```

Figura 119 (Parte 5 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ          AS400SYS   96/07/04 11:32:43         Página     7

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


  158     009900 01  ERRPGM-PARAMETERS.
  159     010000    05  DISPLAY-PARAMETER                  PIC X(8)
          010100                                        VALUE "ORD220D ".
  160     010200    05  DUMMY-ONE                          PIC X(6)
          010300                                        VALUE SPACES.
  161     010400    05  DUMMY-TWO                          PIC X(8)
          010500                                        VALUE SPACES.
  162     010600    05  STATUS-CODE-TWO.
  163     010700       10  PRIMARY                        PIC X(1).
  164     010800       10  SECONDARY                      PIC X(1).
  165     010900       10  FILLER                         PIC X(5)
          011000                                        VALUE SPACES.
          011100
  166     011200 01  SWITCH-AREA.
  167     011300    05  SW01                               PIC 1.
  168     011400       88  NO-MORE-DETAIL-LINE-ITEMS      VALUE B"1".
  169     011500       88  MORE-DETAIL-LINE-ITEMS-EXIST   VALUE B"0".
  170     011600    05  SW02                               PIC 1.
  171     011700       88  WRITE-DISPLAY                  VALUE B"1".
  172     011800       88  READ-DISPLAY                   VALUE B"0".
  173     011900    05  SW03                               PIC 1.
  174     012000       88  SUBCTL1-FORMAT                 VALUE B"1".
  175     012100       88  NOT-SUBCTL1-FORMAT             VALUE B"0".
  176     012200    05  SW04                               PIC 1.
  177     012300       88  SUB1-FORMAT                    VALUE B"1".
  178     012400       88  NOT-SUB1-FORMAT                VALUE B"0".
          012500
  179     012600 01  INDICATOR-AREA.
  180     012700    05  IN98                               PIC 1 INDIC 98.
  181     012800       88  END-OF-EXISTING-ORDER-INQUIRY  VALUE B"1".
  182     012900    05  IN97                               PIC 1 INDIC 97.
  183     013000       88  CONTINUE-DETAIL-LINES-DISPLAY  VALUE B"1".
  184     013100    05  IN62                               PIC 1 INDIC 62.
  185     013200       88  CUSTOMER-NOT-FOUND             VALUE B"1".
  186     013300       88  CUSTOMER-EXIST                 VALUE B"0".
  187     013400    05  IN61                               PIC 1 INDIC 61.
  188     013500       88  ORDER-NOT-FOUND                VALUE B"1".
  189     013600       88  ORDER-EXIST                    VALUE B"0".
  190     013700    05  IN58                               PIC 1 INDIC 58.
  191     013800       88  CLEAR-SUBFILE                  VALUE B"1".
  192     013900       88  DISPLAY-SUBFILE-CONTROL        VALUE B"0".
  193     014000    05  IN57                               PIC 1 INDIC 57.
  194     014100       88  DISPLAY-SUBFILE                VALUE B"1".
  195     014200    05  IN47                               PIC 1 INDIC 47.
  196     014300       88  NO-DETAIL-LINES-FOR-ORDER      VALUE B"1".
  197     014400       88  DETAIL-LINES-FOR-ORDER-EXIST   VALUE B"0".
  198     014500    05  IN45                               PIC 1 INDIC 45.
  199     014600       88  END-OF-ORDER                   VALUE B"1".
          014700
  200     014800 PROCEDURE DIVISION.
          014900
  201     015000 DECLARATIVES.
          015100 TRANSACTION-ERROR SECTION.
          015200     USE AFTER STANDARD ERROR PROCEDURE
          015300         EXISTING-ORDER-DISPLAY-FILE.
          015400 WORK-STATION-ERROR-HANDLER.
  202     015500     IF NOT (SUBFILE-IS-FULL) THEN
```

Figura 119 (Parte 6 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página     8

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN  S NOMCOPIA   FEC CAMB


  203     015600         DISPLAY "WORK-STATION ERROR"   STATUS-CODE-ONE
          015700     END-IF.                                                                    94/03/16
          015800 END DECLARATIVES.
          015900
          016000 MAIN-PROGRAM SECTION.
          016100 MAINLINE.
  204     016200     OPEN INPUT  ORDER-HEADER-FILE
          016300                 ORDER-DETAIL-FILE
          016400                 CUSTOMER-MASTER-FILE
          016500          I-O    EXISTING-ORDER-DISPLAY-FILE.
  205     016600     MOVE SPACES TO   CUST   OF SUBCTL1-O
          016700                      NAME   OF SUBCTL1-O
          016800                      ADDR   OF SUBCTL1-O
          016900                      CITY   OF SUBCTL1-O
          017000                      STATE  OF SUBCTL1-O
          017100                      STSORD OF SUBCTL1-O
          017200                      STSOPN OF SUBCTL1-O
          017300                      CUSORD OF SUBCTL1-O.
  206     017400     MOVE ZEROS TO    ORDERN OF SUBCTL1-O
          017500                      ORDDAT OF SUBCTL1-O
          017600                      ZIP    OF SUBCTL1-O
          017700                      ORDAMT OF SUBCTL1-O
          017800                      PRTDAT OF SUBCTL1-O
          017900                      INVNUM OF SUBCTL1-O
          018000                      ACTMTH OF SUBCTL1-O
          018100                      ACTYR  OF SUBCTL1-O.
  207     018200     MOVE B"0" TO INDICATOR-AREA.
  208     018300     SET READ-DISPLAY
          018400         NOT-SUBCTL1-FORMAT
          018500         NOT-SUB1-FORMAT TO TRUE.
  209     018600     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
                *** Elementos CORRESPONDING para la instrucción 018600:
                ***       IN62
                ***       IN61
                ***       IN58
                ***       IN57
                ***       IN47
                ***       IN45
                *** Fin de elementos CORRESPONDING para la instrucción 018600
  210     018700     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1"
          018800     END-WRITE
  211     018900     READ EXISTING-ORDER-DISPLAY-FILE RECORD.
  212     019000     MOVE CORR SUBCTL1-I-INDIC TO INDICATOR-AREA.
                *** Elementos CORRESPONDING para la instrucción 019000:
                ***       IN97
                ***       IN98
                ***       IN57
                ***       IN58
                ***       IN61
                ***       IN47
                ***       IN62
                *** Fin de elementos CORRESPONDING para la instrucción 019000
          019100
  213     019200     PERFORM EXISTING-ORDER-INQUIRY
          019300         UNTIL END-OF-EXISTING-ORDER-INQUIRY.
          019400
  214     019500      CLOSE     ORDER-HEADER-FILE
```

Figura 119 (Parte 7 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ          AS400SYS   96/07/04 11:32:43         Página     9

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         019600                ORDER-DETAIL-FILE
         019700                CUSTOMER-MASTER-FILE
         019800                EXISTING-ORDER-DISPLAY-FILE.
  215    019900      STOP RUN.
         020000
         020100 EXISTING-ORDER-INQUIRY.
  216    020200     IF CONTINUE-DETAIL-LINES-DISPLAY THEN
  217    020300         PERFORM READ-NEXT-ORDER-DETAIL-RECORD
  218    020400         IF MORE-DETAIL-LINE-ITEMS-EXIST THEN
  219    020500             IF ORDERN OF ORDER-DETAIL-RECORD IS NOT EQUAL TO
         020600                ORDERN OF ORDER-HEADER-RECORD THEN
  220    020700                    SET DISPLAY-SUBFILE TO TRUE
  221    020800                    SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
         020900             ELSE
  222    021000                 PERFORM SUBFILE-SET-UP
         021100             END-IF
         021200         ELSE
  223    021300             SET DISPLAY-SUBFILE TO TRUE
  224    021400             SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
         021500         END-IF
         021600     ELSE
  225    021700         PERFORM ORDER-NUMBER-VALIDATION
         021800     END-IF
  226    021900     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
               *** Elementos CORRESPONDING para la instrucción 021900:
               ***       IN62
               ***       IN61
               ***       IN58
               ***       IN57
               ***       IN47
               ***       IN45
               *** Fin de elementos CORRESPONDING para la instrucción 021900
  227    022000     SET WRITE-DISPLAY TO TRUE.
  228    022100     SET SUBCTL1-FORMAT TO TRUE.
  229    022200     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1".
  230    022300     READ EXISTING-ORDER-DISPLAY-FILE RECORD.
  231    022400     MOVE CORR SUBCTL1-I-INDIC TO INDICATOR-AREA.
               *** Elementos CORRESPONDING para la instrucción 022400:
               ***       IN97
               ***       IN98
               ***       IN57
               ***       IN58
               ***       IN61
               ***       IN47
               ***       IN62
               *** Fin de elementos CORRESPONDING para la instrucción 022400
         022500
         022600 ORDER-NUMBER-VALIDATION.
  232    022700     PERFORM READ-ORDER-HEADER-FILE.
  233    022800     IF ORDER-EXIST THEN
  234    022900         PERFORM READ-CUSTOMER-MASTER-FILE
  235    023000         IF CUSTOMER-EXIST THEN
  236    023100             PERFORM READ-FIRST-ORDER-DETAIL-RECORD
  237    023200             IF DETAIL-LINES-FOR-ORDER-EXIST THEN
  238    023300                 PERFORM SUBFILE-SET-UP
         023400             END-IF
         023500         END-IF
```

Figura 119 (Parte 8 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ           AS400SYS   96/07/04 11:32:43         Página    10

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         023600     END-IF.                                                                          94/03/16
         023700
         023800 READ-ORDER-HEADER-FILE.
  239    023900     MOVE ORDERN OF SUBCTL1-I OF EXISTING-ORDER-DISPLAY-RECORD
         024000         TO ORDERN OF ORDER-HEADER-RECORD.
  240    024100     READ ORDER-HEADER-FILE
  241    024200         INVALID KEY SET ORDER-NOT-FOUND TO TRUE
         024300     END-READ.                                                                        94/03/16
         024400
         024500 READ-CUSTOMER-MASTER-FILE.
  242    024600     MOVE CUST OF ORDER-HEADER-RECORD
         024700         TO CUST OF CUSTOMER-MASTER-RECORD.
  243    024800     READ CUSTOMER-MASTER-FILE
  244    024900         INVALID KEY SET CUSTOMER-NOT-FOUND TO TRUE
         025000     END-READ.                                                                        94/03/16
         025100
         025200 READ-FIRST-ORDER-DETAIL-RECORD.
  245    025300     MOVE ORDERN OF ORDER-HEADER-RECORD
         025400         TO ORDERN OF ORDER-DETAIL-RECORD.
  246    025500     MOVE 1 TO LINNUM OF ORDER-DETAIL-RECORD.
  247    025600     READ ORDER-DETAIL-FILE
  248    025700         INVALID KEY SET NO-DETAIL-LINES-FOR-ORDER TO TRUE
         025800     END-READ.                                                                        94/03/16
         025900
         026000 SUBFILE-SET-UP.
  249    026100     SET CLEAR-SUBFILE TO TRUE.
  250    026200     MOVE CORR INDICATOR-AREA TO SUBCTL1-O-INDIC.
               *** Elementos CORRESPONDING para la instrucción 026200:
               ***       IN62
               ***       IN61
               ***       IN58
               ***       IN57
               ***       IN47
               ***       IN45
               *** Fin de elementos CORRESPONDING para la instrucción 026200
  251    026300     SET WRITE-DISPLAY TO TRUE.
  252    026400     SET SUBCTL1-FORMAT TO TRUE.
  253    026500     WRITE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUBCTL1"
         026600     END-WRITE
  254    026700     SET DISPLAY-SUBFILE-CONTROL TO TRUE.
  255    026800     PERFORM BUILD-DISPLAY-SUBFILE
         026900         UNTIL NO-MORE-DETAIL-LINE-ITEMS OR SUBFILE-IS-FULL.
  256    027000     MOVE CORR ORDHDR OF ORDER-HEADER-RECORD
               *** Elementos CORRESPONDING para la instrucción 027000:
               ***       CUST
               ***       ORDERN
               ***       ORDDAT
               ***       CUSORD
               ***       SHPVIA
               ***       ORDAMT
               ***       INVNUM
               ***       PRTDAT
               ***       ACTMTH
               ***       ACTYR
               ***       STATE
               *** Fin de elementos CORRESPONDING para la instrucción 027000
         027100         TO SUBCTL1-O OF EXISTING-ORDER-DISPLAY-RECORD.
```

Figura 119 (Parte 9 de 10). Ejemplo de un programa de consulta de pedidos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/ORDINQ          AS400SYS   96/07/04 11:32:43        Página    11

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


  257     027200     MOVE CORR CUSMST OF CUSTOMER-MASTER-RECORD
                *** Elementos CORRESPONDING para la instrucción 027200:
                ***       CUST
                ***       NAME
                ***       ADDR
                ***       CITY
                ***       STATE
                ***       ZIP
                *** Fin de elementos CORRESPONDING para la instrucción 027200
          027300         TO SUBCTL1-O OF EXISTING-ORDER-DISPLAY-RECORD.
  258     027400     MOVE ORDER-STATUS(ORDSTS) TO STSORD.
  259     027500     MOVE OPEN-STATUS(OPNSTS) TO STSOPN.
  260     027600     SET MORE-DETAIL-LINE-ITEMS-EXIST TO TRUE.
  261     027700     MOVE ZEROS TO SUBFILE-RECORD-NUMBER.
          027800
          027900 BUILD-DISPLAY-SUBFILE.
  262     028000     MOVE CORR ORDDTL OF ORDER-DETAIL-RECORD
                *** Elementos CORRESPONDING para la instrucción 028000:
                ***       ITEM
                ***       QTYORD
                ***       DESCRP
                ***       PRICE
                ***       EXTENS
                *** Fin de elementos CORRESPONDING para la instrucción 028000
          028100         TO SUB1 OF EXISTING-ORDER-DISPLAY-RECORD.
  263     028200     SET WRITE-DISPLAY TO TRUE.
  264     028300     SET SUB1-FORMAT TO TRUE.
  265     028400     ADD 1 TO SUBFILE-RECORD-NUMBER.
  266     028500     WRITE SUBFILE EXISTING-ORDER-DISPLAY-RECORD FORMAT IS "SUB1"
          028600     END-WRITE
  267     028700     IF SUBFILE-IS-FULL THEN
  268     028800         SET DISPLAY-SUBFILE TO TRUE
          028900     ELSE
  269     029000         PERFORM READ-NEXT-ORDER-DETAIL-RECORD
  270     029100         IF MORE-DETAIL-LINE-ITEMS-EXIST THEN
  271     029200             IF ORDERN OF ORDER-DETAIL-RECORD IS NOT EQUAL TO
          029300                ORDERN OF ORDER-HEADER-RECORD THEN
  272     029400                   SET DISPLAY-SUBFILE TO TRUE
  273     029500                   SET NO-MORE-DETAIL-LINE-ITEMS TO TRUE
          029600             END-IF
          029700         END-IF
          029800     END-IF.                                                                          94/03/16
          029900
          030000 READ-NEXT-ORDER-DETAIL-RECORD.
  274     030100     READ ORDER-DETAIL-FILE NEXT RECORD
  275     030200         AT END SET DISPLAY-SUBFILE TO TRUE
  276     030300                SET NO-MORE-DETAIL-LINE-ITEMS TO TRUE
          030400     END-READ.                                                                        94/03/16

                     * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 119 (Parte 10 de 10). Ejemplo de un programa de consulta de pedidos

La pantalla de solicitud de entrada de pedido inicial escrita en la estación de trabajo es la siguiente:

```
Conulta de pedidos existentes            Total  000000000
                                         Estado
Pedido 12400                             Situación
Fecha  000000                            Orden de cliente
N  cliente                               Inst. de envío
                                00000    Fecha de impresión 000000
                              Factura 00000                    MCV 00  Año 00
Artíc. Cant.Descripción                     Precio    Extensión
```

La pantalla que aparece si hay registros de detalles de pedido para el cliente cuyo número de pedido se ha entrado en la primera pantalla es la siguiente:

```
Consulta de pedidos existentes          Total  007426656
                                        Estado 7-INVOICED
Pedido 17924   ABC HARDWARE LTD.        Situación 2-CLOSED
Fecha 110896   123 ANYWHERE AVE.        Orden de cliente TESTCS17933001I
N  cliente 11200  TORONTO               Inst. envío      TRUCKCO
              ONT           M4K 0A0     Fecha de impresión 110896
                          Factura 17924                    MVC 12  Año 88
Artíc. Cant.Descripción                   Precio      Extensión
33001  003  TORQUE WRENCH 75LB 14 INCH     009115       273.45
33100  001  TORQUE WRENCH W/GAUGE 200 LB   015777       650.95
44529  004  WOOD CHISEL - 3 1/4            006840        56.87
44958  002  POWER DRILL 1/2 REV            008200       797.50
46102  001  WROUGHT IRON RAILING 4FTX6FT   007930       237.75
46201  001  WROUGHT IRON HAND RAIL 6FT     007178        77.35
47902  002  ESCUTCHEON BRASS 15X4 INCHES   044488       213.00
```

La pantalla que aparece si el archivo ORDHDRP no contiene ningún registro para el número de pedido entrado en la primera pantalla es la siguiente:

```
Conulta de pedidos existentes           Total  000000000
                                        Estado
Pedido 12400                            Situación
Fecha  000000                           Pedido del cliente
N  cliente                              Inst. de envío
                            00000       Fecha de impresión 000000
                          Factura 00000                    MCV 00  Año 00
Artíc. Cant.Descripción                    Precio    Extensión









Número de pedido no encontrado
```

Ejemplo de utilización de READ SUBFILE...NEXT MODIFIED y REWRITE SUBFILE en un programa de actualización de pagos

La Figura 122 en la página 462 muestra un ejemplo de programa de actualización de pagos, PAYUPDT. Si desea consultar las DDS relacionadas, vea la Figura 120 en la página 460 y la Figura 121 en la página 461. Si desea ver los ejemplos de pantalla, vaya a la página 474. Si desea consultar las DDS del archivo maestro de clientes, CUSMSTP, vea la Figura 102 en la página 406.

En este ejemplo, los pagos de los clientes están registrados.Al administrativo se le pide que entre uno o más números de clientes y el importe de crédito que se ha de anotar en la cuenta de cada cliente. El programa comprueba el número de cliente y acepta sin ninguna condición cualquier pago de un cliente existente que tenga facturas pendientes.Si como resultado del importe del pago realizado el cliente ha pagado dinero de más, el administrativo tiene la opción de aceptar o rechazar el pago. Si no existe ningún registro de cliente para un número de cliente, se emite

un mensaje de error.Se pueden entrar pagos hasta que el administrativo pulse F12.

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A** ESTE ES EL ARCHIVO LOGICO DE CABECERA DE PEDIDO -- ORDHDRL
     A
     A
     A                                      UNIQUE
     A          R ORDHDR                    PFILE(ORDHDRP)
     A*
     A            CUST
     A            INVNUM
     A            ORDERN
     A            ORDDAT
     A            CUSORD
     A            SHPVIA
     A            ORDSTS
     A            OPRNAM
     A            ORDAMT
     A            CUSTYP
     A            PRTDAT
     A            OPNSTS
     A            TOTLIN
     A            ACTMTH
     A            ACTYR
     A            STATE
     A            AMPAID
     A          K CUST
     A          K INVNUM
```

Figura 120. Ejemplo de especificación de descripción de datos para un programa de actualización de pagos - archivo lógico de pedidos

```
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
     A* ESTE ES EL ARCHIVO DE DISPOSITIVO DE PANTALLA DE PAYUPDT -- PAYUPDTD
     A* ACTUALIZACION INTERACTIVA DE PAGOS DE CUENTAS POR COBRAR
     A*
     A
     A          R SUBFILE1                  SFL
     A                                      TEXT('SUBARCHIVO DE PAGOS CLIENTE'
     A*
     A            ACPPMT         4A  I  5  4TEXT('ACEPTAR PAGO')
     A                                      VALUES('*YES' '*NO')
     A  51                                  DSPATR(RI MDT)
     A N51                                  DSPATR(ND PR)
     A*
     A            CUST           5   B  5 15TEXT('NÚMERO CLIENTE')
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            AMPAID         8 02B  5 24TEXT('IMPORTE PAGADO')
     A                                      CHECK(FE)
     A                                      AUTO(RAB)
     A                                      CMP(GT 0)
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(PR)
     A*
     A            ECPMSG        31A  O  5 37TEXT('MENSAJE DE EXCEPCIÓN')
     A  52                                  DSPATR(RI)
     A  53                                  DSPATR(ND)
     A  54                                  DSPATR(BL)
     A*
     A            OVRPMT         8Y 2O  5 70TEXT('DINERO PAGADO DE MÁS')
     A                                      EDTCDE(1)
     A  55                                  DSPATR(BL)
     A N56                                  DSPATR(ND)
     A*
     A            STSCDE         1A  H      TEXT('CÓDIGO DE ESTADO')
     A          R CONTROL1                  TEXT('SUBARCHIVO DE CONTROL')
     A                                      SFLCTL(SUBFILE1)
     A                                      SFLSIZ(17)
     A                                      SFLPAG(17)
     A  61                                  SFLCLR
     A  62                                  SFLDSP
     A  62                                  SFLDSPCTL
     A                                      OVERLAY
     A                                      LOCK
     A*
     A                                      HELP(99 'TECLA AYUDA')
     A                                      CA12(98 'FIN ACTUALIZACIÓN PAGOS')
     A                                      CA11(97 'IGNORAR ENTRADA')
     A*
     A  99                                  SFLMSG(' F11 - IGNORAR ENTRADA NO +
     A                                                 VÁLIDA F12 - FIN DE +
     A                                      ACTUALIZACIÓN PAGOS')
     A*
     A                                   1  2'SOLICITUD ACTUALIZACIÓN PAGOS CLIENTES'
     A                                   1 65'FECHA'
     A                                   1 71DATE EDTCDE(Y)
     A  63                               3  2'ACEPTAR'
     A  63                               4  2'PAGO'
     A                                   3 14'CLIENTE'
     A                                   3 26'IMPORTE'
     A  64                               3 37'MENSAJE DE EXCEPCIÓN'
     A*
     A          R MESSAGE1                  TEXT('REGISTRO DE MENSAJES')
     A                                      OVERLAY
     A                                      LOCK
     A*
     A  71                              24  2' VALORES DE ACEPTAR PAGO: (*NO *YES)
                                            DSPATR(RI)
```

Figura 121. Ejemplo de especificación de descripción de datos para un programa de actualización de pagos - archivo de dispositivo de pantalla

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


         000010 PROCESS APOST
    1    000020 IDENTIFICATION DIVISION.
    2    000030 PROGRAM-ID.  PAYUPDT.
         000040
    3    000050 ENVIRONMENT DIVISION.
    4    000060   CONFIGURATION SECTION.
    5    000070   SOURCE-COMPUTER. IBM-AS400.
    6    000080   OBJECT-COMPUTER. IBM-AS400.
    7    000090   INPUT-OUTPUT SECTION.
    8    000100   FILE-CONTROL.
    9    000110     SELECT CUSTOMER-INVOICE-FILE
   10    000120         ASSIGN TO DATABASE-ORDHDRL
   11    000130         ORGANIZATION IS INDEXED
   12    000140         ACCESS MODE IS SEQUENTIAL
   13    000150         RECORD KEY IS COMP-KEY
   14    000160         FILE STATUS IS STATUS-CODE-ONE.
   15    000170     SELECT CUSTOMER-MASTER-FILE
   16    000180         ASSIGN TO DATABASE-CUSMSTP
   17    000190         ORGANIZATION IS INDEXED
   18    000200         ACCESS IS RANDOM
   19    000210         RECORD KEY IS CUST OF CUSTOMER-MASTER-RECORD.
   20    000220     SELECT PAYMENT-UPDATE-DISPLAY-FILE
   21    000230         ASSIGN TO WORKSTATION-PAYUPDTD
   22    000240         ORGANIZATION IS TRANSACTION
   23    000250         ACCESS IS DYNAMIC
   24    000260         RELATIVE KEY IS REL-NUMBER
   25    000270         FILE STATUS IS STATUS-CODE-ONE
   26    000280         CONTROL-AREA IS WS-CONTROL.
         000290
   27    000300 DATA DIVISION.
   28    000310 FILE SECTION.
   29    000320 FD  CUSTOMER-INVOICE-FILE.
   30    000330 01  CUSTOMER-INVOICE-RECORD.
         000340     COPY DDS-ORDHDR OF ORDHDRL.
        +000001*    FORMATO E-S:ORDHDR     DEL ARCH. ORDHDRL    DE LA BIBL. TESTLIB          ORDHDR
        +000002*                                                                             ORDHDR
        +000003*          CLAVE PROPORCIONADA POR USUARIO CON CLAUSULA RECORD KEY             ORDHDR
   31   +000004       05  ORDHDR.                                                           ORDHDR
   32   +000005           06 CUST                  PIC X(5).                               ORDHDR
        +000006*                  NUMERO DE CLIENTE                                        ORDHDR
   33   +000007           06 INVNUM                PIC S9(5)       COMP-3.                 ORDHDR
        +000008*                  NUMERO DE FACTURA                                        ORDHDR
   34   +000009           06 ORDERN                PIC S9(5)       COMP-3.                 ORDHDR
        +000010*                  NUMERO DE PEDIDO                                         ORDHDR
   35   +000011           06 ORDDAT                PIC S9(6)       COMP-3.                 ORDHDR
        +000012*                  FECHA DE ENTRADA DEL PEDIDO                              ORDHDR
   36   +000013           06 CUSORD                PIC X(15).                              ORDHDR
        +000014*                  NUMERO DE ORDEN DE COMPRA DEL CLIENTE                    ORDHDR
   37   +000015           06 SHPVIA                PIC X(15).                              ORDHDR
        +000016*                  INSTRUCCIONES DE ENVIO                                   ORDHDR
   38   +000017           06 ORDSTS                PIC S9(1)       COMP-3.                 ORDHDR
        +000018*                  ESTADO DEL PEDIDO 1PCS 2CNT  3CHK 4RDY 5PRT 6PCK         ORDHDR
   39   +000019           06 OPRNAM                PIC X(10).                              ORDHDR
        +000020*                  OPERADOR QUE HA ENTRADO EL PEDIDO                        ORDHDR
   40   +000021           06 ORDAMT                PIC S9(6)V9(2)  COMP-3.                 ORDHDR
```

Figura 122 (Parte 1 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


       +000022*                 IMPORTE DEL PEDIDO                                        ORDHDR
  41   +000023           06 CUSTYP                PIC S9(1)       COMP-3.                 ORDHDR
       +000024*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT              ORDHDR
  42   +000025           06 PRTDAT                PIC S9(6)       COMP-3.                 ORDHDR
       +000026*                 FECHA DE IMPRESION DEL PEDIDO                             ORDHDR
  43   +000027           06 OPNSTS                PIC S9(1)       COMP-3.                 ORDHDR
       +000028*                 ESTADO DEL PEDIDO 1=OPEN 2= CLOSE 3=CANCEL                ORDHDR
  44   +000029           06 TOTLIN                PIC S9(3)       COMP-3.                 ORDHDR
       +000030*                 LINEAS DE DETALLE EN TOTAL DEL PEDIDO                     ORDHDR
  45   +000031           06 ACTMTH                PIC S9(2)       COMP-3.                 ORDHDR
       +000032*                 MES CONTABLE DE LA VENTA                                  ORDHDR
  46   +000033           06 ACTYR                 PIC S9(2)       COMP-3.                 ORDHDR
       +000034*                 EJERCICIO CONTABLE DE LA VENTA                            ORDHDR
  47   +000035           06 STATE                 PIC X(2).                               ORDHDR
       +000036*                 PROVINCIA                                                 ORDHDR
  48   +000037           06 AMPAID                PIC S9(6)V9(2)  COMP-3.                 ORDHDR
       +000038*                 IMPORTE PAGADO                                            ORDHDR
  49    000350 66  COMP-KEY RENAMES CUST THRU INVNUM.
        000360
  50    000370 FD  CUSTOMER-MASTER-FILE.
  51    000380 01  CUSTOMER-MASTER-RECORD.
        000390     COPY DDS-CUSMST OF CUSMSTP.
       +000001*    FORMATO E-S:CUSMST     DEL ARCH. CUSMSTP    DE LA BIBL. TESTLIB          CUSMST
       +000002*                          REGISTRO MAESTRO DE CLIENTES                       CUSMST
       +000003*          CLAVE PROPORCIONADA POR USUARIO CON CLAUSULA RECORD KEY             CUSMST
  52   +000004       05  CUSMST.                                                           CUSMST
  53   +000005           06 CUST                  PIC X(5).                               CUSMST
       +000006*                 NUMERO DE CLIENTE                                         CUSMST
  54   +000007           06 NAME                  PIC X(25).                              CUSMST
       +000008*                 NOMBRE DEL CLIENTE                                        CUSMST
  55   +000009           06 ADDR                  PIC X(20).                              CUSMST
       +000010*                 DIRECCION DEL CLIENTE                                     CUSMST
  56   +000011           06 CITY                  PIC X(20).                              CUSMST
       +000012*                 LOCALIDAD DEL CLIENTE                                     CUSMST
  57   +000013           06 STATE                 PIC X(2).                               CUSMST
       +000014*                 PROVINCIA                                                 CUSMST
  58   +000015           06 ZIP                   PIC S9(5)       COMP-3.                 CUSMST
       +000016*                 CODIGO POSTAL                                             CUSMST
  59   +000017           06 SRHCOD                PIC X(6).                               CUSMST
       +000018*                 CODIGO DE BUSQUEDA DE NUMERO DE CLIENTE                   CUSMST
  60   +000019           06 CUSTYP                PIC S9(1)       COMP-3.                 CUSMST
       +000020*                 TIPO DE CLIENTE 1=GOV 2=SCH 3=BUS 4=PVT 5=OT              CUSMST
  61   +000021           06 ARBAL                 PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000022*                 SALDO DE CUENTAS POR COBRAR                               CUSMST
  62   +000023           06 ORDBAL                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000024*                 IMPORTE DE CC EN ARCHIVO DE PEDIDOS                       CUSMST
  63   +000025           06 LSTAMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000026*                 ULTIMO IMPORTE PAGADO EN CC                               CUSMST
  64   +000027           06 LSTDAT                PIC S9(6)       COMP-3.                 CUSMST
       +000028*                 ULTIMA FECHA DE PAGO EN CC                                CUSMST
  65   +000029           06 CRDLMT                PIC S9(6)V9(2)  COMP-3.                 CUSMST
       +000030*                 LIMITE DE CREDITO DEL CLIENTE                             CUSMST
  66   +000031           06 SLSYR                 PIC S9(8)V9(2)  COMP-3.                 CUSMST
       +000032*                 VENTAS DEL CLIENTE EN EL AÑO ACTUAL                       CUSMST
  67   +000033           06 SLSLYR                PIC S9(8)V9(2)  COMP-3.                 CUSMST
       +000034*                 VENTAS DEL CLIENTE EL AÑO PASADO                          CUSMST
        000400
```

Figura 122 (Parte 2 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     4

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


   68     000410 FD  PAYMENT-UPDATE-DISPLAY-FILE.
   69     000420 01  PAYMENT-UPDATE-DISPLAY-RECORD.
          000430     COPY DDS-ALL-FORMATS OF PAYUPDTD.
   70    +000001       05  PAYUPDTD-RECORD PIC X(59).                                      <-ALL-FMTS
         +000002*  FORMATO ENTRADA:SUBFILE1     DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB     <-ALL-FMTS
         +000003*                          SUBARCHIVO PARA PAGO DE CLIENTE                  <-ALL-FMTS
   71    +000004       05  SUBFILE1-I    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
   72    +000005           06 ACPPMT                PIC X(4).                              <-ALL-FMTS
         +000006*                 ACEPTAR PAGO                                            <-ALL-FMTS
   73    +000007           06 CUST                  PIC X(5).                              <-ALL-FMTS
         +000008*                 NUMERO DE CLIENTE                                       <-ALL-FMTS
   74    +000009           06 AMPAID                PIC S9(6)V9(2).                        <-ALL-FMTS
         +000010*                 IMPORTE PAGADO                                          <-ALL-FMTS
   75    +000011           06 ECPMSG                PIC X(31).                             <-ALL-FMTS
         +000012*                 MENSAJE DE EXCEPCION                                    <-ALL-FMTS
   76    +000013           06 OVRPMT                PIC S9(6)V9(2).                        <-ALL-FMTS
         +000014*                 DINERO PAGADO DE MAS                                    <-ALL-FMTS
   77    +000015           06 STSCDE                PIC X(1).                              <-ALL-FMTS
         +000016*                 CODIGO DE ESTADO                                        <-ALL-FMTS
         +000017*  FORMATO SALIDA:SUBFILE1     DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB     <-ALL-FMTS
         +000018*                          SUBARCHIVO PARA PAGO DE CLIENTE                  <-ALL-FMTS
   78    +000019       05  SUBFILE1-O    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
   79    +000020           06 SUBFILE1-O-INDIC.                                            <-ALL-FMTS
   80    +000021                07 IN51             PIC 1  INDIC 51.                      <-ALL-FMTS
   81    +000022                07 IN52             PIC 1  INDIC 52.                      <-ALL-FMTS
   82    +000023                07 IN53             PIC 1  INDIC 53.                      <-ALL-FMTS
   83    +000024                07 IN54             PIC 1  INDIC 54.                      <-ALL-FMTS
   84    +000025                07 IN55             PIC 1  INDIC 55.                      <-ALL-FMTS
   85    +000026                07 IN56             PIC 1  INDIC 56.                      <-ALL-FMTS
   86    +000027           06 CUST                  PIC X(5).                              <-ALL-FMTS
         +000028*                 NUMERO DE CLIENTE                                       <-ALL-FMTS
   87    +000029           06 AMPAID                PIC S9(6)V9(2).                        <-ALL-FMTS
         +000030*                 IMPORTE PAGADO                                          <-ALL-FMTS
   88    +000031           06 ECPMSG                PIC X(31).                             <-ALL-FMTS
         +000032*                 MENSAJE DE EXCEPCION                                    <-ALL-FMTS
   89    +000033           06 OVRPMT                PIC S9(6)V9(2).                        <-ALL-FMTS
         +000034*                 DINERO PAGADO DE MAS                                    <-ALL-FMTS
   90    +000035           06 STSCDE                PIC X(1).                              <-ALL-FMTS
         +000036*                 CODIGO DE ESTADO                                        <-ALL-FMTS
         +000037*  FORMATO ENTRADA:CONTROL1     DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB     <-ALL-FMTS
         +000038*                          CONTROL DE SUBARCHIVO                           <-ALL-FMTS
   91    +000039       05  CONTROL1-I    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
   92    +000040           06 CONTROL1-I-INDIC.                                            <-ALL-FMTS
   93    +000041                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
         +000042*                        TECLA AYUDA                                      <-ALL-FMTS
   94    +000043                07 IN98             PIC 1  INDIC 98.                      <-ALL-FMTS
         +000044*                        FIN DE LA ACTUALIZACION DE PAGOS                 <-ALL-FMTS
   95    +000045                07 IN97             PIC 1  INDIC 97.                      <-ALL-FMTS
         +000046*                        HACER CASO OMISO DE LO ENTRADO                   <-ALL-FMTS
         +000047*  FORMATO SALIDA:CONTROL1     DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB     <-ALL-FMTS
         +000048*                          CONTROL DE SUBARCHIVO                           <-ALL-FMTS
   96    +000049       05  CONTROL1-O    REDEFINES PAYUPDTD-RECORD.                       <-ALL-FMTS
   97    +000050           06 CONTROL1-O-INDIC.                                            <-ALL-FMTS
   98    +000051                07 IN61             PIC 1  INDIC 61.                      <-ALL-FMTS
   99    +000052                07 IN62             PIC 1  INDIC 62.                      <-ALL-FMTS
  100    +000053                07 IN99             PIC 1  INDIC 99.                      <-ALL-FMTS
         +000054*                        TECLA AYUDA                                      <-ALL-FMTS
```

Figura 122 (Parte 3 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     5

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


  101    +000055               07 IN63             PIC 1  INDIC 63.                          <-ALL-FMTS
  102    +000056               07 IN64             PIC 1  INDIC 64.                          <-ALL-FMTS
         +000057* FORMATO ENTRADA:MESSAGE1    DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB       <-ALL-FMTS
         +000058*                          REGISTRO DE MENSAJE                                 <-ALL-FMTS
         +000059*      05  MESSAGE1-I    REDEFINE REGISTRO PAYUPDTD.                           <-ALL-FMTS
         +000060* FORMATO SALIDA:MESSAGE1     DEL ARCH. PAYUPDTD    DE LA BIBL. TESTLIB       <-ALL-FMTS
         +000061*                          REGISTRO DE MENSAJE                                 <-ALL-FMTS
  103    +000062       05  MESSAGE1-O    REDEFINE REGISTRO  PAYUPDTD.                            <-ALL-FMTS
  104    +000063           06 MESSAGE1-O-INDIC.                                                 <-ALL-FMTS
  105    +000064               07 IN71             PIC 1  INDIC 71.                          <-ALL-FMTS
          000440
  106     000450 WORKING-STORAGE SECTION.
          000460
  107     000470 01  REL-NUMBER                   PIC 9(05)
          000480                                    VALUE ZEROS.
          000490
  108     000500 01  WS-CONTROL.
  109     000510     05  WS-IND                   PIC X(02).
  110     000520     05  WS-FORMAT                PIC X(10).
  111     000530 01  SYSTEM-DATE.
  112     000540     05  SYSTEM-YEAR              PIC 99.
  113     000550     05  SYSTEM-MONTH             PIC 99.
  114     000560     05  SYSTEM-DAY               PIC 99.
  115     000570 01  PROGRAM-DATE.
  116     000580     05  PROGRAM-MONTH            PIC 99.
  117     000590     05  PROGRAM-DAY              PIC 99.
  118     000600     05  PROGRAM-YEAR             PIC 99.
  119     000610 01  FILE-DATE REDEFINES PROGRAM-DATE
          000620                                   PIC S9(6).
  120     000630 01  EXCEPTION-STATUS.
  121     000640     05  STATUS-CODE-ONE          PIC XX.
  122     000650         88  SUBFILE-IS-FULL      VALUE '0M'.
  123     000660 01  EXCEPTION-MESSAGES.
  124     000670     05  MESSAGE-ONE              PIC X(31)
          000680         VALUE 'EL CLIENTE NO EXISTE          '.
  125     000690     05  MESSAGE-TWO              PIC X(31)
          000700         VALUE 'NO EXISTEN FACTURAS PARA CLIENTE'.
  126     000710     05  MESSAGE-THREE            PIC X(31)
          000720         VALUE 'CLIENTE TIENE SALDO A FAVOR DE'.
  127     000730 01  PROGRAM-VARIABLES.
  128     000740     05  AMOUNT-OWED              PIC S9(6)V99.
  129     000750     05  AMOUNT-PAID              PIC S9(6)V99.
  130     000760     05  INVOICE-BALANCE          PIC S9(6)V99.
  131     000770 01  ERRPGM-PARAMETERS.
  132     000780     05  DISPLAY-PARAMETER        PIC X(8)
          000790                                    VALUE 'PAYUPDTD'.
  133     000800     05  DUMMY-ONE                PIC X(6)
          000810                                    VALUE SPACES.
  134     000820     05  DUMMY-TWO                PIC X(6)
          000830                                    VALUE SPACES.
  135     000840     05  STATUS-CODE-TWO.
  136     000850        10  PRIMARY               PIC X(1).
  137     000860        10  SECONDARY             PIC X(1).
  138     000870        10  FILLER                PIC X(5)
          000880                                    VALUE SPACES.
  139     000890     05  DUMMY-THREE              PIC X(10)
          000900                                    VALUE SPACES.
```

Figura 122 (Parte 4 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     6

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         000910
  140    000920 01  SWITCH-AREA.
  141    000930    05  SW01                       PIC 1.
  142    000940       88  WRITE-DISPLAY                VALUE B'1'.
  143    000950       88  READ-DISPLAY                 VALUE B'0'.
  144    000960    05  SW02                       PIC 1.
  145    000970       88  SUBFILE1-FORMAT              VALUE B'1'.
  146    000980       88  NOT-SUBFILE1-FORMAT          VALUE B'0'.
  147    000990    05  SW03                       PIC 1.
  148    001000       88  CONTROL1-FORMAT              VALUE B'1'.
  149    001010       88  NOT-CONTROL1-FORMAT          VALUE B'0'.
  150    001020    05  SW04                       PIC 1.
  151    001030       88  NO-MORE-TRANSACTIONS-EXIST   VALUE B'1'.
  152    001040       88  TRANSACTIONS-EXIST           VALUE B'0'.
  153    001050    05  SW05                       PIC 1.
  154    001060       88  CUSTOMER-NOT-FOUND           VALUE B'1'.
  155    001070       88  CUSTOMER-EXIST               VALUE B'0'.
  156    001080    05  SW06                       PIC 1.
  157    001090       88  NO-MORE-INVOICES-EXIST       VALUE B'1'.
  158    001100       88  CUSTOMER-INVOICE-EXIST       VALUE B'0'.
  159    001110    05  SW07                       PIC 1.
  160    001120       88  NO-MORE-PAYMENT-EXIST        VALUE B'1'.
  161    001130       88  PAYMENT-EXIST                VALUE B'0'.
  162    001140    05  SW08                       PIC 1.
  163    001150       88  INPUT-ERRORS-EXIST           VALUE B'1'.
  164    001160       88  NO-INPUT-ERRORS-EXIST        VALUE B'0'.
  165    001170    05  SW09                       PIC 1.
  166    001180       88  OVER-PAYMENT-DISPLAYED-ONCE  VALUE B'1'.
  167    001190       88  OVER-PAYMENT-NOT-DISPLAYED   VALUE B'0'.
         001200
  168    001210 01  INDICATOR-AREA.
  169    001220    05  IN99                       PIC 1 INDIC 99.
  170    001230       88 HELP-IS-NEEDED              VALUE B'1'.
  171    001240       88 HELP-IS-NOT-NEEDED          VALUE B'0'.
  172    001250    05  IN98                       PIC 1 INDIC 98.
  173    001260       88 END-OF-PAYMENT-UPDATE       VALUE B'1'.
  174    001270    05  IN97                       PIC 1 INDIC 97.
  175    001280       88 IGNORE-INPUT                VALUE B'1'.
  176    001290    05  IN51                       PIC 1 INDIC 51.
  177    001300       88 DISPLAY-ACCEPT-PAYMENT      VALUE B'1'.
  178    001310       88 DO-NOT-DISPLAY-ACCEPT-PAYMENT  VALUE B'0'.
  179    001320    05  IN52                       PIC 1 INDIC 52.
  180    001330       88 REVERSE-FIELD-IMAGE         VALUE B'1'.
  181    001340       88 DO-NOT-REVERSE-FIELD-IMAGE  VALUE B'0'.
  182    001350    05  IN53                       PIC 1 INDIC 53.
  183    001360       88 DO-NOT-DISPLAY-FIELD        VALUE B'1'.
  184    001370       88 DISPLAY-FIELD               VALUE B'0'.
  185    001380    05  IN54                       PIC 1 INDIC 54.
  186    001390       88 PROTECT-INPUT-FIELD         VALUE B'1'.
  187    001400       88 DO-NOT-PROTECT-INPUT-FIELD  VALUE B'0'.
  188    001410    05  IN55                       PIC 1 INDIC 55.
  189    001420       88 MAKE-FIELD-BLINK            VALUE B'1'.
  190    001430       88 DO-NOT-MAKE-FIELD-BLINK     VALUE B'0'.
  191    001440    05  IN56                       PIC 1 INDIC 56.
  192    001450       88 DISPLAY-OVER-PAYMENT        VALUE B'1'.
  193    001460       88 DO-NOT-DISPLAY-OVER-PAYMENT VALUE B'0'.
  194    001470    05  IN61                       PIC 1 INDIC 61.
```

Figura 122 (Parte 5 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     7

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


  195     001480       88 CLEAR-SUBFILE                  VALUE B'1'.
  196     001490       88 DO-NOT-CLEAR-SUBFILE           VALUE B'0'.
  197     001500    05  IN62                       PIC 1 INDIC 62.
  198     001510       88 DISPLAY-SCREEN                 VALUE B'1'.
  199     001520       88 DO-NOT-DISPLAY-SCREEN          VALUE B'0'.
  200     001530    05  IN63                       PIC 1 INDIC 63.
  201     001540       88 DISPLAY-ACCEPT-HEADING         VALUE B'1'.
  202     001550       88 DO-NOT-DISPLAY-ACCEPT-HEADING  VALUE B'0'.
  203     001560    05  IN64                       PIC 1 INDIC 64.
  204     001570       88 DISPLAY-EXCEPTION              VALUE B'1'.
  205     001580       88 DO-NOT-DISPLAY-EXCEPTION       VALUE B'0'.
  206     001590    05  IN71                       PIC 1 INDIC 71.
  207     001600       88 DISPLAY-ACCEPT-MESSAGE         VALUE B'1'.
  208     001610       88 DO-NOT-DISPLAY-ACCEPT-MESSAGE  VALUE B'0'.
          001620
  209     001630 PROCEDURE DIVISION.
          001640
  210     001650 DECLARATIVES.
          001660
          001670 TRANSACTION-ERROR SECTION.
          001680     USE AFTER STANDARD ERROR PROCEDURE
          001690         PAYMENT-UPDATE-DISPLAY-FILE.
          001700 WORK-STATION-ERROR-HANDLER.
  211     001710     IF NOT (SUBFILE-IS-FULL) THEN
  212     001720       DISPLAY 'ERROR EN PAYMENT-DISPLAY' STATUS-CODE-ONE
          001730     END-IF.
          001740 END DECLARATIVES.
          001750
          001760 MAIN-PROGRAM SECTION.
          001770 MAINLINE.
  213     001780     OPEN I-O      CUSTOMER-INVOICE-FILE
          001790                   CUSTOMER-MASTER-FILE
          001800                   PAYMENT-UPDATE-DISPLAY-FILE.
          001810
  214     001820     MOVE ALL B'0' TO INDICATOR-AREA
          001830                    SWITCH-AREA.
  215     001840     ACCEPT SYSTEM-DATE FROM DATE
          001850     END-ACCEPT.
  216     001860     MOVE SYSTEM-YEAR  TO PROGRAM-YEAR.
  217     001870     MOVE SYSTEM-MONTH TO PROGRAM-MONTH.
  218     001880     MOVE SYSTEM-DATE  TO PROGRAM-DAY.
  219     001890     SET WRITE-DISPLAY
          001900         CONTROL1-FORMAT
          001910         DO-NOT-DISPLAY-OVER-PAYMENT
          001920         DO-NOT-PROTECT-INPUT-FIELD
          001930         DO-NOT-REVERSE-FIELD-IMAGE
          001940         DO-NOT-MAKE-FIELD-BLINK
          001950         CLEAR-SUBFILE TO TRUE.
  220     001960     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
                *** Elementos CORRESPONDING para la instrucción 001960:
                ***       IN99
                ***       IN61
                ***       IN62
                ***       IN63
                ***       IN64
                *** Fin de elementos CORRESPONDING para la instrucción 001960
  221     001970     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
```

Figura 122 (Parte 6 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     8

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         001980         FORMAT IS 'CONTROL1'
         001990     END-WRITE.
  222    002000     SET DO-NOT-CLEAR-SUBFILE TO TRUE.
  223    002010     PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES.
  224    002020     SET DISPLAY-SCREEN TO TRUE.
  225    002030     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
               *** Elementos CORRESPONDING para la instrucción 002030:
               ***       IN99
               ***       IN61
               ***       IN62
               ***       IN63
               ***       IN64
               *** Fin de elementos CORRESPONDING para la instrucción 002030
  226    002040     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
         002050         FORMAT IS 'CONTROL1'
         002060     END-WRITE.
  227    002070     READ PAYMENT-UPDATE-DISPLAY-FILE RECORD
         002080         FORMAT IS 'CONTROL1'
         002090     END-READ.
  228    002100     MOVE CORR CONTROL1-I-INDIC TO INDICATOR-AREA.
               *** Elementos CORRESPONDING para la instrucción 002100:
               ***       IN99
               ***       IN98
               ***       IN97
               *** Fin de elementos CORRESPONDING para la instrucción 002100
         002110
  229    002120     PERFORM PROCESS-TRANSACTION-FILE
         002130       UNTIL END-OF-PAYMENT-UPDATE.
         002140
  230    002150     CLOSE CUSTOMER-INVOICE-FILE
         002160           CUSTOMER-MASTER-FILE
         002170           PAYMENT-UPDATE-DISPLAY-FILE.
  231    002180     STOP RUN.
         002190
         002200 PROCESS-TRANSACTION-FILE.
  232    002210     IF HELP-IS-NOT-NEEDED THEN
  233    002220        IF  IGNORE-INPUT THEN
  234    002230            SET WRITE-DISPLAY
         002240                CONTROL1-FORMAT
         002250                CLEAR-SUBFILE
         002260                DISPLAY-FIELD
         002270                DO-NOT-DISPLAY-OVER-PAYMENT
         002280                DO-NOT-PROTECT-INPUT-FIELD
         002290                DO-NOT-REVERSE-FIELD-IMAGE
         002300                DO-NOT-DISPLAY-ACCEPT-PAYMENT
         002310                DO-NOT-DISPLAY-ACCEPT-HEADING
         002320                DO-NOT-DISPLAY-ACCEPT-MESSAGE
         002330                DO-NOT-MAKE-FIELD-BLINK TO TRUE
  235    002340            MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC
               *** Elementos CORRESPONDING para la instrucción 002340:
               ***       IN99
               ***       IN61
               ***       IN62
               ***       IN63
               ***       IN64
               *** Fin de elementos CORRESPONDING para la instrucción 002340
  236    002350            WRITE PAYMENT-UPDATE-DISPLAY-RECORD
```

Figura 122 (Parte 7 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página     9

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         002360                FORMAT IS 'CONTROL1'
         002370             END-WRITE
  237    002380             SET DO-NOT-CLEAR-SUBFILE TO TRUE
  238    002390             MOVE 0 TO REL-NUMBER
  239    002400             PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES
         002410          ELSE
  240    002420             SET TRANSACTIONS-EXIST
         002430                 DO-NOT-DISPLAY-ACCEPT-HEADING
         002440                 DO-NOT-DISPLAY-ACCEPT-MESSAGE
         002450                 DO-NOT-DISPLAY-EXCEPTION TO TRUE
  241    002460             PERFORM READ-MODIFIED-SUBFILE-RECORD
  242    002470             PERFORM TRANSACTION-VALIDATION
         002480                 UNTIL NO-MORE-TRANSACTIONS-EXIST
  243    002490             SET NO-INPUT-ERRORS-EXIST TO TRUE
  244    002500             PERFORM TEST-FOR-RECORD-INPUT-ERRORS
         002510                 VARYING REL-NUMBER
         002520                 FROM    1
         002530                 BY      1
         002540                 UNTIL   REL-NUMBER IS GREATER THAN 17
         002550                 OR      INPUT-ERRORS-EXIST
  245    002560             IF  NO-INPUT-ERRORS-EXIST THEN
  246    002570                 IF  OVER-PAYMENT-DISPLAYED-ONCE THEN
  247    002580                     SET WRITE-DISPLAY
         002590                         CONTROL1-FORMAT
         002600                         DO-NOT-DISPLAY-OVER-PAYMENT
         002610                         DO-NOT-PROTECT-INPUT-FIELD
         002620                         DO-NOT-REVERSE-FIELD-IMAGE
         002630                         DO-NOT-MAKE-FIELD-BLINK
         002640                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
         002650                         DO-NOT-DISPLAY-ACCEPT-HEADING
         002660                         DO-NOT-DISPLAY-ACCEPT-MESSAGE
         002670                         DO-NOT-DISPLAY-EXCEPTION
         002680                         CLEAR-SUBFILE
         002690                         DISPLAY-FIELD
         002700                           TO TRUE
  248    002710                     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC
               *** Elementos CORRESPONDING para la instrucción 002710:
               ***       IN99
               ***       IN61
               ***       IN62
               ***       IN63
               ***       IN64
               *** Fin de elementos CORRESPONDING para la instrucción 002710
  249    002720                     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
         002730                         FORMAT IS 'CONTROL1'
         002740                     END-WRITE
  250    002750                     SET DO-NOT-CLEAR-SUBFILE TO TRUE
  251    002760                     MOVE 0 TO REL-NUMBER
  252    002770                     PERFORM INITIALIZE-SUBFILE-RECORD 17 TIMES
         002780                 ELSE
  253    002790                     SET OVER-PAYMENT-DISPLAYED-ONCE TO TRUE
         002800                 END-IF
         002810             END-IF
         002820         END-IF
         002830     END-IF.
  254    002840     SET WRITE-DISPLAY, DISPLAY-SCREEN TO TRUE.
  255    002850     MOVE CORR INDICATOR-AREA TO MESSAGE1-O-INDIC.
```

Figura 122 (Parte 8 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página    10

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


               *** Elementos CORRESPONDING para la instrucción 002850:
               ***       IN71
               *** Fin de elementos CORRESPONDING para la instrucción 002850
  256     002860     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          002870         FORMAT IS 'MESSAGE1'
          002880     END-WRITE.
  257     002890     SET WRITE-DISPLAY, CONTROL1-FORMAT TO TRUE.
  258     002900     MOVE CORR INDICATOR-AREA TO CONTROL1-O-INDIC.
               *** Elementos CORRESPONDING para la instrucción 002900:
               ***       IN99
               ***       IN61
               ***       IN62
               ***       IN63
               ***       IN64
               *** Fin de elementos CORRESPONDING para la instrucción 002900
  259     002910     WRITE PAYMENT-UPDATE-DISPLAY-RECORD
          002920         FORMAT IS 'CONTROL1'
          002930     END-WRITE.
  260     002940     READ PAYMENT-UPDATE-DISPLAY-FILE RECORD
          002950         FORMAT IS 'CONTROL1'
          002960     END-READ.
  261     002970     MOVE CORR CONTROL1-I-INDIC TO INDICATOR-AREA.
               *** Elementos CORRESPONDING para la instrucción 002970:
               ***       IN99
               ***       IN98
               ***       IN97
               *** Fin de elementos CORRESPONDING para la instrucción 002970
          002980
          002990 READ-MODIFIED-SUBFILE-RECORD.
  262     003000     READ SUBFILE PAYMENT-UPDATE-DISPLAY-FILE
          003010         NEXT MODIFIED RECORD FORMAT IS 'SUBFILE1'
  263     003020         AT END SET NO-MORE-TRANSACTIONS-EXIST TO TRUE
          003030     END-READ.
          003040
          003050 TEST-FOR-RECORD-INPUT-ERRORS.
  264     003060     READ SUBFILE PAYMENT-UPDATE-DISPLAY-FILE RECORD
          003070         FORMAT IS 'SUBFILE1'
          003080     END-READ.
  265     003090     IF  STSCDE OF SUBFILE1-I IS EQUAL TO '1' THEN
  266     003100         SET INPUT-ERRORS-EXIST TO TRUE
          003110     END-IF.
          003120
          003130 TRANSACTION-VALIDATION.
  267     003140     MOVE CUST OF SUBFILE1-I OF PAYMENT-UPDATE-DISPLAY-RECORD
          003150         TO CUST OF CUSTOMER-MASTER-RECORD.
  268     003160     SET CUSTOMER-EXIST TO TRUE.
  269     003170     READ CUSTOMER-MASTER-FILE
  270     003180         INVALID KEY SET CUSTOMER-NOT-FOUND TO TRUE
          003190     END-READ.
  271     003200     IF  CUSTOMER-EXIST THEN
  272     003210         MOVE CUST OF CUSMST TO CUST OF ORDHDR
  273     003220         MOVE ZEROES TO INVNUM
  274     003230         SET CUSTOMER-INVOICE-EXIST TO TRUE
  275     003240         PERFORM START-ON-CUSTOMER-INVOICE-FILE
  276     003250         IF  CUSTOMER-INVOICE-EXIST THEN
  277     003260             PERFORM READ-CUSTOMER-INVOICE-RECORD
  278     003270             IF CUSTOMER-INVOICE-EXIST THEN
```

Figura 122 (Parte 9 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página    11

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


  279     003280                 PERFORM CUSTOMER-MASTER-FILE-UPDATE
  280     003290                 MOVE AMPAID OF SUBFILE1-I TO AMOUNT-PAID
  281     003300                 SET PAYMENT-EXIST TO TRUE
  282     003310                 PERFORM PAYMENT-UPDATE
          003320                     UNTIL NO-MORE-INVOICES-EXIST
          003330                     OR NO-MORE-PAYMENT-EXIST
  283     003340                 IF  ARBAL OF CUSTOMER-MASTER-RECORD IS NEGATIVE
  284     003350                     SET MAKE-FIELD-BLINK
          003360                         DISPLAY-FIELD
          003370                         DO-NOT-REVERSE-FIELD-IMAGE
          003380                         OVER-PAYMENT-NOT-DISPLAYED
          003390                         DISPLAY-OVER-PAYMENT
          003400                         DISPLAY-EXCEPTION
          003410                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
          003420                         PROTECT-INPUT-FIELD TO TRUE
  285     003430                     MOVE ARBAL TO OVRPMT OF SUBFILE1-O
  286     003440                     MOVE MESSAGE-THREE TO ECPMSG OF SUBFILE1-O
  287     003450                     MOVE '0' TO STSCDE OF SUBFILE1-O
  288     003460                     PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          003470                 ELSE
  289     003480                     SET DO-NOT-DISPLAY-FIELD
          003490                         DO-NOT-DISPLAY-OVER-PAYMENT
          003500                         DO-NOT-REVERSE-FIELD-IMAGE
          003510                         DO-NOT-MAKE-FIELD-BLINK
          003520                         DO-NOT-DISPLAY-ACCEPT-PAYMENT
          003530                         PROTECT-INPUT-FIELD TO TRUE
  290     003540                     MOVE SPACES TO ECPMSG OF SUBFILE1-O
  291     003550                     MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  292     003560                     MOVE '0' TO STSCDE OF SUBFILE1-O
  293     003570                     PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          003580                 END-IF
          003590             ELSE
  294     003600                 PERFORM NO-CUSTOMER-INVOICE-ROUTINE
          003610             END-IF
          003620         ELSE
  295     003630             PERFORM NO-CUSTOMER-INVOICE-ROUTINE
          003640         END-IF
          003650     ELSE
  296     003660         SET REVERSE-FIELD-IMAGE
          003670             DO-NOT-PROTECT-INPUT-FIELD
          003680             DISPLAY-FIELD
          003690             DO-NOT-DISPLAY-OVER-PAYMENT
          003700             DO-NOT-MAKE-FIELD-BLINK
          003710             DISPLAY-EXCEPTION
          003720             DO-NOT-DISPLAY-ACCEPT-PAYMENT
          003730             DO-NOT-PROTECT-INPUT-FIELD TO TRUE
  297     003740         MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  298     003750         MOVE MESSAGE-ONE TO ECPMSG OF SUBFILE1-O
  299     003760         MOVE '1' TO STSCDE OF SUBFILE1-O
  300     003770         PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
          003780     END-IF.
  301     003790     PERFORM READ-MODIFIED-SUBFILE-RECORD.
          003800
          003810 START-ON-CUSTOMER-INVOICE-FILE.
  302     003820     START CUSTOMER-INVOICE-FILE
          003830         KEY IS GREATER THAN COMP-KEY
  303     003840         INVALID KEY SET NO-MORE-INVOICES-EXIST TO TRUE
```

Figura 122 (Parte 10 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página    12

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         003850     END-START.
         003860
         003870 READ-CUSTOMER-INVOICE-RECORD.
  304    003880     READ CUSTOMER-INVOICE-FILE NEXT RECORD
  305    003890         AT END SET NO-MORE-INVOICES-EXIST TO TRUE
         003900     END-READ.
  306    003910     IF  CUST OF CUSTOMER-MASTER-RECORD
         003920         IS NOT EQUAL TO CUST OF CUSTOMER-INVOICE-RECORD THEN
  307    003930         SET NO-MORE-INVOICES-EXIST TO TRUE
         003940     END-IF.
         003950
         003960 CUSTOMER-MASTER-FILE-UPDATE.
  308    003970     MOVE FILE-DATE TO LSTDAT OF CUSTOMER-MASTER-RECORD.
  309    003980     MOVE AMPAID OF SUBFILE1-I
         003990         TO LSTAMT OF CUSTOMER-MASTER-RECORD.
  310    004000     SUBTRACT AMPAID OF SUBFILE1-I
         004010         FROM ARBAL OF CUSTOMER-MASTER-RECORD.
  311    004020     REWRITE CUSTOMER-MASTER-RECORD
         004030         INVALID KEY
  312    004040             DISPLAY 'ERROR IN REWRITE OF CUSTOMER MASTER'
         004050     END-REWRITE.
         004060
         004070  REWRITE-DISPLAY-SUBFILE-RECORD.
  313    004080     MOVE AMPAID OF SUBFILE1-I TO AMPAID OF SUBFILE1-O.
  314    004090     MOVE CUST OF SUBFILE1-I TO CUST OF SUBFILE1-O.
  315    004100     SET WRITE-DISPLAY TO TRUE.
  316    004110     SET SUBFILE1-FORMAT TO TRUE.
  317    004120     MOVE CORR INDICATOR-AREA TO SUBFILE1-O-INDIC.
               *** Elementos CORRESPONDING para la instrucción 004120:
               ***       IN51
               ***       IN52
               ***       IN53
               ***       IN54
               ***       IN55
               ***       IN56
               *** Fin de elementos CORRESPONDING para la instrucción 004120
  318    004130     REWRITE SUBFILE PAYMENT-UPDATE-DISPLAY-RECORD
         004140         FORMAT IS 'SUBFILE1'
         004150     END-REWRITE.
         004160
         004170 NO-CUSTOMER-INVOICE-ROUTINE.
  319    004180     IF  STSCDE OF SUBFILE1-I IS EQUAL TO '1' THEN
  320    004190        IF  ACPPMT OF SUBFILE1-I IS EQUAL TO '*NO' THEN
  321    004200           SET DO-NOT-DISPLAY-FIELD
         004210               DO-NOT-DISPLAY-OVER-PAYMENT
         004220               DO-NOT-REVERSE-FIELD-IMAGE
         004230               DO-NOT-MAKE-FIELD-BLINK
         004240               DO-NOT-DISPLAY-ACCEPT-PAYMENT
         004250               PROTECT-INPUT-FIELD
         004260                 TO TRUE
  322    004270            MOVE SPACES TO ECPMSG OF SUBFILE1-O
  323    004280            MOVE ZEROES TO OVRPMT OF SUBFILE1-O
  324    004290            MOVE '0' TO STSCDE OF SUBFILE1-O
  325    004300            PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
         004310        ELSE
  326    004320            PERFORM CUSTOMER-MASTER-FILE-UPDATE
  327    004330            SET MAKE-FIELD-BLINK
```

Figura 122 (Parte 11 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página    13

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


         004340                DISPLAY-FIELD
         004350                DO-NOT-REVERSE-FIELD-IMAGE
         004360                OVER-PAYMENT-NOT-DISPLAYED
         004370                DISPLAY-OVER-PAYMENT
         004380                DISPLAY-EXCEPTION
         004390                DO-NOT-DISPLAY-ACCEPT-PAYMENT
         004400                PROTECT-INPUT-FIELD
         004410                  TO TRUE
  328    004420             MOVE ARBAL TO OVRPMT OF SUBFILE1-O
  329    004430             MOVE MESSAGE-THREE TO ECPMSG OF SUBFILE1-O
  330    004440             MOVE '0' TO STSCDE OF SUBFILE1-O
  331    004450             PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
         004460         END-IF
         004470     ELSE
  332    004480         SET REVERSE-FIELD-IMAGE
         004490             DISPLAY-FIELD
         004500             DO-NOT-PROTECT-INPUT-FIELD
         004510             DO-NOT-DISPLAY-OVER-PAYMENT
         004520             DISPLAY-EXCEPTION
         004530             DISPLAY-ACCEPT-PAYMENT
         004540             DISPLAY-ACCEPT-HEADING
         004550             DISPLAY-ACCEPT-MESSAGE
         004560             DO-NOT-MAKE-FIELD-BLINK
         004570               TO TRUE
  333    004580         MOVE ZEROS TO OVRPMT OF SUBFILE1-O
  334    004590         MOVE MESSAGE-TWO TO ECPMSG OF SUBFILE1-O
  335    004600         MOVE '1' TO STSCDE OF SUBFILE1-O
  336    004610         PERFORM REWRITE-DISPLAY-SUBFILE-RECORD
         004620     END-IF.
         004630
         004640 PAYMENT-UPDATE.
  337    004650     SUBTRACT AMPAID OF CUSTOMER-INVOICE-RECORD
         004660         FROM ORDAMT OF CUSTOMER-INVOICE-RECORD
         004670         GIVING AMOUNT-OWED.
  338    004680     SUBTRACT AMOUNT-PAID
         004690         FROM AMOUNT-OWED
         004700         GIVING INVOICE-BALANCE.
  339    004710     IF  INVOICE-BALANCE IS LESS THAN 0.01 THEN
  340    004720         MOVE 2 TO OPNSTS OF CUSTOMER-INVOICE-RECORD
  341    004730         MOVE ORDAMT OF CUSTOMER-INVOICE-RECORD
         004740           TO AMPAID OF CUSTOMER-INVOICE-RECORD
  342    004750         SUBTRACT AMOUNT-OWED
         004760             FROM AMOUNT-PAID
         004770     ELSE
  343    004780         ADD AMOUNT-PAID TO AMPAID OF CUSTOMER-INVOICE-RECORD
  344    004790         SET NO-MORE-PAYMENT-EXIST TO TRUE
         004800     END-IF.
  345    004810     REWRITE CUSTOMER-INVOICE-RECORD
         004820         INVALID KEY
  346    004830             DISPLAY 'ERROR IN REWRITE OF CUSTOMER INVOICE'
         004840     END-REWRITE.
  347    004850     IF  PAYMENT-EXIST THEN
  348    004860         PERFORM READ-CUSTOMER-INVOICE-RECORD
         004870     END-IF.
         004880
         004890 INITIALIZE-SUBFILE-RECORD.
  349    004900     MOVE SPACES TO CUST OF SUBFILE1-O.
```

Figura 122 (Parte 12 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/PAYUPDT          AS400SYS   96/07/04 11:36:37         Página    14

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


  350     004910     MOVE SPACES TO ECPMSG OF SUBFILE1-O.
  351     004920     MOVE '0' TO STSCDE OF SUBFILE1-O.
  352     004930     MOVE ZEROS TO AMPAID OF SUBFILE1-O.
  353     004940     MOVE ZEROS TO OVRPMT OF SUBFILE1-O.
  354     004950     ADD 1 TO REL-NUMBER.
  355     004960     MOVE CORR INDICATOR-AREA TO SUBFILE1-O-INDIC.
                *** Elementos CORRESPONDING para la instrucción 004960:
                ***       IN51
                ***       IN52
                ***       IN53
                ***       IN54
                ***       IN55
                ***       IN56
                *** Fin de elementos CORRESPONDING para la instrucción 004960
  356     004970     WRITE SUBFILE PAYMENT-UPDATE-DISPLAY-RECORD
          004980         FORMAT IS 'SUBFILE1'
          004990     END-WRITE.

                         * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 122 (Parte 13 de 13). Listado fuente de un ejemplo de programa de actualización de pagos

La pantalla inicial que se escribe en la estación de trabajo para pedir el número de cliente y el pago es la siguiente:

```
Solicitud de actualización de pagos de clientes                Fecha 08/11/96

            Cliente     Pagos


            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________

            ______   __________
            ______   __________
            ______   __________
            ______   __________
            ______   __________
```

Entre los números de cliente y los pagos:

```
Solicitud de actualización de pagos de clientes                Fecha 11/08/96

            Cliente     Pago


             34500      2000
             40500     30000
             36000      2500
             12500       200
             22799      4500
             41900      7500
             10001      5000
             49500      2500
             13300      3500
             56900      4000
```

Los pagos que darían como resultado un dinero pagado de más o que tienen un número de cliente incorrecto se dejan en la pantalla y se añaden los mensajes pertinentes:

```
Solicitud de actualización de pagos de clientes                Fecha 11/08/96

Aceptar     Cliente     Pago       Mensaje de excepción
pago

_____        40500       30000     NO EXISTEN FACTURAS PARA EL CLIENTE

_____        12500         200     NO EXISTEN FACTURAS PARA EL CLIENTE

_____        41900        7500     NO EXISTEN FACTURAS PARA EL CLIENTE
             10001        5000     EL CLIENTE NO EXISTE

_____        13300        3500     NO EXISTEN FACTURAS PARA EL CLIENTE









Valores de aceptar pago: (*NO *YES)
```

Indique qué pagos se han de aceptar:

```
Solicitud de actualización de pagos de clientes                Fecha 11/08/96

Aceptar     Cliente     Pago       Mensaje de excepción
pago

  *NO        40500       30000     NO EXISTEN FACTURAS PARA EL CLIENTE

  *YES       12500         200     NO EXISTEN FACTURAS PARA EL CLIENTE

  *NO        41900        7500     NO EXISTEN FACTURAS PARA EL CLIENTE
             10001        5000     EL CLIENTE NO EXISTE

  *NO        13300        3500     NO EXISTEN FACTURAS PARA EL CLIENTE









Valores de aceptar pago: (*NO *YES)
```

Los pagos aceptados se procesan y se visualiza información sobre el dinero pagado de más:

```
Solicitud de actualización de pagos de clientes                Fecha 11/08/96

Aceptar     Cliente     Pago       Mensaje de excepción
pago

             12500         200     EL CLIENTE TIENE UN SUELDO A SU FAVOR DE 58.50

             10001        5000     EL CLIENTE NO EXISTE
```

Apéndice A. Nivel de soporte de lenguaje

Norma COBOL

La norma COBOL (tal como se define en “Acerca de la publicación ILE COBOL/400 Guía del programador, SC10-9658 (SC09-2072)” en la página xxi) consta de once módulos de proceso funcionales, siete de los cuales son obligatorios y cinco opcionales.

Los siete módulos obligatorios son: núcleo, E-S secuencial, E-S relativa, E-S indexada, comunicación entre programas, clasificación-fusión y manipulación de texto fuente. Los cinco módulos opcionales son los siguientes: función intrínseca, transcriptor de informes, comunicación, depuración y segmentación.

Los elementos de lenguaje de los módulos pueden clasificarse como elementos de nivel 1 y de nivel 2. Los elementos de nueve de los módulos se dividen en elementos de nivel 1 y elementos de nivel 2. Tres de los módulos (SORT-MERGE, REPORT WRITER e INTRINSIC FUNCTION) contienen sólo elementos de nivel 1. Por ejemplo, los elementos de nivel 1 del núcleo realizan operaciones internas básicas. Los elementos de nivel 2 del núcleo proporcionan un proceso interno más amplio y sofisticado.

Los tres subconjuntos de la norma COBOL son el subconjunto superior, el intermedio y el mínimo. Cada subconjunto está compuesto por un nivel de los siete módulos obligatorios: núcleo, E-S secuencial, E-S relativa, E-S indexada, comunicación entre programas, clasificación-fusión y manipulación de texto fuente. En los tres subconjuntos de la norma COBOL no se requieren los cinco módulos opcionales (función intrínseca, transcriptor de informes, comunicación, depuración y segmentación).

El subconjunto superior está compuesto por todos los elementos de lenguaje del nivel más alto de todos los módulos obligatorios. Es decir:

- Los elementos de nivel 2 de núcleo, E-S secuencial, E-S relativa, E-S indexada, comunicación entre programas y manipulación de texto fuente.
- Los elementos de nivel 1 de clasificación-fusión.

El subconjunto intermedio está compuesto por todos los elementos de nivel 1 de todos los módulos obligatorios. Es decir:

- Los elementos de nivel 1 de núcleo, E-S secuencial, E-S relativa, E-S indexada, comunicación entre programas, clasificación-fusión y manipulación de texto fuente.

El subconjunto mínimo está compuesto por todos los elementos de lenguaje de nivel 1 de los módulos núcleo, E-S secuencial y comunicación entre programas.

Los cinco módulos opcionales no forman parte integral de ninguno de los subconjuntos.Sin embargo, con cualquiera de los subconjuntos puede haber asociada cualquier combinación de parte de ellos, de todos o de ninguno.

Nivel de soporte de lenguaje ILE COBOL/400

El compilador ILE COBOL/400 da soporte:

- Al nivel 2 de los módulos E-S secuencial y de manipulación de texto fuente.
- Al nivel 1 de los módulos núcleo, E-S relativa, E-S indexada, comunicación entre programas y clasificación-fusión.

El compilador ILE COBOL/400 no soporta los módulos transcriptor de informes, comunicación depuración y segmentación de la norma COBOL.

El compilador ILE COBOL/400 soporta parcialmente el módulo de función intrínseca ANSI X3.23a-1989.

El nivel de soporte dado por el compilador ILE COBOL/400 está representado en la Tabla 22 en la página 479. La tabla:

- muestra el nivel de soporte del compilador ILE COBOL/400 para cada módulo de proceso funcional de la norma COBOL
- describe cada uno de los módulos.

A continuación, se da una explicación de la notación utilizada en la tabla:

```
1     REL     0,2
└┘    └──┘    └──┘
 ↑     ↑       ↑
 1     2       3
```

1 Nivel de este módulo soportado por el compilador ILE COBOL/400. En este ejemplo, se da soporte al nivel 1 del módulo E-S relativa.

2 Código de 3 caracteres que identifica el módulo.En este ejemplo, se hace referencia al módulo E-S relativa.

3 **Rango** de niveles de soporte **definidos** por la norma COBOL. El nivel 0 significa que una norma **mínima** de COBOL no necesita dar soporte a este módulo para acatar la norma.

Tabla 22. Nivel de soporte del compilador ILE COBOL/400

Nivel de lenguaje ILE COBOL/400 soportado	**Descripción del módulo**
Núcleo 1 NUC 1,2	Contiene los elementos de lenguaje necesarios para el proceso interno de los datos dentro de las cuatro divisiones básicas de un programa y la capacidad de definir y acceder a tablas.
E-S secuencial 2 SEQ 1,2	Proporciona acceso a los registros de archivo siguiendo la secuencia establecida en la que se han escrito en el archivo.
E-S relativa 1 REL 0,2	Proporciona acceso a los registros de una forma aleatoria o secuencial. Cada registro está identificado de forma exclusiva por un entero que representa la posición lógica del registro en el archivo.
E-S indexada 1 INX 0,2	Proporciona acceso a los registros de una forma aleatoria o secuencial. Cada registro de un archivo indexado está identificado de forma exclusiva por una clave de registro.
Comunicación entre programas 1 IPC 1,2	Le permite a un programa COBOL comunicarse con otros programas mediante transferencias de control y acceso a datos comunes.
Clasificación-fusión 1 SRT 0,1	Ordena uno o más archivos de registros o bien combina dos o más archivos ordenados de forma idéntica según las claves especificadas por el usuario.
Manipulación de texto fuente 2 STM 0,2	Permite la inserción de texto COBOL predefinido en un programa durante la compilación.
Transcriptor de informes 0 RPW 0,1	Proporciona la generación semiautomática de informes impresos.
Comunicaciones 0 COM 0,2	Proporciona la posibilidad de acceder, procesar y crear mensajes o partes de mensajes; también permite la comunicación por medio de un sistema de control de mensajes con dispositivos de comunicación locales y remotos.
Depuración 0 DEB 0,2	Permite especificar instrucciones y procedimientos para la depuración.
Función intrínseca 0 ITR 0,1	Proporciona la posibilidad de hacer referencia a un elemento de datos cuyo valor se derive de forma automática en el momento de referencia durante la ejecución del programa objeto.
Segmentación 0 SEG 0,2	Proporciona el solapamiento de las secciones de la Procedure Division durante la creación de objetos.

Soporte de interfaz común de programación (CPI) de Arquitectura para Aplicaciones de Sistemas* (SAA*)

El archivo fuente QCBLLESRC de la biblioteca de producto QSYSINC contiene miembros que tienen especificaciones para varias interfaces comunes de programación SAA. En dichas especificaciones se describen las interfaces de parámetros. Este archivo es propiedad de IBM y no debe modificarse.

Si desea personalizar cualquiera de las especificaciones, debe copiar los miembros que desee modificar en un archivo fuente de una de sus bibliotecas. Para hacerlo,

puede utilizar el mandato Copiar Archivo (CPYF). Para obtener más información sobre el mandato CPYF, consulte la publicación *CL Reference*.

Si copia las especificaciones en su biblioteca, deberá renovar las copias al instalar un nuevo release del producto o cuando se realicen cambios con un arreglo temporal del programa (PTF). IBM da mantenimiento a estas especificaciones sólo en las bibliotecas en las que se distribuyen.

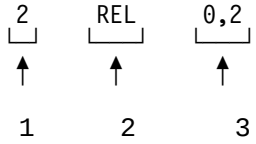
Apéndice B. El distintivo de norma de proceso de información federal (FIPS)

Los distintivos FIPS pueden especificarse para supervisar un subconjunto COBOL FIPS, cualquiera de los módulos opcionales, todos los elementos de lenguaje obsoletos o una combinación de subconjunto COBOL FIPS, módulos opcionales y todos los elementos de lenguaje obsoletos.

La supervisión es un análisis comparativo de la sintaxis utilizada en el programa fuente y de la sintaxis incluida en los módulos opcionales y el subconjunto FIPS seleccionado por el usuario. Se identifica la sintaxis utilizada en el programa fuente que no se ajusta al subconjunto COBOL FIPS seleccionado ni a los módulos opcionales. También se identificará la sintaxis de un elemento de lenguaje obsoleto utilizado en el programa fuente (dependiendo de la opción de compilador elegida). Consulte el página 40 para obtener más información sobre los parámetros para colocar distintivos FIPS.

Las especificaciones COBOL FIPS 21-4 son las especificaciones de lenguaje contenidas en la norma COBOL (tal como se describen en el manual “Acerca de la publicación ILE COBOL/400 Guía del programador, SC10-9658 (SC09-2072)” en la página xxi). COBOL FIPS está dividido en tres subconjuntos y cuatro módulos opcionales. Los tres subconjuntos son mínimo, intermedio y superior. Los cuatro módulos opcionales son: transcriptor de informes, comunicación, depuración y segmentación. Estos cuatro módulos opcionales no forman parte integral de ninguno de los subconjuntos; sin embargo, con cualquiera de los subconjuntos puede haber asociada una combinación de parte de ellos, de todos o de ninguno. Los programas escritos para cumplir la norma COBOL FIPS 21-4 deben acatar uno de los subconjuntos de COBOL FIPS 21-4. La Tabla 23 en la página 482 muestra los módulos de proceso COBOL de la norma ANSI incluidos en cada uno de los subconjuntos de COBOL FIPS 21-4.

A continuación, se da una explicación de la notación utilizada en la tabla:



1 El nivel de este módulo soportado por la norma COBOL FIPS 21-4. En este ejemplo, se da soporte al nivel 2 del módulo de E-S relativa.

2 Código de 3 caracteres que identifica el módulo.En este ejemplo, se hace referencia al módulo E-S relativa.

3 **Rango** de niveles de soporte **definidos** por la norma COBOL. El nivel 0 significa que una norma **mínima** de COBOL no necesita dar soporte a este módulo para acatar la norma.

Tabla 23. Norma COBOL y COBOL FIPS 21-4

Nombre de módulo ANSI	FIPS alto	FIPS intermedio	FIPS mínimo
Núcleo	2 NUC 1,2	1 NUC 1,2	1 NUC 1,2
E-S secuencial	2 SEQ 1,2	1 SEQ 1,2	1 SEQ 1,2
E-S relativa	2 REL 0,2	1 REL 0,2	0 REL 0,2
E-S indexada	2 INX 0,2	1 INX 0,2	0 INX 0,2
Manipulación de texto fuente	2 STM 0,2	1 STM 0,2	0 STM 0,2
Clasificación-fusión	1 SRT 0,1	1 SRT 0,1	0 SRT 0,1
Función intrínseca	1 ITR 0,1	0 ITR 0,1	0 ITR 0,1
Comunicación entre programas	2 IPC 1,2	1 IPC 1,2	1 IPC 1,2
Transcriptor de informes	0 ó 1 RPW 0,1	0 ó 1 RPW 0,1	0 ó 1 RPW 0,1
Segmentación	0,1 ó 2 SEG 0,2	0,1 ó 2 SEG 0,2	0,1 ó 2 SEG 0,2
Depuración	0,1 ó 2 DEB 0,2	0,1 ó 2 DEB 0,2	0,1 ó 2 DEB 0,2
Comunicaciones	0,1 ó 2 COM 0,2	0,1 ó 2 COM 0,2	0,1 ó 2 COM 0,2

Los elementos que están especificados en el programa fuente ILE COBOL/400 y que no están incluidos en COBOL FIPS 21-4 se describen en el Apéndice A, “Nivel de soporte de lenguaje” en la página 477.

Apéndice C. Mensajes ILE COBOL/400

En este apéndice se facilita una descripción general de los mensajes que IBM suministra con el programa bajo licencia ILE COBOL/400.

Descripción de mensajes COBOL

Los mensajes del programa bajo licencia ILE COBOL/400 empiezan por el prefijo LNC, LNM, LNP, LNR o LNT.

Los mensajes LNC los emite el corrector de sintaxis COBOL cuando se utiliza SEU para entrar el código fuente ILE COBOL/400. También son mensajes generados por el compilador.

Los mensajes LNM se utilizan como cabeceras en un vuelco con formato ILE COBOL/400 durante la ejecución.

Los mensajes LNP se utilizan en los mandatos CL y menús de ILE COBOL/400.

Los mensajes LNR facilitan información adicional sobre el funcionamiento del sistema durante la ejecución.

Los mensajes LNT se utilizan como cabeceras para las diversas partes del listado del compilador ILE COBOL/400.

Los números de mensajes se asignan de la forma siguiente:

Mensaje de error	Descripción
Del LNR7000 al LNR7199	Mensajes de escape
Del LNR7200 al LNR7999	Mensajes de ejecución
Del LNR8000 al LNR8200	Mensajes de escape
Del LNC0000 al LNC0999	Mensajes de gravedad inferior a 30
Del LNC1000 al LNC2999	Mensajes de gravedad superior o igual a 30
Del LNC8000 al LNC8799	Mensajes sobre distintivos FIPS
Del LNC9001 al LNC9099	Mensajes del compilador

Niveles de gravedad

El programa bajo licencia ILE COBOL/400 proporciona los niveles de gravedad siguientes:

Gravedad Significado

00 Informativo: Este nivel se utiliza para dar información al usuario. No se ha producido ningún error. Los mensajes informativos se listan sólo cuando se especifica la opción FLAG (00).

10 Aviso: Este nivel indica que se ha detectado un error, pero no es lo suficientemente grave para interferir con la ejecución del programa.

20 Error: Este nivel indica que se ha cometido un error, pero el compilador está realizando una recuperación que podría dar el código deseado.

30 Error grave: Este nivel indica que se ha detectado un error grave. La compilación finaliza, pero no se crea el objeto de módulo y no se puede intentar la ejecución del programa.

40 Irrecuperable: Este nivel indica normalmente un error del usuario que provoca la terminación del proceso.

50 Irrecuperable: Este nivel indica normalmente un error del compilador que provoca la terminación del proceso.

99 Acción: Se requiere una acción manual, como por ejemplo entrar una respuesta, cambiar el papel de la impresora o sustituir los disquetes.

Nota: Los mensajes 00, 10 y 20 se suprimen cuando se utiliza la opción FLAG(30) de la instrucción PROCESS o cuando el mandato CRTCBLMOD/CRTBNDCBL especifica FLAG(30) y no lo altera temporalmente la instrucción PROCESS. Consulte el apartado “Utilización de la instrucción PROCESS para especificar opciones de compilador” en la página 47 para obtener más información.

El compilador siempre intenta dar el diagnóstico completo de todo el texto fuente del programa aunque se hayan detectado errores. Si el compilador no puede continuar en una instrucción dada, el mensaje notifica que el compilador no puede continuar y que hará caso omiso del resto de la instrucción. Cuando se produce este error, el programador debe examinar la instrucción entera.

Para generar todos los mensajes se utiliza el recurso de mensajes de OS/400. Los mensajes del compilador ILE COBOL/400 residen en el archivo de mensajes QLNCMSG y los mensajes de ejecución en el archivo de mensajes QLNRMSG.

Las variables de sustitución y los valores de respuesta válidos los determina el programa que envía el mensaje, *no* la descripción del mensaje almacenada en el archivo de mensajes. Sin embargo, ciertos elementos de la descripción de un mensaje pueden cambiarse: por ejemplo, el texto, el nivel de gravedad, la respuesta por omisión o la lista de vuelco. Para efectuar dichos cambios, es necesario definir otra descripción de mensaje con el mandato Añadir descripción de mensaje (ADDMSGD), colocar la descripción modificada en un archivo de mensajes creado por el usuario[1] y especificar dicho archivo con el mandato Alterar temporalmente archivo de mensajes (OVRMSGF). La utilización del mandato OVRMSGF permite al compilador recuperar mensajes del archivo especificado. Vea los mandatos ADDMSGD y OVRMSGF en la publicación *CL Reference* para obtener información adicional.

PRECAUCIÓN: Alterar temporalmente un mensaje suministrado por IBM con un mensaje creado por el usuario puede dar resultados no previstos. Si no se conservan los valores de respuesta, es posible que el programa no responda a las respuestas. Cambiar las respuestas por omisión en los mensajes de tipo *NOTIFY podría afectar a la capacidad del programa de ejecutarse en modalidad desatendida. Cambiar la gravedad podría cancelar un trabajo no cancelado anteriormente. Tenga cuidado al alterar temporalmente los mensajes suministrados por IBM con mensajes creados por el usuario.

[1] Si debe cambiarse un mensaje suministrado por IBM y sustituirse en *su* archivo de mensajes, llame al representante del servicio técnico.

Mensajes de compilación

Los mensajes LNC se imprimen en el listado del programa cuando se encuentran errores durante la compilación del programa. Los mensajes LNC incluyen el mensaje emitido cuando se solicitan los distintivos FIPS (norma de proceso de información federal); para obtener más información sobre los mensajes FIPS, consulte la página 481 de este apéndice.

Listados de programas

En la salida del compilador, al listado fuente le sigue el listado de mensajes ILE COBOL/400.El listado de mensajes ILE COBOL/400 proporciona el identificador de mensajes, la gravedad, el texto, la ubicación del error (normalmente) y el resumen de los mensajes.

Cuando se especifica el valor *IMBEDERR con el parámetro OPTION del mandato CRTCBLMOD o CRTBNDCBL, en el listado fuente también se facilita el texto del mensaje de primer nivel inmediatamente a continuación de la línea en la que se ha detectado el error.

Para obtener más información sobre los listados de programa, consulte el apartado “Listado fuente” en la página 59.

Mensajes interactivos

En un entorno interactivo, los mensajes se visualizan en la pantalla de la estación de trabajo. Pueden aparecer en la pantalla actual como resultado de la ejecución del programa o en respuesta a la entrada de las pantallas de solicitud, de menú o de entrada de mandatos o en las herramientas de Juego de Herramientas para el Desarrollo de Aplicaciones/400. Los mensajes pueden aparecer también como resultado de una petición, bien sea de un mandato de visualización o de la opción de un menú.

Los mensajes del programa bajo licencia ILE COBOL/400 empiezan por el prefijo LNC o LNR.

Los mensajes LNC los emite el corrector de sintaxis ILE COBOL/400 cuando se utiliza el Programa de Utilidad para Entrada del Fuente (SEU) para entrar el código fuente ILE COBOL/400. Por ejemplo, la pantalla siguiente aparece después de entrar de forma incorrecta el nombre de programa en el párrafo PROGRAM-ID.

```
 Columnas  . . :    1  71           Edición                    XMPLIB/QCBLLESRC
 SEU==> _____________________________________________________________      TESTPR
 FMT CB .....-A+++B++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ***************  Inicio de datos  ***********************************
0000.10         IDENTIFICATION DIVISION.
0000.20         PROGRAM-ID. #TESTPR.
0000.70         ENVIRONMENT DIVISION.
0000.90         SOURCE-COMPUTER. IBM-AS400.
        ****************** Fin de datos ***************************************









 F3=Salir   F4=Solicitud   F5=Renovar   F9=Recuperar  F10=Cursor
 F16=Repetir búsqueda      F17=Repetir cambio         F24=Más teclas
# no está en el juego de caracteres COBOL. Línea rechazada.
```

Figura 123. Ejemplo de mensaje del corrector sintáctico ILE COBOL/400

Los mensajes LNC también se emiten durante la compilación del programa. Consulte el apartado “Mensajes de compilación” en la página 485 para obtener una descripción.

Los mensajes LNR facilitan información adicional sobre el funcionamiento del sistema durante la ejecución. Por ejemplo, podría ver la pantalla siguiente si hay un error durante la ejecución:

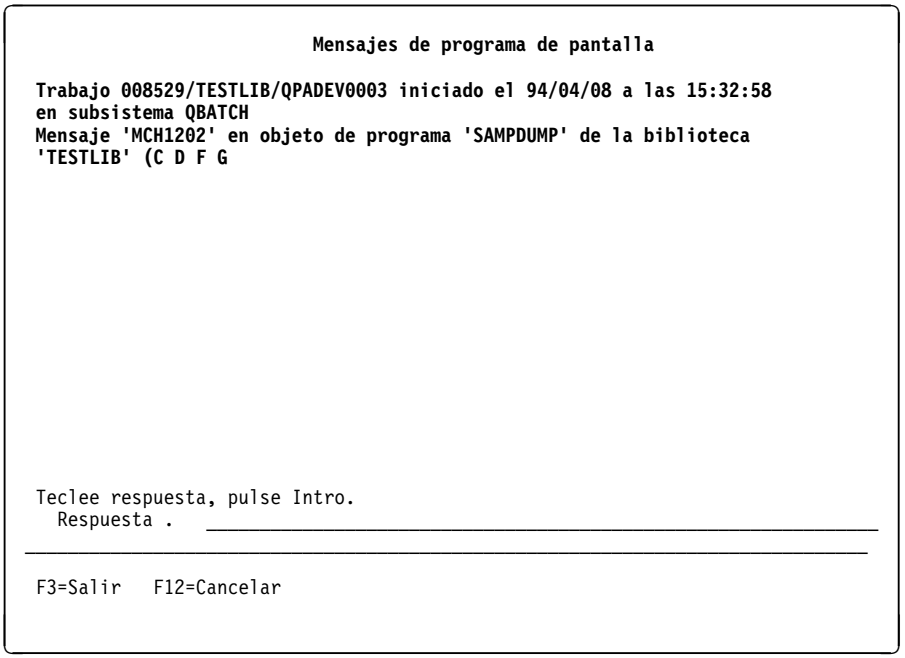


Figura 124. Mensaje de error de ejecución

Si coloca el cursor en la línea en la que se indica el número de mensaje MCH1202 y pulsa la tecla Ayuda o la tecla F1, se visualizará la información del mensaje tal y como se muestra a continuación:

```
Información de Mensaje Adicional

 ID de mensaje  . . . . :   LNR7200       Gravedad . . . . . . . :   50
 Tipo de mensaje  . . . :   Consulta
 Fecha de envío . . . . :   96/11/08      Hora de envío. . . . . :   15:33:31

 Mensaje . . . . :   Mensaje 'MCH1202' en objeto de programa 'SAMPDUMP' de la
                     biblioteca 'TESTLIB' (C D F G).
 Causa . . . . . :   Se ha detectado el mensaje 'MCH1202' en la instrucción
   de COBOL 42 del programa COBOL 'SAMPDUMP' en el objeto de programa 'SAMPDUMP'
   de la biblioteca 'TESTLIB'.
 Recuperación  . :   Entre una G para continuar el programa en la siguiente
   instrucción MI una C si no desea que se efectúe un vuelco, una D si desea
   que se efectúe un vuelco de los identificadores de COBOL o una F para volcar
   tanto los identificadores de COBOL como la información de archivo. El texto
   del mensaje para 'MCH1202' es el siguiente: 'Error de datos decimales.'
 Elecciones posibles de respuesta al mensaje  . . . . . . . . . . . . . :
   C -- No se proporciona ningún vuelco formateado
   D -- Se proporciona un vuelco de los identificadores de COBOL
                                                                        Más...
 Pulse Intro para continuar

 F3=Salir   F6=Imprimir   F10=Visualizar mensajes de anotaciones de trabajo
 F11=Vis. detalles mensajes   F12=Cancelar   F21=Selecc. nivel asistencia
```

```
Información de Mensaje Adicional

 ID de mensaje  . . . . :   LNR7200       Gravedad . . . . . . . :   50
 Tipo de mensaje  . . . :   Consulta

   F -- Se proporciona un vuelco de los identificadores de COBOL y de la
        información de archivos
   G -- Continuar el programa en la siguiente instrucción MI.









                                                                        Final
 Pulse Intro para continuar

 F3=Salir   F6=Imprimir   F10=Visualizar mensajes de anotaciones de trabajo
 F11=Vis. detalles mensajes   F12=Cancelar   F21=Selecc. nivel asistencia
```

Figura 125. Mensaje de error de ejecución—texto de segundo nivel

En el apartado “Respuesta a mensajes” en la página 489 se explica cómo visualizar el texto de mensajes de segundo nivel y cómo responder a los mensajes.

Los mensajes LNM que van del 0001 al 0050 se utilizan como cabeceras de la información impresa durante un vuelco con formato ILE COBOL/400.

Respuesta a mensajes

En un entorno interactivo, se indica un mensaje por medio de una o de varias de las situaciones siguientes:

- Aparece un mensaje breve (denominado texto de primer nivel) en la línea de mensajes
- La imagen del campo de entrada que tiene en error aparece resaltada con contraste invertido
- El teclado se bloquea
- Se escucha el sonido de una alarma (si está instalada la opción de alarma).

En los siguientes párrafos se describen brevemente algunos métodos de respuesta a los mensajes de error; hay más información disponible en las publicaciones *System Operation for New Users* y Juego de herramientas para el desarrollo de aplicaciones/400.

Si la corrección necesaria resulta obvia desde la pantalla inicial, puede pulsar la tecla Restablecer Error (si el teclado está bloqueado), entrar la información correcta y continuar trabajando.

Si el mensaje hace necesario que elija una respuesta (como por ejemplo **C** para cancelar, **D** para volcar los identificadores COBOL, **F** para volcar los identificadores COBOL y la información de archivo o **G** para reanudar el proceso en la siguiente instrucción COBOL), las opciones de respuesta aparecen entre paréntesis en el texto de primer nivel. Si desea ver un ejemplo, vaya a la Figura 124 en la página 487.

Si la información de la pantalla de información inicial no facilita los datos suficientes para manejar el error, puede pulsar la tecla Ayuda (después de colocar el cursor en la línea de mensajes, si es necesario) para obtener una pantalla de segundo nivel con información adicional sobre la forma de corregir el error. Para volver a la pantalla inicial, pulse la tecla Intro; a continuación, pulse la tecla Restablecer Error (si el teclado está bloqueado) y realice la corrección o de una respuesta.

Si el error se produce al compilar o al ejecutar un programa, es posible que sea necesario modificar la instrucciones ILE COBOL/400 o los mandatos del lenguaje de control (CL). Consulte la publicación *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)* para obtener más información sobre la forma de modificar las instrucciones.

Apéndice D. Soporte a idiomas internacionales con juegos de caracteres de doble byte

En este apéndice se describen únicamente las mejoras realizadas en el lenguaje de programación COBOL para escribir programas que procesan caracteres de doble byte.

Más concretamente, en este apéndice se señalan los lugares en los que puede utilizar caracteres DBCS (juego de caracteres de doble byte) en cada parte de un programa COBOL así como diversas consideraciones para trabajar con datos DBCS en el lenguaje ILE COBOL/400.

Existen dos maneras de especificar caracteres DBCS:

- DBCS delimitado
- Datos gráficos DBCS

En general, COBOL maneja los caracteres DBCS delimitados de la misma forma que maneja los caracteres alfanuméricos. **DBCS delimitado** es una serie de caracteres en las que cada carácter está representado por dos bytes.El carácter empieza por un carácter de desplazamiento a teclado ideográfico (SO) y termina con un carácter de desplazamiento a teclado estándar (SI). A usted le corresponde el saber (o hacer que el programa COBOL compruebe) qué datos contienen caracteres DBCS y el asegurarse que el programa recibe y procesa esta información correctamente.

Ya puede utilizar descripciones DDS que definan campos de datos gráficos DBCS con los programas ILE COBOL/400. Un campo de datos **gráficos DBCS** es una serie de caracteres en la que cada carácter está representado por dos bytes.La serie de caracteres no contiene caracteres de desplazamiento a teclado ideográfico ni caracteres de desplazamiento a teclado estándar. Para obtener más información sobre la especificación de datos gráficos con los programas ILE COBOL/400, consulte el apartado “Campos gráficos DBCS” en la página 335.

Utilización de los caracteres DBCS en literales

Un literal mixto consta de caracteres DBCS (juego de caracteres de doble byte) y SBCS (juego de caracteres de un solo byte).

Para procesar caracteres DBCS en literales mixtos, está disponible la opción GRAPHIC de la instrucción PROCESS. Cuando se especifica la opción GRAPHIC, los literales mixtos se manejan partiendo del supuesto que los caracteres hexadecimales 0E y 0F son los caracteres de desplazamiento a teclado estándar e ideográfico, respectivamente y que delimitan los caracteres DBCS del literal mixto. Cuando se especifica NOGRAPHIC, o se establece de forma implícita, el compilador ILE COBOL/400 tratará a los literales no numéricos que contengan los caracteres hexadecimales 0E y 0F como si sólo contuviesen caracteres SBCS.Hex 0E y hex 0F no se tratan como caracteres de desplazamiento a teclado estándar y a teclado ideográfico, si no que se considerará que forman parte de una serie de caracteres SBCS.

Un literal DBCS consta únicamente de caracteres del juego de caracteres de doble byte y se trata siempre como serie de caracteres DBCS.

Nota: No ha de confundirse la opción GRAPHIC de la instrucción PROCESS con los valores *PICXGRAPHIC o *PICGGRAPHIC del parámetro CVTOPT del mandato CRTCBLMOD o CRTBNDCBL ni con las opciones CVTPICXGRAPHIC y CVTPICGGRAPHIC de la instrucción PROCESS, que se utilizan para especificar datos gráficos de doble byte desde una descripción DDS. Para obtener más información sobre la especificación de datos gráficos, consulte el apartado “Campos gráficos DBCS” en la página 335.

Cómo especificar literales que contengan caracteres DBCS

Al especificar literales que contengan caracteres DBCS, siga las mismas reglas que se aplican para especificar literales alfanuméricos, además de observar las reglas siguientes, que son específicas de los literales mixtos y de DBCS:

- Los literales pueden tener muchas formas distintas. Los siguientes son sólo dos ejemplos posibles:

  "SINGLE0_EK1K2K30$_F$BYTES"

  "0_EK1K20$_F$"

- Los literales DBCS empiezan por

  G"O_E o N"0_E

  seguido de uno o más caracteres de doble byte y terminan con

  0_F"

  Un ejemplo de ello es lo siguiente:

  G"0_EKIK20$_F$"
  N"0_E 0_F"

- Los literales mixtos utilizan implícitamente USAGE DISPLAY. Los literales DBCS utilizan implícitamente USAGE DISPLAY-1.
- En el literal mixto, antes o después de una cadena de caracteres DBCS pueden aparecer caracteres EBCDIC.
- Todas las series DBCS aparecen entre carácter de desplazamiento a teclado ideográfico y estándar. Un **carácter de desplazamiento a teclado ideográfico** es un carácter de control (0E hexadecimal) que indica el inicio de una serie de caracteres de doble byte. Ocupa 1 byte. Un **carácter de desplazamiento a teclado estándar** es un carácter de control (0F hexadecimal) que indica el final de una serie de caracteres de doble byte. Un carácter de desplazamiento a teclado estándar ocupa 1 byte.
- Todas las comillas SBCS que haya dentro de un literal mixto deben doblarse. No es necesario doblar las comillas DBCS en los literales G", pero sí las comillas DBCS de los literales N". Por ejemplo:

  "Mixto ""0_EK1K2K30$_F$"" literal"

  G"0_EK1K2K3"K4"K5K60$_F$"

  N"0_EK1K2K3""K4""K5K60$_F$"

- Las serie DBCS nulas (caracteres de desplazamiento a teclado ideográfico y estándar sin ningún carácter DBCS) pueden utilizarse en un literal mixto *sólo* cuando el literal contiene al menos un carácter SBCS.

Los caracteres de desplazamiento a teclado ideográfico y estándar no pueden anidarse.

Los caracteres de desplazamiento forman parte de un literal mixto (no de un literal DBCS puro) y participan en todas las operaciones.

Otras consideraciones

Comillas: Aunque en la explicación anterior se utiliza el término *comillas* para describir el carácter que identifica un literal, el carácter que realmente se utilice puede variar dependiendo de la opción especificada en los mandatos CRTCBLMOD o CRTBNDCBL o bien en la instrucción PROCESS. Si especifica la opción APOST, se utilizará un apóstrofo ('). De lo contrario, se utilizarán comillas ("). En este apéndice, *comillas* hace referencia tanto al apóstrofo como a las comillas dobles. El carácter que elija no afecta a las reglas de especificación de literales.

Caracteres de desplazamiento: Los caracteres de desplazamiento a teclado ideográfico y estándar separan a los caracteres EBCDIC de los DBCS. Forman parte del literal mixto. Por lo tanto, los caracteres de código de desplazamiento participan en todas las operaciones cuando aparecen en literales mixtos. No participan en ninguna operación cuando aparecen en literales DBCS.

Cómo comprueba el compilador COBOL los caracteres DBCS

Cuando el compilador COBOL encuentra una serie DBCS, la comprueba explorándola carácter DBCS a carácter.

Las condiciones siguientes hacen que el compilador COBOL diagnostique como *no válido* un literal que contiene caracteres DBCS:

- La sintaxis del literal es incorrecta.
- El literal mixto tiene una longitud superior a una línea y *no* sigue las reglas de continuación de los literales no numéricos (consulte el apartado “Cómo continuar los literales mixtos en una línea nueva” para obtener más información).
- La longitud del literal DBCS es superior a una línea.

Para cada literal DBCS, mixto o SBCS que no sea válido, el compilador genera un mensaje de error y acepta o hace caso omiso del literal.

Cómo continuar los literales mixtos en una línea nueva

Para continuar un literal mixto en otra línea del código fuente, realice *todas* las acciones siguientes:

- Coloque un carácter de desplazamiento a teclado estándar en la columna 71 o en la 72 de la línea a continuar (si lo pone en la columna 71, el blanco de la columna 72 no se toma en cuenta)
- Coloque un guión (-) en la columna 7 (el área de continuación) de la línea nueva
- Coloque una comilla, un carácter de desplazamiento a teclado ideográfico y el resto del literal en el área B de la línea nueva.

Por ejemplo:

```
-A 1 B
  ⋮
 01  DBCS1             PIC X(12)                VALUE "0EK1K2K30F
-    "0EK4K50F".
  ⋮
```

El valor de DBCS1 es "0_EK1K2K3K4K50_F".

En la longitud del literal mixto no se cuentan el carácter de desplazamiento a teclado estándar, la comilla ni el carácter de desplazamiento a teclado ideográfico utilizados para continuar una línea. Sí se cuentan el primer carácter de desplazamiento a teclado ideográfico y el último carácter de desplazamiento a teclado estándar.

Dónde puede utilizar caracteres DBCS en un programa COBOL

En general, puede utilizar literales mixtos siempre que estén permitidos los literales no numéricos. Los literales de los elementos siguientes, sin embargo, no pueden incluir caracteres de doble byte:

- Cláusula ALPHABET-nombre
- Cláusula CURRENCY SIGN
- Cláusula ASSIGN
- Cláusula CLASS-nombre
- Instrucción CALL
- Instrucción CANCEL.

Puede utilizar literales DBCS siempre que estén permitidos los literales no numéricos, excepto como literales, en los elementos siguientes:

- Cláusula ALPHABET
- Cláusula ASSIGN
- Cláusula CLASS
- Cláusula CURRENCY SIGN
- Cláusula LINKAGE
- Instrucción CALL id-programa
- Instrucción CANCEL.
- Instrucción END PROGRAM
- Cláusula PADDING CHARACTER
- Párrafo PROGRAM-ID
- Instrucción ACQUIRE.
- Instrucción DROP.
- Como nombre-texto en una instrucción COPY
- Como nombre-biblioteca en una instrucción COPY

Nota: No puede utilizar caracteres DBCS para palabras o nombres COBOL. Consulte la publicación *ILE COBOL/400 Reference* para obtener información sobre las reglas de formato COBOL de los nombres de sistema, palabras reservadas y palabras definidas por el usuario, tales como nombres de datos y nombres de archivo.

Cómo escribir comentarios

Puede escribir comentarios que contengan caracteres DBCS en un programa COBOL poniendo un asterisco (*) o una barra inclinada (/) en la columna siete de la línea del programa. Cualquiera de estos símbolos hace que el compilador trate como documentación la información que sigue a la columna siete. La barra inclinada provoca también un salto de página. Dado que el compilador COBOL no comprueba el contenido de las líneas de comentario, los caracteres DBCS de los comentarios no se detectan. Los caracteres DBCS que no sean válidos pueden provocar una impresión incorrecta del listado del compilador.

Identification Division

Puede poner entradas de comentario que contengan caracteres DBCS en cualquier parte de Identification Division excepto en el párrafo PROGRAM-ID. El nombre de programa especificado en el párrafo PROGRAM-ID debe ser alfanumérico.

Environment Division

Configuration Section

Puede utilizar caracteres DBCS en las entradas de comentario sólo en el párrafo Configuration Section. Todos los nombre-función, nombres-mnemotécnicos, nombre-condición y nombres-alfabeto deben especificarse con caracteres alfanuméricos. Para la entrada SOURCE-COMPUTER y OBJECT-COMPUTER, utilice el nombre de sistema alfanumérico:

```
IBM-AS400
```

No puede utilizar literales mixtos en Configuration Section. En su lugar, utilice literales alfanuméricos para definir un nombre-alfabeto y el literal en la cláusula CURRENCY SIGN del párrafo SPECIAL-NAMES. No existe ningún alfabeto ni clase DBCS. Utilice el juego de caracteres EBCDIC en su lugar.

Input-Output Section

Especifique todos los nombres de datos, de archivo y de asignación con caracteres alfanuméricos. Puede utilizar los caracteres DBCS en los comentarios.

Para archivos indexados, el nombre de datos de la cláusula RECORD KEY puede hacer referencia a un dato DBCS de un registro.

No puede utilizar datos mixtos DBCS como RELATIVE KEY en archivos relativos.

Párrafo File Control

Cláusula ASSIGN: No puede utilizar literales que contengan caracteres DBCS en la cláusula ASSIGN para especificar un medio externo, como por ejemplo una impresora o una base de datos.

Data Division

File Section

Para la entrada FD (descripción de archivo), puede utilizar literales o datos DBCS en la cláusula VALUE OF. La cláusula DATA RECORDS sólo puede hacer referencia a datos. Dado que el compilador ILE COBOL/400 trata como documentación tanto la cláusula VALUE OF como la cláusula DATA RECORDS de File Section, ninguna de las dos tiene ningún efecto al ejecutar el programa.Sin embargo, el compilador COBOL comprueba todos los literales de la cláusula VALUE OF para asegurarse de que son válidos.

En el caso de las cintas magnéticas, el sistema sólo puede leer los caracteres DBCS de la cinta, o escribirlos en ella, con el formato EBCDIC. El sistema no puede realizar funciones de cinta en las que participe una cinta de formato ASCII. Defina el nombre-alfabeto de la cláusula CODE-SET como NATIVE o EBCDIC.

Working-Storage Section

Cláusula REDEFINES: Las reglas existentes para redefinir datos también son pertinentes para los datos que contienen caracteres DBCS. Al determinar la longitud de un dato que redefine o de un dato redefinido, recuerde que cada carácter DBCS es el doble de grande que un carácter alfanumérico.

Así mismo, asegúrese de que los datos redefinidos contienen los caracteres de control de desplazamiento cuándo y dónde sea necesario.

Cláusula OCCURS: Utilice esta cláusula para definir tablas para almacenar datos DBCS. Si especifica la expresión ASCENDING/DESCENDING KEY, COBOL supone que el contenido de la tabla está en el orden de clasificación de programa EBCDIC. Los caracteres de control de desplazamiento de los datos mixtos participan en el orden de clasificación.

Para obtener más información sobre el manejo de la tablas que contienen caracteres DBCS, consulte el apartado “Manejo de tablas–instrucción SEARCH” en la página 503.

Cláusula JUSTIFIED RIGHT: Utilice la cláusula JUSTIFIED RIGHT para alinear los datos DBCS en la posición situada más a la derecha de un campo de recepción elemental.Si el campo de recepción es más pequeño que el campo de envío, COBOL trunca los caracteres situados más a la derecha. Si el campo de recepción es más grande que el campo de envío, COBOL rellena con blancos el espacio no utilizado de la izquierda del campo de recepción.

La cláusula JUSTIFIED no afecta al valor inicial de la cláusula VALUE.

Cláusula VALUE: Puede utilizar literales mixtos para especificar un valor inicial para un dato que no sea numérico o para definir valores para las entradas nombre-condición de nivel 88. Los literales DBCS deben utilizarse para especificar los valores iniciales de datos DBCS o DBCS editados.

Los caracteres de control de desplazamiento que haya en el literal están considerados parte de la serie de imagen del literal, excepto cuando se utilizan para continuar en una línea nueva. Cuando se continua un literal mixto, el compilador *no* incluye el carácter de desplazamiento a teclado estándar de la columnas 71 ó 72 ni la comilla (") inicial y el carácter de desplazamiento a teclado ideográfico como

parte del literal mixto. Asegúrese, no obstante, de que el literal mixto no sobrepasa el tamaño del dato especificado en la cláusula PICTURE ya que, de lo contrario, se truncará.

Los literales DBCS pueden utilizarse para inicializar datos DBCS.

Cuando se utilizan literales que contienen caracteres DBCS en la cláusula VALUE de las entradas nombre-condición de nivel 88, COBOL trata como alfanuméricos los caracteres DBCS.Por lo tanto, siga las normas de especificación de datos alfanuméricos, incluido el permitir una opción THROUGH.Esta opción utiliza el orden de clasificación EBCDIC normal pero recuerde que los caracteres de control de desplazamiento en DBCS participan en el orden de clasificación.

Cláusula PICTURE: Utilice el símbolo X de PICTURE para definir datos mixtos y G o N para datos DBCS. Un dato DBCS que contenga *n* caracteres DBCS se definiría como:

```
PICTURE G(n) o PICTURE N(n)
```

Un dato mixto que contenga *m* caracteres SBCS y una serie de *n* caracteres DBCS se definiría como:

```
PICTURE X(m+2n+2)
```

Puede utilizar todos los símbolos PICTURE alfanuméricos editados para datos mixtos. Los símbolos de edición tienen el mismo efecto en los datos DBCS de estos elementos que en los datos alfanuméricos. Compruebe que ha obtenido los resultados deseados. Los datos DBCS puros sólo pueden utilizar el símbolo de edición B.

Cláusula RENAMES: Utilice esta cláusula para especificar agrupaciones alternativas de datos elementales. Las reglas existentes para redenominar datos alfanuméricos también son pertinentes para los datos DBCS.

Procedure Division

Funciones intrínsecas

Puede utilizar datos DBCS, literales DBCS y literales mixtos como argumentos en algunas funciones intrínsecas.

Las funciones intrínsecas también pueden devolver un dato DBCS si uno de los argumentos de la función intrínseca es un dato DBCS o un literal DBCS.

Para obtener más información acerca de las funciones intrínsecas que dan soporte a los elementos DBCS, consulte el capítulo sobre Funciones intrínsecas en el manual *ILE COBOL/400 Reference*.

Expresiones condicionales

Dado que los nombres-condición (entradas de nivel 88) pueden hacer referencia a datos que contienen caracteres DBCS, puede utilizar la condición nombre-condición para probar dichos datos (consulte el apartado “Cláusula VALUE” en la página 496). Siga las reglas relacionadas en la publicación *ILE COBOL/400 Reference* para utilizar los nombres de condición y las variables condicionales.

Puede utilizar datos DBCS o literales mixtos como operandos en una condición de relación. Dado que COBOL trata a los datos mixtos como alfanuméricos, todas las comparaciones se realizan de acuerdo con las reglas de los operandos alfanuméricos.Los datos DBCS sólo pueden compararse con otros datos DBCS. Tenga presente lo siguiente:

- El sistema no reconoce el contenido mixto.
- El sistema utiliza los códigos de desplazamiento en las comparaciones de datos mixtos.
- El sistema compara los datos utilizando el orden de clasificación EBCDIC o una secuencia definida por el usuario.
- En una comparación de elementos DBCS con elementos similares de tamaño desigual, se rellena por la derecha con espacios el elemento más pequeño.

Consulte el apartado dedicado al párrafo SPECIAL-NAMES de la publicación *ILE COBOL/400 Reference* para obtener más información.

Puede utilizar las condiciones de clase y las de estado de conmutación según lo descrito en la publicación *ILE COBOL/400 Reference*.

Instrucciones de entrada/salida

Instrucción ACCEPT: Los datos de entrada recibidos desde un dispositivo utilizando la instrucción ACCEPT de formato 1 pueden incluir DBCS. Todos los datos DBCS deben estar identificados mediante la sintaxis correcta. Los datos de entrada, excluidos los caracteres de control de desplazamiento, sustituyen al contenido existente de un dato DBCS. Los caracteres de control de desplazamiento se incluyen en el contenido de los datos mixtos. COBOL no realiza ninguna edición especial ni comprobación de errores en los datos.

Si utiliza la instrucción ACCEPT de formato 3 para obtener información OPEN-FEEDBACK sobre un archivo, esta información incluirá un campo que mostrará si el archivo tiene datos DBCS o mixtos.

La información recibida del área de datos local mediante la instrucción ACCEPT de formato 4 puede incluir series de caracteres DBCS o mixtos. La información recibida sustituye al contenido existente. COBOL no realiza ninguna edición ni comprobación de errores. Esto atañe también a la información recibida del área de datos PIP utilizando la instrucción ACCEPT de formato 5 y del área de datos definida por el usuario utilizando una instrucción ACCEPT de formato 9.

Con la instrucción ACCEPT de formato 6, puede obtener los atributos de una pantalla de estación de trabajo y de su teclado.Para las estaciones de pantalla que visualizan caracteres DBCS, el sistema establece el valor adecuado en el dato ATTRIBUTE-DATA. No puede utilizar caracteres DBCS para dar nombre a un dispositivo.

Si utiliza la instrucción ACCEPT ampliada (de formato 7) para la entrada de estación de trabajo a nivel de campo, debe asegurarse de que los datos DBCS no estén divididos en varias líneas. COBOL no realiza ninguna comprobación de errores ni edición, excepto para la eliminación de caracteres de desplazamiento a teclado ideográfico y estándar en los casos en que es necesario.

Instrucción DISPLAY: Puede especificar literales o datos DBCS o mixtos en la instrucción DISPLAY. Puede mezclar tipos de datos. Los datos DBCS y mixtos, procedentes de datos o literales, se envían tal y como aparecen al dispositivo de programa, al área de datos local o al área de datos definida por el usuario que sea el destino mencionado en la instrucción DISPLAY.

Dado que COBOL desconoce las características del dispositivo en el que se visualizan los datos, el usuario debe asegurarse de que los datos DBCS y mixtos sean correctos.

Nota: ALL es una opción válida para los literales mixtos.

Si utiliza una sentencia DISPLAY de formato 3 o de formato 4 para la salida a nivel de campo de la estación de trabajo, el usuario debe asegurarse de que los datos DBCS no estén divididos en varias líneas.

Instrucción READ: Puede utilizar datos DBCS como RECORD KEY para un archivo indexado. Consulte el apartado “Input-Output Section” en la página 495 para obtener más información.

Expresión INTO: Puede leer un registro en un dato DBCS utilizando la expresión INTO. Esta expresión hace que se realice una instrucción MOVE (sin la opción CORRESPONDING). El compilador mueve los datos DBCS de la misma forma que los alfanuméricos. No se asegura que sean válidos.

Instrucción REWRITE: Utilice la expresión FROM de esta instrucción para transferir datos DBCS de un dato DBCS a un registro existente. La expresión FROM hace que ambos tipos de datos se muevan de la misma forma que la expresión INTO con la instrucción READ (consulte el apartado “Instrucción READ”).

Instrucción START: Si utiliza caracteres DBCS en la clave de un archivo indexado, especifique el dato correspondiente en la expresión KEY de la instrucción START.

Uno de los puntos siguientes debe ser cierto:

- El dato debe ser el mismo que el especificado en la cláusula RECORD KEY del párrafo FILE-CONTROL.
- El dato tiene el mismo primer carácter que la clave de registro y su longitud no es superior a la de la clave de registro.

Puede especificar operadores válidos (tales como EQUAL, GREATER THAN, NOT LESS THAN) en la expresión KEY. El sistema puede seguir el orden de clasificación EBCDIC o uno definido por el usuario.

Instrucción WRITE: Utilice la expresión FROM de esta sentencia para escribir datos DBCS en un registro. Esta expresión mueve los datos de la misma forma que la instrucción REWRITE (consulte el apartado “Instrucción REWRITE”).

Al escribir datos en un archivo de dispositivo, debe incluir los caracteres de control de desplazamiento.

Instrucciones de manipulación de datos

Instrucciones aritméticas: Dado que COBOL trata los caracteres DBCS de la misma manera que los caracteres SBCS, no utilice caracteres DBCS en operaciones numéricas ni los manipule con instrucciones aritméticas.

Instrucción INSPECT: Puede utilizar cualquier dato DBCS como operando de la instrucción INSPECT. El sistema repasa y sustituye cada mitad de un carácter DBCS, incluidos los caracteres de control de desplazamiento en estas operaciones. Por lo tanto, es posible que los datos no concuerden correctamente.

Sólo puede utilizar operandos de caracteres DBCS con otros datos o literales de caracteres DBCS. Los operandos mixtos se tratan como alfanuméricos. Si utiliza la expresión REPLACING, puede provocar la sustitución por datos alfanuméricos de un dato mixto inspeccionado o viceversa.

No puede sustituir una serie de caracteres por una serie de longitud diferente. Téngalo presente a la hora de sustituir caracteres SBCS por caracteres DBCS en un dato mixto o viceversa.

Si desea controlar el uso de la instrucción INSPECT con elementos mixtos que contengan caracteres DBCS, defina datos que contengan caracteres de control de desplazamiento. Utilice los caracteres de desplazamiento a teclado ideográfico y estándar como operandos BEFORE/AFTER en la sentencia INSPECT.

El ejemplo siguiente muestra la forma en que puede utilizar la instrucción INSPECT para sustituir un carácter DBCS por otro en un dato mixto.

```
01  SUBJECT-ITEM                PICTURE X(50).
01  DBCS-CHARACTERS             VALUE "0EK1K20F".
    05  SHIFT-OUT               PICTURE X.
    05  DBCS-CHARACTER-1        PICTURE XX.
    05  DBCS-CHARACTER-2        PICTURE XX.
    05  SHIFT-IN                PICTURE X.
```

La instrucción INSPECT estaría codificada de la siguiente manera:

```
INSPECT SUBJECT-ITEM
    REPLACING ALL DBCS-CHARACTER-1
              BY  DBCS-CHARACTER-2
    AFTER INITIAL SHIFT-OUT.
```

Nota: La utilización de la expresión AFTER INITIAL SHIFT-OUT sirve de ayuda para evitar el riesgo de sustituir de forma accidental dos caracteres alfanuméricos consecutivos que tengan los mismos valores EBCDIC que DBCS-CHARACTER-1 (en aquellos casos en los que SUBJECT-ITEM contiene datos mixtos).

También puede utilizar la sentencia INSPECT para determinar si el dato contiene caracteres DBCS, de manera que pueda tener lugar el proceso oportuno. Por ejemplo:

```
01  SUBJECT-FIELD          PICTURE X(50).
01  TALLY-FIELD            PICTURE 9(3) COMP.
01  SHIFTS                 VALUE "0E0F".
    05  SHIFT-OUT          PICTURE X.
    05  SHIFT-IN           PICTURE X.
```

En Procedure Division podría entrar lo siguiente:

```
MOVE ZERO TO TALLY-FIELD.
INSPECT SUBJECT-FIELD TALLYING TALLY-FIELD
                                FOR ALL  SHIFT-OUT.
IF TALLY-FIELD IS GREATER THAN ZERO THEN
    PERFORM DBCS-PROCESSING
ELSE
    PERFORM A-N-K-PROCESSING.
```

Instrucción MOVE: Todos los caracteres DBCS se mueven como series de caracteres alfanuméricos. El sistema no convierte los datos ni los examina.

Puede mover literales mixtos a elementos de grupo y elementos alfanuméricos. Sólo puede mover datos o literales DBCS a datos DBCS.

Si la longitud del campo de recepción es diferentes de la del campo de envío, COBOL realiza una de las acciones siguientes:

- Trunca los caracteres del elemento de envío si su longitud es superior a la del elemento de recepción. Esta operación puede reducir la integridad de los datos.
- Rellena con blancos el elemento de envío si su longitud es inferior a la del elemento de recepción.

Para saber más sobre el efecto de los símbolos de edición de la cláusula PICTURE del dato de recepción, consulte la publicación *ILE COBOL/400 Reference*.

Instrucción SET (formato nombre-condición): Cuando se establece el nombre de condición como TRUE en esta instrucción, COBOL mueve el literal de la cláusula VALUE al dato asociado. Puede mover un literal con caracteres DBCS.

Instrucción STRING: Puede utilizar la instrucción STRING para construir un dato que contenga subcampos DBCS. Se mueven todos los datos de los literales o datos fuente, incluidos los caracteres de control de desplazamiento, al dato de recepción, mitad a mitad de los caracteres DBCS.

Instrucción UNSTRING: La instrucción UNSTRING trata los datos DBCS y mixtos como los alfanuméricos. La operación UNSTRING se realiza mitad a mitad de los caracteres DBCS.

Los datos pueden contener caracteres DBCS y alfanuméricos dentro del mismo campo.

Utilice la expresión DELIMITED BY para localizar subcampos de doble byte y alfanuméricos dentro de un campo de datos. Identifique los datos que contienen caracteres de control de desplazamiento y utilice dichos datos como identificadores en la expresión DELIMITED BY. Vea los ejemplos siguientes para obtener más información sobre la manera de hacerlo.Utilice la variable POINTER para continuar explorando los subcampos del campo de envío.

Una vez el sistema haya realizado la operación UNSTRING, usted puede contrastar los delimitadores almacenados por las expresiones DELIMITER IN con los valores de carácter de control de desplazamiento para ver qué subcampos contienen DBCS y cuáles caracteres alfanuméricos.

El ejemplo siguiente muestra la forma en que podría configurar los campos para preparar la operación UNSTRING de una serie de caracteres que contenga datos mixtos:

```
01  SUBJECT-FIELD      PICTURE X(40)
01  FILLER.
    05  UNSTRING-TABLE OCCURS 4 TIMES.
        10  RECEIVER   PICTURE X(40).
        10  DELIMTR    PICTURE X.
        10  COUNTS     PICTURE 99 COMP.
01  SHIFTS             VALUE "0E0F".
    05  SHIFT-OUT      PICTURE X.
    05  SHIFT-IN       PICTURE X.
```

El código de la instrucción UNSTRING es el siguiente:

```
UNSTRING SUBJECT-FIELD  DELIMITED BY SHIFT-OUT
                                     OR SHIFT-IN
INTO RECEIVER (1) DELIMITER IN DELIMTR (1)
                  COUNT     IN COUNTS (1)
INTO RECEIVER (2) DELIMITER IN DELIMTR (2)
                  COUNT     IN COUNTS (2)
INTO RECEIVER (3) DELIMITER IN DELIMTR (3)
                  COUNT     IN COUNTS (3)
INTO RECEIVER (4) DELIMITER IN DELIMTR (4)
                  COUNT     IN COUNTS (4)
ON OVERFLOW PERFORM UNSTRING-OVERFLOW-MESSAGE.
```

Esta instrucción UNSTRING divide una serie de caracteres en sus partes alfanumérico y DBCS. Suponiendo que los datos de la serie de caracteres sean válidos, un valor de delimitador de desplazamiento a teclado ideográfico indica que el campo de recepción correspondiente contiene datos alfanuméricos, mientras que un valor de desplazamiento a teclado estándar indica que el campo de recepción correspondiente tiene datos DBCS.Puede comprobar los datos COUNT para determinar si cada campo de recepción ha recibido algún carácter. operación UNSTRING que se acaba de describir:

SUBJECT-FIELD = ABC0_EK1K2K30$_F$D0$_E$K4K5K60$_F$
RECEIVER (1) = ABC DELIMTR (1) = 0_E COUNTS (1) = 3
RECEIVER (2) = K1K2K3 DELIMTR (2) = 0_F COUNTS (2) = 6
RECEIVER (3) = D DELIMTR (3) = 0_E COUNTS (3) = 1
RECEIVER (4) = K4K5K6 DELIMTR (4) = 0_F COUNTS (4) = 6

SUBJECT-FIELD = 0_EK1K2K30$_F$ABC0$_E$K40$_F$
RECEIVER (1) = (blanks) DELIMTR (1) = 0_E COUNTS (1) = 0
RECEIVER (2) = K1K2K3 DELIMTR (2) = 0_F COUNTS (2) = 6
RECEIVER (3) = ABC DELIMTR (3) = 0_E COUNTS (3) = 3
RECEIVER (4) = K4 DELIMTR (4) = 0_F COUNTS (4) = 2

Sentencias de bifurcación de procedimiento

Puede utilizar un literal mixto como operando de la instrucción STOP. Al hacerlo, el sistema visualiza el literal tal y como lo haya entrado en la estación de trabajo para trabajos interactivos.Para trabajos por lotes, el sistema visualiza signos de subrayado en aquellos puntos en los que normalmente aparecería el literal en la cola de mensajes del operador del sistema. El sistema no edita ni comprueba el contenido del literal.

Manejo de tablas–instrucción SEARCH

Puede realizar una instrucción SEARCH de formato 1 (búsqueda secuencial de una tabla) en una tabla que contenga datos DBCS de mitad en mitad de los caracteres DBCS.

También puede realizar una instrucción SEARCH de formato 2 (SEARCH ALL) en una tabla DBCS. Ordene la tabla según el orden de clasificación elegido.

Nota: Los caracteres de control de desplazamiento de los datos DBCS participan en la comparación.

SORT/MERGE

Puede utilizar los datos DBCS como claves en una instrucción SORT o MERGE. La operación SORT ordena los datos de acuerdo con el orden de clasificación especificado en el párrafo SORT, MERGE o SPECIAL NAMES. El sistema ordena los caracteres de control de desplazamiento que haya contenidos en las claves DBCS y mixtas.

Utilice la instrucción RELEASE para transferir registros que contengan caracteres DBCS de un área de entrada/salida a la fase inicial de la operación de clasificación. El sistema realiza la expresión FROM con la instrucción RELEASE de la misma forma que realiza la expresión FROM con la sentencia WRITE (consulte el apartado “Instrucción WRITE” en la página 499).

También puede utilizar la instrucción RETURN para transferir registros que contengan caracteres DBCS de la fase final de una operación de clasificación o fusión a un área de entrada/salida. El sistema realiza la expresión INTO con la instrucción RETURN de la misma forma que realiza la expresión INTO con la sentencia READ (consulte el apartado “Instrucción READ” en la página 499).

Instrucciones que dirigen al compilador

Instrucción COPY

Puede utilizar la instrucción COPY para copiar texto fuente que contenga caracteres DBCS en un programa COBOL. Al hacerlo, asegúrese de que especifica el nombre de miembro, de archivo y de biblioteca utilizando datos alfanuméricos y que los especifica según las reglas expuestas en la publicación *ILE COBOL/400 Reference*.

Utilice la instrucción COPY de formato 2 para copiar los campos definidos en las especificaciones de descripción de datos (DDS). Los datos DBCS (el valor de la columna 35 del formato DDS es G) y mixtos (el valor de la columna 35 del formato DDS es O) se copian en un programa COBOL en el formato PICTURE X(n). Si se selecciona *PICGGRAPHIC, los datos DBCS (formato G) se copian en el formato PICTURE G(n). El listado del compilador no indica que estos campos contengan caracteres DBCS, a menos que un campo sea el campo clave. En tales casos, el sistema imprime una O en la tabla de comentarios de las claves.

Los datos gráficos DBCS se copian en un programa COBOL en el formato PICTURE X(n). El listado del compilador indica que estos campos contienen datos gráficos. Consulte el apartado “Campos gráficos DBCS” en la página 335 para obtener una descripción del tipo de datos DBCS.

Puede poner caracteres DBCS en los comentarios de texto copiados de las DDS si el campo en las DDS asociadas tiene comentarios.

Si especifica la expresión REPLACING de la instrucción COPY, considere lo siguiente:

- El pseudotexto puede contener cualquier combinación de caracteres DBCS y alfanuméricos.
- Puede utilizar literales con contenido DBCS.
- Los identificadores pueden hacer referencia a datos que contengan caracteres DBCS.

Instrucción REPLACE

La instrucción REPLACE se parece a la expresión REPLACING de la instrucción COPY, con la excepción de que actúa sobre el programa fuente entero y no sólo sobre el texto de las bibliotecas COPY.

Si especifica la instrucción REPLACE, considere lo siguiente:

- El pseudotexto puede contener cualquier combinación de caracteres DBCS y alfanuméricos.
- Puede utilizar literales con contenido DBCS.
- Los identificadores pueden hacer referencia a datos que contengan caracteres DBCS.

Instrucción TITLE

Puede utilizar literales DBCS como literal de la instrucción TITLE.

Comunicaciones entre programas

Puede especificar entradas para datos alfanumérico que contengan caracteres DBCS o mixtos en la Linkage Section de la Data Division. Si los datos DBCS o los literales DBCS se pasan a un programa, también puede definir los elementos de la Linkage Section de recepción como datos DBCS.

Puede pasar caracteres DBCS de un programa a otro especificando dichos datos en la expresión USING. USING BY CONTENT y USING BY VALUE permiten que se pasen literales mixtos y DBCS.

No puede utilizar caracteres DBCS en la instrucción CALL para el nombre-programa del programa al que se llama. No puede utilizar caracteres DBCS en la instrucción CANCEL porque especifican nombres-programa.

Distintivos FIPS

Las mejoras del lenguaje COBOL que le permiten utilizar caracteres DBCS se señalan (identifican) por medio de distintivos FIPS (norma de proceso de información federal) proporcionadas por el compilador como extensiones IBM.

Listados de programa COBOL

En los listados originados en archivos fuente que admiten DBCS y generados en sistemas que admiten DBCS pueden aparecer caracteres DBCS.

Los caracteres DBCS que aparecen en un listado de programa se originan en el archivo fuente, el texto fuente generado por la instrucción COPY o los mensajes del compilador COBOL.

Un listado que contenga caracteres DBCS debe ser la salida de un archivo de impresora capaz de procesar datos DBCS.Los listados que contienen caracteres DBCS se manejan correctamente si una de las condiciones siguientes es cierta:

- El archivo fuente está definido como capaz de contener datos DBCS con el parámetro IGCDTA del mandato CRTSRCPF. En tal caso, el programa altera temporalmente el valor existente del atributo del archivo de salida de impresora.
- El usuario ha especificado el atributo necesario para la impresora de salida con el parámetro IGCDTA del mandato OVRPRTF antes de compilar el programa.

Nota: El parámetro IGCDTA sólo está disponible en sistemas DBCS y no puede definirse ni visualizarse en sistemas no DBCS.No obstante, puede crear objetos con atributos DBCS en un sistema no DBCS copiándolos desde un sistema DBCS.Si lo hace, compruebe las posibles incompatibilidades.

El compilador puede utilizar caracteres del programa fuente como parámetros de sustitución en los mensajes del compilador y del corrector de sintaxis.El sistema no comprueba ni edita los parámetros de sustitución. Si no especifica correctamente los caracteres DBCS, es posible que el sistema imprima o visualice incorrectamente parte de los mensajes.

Funciones intrínsecas sensibles al orden de clasificación

Las funciones intrínsecas CHAR y ORD dependen de las posiciones ordinales de los caracteres. Estas funciones intrínsecas no se proporcionan para tipos de datos DBCS (por ejemplo, se soportan para caracteres de un solo byte, alfabéticos o numéricos). El resultado de estas funciones se base en el orden de clasificación vigente. La página de códigos actual no afecta el resultado de estas funciones intrínsecas.

Apéndice E. Ejemplo de vuelco con formato COBOL

La Figura 127 en la página 510 muestra un ejemplo de vuelco con formato COBOL. Normalmente, hay un vuelco disponible si algo va mal al intentar ejecutar el programa.

Puede solicitar dos tipos de vuelco, un vuelco de datos y un vuelco ampliado. El ejemplo de la Figura 127 en la página 510 es el de un vuelco ampliado.

El **vuelco de datos** contiene la información siguiente. Las etiquetas identifican el punto del vuelco con formato en el que hallará la información.

- A Nombre de cada variable
- B Tipo de datos
- C Valor por omisión
- D Valor hexadecimal

Nota: En el vuelco sólo se muestran los primeros 250 caracteres.

El **vuelco ampliado** también contiene la información adicional siguiente. Las etiquetas identifican el punto del vuelco con formato en el que hallará la información.

- E Nombre de cada archivo
- F Nombre de sistema de cada archivo
- G Distintivo externo/interno
- H Ultima operación de E/S intentada
- I Ultimo estado de archivo
- J Ultimo estado ampliado
- K Estado de apertura/cierre
- L Información de bloques
- M Factor de bloqueo
- N Información del área de información de retorno de E/S
- O Información del área de información de retorno de apertura

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPDUMP         AS400SYS   96/07/04 11:38:12         Página     2

                                 F u e n t e

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTIFN  S NOMCOPIA   FEC CAMB


    1     000100 IDENTIFICATION DIVISION.
    2     000200 PROGRAM-ID.     SAMPDUMP.
          000300
    3     000400 ENVIRONMENT DIVISION.
    4     000500 CONFIGURATION SECTION.
    5     000600   SOURCE-COMPUTER. IBM-AS400.
    6     000700   OBJECT-COMPUTER. IBM-AS400.
    7     000800 INPUT-OUTPUT SECTION.
    8     000900 FILE-CONTROL.
    9     001000     SELECT FILE-1 ASSIGN TO DISK-DBSRC.                                          94/04/07
   11     001100 DATA DIVISION.
   12     001200 FILE SECTION.
   13     001300 FD  FILE-1.
   14     001400 01  RECORD-1.
   15     001500   05 R-TYPE             PIC X(1).
   16     001600   05 R-AREA-CODE        PIC 9(2).
   17     001700     88 R-NORTH-EAST VALUES 15 THROUGH 30.
   18     001800   05 R-SALES-CAT-1      PIC S9(5)V9(2) COMP-3.
   19     001900   05 R-SALES-CAT-2      PIC S9(5)V9(2) COMP-3.
   20     002000   05 FILLER             PIC X(1).
          002100
   21     002200 WORKING-STORAGE SECTION.
   22     002300 01  W-SALES-VALUES.
   23     002400     05 W-CAT-1          PIC S9(8)V9(2).
   24     002500     05 W-CAT-2          PIC S9(8)V9(2).
   25     002600     05 W-TOTAL          PIC S9(8)V9(2).
          002700
   26     002800 01  W-EDIT-VALUES.
   27     002900     05 FILLER           PIC X(8) VALUE "TOTALS: ".
   28     003000     05 W-EDIT-1         PIC Z(7)9.9(2)-.
   29     003100     05 FILLER           PIC X(3) VALUE SPACES.
   30     003200     05 W-EDIT-2         PIC Z(7)9.9(2)-.
   31     003300     05 FILLER           PIC X(3) VALUE SPACES.
   32     003400     05 W-EDIT-TOTAL     PIC Z(7)9.9(2)-.
          003500
   33     003600 01  END-FLAG            PIC X(1) VALUE SPACE.
   34     003700     88 END-OF-INPUT VALUE "Y".
          003800
   35     003900 PROCEDURE DIVISION.
          004000 MAIN-PROGRAM SECTION.
          004100 MAINLINE.
          004200*****************************************************************
          004300* ABRIR ARCHIVO DE ENTRADA Y BORRAR TOTALES.                    *
          004400*****************************************************************
   36     004500     OPEN INPUT FILE-1.
   37     004600     MOVE ZEROS TO W-SALES-VALUES.
          004700
          004800*****************************************************************
          004900* LEER ARCHIVO DE ENTRADA PROCESANDO SOLO LOS REGISTROS DEL    *
          005000* AREA NORESTE. AL LLEGAR FIN-DE-ENTRADA, ESTABLECER INDICADOR *
          005100*****************************************************************
   38     005200     PERFORM UNTIL END-OF-INPUT
   39     005300         READ FILE-1
   40     005400             AT END SET END-OF-INPUT TO TRUE
          005500         END-READ
```

Figura 126 (Parte 1 de 2). Programa COBOL utilizado para generar un vuelco con formato COBOL

```
5716CB1 V3R7M0  961108 LN        IBM ILE COBOL/400       TESTLIB/SAMPDUMP         AS400SYS   96/07/04 11:38:12         Página     3

 INST NA NUMSEC -A 1 B..+....2....+....3....+....4....+....5....+....6....+....7..IDENTFCN S NOMCOPIA   FEC CAMB


   41     005600         IF R-NORTH-EAST AND NOT END-OF-INPUT THEN
   42     005700             ADD R-SALES-CAT-1 TO W-CAT-1, W-TOTAL
   43     005800             ADD R-SALES-CAT-2 TO W-CAT-2, W-TOTAL
          005900         END-IF
          006000     END-PERFORM.
          006100
          006200*****************************************************************
          006300* VISUALIZAR RESULTADOS Y FINALIZAR PROGRAMA.                   *
          006400*****************************************************************
   44     006500     MOVE W-CAT-1 TO W-EDIT-1.
   45     006600     MOVE W-CAT-2 TO W-EDIT-2.
   46     006700     MOVE W-TOTAL TO W-EDIT-TOTAL.
   47     006800     DISPLAY W-EDIT-VALUES.
   48     006900     STOP RUN.

                        * * * * *  F I N  D E  F U E N T E  * * * * *
```

Figura 126 (Parte 2 de 2). Programa COBOL utilizado para generar un vuelco con formato COBOL

```
LNR7200 exception in module 'SAMPDUMP  ', program 'SAMPDUMP  ' in library 'TESTLIB   ' at statement number 42.

Formatted data dump for module 'SAMPDUMP  ', program 'SAMPDUMP  ' in library 'TESTLIB   '.
NAME     ATTRIBUTE                     VALUE
DB-FORMAT-NAME  A
         CHAR(10)  B                    "DBSRC     "  C
                                       "C4C2E2D9C34040404040"X  D
END-FLAG
         CHAR(1)                       " "
                                       "40"X
R-AREA-CODE   OF RECORD-1 OF FILE-1
         ZONED(2 0)                     0.
                                       "0000"X
R-SALES-CAT-1   OF RECORD-1 OF FILE-1
         PACKED(7 2)                   00000.00
                                       "00000000"X
R-SALES-CAT-2   OF RECORD-1 OF FILE-1
         PACKED(7 2)                   0000Û.7
                                       "0000B7A0"X
RETURN-CODE
         BIN(2)                        0000.
                                       "0000"X
W-CAT-1   OF W-SALES-VALUES
         ZONED(10 2)                   00311111.08
                                       "F0F0F3F1F1F1F1F1F0F8"X
W-CAT-2   OF W-SALES-VALUES
         ZONED(10 2)                   00622222.16
                                       "F0F0F6F2F2F2F2F2F1F6"X
W-EDIT-TOTAL   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-EDIT-1   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-EDIT-2   OF W-EDIT-VALUES
         CHAR(12)                      "            "
                                       "40404040404040404040404040"X
W-TOTAL   OF W-SALES-VALUES
         ZONED(10 2)                   00933333.24
                                       "F0F0F9F3F3F3F3F2F4"X
                        E            F
Current active file: FILE-1 (DISK-DBSRC).
Information pertaining to file FILE-1 (DISK-DBSRC).
  File is internal.  G
  Last I-O operation attempted for file: READ.  H
  Last file status: '00'.  I
  Last extended file status: '    '.  J
  File is open.  K
  Blocking is in effect.  L
    Blocking factor: 17.  M
  I-O Feedback Area.  N
    Number of successful PUT operations: 0.
```

Figura 127 (Parte 1 de 2). Ejemplo de vuelco con formato COBOL

```
   Number of successful GET operations: 1.
   Number of successful PUTGET operations: 0.
   Number of other successful operations: 0.
   Current data management operation: 1.
   Record format: 'DBSRC     '.
   Device class and type: '  '.
   Program device name: '          '.
   Length of last record: 228.
   Number of records for blocked PUT or GET: 17.
   Length of all data returned: 0.
   Number of blocks successfully read or written: 0.
   Offset: '090'. Value: '00000000000000000001000004800004'.
   Offset: '0A0'. Value: '00000000000000000001000000110000'.
   Offset: '0B0'. Value: '0000'.
 Open Feedback Area.  O
   Actual file name: 'DBSRC     '.
   Actual library name: 'TESTLIB   '.
   Member name: 'SALES     '.
   File type: 21.
   Open file count: 1.
   Max record length: 0.
   CCSID: 65535.
   Offset: '000'. Value: 'C4C2C4C2E2D9C34040404040D9D4C9E2'.
   Offset: '010'. Value: 'E3D9E8404040000000000000000000000'.
   Offset: '020'. Value: '00000000000000000000000000E40000'.
   Offset: '030'. Value: 'E2C1D3C5E24040404040FFFFFFE00000'.
   Offset: '040'. Value: '00000015000000000000000000011C1'.
   Offset: '050'. Value: 'D900D5A5000000000000500000000000'.
   Offset: '060'. Value: '00000000000000000000011000000EF00'.
   Offset: '070'. Value: '0003E0000000000000000000000000001'.
   Offset: '080'. Value: '0000000102007300000FFFF0000000000'.
   Offset: '090'. Value: '00010001C4C1E3C1C2C1E2C540400000'.
   Offset: '0A0'. Value: '00000000000000000302000E00450045'.
   Offset: '0B0'. Value: '0045004500450045006F004500450045'.
   Offset: '0C0'. Value: '00450BFD068E0045000D001100000001'.
   Offset: '0D0'. Value: '00000000000000000000000000000000'.
   Offset: '0E0'. Value: '00000000000000000000000000000000'.
   Offset: '0F0'. Value: '00000000000000000000000000000000'.
   Offset: '100'. Value: '00000000000000000000000000000000'.
   Offset: '110'. Value: '000000000000'.
```

Figura 127 (Parte 2 de 2). Ejemplo de vuelco con formato COBOL

Apéndice F. Consideraciones sobre migración y compatibilidad entre OPM COBOL/400 e ILE COBOL/400

En este apéndice se describen las diferencias entre ILE COBOL/400 y OPM COBOL/400.

Si traslada sus programas y aplicaciones OPM COBOL/400 existentes, a ILE COBOL/400, debe tener presente las diferencias que hay entre el compilador OPM COBOL/400 y el compilador ILE COBOL/400. En algunos casos, será necesario realizar cambios en los programas.

Estrategia de migración

Al migrar los programas y aplicaciones OPM COBOL/400 existentes a ILE COBOL/400, se recomienda la siguiente estrategia:

- Migre aplicaciones enteras (o unidades de ejecución COBOL) de una en una a un entorno ILE puro en lugar de migrar los programas de uno en uno.
- Correlacione una unidad de ejecución COBOL con un grupo de activación ILE. Por ejemplo, para una unidad de ejecución COBOL que contenga una serie de programas COBOL, puede realizar una de las acciones siguientes para conservar la semántica de la unidad de ejecución COBOL:
  - Crear todos los programas COBOL con el mandato CRTBNDCBL. En tal caso, todos los programas se ejecutarán en el grupo de activación QILE.
  - Crear todos los programas COBOL con el mandato CRTCBLMOD seguido de CRTPGM con ACTGRP(cualquiera). En tal caso, todos los programas se ejecutarán en el grupo de activación denominado "cualquiera".
  - Crear el primer programa COBOL con ACTGRP(*NEW) utilizando el mandato CRTPGM y crear el resto de programas de la aplicación con ACTGRP(*CALLER). En tal caso, todos los programas se ejecutarán en el grupo de activación *NEW del primer programa COBOL.
- Asegúrese de que el emisor de la llamada a programas creador con la opción ACTGRP(*CALLER) del mandato no sea un programa OPM.

  Nota: *No* se recomienda mezclar programas OPM COBOL/400 y ILE COBOL/400 en la misma unidad de ejecución.
- Preste especial atención a las funciones de sistema que permiten diferentes opciones de ámbito.Por ejemplo, el ámbito por omisión de las funciones de sistema siguientes se cambia por *ACTGRPDFN (nivel de grupo de activación) cuando se utilizan en un grupo de activación ILE mientras que tienen otros valores por omisión, tales como *CALLLVL (nivel de llamada), cuando se utilizan en programas OPM.
  - Para OPNDBF y OPNQRYF, es posible que sea necesario cambiar OPNSCOPE dependiendo de la aplicación.Por ejemplo, si se ejecuta la aplicación en grupos de activación diferentes y es necesario compartir archivos, será necesario que cambie el ámbito por *JOB.
  - Alteraciones temporales
  - Control de compromiso.
- RCLRSRC no tiene ningún efecto sobre los grupos de activación ILE. En su lugar, utilice RCLACTGRP para borrar grupos de activación ILE.

Consideraciones sobre compatibilidad

En este apartado se describen las consideraciones sobre compatibilidad entre ILE COBOL/400 y OPM COBOL/400.

Consideraciones generales

Comprobación de área

En ILE COBOL/400, la comprobación de área está activa sólo para el primer símbolo de una línea. Los símbolos posteriores no se comprueban para ver si están en el área correcta.

El compilador OPM COBOL/400 comprueba todos los símbolos.

Campos de atributos de la sección Data Division Map del listado del compilador

En ILE COBOL/400, los atributos de los cuales sólo se ha comprobado la sintaxis (por ejemplo, la información de SAME SORT AREA, SAME SORT-MERGE AREA, SAME AREA, LABEL) no se notifican en la sección Data Division Map del listado del compilador.

En ILE COBOL/400, los nombres de condición no se listan en la sección Data Division Map del listado del compilador.

OPM COBOL/400 lista los nombres de condición, pero no especifica la información de atributos.

Archivos MIXED, COMMUNICATIONS y BSC

Los archivos MIXED, COMMUNICATIONS y BSC no están soportados en ILE COBOL/400. Estos tipos de archivo son válidos en el entorno de los sistemas 38 y no están soportados por el compilador ILE COBOL/400 durante la compilación (para COPY DDS) ni durante la ejecución.

Palabras reservadas

ILE COBOL/400 da soporte a una serie de palabras reservadas que no está soportadas actualmente por OPM COBOL/400. Por ejemplo, SORT-RETURN y RETURN-CODE son registros especiales. La aparición de SORT-RETURN o RETURN-CODE en un programa OPM COBOL/400 generaría un mensaje de gravedad 10, lo que indica que son palabras reservadas en otras implementaciones de COBOL.

ILE COBOL/400 reconoce estas palabras reservadas y, en situaciones similares, emite un mensaje de gravedad 30 que indica que se ha encontrado una palabra reservada en un punto en el que se requeriría una palabra definida por el usuario.

Archivos fuente para estructuras de datos CPI SAA

En ILE COBOL/400, los archivos fuente para estructuras de datos CPI SAA se encuentran en el archivo QCBLLESRC de la biblioteca QSYSINC.

En OPM COBOL/400, los archivos fuente para estructuras de datos CPI SAA se encuentran en el archivo QILBINC de las bibliotecas QLBL y QLBLP.

Mandatos CL

Mandato CRTCBLPGM sustituido por los mandatos CRTCBLMOD y CRTBNDCBL

El compilador OPM COBOL/400 lo invoca el mandato CL CRTCBLPGM. El mandato CL CRTCBLPGM crea un objeto *PGM.

El compilador ILE COBOL/400 lo invoca el mandato CL CRTCBLMOD o CRTBNDCBL.El mandato CL CRTCBLMOD crea un objeto *MODULE y el mandato CL CRTBNDCBL crea un objeto *PGM.

Las opciones y parámetros CRTCBLPGM siguientes (y sus opciones de instrucción PROCESS asociadas) no se encuentran en los mandatos CRTCBLMOD y CRTBNDCBL:

- Parámetro GENOPT (los restantes detalles de GENOPT se han trasladado a los detalles de OPTION)
- Parámetro PRTFILE
- Parámetro SAAFLAG
- Parámetro DUMP
- Parámetro ITDUMP
- Opción NOSRCDBG/SRCDBG del parámetro OPTION
- Opción NOLSTDBG/LSTDBG del parámetro OPTION
- Opción PRINT/NOPRINT del parámetro OPTION
- Opción LIST/NOLIST del parámetro GENOPT
- Opción NOPATCH/PATCH del parámetro GENOPT
- Opción NODUMP/DUMP del parámetro GENOPT
- Opción NOATR/ATR del parámetro GENOPT
- Opción NOOPTIMIZE/OPTIMIZE del parámetro GENOPT
- Opción STDERR/NOSTDERR del parámetro GENOPT
- Opción NOEXTACCDSP/EXTACCDSP del parámetro GENOPT
- Opción FS21DUPKY/NOFS21DUPKY del parámetro GENOPT

Los parámetros y opciones siguientes han cambiado:

- Para el parámetro SRCFILE, el nombre de archivo fuente por omisión es QCBLLESRC
- Para el parámetro CVTOPT, la palabra clave GRAPHIC/NOGRAPHIC de CRTCBLPGM se ha cambiado por PICXGRAPHIC/NOPICXGRAPHIC en CRTCBLMOD y CRTBNDCBL.
- Para el parámetro MSGLMT, el nivel de gravedad máximo por omisión es 30
- Para el parámetro GENLVL, el nivel de gravedad por omisión es 30
- Para el parámetro FLAGSTD, las opciones NOSEG/SEG1/SEG2 y NODEB/DEB1/DEB2 de CRTCBLPGM ya no existen en CRTCBLMOD ni CRTBNDCBL.
- Para el parámetro OPTION, el valor por omisión de la opción NOUNREF/UNREF se ha cambiado por NOUNREF.
- Para el parámetro OPTION, el valor por omisión de la opción NOSECLVL/SECLVL se ha cambiado por NOSECLVL.

Los parámetros y opciones siguientes son nuevos en los mandatos CRTCBLMOD y CRTBNDCBL:

- Parámetro MODULE sólo para CRTCBLMOD
- Parámetro PGM sólo para CRTBNDCBL

- Parámetro OUTPUT
- Parámetro DBGVIEW
- Parámetro OPTIMIZE
- Parámetro LINKLIT
- Parámetro SIMPLEPGM sólo para CRTBNDCBL
- Opción MONOPRC/NOMONOPRC del parámetro OPTION
- Opción NOSTDTRUNC/STDTRUNC del parámetro OPTION
- Opción NOIMBEDERR/IMBEDERR del parámetro OPTION
- Opción NOCHGPOSSGN/CHGPOSSGN del parámetro OPTION
- Opción NOEVENTF/EVENTF del parámetro OPTION
- Opción NOPICGGRAPHIC/PICGGRAPHIC del parámetro CVTOPT
- Opción NOFLOAT/FLOAT del parámetro CVTOPT
- Parámetro ENBPFRCOL
- Parámetro BNDDIR sólo para CRTBNDCBL
- Parámetro ACTGRP sólo para CRTBNDCBL

Todas las supresiones, cambios y adiciones realizados en los parámetros y opciones quedan reflejados también en los cambios asociados efectuados en las opciones de instrucción PROCESS.

Se ha añadido la opción de instrucción NOGRAPHIC PROCESS a ILE COBOL/400 como valor por omisión de la opción GRAPHIC de la instrucción PROCESS.

Las opciones de instrucción PROCESS de OPM COBOL/400 siguientes no se encuentran en ILE COBOL/400:

- FS9MTO0M/NOFS9MTO0M
- FS9ATO0A/NOFS9ATO0A

Identificadores de juego de caracteres (CCSID)

En ILE COBOL/400, la normalización de CCSID se realiza en el CCSID del archivo fuente primario.En OPM COBOL/400, se realiza en el CCSID del trabajo de compilación.

Tipo de miembro fuente por omisión

En ILE COBOL/400, el tipo de miembro fuente por omisión es CBLLE. En OPM COBOL/400, es CBL.

Mensajes de error

En ILE COBOL/400, los mensajes de error de compilación llevan el prefijo LNC. Además, algunos números de mensaje no siempre son los mismos en OPM COBOL/400.

Parámetro GENLVL

ILE COBOL/400 *no* genera ningún código cuando se produce un error con un nivel de gravedad *mayor o igual que* la gravedad especificada por GENLVL.

OPM COBOL/400 *no* genera ningún código cuando se produce un error con un nivel de gravedad *mayor que* la gravedad especificada por GENLVL.

Distintivos SAA

Los distintivos SAA no están soportados en ILE COBOL/400.

Mandatos CL STRCBLDBG y ENDCBLDBG

Los mandatos STRCBLDBG y ENDCBLDBG no están soportados en ILE COBOL/400.

Instrucciones que dirigen al compilador

Instrucción COPY

Comentarios después de un campo de longitud variable: En OPM COBOL/400, un fuente de DDS con el tipo de datos G y VARLEN dará lo siguiente:

```
06    FILLER      PIC X(10)
      (Campo de longitud variable)
```

ILE COBOL/400 añade un comentario después del comentario del campo de longitud variable, que es más preciso:

```
06    FILLER      PIC X(10)
      (Campo de longitud variable)
      (Campo gráfico)
```

Nombre de archivo fuente por omisión: En ILE COBOL/400, el nombre de archivo fuente por omisión es QCBLLESRC. En ILE COBOL/400, la instrucción COPY sin el calificador de fuente utilizará QCBLLESRC. Si se utiliza el nombre de archivo por omisión y no se encuentra el miembro fuente en el archivo QCBLLESRC, también se comprobará el archivo QLBLSRC.

En OPM COBOL/400 el nombre de archivo fuente por omisión es QLBLSRC.

Instrucción PROCESS

Instrucción *CBL/*CONTROL: Si se encuentra *CONTROL en la instrucción PROCESS, no se maneja como directriz sino como opción PROCESS no válida. La directriz *CBL/*CONTROL debe ser la única instrucción de una línea dada.

Opciones INTERMEDIATE y MINIMUM (distintivos FIPS): En ILE COBOL/400, si no se solicita la colocación de distintivos FIPS en los mandatos CRTCBLMOD o CRTBNDCBL y hay una instrucción COPY dentro de una instrucción PROCESS, no se colocará ningún distintivo FIPS en el miembro de copia cuando se especifique INTERMEDIATE o MINIMUM después de la instrucción COPY. Sin embargo, si se especifica INTERMEDIATE o MINIMUM antes de la instrucción COPY, se colocarán distintivos FIPS en el miembro de copia.

En OPM COBOL/400, independientemente de si se ha especificado INTERMEDIATE o MINIMUM antes o después de la instrucción COPY STATEMENT, se colocarán distintivos FIPS en el miembro de copia.

Opción NOSOURCE: En OPM COBOL/400, cuando se especifica la opción NOSOURCE en la instrucción PROCESS los valores de Options in Effect (Opciones en vigor) se imprimen en el listado del compilador.

En ILE COBOL/400, cuando se especifica la opción NOSOURCE en la instrucción PROCESS los valores de Options in Effect (Opciones en vigor) no se imprimen en el listado del compilador.

USE FOR DEBUGGING

OPM COBOL/400 acepta USE FOR DEBUGGING cuando se especifica WITH DEBUGGING MODE.

ILE COBOL/400 no da soporte a USE FOR DEBUGGING. El texto se trata como comentario hasta el inicio del siguiente apartado o el final de DECLARATIVES. Se emiten un mensaje de error de gravedad 0 y uno de gravedad 20.

Environment Division

Orden de DATA DIVISION y ENVIRONMENT DIVISION

OPM COBOL/400 es bastante flexible con respecto a entremezclar el orden de DATA DIVISION y ENVIRONMENT DIVISION. OPM COBOL/400 emite mensajes de gravedad 10 y 20 si se encuentra cláusulas, expresiones, secciones y divisiones que no estén en el orden correcto.

ILE COBOL/400 no permite que el orden de DATA DIVISION y ENVIRONMENT DIVISION se entremezcle. ILE COBOL/400 emite mensajes de gravedad 30 si se encuentra cláusulas, expresiones, secciones y divisiones que no estén en el orden correcto.

Párrafos FILE-CONTROL e I-O-CONTROL

Si se produce una cláusula duplicada en una entrada FILE-CONTROL o I-O-CONTROL y sólo se permite una cláusula de esta clase, OPM COBOL/400 utiliza la última especificada.

En la misma situación, ILE COBOL/400 utiliza la primera especificada.

Cláusula SELECT

El compilador OPM COBOL/400 acepta varias cláusulas SELECT que hagan referencia a un nombre de archivo dado, si los atributos especificados son coherentes. El algunos casos no se emite ningún mensaje de error. En otros, se emiten mensajes de gravedad 10 ó 20. En el caso de que los atributos especificados sean incoherentes, se emiten mensajes de gravedad 30.

El compilador ILE COBOL/400 emite mensajes de gravedad 30 en todos los casos en que varias cláusulas SELECT hagan referencia a un nombre de archivo dado.

Data Division

Orden de DATA DIVISION y ENVIRONMENT DIVISION

Consulte el subapartado “Orden de DATA DIVISION y ENVIRONMENT DIVISION” del apartado “Environment Division”.

Entradas FD o SD

Si se produce una cláusula duplicada en una entrada FD o SD y sólo se permite una cláusula de esta clase, OPM COBOL/400 utiliza la última especificada.

En la misma situación, ILE COBOL/400 utiliza la primera especificada.

WORKING-STORAGE SECTION

En ILE COBOL/400, la asignación de almacenamiento de elementos Working-Storage independientes no queda reflejada en el orden en que dichos elementos están declarados en la sección Working-Storage, como era el caso en OPM COBOL/400.

El impacto potencial de este cambio en la forma de asignar el almacenamiento, recae sobre aquellos programas que utilizan un esquema de circunvención para mitigar la limitación de tamaño máximo de tablas, 32 Kb, de OPM COBOL/400. Si el programa utiliza un esquema de circunvención para incrementar el tamaño de tabla allí donde se declaren de forma consecutiva varios elementos Working-Storage independientes y la comprobación de rango está desactivada, el esquema dejará de funcionar.Si un programa que utilice un esquema de esta clase se ejecuta utilizando ILE COBOL/400, el programa dará resultados imprevisibles.

Para ILE COBOL/400, el tamaño máximo de tabla es ahora de 16 711 568 y así, el problema que desencadenaba el esquema de circunvención ya no existe. Sin embargo, los programas que utilicen el esquema de circunvención tendrán que recodificarse.

Cláusula LIKE

Cuando se encuentra la cláusula REDEFINES después de una cláusula LIKE, el compilador OPM COBOL/400 emite un mensaje de gravedad 20 que indica que se ha hecho caso omiso de la cláusula REDEFINES porque aparece después de una cláusula LIKE.

En la misma situación, el compilador ILE COBOL/400 emite un mensaje de gravedad 10 cuando se encuentra la cláusula REDEFINES y acepta la cláusula REDEFINES, pero también emite un mensaje de gravedad 30 que indica que la cláusula LIKE no es compatible con la cláusula REDEFINES.

Tal situación puede darse también en el caso de otras cláusulas incompatibles tales como LIKE y USAGE o bien LIKE y PICTURE.

Cláusula LINAGE

OPM COBOL/400 señala un entero con signo LINAGE con el mensaje LBL1350, pero no emite ningún mensaje para FOOTING, TOP y BOTTOM con signo.

ILE COBOL/400 emite el mensaje LNC1350 en los 4 casos.

Cláusula PICTURE

El compilador ILE COBOL/400 no acepta la serie PICTURE .$$. Igualmente, tampoco se aceptan las series PICTURE +.$$ y -.$$.

Cuando aparecen CR o DB en las posiciones de carácter 30 y 31 de una serie de caracteres, el compilador ILE COBOL/400 no las acepta como válidas. La serie PICTURE entera debe estar contenida en 30 caracteres.

Cláusula REDEFINES

OPM COBOL/400 inicializa los elementos redefinidos.

ILE COBOL/400 no inicializa los elementos redefinidos. El valor inicial lo determina el valor por omisión del dato original.

Cláusula VALUE

En ILE COBOL/400, se truncará un literal numérico especificado en la cláusula VALUE si la longitud del mismo es superior a la de la serie PICTURE que lo define. En OPM COBOL/400, se supone el valor 0.

Procedure Division

Consideraciones generales

Datos binarios: En OPM COBOL/400, cuando los datos se tienen en forma de datos binarios en los que el valor del elemento sobrepasa el valor descrito por la cláusula PICTURE, se obtienen resultados imprevisibles.En general, cuando se utilice este elemento, es posible que se trunque al número real de dígitos descrito por la cláusula PICTURE.Normalmente, depende de si se utiliza PACKED intermedio para copiar el valor.

En ILE COBOL/400, también se obtendrán resultados imprevisibles, pero serán diferentes de los generados por OPM COBOL/400.

Alineación de datos binarios de 8 bytes: En OPM COBOL/400, los datos binarios de 8 bytes se alinean con límites de 4 bytes si se especifica la opción *SYNC en el parámetro GENOPT del mandato CRTCBLPGM.

En ILE COBOL/400, los datos binarios de 8 bytes se alinean con límites de 8 bytes si se especifica la opción *SYNC en el parámetro OPTION del mandato CRTCBLMOD o CRTBNDCBL.

Nombres de párrafo duplicados: Cuando se encuentran nombres de párrafo duplicados en un programa COBOL, el compilador OPM COBOL/400 genera un mensaje de gravedad 20.

En la misma situación, el compilador ILE COBOL/400 genera un mensaje de gravedad 30.

Número de subíndice: Cuando se especifica un número incorrecto de subíndices para un elemento (demasiados, insuficientes, ninguno para un elemento que los necesita o si se especifican para un elemento que no los necesita) se genera un mensaje de gravedad 30 en ILE COBOL/400.

En la misma situación, OPM COBOL/400 genera un mensaje de gravedad 20.

Segmentación: La segmentación no está soportada en ILE COBOL/400. Como consecuencia, no se realiza la comprobación sintáctica de los números de segmento.

Expresiones comunes

Expresión (NOT) ON EXCEPTION: La expresión (NOT) ON EXCEPTION se ha añadido a la instrucción DISPLAY. La adición de estas expresiones podría requerir la adición del delimitador de ámbito END-DISPLAY para evitar errores de compilación.

Por ejemplo:

```
 ACCEPT B AT LINE 3 COLUMN 1
ON EXCEPTION
       DISPLAY "IN ON EXCEPTION"
   NOT ON EXCEPTION
       MOVE A TO B
 END-ACCEPT.
```

Las expresiones ON EXCEPTION y NOT ON EXCEPTION están pensadas para instrucciones ACCEPT; sin embargo, si no se utiliza END-DISPLAY como ocurre a continuación, NOT ON EXCEPTION se consideraría parte de la instrucción DISPLAY.

```
  ACCEPT B AT LINE 3 COLUMN 1
ON EXCEPTION
       DISPLAY "IN ON EXCEPTION"
       END-DISPLAY
   NOT ON EXCEPTION
       MOVE A TO B
  END-ACCEPT.
```

Expresión INVALID KEY: En ILE COBOL/400, la expresión INVALID KEY no está permitida para el acceso secuencial a archivos relativos ya que el significado de la clave no válida sería indeterminado en dichas circunstancias. El compilador ILE COBOL/400 emite un mensaje de error de gravedad 30 en esta situación.

El compilador OPM COBOL/400 no emite ningún mensaje de error en esta situación.

Expresión ON SIZE ERROR: Para las operaciones aritméticas y las expresiones condicionales en ILE COBOL/400, cuando no se especifica ON SIZE ERROR y se produce un error de tamaño, los resultados son imprevisibles. Los resultados pueden ser diferentes a los que existían en OPM COBOL/400.

Para las operaciones aritméticas y las expresiones condicionales en ILE COBOL/400, cuando no se especifica ON SIZE ERROR y se produce un error de división entre cero, los resultados son imprevisibles. Los resultados pueden ser diferentes a los que existían en OPM COBOL/400.

Procedimientos DECLARATIVE

Procedimiento declarativo implementado como procedimiento ILE: En ILE COBOL/400, cada procedimiento DECLARATIVE es un procedimiento ILE. Así pues, cada procedimiento DECLARATIVE se ejecuta en su propia invocación por separado de otros procedimientos declarativos y por separado de la parte no declarativa del programa ILE COBOL/400. Como resultado, la utilización de recursos de sistema sensibles a las invocaciones tales como el envío y la recepción de mensaje, el mandato CL RCLRSC y las alteraciones temporales, será diferentes en ILE COBOL/400 de lo que es en OPM COBOL/400.

Llamada a un procedimiento declarativo desde otro procedimiento declarativo: En ILE COBOL/400, puede invocarse a un procedimiento declarativo desde otro procedimiento declarativo debido a un error de E-S siempre y cuando no se haya invocado al primero por cualquier otra razón.

OPM COBOL/400 impide que se invoque a un procedimiento declarativo desde otro procedimiento declarativo debido a un error de E-S.

Expresiones

Expresiones de condición de clase: En ILE COBOL/400, el identificador de una expresión de clase no puede se un elemento de grupo que contenga uno o más elementos elementales numéricos con signo.

Expresiones condicionales abreviadas: Para ILE COBOL/400, la utilización de paréntesis en condiciones relacionales combinadas abreviadas no es válida.OPM COBOL/400 no impone esta regla.

Comparación de constantes figurativas con constantes figurativas: En OPM COBOL/400, cuando se compara una constante figurativa con otra constante figurativa, se emite un mensaje de error de gravedad 20 y se acepta la instrucción.

En ILE COBOL/400, cuando se compara una constante figurativa con otra constante figurativa, se emite un mensaje de error de gravedad 30 y se rechaza la instrucción.

Comparación de elementos con zona con elementos no numéricos: Cuando se comparan elementos con zona con un elemento no numérico, OPM COBOL/400 emite un mensaje de gravedad 20. ILE COBOL/400 no emite ningún mensaje de esta clase.

NOT en una expresión relacional: OPM COBOL/400 acepta la expresión " A NOT NOT = B ", pero se genera un mensaje de gravedad 20.

En la misma situación, ILE COBOL/400 genera un mensaje de gravedad 30.

NOT LESS THAN OR EQUAL TO: ILE COBOL/400 permite ciertas formas de expresión condicional que no permite OPM COBOL/400. En concreto, NOT LESS THAN OR EQUAL y NOT GREATER THAN OR EQUAL.

Registros especiales

Registro especial DEBUG-ITEM: ILE COBOL/400 ya no da soporte al registro especial DEBUG-ITEM. Cuando se lo encuentra, sólo comprueba la sintaxis del mismo.

Registro especial LINAGE-COUNTER: En OPM COBOL/400, cuando aparece un entero en la cláusula LINAGE, se define LINAGE-COUNTER como elemento binario de 5 dígitos y 2 bytes.

En ILE COBOL/400, cuando aparece un entero en la cláusula LINAGE, se define LINAGE-COUNTER como elemento binario de 9 dígitos y 4 bytes.

Registro especial WHEN-COMPILED: En OPM COBOL/400, el registro especial WHEN-COMPILED sólo se puede utilizar con la instrucción MOVE.

En ILE COBOL/400, el registro especial WHEN-COMPILED se puede utilizar con cualquier instrucción.

Instrucciones ACCEPT y DISPLAY ampliadas

Consideraciones sobre compilación: OPM COBOL/400 requiere el valor EXTACCDSP en el parámetro GENOPT del mandato CRTCBLPGM a fin de habilitar las instrucciones ACCEPT y DISPLAY ampliadas. La opción EXTACCDSP no existe en los mandatos CRTCBLMOD/CRTBNDCBL para ILE COBOL/400. En ILE COBOL/400, las instrucciones ACCEPT y DISPLAY ampliadas están siempre habilitadas. Dado que ya no existe la opción EXTACCDSP en la instrucción PROCESS para ILE COBOL/400, cualquier programa OPM COBOL/400 que especifique dicha opción en la instrucción PROCESS puede comportarse de forma diferente al compilarlo con el compilador ILE COBOL/400. El compilador ILE COBOL/400 determina si se ha ampliado una instrucción ACCEPT o DISPLAY buscando CONSOLE IS CRT en el párrafo SPECIAL NAMES o buscando las expresiones encontradas en la instrucción ACCEPT de formato 7 o la instrucción DISPLAY de formato 3.

En ILE COBOL/400, las palabras siguientes son siempre palabras reservadas COBOL:

- AUTO
- BEEP
- BELL
- FULL
- BLINK
- COL
- COLUMN
- PROMPT
- UPDATE
- NO-ECHO
- REQUIRED
- AUTO-SKIP
- HIGHLIGHT
- UNDERLINE
- ZERO-FILL
- EMPTY-CHECK
- LEFT-JUSTIFY
- LENGTH-CHECK
- REVERSE-VIDEO
- RIGHT-JUSTIFY
- TRAILING-SIGN

En OPM COBOL/400, la instrucción DISPLAY sólo da soporte a las tablas fijas. En ILE COBOL/400, las instrucciones ACCEPT y DISPLAY dan soporte a cualquier tabla.

Los datos modificados de referencia están soportados en las instrucciones ACCEPT y DISPLAY para ILE COBOL/400. No están soportados en OPM COBOL/400.

En OPM COBOL/400 ha de utilizarse la opción *NOUNDSPCHR para poder utilizar el juego de caracteres ampliado además del juego de caracteres DBCS básico. En ILE COBOL/400, puede utilizar *NOUNDSPCHR o *UNDSPCHR y seguirá manejando los caracteres DBCS correctamente.

El compilador OPM COBOL/400 emite un mensaje de error de gravedad 30 cuando se encuentra un datos cuya longitud es superior a la capacidad de la pantalla.El compilador ILE COBOL/400 no emite ningún error de esta clase.

En ILE COBOL/400, se emite un mensaje de error de gravedad 30 cuando un identificador o un entero de la expresión COLUMN sobrepasa los 8 dígitos. OPM COBOL/400 no emite ningún error.

Para expresiones en las que sólo se ha comprobado la sintaxis en la instrucción DISPLAY, el compilador ILE COBOL/400 realiza una comprobación completa de la sintaxis de las expresiones PROMPT, BACKGROUND-COLOR y FOREGROUND-COLOR. Si alguna de estas expresiones está codificada de forma incorrecta, el compilador ILE COBOL/400 emite mensajes de error de gravedad 30.El compilador OPM COBOL/400 no realiza una comprobación completa de la sintaxis de las expresiones PROMPT, BACKGROUND-COLOR y FOREGROUND-COLOR ni emite ningún mensaje de error de compilación.

Consideraciones sobre ejecución: En OPM COBOL/400, se inhabilita PRINT KEY durante el funcionamiento de ACCEPT ampliada. En ILE COBOL/400, PRINT KEY estará siempre habilitada de forma incondicional.

En OPM COBOL/400, la expresión SIZE sólo está soportada en la instrucción DISPLAY.En ILE COBOL/400, la expresión SIZE está soportada en las instrucciones ACCEPT y DISPLAY. Cuando el tamaño especificado es superior al tamaño establecido implícitamente por la longitud de datos de la cláusula PICTURE, OPM COBOL/400 rellena con blancos por la izquierda los datos alfanuméricos cuando se justifican.ILE COBOL/400 los rellena siempre con blancos por la derecha.

En OPM COBOL/400, se emite el mensaje de error LBE7208 cuando el dato no cabe dentro de la pantalla. En ILE COBOL/400, los datos alfanuméricos que no caben en la pantalla se truncan y los datos numéricos que no caben en la pantalla no se visualizan. No se emite ningún error de ejecución.

Cuando se utilizan las teclas HELP y CLEAR para finalizar la operación ACCEPT en una estación de trabajo conectada a un controlador remoto 3174 o 3274, ILE COBOL/400 emitirá un error de ejecución. OPM COBOL/400 finalizará satisfactoriamente esta operación ACCEPT sin emitir ningún mensaje de error.

OPM COBOL/400 siempre actualiza todos los campos manejados por la instrucción ACCEPT. ILE COBOL/400 sólo actualiza los campos que ha modificado el usuario antes de pulsar la tecla Intro para una instrucción ACCEPT. Como resultado, los dos compiladores se comportan de manera diferente en tres situaciones:

- cuando se especifica la expresión SECURE en la instrucción ACCEPT y no se entra ningún valor.
- cuando está en vigor la opción ACCUPDNE y la instrucción ACCEPT maneja datos editados no numéricos.
- cuando el campo se previsualiza con datos alfanuméricos y la expresión RIGHT-JUSTIFIED está especificada en la instrucción ACCEPT.

Instrucción CALL

Caracteres en minúsculas en el literal o identificador CALL/CANCEL: OPM COBOL/400 permite que el literal o identificador CALL/CANCEL contengan caracteres en minúsculas; sin embargo un nombre de objeto de programa que no sea un nombre del sistema entre comillas (nombre ampliado) no permite los caracteres en minúsculas y así, la operación CALL/CANCEL resultante fallará.

ILE COBOL/400 da soporte a dos valores nuevos en el parámetro OPTION de los mandatos CRTCBLMOD y CRTBNDCBL: *MONOPRC y *NOMONOPRC. El valor por omisión, *MONOPRC, hace que las minúsculas del literal o identificador CALL/CANCEL se conviertan en mayúsculas. El valor *NOMONOPRC especifica que el literal o identificador CALL/CANCEL no se ha de convertir a mayúsculas.

Traspaso de un nombre-archivo en la expresión USING: Tanto OPM COBOL/400 como ILE COBOL/400 permiten que se pase un nombre-archivo en la expresión USING de la llamada CALL; sin embargo, OPM COBOL/400 pasa un puntero al FIB (bloque de información de archivo) mientras que ILE COBOL/400 pasa un puntero a un puntero NULL.

Llamadas recursivas: ILE COBOL/400 *no* permite que se llame a los programas de forma recursiva. ILE COBOL/400 impide estrictamente la recursividad dentro de un grupo de activación. ILE COBOL/400 genera un mensaje de error de ejecución cuando se detecta la recursividad.

OPM COBOL/400 no impide la recursividad. Sin embargo, si se intenta la recursividad con OPM COBOL/400 los resultados pueden ser imprevisibles.

Instrucción CANCEL

En ILE COBOL/400, la instrucción CANCEL sólo cancelará programas ILE COBOL/400 dentro del mismo grupo de activación. En ILE COBOL/400, se mantiene, a nivel de grupo de activación (unidad de ejecución) una lista de los objetos de programa llamados.Si el programa a cancelar no aparece en la lista, se hace caso omiso de la cancelación.

En OPM COBOL/400, la instrucción CANCEL emitirá un mensaje de error si el programa a cancelar no existe en la lista de bibliotecas.

Instrucción COMPUTE

En algunos casos, el resultado de elevar a una potencia en ILE COBOL/400 puede ser ligeramente diferente al resultado de elevar a una potencia en OPM COBOL/400.

Cuando se realiza una instrucción COMPUTE de una expresión de elevar a una potencia con un valor negativo para la mantisa y un valor fraccional negativo para el exponente, OPM COBOL/400 da resultados indefinidos. En la misma situación, ILE COBOL/400 genera una excepción CEE2020.

Instrucción DELETE

En OPM COBOL/400, el estado de archivo se establece como 90 cuando en la instrucción DELETE se utiliza un formato de registro no válido para un archivo.

En ILE COBOL/400, el estado de archivo se establece como 9K cuando en la instrucción DELETE se utiliza un formato de registro no válido para un archivo.

Instrucción EVALUATE

En OPM COBOL/400, cuando se especifica la expresión WHEN con ZERO THRU *identificador-alfabético*, se permite la instrucción y no se emite ningún mensaje de diagnóstico.

En la misma situación, ILE COBOL/400 emite un mensaje de error de gravedad 30.

Nota: ILE COBOL/400 ha flexibilizado esta regla en el caso de *identificador-alfanumérico* THRU *identificador-alfabético* desde que el identificador-alfanumérico sólo puede contener caracteres alfabéticos.

Instrucción IF

El límite de anidación en OPM COBOL/400 es de 30 niveles para las instrucciones IF.

ILE COBOL/400 carece de límite práctico con respecto a la anidación de las instrucciones IF.

Instrucción INSPECT

ILE COBOL/400 da soporte a la modificación de referencia en la instrucción INSPECT.

OPM COBOL/400 no incluye este soporte.

Instrucción MOVE

Literales alfanuméricos y nombres de índice: Cuando se mueve un literal alfanumérico a un nombre de índice, OPM COBOL/400 emite un mensaje de error de gravedad 20. En la misma situación, ILE COBOL/400 emite un mensaje de error de gravedad 30.

Valores alfanuméricos y literales editados numéricos: Cuando se mueve un valor alfanumérico que contiene sólo caracteres numéricos a un literal editado numérico (por ejemplo, `MOVE "12.34" TO NUMEDIT`), OPM COBOL/400 toma por omisión el valor 0 para el literal. En la misma situación, ILE COBOL/400 emite un mensaje de error de gravedad 30.

Valores booleanos: OPM COBOL/400 permite que se mueva un valor booleano a un identificador alfabético cuya referencia se ha modificado. ILE COBOL/400 no lo permite y emite un mensaje de error de gravedad 30.

Expresión CORRESPONDING: Las instrucciones MOVE, ADD y SUBTRACT CORRESPONDING de ILE COBOL/400 utilizan un algoritmo distinto al de OPM COBOL/400 para determinar qué elementos son los que corresponden. ILE COBOL/400 podría generar un mensaje de error de gravedad 30 mientras que en OPM COBOL/400 no se emitiría ningún mensaje.

```
    01 A.
       05  B.
         10  C PIC X(5).
       05  C PIC X(5).
    01 D.
       05  B.
         10  C PIC X(5).
       05  C PIC X(5).
  MOVE CORRESPONDING A TO D.
```

OPM COBOL/400 no emite ningún mensaje; ILE COBOL/400 emitirá el mensaje LNC1463.

Series origen y destino que se solapan: Si la serie origen y destino de una instrucción MOVE se solapan, el resultado es imprevisible. La operación puede no comportarse como lo haría en OPM COBOL/400 en la misma situación.

Instrucción OPEN

Creación dinámica de archivos: Hay dos cuestiones de compatibilidad con respecto a la creación dinámica de archivos.

- El compilador OPM COBOL/400 daba soporte a la creación dinámica de archivos indexados.

  El compilador ILE COBOL/400 *no* da este soporte.

- Un archivo se creará dinámicamente sólo si está asignado al tipo de dispositivo COBOL DISK.

  OPM COBOL/400 crea archivos (de base de datos) que se asignan a tipos de dispositivo COBOL distintos de DISK si existe una alteración temporal a un archivo de base de datos (OVRDBF).

Apertura de FORMATFILE: En ILE COBOL/400, FORMATFILE sólo puede abrirse para OUTPUT. Se puede utilizar la instrucción WRITE para escribir registros de salida en FORMATFILE.

En OPM COBOL/400, FORMATFILE se puede abrir para INPUT, I-O y OUTPUT.

OPEN OUTPUT u OPEN I-O para archivos OPTIONAL: En ILE COBOL/400, OPEN OUTPUT u OPEN I-O para archivos OPTIONAL no creará el archivo, si éste no existe, cuando la organización de archivos sea INDEXED.

En OPM COBOL/400, se crea el archivo.

Instrucción PERFORM

En ILE COBOL/400, dentro de la expresión VARYING...AFTER de la instrucción PERFORM, se aumenta *identificador-2* antes de establecer *identificador-5*. En OPM COBOL/400, se establece *identificador-5* antes de aumentar *identificador-2*.

Los resultados de la instrucción PERFORM de formato 4 con la expresión AFTER es diferente en ILE COBOL/400 comparado con OPM COBOL/400. Considere el ejemplo siguiente:

```
PERFORM PARAGRAPH-NAME-1
        VARYING X FROM 1 BY 1 UNTIL X > 3
                AFTER Y FROM X BY 1 UNTIL Y > 3.
```

En OPM COBOL/400, *PARAGRAPH-NAME-1* se realiza con los valores (X,Y) de (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,2), (3,3).

En ILE COBOL/400, *PARAGRAPH-NAME-1* se realiza con los valores (X,Y) de (1,1), (1,2), (1,3), (2,2), (2,3), (3,3).

Instrucción READ

AT END no permitido para lecturas aleatorias de archivos relativos: En ILE COBOL/400, la expresión AT END no está permitida para lecturas aleatorias de archivos relativos ya que el significado de la lectura aleatoria sería indeterminado en dichas circunstancias. El compilador ILE COBOL/400 emite un mensaje de error de gravedad 30 en esta situación.

El compilador OPM COBOL/400 no emite ningún mensaje de error en esta situación.

Mensajes de error: En ILE COBOL/400, se emite el mensaje de error LNC1408, y no LNC0651, para la expresión FORMAT cuando se va a realizar una instrucción READ sobre un archivo FORMATFILE.

Se emite el mensaje de error LNC1408 cuando el dispositivo a leer es distinto de DATABASE. Se emite el mensaje de error LNC0651 cuando el dispositivo es DATABASE, pero ORGANIZATION no es indexada.

Instrucción REWRITE

En OPM COBOL/400, el estado de archivo se establece como 90 cuando en la instrucción REWRITE se utiliza un formato de registro no válido para un archivo.

En ILE COBOL/400, el estado de archivo se establece como 9K cuando en la instrucción REWRITE se utiliza un formato de registro no válido para un archivo.

Instrucción SET

Cuando se establece un nombre-condición como TRUE y la variable de condición asociada es un elemento editado, OPM COBOL/400 edita el valor del nombre-condición cuando se mueve a la variable de condición.

ILE COBOL/400 no realiza ninguna edición cuando se mueve el valor del nombre-condición a la variable de condición.

Instrucciones SORT/MERGE

Expresión GIVING y las cláusulas SAME AREA/SAME RECORD AREA: En ILE COBOL/400, los nombre-archivo asociados con la expresión GIVING no pueden especificarse en la misma cláusula SAME AREA o SAME RECORD AREA. El compilador ILE COBOL/400 emite un mensaje de error de gravedad 30 cuando se produce tal situación.

El compilador OPM COBOL/400 no emite ningún mensaje en esta situación.

Instrucción STOP RUN

Cuando se emite STOP RUN en un grupo de activación ILE, se producirá un COMMIT implícito, caso que no ocurre en OPM COBOL/400.

Nota: Si se emite STOP RUN en el grupo de activación por omisión del trabajo (*DFTACTGRP), no se produce ningún COMMIT implícito.

Instrucciones STRING/UNSTRING

En OPM COBOL/400, se utiliza PROGRAM COLLATING SEQUENCE para determinar el valor de verdad de las condiciones relacionales implícitas de las operaciones STRING/UNSTRING.

En ILE COBOL/400, se hace caso omiso de PROGRAM COLLATING SEQUENCE al determinar el valor de verdad de las condiciones relacionales implícitas de las operaciones STRING/UNSTRING.

Interfaces de Programación de Aplicaciones (API)

API de enlace ILE COBOL/400

ILE COBOL/400 utiliza cuatro API de enlace nuevas en lugar de las rutinas de ejecución OPM.

- La API de enlace ILE QlnRtvCobolErrorHandler sustituye a QLRRTVCE.
- La API de enlace ILE QlnSetCobolErrorHandler sustituye a QLRSETCE.
- La API de enlace ILE QlnDumpCobol sustituye a QLREXHAN para generar un vuelco con formato.
- QLRCHGCM no está soportado en ILE COBOL/400. Utilice los grupos de activación ILE mencionados para obtener varias unidades de ejecución.

Llamada a las API OPM COBOL/400

Las API OPM COBOL/400 pueden llamarse desde ILE COBOL/400, pero sólo afectarán a las unidades de ejecución OPM COBOL/400.

Para afectar a las unidades de ejecución ILE COBOL/400, utilice las API ILE correspondientes o el parámetro ACTGRP del mandato CRTPGM.

Ejecución

Conservación de la semántica de las unidades de ejecución compatibles con OPM

Puede conservar al máximo la semántica de las unidades de ejecución compatibles con OPM en:

- una aplicación que conste sólo de programas ILE COBOL/400 o
- una aplicación que mezcle programas OPM COBOL/400 y programas ILE COBOL/400

Conservación de la semántica de las unidades de ejecución compatibles con OPM en una aplicación ILE COBOL/400***:*** Para conservar la semántica de las unidades de ejecución compatibles con OPM en una aplicación ILE COBOL/400, deben cumplirse las condiciones siguientes:

- Todas las unidades participantes (ILE COBOL/400 u otros programas/procedimientos ILE) deben ejecutarse en un solo grupo de activación ILE.

  Nota: Utilizando un grupo de activación ILE con nombre para todos los programas participantes, no es necesario que especifique un programa ILE COBOL/400 concreto para que sea el programa principal antes de la ejecución. Por otro lado, si se sabe que un programa ILE COBOL/400 concreto es el programa principal antes de la ejecución, puede especificar el atributo *NEW para la opción ACTGRP al crear un objeto *PGM

utilizando el programa ILE COBOL/400 como UEP. Todos los demás programas participantes deben especificar el atributo *CALLER para la opción ACTGRP.

- La invocación más antigua del grupo de activación ILE debe ser la de ILE COBOL/400. Este es el programa principal de la unidad de ejecución.

Si no se cumplen estas condiciones, es posible que un STOP RUN explícito o implícito de un grupo de activación ILE no finalice el grupo de activación. Con el grupo de activación todavía activo, los diversos programas ILE COBOL/400 estarán en el último estado utilizado.

Nota: La condición anterior dicta que un programa ILE COBOL/400 en ejecución en *DFTACTGRP se ejecute generalmente en una unidad de ejecución que no es compatible con OPM. A los programas ILE COBOL/400 en ejecución en *DFTACTGRP no se les reclamará físicamente su almacenamiento estático hasta que finalice el trabajo. Un programa ILE COBOL/400, con *CALLER especificado para el parámetro ACTGRP del mandato CRTPGM, se ejecutará en *DFTACTGRP si lo llama un programa OPM.

Conservación de la semántica de las unidades de ejecución compatibles con OPM en una aplicación mixta OPM COBOL/400 y ILE COBOL/400: A fin de mezclar programas OPM COBOL/400 con programas ILE COBOL/400 y conservar la semántica de las unidades de ejecución compatibles lo máximo posible, es necesario que se cumplan las condiciones siguientes:

- La invocación del programa OPM COBOL/400 (no la del ILE COBOL/400) debe ser la primera invocación COBOL
- STOP RUN lo emite un programa OPM COBOL/400
- Todos los programas participantes en la unidad de ejecución (OPM COBOL/400) deben ejecutarse en el grupo de activación *DFTACTGRP.

Si no se cumplen las condiciones anteriores, no se conservará la semántica de las unidades de ejecución compatibles con OPM para la aplicación mixta OPM /ILE. Por ejemplo, si se ejecuta un programa ILE COBOL/400 en *DFTACTGRP y emite un STOP RUN, tanto los programas OPM COBOL/400 como los ILE COBOL/400 se dejarán en el último estado utilizado.

En ILE COBOL/400, el flujo de control de operaciones, CALL, CANCEL, EXIT PROGRAM, STOP RUN y GOBACK, hará que la unidad de ejecución se comporte de manera diferente a menos que se utilice una unidad de ejecución compatible con OPM.

Mensajes de error

En ILE COBOL/400, los mensajes de error de ejecución llevan el prefijo LNR. Además, algunos números de mensaje no siempre son los mismos en OPM COBOL/400.

En ILE COBOL/400, cuando la unidad de ejecución termina anormalmente, se devuelve el mensaje CEE9901 al emisor de la llamada. En OPM COBOL/400, se devuelve el mensaje LBE9001 al emisor de la llamada en las mismas circunstancias.

Debido a las diferencias que hay entre el manejo de excepciones ILE y el de OPM, es posible que reciba más excepciones en una instrucción ILE COBOL/400 en comparación con una instrucción OPM COBOL/400.

Estado de archivo 9A cambiado por 0A

En OPM COBOL/400, el estado de archivo se establece como 9A cuando un trabajo finaliza de una manera controlada.

En ILE COBOL/400, el estado de archivo se establece como 0A cuando un trabajo finaliza de una manera controlada.

Estado de archivo 9M cambiado por 0M

En OPM COBOL/400, el estado de archivo se establece como 9M cuando se escribe el último registro en un subarchivo.

En ILE COBOL/400, el estado de archivo se establece como 0M cuando se escribe el último registro en un subarchivo.

Apéndice G. Glosario de abreviaturas

Abreviatura	Significado	Explicación
AG	Grupo de activación	Partición de los recursos dentro de un trabajo. Un grupo de activación consta de recursos del sistema (almacenamiento para variables de procedimiento o programa, definiciones de compromiso y archivos abiertos) asignados a uno o más programas.
API	Interfaz de programación de aplicaciones	Interfaz funcional proporcionada por el sistema operativo o por un programa bajo licencia que puede pedirse por separado y que permite que un programa de aplicación escrito en un lenguaje de alto nivel utilice datos o funciones específicos del sistema operativo o del programa bajo licencia.
ANSI	Instituto Americano de Normas	Organización formada por fabricantes, consumidores y grupos de interés general que establece los procedimientos con los que las organizaciones acreditadas crean y mantienen normas industriales voluntarias en los Estados Unidos.
ASCII	Código estándar americano para el intercambio de información	Código desarrollado por el Instituto Americano de Normas para el intercambio de información entre sistemas de proceso de datos, sistemas de comunicaciones de datos y el equipo asociado. El juego de caracteres ASCII consta de caracteres de 8 bits formados por caracteres simbólicos y caracteres de control de 7 bits más un bit de comprobación de paridad.
CICS	Servicio de control de información del cliente	Programa bajo licencia de IBM que permite a las transacciones entradas en estaciones de trabajo remotas sean procesadas de forma concurrente por programas de aplicación escritos por el usuario. El programa bajo licencia incluye funciones para construir, utilizar y mantener bases de datos y para comunicarse con CICS en otros sistemas operativos.
CL	Lenguaje de control	Conjunto de todos los mandatos con los que el usuario solicita funciones del sistema.

Abreviatura	Significado	Explicación
DBCS	Juego de caracteres de doble byte	Juego de caracteres en el que cada carácter está representado por 2 bytes.Idiomas como el japonés, el chino y el coreano, que contienen más símbolos de los que pueden representarse con 256 puntos de código, requieren juegos de caracteres de doble byte. Dado que cada carácter requiere 2 bytes, el tecleo, la visualización y la impresión de los caracteres DBCS requiere un hardware y unos programas que den soporte a DBCS. Hay cuatro juegos de caracteres de doble byte a los que da soporte el sistema: japonés, coreano, chino simplificado y chino tradicional.Compárese con juego de caracteres de un solo byte.
DDM	Gestión de Datos Distribuidos	Función del sistema operativo que permite que un programa de aplicación o que el usuario de un sistema utilicen los archivos de datos almacenados en sistemas remotos.Los sistemas deben estar conectados por una red de comunicaciones y los sistemas remotos deben utilizar también DDM.
DDS	Especificaciones de descripción de datos	Descripción de los archivos de dispositivo o de la base de datos del usuario que se entra en el sistema con un formato fijo. La descripción se utiliza para crear archivos.
EBCDIC	Código binario ampliado para intercambio de información decimal	Juego de caracteres codificado que consta de 256 caracteres de ocho bits.
EPM	Modelo de programa ampliado	Conjunto de funciones para compilar el código fuente y crear programas en los lenguajes de alto nivel del sistema AS/400 que definen llamadas a procedimientos.
FIPS	Norma de proceso de información federal	Norma oficial para mejorar la utilización y la gestión de los sistemas informáticos y el proceso de datos en las actividades de negocios.
HLL	Lenguaje de alto nivel	Lenguaje de programación cuyos conceptos y estructuras resultan convenientes para el raciocinio humano; por ejemplo C, COBOL y RPG. Los lenguajes de alto nivel son independientes de las estructuras de los sistemas informáticos y de las estructuras de funcionamiento.
ICF	Función de comunicaciones entre sistemas	Función del sistema operativo que permite a un programa comunicarse de forma interactiva con otro programa o sistema.
ILE	Entorno de Lenguajes Integrado	Conjunto de estructuras e interfaces que proporciona un entorno de ejecución común e interfaces de programación de aplicaciones (API) vinculables en la ejecución para los lenguajes de alto nivel que acaten ILE.

Abreviatura	Significado	Explicación
E/S	Entrada/salida	Datos procedentes del sistema informático o resultantes del proceso informático.
LVLCHK	Comprobación de nivel	Función que compara los identificadores a nivel de formato de registro, de un archivo que ha de abrirse con la descripción de archivo que forma parte de un programa compilado, para determinar si el formato de registro del archivo ha cambiado desde que se compiló el programa.
ODP	Vía de datos abierta	Bloque de control creado al abrir un archivo.Una ODP contiene información sobre los atributos de archivo fusionado y la información devuelta por las operaciones de entrada y salida. La ODP sólo existe mientras está abierto el archivo.
ODT	Tabla de definición de objetos	Tabla construida por el sistema durante la compilación para hacer un seguimiento de los objetos declarados en el programa.Los objetos de programa de la tabla incluyen variables, constantes, etiquetas, listas de operandos y descripciones de excepción. La tabla reside en el objeto de programa compilado.
OPM	Modelo de programa original	Conjunto de funciones para compilar el código fuente y crear programas en lenguajes de alto nivel en el sistema AS/400, utilizado antes de que se introdujese el modelo Entorno de Lenguajes Integrado (ILE).
OS/400	Operating System/400	El sistema operativo AS/400
PEP	Procedimiento de entrada de programa	Procedimiento proporcionado por el compilador que es el punto de entrada de un programa ILE en una llamada dinámica de programa. Compárese con *procedimiento de entrada de usuario*.
SDA	Ayuda para el diseño de pantallas	Función del programa bajo licencia Juego de Herramientas para el Desarrollo de Aplicaciones/400 que ayuda al usuario a diseñar, crear y mantener pantallas y menús.
SEU	Programa de Utilidad para Entrada del Fuente	Función del programa bajo licencia Juego de Herramientas para el Desarrollo de Aplicaciones/400 que se utiliza para crear y cambiar miembros fuente.
SQL/400	Lenguaje de Consulta Estructurada (SQL)/400	Programa bajo licencia de IBM que da soporte a la base de datos relacional que se utiliza para poner información en una base de datos así como para obtener y organizar información seleccionada de una base de datos.

Abreviatura	Significado	Explicación
UEP	Procedimiento de entrada de usuario	Procedimiento de entrada escrito por un programador de aplicaciones y que es el destino de la llamada dinámica de programa. A este procedimiento lo llama el procedimiento de entrada de programa (PEP). Compárese con *procedimiento de entrada de programa*.
UPSI	Conmutador de indicador de estado del programa	Conmutador de programa externo que realiza las funciones de un conmutador de hardware.Se proporcionan ocho conmutadores: UPSI 0 - 7.

Nota: En esta relación no figuran las abreviaturas de los mandatos OS/400. Consulte la publicación *CL Reference* para saber cuáles son los mandatos OS/400 y su utilización.

Bibliografía

Para obtener información adicional sobre los temas relacionados con la programación en ILE COBOL/400 en el sistema AS/400, consulte las publicaciones de IBM AS/400 siguientes:

- *ADTS/400: Gestor para el desarrollo de aplicaciones. ADM/400 Guía de usuario*, SC10-9609 (SC09-2133), describe la forma de crear y gestionar proyectos definidos para la característica Gestor para el Desarrollo de Aplicaciones/400 además de la utilización del programa para desarrollar aplicaciones.
- *ADTS/400: Gestor de Desarrollo de Programas (PDM)*, SC10-9419 (SC09-1771), facilita información sobre la utilización del gestor para desarrollo de programación del Juego de Herramientas para el Desarrollo de Aplicaciones (ADTS)/400 para trabajar con listas de bibliotecas, objetos, miembros y opciones definidas por el usuario a fin de realizar con facilidad operaciones tales como copiar, suprimir y redenominar. Contiene actividades y material de referencia para ayudar al usuario a aprender el manejo de PDM.Las teclas de función y operaciones utilizadas con mayor frecuencia están explicadas con detalle por medio de ejemplos.
- *ADTS/400: Programa de Utilidad para Entrada del Fuente (SEU)*, SC10-9422 (SC09-1774), facilita información sobre la utilización del Programa de Utilidad para Entrada del Fuente (SEU) del Juego de Herramientas para el Desarrollo de Aplicaciones (ADTS)/400 para crear y editar miembros fuente. En el manual se explica la forma de iniciar y finalizar una sesión de SEU y la manera de utilizar las diversas características de este editor de texto de pantalla completa.Este manual contiene ejemplos que sirven de ayuda tanto a usuarios nuevos como a los que ya tienen experiencia para efectuar diversas tareas de edición, desde los mandatos de línea más simples a la utilización de solicitudes predefinidas para formatos de datos y lenguajes de alto nivel.
- *Application Display Programming*, SC41-4715, facilita información sobre:
  - La utilización de las DDS para crear y mantener pantallas para aplicaciones;
  - La manera de crear y trabajar con archivos de pantalla en el sistema;
  - La creación de información de ayuda en línea;
  - La utilización de UIM para definir paneles y diálogos para un aplicación;
  - La utilización de documentos, registros o grupos de paneles
- *Backup and Recovery – Advanced*, SC41-4305, facilita información sobre la configuración y la gestión de lo siguiente:
  - El registro por diario, la protección de vías de acceso y el control de compromiso
  - Las agrupaciones de almacenamiento auxiliar (ASP) de usuario
  - La protección de disco (paridad de dispositivo, duplicación y suma de comprobación)

  Facilita información sobre los medios de copia de seguridad y las operaciones salvar/restaurar.También incluye temas avanzados sobre copia de seguridad y recuperación, como por ejemplo la utilización del soporte salvar mientras está activo, salvar y restaurar a un release diferente, y consejos y técnicas de programación.
- *CICS/400 Application Programming Guide*, SC33-1386, facilita información sobre la programación de aplicaciones para CICS/400. Incluye información de guía y referencia sobre los mandatos de interfaz de programación del sistema y la interfaz de programación de aplicaciones de CICS además de facilitar información sobre el desarrollo de nuevas aplicaciones y la migración de aplicaciones existentes desde otras plataformas CICS.
- *CL Programación*, SC10-9637 (SC41-4721), contiene una amplia exposición de los temas de programación en AS/400, incluida una exposición general sobre objetos y bibliotecas, programación en CL, control de flujo y comunicaciones entre programas, la forma de trabajar con objetos en programas CL y la creación de programas CL. Otros temas tratados son los mensajes predefinidos y improvisados, el manejo de mensajes, la definición y creación de menús y mandatos definidos por el usuario y por último, la forma de probar aplicaciones, incluidos la modalidad de depuración, los puntos de interrupción, los rastreos y las funciones de visualización.
- *CL Reference*, SC41-4722, da una descripción del lenguaje de control (CL) de AS/400 y de sus mandatos OS/400 (los mandatos que no son del OS/400 están descritos en las publicaciones de los respectivos programas bajo licencia). También proporciona una visión general de *todos* los mandatos CL del sistema AS/400 y describe las reglas sintácticas precisas para codificarlos.
- *Gestión de Comunicaciones*, SC10-9451 (SC41-3406), facilita información sobre la gestión del trabajo en un entorno de comunicaciones, el estado de las comunicaciones, el rastreo y diagnóstico de los problemas de comunicaciones, el manejo y recuperación de errores, el rendimiento, e

información específica sobre la velocidad de línea y el almacenamiento de subsistemas.

- *Gestión de datos*, SC10-9635 (SC41-4710), facilita información sobre la utilización de los archivos en los programas de aplicación. Incluye información sobre los temas siguientes:
  - Estructura y conceptos fundamentales del soporte a la gestión de datos en el sistema
  - Alteraciones temporales y redirección de archivos (realizar cambios temporales en los archivos cuando se ejecuta el programa de aplicación)
  - Copia de archivos utilizando mandatos del sistema para copiar datos de un lugar a otro.
  - Adaptción de un sistema con datos de doble byte
- *DB2/400 Programación de la base de datos*, SC10-9634 (SC41-4701), contiene una exposición detallada de la organización de base de datos de AS/400 incluida información sobre la forma de crear, describir y actualizar archivos de base de datos en el sistema. También describe la manera de definir archivos al sistema con las palabras clave de especificaciones de descripción de datos (DDS) de OS/400.
- *DB/2 for OS/400 SQL Programming*, SC41-4611, facilita información sobre la manera de utilizar el programa bajo licencia Gestor de Consultas y Kit de Desarrollo SQL de DB2/400. Muestra la forma de acceder a los datos de una biblioteca de bases de datos y de preparar, ejecutar y probar un programa de aplicación que contenga sentencias SQL hospedadas. Contiene ejemplos de sentencias SQL/400 y una descripción de la función SQL interactivo. Describe los concepto comunes y las reglas para utilizar sentencias SQL/400 en COBOL/400, ILE COBOL/400, PL/I, ILE C/400, FORTRAN/400, RPG/400, ILE RPG/400 y REXX.
- *DB2/400 Manual de Consulta SQL*, SC10-9632 (SC41-4612), facilita información sobre la manera de utilizar las sentencias DB2/400 del Lenguaje de Consulta Estructurada (SQL)/400 y proporciona detalles sobre la utilización correcta de las sentencias. Los ejemplos de sentencias incluyen diagramas de sintáxis, parámetros y definiciones. También se proporciona una lista de los límites de SQL y una descripción del área de comunicaciones SQL (SQLCA) y del área de descriptor SQL (SQLDA).
- *DDS Reference*, SC41-4712, facilita descripciones detalladas para codificar las especificaciones de descripción de datos (DDS) para archivos que pueden describirse externamente.Estos archivos son físicos, lógicos, de pantalla, de impresión y de función de comunicación intersistemas (ICF).
- *Distributed Data Management*, SC41-4307, facilita información sobre el proceso de archivos remotos. Describe la forma de definir un archivo remoto a la Gestión de Datos Distribuidos (DDM) de OS/400, la forma de crear un archivo DDM, qué programas de utilidad de archivo está soportados por medio de DDM y cuáles son los requisitos de DDM de OS/400 en relación con otros sistemas.
- *Experience RPG IV Tutorial* es un programa autodidáctico interactivo que explica las diferencias entre RPG III y RPG IV y la forma de trabajar con el nuevo entorno ILE. El libro de ejercicios que lo acompaña contiene ejercicios adicionales como referencia, una vez realizada por completo la guía de aprendizaje. Los ejemplos de código de ILE RPG/400 se adjuntan con la guía de aprendizaje y se ejecutan directamente en AS/400. Si desea pedir la guía de aprendizaje, llame al 1-800-IBM-CALL.
- *GDDM Programming*, SC41-3717, facilita información sobre la utilización del gestor de visualización de datos gráficos (GDDM) de OS/400 para escribir programas gráficos de aplicación. Incluye muchos ejemplos de programas e información para enseñar a los usuarios cómo encaja el producto en los sistemas de proceso de datos.
- *GDDM Reference*, SC41-3718, facilita información sobre la utilización del gestor de visualización de datos gráficos (GDDM) de OS/400 para escribir programas de aplicación gráficos. Este manual proporciona descripciones detalladas de todas las rutinas gráficas disponibles en GDDM. También facilita información sobre las interfaces de lenguaje de alto nivel en GDDM.
- *ICF Programming*, SC41-3442, facilita la información necesaria para escribir programas de aplicación que utilicen las comunicaciones AS/400 y la función de comunicaciones intersistemas de OS/400 (OS/400-ICF). También contiene información sobre las palabras clave de especificaciones de descripción de datos (DDS), formatos proporcionados por el sistema, códigos de retorno, soporte de transferencia de archivos y ejemplos de programa.
- *IDDU Use*, SC41-3704, describe la forma de utilizar el programa de utilidad de definición interactiva de datos (IDDU) de AS/400 para describir diccionarios de datos, archivos y registros al sistema. Incluye:
  - Una introducción a los conceptos de archivo del sistema y definición de datos
  - Una introducción a la utilización de IDDU para describir los datos utilizados en las consultas y los documentos
  - Tareas representativas relacionadas con la creación, el mantenimiento y la utilización de diccionarios de datos, archivos, formatos de registro y campos

  - Información avanzada sobre la utilización de IDDU para trabajar con archivos creados en otros sistemas e información sobre la recuperación de errores y la prevención de problemas.
- *ILE C/400 Programmer's Guide*, SC09-2069, facilita información sobre la manera de desarrollar aplicaciones con el lenguaje ILE C/400. Incluye información sobre la creación, ejecución y depuración de programas.También incluye consideraciones sobre programación para archivos de dispositivo y descritos externamente, bases de datos, manejo de excepciones, escenarios, llamadas a programas y procedimientos interlenguaje.También se dan algunos consejos sobre rendimiento. Uno de los apéndices incluye información sobre la migración de código fuente desde EPM C/400 o System C/400 a ILE C/400.
- *ILE C/400 Programmer's Reference*, SC09-2070, facilita información sobre la manera de escribir programas que se adhieran a la definición de nivel 2 de C de la Arquitectura para Aplicaciones de Sistemas y utilizar funciones específicas de ILE C/400 tales como E/S de registro.También facilita información sobre las funciones de biblioteca de interfaz de máquina ILE C/400.
- *ILE C/400 Reference Summary*, SX09-1304, facilita información de consulta rápida sobre la sintaxis de los mandatos ILE C/400, los elementos de C, las funciones de biblioteca de SAA C, las ampliaciones de biblioteca ILE C/400 a SAA C y las ampliaciones de biblioteca de interfaz de máquina (MI) ILE C/400.
- *ILE COBOL/400 Reference*, SC09-2073, da una descripción del lenguaje de programación ILE COBOL/400. Facilita información sobre la estructura del lenguaje de programación ILE COBOL/400 y la estructura de un programa fuente ILE COBOL/400. También proporciona una descripción de todos los párrafos de Identification Division, de las cláusulas de Environment Division, de las cláusulas de Data Division, de las instrucciones de Procedure Division y de las instrucciones que dirigen al compilador.
- *ILE COBOL/400 Reference Summary*, SX09-1305, facilita información de consulta rápida sobre la estructura del lenguaje de programación ILE COBOL/400 y la estructura de un programa fuente ILE COBOL/400.También proporciona los diagramas de sintaxis de todos los párrafos de Identification Division, de las cláusulas de Environment Division, de las cláusulas de Data Division, de las instrucciones de Procedure Division y de las instrucciones que dirigen al compilador.
- *ILE Conceptos*, SC10-9631 (SC41-4606), explica los conceptos y la terminología perteneciente al Entorno de Lenguajes Integrado (ILE) de la arquitectura del programa bajo licencia OS/400. Los temas tratados incluyen la creación de móulos, la vinculación, la ejecución de programas, la depuración de programas y el manejo de excepciones.
- *ILE RPG/400 Guía del programador*, SC10-9659 (SC09-2074), facilita información sobre el lenguaje de programación ILE RPG/400, que es una implementación del lenguaje RPG IV en el Entorno de Lenguajes Integrado (ILE) del sistema AS/400. Incluye información sobre la creación y ejecución de programas, con consideraciones para la programación interlenguaje y de llamadas a procedimientos. La guía trata también la depuración y el manejo de errores y explica la forma de utilizar los dispositivos y archivos AS/400 en programas RPG. Los apéndices incluyen información sobre la migración a RPG IV y ejemplos de listados del compilador. Está pensado para personas con conocimientos básicos de los conceptos del proceso de datos y del lenguaje RPG.
- *ILE RPG/400 Reference*, SC09-2077, facilita información sobre el lenguaje de programación ILE RPG/400. Este manual describe, posición por posición y palabra clave por palabra clave, las entradas válidas de todas las especificaciones RPG IV y proporciona una descripción detallada de todos los códigos de operación y de las funciones incorporadas.También contiene información sobre los indicadores, funciones de edición, tablas y matrices, y ciclo de la lógica RPG.
- *ILE RPG Reference Summary*, SX09-1306, facilita información sobre los lenguajes de programación RPG III y RPG IV. Este manual contiene tablas y listados de todas las especificaciones y operaciones de ambos lenguajes. Se proporciona una clave para correlacionar las especificaciones y operaciones RPG III con especificaciones y operaciones RPG IV.
- *Configuración de dispositivos locales*, SC10-9618 (SC41-4121), facilita información sobre la configuración de dispositivos locales en el sistema AS/400. Esto incluye información sobre la forma de configurar lo siguiente:
  - Controladores de estación de trabajo locales (incluidos los twinaxiales)
  - Controladores de cinta
  - Dispositivos conectados localmente (incluidos los twinaxiales)
- *Programación de dispositivos de impresora*, SC10-3047 (SC41-4713), facilita información que sirve de ayuda para controlar la impresión. Facilita información específica sobre los elementos de impresión y los conceptos del sistema AS/400, el soporte de spooling de impresión para operaciones de impresión y la conectividad de impresoras. Incluye consideraciones para utilizar PC, otras funciones de impresión tales como el Programa de Utilidad de Gráficos de Empresa (BGU), Funciones Avanzadas de Impresión (AFP) y ejemplos de cómo

trabajar con los elementos de impresión del sistema AS/400 tales como la forma de mover archivos de salida de spool de una cola de salida a otra diferente. También incluye un apéndice con los mandatos de lenguaje de control (CL) utilizados para gestionar la carga de trabajo de impresión.
También se proporcionan los fonts disponibles para su utilización con el sistema AS/400. Las tablas de sustitución de fonts proporcionan una referencia cruzada de los fonts sustituidos si las impresoras no dan soporte a los fonts especificados por la aplicación.

- *Seguridad OS/400 - Básica*, SC10-3046 (SC41-4301), explica por qué es necesaria la seguridad, define los conceptos más importantes y facilita información sobre la planificación, implementación y supervisión de la seguridad básica en el sistema AS/400.
- *Security – Reference*, SC41-4302, indica cómo puede utilizarse el soporte de seguridad del sistema para proteger el sistema y los datos de una utilización por parte de personas que carecen de la debida autorización, proteger los datos frente a daños o destrucción intencionada o no, mantener actualizada la información de seguridad y configurar la seguridad del sistema.
- *Instalación de Software*, SC10-9617 (SC41-4120), da los procedimientos paso a paso para la instalación inicial, la instalación de programas bajo licencia, arreglos temporales del programa (PTF) y lenguajes secundarios de IBM.
  Este manual es también para aquellos usuarios que ya tienen un sistema AS/400 con un release instalado y desean instalar un release nuevo.
- *System API Reference*, SC41-4801, facilita información al programador experimentado sobre la manera de utilizar las interfaz de programación de aplicaciones (API) para funciones OS/400 tales como:
  - Gestor de pantallas dinámicas
  - Archivos (de base de datos, de spool, jerárquicos)
  - Manejo de mensajes
  - Soporte de idiomas
  - Gestión de red
  - Objetos
  - Gestión de problemas
  - Programa de utilidad de registro
  - Seguridad
  - Productos de software
  - Depuración de fuente
  - Tipo UNIX
  - Comunicaciones definidas por el usuario
  - Interfaz de usuario
  - Gestión del trabajo

  Incluye API OPM (modelo de programa original), ILE (Entorno de Lenguaje Integrado) y de tipo UNIX.,
- *Operación del sistema*, SC10-9621 (SC41-4203), facilita información sobre el manejo de mensajes, la forma de trabajar con tareas y la salida de impresora, comunicaciones de dispositivos, la forma de trabajar con funciones de soporte, la manera de limpiar el sistema, etc.
- *Puesta a punto del sistema y gestión de problemas*, SC10-9623 (SC41-4206), facilita información sobre el panel de control de unidades del sistema, la forma de arrancar y detener el sistema, la utilización de cintas y disquetes, la forma de trabajar con arreglos temporales del programa y el manejo de programas.
- *Tape and Diskette Device Programming*, SC41-4716,

Para obtener información sobre COBOL de la interfaz común de programación (CPI) COBOL de la Arquitectura para Aplicaciones de Sistemas (SAA), consulte la publicación siguiente:

- *Systems Application Architecture Common Programming Interface COBOL Reference*, SC26-4354.

Índice

Caracteres Especiales

Números

A

B

C

CH

D

E

F

G

H

I

J

L

LL

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Hoja de Comentarios

AS/400 Advanced Series
ILE COBOL/400
Guía del Programador
Versión 3 Release 7

Número de Publicación SC10-9658-01

En general, ¿está Ud. satisfecho con la información de este libro?

	Muy satisfecho	Satisfecho	Normal	Insatisfecho	Muy insatisfecho
Satisfacción general	□	□	□	□	□

¿Cómo valora los siguientes aspectos de este libro?

	Muy bien	Bien	Acep-table	Insatisfecho	Muy insatisfecho
Organización	□	□	□	□	□
Información completa y precisa	□	□	□	□	□
Información fácil de encontrar	□	□	□	□	□
Utilidad de las ilustraciones	□	□	□	□	□
Claridad de la redacción	□	□	□	□	□
Calidad de la edición	□	□	□	□	□
Adaptación a los formatos, unidades, etc. del país	□	□	□	□	□

Comentarios y sugerencias:

Nombre

Dirección

Compañía u Organización

Teléfono

Hoja de Comentarios
SC10-9658-01

Corte o Doble
Por la Línea

Dóblese por la línea de puntos **Por favor no lo grape** Dóblese por la línea de puntos

PONER
EL
SELLO
AQUÍ

IBM, S.A.
National Language Solutions Center
Av. Diagonal, 571
08029 Barcelona
España

Dóblese por la línea de puntos **Por favor no lo grape** Dóblese por la línea de puntos

Corte o Doble
Por la Línea

Número de Programa: 5716-CB1

Printed in Denmark by IBM Danmark A/S

SC10-9658-01