



iSeries

WebSphere® Development Studio: Guía del programador en ILE RPG

Versión 5

SC10-3607-00





iSeries

WebSphere® Development Studio:
Guía del programador en ILE RPG

Versión 5

SC10-3607-00

¡Nota!

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general que se encuentra bajo “Avisos” en la página 491.

Primera edición (mayo de 2001)

Este manual es la traducción del original inglés: WebSphere Development Studio: ILE RPG Programmer's Guide. Esta edición se aplica a la Versión 5, Release 1, Modificación 0, de IBM WebSphere Development Studio para iSeries (5722-WDS), ILE RPG compiler, y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones. Esta edición es válida únicamente para sistemas RISC (sistema con conjunto reducido de instrucciones).

Esta edición sustituye a SC10-9414-02.

Haga sus pedidos de publicaciones a través de un representante de IBM o de la sucursal de IBM de su localidad. Las publicaciones no están almacenadas en la dirección indicada más abajo.

IBM recibirá con agrado sus comentarios. Puede enviarlos a:

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

También puede enviar sus comentarios por fax o electrónicamente a IBM. Consulte el apartado “Cómo enviar sus comentarios” para obtener una descripción de los métodos.

Al enviar información a IBM, otorga a IBM un derecho no exclusivo de utilizar o distribuir la información de cualquier manera que considere adecuada, sin incurrir en ninguna obligación hacia usted.

© Copyright International Business Machines Corporation 1994, 2001. Reservados todos los derechos.

Contenido

Acerca de esta guía	ix
Quién debería usar esta guía	ix
Requisitos previos e información relacionada	ix
Cómo enviar sus comentarios	x
Novedades en este release	x
Cambios en esta guía desde V4R4	xvi

Parte 1. Introducción a ILE RPG . . . 1

Capítulo 1. Visión general del lenguaje de programación RPG IV	3
Especificaciones de RPG IV	3
Programación de ciclos	4
Lógica de subprocedimiento	6
Indicadores	6
Códigos de operación	6
Ejemplo de un programa ILE RPG	7
Utilización del sistema OS/400	13
Interacción con el sistema	13
WebSphere Development Studio para iSeries	14
Application Development ToolSet	14
WebSphere Development Tools for iSeries	15

Capítulo 2. Programación de RPG en ILE	19
Creación de programas	19
Gestión de programas	21
Llamada a programas	21
Depuración del fuente	22
API enlazables	22
Aplicaciones multihebra	23

Capítulo 3. Estrategias de creación de programas	25
Estrategia 1: aplicación compatible con OPM	25
Método	25
Ejemplo de un programa compatible con OPM	26
Información relacionada	27
Estrategia 2: programa ILE utilizando CRTBNDRPG	27
Método	27
Ejemplo de un programa ILE utilizando CRTBNDRPG	28
Información relacionada	29
Estrategia 3: aplicación ILE utilizando CRTRPGMOD	30
Método	30
Marco hipotético de una aplicación ILE en un solo lenguaje	31
Marco hipotético de una aplicación ILE en lenguaje mixto	31
Marco hipotético de una aplicación avanzada	32
Información relacionada	33
Una estrategia que debe evitar	34

Capítulo 4. Creación de una aplicación utilizando múltiples procedimientos	35
Un módulo de múltiples procedimientos — Visión general	35
Procedimientos principales y subprocedimientos	35
Llamadas de prototipos	36
Ejemplo de un módulo con múltiples procedimientos	39
El programa ARRSRPT completo	45
Consideraciones acerca de la codificación	49
Consideraciones generales	49
Creación del programa	50
Consideraciones acerca del procedimiento principal	50
Consideraciones acerca de los subprocedimientos	50
Para obtener más información	51
Procedimientos principales	51
Subprocedimientos	51
Llamada de prototipos	52

Parte 2. Creación y ejecución de una aplicación ILE RPG . . . 53

Capítulo 5. Entrar sentencias fuente	55
Creación de una biblioteca y de un archivo físico fuente	55
Utilización del programa de utilidad para entrada del fuente (SEU)	56
Utilización de sentencias SQL	59

Capítulo 6. Creación de un programa con el mandato CRTBNDRPG	61
Utilización del mandato CRTBNDRPG	61
Creación de un programa para la depuración del fuente	63
Creación de un programa con enlace estático	64
Creación de un objeto de programa compatible con OPM	65
Utilización de un listado del compilador	67
Obtención de un listado del compilador	67
Personalización de un listado del compilador	68
Corrección de errores de compilación	70
Corrección de errores durante la ejecución	72
Utilización de un listado del compilador para el mantenimiento	73
Acceso al área de datos RETURNCODE	74

Capítulo 7. Creación de un programa con los mandatos CRTRPGMOD y CRTPGM	77
Creación de un objeto de módulo	77
Utilización del mandato CRTRPGMOD	78

Creación de un módulo para la depuración del fuente	82
Ejemplos adicionales	84
Comportamiento de los módulos ILE RPG enlazados	84
Mandatos CL relacionados	84
Enlace de módulos a un programa	84
Utilización del mandato CRTPGM.	86
Ejemplos adicionales	88
Mandatos CL relacionados	89
Utilización de un listado del enlazador	89
Modificación de un módulo o programa.	90
Utilización del mandato UPDPGM	90
Modificación del nivel de optimización	91
Eliminación de la información observable	92
Reducción del tamaño de un objeto	92

Capítulo 8. Creación de un programa de servicio 93

Visión general de los programas de servicio	93
Métodos para crear programas de servicio	94
Creación de un programa de servicio utilizando CRTSRVPGM.	94
Modificar un programa de servicio	95
Mandatos CL relacionados	95
Programa de servicio de ejemplo	96
Creación del programa de servicio.	99
Enlace a un programa	100
Actualización del programa de servicio.	101
Ejemplo de listado del enlazador	102

Capítulo 9. Ejecución de un programa 105

Ejecución de un programa utilizando el mandato CL CALL.	105
Pasar parámetros utilizando el mandato CL CALL	105
Ejecución de un programa desde una aplicación dirigida por un menú	107
Ejecución de un programa utilizando un mandato creado por el usuario.	110
Respuesta a los mensajes de consulta durante la ejecución	110
Finalización de un programa ILE	111
Gestión de los grupos de activación	112
Cómo especificar un grupo de activación	112
Ejecución en el grupo de activación OPM por omisión	113
Mantenimiento de la compatibilidad entre los programas OPM RPG/400 e ILE RPG	113
Suprimir un grupo de activación	114
Mandato Reclamar Recursos	114
Gestión del almacenamiento asignado dinámicamente	115
Gestión del área de almacenamiento dinámico por omisión utilizando operaciones RPG	116
Problemas de almacenamiento dinámico	121
Gestión del propio almacenamiento dinámico utilizando las API enlazables ILE.	122

Capítulo 10. Llamadas a programas y procedimientos 131

Visión general de las llamadas a programas o procedimientos.	131
Llamadas a programas	132
Llamadas a procedimientos.	132
Pila de llamadas	133
Llamadas recurrentes.	134
Consideraciones acerca de cómo pasar parámetros	136
Utilización de una llamada de prototipos	137
Utilización de la operación CALLP	138
Realización de una llamada en una expresión	138
Ejemplos de llamada de formato libre	139
Pasar parámetros de prototipos	139
Métodos para pasar parámetros	139
Utilización de descriptores operativos	142
Omisión de parámetros	143
Comprobación del número de parámetros pasados	145
Pasar menos datos de los necesarios.	150
Orden de evaluación	151
Llamadas entre lenguajes	151
Consideraciones sobre las llamadas entre lenguajes	152
Utilización de las operaciones de llamada de formato fijo	153
Ejemplos de CALL y CALLB	154
Pasar parámetros utilizando PARM y PLIST	154
Volver de un programa o procedimiento llamado	156
Volver de un procedimiento principal	157
Volver de un subprocedimiento	159
Volver utilizando API ILE enlazables	159
Utilización de API enlazables	160
Ejemplos de utilización de API enlazables	161
Llamada a una rutina de gráficos.	161
Llamada a rutinas especiales	162
Consideraciones sobre las hebras múltiples	162
Cómo compartir datos entre más de un módulo	163
Cómo evitar los puntos muertos entre módulos	164

Capítulo 11. RPG y el mundo del eBusiness. 167

RPG y XML	167
RPG y MQSeries	167
RPG y Java	168
Introducción a Java y RPG	168
Llamada a métodos de Java desde ILE RPG	172
Métodos nativos RPG	177
Codificación RPG adicional para utilizar Java	180
Consideraciones adicionales	186
Codificación JNI avanzada	187

Parte 3. Depuración y manejo de excepciones 193

Capítulo 12. Depuración de programas 195

El depurador del fuente ILE	195
Mandatos de depuración	196
Preparación de un programa para su depuración	198

Creación de una vista del fuente raíz	199
Creación de una vista del fuente COPY.	200
Creación de una vista de listado	200
Creación de una vista de sentencias	201
Inicio del depurador del fuente ILE	202
Ejemplo de STRDBG	203
Establecimiento de opciones de depuración	204
Adición/eliminación de programas de una sesión de depuración	204
Ejemplo de adición de un programa de servicio a una sesión de depuración.	205
Ejemplo de eliminación de programas ILE de una sesión de depuración	206
Visualización del fuente del programa	206
Visualización de un módulo diferente	207
Cambio de la vista de un módulo	208
Establecimiento y eliminación de puntos de interrupción	209
Establecimiento y eliminación de puntos de interrupción de trabajo incondicionales	210
Establecimiento y eliminación de puntos de interrupción incondicionales	212
Establecimiento y eliminación de puntos de interrupción de trabajo condicionales	213
Secuencia de Ordenación de Idiomas Nacionales (NLSS)	216
Establecimiento y eliminación de puntos de interrupción de trabajo utilizando números de sentencia	217
Establecimiento y eliminación de puntos de interrupción de hebra condicionales	220
Eliminación de todos los puntos de interrupción de trabajo y hebra	220
Establecimiento y eliminación de condiciones de observación	221
Características de las observaciones	221
Establecimiento de condiciones de observación	222
Visualización de observaciones activas	224
Eliminación de condiciones de observación	224
Ejemplo de establecimiento de una condición de observación	225
Seguir los pasos del objeto de programa	227
Saltar sentencias de llamada	228
Entrar en sentencias de llamada	228
Visualización de datos y expresiones	232
Cambio del valor de los campos	241
Visualización de atributos de un campo	243
Igualar un nombre a un campo, expresión o mandato	244
Soporte de Idiomas Nacionales de depuración del fuente para ILE RPG	245
Ejemplo de fuente para ejemplos de depuración	245
Capítulo 13. Manejo de excepciones	251
Visión general del manejo de excepciones	252
Manejo de excepciones de ILE RPG	254
Utilización de manejadores de excepciones	257
Prioridad del manejador de excepciones	257
Excepciones anidadas	257
Excepciones no manejadas	257
Consideraciones sobre optimización	260

Utilización de manejadores específicos de RPG	261
Especificación de indicadores de error o del amplificador de código de operación 'E'	261
Utilización de un grupo MONITOR	262
Utilización de una subrutina de error	264
Especificación de un punto de retorno en la operación ENDSR	274
Manejadores de condiciones ILE	275
Utilización de un manejador de condiciones	275
Utilización de manejadores de cancelación.	282
Problemas cuando ILE CL supervisa los mensajes de estado y notificación	285

Capítulo 14. Obtención de un vuelco 289

Obtención de un vuelco con formato ILE RPG	289
Utilización del código de operación DUMP	289
Ejemplo de vuelco con formato	290

Parte 4. Trabajar con archivos y dispositivos 297

Capítulo 15. Definición de archivos 299

Asociación de archivos con dispositivos de entrada/salida	299
Denominación de archivos	301
Tipos de descripciones de archivos	301
Utilización de archivos descritos externamente como archivos descritos por programa	302
Ejemplo de algunas relaciones habituales entre programas y archivos.	303
Definición de archivos descritos externamente	304
Cambio de nombre de formatos de registro	304
Cambio de nombres de campo	305
Ignorar formatos de registro	305
Utilización de las especificaciones de entrada para modificar una descripción externa.	306
Utilización de las especificaciones de salida	308
Comprobación de nivel	309
Definición de archivos descrito por programa	310
Operaciones de gestión de datos y ILE RPG	
Operaciones E/S	310

Capítulo 16. Consideraciones generales sobre archivos 313

Alteración temporal y redirección de la entrada y salida de archivos	313
Ejemplo de redirección de entrada y salida de archivos	314
Bloqueo de archivos	315
Bloqueo de registros	316
Compartimiento de una vía de acceso de datos abierta.	317
Spooling	319
Spooling de salida.	319
SRTSEQ/ALTSEQ en un programa RPG en relación con un archivo de DDS	320

Capítulo 17. Acceso a archivos de base de datos 321

Archivos de base de datos	321
Archivos lógicos y archivos físicos	321
Archivos de datos y archivos fuente.	321
Utilización de archivos DISK descritos de forma externa	322
Especificaciones de formato de registro.	322
Vía de acceso	323
Claves válidas para un registro o archivo	325
Bloqueo y desbloqueo de registros	327
Utilización de archivos DISK descritos por programa.	328
Archivo indexado	328
Archivo secuencial	331
Archivo de direcciones de registros	331
Métodos para el proceso de archivos DISK	332
Proceso consecutivo	333
Proceso secuencial por clave	334
Proceso al azar por clave	339
Proceso secuencial entre límites	341
Proceso por número relativo de registro	343
Operaciones de archivo válidas	344
Utilización del control de compromiso	346
Inicio y final del control de compromiso	347
Especificación de archivos para el control de compromiso	349
Utilización de la operación COMMIT	349
Especificación del control de compromiso condicional	351
Control de compromiso en el ciclo del programa	352
Archivos DDM	352
Utilización de archivos DDM anteriores a la Versión 3, Release 1	353

Capítulo 18. Acceso a dispositivos conectados externamente 355

Tipos de archivos de dispositivo	355
Acceso a dispositivos de impresora	356
Especificación de archivos PRINTER	356
Manejo del desbordamiento de página	356
Utilización de la rutina de búsqueda de desbordamiento en archivos descritos por programa.	360
Modificación de la información de control de formularios en un archivo descrito por programa.	363
Acceso a dispositivos de cinta	365
Acceso a dispositivos de pantalla	366
Utilización de archivos secuenciales	366
Especificación de un archivo secuencial.	366
Utilización de archivos SPECIAL	367
Ejemplo de utilización de un archivo SPECIAL	369

Capítulo 19. Utilización de archivos WORKSTN 371

función de comunicaciones intersistemas	371
Utilización de archivos WORKSTN descritos externamente	371
Especificación de indicadores de teclas de función en archivos de dispositivo de pantalla	374

Especificación de teclas de mandato en archivos de dispositivo de pantalla	374
Proceso de un archivo WORKSTN descrito externamente	375
Utilización de subarchivos	375
Utilización de archivos WORKSTN descritos por programa.	379
Utilización de un archivo WORKSTN descrito por programa con un nombre de formato	380
Utilización de un archivo WORKSTN descrito por programa sin nombre de formato	381
Operaciones válidas del archivo WORKSTN	382
Operación EXFMT.	383
Operación READ	383
Operación WRITE	383
Archivos de múltiples dispositivos	383

Capítulo 20. Ejemplo de una aplicación interactiva 387

Archivo físico de base de datos	387
Consulta de menú principal	388
MENUPRIN: DDS para un archivo de dispositivo de pantalla	388
CLIPRIN: fuente RPG	390
Mantenimiento de archivo	391
MAECLIL1: DDS para un archivo lógico	392
MENUmnt: DDS para un archivo de dispositivo de pantalla	393
MNTCLI: Fuente RPG	395
Búsqueda por código postal	402
MAECLIL2: DDS para un archivo lógico	403
MENUMCOD: DDS para un archivo de dispositivo de pantalla	404
BUSCOD: Fuente RPG	406
Búsqueda y consulta por nombre.	410
MECLIL3: DDS para un archivo lógico	411
MENUMBNOM: DDS para un archivo de dispositivo de pantalla	412
NOMBUS: Fuente RPG	415

Parte 5. Apéndices 421

Apéndice A. Diferencias de comportamiento entre OPM RPG/400 y ILE RPG para AS/400 423

Compilación.	423
Ejecución.	424
Depuración y manejo de excepciones	424
E/S.	425
Datos DBCS en campos de tipo carácter	427

Apéndice B. Utilización de la ayuda para la conversión de RPG III a RPG IV 429

Visión general de la conversión	429
Consideraciones sobre los archivos	430
El archivo de anotaciones cronológicas	431
Requisitos de la herramienta de ayuda para la conversión	432

Funciones que no realiza la ayuda para la conversión	432
Conversión del código fuente	432
El mandato CVTRPGSRC	433
Conversión de un miembro mediante los valores por omisión	438
Conversión de todos los miembros de un archivo	439
Conversión de algunos miembros de un archivo	439
Realización de una conversión de prueba	439
Obtención de informes de conversión	440
Conversión de miembros fuente del generador automático de informes	440
Conversión de miembros fuente con SQL incluido	441
Inserción de plantillas de especificaciones	441
Conversión de fuente desde un archivo de datos	441
Ejemplo de conversión de fuente	442
Análisis de la conversión	444
Utilización del informe de conversión	445
Utilización del archivo de anotaciones cronológicas	447
Resolución de problemas de conversión	448
Errores de compilación en código RPG III existente	449
Características de RPG III no soportadas	449
Utilización de la directiva del compilador /COPY	449
Utilización de estructuras de datos descritos externamente	452
Diferencias de tiempo de ejecución	454

Apéndice C. Los mandatos para crear 455

Utilización de los mandatos de CL	455
Cómo interpretar diagramas de sintaxis	455
Mandato CRTBNDRPG	456
Descripción del mandato CRTBNDRPG	459
Mandato CRTRPGMOD	472
Descripción del mandato CRTRPGMOD	474

Apéndice D. Listados del compilador 475

Lectura de un listado del compilador	476
Prólogo	476
Sección de fuente	478
Mensajes de diagnóstico adicionales	483
Posiciones del almacenamiento intermedio de salida	484
Tabla de miembros /COPY	484
Datos de tiempo de compilación	485
Información sobre campos de clave	485
Tabla de referencias cruzadas	486
Lista de referencias externas	487
Resumen de mensajes	488
Resumen final	488
Errores de generación de código y de enlace	489

Avisos 491

Información de la interfaz de programación	492
Marcas registradas y marcas de servicio	492

Bibliografía 495

Índice. 499

Acerca de esta guía

Esta guía proporciona información que muestra cómo utilizar el compilador ILE RPG (ILE RPG) de Integrated Language Environment. ILE RPG es una implementación del lenguaje RPG IV en el servidor iSeries con el sistema operativo Operating System/400 (OS/400). Utilice esta guía para crear y ejecutar aplicaciones ILE desde el fuente RPG IV.

Esta guía le muestra cómo:

- Entrar sentencias fuente de RPG IV
- Crear módulos
- Especificar enlaces lógicos de módulos
- Ejecutar un programa ILE
- Llamar a otros objetos
- Depurar un programa ILE
- Manejar excepciones
- Definir y procesar archivos
- Acceder a dispositivos
- Convertir programas de un formato RPG III a un formato RPG IV
- Leer listados del compilador

Quién debería usar esta guía

Esta guía esta pensada para programadores que conocen el lenguaje de programación RPG pero que desean aprender cómo utilizarlo en el entorno ILE. Esta guía también está pensada para programadores que quieran convertir programas RPG III al formato RPG IV. Está diseñada para servir de guía en la utilización del compilador ILE RPG en el sistema iSeries.

Si bien esta guía muestra cómo utilizar RPG IV en una estructura ILE, no proporciona información detallada acerca de las especificaciones y operaciones de RPG IV. Para obtener una descripción detallada del lenguaje, consulte la publicación *ILE RPG Reference*, SC09-2508-03.

Antes de utilizar esta guía, debe:

- Saber utilizar los menús y las pantallas del servidor iSeries o los mandatos del lenguaje de control (CL).
- Tener la autorización adecuada para los mandatos CL y los objetos que se describen en esta guía.
- Entender perfectamente los conceptos de ILE tal como se describen de forma detallada en la publicación *ILE Concepts*, SC41-5606-05.

Requisitos previos e información relacionada

Utilice el iSeries Information Center como punto de partida para buscar información técnica en relación con iSeries y AS/400e. Puede acceder al Information Center de dos formas:

- En el sitio Web siguiente:
<http://www.ibm.com/eserver/iseries/infocenter>

- En los CD-ROM que se distribuyen con el pedido de Operating System/400: *iSeries Information Center*, SK3T-4091-00. Este paquete también contiene las versiones en formato PDF de los manuales de iSeries, *iSeries Information Center: Supplemental Manuals*, SK3T-4092-00 (que sustituye al CD-ROM de biblioteca en soporte software).

El iSeries Information Center contiene asesores y temas importantes, tales como mandatos CL, interfaces de programación de aplicaciones (API) del sistema, particiones lógicas, clusters, Java, TCP/IP, servicio Web y redes seguras. También facilita enlaces con Libros rojos de IBM relacionados y enlaces de Internet con otros sitios Web de IBM como por ejemplo Technical Studio y la página de presentación de IBM.

Los manuales que son más relevantes para el ILE RPG compiler figuran en la "Bibliografía" en la página 495.

Cómo enviar sus comentarios

Sus comentarios son importantes, pues nos permiten proporcionar información más precisa y de mayor calidad. IBM acogerá con agrado cualquier comentario sobre este manual o cualquier otra documentación relacionada con iSeries.

- Si prefiere enviar sus comentarios por correo, utilice la dirección siguiente:

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

Si va a enviar una hoja de comentarios del lector desde un país diferente de los Estados Unidos, puede entregarla a la delegación local de IBM o a un representante de IBM para evitar los costes de franqueo.

- Si prefiere enviar sus comentarios por fax, utilice el número siguiente:
 - 34 93 321 6134
- Si prefiere enviar sus comentarios electrónicamente, utilice una de las siguientes direcciones de correo electrónico:
 - Comentarios sobre manuales:
torrcf@ca.ibm.com
HOJACOM@VNET.IBM.COM
 - Comentarios sobre el Information Center:
RCHINFOC@us.ibm.com

Asegúrese de incluir la información siguiente:

- El nombre del manual.
- El número de publicación del manual.
- El número de página o el tema al que hace referencia su comentario.

Novedades en este release

El compilador ILE RPG forma parte del producto IBM WebSphere Development Studio, que ahora incluye los compiladores C/C++ y COBOL, así como las herramientas del conjunto de herramientas para el desarrollo de aplicaciones.

Las mejoras más importantes realizadas en RPG IV desde V4R4 son una mayor facilidad de intercambio de información con Java, nuevas funciones incorporadas, especificaciones de cálculo de formato libre, control de qué archivo está abierto, nombres de subcampos calificados y manejo de errores mejorado.

La lista siguiente describe estas mejoras:

- Soporte mejorado para llamadas entre Java e ILE RPG utilizando la interfaz nativa Java (JNI):
 - Un nuevo tipo de datos de objeto.
 - Una nueva palabra clave de especificación de definición: CLASS.
 - La palabra clave de especificación de definición LIKE se ha extendido para dar soporte a objetos.
 - La palabra clave de especificación de definición EXTPROC se ha extendido para dar soporte a procedimientos Java.
 - Nuevos códigos de estado.
- Nuevas funciones incorporadas:
 - Funciones para convertir un número en una duración que puede emplearse en expresiones aritméticas: %MSECONDS, %SECONDS, %MINUTES, %HOURS, %DAYS, %MONTHS y %YEARS.
 - La función %DIFF, para restar un valor de fecha, hora o indicación de la hora de otro.
 - Funciones para convertir una serie de caracteres (o fecha o indicación de la hora) en una fecha, hora o indicación de la hora: %DATE, %TIME y %TIMESTAMP.
 - La función %SUBDT, para extraer un subconjunto de una fecha, hora o indicación de la hora.
 - Funciones para asignar o reasignar almacenamiento: %ALLOC y %REALLOC.
 - Funciones para buscar un elemento en una matriz: %LOOKUP, %LOOKUPGT, %LOOKUPGE, %LOOKUPLT y %LOOKUPLE.
 - Funciones para buscar un elemento en una tabla: %TLOOKUP, %TLOOKUPGT, %TLOOKUPGE, %TLOOKUPLT y %TLOOKUPLE.
 - Funciones para verificar que una serie contenga sólo los caracteres especificados (o buscar la primera o última excepción a esta norma): %CHECK y %CHECKR
 - La función %XLATE, para convertir una serie a partir de una lista de caracteres origen en caracteres destino.
 - La función %OCCUR, para obtener o establecer la aparición actual de una estructura de datos de apariciones múltiples.
 - La función %SHTDN, para determinar si el operador ha solicitado la conclusión.
 - La función %SQRT, para calcular la raíz cuadrada de un número.
- Una nueva sintaxis de formato libre para las especificaciones de cálculo. Un bloque de especificaciones de cálculo de formato libre se delimita mediante las directivas de compilación /FREE y /END-FREE.
- Puede especificar las palabras clave EXTFILE y EXTMBR en la especificación de archivo para controlar qué archivo externo se utiliza cuando se abre un archivo.
- Soporte para nombres calificados en estructuras de datos:
 - Una nueva palabra clave de especificación de definición: QUALIFIED. Esta palabra clave especifica que los nombres de subcampos se calificarán con el nombre de la estructura de datos.

- Una nueva palabra clave de especificación de definición: LIKEDS. Esta palabra clave especifica que los subcampos se replican a partir de otra estructura de datos. Los nombres de subcampos se calificarán con el nombre de la nueva estructura de datos. La palabra clave LIKEDS está permitida para los parámetros de prototipos; permite que los subcampos del parámetro se utilicen directamente en el procedimiento llamado.
- La palabra clave de especificación de definición INZ se ha extendido para permitir que una estructura de datos se inicialice a partir de su estructura de datos padre.
- Manejo de errores mejorado:
 - Tres nuevos códigos de operación (MONITOR, ON-ERROR y ENDMON) permiten definir un grupo de operaciones con manejo de errores condicional en función del código de estado.

Asimismo se han efectuado otras mejoras en este release. Son las siguientes:

- Puede especificar paréntesis en una llamada de procedimiento que no tiene parámetros.
- Puede especificar que un procedimiento utilice los convenios de llamada C de ILE o CL de ILE, en la palabra clave de especificación de definición EXTPROC.
- Los siguientes nombres /DEFINE están predefinidos: *VnRnMn, *ILERPG, *CRTBNDRPG y *CRTRPGMOD.
- La longitud de la serie de búsqueda en una operación %SCAN ahora puede ser superior a la de la serie donde se busca. (La serie no se encontrará, pero no se producirá una condición de error.)
- El parámetro para las palabras clave DIM, OCCURS y PERRCD ya no necesita estar definido con anterioridad.
- La función incorporada %PADDR ahora puede tomar un nombre de prototipo o un nombre de punto de entrada como argumento.
- Un nuevo código de operación, ELSEIF, combina los códigos de operación ELSE e IF sin que sea necesario un código ENDIF adicional.
- El código de operación DUMP ahora proporciona soporte para el expansor A, lo que significa que siempre se genera un vuelco, aunque se haya especificado DEBUG(*NO).
- Una nueva directiva, /INCLUDE, equivale a /COPY con la excepción de que el preprocesador SQL no expande /INCLUDE. Los archivos incluidos no pueden contener variables del lenguaje principal ni SQL intercalado.
- La palabra clave de especificación de archivo OFLIND ahora puede tomar cualquier indicador, incluido un indicador con nombre, como argumento.
- La palabra clave LICOPT (opciones de código interno bajo licencia) ahora está disponible en los mandatos CRTRPGMOD y CRTBNDRPG.
- La palabra clave de descripción de archivo PREFIX ahora puede tomar un literal de caracteres en mayúsculas como argumento. El literal puede terminar en un punto, lo que permite que el archivo se utilice con subcampos calificados.
- La palabra clave de especificación de definición PREFIX también puede tomar un literal de caracteres en mayúsculas como argumento. Este literal no puede terminar en un punto.

Las tablas siguientes resumen los elementos del lenguaje modificados y nuevos, en función de la parte del lenguaje afectada.

Tabla 1. Elementos del lenguaje modificados desde V4R4

Unidad de lenguaje	Elemento	Descripción
Funciones incorporadas	%CHAR(expresión[:formato])	El segundo parámetro opcional especifica el formato deseado para una fecha, hora o indicación de la hora. El resultado utiliza el formato y los separadores del formato especificado, no el formato y los separadores de la entrada.
	%PADDD(nombre-prototipo)	Esta función ahora puede tomar un nombre de prototipo o un nombre de punto de entrada como argumento.
Palabras clave de especificación de definición	EXTPROC(*JAVA:nombre-clase:nombre-proc)	Especifica que se llama a un método Java.
	EXTPROC(*CL:nombre-proc)	Especifica un procedimiento que utiliza los convenios CL de ILE para los valores de retorno.
	EXTPROC(*CWIDEN:nombre-proc)	Especifica un procedimiento que utiliza los convenios C de ILE con ensanchamiento de parámetros.
	EXTPROC(*CNOWIDEN:nombre-proc)	Especifica un procedimiento que utiliza los convenios C de ILE sin ensanchamiento de parámetros.
	INZ(*LIKEDS)	Especifica que una estructura de datos definida con la palabra clave LIKEDS hereda la inicialización de su estructura de datos padre.
	LIKE(nombre-objeto)	Especifica que un objeto tiene la misma clase que otro objeto.
	PREFIX(literal-caracteres[:número])	Añade como prefijo a los subcampos el literal de caracteres especificado; opcionalmente sustituye el número especificado de caracteres.
Palabras clave de especificación de archivo	OFLIND(nombre)	Esta palabra clave ahora puede tomar cualquier indicador con nombre como parámetro.
	PREFIX(literal-caracteres[:número])	Añade como prefijo a los subcampos el literal de caracteres especificado; opcionalmente sustituye el número especificado de caracteres.
Códigos de operación	DUMP (A)	Este código de operación ahora puede tomar el expansor A, lo que hace que siempre se genere un vuelco aunque se haya especificado DEBUG(*NO).

Tabla 2. Elementos del lenguaje nuevos desde V4R4

Unidad de lenguaje	Elemento	Descripción
Tipos de datos	Objeto	Se utiliza para los objetos Java.
Directivas del compilador	/FREE ... /END-FREE	Las directivas del compilador /FREE... /END-FREE indican un bloque de especificaciones de cálculo de formato libre.
	/INCLUDE	Equivala a /COPY, con la excepción de que el preprocesador SQL no expande /INCLUDE. Puede utilizarse para incluir archivos anidados que están dentro del archivo copiado. El archivo copiado no puede tener variables del lenguaje principal ni SQL intercalado.

Tabla 2. Elementos del lenguaje nuevos desde V4R4 (continuación)

Unidad de lenguaje	Elemento	Descripción
Palabras clave de especificación de definición	CLASS(*JAVA:nombre-clase)	Especifica la clase de un objeto.
	LIKEDS(nombre-estr-datos)	Especifica que una estructura de datos, un parámetro de prototipos o un valor de retorno hereda los subcampos de otra estructura de datos.
	QUALIFIED	Especifica que los nombres de subcampos de una estructura de datos se califican con el nombre de la estructura de datos.
Palabras clave de especificación de archivo	EXTFILE(nombrearchivo)	Especifica qué archivo se abre. El valor puede ser un literal o una variable. El nombre de archivo por omisión es el nombre especificado en la posición 7 de la especificación de archivo. La biblioteca por omisión es *LIBL.
	EXTMBR(nombremiembro)	Especifica qué miembro se abre. El valor puede ser un literal o una variable. El valor por omisión es *FIRST.

Tabla 2. Elementos del lenguaje nuevos desde V4R4 (continuación)

Unidad de lenguaje	Elemento	Descripción
Funciones incorporadas	%ALLOC(núm)	Asigna la cantidad de almacenamiento especificada.
	%CHECK(comparador:base{:inicio})	Localiza el primer carácter de la serie base que no está en el comparador.
	%CHECKR(comparador:base{:inicio})	Localiza el último carácter de la serie base que no está en el comparador.
	%DATE(expresión{:formato-fecha})	Convierte la expresión en una fecha.
	%DAYS(núm)	Convierte el número en una duración, en días.
	%DIFF(op1:op2:unidad)	Calcula la diferencia (duración) entre dos valores de fecha, hora o indicación de la hora en las unidades especificadas.
	%HOURS(núm)	Convierte el número en una duración, en horas.
	%LOOKUPxx(arg:matriz{:índiceinicio{:númelems}})	Localiza el argumento especificado, o el tipo de coincidencia aproximada especificado, en la matriz especificada.
	%MINUTES(núm)	Convierte el número en una duración, en minutos.
	%MONTHS(núm)	Convierte el número en una duración, en meses.
	%MSECONDS(núm)	Convierte el número en una duración, en microsegundos.
	%OCCUR(nombre-estr-datos)	Establece u obtiene la posición actual de una estructura de datos de apariciones múltiples.
	%REALLOC(puntero:número)	Reasigna la cantidad de almacenamiento especificada para el puntero puntero.
	%SECONDS(núm)	Convierte el número en una duración, en segundos.
	%SHTDN	Comprueba si el operador del sistema ha solicitado la conclusión.
	%SQRT(expresión-numérica)	Calcula la raíz cuadrada del número especificado.
	%SUBDT(valor:unidad)	Extrae la parte especificada de un valor de fecha, hora o indicación de la hora.
	%THIS	Devuelve un valor de objeto que contiene una referencia a la instancia de clase en cuyo nombre se llama al método nativo.
	%TIME(expresión{:formato-hora})	Convierte la expresión en una hora.
	%TIMESTAMP(expresión{:ISO *ISO0})	Convierte la expresión en una indicación de la hora.
	%TLOOKUP(arg:tabla-búsqueda{:tabla-alt})	Localiza el argumento especificado, o el tipo de coincidencia aproximada especificado, en la tabla especificada.
	%XLATE(origen:destino:serie{:posinic})	Convierte la serie especificada, en función de la serie origen y la serie destino.
	%YEARS(núm)	Convierte el número en una duración, en años.

Tabla 2. Elementos del lenguaje nuevos desde V4R4 (continuación)

Unidad de lenguaje	Elemento	Descripción
Códigos de operación	MONITOR	Empieza un grupo de operaciones con manejo de errores condicional.
	ON-ERROR	Lleva a cabo el manejo de errores condicional, según el código de estado.
	ENDMON	Finaliza un grupo de operaciones con manejo de errores condicional.
	ELSEIF	Equivale a un código de operación ELSE seguido de un código de operación IF.
Palabras clave CRTBNDRPG y CRTRPGMOD	LICOPT(opciones)	Especifica opciones de código interno bajo licencia (LIC).

Cambios en esta guía desde V4R4

Esta guía de la versión V5R1, *ILE RPG Programmer's Guide*, SC10-9414-03 (SC09-2507-03) difiere en muchos puntos de la guía de la versión V4R4, *ILE RPG Programmer's Guide*, SC10-9414-02 (SC09-2507-02). La mayor parte de los cambios están relacionados con las ampliaciones realizadas desde V4R4; otros cambios responden a pequeñas correcciones técnicas. Para ayudarle a utilizar este manual, los cambios técnicos y las ampliaciones se indican con una barra vertical (|).

Nota: muchos de los ejemplos que contiene esta guía se han modificado a fin de mostrarse con el estilo de codificación de formato libre, en lugar del tradicional. Estos ejemplos modificados no se han marcado con una barra vertical. Consulte en la publicación *ILE RPG Reference* una descripción detallada de las diferencias entre los dos estilos de codificación.

Parte 1. Introducción a ILE RPG

Antes de utilizar ILE RPG para crear un programa, debe conocer ciertos aspectos del entorno en el que lo utilizará. Esta parte proporciona información acerca de los siguientes temas que debería conocer:

- Visión general del lenguaje RPG IV
- Papel de los componentes de Integrated Language Environment en la programación en RPG
- Estrategias de creación de programas Integrated Language Environment
- Visión general de cómo codificar un módulo con más de un procedimiento y llamadas de prototipos

Capítulo 1. Visión general del lenguaje de programación RPG IV

Este capítulo presenta una visión general de las características del lenguaje de programación RPG IV que distinguen RPG de otros lenguajes de programación. Debe estar familiarizado con todas estas características antes de programar en el lenguaje RPG IV. Las características que aquí se tratan incluyen las cuestiones siguientes:

- Codificación de especificaciones
- Ciclo del programa
- Indicadores
- Códigos de operación

Para obtener más información sobre RPG IV, consulte la publicación *ILE RPG Reference*.

Especificaciones de RPG IV

El código RPG se escribe en diversos formularios de especificación, cada uno con un grupo de funciones determinado. Muchas de las entradas que componen un tipo de especificación dependen de la posición. Cada entrada debe empezar en una posición determinada dependiendo del tipo de entrada y del tipo de especificación.

Hay siete tipos de especificaciones de RPG IV. Todos los tipos de especificación son optativos. Las especificaciones deben entrarse en el programa fuente en el orden que se muestra a continuación.

Sección fuente principal:

1. Las **especificaciones de control** proporcionan al compilador información acerca de la generación y ejecución de los programas, como por ejemplo el nombre del programa, el formato de fecha y la utilización de un orden de clasificación alternativo o de conversión de archivos.
2. Las **especificaciones de descripción de archivos** describen todos los archivos que utiliza el programa.
3. Las **especificaciones de definición** describen los datos que utiliza el programa.
4. Las **especificaciones de entrada** describen los campos y registros de entrada que utiliza el programa.
5. Las **especificaciones de cálculo** describen los cálculos efectuados en los datos y el orden de los cálculos. Las especificaciones de cálculo controlan también ciertas operaciones de entrada y salida.
6. Las **especificaciones de salida** describen los campos y registros de salida que utiliza el programa.

Sección de subprocedimiento:

1. Las **especificaciones de procedimiento** marcan el principio y el final del subprocedimiento, indican el nombre del subprocedimiento y si se ha de exportar.
2. Las **especificaciones de definición** describen los datos locales que utiliza el subprocedimiento.

Visión general de RPG IV

3. Las **especificaciones de cálculo** describen los cálculos efectuados en los datos globales y locales y el orden de los cálculos.

Programación de ciclos

Cuando un sistema procesa datos, debe realizar el proceso en un orden determinado. Este orden lógico viene proporcionado por:

- El compilador ILE RPG
- El código del programa

La lógica que proporciona el compilador se denomina **ciclo del programa**. Cuando deja que el compilador proporcione la lógica de los programas, esto se denomina **programación de ciclos**.

El ciclo del programa es un serie de pasos que el programa repite hasta que alcanza una condición de fin de archivo. En función de las especificaciones que codifique, el programa puede o no utilizar cada paso del ciclo.

Si desea que el ciclo controle los archivos, no es necesario que especifique la información de codificación de las especificaciones de RPG del programa fuente acerca de registros de estos archivos. El compilador suministra el orden lógico de estas operaciones, y algunas operaciones de salida, cuando se compila el programa fuente.

Si no desea que el ciclo controle los archivos, debe finalizar su programa de otro modo, ya sea creando una condición de fin de archivo activando el indicador de último registro (LR), creando una condición de retorno activando el indicador de retorno (RT) o regresando directamente mediante la operación RETURN.

Nota: no se genera ningún código de ciclo para los subprocedimientos ni cuando se especifica NOMAIN en la especificación de control.

La Figura 1 en la página 5 muestra los pasos específicos del flujo general del ciclo del programa RPG.

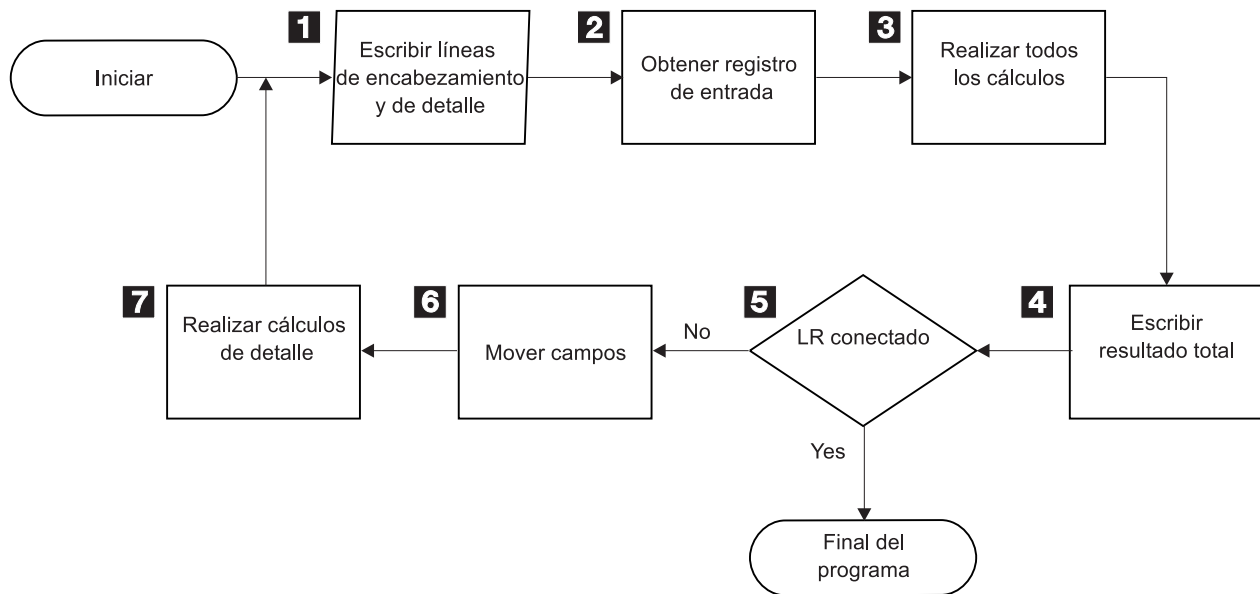


Figura 1. Ciclo de lógica del programa RPG

- 1** RPG procesa todas las líneas de cabecera y de detalle (H o D en la posición 17 de las especificaciones de salida).
- 2** RPG lee el siguiente registro y activa los indicadores de identificación de registro y de nivel de control.
- 3** RPG procesa los cálculos de totales (condicionados mediante los indicadores de nivel de control L1 a L9, un indicador LR o una entrada L0).
- 4** RPG procesa todas las líneas de salida de totales (identificadas mediante una T en la posición 17 de las especificaciones de salida).
- 5** RPG determina si el indicador LR está activado. Si está activado, el programa finaliza.
- 6** Los campos de los registros de entrada seleccionados se mueven del registro a un área de proceso. RPG activa los indicadores de campo.
- 7** RPG procesa todos los cálculos de detalle (no condicionados mediante los indicadores de nivel de control en las posiciones 7 y 8 de las especificaciones de cálculo). Utiliza los datos del registro al principio del ciclo.

El primer ciclo

La primera y última vez del ciclo del programa difieren un poco de otros ciclos. Antes de leer el primer registro por primera vez mediante el ciclo, el programa efectúa tres cosas:

- maneja parámetros de entrada, abre archivos, inicializa datos del programa
- escribe los registros condicionados mediante el indicador 1P (primera página)
- procesa todas las operaciones de salida de detalle y de cabecera.

Por ejemplo, las líneas de cabecera impresas antes de leer el primer registro podrían estar formadas por información de cabecera de página o constante, o

Visión general de RPG IV

campos especiales como por ejemplo PAGE y *DATE. El programa también ignora los pasos de cálculos de totales y salida de totales en el primer ciclo.

El último ciclo

La última vez que un programa pasa por el ciclo, cuando no hay más registros disponibles, el programa *activa* el indicador LR (último registro) y los indicadores L1 a L9 (nivel de control). El programa procesa los cálculos de totales y la salida de totales, a continuación se cierran todos los archivos y luego finaliza el programa.

Lógica de subprocedimiento

El flujo general de un subprocedimiento es mucho más sencillo; los cálculos de un subprocedimiento se realizan una vez y a continuación el subprocedimiento regresa. No se genera código de ciclo para un subprocedimiento.

Indicadores

Un **indicador** es un campo de un carácter de byte que está activado ('1') o desactivado ('0'). Normalmente se utiliza para indicar el resultado de una operación o para condicionar (controlar) el proceso de una operación. Los indicadores son como interruptores del flujo de la lógica del programa. Determinan la vía que tomará el programa durante el proceso, en función de cómo estén establecidos o de cómo se utilicen.

Los indicadores pueden definirse como variables en las especificaciones de definición. También puede utilizar indicadores de RPG IV, definidos por una entrada de una especificación o por el propio programa RPG IV.

Cada indicador de RPG IV tiene un nombre de dos caracteres (por ejemplo, LR, 01, H3), y se le hace referencia en algunas entradas de algunas especificaciones sólo mediante el nombre de dos caracteres, y en otras mediante el nombre especial *INxx en el que xx es el nombre de dos caracteres. Puede utilizar varios tipos de estos indicadores; cada tipo indica algo diferente. Las posiciones de la especificación en la que defina un indicador determina la utilización del indicador. Una vez que haya definido un indicador en el programa, el indicador puede limitar o controlar las operaciones de cálculo y de salida.

Las variables de indicadores pueden utilizarse en cualquier lugar donde puede utilizarse un indicador con el formato *INxx con la excepción de las palabras clave OFLIND y EXTIND en las especificaciones de descripción de archivo.

Un programa en RPG establece y restablece ciertos indicadores en momentos determinados durante el ciclo de programa. Además, el estado de los indicadores se puede cambiar explícitamente en las operaciones de cálculo.

Códigos de operación

El lenguaje de programación RPG IV le permite realizar muchos tipos de operaciones distintas con los datos. Los **códigos de operación**, entrados en las especificaciones de cálculo, indican qué operaciones se efectuarán. Por ejemplo, si desea leer un registro nuevo, puede utilizar el código de operación READ. A continuación aparece una lista de los tipos de operaciones disponibles.

- Operaciones aritméticas
- Operaciones de matrices

- Operaciones de bit
- Operaciones de bifurcación
- Operaciones de llamada
- Operaciones de comparación
- Operaciones de conversión
- Operaciones del área de datos
- Operaciones de fecha
- Operaciones declarativas
- Operaciones de manejo de errores
- Operaciones de archivo
- Operación de activación de indicador
- Operaciones de información
- Operaciones de inicialización
- Operaciones de gestión de memoria
- Operaciones de mover
- Operaciones de mover zona
- Operaciones de resultado
- Operaciones de tamaño
- Operaciones de serie
- Operaciones de programación estructurada
- Operaciones de subrutina
- Operaciones de prueba

Ejemplo de un programa ILE RPG

Este apartado muestra un programa ILE RPG sencillo que realiza los cálculos de nóminas.

Enunciado del problema

El departamento de nóminas de una empresa pequeña desea crear una salida impresa que liste la paga de los empleados para aquella semana. Suponga que hay dos archivos en discos, EMPLEADO y TRANSACC en el sistema.

El primer archivo EMPLEADO, contiene los registros de empleados. La figura que se encuentra a continuación muestra el formato de un registro de empleados:

EMP_REC

EMP_NUMBER	EMP_NAME	EMP_RATE	
1	6	22	27

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.....T.Nombre++++RLon++TDpB.....Funciones+++++*****
A          R EMP_REC
A          EMP_NUMBER      5          TEXT('NÚMERO DE EMPLEADO')
A          EMP_NAME        16          TEXT('NOMBRE DE EMPLEADO')
A          EMP_RATE         5  2       TEXT('TARIFA DE EMPLEADO')
A          K EMP_NUMBER

```

Figura 2. DDS para archivo físico Empleado

El segundo archivo, TRANSACC, tiene el número de horas que ha trabajado cada empleado durante aquella semana, y la bonificación que pueda haber recibido el

Ejemplo de un programa ILE RPG

empleado. La figura a continuación muestra el formato de un registro de transacciones:

TRN_REC

TRN_NUMBER	TRN_HOURS	TRN_BONUS	
1	6	10	16

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+
A.....T.Nombre++++RLon++TDpB.....Funciones+++++
A          R TRN_REC
A          TRN_NUMBER      5          TEXT('NUMERO DE EMPLEADO')
A          TRN_HOURS       4 1        TEXT('HORAS TRABAJADAS')
A          TRN_BONUS       6 2        TEXT('BONIFICACIÓN')
```

Figura 3. DDS para archivo físico TRANSACC

La paga de cada empleado se calcula multiplicando las "horas" (del archivo TRANSACC) y la "tarifa" (del archivo EMPLEADO) y añadiendo la "bonificación" del archivo TRANSACC. Si se trabaja más de 40 horas, se paga al empleado 1,5 veces la tarifa normal.

Especificaciones de control

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
HPalabrasclave+++++
H DATEDIT(*DMY/)
```

La fecha de hoy se imprimirá en formato de día, mes y año con una barra "/" como separador.

Especificación de descripción de archivo

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
FNombarch++IPEASFLreg+LonLK+AIDispost+.Palabrasclave+++++
FTRANSACC IP E          K DISK
FEMPLEADO IF E          K DISK
FQSYSPRT  0  F  80      PRINTER
```

Hay tres archivos definidos en las especificaciones de descripción de archivo:

- El archivo TRANSACC está definido como el archivo Primario de Entrada. El ciclo del programa ILE RPG controla la lectura de los registros de este archivo.
- El archivo EMPLEADO está definido como el archivo de Procedimiento Completo de Entrada. La lectura de los registros de este archivo está controlada por las operaciones de las especificaciones de cálculo.
- El archivo QSYSPRT está definido como archivo de salida de impresora.

Especificaciones de definición

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
D+Nombre+++++ETDsDesde+++A/L+++IDc.Palabrasclave+++++
D Paga          S          8P 2
D Cabecera1     C          'NUMERO  NOMBRE          TARIFA  H-
D              '          ORAS  BONIFIC.  PAGA      '
D Cabecera2     C          '
D              '          '          '          '
D CalcPay       PR          8P 2
D Tarifa        5P 2 VALUE
D Horas         10U 0 VALUE
D Bonificación  5P 2 VALUE

```

Utilizando las especificaciones de definición, declare una variable llamada "Paga" para que contenga la paga de una semana de un empleado y dos constantes "Cabecera1" y "Cabecera2" como ayuda para imprimir las cabeceras del informe.

Especificaciones de cálculo

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...
/free
  chain trn_number emp_rec;
  if %found(emp_rec);
    pay = CalcPay (emp_rate: trn_hours: trn_bonus);
  endif;
/end-free

```

Las entradas de codificación de las especificaciones de cálculo son:

- Utilizando el código de operación CHAIN, el campo TRN_NUMBER del archivo de transacciones se utiliza para buscar el registro con el mismo número de empleado en el archivo de empleados.
- Si la operación CHAIN es satisfactoria (es decir, el indicador 99 está desactivado), se evalúa la paga para el empleado. El resultado se "redondea" y se almacena en la variable llamada Paga.

Ejemplo de un programa ILE RPG

Especificaciones de salida

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+...						
ONombarch++DF..N01N02N03Nomexc+++B++A++Sb+Sa+.....						
O.....N01N02N03Campo+++++++YB.Fin++PConstant/palabraedic/DTformato						
0	QSYSPRT	H	1P	2	3	
0					35	'REGISTRO DE NOMINAS'
0			*DATE	Y	60	
0		H	1P	2		
0					60	Cabecera1
0		H	1P	2		
0					60	Cabecera2
0		D	N1PN99	2		
0			TRN_NUMBER		5	
0			EMP_NAME		24	
0			EMP_RATE	L	33	
0			TRN_HOURS	L	40	
0			TRN_BONUS	L	49	
0			Paga		60	'\$ 0. '
0		D	N1P 99	2		
0			TRN_NUMBER		5	
0					35	'** NO EN ARCHIVO DE EMPLEADOS **'
0		T	LR			
0					33	'FIN DEL LISTADO'

Las especificaciones de salida describen qué campos se han de grabar en la salida QSYSPRT:

- Las Líneas de Cabecera que contienen la serie de caracteres de la constante 'REGISTRO DE NOMINA' además de las cabeceras para la información de detalle se imprimirán si está activado el indicador 1P. El ciclo del programa ILE RPG activa el indicador 1P durante el primer ciclo.
- Las Líneas de Detalle están condicionadas mediante los indicadores 1P y 99. Las Líneas de Detalle no se imprimen en el tiempo de 1P. La N99 sólo permitirá que las Líneas de detalle se impriman si el indicador 99 está desactivado, el cual indica que se ha encontrado el registro del empleado correspondiente. Si el indicador 99 está activado, se imprimirá en su lugar el número de empleado y la serie de caracteres de la constante '** NO EN ARCHIVO DE EMPLEADOS **'.
- La Línea de Total contiene la serie de caracteres de la constante 'FIN DE LISTADO'. Se imprimirá durante el último ciclo del programa.

Un subprocedimiento

El subprocedimiento calcula la paga del empleado utilizando los parámetros que se le han pasado. El valor resultante se devuelve al llamador utilizando la sentencia RETURN.

Las especificaciones de procedimiento indican el principio y el fin del procedimiento. Las especificaciones de definición definen el tipo de retorno del procedimiento, los parámetros para el procedimiento y las horas extras de la variable local.

Ejemplo de un programa ILE RPG

```

P CalcPay          B
D CalcPay          PI          8P 2
D Tarifa           5P 2 VALUE
D Horas           10U 0 VALUE
D Bonificación     5P 2 VALUE
D Horas extra      S          5P 2 INZ(0)

/free
// Determine las horas extra a pagar.
if Horas > 40;
    Horas extra = (Horas - 40) * Tarifa * 1.5
    Horas = 40;
endif;
// Calcule la paga total y devuelva el resultado al llamador.
return Tarifa * Horas + Bonificación + Horas extra;
/end-free
P CalcPay          E

```

Todo el programa fuente

La figura siguiente combina todas las especificaciones utilizadas en este programa. Esto es lo que se debe entrar en el archivo fuente para este programa.

```

*-----*
* DESCRIPCIÓN: Este programa crea una salida impresa de la paga semanal *
*               de los empleados.                                     *
*-----*
H DATEDIT(*DMY/)
*-----*
* Definiciones de archivo                                           *
*-----*
FTRANSACC IP E          K DISK
FEMPLEADO IF E          K DISK
FQSYSPRT  0 F 80       PRINTER
*-----*
* Declaraciones de variables                                       *
*-----*
D Paga          S          8P 2

```

Figura 4. Ejemplo de un programa de cálculo de nóminas (Pieza 1 de 3)

Ejemplo de un programa ILE RPG

```

*-----*
* Declaraciones de constantes                                     *
*-----*
D Cabecera1      C              'NÚMERO  NOMBRE      TARIFA H-
D                                     ORAS  BONIFIC. PAGA  '
D Cabecera2      C              '_____'
D                                     _____'
D                                     _____'
*-----*
* Definición de prototipo para subprocedimiento CalcPay        *
*-----*
D CalcPay        PR              8P 2
D Tarifa         5P 2 VALUE
D Horas          10U 0 VALUE
D Bonificación   5P 2 VALUE
*-----*
* Para cada registro del archivo de transacciones (TRANSACC), si se *
* encuentra empleado, calcular paga de empleado e imprimir detalle. *
*-----*
/free
  chain trn_number emp_rec;
  if %found(emp_rec);
    pay = CalcPay (emp_rate: trn_hours: trn_bonus);
  endif;
/end-free
*-----*
* Diseño de informe                                           *
* -- imprimir las líneas de cabecera si está activado 1P      *
* -- si se encuentra el registro (el indicador 99 está desactivado) *
*   imprimir los detalles de nómina, de lo contrario imprimir un *
*   registro de excepción                                       *
* -- imprimir 'FIN DEL LISTADO' cuando LR se active           *
*-----*
OQSYSPT  H    1P              2 3
0
0          *DATE              Y    60
0          H    1P              2
0          60 Cabecera1
0          H    1P              2
0          60 Cabecera2
0          D    N1PN99          2
0          TRN_NUMBER          5
0          EMP_NAME            24
0          EMP_RATE            L 33
0          TRN_HOURS           L 40
0          TRN_BONUS           L 49
0          Paga                60 '$      0.  '
0          D    N1P 99          2
0          TRN_NUMBER          5
0          35 '** NO EN ARCHIVO DE EMPLEADOS **'
0          T    LR
0          33 'FIN DEL LISTADO'

```

Figura 4. Ejemplo de un programa de cálculo de nóminas (Pieza 2 de 3)

```

*-----*
* Subprocedimiento  -- calcula la paga de horas extra.      *
*-----*
P CalcPay          B
D CalcPay          PI          8P 2
D Tarifa           5P 2 VALUE
D Horas           10U 0 VALUE
D Bonificación     5P 2 VALUE
D Horas extra      S          5P 2 INZ(0)

/free
// Determine las horas extra a pagar.
if Horas > 40;
    Horas extra = (Horas - 40) * Tarifa * 1.5
    Horas = 40;
endif;
// Calcule la paga total y devuelva el resultado al llamador.
return Tarifa * Horas + Bonificación + Horas extra;
/end-free
P CalcPay          E

```

Figura 4. Ejemplo de un programa de cálculo de nóminas (Pieza 3 de 3)

Utilización del sistema OS/400

El sistema operativo que controla todas las interacciones del usuario con el sistema iSeries se denomina Operating System/400 (OS/400). Desde la estación de trabajo, el sistema OS/400 le permite:

- Iniciar y finalizar una sesión
- Operar de forma interactiva con las pantallas
- Utilizar la información de ayuda en línea
- Entrar mandatos de control y procedimientos
- Responder a mensajes
- Gestionar archivos
- Ejecutar programas de utilidad y otros programas

Si tiene acceso a Internet, puede obtener una lista completa de las publicaciones que versan sobre el sistema OS/400 en el URL siguiente:

<http://publib.boulder.ibm.com/>

También puede solicitar la publicación *AS/400 V4 System Library Poster*, G325-6334-02.

Interacción con el sistema

Puede trabajar con el sistema OS/400 utilizando el lenguaje de control (CL). Puede actuar de forma interactiva con el sistema entrando o seleccionando mandatos CL. El sistema a menudo visualiza una serie de mandatos CL o parámetros de mandato adecuados para la situación de la pantalla. Entonces puede seleccionar el mandato o los parámetros que desee.

Mandatos del lenguaje de control utilizados con frecuencia

La tabla siguiente lista algunos de los mandatos CL más utilizados y los motivos por los que podría utilizarlos.

Utilización del sistema OS/400

Tabla 3. Mandatos CL utilizados con frecuencia

Acción	Mandato CL	Resultado
Utilizar menús del sistema	GO MAIN	Visualiza menú principal
	GO INFO	Visualiza menú de ayuda
	GO CMDRPG	Lista los mandatos para RPG
	GO CMDCRT	Lista los mandatos para crear
	GO CMDxxx	Lista los mandatos para 'xxx'
Llamar	CALL <i>programa</i>	Ejecuta un programa
Compilar	CRTxxxMOD	Crea un módulo xxx
	CRTBNDxxx	Crea un programa enlazado xxx
Enlazar	CRTPGM	Crea un programa a partir de módulos ILE
	CRTSRVPGM	Crea un programa de servicio
	UPDPGM	Actualiza un objeto de programa enlazado
Depurar	STRDBG	Arranca el depurador del fuente ILE
	ENDDBG	Finaliza el depurador del fuente ILE
Crear archivos	CRTPRTF	Crea archivo de impresión
	CRTPF	Crea archivo físico
	CRTSRCPF	Crea archivo físico fuente
	CRTL	Crea archivo lógico

WebSphere Development Studio para iSeries

WebSphere Development Studio para iSeries ofrece un completo conjunto de compiladores y herramientas para el desarrollo de aplicaciones para dar respuesta a sus necesidades de programación.

Application Development ToolSet

Application Development ToolSet (ADTS) proporciona un conjunto integrado de herramientas basadas en sistema principal diseñadas para satisfacer las necesidades del desarrollador de aplicaciones. Este conjunto de herramientas proporciona herramientas para trabajar con fuentes, objetos y archivos de base de datos de OS/400. Algunas de las herramientas que se proporcionan son: Gestor para el desarrollo de programación (PDM), Programa de utilidad para entrada del fuente (SEU) y Ayuda para el diseño de pantallas (SDA). Tiene a su disposición una interfaz dirigida por menús, desde la que puede realizar todas las tareas necesarias para el desarrollo de aplicaciones, como por ejemplo gestión de objetos, edición, compilación y depuración.

Application Development Manager

Application Development Manager proporciona a las organizaciones de desarrollo de aplicaciones un mecanismo para una gestión eficaz y efectiva de los objetos de

la aplicación durante la vida de la aplicación. Esta característica de ADTS permite que un grupo de desarrolladores creen, gestionen y organicen múltiples versiones de su aplicación a través de la interfaz Gestor para el desarrollo de programación (PDM) o directamente desde la línea de mandatos.

Un equipo de desarrollo de aplicaciones que utilice ADM puede:

- Definir un entorno flexible donde la producción, la prueba y el mantenimiento se puedan gestionar de modo simultáneo.
- Organizar el trabajo de varios desarrolladores en la misma aplicación.
- Crear (o compilar) una aplicación de modo fácil y sencillo, compilando únicamente aquellos componentes que sea necesario compilar.
- Crear y mantener varias versiones de una aplicación.

Application Dictionary Services

Application Dictionary Services es una herramienta de análisis de incidencia que acelera el análisis de las aplicaciones. La herramienta almacena la información sobre los objetos de aplicación y su relación con otros objetos de aplicación en un **diccionario**. Un diccionario extrae la información de referencias cruzadas sobre todos los objetos de una biblioteca de la aplicación y los salva en un conjunto de archivos de base de datos almacenados en una biblioteca. A medida que cambian los objetos de la aplicación y su relación con otros objetos, la información del diccionario se actualiza automáticamente.

Puede utilizar esta característica de ADTS para:

- Determinar la incidencia del cambio de un campo
- Trabajar con campos, archivos o programas que resultarían afectados por el cambio de un campo
- Volver a crear todos los objetos afectados por el cambio de un campo
- Ver la estructura de una aplicación
- Determinar la jerarquía de referencias de campo
- Crear, modificar o suprimir programas o archivos que están documentados en un diccionario
- Modificar los campos o registros que están documentados en un diccionario
- Examinar la jerarquía de llamadas

WebSphere Development Tools for iSeries

WebSphere Development Tools for iSeries es un producto de estación de trabajo (Windows) que incluye dos programas de acceso al servidor:

- CoOperative Development Environment/400 (CODE/400)
- VisualAge RPG
- WebSphere Studio
- VisualAge para Java

CODE/400 contiene diversas características para ayudarle a editar, compilar y depurar programas fuente de sistema principal RPG, ILE RPG, COBOL, ILE COBOL, CL (Lenguaje de control), ILE C/C++ e ILE CL; a diseñar archivos de sistema principal de pantalla, impresora y base de datos, y a gestionar los componentes que integran una aplicación. Esto mejora el desarrollo de programas y hace que la carga de trabajo del desarrollo de programas se desplace fuera del

WebSphere Development Studio para iSeries

sistema principal. La aplicación, una vez creada, se ejecuta en un servidor iSeries. Para el desarrollo y el mantenimiento de aplicaciones RPG e ILE RPG, CODE/400 proporciona:

- Edición sensible al lenguaje: incluye resaltado de señales, líneas de formato, una serie completa de solicitudes y ayuda en línea.
- Comprobación de sintaxis incremental: proporciona información de retorno sobre errores inmediata a medida que se entra cada línea del fuente.
- Verificación de programas: realiza en la estación de trabajo toda la comprobación sintáctica y semántica que lleva a cabo el compilador, sin generar el código objeto.
- Conversión de programas: lleva a cabo en la estación de trabajo una conversión de OPM a ILE RPG.
- Un entorno de ventanas para someter las compilaciones y los enlaces del sistema principal.
- Depuración a nivel de fuente.
- Un programa de utilidad de diseño de DDS: permite modificar con facilidad pantallas, informes y archivos de bases de datos.
- Acceso a Application Dictionary Services.

VisualAge RPG ofrece un entorno de desarrollo visual en la plataforma de la estación de trabajo que permite a los desarrolladores de aplicaciones RPG desarrollar, mantener y documentar aplicaciones de cliente/servidor. Las aplicaciones pueden editarse, compilarse y depurarse en la estación de trabajo. Las aplicaciones, una vez creadas, se arrancan en una estación de trabajo y pueden acceder a los datos de sistema principal del iSeries y a otros objetos del iSeries. Sus componentes integrados permiten a los desarrolladores de aplicaciones conservar sus posibilidades actuales y desarrollar fácilmente aplicaciones RPG de iSeries con interfaces gráficas de usuario.

IBM WebSphere Studio es el mejor entorno que hay disponible para los equipos de desarrollo en la Web a fin de organizar y gestionar los proyectos de este tipo. Studio ofrece el producto de desarrollo en la Web más completo que existe hoy en día con vistas al desarrollo y a la publicación de sitios Web interactivos para IBM WebSphere Application Server. El soporte incorporado para dispositivos inalámbricos permite crear fácilmente páginas WML, Compact HTML y Voice (VXML). Los asistentes de Studio permiten importar de forma sencilla un sitio Web HTML existente y empezar a añadir JavaServer Pages (JSP) dirigidas por datos. Studio ofrece todo lo necesario para dar soporte a las necesidades de desarrollo de aplicaciones Web, desde el desarrollo de sitios Web de extremo a extremo hasta la depuración de la lógica del lado del servidor remota o el despliegue de las aplicaciones Web.

VisualAge para Java es un entorno visual integrado que proporciona soporte para el ciclo completo del desarrollo de programas Java. Puede utilizar las características de programación visual de VisualAge para Java a fin de desarrollar rápidamente applets y aplicaciones Java. En el editor de composición visual, basta con apuntar y pulsar para diseñar la interfaz de usuario del programa, especificar el funcionamiento de los elementos de la interfaz de usuario y definir la relación entre la interfaz de usuario y el resto del programa. VisualAge para Java también le ofrece SmartGuides (asistentes) para guiarle de forma rápida en numerosas tareas como, por ejemplo, crear nuevos applets, paquetes o clases. El entorno de desarrollo integrado (IDE) de VisualAge para Java automáticamente compila el código fuente de Java en bytecode de Java. Cuando el código fuente se importa en

WebSphere Development Studio para iSeries

el espacio de trabajo (desde archivos .java) o se añade desde el depósito, se compila y analiza en relación con el contenido existente del espacio de trabajo. Utilice VisualAge para Java para:

- Crear, modificar y utilizar JavaBeans
- Examinar el código al nivel de proyecto, paquete, clase o método
- Utilizar el depurador visual integrado para examinar y actualizar código mientras se ejecuta
- Utilizar el depurador distribuido para depurar aplicaciones Java desarrolladas fuera del IDE

Si desea aprender más acerca de WebSphere Development Tools for iSeries, consulte la información más actualizada en la dirección de Internet ibm.com/software/ad/wdt400/.

Capítulo 2. Programación de RPG en ILE

ILE RPG es una implementación del lenguaje de programación RPG IV de Integrated Language Environment. Forma parte de la familia de compiladores de ILE disponibles en el sistema iSeries.

ILE es un método reciente de programación en el sistema iSeries. Es el resultado de mejoras importantes en la arquitectura del iSeries y en el sistema operativo OS/400. La familia de compiladores ILE incluye: ILE RPG, ILE C, ILE COBOL, ILE CL y VisualAge for C++. La Tabla 4 muestra los lenguajes de programación soportados por el sistema operativo OS/400. Además del soporte para los lenguajes ILE, se ha mantenido el soporte para los lenguajes del modelo del programa original (OPM) y del modelo del programa ampliado (EPM).

Tabla 4. Lenguajes de programación soportados en iSeries

Integrated Language Environment (ILE)	Modelo del programa original (OPM)	Modelo del programa ampliado (EPM)
C++	BASIC (PRPQ)	FORTRAN
C	CL	PASCAL (PRPQ)
CL	COBOL	
COBOL	PL/I (PRPQ)	
RPG	RPG	

Comparado con OPM, ILE proporciona a los usuarios de RPG mejoras y ampliaciones en las siguientes áreas de desarrollo de aplicaciones:

- Creación de programas
- Gestión de programas
- Llamada a programas
- Depuración del fuente
- Interfaces de programación de aplicaciones (API) enlazables

Cada una de las áreas anteriores se describe brevemente en los párrafos siguientes y se trata con más detalle en los capítulos que siguen.

Creación de programas

En ILE, la creación de programas consiste en:

1. Compilación del código fuente en módulos
2. Enlace (combinación) de uno o más módulos en un objeto de programa

Puede crear un objeto de programa de modo muy similar a como lo haría con la estructura del OPM, con un proceso de un solo paso mediante el mandato Crear programa en RPG enlazado (CRTBNDRPG). Este mandato crea un módulo temporal que se enlaza a continuación con un objeto de programa. También le permite enlazar otros objetos mediante la utilización de un directorio de enlace.

De forma alternativa, puede crear un programa utilizando mandatos separados para la compilación y para el enlace. Este proceso de dos pasos le permite volver a utilizar un módulo o actualizar un módulo sin volver a compilar los otros módulos

Programación de RPG en ILE

de un programa. Además, como puede combinar módulos a partir de cualquier lenguaje ILE, puede crear y mantener programas en lenguajes mixtos.

En el proceso de dos pasos, debe crear un objeto de módulo utilizando el mandato Crear módulo en RPG (CRTRPGMOD). Este mandato compila las sentencias fuente en un objeto de módulo. Un módulo es un objeto no ejecutable; debe estar enlazado dentro de un objeto de programa para que se pueda ejecutar. Para enlazar uno o más módulos utilice el mandato Crear programa (CRTPGM).

Los programas de servicio son un medio de empaquetar los procedimientos de uno o más módulos en un objeto enlazado independientemente. Otros programas ILE pueden acceder a los procedimientos del programa de servicio, aunque sólo exista una copia del programa de servicio en el sistema. La utilización de programas de servicio proporciona modularidad y facilidad de mantenimiento. Puede utilizar programas de servicio desarrollados por terceros o, a la inversa, empaquetar sus propios programas de servicio para que los utilicen terceros. Un programa de servicio se crea mediante el mandato Crear programa de servicio (CRTSRVPGM).

Puede crear un directorio de enlace que contenga los nombres de los módulos y programas de servicio que puede necesitar su programa o programa de servicio. Puede especificarse una lista de directorios de enlace al crear un programa en los mandatos CRTBNDRPG, CRTSRVPGM y CRTPGM. También puede especificarse en el mandato CRTRPGMOD; sin embargo, la búsqueda de un directorio de enlace se efectúa cuando el módulo se enlaza al ejecutarse CRTPGM o CRTSRVPGM. Un directorio de enlace puede reducir el tamaño del programa ya que los módulos o programas de servicio listados en un directorio de enlace únicamente se utilizan si son necesarios.

La Figura 5 muestra los dos procesos de creación de programas.

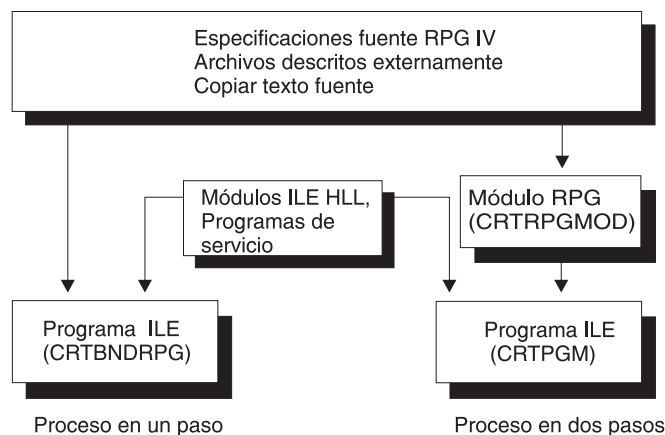


Figura 5. Creación de programas en ILE

Una vez que haya creado un programa, puede actualizarlo utilizando los mandatos Actualizar programa (UPDPGM) o Actualizar programa de servicio (UPDSRVPGM). Esto es útil, porque significa que tan sólo es necesario tener disponibles los objetos de módulo nuevos o modificados para actualizar el programa.

Para obtener más información acerca del proceso de un solo paso, consulte el "Capítulo 6. Creación de un programa con el mandato CRTBNDRPG" en la página 61

página 61. Para obtener más información acerca del proceso de dos pasos, consulte el “Capítulo 7. Creación de un programa con los mandatos CRTRPGMOD y CRTPGM” en la página 77. Para obtener más información sobre los programas de servicio, consulte el apartado “Capítulo 8. Creación de un programa de servicio” en la página 93.

Gestión de programas

ILE proporciona una base común para:

- Gestionar el flujo de programas
- Compartir recursos
- Utilizar interfaces de programación de aplicaciones (API)
- Manejar excepciones durante el tiempo de ejecución de un programa

Ofrece a los usuarios de RPG un control sobre los recursos mucho mejor del que era posible anteriormente.

Los programas y programas de servicio ILE se activan en grupos de activación que se especifican en el momento de creación del programa. El proceso mediante el cual un programa o un programa de servicio se prepara para ejecutarse se denomina activación. La activación asigna recursos en un trabajo, de modo que puedan ejecutarse uno o más programas en dicho espacio. Si el grupo de activación especificado para un programa no existe cuando se llama al programa, entonces se crea dentro del trabajo para dar cabida a la activación del programa.

Un grupo de activación es el elemento clave que rige los recursos y el comportamiento de una aplicación ILE. Por ejemplo, puede definir el ámbito de las operaciones de control de compromiso en el nivel del grupo de activación. También puede definir el ámbito de las alteraciones temporales de archivos y las vías de datos abiertas y compartidas en el grupo de activación de la aplicación actual. Por último, el comportamiento de un programa hasta su terminación también se ve afectado por el grupo de activación en el que se ejecuta el programa.

Para obtener más información sobre los grupos de activación, consulte “Gestión de los grupos de activación” en la página 112.

Utilizando las API enlazables que se proporcionan para todos los lenguajes de programación ILE, puede asignar dinámicamente almacenamiento para una matriz de tiempo de ejecución. Estas API permiten que aplicaciones de lenguaje único y de lenguaje mixto accedan a un conjunto central de funciones de gestión de almacenamiento y ofrecen un modelo de almacenamiento a los lenguajes que no proporcionan ninguno. RPG ofrece algunas posibilidades de gestión de almacenamiento mediante códigos de operación. Para obtener más información sobre la gestión de almacenamiento, consulte “Gestión del almacenamiento asignado dinámicamente” en la página 115.

Llamada a programas

En ILE, puede escribir aplicaciones en las que los programas ILE RPG y los programas OPM RPG/400 continúen interrelacionándose mediante la utilización tradicional de las llamadas dinámicas a programas. Al utilizar dichas llamadas, el programa que efectúa la llamada (programa de llamada) especifica el nombre del programa llamado en una sentencia de llamada. El nombre del programa llamado se resuelve en una dirección durante la ejecución, justo antes de que el programa de llamada transfiera el control al programa llamado.

Programación de RPG en ILE

También puede escribir aplicaciones ILE que se interrelacionen con llamadas estáticas más rápidas. Las llamadas estáticas implican llamadas entre procedimientos. Un procedimiento es un juego de códigos independiente que realiza una tarea y a continuación regresa al llamador. Un módulo ILE RPG consta de un procedimiento principal opcional seguido de cero o más subprocedimientos. Dado que los nombres de procedimientos se resuelven durante el enlace (es decir, al crear el programa) las llamadas estáticas son más rápidas que las llamadas dinámicas.

Las llamadas estáticas también permiten:

- Descriptores operativos
- Parámetros omitidos
- Especificar parámetros según su valor
- La utilización de valores de retorno
- Especificar un mayor número de parámetros

Los descriptores operativos y los parámetros omitidos pueden ser de utilidad al llamar a interfaces API enlazables o a procedimientos escritos en otros lenguajes ILE.

Para obtener información sobre cómo ejecutar un programa, consulte el “Capítulo 9. Ejecución de un programa” en la página 105. Si desea información sobre las llamadas a programas y procedimientos, consulte el “Capítulo 10. Llamadas a programas y procedimientos” en la página 131.

Depuración del fuente

En ILE puede realizar la depuración a nivel del fuente en cualquier aplicación ILE de lenguaje único o mixto. El depurador del fuente ILE también proporciona soporte para programas OPM. Puede controlar el flujo de un programa utilizando los mandatos de depuración mientras se ejecuta el programa. Puede establecer puntos de interrupción de trabajo o de hebra condicionales e incondicionales antes de ejecutar el programa. A continuación, después de llamar al programa, puede desplazarse por un número determinado de sentencias y visualizar o modificar las variables. Cuando se detiene un programa debido a un punto de interrupción, a un mandato de pasos o a un error de ejecución, el módulo correspondiente aparece en la pantalla en el punto en el que se ha detenido el programa. En ese punto, puede entrar más mandatos de depuración.

Para obtener información sobre el depurador, consulte el “Capítulo 12. Depuración de programas” en la página 195.

API enlazables

ILE ofrece una serie de interfaces API enlazables que se pueden utilizar como complemento de la función que ILE RPG ofrece actualmente. Las API enlazables proporcionan posibilidades de llamada a programas y activación, gestión de condiciones y almacenamiento, funciones matemáticas y gestión dinámica de pantallas.

Algunas de las API que puede plantearse utilizar en una aplicación ILE RPG son:

- CEETREC: señalización de la condición de terminación inminente
- CEE4ABN: terminación anormal
- CEECRHP: creación del área de almacenamiento dinámico propia

- CEEDSHP: descarte del área de almacenamiento dinámico propia
- CEEFRST: liberación del almacenamiento de área de almacenamiento dinámico propia
- CEEGTST: obtención del almacenamiento de área de almacenamiento dinámico propia
- CEECZST: reasignación del almacenamiento de área de almacenamiento dinámico propia
- CEEDOD: descomposición del descriptor operativo

Nota: no se puede utilizar estas ni ninguna otra API enlazable de ILE desde un programa creado con DFTACTGRP(*YES). Esto es debido a que en este tipo de programa no se permiten las llamadas enlazadas.

Para obtener más información sobre las API enlazables de ILE, consulte el “Capítulo 9. Ejecución de un programa” en la página 105.

Aplicaciones multihebra

El sistema iSeries ahora proporciona soporte para la ejecución multihebra. ILE RPG no proporciona directamente soporte para iniciar o gestionar hebras de programa. No obstante, los procedimientos ILE RPG pueden ejecutarse como hebras en entornos multihebra. Si desea llamar a un procedimiento ILE RPG en una aplicación multihebra, debe asegurarse de que el procedimiento ILE RPG permita la ejecución multihebra. Asimismo, debe cerciorarse de que las funciones del sistema a las que acceda el procedimiento también permitan la ejecución multihebra.

Puede especificarse la palabra clave de especificación de control THREAD(*SERIALIZE) para proteger las hebras en un módulo ILE RPG. Al especificar THREAD(*SERIALIZE) se impide que varias hebras puedan acceder de forma inadecuada a la mayoría de las variables y a todas las estructuras de control internas. El módulo de protección de hebras se bloqueará cuando se acceda a un procedimiento del módulo y se desbloqueará cuando ya no haya ningún procedimiento del módulo en ejecución. Este acceso en serie garantiza que sólo haya una hebra activa en un módulo cualquiera, dentro de un grupo de activación, en todo momento. Sin embargo, el programador sigue siendo el encargado de manejar la seguridad de las hebras en el almacenamiento compartido por varios módulos. Esto se consigue añadiendo lógica a la aplicación para sincronizar el acceso al almacenamiento.

Para obtener más información, consulte el apartado “Consideraciones sobre las hebras múltiples” en la página 162.

Capítulo 3. Estrategias de creación de programas

Hay muchos enfoques que puede adoptar al crear programas utilizando un lenguaje ILE. Esta sección presenta tres estrategias corrientes para crear programas ILE utilizando ILE RPG u otros lenguajes ILE.

1. Creación de un programa utilizando CRTBNDRPG para maximizar la compatibilidad con OPM.
2. Creación de un programa ILE utilizando CRTBNDRPG.
3. Creación de un programa ILE utilizando CRTRPGMOD y CRTPGM.

La primera estrategia se recomienda de forma temporal. Está pensada para usuarios que tengan aplicaciones OPM y que, quizá por falta de tiempo, no puedan trasladar sus aplicaciones a ILE todas a la vez. La segunda estrategia también puede ser temporal. Le da tiempo para poder aprender más acerca de ILE, pero también le permite utilizar de inmediato algunas de sus funciones. La tercera estrategia es más complicada, pero es la que ofrece más flexibilidad.

Tanto la primera como la segunda estrategia utilizan el proceso de creación de programas de un solo paso, es decir, CRTBNDRPG. La tercera estrategia utiliza el proceso de creación de programas en dos pasos, es decir, CRTRPGMOD y a continuación CRTPGM.

Estrategia 1: aplicación compatible con OPM

La estrategia 1 produce un programa ILE que interactúa bien con los programas OPM. Le permite beneficiarse de las mejoras de RPG IV pero no de todas las mejoras de ILE. Puede que desee utilizar este programa de forma temporal mientras completa la migración a ILE.

Método

Utilice el siguiente enfoque general para crear un programa de este tipo:

1. Convierta el fuente a RPG IV utilizando el mandato CVTRPGSRC.
Asegúrese de que convierte todos los miembros /COPY que utilice el fuente que está convirtiendo.
2. Cree un objeto de programa utilizando el mandato CRTBNDRPG, especificando DFTACTGRP(*YES).

Si especifica DFTACTGRP(*YES) significa que el objeto de programa sólo se ejecutará en el grupo de activación por omisión. (El grupo de activación por omisión es el grupo de activación en el que se ejecutan todos los programas OPM.) En consecuencia, el objeto de programa interactuará bien con los programas OPM en las áreas de ámbito de alteración temporal, de ámbito de apertura y de RCLRSC.

Cuando utiliza este enfoque, no puede utilizar el enlace estático de ILE. Esto significa que al crear este programa no puede codificar una llamada de procedimiento enlazada en el fuente ni utilizar los parámetros BNDDIR o ACTGRP en el mandato CRTBNDRPG.

Ejemplo de un programa compatible con OPM

La Figura 6 muestra la vista de ejecución de una aplicación a modo de ejemplo en la que podría desear un programa compatible con OPM. La aplicación OPM estaba formada por un programa en CL y dos programas en RPG. En este ejemplo, uno de los programas en RPG se ha trasladado a ILE; los demás programas no se han modificado.

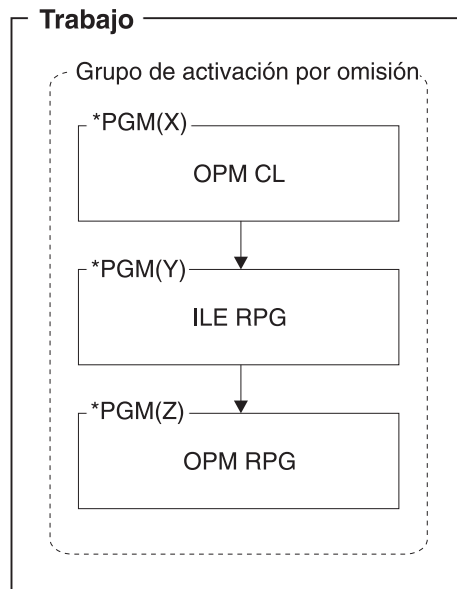


Figura 6. Aplicación compatible con OPM

Efecto de ILE

A continuación se explica el efecto de ILE en el manejo de la aplicación:

Llamada a programas

Los programas OPM se comportan igual que antes. El sistema crea de forma automática el grupo de activación OPM por omisión cuando arranca el trabajo, y se ejecutan en el mismo todas las aplicaciones OPM. Un programa puede llamar a otro programa del grupo de activación por omisión utilizando una llamada dinámica.

Datos

Se crea el almacenamiento para datos estáticos cuando se activa el programa, que existirá hasta que se desactive el programa. Cuando finalice el programa (de manera normal o anormal), se suprimirá el almacenamiento del programa. Para eliminar el almacenamiento de un programa que retorna sin finalizar, utilice el mandato Reclamar recurso (RCLRSC).

Archivos

El proceso de archivos es igual que en los releases anteriores. Los archivos se cierran cuando finaliza el programa de forma normal o anormal.

Errores

Como en los releases anteriores, el compilador maneja los errores de cada programa por separado. Los errores que observe que se hayan originado en el programa son los mismos que antes. Sin embargo, ahora los errores se comunican entre programas mediante el gestor de condiciones de ILE, de forma que es posible que observe mensajes distintos entre los programas. Los mensajes

pueden tener nuevos ID de mensaje, de modo que si el programa CL supervisa un ID de mensaje determinado, deberá modificar este ID.

Información relacionada

Conversión a RPG IV	“Conversión del código fuente” en la página 432
Proceso de creación de un solo paso	“Capítulo 6. Creación de un programa con el mandato CRTBNDRPG” en la página 61
Enlace estático de ILE	“Capítulo 10. Llamadas a programas y procedimientos” en la página 131; también <i>ILE Concepts</i>
Diferencias de manejo de excepciones	“Diferencias entre el manejo de excepciones de OPM e ILE RPG” en la página 256

Estrategia 2: programa ILE utilizando CRTBNDRPG

La estrategia 2 tiene como resultado un programa ILE que puede beneficiarse del enlace estático de ILE. El fuente puede contener llamadas a procedimientos estáticos ya que el usuario puede enlazar el módulo a otros módulos o programas de servicio que utilicen un directorio de enlace. También puede especificar el grupo de activación en el que se ejecutará el programa.

Método

Utilice el siguiente método general para crear un programa de este tipo:

1. Si empieza con un fuente RPG III, convierta el fuente a RPG IV utilizando el mandato CVTRPGSRC.
Si lo convierte, asegúrese de que convierte todos los miembros /COPY y los programas a los que llama el fuente que está convirtiendo. Además, si utiliza CL para llamar al programa, también debe asegurarse de utilizar CL ILE en lugar de CL OPM.
2. Determine el grupo de activación en el que se ejecutará el programa.
Es posible que desee darle el mismo nombre que el de la aplicación, como en este ejemplo.
3. Identifique los nombres de los directorios de enlace que se van a utilizar, si existen.
Con este método, se asume que si utiliza un directorio de enlace, éste será uno que ya ha creado. Por ejemplo, puede que haya un programa de servicio de terceros que desee enlazar a su fuente. Por consiguiente, todo lo que necesita saber es el nombre del directorio de enlace.
4. Cree un programa ILE mediante CRTBNDRPG, especificando DFTACTGRP(*NO), el grupo de activación en el parámetro ACTGRP y el directorio de enlace, si existe, en el parámetro BNDDIR.

Observe que si se especifica ACTGRP(*CALLER) y a este programa le llama un programa que se ejecuta en el grupo de activación por omisión, este programa se comportará según la semántica de ILE en las áreas de ámbito de alteración temporal, ámbito de apertura y RCLRSC.

Programa ILE utilizando CRTBNDRPG

El principal inconveniente de esta estrategia es que carece de un módulo de objeto permanente que se pueda volver a utilizar más tarde para enlazarlo con otros módulos y crear un programa ILE. Además, las llamadas a procedimientos deben realizarse a módulos o programas de servicio que estén identificados en un directorio de enlace. Si desea enlazar dos o más módulos sin utilizar un directorio de enlace al crear el programa, debe utilizar la tercera estrategia.

Ejemplo de un programa ILE utilizando CRTBNDRPG

La Figura 7 muestra la vista de ejecución de una aplicación en la que un programa CL ILE llama a un programa ILE RPG que está enlazado a un programa de servicio suministrado. La aplicación se ejecuta en el grupo de activación denominado XYZ.

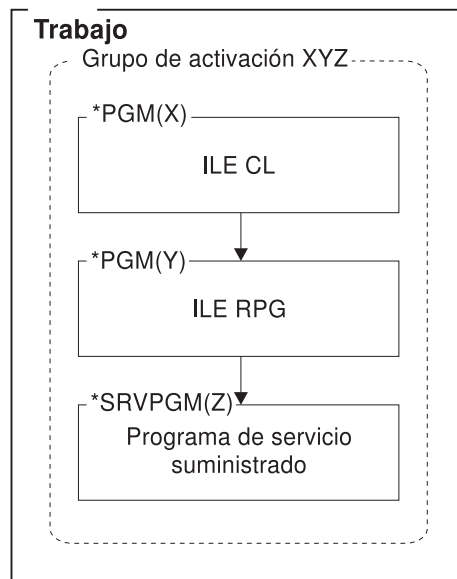


Figura 7. Programa ILE utilizando CRTBNDRPG

Efecto de ILE

A continuación se explica el efecto de ILE en el manejo del programa:

Llamada a programas

El sistema crea de forma automática el grupo de activación, si éste todavía no existe, al arrancar la aplicación.

La aplicación puede contener llamadas a programas dinámicas o llamadas a procedimientos estáticas. Los procedimientos de los programas enlazados se llaman entre sí utilizando llamadas estáticas. Los procedimientos llaman a los programas ILE y OPM utilizando llamadas dinámicas.

Datos

La duración del almacenamiento de un programa es igual que la del grupo de activación. El almacenamiento permanecerá activo hasta que se suprima el grupo de activación.

La ejecución de ILE RPG gestiona los datos de forma que la semántica de finalización de programas y reinicialización de datos sea la misma que para OPM RPG, aunque el almacenamiento real no se suprime como ocurría cuando finalizaba un programa OPM

Programa ILE utilizando CRTBNDRPG

RPG. Los datos se reinician si la anterior llamada al procedimiento terminó con el indicador LR activado, o terminó de forma anormal.

Los datos del programa que se identifican como exportados o importados (utilizando las palabras clave EXPORT e IMPORT respectivamente) son externos a los módulos individuales. Los módulos que están enlazados en un programa tienen conocimiento de ello.

Archivos Por omisión, el proceso de archivos (incluyendo la apertura, compartimiento, alteración temporal y control de compromiso) por el sistema se concentra dentro del ámbito del grupo de activación. No se pueden compartir archivos a nivel de gestión de datos con programas de grupos de activación distintos. Si desea compartir un archivo con otros grupos de activación, debe abrirlo a nivel del trabajo especificando SHARE(*YES) en un mandato de alteración temporal o crear el archivo con SHARE(*YES).

Errores Cuando llama a un programa ILE RPG o a un procedimiento del mismo grupo de activación, si éste obtiene una excepción que anteriormente hubiera dado como resultado la visualización de un mensaje de consulta, ahora será el programa de llamada el que verá esta excepción primero.

Si el programa de llamada tiene un indicador de error o *PSSR, el programa o procedimiento que obtuvo la excepción terminará de forma anormal sin que se visualice el mensaje de consulta. El programa de llamada se comportará igual (se activará el indicador de error o se invocará a *PSSR).

Cuando llama a un programa OPM, a un programa o a un procedimiento principal de un grupo de activación diferente, el manejo de excepciones será el mismo que en OPM RPG, es decir, cada programa maneja sus propias excepciones. Los mensajes que vea pueden tener nuevos ID de mensaje, por lo que si durante la supervisión busca un ID de mensaje en concreto puede que tenga que modificar este ID.

Cada lenguaje procesa sus propios errores y puede procesar los errores que se producen en los módulos escritos en otro lenguaje ILE. Por ejemplo, RPG manejará los errores de C si se ha codificado un indicador de error. C puede manejar cualquier error de RPG.

Información relacionada

Conversión a RPG IV	“Conversión del código fuente” en la página 432
Proceso de creación de un solo paso	“Capítulo 6. Creación de un programa con el mandato CRTBNDRPG” en la página 61
Grupos de activación	“Gestión de los grupos de activación” en la página 112
RCLRSC	“Mandato Reclamar Recursos” en la página 114
Enlace estático de ILE	“Capítulo 10. Llamadas a programas y procedimientos” en la página 131; también la publicación <i>ILE Concepts</i>

Programa ILE utilizando CRTBNDRPG

Diferencias de manejo de excepciones

“Diferencias entre el manejo de excepciones de OPM e ILE RPG” en la página 256

Ámbito de alteración temporal y de apertura

“Alteración temporal y redirección de la entrada y salida de archivos” en la página 313 y

“Compartimiento de una vía de acceso de datos abierta” en la página 317; también la publicación *ILE Concepts*

Estrategia 3: aplicación ILE utilizando CRTRPGMOD

Esta estrategia le permite utilizar en su totalidad los conceptos que ILE ofrece. Sin embargo, a pesar de ser el método más flexible, también es el más complicado. Esta sección presenta tres marcos hipotéticos de creación:

- Una aplicación en un solo lenguaje
- Una aplicación en lenguaje mixto
- Una aplicación avanzada

El efecto de ILE es el mismo que se ha descrito en el apartado “Efecto de ILE” en la página 28.

Puede que desee leer acerca de los conceptos básicos de ILE en la publicación *ILE Concepts* antes de utilizar este método.

Método

Como este enfoque es el más flexible, incluye varios métodos para crear una aplicación ILE. La lista siguiente describe los pasos principales que puede que necesite realizar:

1. Cree un módulo a partir de cada miembro fuente utilizando el mandato apropiado, por ejemplo, CRTRPGMOD para el fuente RPG, CRTCLMOD para el fuente CL, etc.
2. Determine las características de ILE para la aplicación, como por ejemplo:
 - Determine qué módulo contendrá el procedimiento que será el punto de arranque de la aplicación. El módulo elegido como módulo de entrada es el primero que desea que obtenga el control. En una aplicación OPM, se trataría del programa de proceso de mandatos, o el programa que se llama cuando se selecciona un elemento de menú.
 - Determine el grupo de activación en el que se ejecutará la aplicación. (Probablemente deseará ejecutarla en un grupo de activación con nombre, donde el nombre se basa en el nombre de la aplicación.)
 - Determine las exportaciones e importaciones que se utilizarán.
3. Determine si algunos de los módulos se enlazarán entre sí para crear un programa de servicio. En caso afirmativo, cree los programas de servicio utilizando CRTSRVPGM.
4. Identifique los nombres de los directorios de enlace que se van a utilizar, si existen.

Con este método, se asume que si utiliza un directorio de enlace, éste es uno que ya ha creado. Por ejemplo, puede que haya un programa de servicio de terceros que desee enlazar a su fuente. Por consiguiente, todo lo que necesita saber es el nombre del directorio de enlace.

5. Enlace entre sí los módulos y programas de servicio apropiados utilizando CRTPGM, especificando valores para los parámetros basados en las características determinadas en el paso 2 en la página 30.

Una aplicación que se haya creado utilizando este enfoque puede ejecutarse con protección completa, es decir, dentro de su propio grupo de activación. Además, se puede actualizar fácilmente utilizando los mandatos UPDPGM o UPDSRVPGM. Con estos mandatos puede añadir o sustituir uno o más módulos sin tener que volver a crear el objeto de programa.

Marco hipotético de una aplicación ILE en un solo lenguaje

En este marco hipotético se compilan múltiples archivos fuente en módulos y se enlazan en un programa al que llama un programa ILE RPG. La Figura 8 muestra la vista de ejecución de esta aplicación.

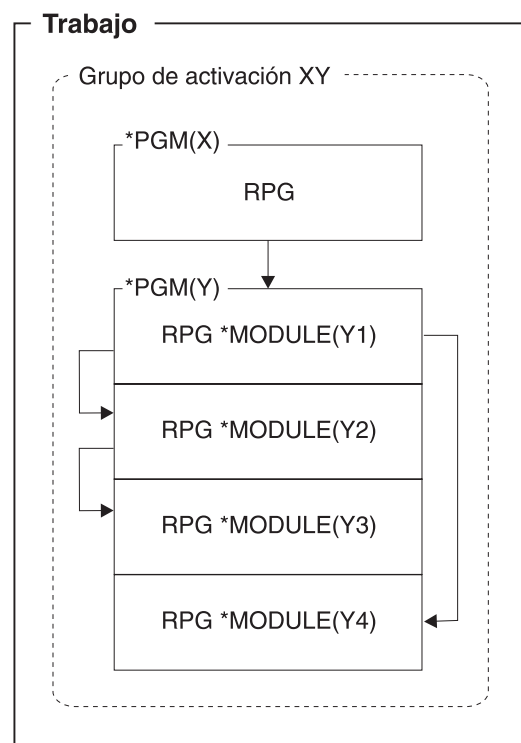


Figura 8. Aplicación en un solo lenguaje utilizando CRTRPGMOD y CRTPGM

La llamada del programa X al programa Y es una llamada dinámica. Las llamadas entre los módulos del programa Y son llamadas estáticas.

Consulte el apartado “Efecto de ILE” en la página 28 para ver detalles sobre los efectos de ILE en la manera en que la aplicación maneja llamadas, datos, archivos y errores.

Marco hipotético de una aplicación ILE en lenguaje mixto

En este marco hipotético, usted puede crear aplicaciones integradas en lenguaje mixto. El módulo principal, escrito en un lenguaje ILE llama a los procedimientos escritos en otro lenguaje ILE. El módulo principal abre los archivos que los otros módulos comparten a continuación. Debido a la utilización de lenguajes distintos, es posible que no se espere un comportamiento coherente. Sin embargo, ILE se asegura de que se produzca.

Aplicación ILE utilizando CRTRPGMOD

La Figura 9 muestra la vista en ejecución de una aplicación que contiene un programa ILE en lenguaje mixto donde un módulo llama a una API no enlazable, QUSCRTUS (Crear espacio de usuario).

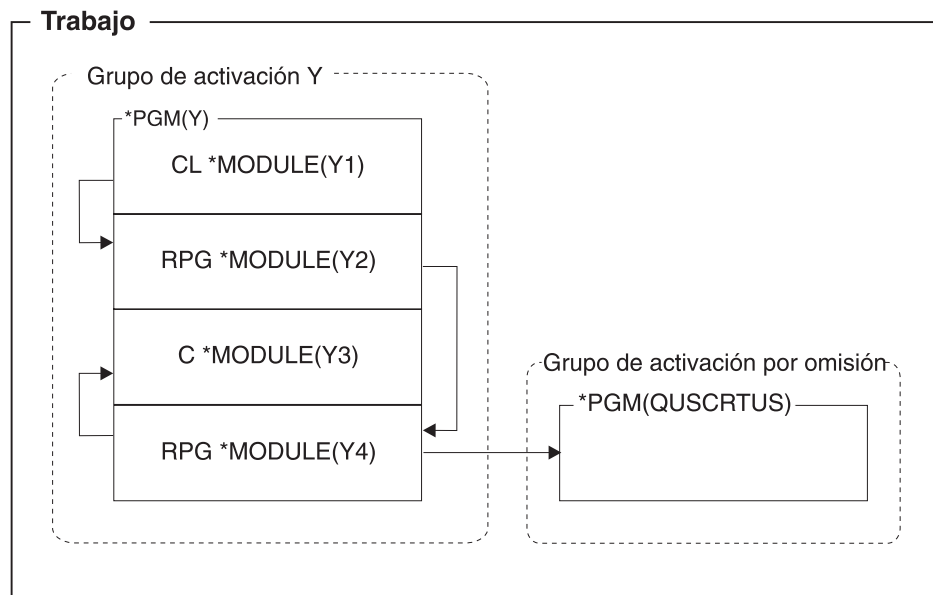


Figura 9. Aplicación en lenguaje mixto

La llamada del programa Y a la API OPM es una llamada dinámica. Las llamadas entre los módulos del programa Y son llamadas estáticas.

Consulte el apartado “Efecto de ILE” en la página 28 para ver detalles sobre los efectos de ILE en la manera en que la aplicación maneja llamadas, datos, archivos y errores.

Marco hipotético de una aplicación avanzada

En este marco hipotético, se beneficia de todas las ventajas de la función ILE, incluidos los programas de servicio. La utilización de llamadas enlazadas en procedimientos de módulos y programas de servicio, proporciona un mejor rendimiento, especialmente si el programa de servicio se ejecuta en el mismo grupo de activación que el del llamador.

La Figura 10 en la página 33 muestra un ejemplo en el que un programa ILE está enlazado a dos programas de servicio.

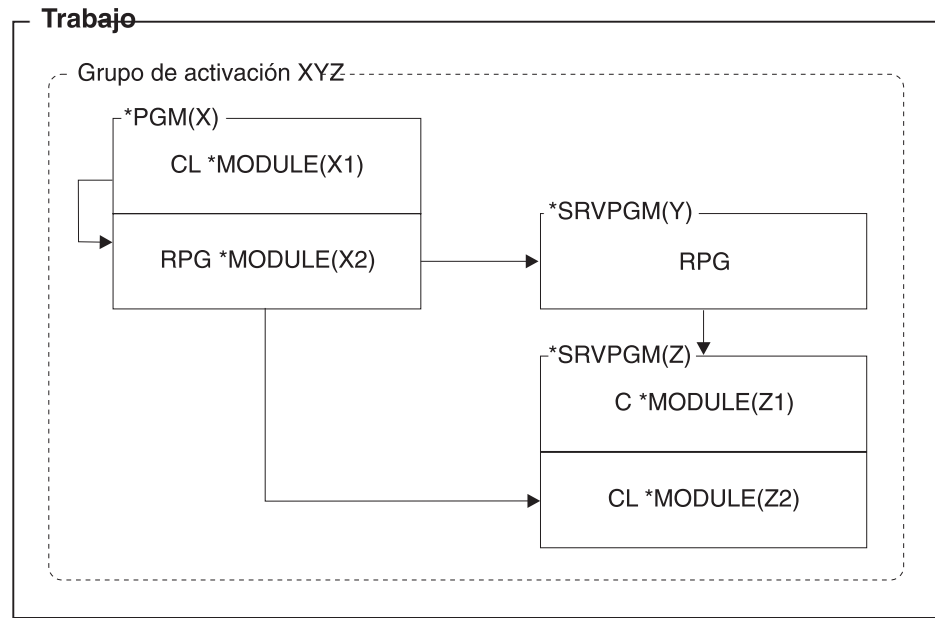


Figura 10. Aplicación avanzada

Las llamadas del programa X a los programas de servicio Y y Z son llamadas estáticas.

Consulte el apartado “Efecto de ILE” en la página 28 para ver detalles sobre los efectos de ILE en la manera en que la aplicación maneja llamadas, datos, archivos y errores.

Información relacionada

Proceso de creación de dos pasos

“Capítulo 7. Creación de un programa con los mandatos CRTRPGMOD y CRTPGM” en la página 77

Grupos de activación

“Gestión de los grupos de activación” en la página 112

Enlace estático de ILE

“Capítulo 10. Llamadas a programas y procedimientos” en la página 131; también *ILE Concepts*

Manejo de excepciones

“Capítulo 13. Manejo de excepciones” en la página 251; también *ILE Concepts*

Programas de servicio

“Capítulo 8. Creación de un programa de servicio” en la página 93; también la publicación *ILE Concepts*

Actualización de un programa

“Utilización del mandato UPDPGM” en la página 90

Una estrategia que debe evitar

ILE proporciona muchas alternativas para crear programas y aplicaciones. Sin embargo, no todas son igual de buenas. En general, debe evitar una situación en la que una aplicación compuesta por programas OPM e ILE esté dividida entre el grupo de activación por omisión OPM y un grupo de activación con nombre. En otras palabras, intente evitar el marco hipotético que se muestra en la Figura 11 .

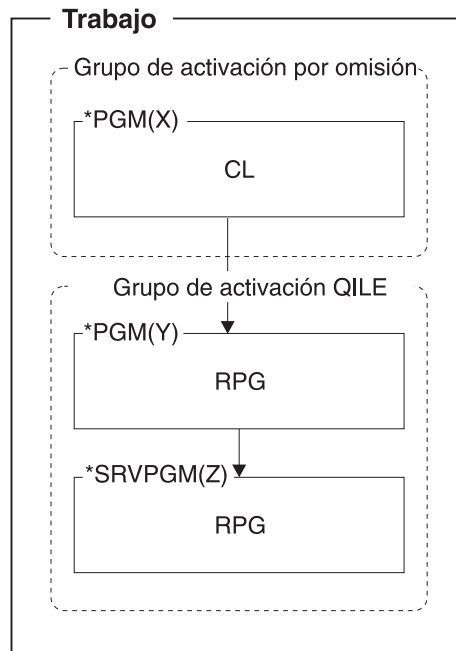


Figura 11. Marco hipotético que debe evitar. Una aplicación se divide entre el grupo de activación por omisión OPM y un grupo de activación con nombre.

Al realizar una división entre el grupo de activación por omisión y cualquier grupo de activación con nombre, mezcla el comportamiento OPM con el comportamiento ILE. Por ejemplo, los programas del grupo de activación por omisión pueden esperar que los programas ILE liberen sus recursos cuando el programa finalice. Sin embargo, esto no sucederá hasta que finalice el grupo de activación.

De forma similar, será más difícil gestionar el ámbito de las alteraciones temporales y de los ODP compartidos cuando una aplicación esté dividida entre el grupo de activación por omisión y uno con nombre. Por omisión, el ámbito del grupo con nombre será a nivel del grupo de activación, pero en el caso del grupo de activación por omisión puede ser a nivel de llamada o a nivel de trabajo, no a nivel de grupo de activación.

Capítulo 4. Creación de una aplicación utilizando múltiples procedimientos

La capacidad de codificar más de un procedimiento en un módulo ILE RPG aumenta enormemente su capacidad de codificar una aplicación modular. Este capítulo trata sobre cómo y por qué puede utilizar dicho módulo en su aplicación. Específicamente este capítulo presenta:

- Una visión general de los conceptos clave
- Un ejemplo de módulo con más de un procedimiento
- Consideraciones acerca de la codificación

Consulte el final de esta sección para ver dónde ha de buscar una información más detallada sobre la codificación de módulos con múltiples procedimientos.

Un módulo de múltiples procedimientos — Visión general

Un programa ILE consta de uno o más módulos; un módulo está compuesto por uno o más procedimientos. Un **procedimiento** es un segmento de código al que puede llamarse mediante una llamada enlazada. ILE RPG tiene dos tipos de procedimientos: un procedimiento principal y un subprocedimiento. El modo en que se ha de llamar a un subprocedimiento se denomina llamada de prototipos.

Nota: en la documentación sobre RPG, el término 'procedimiento' hace referencia tanto al principal como a los subprocedimientos.

Procedimientos principales y subprocedimientos

Un módulo ILE RPG consta de un procedimiento principal y cero o más subprocedimientos. (Si existen subprocedimientos, el procedimiento principal es opcional.) Un **procedimiento principal** es un procedimiento que puede especificarse como el procedimiento de entrada del programa (y, por lo tanto, puede recibir el control cuando se llama por primera vez a un programa ILE). El procedimiento principal se define en la **sección fuente principal**, que es el conjunto de especificaciones H, F, D, I, C y O que inician un módulo. En V3R1, todos los módulos ILE RPG tenían un procedimiento principal y ningún otro procedimiento.

Un **subprocedimiento** es un procedimiento que se especifica después de la sección fuente principal. Un subprocedimiento difiere de un procedimiento principal primordialmente en que:

- Los nombres definidos en un subprocedimiento no son accesibles fuera del subprocedimiento.
- No se genera código de ciclo para el subprocedimiento.
- La llamada de interfaz debe ser de prototipos.
- Las llamadas a subprocedimientos deben ser llamadas de procedimiento enlazadas.
- Sólo pueden utilizarse las especificaciones P, D y C.

Los subprocedimientos pueden proporcionar independencia de otros subprocedimientos porque los ítems de datos son locales. Normalmente, los ítems

Módulo de múltiples procedimientos

de datos locales se almacenan en almacenamiento automático, lo que significa que no se conserva el valor de una variable local entre las llamadas al procedimiento.

Los subprocedimientos ofrecen otra característica. Puede pasar parámetros a un subprocedimiento por valor y puede llamar a un subprocedimiento en una expresión para devolver un valor. La Figura 12 muestra el aspecto que puede tener un módulo con múltiples procedimientos.

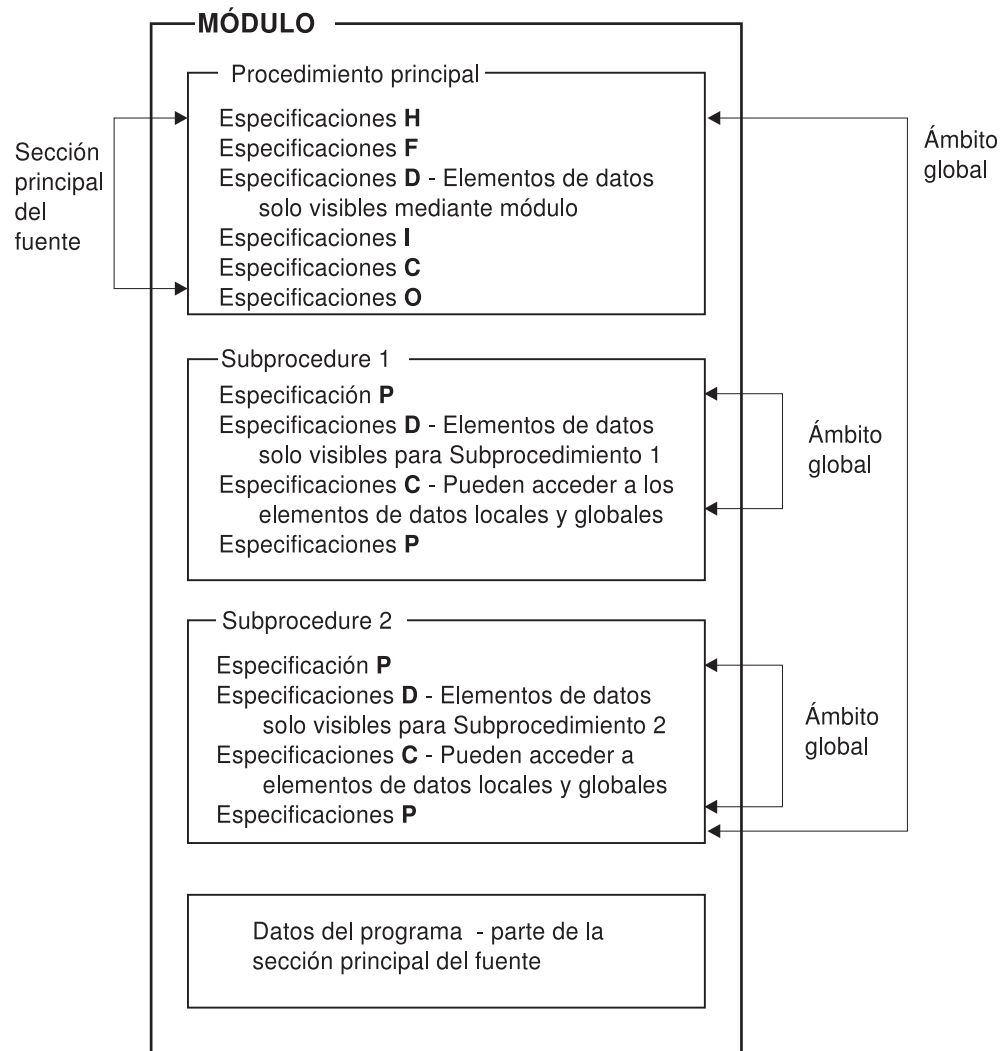


Figura 12. Un módulo ILE RPG con múltiples procedimientos

Como indica el gráfico, ahora puede codificar subprocedimientos para manejar tareas determinadas. Estas tareas pueden necesitarlas los procedimientos principales u otros módulos de la aplicación. Además, puede declarar ítems de datos temporales en los subprocedimientos, de modo que no tendrá que preocuparse de si los ha declarado en otro lugar del módulo.

Llamadas de prototipos

Para llamar a un subprocedimiento, debe utilizar una llamada de prototipos. También puede llamar a cualquier programa o procedimiento que se haya grabado en cualquier lenguaje de este modo. Una **llamada de prototipos** es una llamada en la que la interfaz de llamada se comprueba en tiempo de compilación mediante la

utilización de un prototipo. Un **prototipo** es una definición de la interfaz de llamada. Incluye la información siguiente:

- Si la llamada es enlazada (procedimiento) o dinámica (programa)
- Cómo buscar el programa o procedimiento (nombre externo)
- El número y la naturaleza de los parámetros
- Qué parámetros deben pasarse y cuáles se pasan opcionalmente
- Si se pasan los descriptores operativos (para un procedimiento)
- El tipo de datos del valor de retorno, si lo hubiera (para un procedimiento)

El prototipo lo utiliza el compilador para llamar al programa o procedimiento correctamente y para asegurarse de que el llamador pasa los parámetros correctamente. La Figura 13 muestra un prototipo de un procedimiento FmtCust, que da formato a varios campos de un registro de modo que sean legibles. Tiene dos parámetros de salida.

```
// Prototipo para el procedimiento FmtCust (Fíjese en PR
// de la especificación de definición.) Tiene dos parámetros
// de salida.
D FmtCust          PR
D Nombre           100A
D Dirección        100A
```

Figura 13. Prototipo para el procedimiento FmtCust

Para convertir series de caracteres en un número, InArrears llama a un procedimiento CharToNum. CharToNum tiene un parámetro de entrada de caracteres y devuelve un campo numérico. La Figura 14 en la página 38 muestra el prototipo para CharToNum.

Módulo de múltiples procedimientos

```
//-----
// CharToNum - procedimiento para leer un número a partir de una
//               serie y devolver un valor 30p 9
// Parámetros:
//   I:      string - valor en caracteres del número
//   I:(opc) decComma - separador de coma decimal y dígito
//   I:(opc) currency - símbolo de moneda de importes monetarios
// Devuelve: packed(30,9)
//
// Detalles de parámetros:
//   string:  la serie puede tener
//             - blancos en cualquier lugar
//             - signo en cualquier lugar
//             Los signos aceptados son: + - cr CR ( )
//             (véanse ejemplos a continuación)
//             - separadores de dígitos en cualquier lugar
//             - símbolo de moneda en cualquier lugar
//   decComma: si no se pasa, el valor por omisión es
//             coma decimal   = '.'
//             separ. dígito  = ','
//   currency: si no se pasa, el valor por omisión es ' '
//
// Ejemplos de entrada y salida (x indica parámetro no pasado):
//
//      serie      | dec | sep | símbo | result.
//      -----+-----+-----+-----+-----
//      123         | x   | x   | x     | 123
//      +123        | x   | x   | x     | 123
//      123+        | x   | x   | x     | 123
//      -123        | x   | x   | x     | -123
//      123-        | x   | x   | x     | -123
//      (123)       | x   | x   | x     | -123
//      12,3        | ,   | .   | x     | 12.3
//      12.3        | x   | x   | x     | 12.3
//      1,234,567.3 | x   | x   | x     | 1234567.3
//      $1,234,567.3 | .   | ,   | $     | 1234567.3
//      $1.234.567,3 | ,   | .   | $     | 1234567.3
//      123.45CR    | x   | x   | x     | -123.45
//-----
D CharToNum      pr          30p 9
D string         100a  const varying
D decComma       2a  const options(*nopass)
D currency       1a  const options(*nopass)
```

Figura 14. Prototipo para el procedimiento CharToNum

Si el programa o procedimiento es de prototipos, se llama con CALLP o en una expresión si desea utilizar el valor de retorno. Los parámetros se pasan en una lista a continuación del nombre del prototipo, por ejemplo, *nombre (parám1 : parám2 : ...)*.

La Figura 15 muestra una llamada a FmtCust. Tenga en cuenta que los nombres de los parámetros de salida, que aparecen en la Figura 13 en la página 37, no coinciden con los de la sentencia de la llamada. Los nombres de parámetros de un prototipo son meramente ilustrativos. El prototipo sirve para *describir* los atributos de la interfaz de llamada. La definición real de los parámetros de llamada tiene lugar en el propio procedimiento.

C	CALLP	FmtCust(RPTNAME : RPTADDR)
----------	--------------	-----------------------------------

Figura 15. Llamada al procedimiento FmtCust

Utilizando las llamadas de prototipos puede llamar (con la misma sintaxis) a:

Módulo de múltiples procedimientos

- Programas que estén en el sistema durante la ejecución
- Procedimientos exportados en otros módulos o programas de servicio que estén enlazados en el mismo programa o programa de servicio
- Subprocedimientos del mismo módulo

InArrears llama a CharToNum para convertir la cantidad con formato de serie en un número. Dado que InArrears desea utilizar el valor de retorno, la llamada a CharToNum se realiza en una expresión. La Figura 16 muestra la llamada.

```
// Los datos del archivo de entrada vienen de otro
// tipo de sistema y el campo AMOUNTC es una serie de
// caracteres que contiene el valor numérico. Esta serie
// debe convertirse en el campo AMOUNT numérico
// para la impresión.
AMOUNT = CharToNum(AMOUNTC);
```

Figura 16. Llamada al procedimiento CharToNum

La utilización de procedimientos para devolver valores, como se muestra en la figura anterior le permite grabar todas las funciones definidas por usuario que necesite. Además, la utilización de una interfaz de llamada de prototipos abre múltiples opciones nuevas para pasar parámetros.

- Pueden pasarse parámetros de prototipos de diversas formas: por referencia, por valor (sólo para procedimientos) o por referencia de sólo lectura. El método por omisión de RPG es pasar por referencia. Sin embargo, pasar por valor o por referencia de sólo lectura da más opciones para pasar parámetros.
- Si el prototipo indica que está permitido para un parámetro dado, puede efectuar una o más de las operaciones siguientes:
 - Pasar *OMIT
 - Omitir totalmente un parámetro
 - Pasar un parámetro más corto de lo especificado (para parámetros de caracteres y gráficos y para parámetros de matriz)

Ejemplo de un módulo con múltiples procedimientos

Analicemos ahora un ejemplo de un módulo con múltiples procedimientos. En esta 'miniaplicación' estamos escribiendo un programa ARRSRPT para crear un informe de todos los clientes cuyas cuentas están atrasadas. Crearemos un informe básico como un módulo, de modo que pueda enlazarse con otros módulos si es necesario. Para este módulo son necesarias dos tareas principales:

1. Determinar que un registro de una cuenta de un archivo de cliente está atrasado.
2. Dar formato a los datos de modo que resulten adecuados para el informe.

Hemos decidido codificar cada tarea como un subprocedimiento. Conceptualmente, el módulo se parecerá al que se muestra en la Figura 17 en la página 40.

Ejemplo de un módulo con múltiples procedimientos

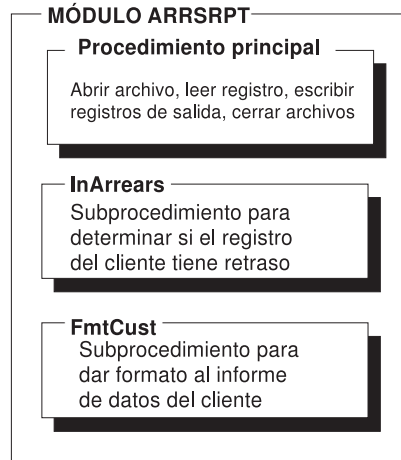


Figura 17. Componentes del módulo ARRSRPT

Ahora fíjese en el primer subprocedimiento, **InArrears**, que aparece en la Figura 18 en la página 41. El procedimiento principal llama a **InArrears** para determinar si el registro actual está atrasado.

SUGERENCIA

Cuando codifique subprocedimientos que utilicen campos globales, puede que desee establecer un convenio de denominación que muestre el ítem como global. En este ejemplo, los nombres de campo en mayúsculas indican campos DDS. Otra opción sería especificar el prefijo 'g_' o alguna otra serie para indicar el ámbito global.

Dado que los datos proceden de otro tipo de sistema, el campo **AMOUNTC** es una serie de caracteres que representa un valor numérico, como por ejemplo '12.35'. En una de las tareas se debe leer una serie que contiene un valor numérico y extraer el valor numérico. Esta conversión es una tarea muy habitual, por lo que hemos decidido codificarla como un subprocedimiento aparte, **CharToNum**. Este subprocedimiento toma una serie de caracteres y devuelve un número empaquetado con 30 dígitos y 9 posiciones decimales.

Si el registro está atrasado, el subprocedimiento devuelve '1' al procedimiento principal.

Ejemplo de un módulo con múltiples procedimientos

```
//-----
// InArrears
//
// Parámetros: (ninguno)
// Globales:  DUEDATE, AMOUNT, CurDate
//
// Devuelve:  '1' si el cliente es moroso
//-----
P InArrears      B      1
D InArrears      PI      1A  2
// Declaraciones locales
D DaysLate       S      10I 0  3
D DateDue        S      D      3
// Cuerpo del procedimiento
/free
    DateDue = %date (DUEDATE: *ISO);
    DaysLate = %diff (CurDate: DateDue: *d);
    // Los datos del archivo de entrada vienen de otro
    // tipo de sistema y el campo AMOUNTC es una serie de
    // caracteres que contiene el valor numérico. Esta serie
    // debe convertirse en el campo AMOUNT numérico
    // para la impresión.
    AMOUNT = CharToNum(AMOUNTC);
    if DaysLate > 60 AND AMOUNT > 100.00;
        return '1';
    endif;
    return '0';
/end-free
P InArrears      E      1
```

Figura 18. Fuente para el subprocedimiento InArrears

La Figura 18 muestra los elementos principales que son comunes a todos los subprocedimientos.

- 1 Todos los subprocedimientos comienzan y finalizan con especificaciones de procedimiento.
- 2 Después de la especificación de inicio de procedimiento (B en la posición 24 de la especificación de procedimiento), codifica una definición de interfaz de procedimiento. El valor de retorno, si lo hay, se define en la especificación PI. Todos los parámetros se listan después de la especificación PI.
- 3 Las variables o los prototipos que utiliza el subprocedimiento se definen después de la definición de interfaz de procedimiento.
- 4 El valor de retorno, si se especifica, se devuelve al llamador con una operación RETURN.
- 5 Si el registro no está atrasado, el subprocedimiento devuelve '0' al procedimiento principal.

Para todos los subprocedimientos y también para un procedimiento principal con parámetros de entrada de prototipos, es necesario que defina una interfaz de procedimiento. Una **definición de interfaz de procedimiento** es una repetición de la información de prototipo en la definición de un procedimiento. Se utiliza para definir los parámetros de entrada para el procedimiento. La definición de interfaz de procedimiento se utiliza también para asegurarse de que la definición interna del procedimiento sea coherente con la definición externa (el prototipo). En el caso de InArrears, no existen parámetros de entrada.

Ejemplo de un módulo con múltiples procedimientos

Examine a continuación el subprocedimiento FmtCust, que se muestra en la Figura 19. A FmtCust lo llama ARRSRPT para dar formato a los campos relevantes de un registro en un registro de salida para el informe final. (El registro representa una cuenta que está atrasada.) FmtCust utiliza datos globales, de modo que no tiene parámetros de entrada. Da formato a los datos en dos campos de salida: uno para el nombre y otro para la dirección.

```
//-----  
// FmtCust da formato a CUSTNAME, CUSTNUM, STREETNAME etc para  
// que sean legibles  
//  
// Parámetros:  Nombre      (salida)  
//              Dirección   (salida)  
// Globales:    CUSTNAME, CUSTNUM, STREETNUM, STREETNAME, CITY  
//              STATE, ZIP  
//-----  
  
P FmtCust      B  
D FmtCust      PI  
D Nombre              100A  
D Dirección          100A  
D ZipChar            S      5A  
  
/free  
//-----  
// CUSTNAME y CUSTNUM obtienen el formato siguiente:  
// A&P Electronics      (Número de cliente 157)  
//-----  
Nombre = CUSTNAME + ' ' + '(Número cliente '  
        + %char(CUSTNUM) + ')';  
  
//-----  
// STREETNUM, STREETNAME, CITY, STATE y ZIP obtienen el formato  
// siguiente:  
// Calle de garbancito, 27 Villachica IN 51423  
//-----  
ZipChar = %editc(ZIP:'X');  
Dirección = %char(STREETNUM) + ' ' + %trimr(STREETNAME)  
        + ', ' + %trim(CITY) + ' ' + %trim(STATE)  
        + ' ' + ZipChar;  
  
/end-free  
P FmtCust      E
```

Figura 19. Fuente para el subprocedimiento FmtCust

Por último, considere el último subprocedimiento de esta aplicación, CharToNum. Fíjese en que CharToNum no aparece en el módulo ARRSRPT, que se muestra en la Figura 17 en la página 40. Hemos decidido colocar CharToNum dentro de otro módulo denominado CVTPROCS. CVTPROCS es un módulo de programa de utilidad que contendrá los procedimientos de conversión que otros módulos pueden necesitar.

La Figura 20 en la página 43 muestra el fuente del módulo CVTPROCS. Dado que se trata de un procedimiento de prototipos, necesita que el prototipo esté disponible. Para que el prototipo pueda compartirse, hemos colocado el prototipo en un archivo /COPY.

Ejemplo de un módulo con múltiples procedimientos

```
//=====
// Fuente para el módulo CVTPROCS. Este módulo no tiene un
// procedimiento principal, como indica la palabra clave NOMAIN.
//=====
H NOMAIN
//-----
// El prototipo debe estar disponible para CADA módulo que contenga
// un procedimiento de prototipos. /COPY extrae el prototipo
// para CharToNum.
//-----
D/COPY QRPGLSRC,CVTPROC_P
P CharToNum      B
D CharToNum      PI          30P 9
D string         100A      CONST VARYING
D decComma       2A        CONST OPTIONS(*NOPASS)
D currency       1A        CONST OPTIONS(*NOPASS)
// Valores por omisión para parámetros opcionales
D decPoint       S          1A  INZ('.')
D comma          S          1A  INZ(',')
D cursym         S          1A  INZ(' ')
// Estructura para crear resultado
D               DS
D result         30S 9  INZ(0)
D resChars       30A      OVERLAY(result)
// Variables para recopilar información de dígito
// pNumPart apunta al área que se recopila actualmente
// (la parte entera o la parte decimal)
D pNumPart       S          *
D numPart        S          30A  VARYING BASED(pNumPart)
D intPart        S          30A  VARYING INZ('')
D decPart        S          30A  VARYING INZ('')
// Otras variables
D intStart       S          10I 0
D decStart       S          10I 0
D sign           S          1A   INZ('+')
D i              S          10I 0
D len            S          10I 0
D c              S          1A
```

Figura 20. Fuente para el módulo CVTPROCS que contiene el subprocedimiento CharToNum (Pieza 1 de 2)

Ejemplo de un módulo con múltiples procedimientos

```
/free
// Alterar valores por omisión si se pasan parámetros opcionales
if      %parms > 1;
    decPoint = %subst(decComma : 1 : 1);
    comma    = %subst(decComma : 2 :1);
endif;
if %parms > 2;
    cursym = currency;
endif;
// Inicialización
len = %len(string);
// Empezar a leer la parte entera
pNumPart = %addr(intPart);
// Iterar en bucle por los caracteres
for i = 1 to len;
    c = %subst(string : i : 1);
    select;
    // Omitir blancos, separador de dígito y símbolo de moneda
    when c = comma or c = *blank or c = cursym;
        iter;
    // Coma decimal: pasar a leer la parte decimal
    when c = decPoint;
        pNumPart = %addr(decPart);
        iter;
    // Signo: recordar el signo más reciente
    when c = '+' or c = '-';
        sign = c;
        iter;
    // Más signos: cr, CR, () son todos signos negativos
    when c = 'C' or c = 'R' or
        c = 'c' or c = 'r' or
        c = '(' or c = ')';
        sign = '-';
        iter;
    // Un dígito: añadirlo al área de creación actual
    other;
        eval numPart = numPart + c;
    ends1;
endfor;
// Copiar las series de dígito en las posiciones correctas de la
// variable con zona, utilizando el carácter de recubrimiento
decStart = %len(result) - %decPos(result) + 1;
intStart = decStart - %len(intPart);
eval %subst(resChars : intStart : %len(intPart)) = intPart;
eval %subst(resChars : decStart : %len(decPart)) = decPart;
// Si el signo es negativo, devolver un valor negativo
if sign = '-';
    return -result;
// Si no, devolver el valor positivo
else;
    return result;
endif;
/end-free
p          e
```

Figura 20. Fuente para el módulo CVTPROCS que contiene el subprocedimiento CharToNum (Pieza 2 de 2)

CVTPROCS es un **módulo NOMAIN**, lo que significa que está compuesto únicamente por subprocedimientos; no existe ningún procedimiento principal. Un módulo NOMAIN compila con mayor rapidez y requiere menos almacenamiento porque no se crea un código de ciclos para el módulo. Un módulo NOMAIN se

Ejemplo de un módulo con múltiples procedimientos

especifica codificando la palabra clave NOMAIN en la especificación de control. Para obtener más información sobre los módulos NOMAIN, consulte “Creación del programa” en la página 50.

El programa ARRSRPT completo

El programa ARRSRPT consta de dos módulos: ARRSRPT y CVTPROCS. La Figura 21 muestra las distintas partes de esta miniaplicación.

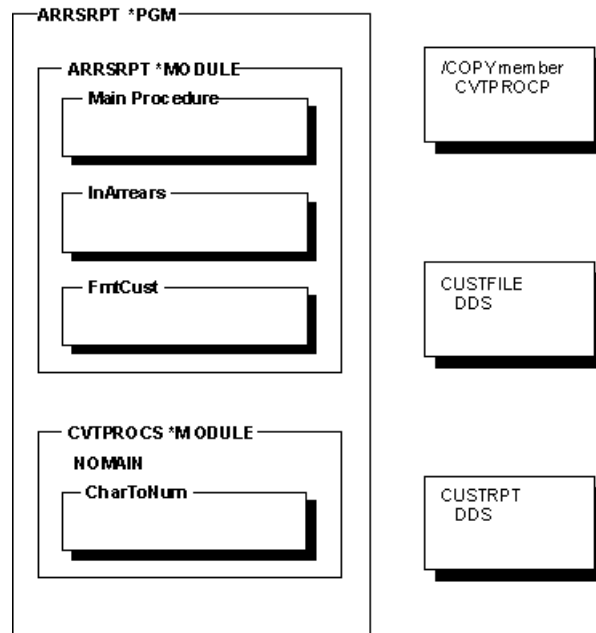


Figura 21. La aplicación ARRSRPT

La Figura 22 muestra el fuente del módulo ARRSRPT completo.

```
//=====
// Fuente para el módulo ARRSRPT. Contiene un procedimiento principal y
// dos subprocedimientos: InArrears y FmtCust.
//
// Módulo relacionado: CVTPROCS (CharToNum llamado por InArrears)
//=====
//-----
// A R C H I V O S
//
// CUSTFILE - contiene información de clientes
// CUSTRPT - archivo de impresora (con el formato ARREARS)
//-----
FCUSTFILE IP E DISK
FCUSTRPT 0 E PRINTER
```

Figura 22. Fuente ILE RPG completo para el módulo ARRSRPT (Pieza 1 de 3)

Ejemplo de un módulo con múltiples procedimientos

```

*-----*
* P R O T O T I P O S
*-----*
/COPY QRPGL,CVTPROCP
*-----*
* InArrears devuelve '1' si el cliente es moroso
*-----*
D InArrears      PR      1A
*-----*
* FmtCust da formato a CUSTNAME, CUSTNUM, STREETNAME etc de
* modo legible
*-----*
D FmtCust      PR
D Nombre      100A
D Dirección    100A
*-----*
* D E F I N I C I O N E S   G L O B A L E S
*-----*
D CurDate      S      D
ICUSTREC      01
*-----*
* P R O C E D I M I E N T O   P R I N C I P A L
*-----*
C      IF      InArrears = '1'
C      CALLP   FmtCust(RPTNAME : RPTADDR)
C      EVAL    RPTNUM = CUSTNUM
C      WRITE   ARREARS
C      ENDIF
C      *INZSR   BEGSR
C      *MDY     MOVEL      UDATE      CurDate
C      ENDSR
*-----*
* S U B P R O C E D I M I E N T O S
*-----*
*-----*
* InArrears
*
* Parámetros: (ninguno)
* Globales:  DUEDATE, AMOUNT, CurDate
*
* Devuelve:  '1' si el cliente es moroso
*-----*

P InArrears      B
D InArrears      PI      1A

* Declaraciones locales
D DaysLate      S      10I 0
D DateDue       S      D

```

Figura 22. Fuente ILE RPG completo para el módulo ARRSRPT (Pieza 2 de 3)

Ejemplo de un módulo con múltiples procedimientos

```

* Cuerpo del procedimiento
C      *ISO      MOVE      DUEDATE      DateDue
C      CurDate   SUBDUR    DateDue      DaysLate:*D
C      IF        DaysLate > 60 AND
C      AMOUNT > 100.00
C      RETURN    '1'
C      ELSE
C      RETURN    '0'
C      ENDIF
P InArrears      E

*-----
* FmtCust da formato a CUSTNAME, CUSTNUM, STREETNAME etc de
* modo legible
*
* Parámetros:  Nombre      (salida)
*              Dirección  (salida)
* Globales:    CUSTNAME, CUSTNUM, STREETNUM, STREETNAME, CITY
*
*              STATE, ZIP
*
* Devuelve:    (ninguno)
*-----
P FmtCust      B
D FmtCust      PI
D Nombre      100A
D Dirección   100A
D ZipChar      S      5A
*-----
* CUSTNAME y CUSTNUM obtienen el formato siguiente:
*      A&P Electronics      (Número de cliente 157)
*-----
C      EVAL      Nombre = CUSTNAME + ' '
C      + '(Número del cliente '
C      + %char(CUSTNUM) + ' )'

*-----
* StreetNum, STREETNAME, CITY, STATE y ZIP obtienen el formato:
*      Calle de garbancito, 27 Villachica 51423
*-----
C      MOVE      ZIP      ZipChar
C      EVAL      Dirección = %char(STREETNUM))
C      + ' ' + %trimr(STREETNAME) + ', '

C      + %trim(CITY) + ' ' + %trim(STATE)
C      + ' ' + ZipChar
P FmtCust      E

```

Figura 22. Fuente ILE RPG completo para el módulo ARRSRPT (Pieza 3 de 3)

Fíjese en lo siguiente acerca de ARRSRPT:

- Las especificaciones de definición comienzan con los prototipos para las llamadas de prototipos. Se utiliza un archivo /COPY para suministrar el prototipo para el procedimiento denominado CharToNum.
Los prototipos no han de ser los primeros, pero debe establecer un orden para los diferentes tipos de definición a fin de obtener coherencia.
- El campo de fecha CurDate es un campo global, lo que significa que todos los procedimientos del módulo pueden acceder a él.
- El procedimiento principal es fácil de seguir. Contiene especificaciones de cálculo para las dos tareas principales; la E/S y una rutina de inicialización.
- Cada subprocedimiento que va a continuación del procedimiento principal contiene los detalles de una de las tareas.

Ejemplo de un módulo con múltiples procedimientos

El ejemplo de salida del programa ARRSRPT se muestra en la Figura 23 .

```
Número del cliente: 00001
ABC Electronics      (Número del cliente 1)
Vía láctea, 15, Villadiego MN 12345
Cantidad pendiente:      $1234.56   Fecha vencim.: 1995-05-01

Número del cliente: 00152
A&P Electronics     (Número de cliente 152)
Calle de garbancito, 27 Villachica MN 51423
Cantidad pendiente:      $26544.50   Fecha vencim.: 1995-02-11
```

Figura 23. Salida para ARRSRPT

La Figura 24 y la Figura 25 en la página 49 muestran el fuente DDS para los archivos CUSTFILE y CUSTRPT respectivamente.

```
A*=====
A* NOMBRE ARCHIVO   : CUSTFILE
A* PROG. RELACIONADO: ARRSRPT
A* DESCRIPCIONES    : ESTE ES UN ARCHIVO FÍSICO CUSTFILE. TIENE
A*                   FORMATO DE REGISTRO LLAMADO CUSTREC.
A*=====
A* ARCHIVO MAESTRO CLIENTE -- CUSTFILE
A      R CUSTREC
A      CUSTNUM      5 0      TEXT('NÚMERO CLIENTE')
A      CUSTNAME     20      TEXT('NOMBRE CLIENTE')
A      STREETNUM    5 0      TEXT('DIRECCIÓN CLIENTE')
A      STREETNAME   20      TEXT('DIRECCIÓN CLIENTE')
A      CITY         20      TEXT('CIUDAD CLIENTE')
A      STATE        2       TEXT('PROVINCIA CLIENTE')
A      ZIP          5 0      TEXT('CÓDIGO POSTAL CLIENTE')
A      AMOUNTC      15      TEXT('CANTIDAD PENDIENTE')
A      DUEDATE      10      TEXT('FECHA VENCIMIENTO')
```

Figura 24. DDS para CUSTFILE

```

A*=====
A* NOMBRE ARCHIVO   : CUSTRPT
A* PROG. RELACIONADO: ARRSRPT

A* DESCRIPCIONES    : ESTE ES UN ARCHIVO FÍSICO CUSTRPT. TIENE
A*                               UN FORMATO DE REGISTRO LLAMADO ARREARS.
A*=====
A      R ARREARS
A
A      2  6
A      'Número de cliente:'
A      RPTNUM      5  0  2 23
A      TEXT('NÚMERO CLIENTE')
A      RPTNAME     100A  3 10
A      TEXT('NOMBRE CLIENTE')
A      RPTADDR     100A  4 10
A      TEXT('DIRECCIÓN CLIENTE')
A      5 10 'Cantidad pendiente:'
A      AMOUNT      10  2  5 35 EDTWRD(' $0. ')
A      TEXT('CANTIDAD PENDIENTE')
A      5 50 'Fecha vencimiento:'
A      DUEDATE     10    5 60
A      TEXT('FECHA VENCIMIENTO')

```

Figura 25. DDS para CUSTRPT

Consideraciones acerca de la codificación

Esta sección presenta algunas consideraciones que debe tener en cuenta antes de comenzar a diseñar aplicaciones con módulos de múltiples procedimientos. Están agrupadas en las categorías siguientes:

- General
- Creación del programa
- Procedimientos principales
- Subprocedimientos

Consideraciones generales

- Cuando codifique un módulo con múltiples procedimientos, deseará utilizar los archivos /COPY, primordialmente para contener cualquier prototipo que requiera su aplicación. Si está creando un programa de servicio, necesitará proporcionar tanto el programa de servicio como los prototipos, si existen.
- El mantenimiento de la aplicación requiere asegurarse de que cada componente esté en el nivel más reciente y que cualquier cambio no afecte a las diferentes partes. Para mantener sus aplicaciones puede que desee utilizar una herramienta como Application Development Manager.

Por ejemplo, suponga que otro programador efectúe un cambio en el archivo /COPY que contiene los prototipos. Cuando solicite que vuelva a crearse una aplicación, se recompilarán automáticamente todos los módulos o programas que utilicen el archivo /COPY. Sabrá rápidamente si los cambios realizados en el archivo /COPY afectarán a las interfaces de llamada o procedimiento de su aplicación. Si existen errores de compilación, puede decidir si acepta el cambio en los prototipos para evitar estos errores o si desea cambiar la interfaz de llamada.

Consideraciones acerca de la codificación

Creación del programa

- Si especifica que un módulo no tenga un procedimiento principal, entonces no podrá utilizar el mandato CRTBNDRPG para crear el programa. (Un módulo no tiene un procedimiento principal si se especifica la palabra clave NOMAIN en una especificación de control.) Esto es debido a que el mandato CRTBNDRPG requiere que el módulo contenga un procedimiento de entrada de programa y únicamente un procedimiento principal puede ser un procedimiento de entrada de programa .
- Del mismo modo, cuando utilice CRTPGM para crear el programa, recuerde que un módulo NOMAIN no puede ser un módulo de entrada ya que no tiene un procedimiento de entrada de programa .
- Un programa que se crea para ejecutarse en el grupo de activación OPM por omisión (especificando DFTACTGRP(*YES) en el mandato CRTBNDRPG) no puede contener llamadas de procedimiento enlazadas.

Consideraciones acerca del procedimiento principal

- Dado que el procedimiento principal es el único procedimiento con un conjunto completo de especificaciones disponibles (excepto la especificación P), debe utilizarse para configurar el entorno de todos los procedimientos del módulo.
- Un procedimiento principal se exporta siempre, lo que significa que otros procedimientos del programa pueden llamar al procedimiento principal utilizando llamadas enlazadas.
- La interfaz de llamada de un procedimiento principal puede definirse de uno de los dos modos siguientes:
 1. Utilizando una interfaz de prototipo y de procedimiento
 2. Utilizando *ENTRY PLIST sin un prototipo
- La funcionalidad de *ENTRY PLIST es similar a una interfaz de llamada de prototipos. Sin embargo, una interfaz de llamada de prototipos es mucho más robusta ya que proporciona comprobación de parámetros en tiempo de compilación. Si el procedimiento principal ha de ser de prototipos, especificando la palabra clave EXTPROC o EXTPGM en la definición de prototipos especificará cómo se le ha de llamar. Si se especifica EXTPGM, se utilizará una llamada de programa externo; si se utiliza EXTPROC o no se especifica ninguna de estas palabras clave, se llamará utilizando una llamada de procedimiento.
- Para un procedimiento principal no puede definir valores de retorno ni puede especificar que sus parámetros se pasen por valor.

Consideraciones acerca de los subprocedimientos

- En un subprocedimiento se pueden codificar todas las operaciones de cálculo. Sin embargo, todos los archivos deben definirse globalmente, por lo que todas las especificaciones de entrada y de salida deben definirse en la sección fuente principal. Del mismo modo, todas las áreas de datos deben definirse en el procedimiento principal, aunque pueden utilizarse en un subprocedimiento.
- La especificación de control únicamente puede codificarse en la sección fuente principal ya que controla todo el módulo.
- Se puede llamar a un subprocedimiento de modo recurrente. Cada llamada recurrente crea una nueva invocación del procedimiento que se ha de colocar en la pila de llamadas. La nueva invocación tiene un almacenamiento nuevo para todos los ítems de datos del almacenamiento automático y dicho almacenamiento no está disponible a las demás invocaciones ya que es local.

Consideraciones acerca de la codificación

(Un ítem de datos definido en un subprocedimiento utiliza almacenamiento automático a menos que se especifique la palabra clave `STATIC` para la definición.)

El almacenamiento automático asociado a las invocaciones anteriores no resulta afectado por las invocaciones posteriores. Todas las invocaciones comparten el mismo almacenamiento estático, por lo que las invocaciones posteriores pueden afectar al valor que contiene una variable en el almacenamiento estático.

La recurrencia puede resultar una técnica de programación de gran utilidad si se ha comprendido correctamente.

- El comportamiento de tiempo de ejecución de un subprocedimiento difiere ligeramente del de un procedimiento principal, dado que no existe un código de ciclos para el subprocedimiento.
 - Cuando finaliza un subprocedimiento, simplemente se devuelve al llamador. No se producen ninguna de las actividades de finalización normales, como por ejemplo el cierre de archivos, hasta que finaliza el procedimiento principal asociado al propio subprocedimiento. Debe codificar un subprocedimiento de "limpieza" al que llamen tanto el procedimiento de entrada de programa en la finalización de la aplicación como un manejador de cancelaciones habilitado para el procedimiento de entrada de programa. Una alternativa es codificar el módulo `NOMAIN`, de modo que no haya una apertura de archivo implícita o un bloqueo del área de datos y que, en todo subprocedimiento, una apertura se corresponda con un cierre, un `IN` con un `OUT`, un `CRT<temp obj>` con un `DLT<temp obj>`, y así sucesivamente. Esta alternativa es válida para los módulos que pueden tener un subprocedimiento activo cuando el procedimiento principal no esté activo.
 - El manejo de excepciones en un subprocedimiento difiere de un procedimiento principal primordialmente en que para los subprocedimientos no existe un manejador de excepciones por omisión. En consecuencia, las situaciones en las que para un procedimiento principal se llamaría al manejador por omisión corresponderían a una finalización anormal del subprocedimiento.

Para obtener más información

Para obtener más información sobre los temas tratados aquí, consulte la lista siguiente:

Procedimientos principales

Tema	Consulte
Manejo de excepciones	"Manejo de excepciones en un procedimiento principal" en la página 255
Finalización del procedimiento principal	"Volver de un procedimiento principal" en la página 157

Subprocedimientos

Tema	Consulte
Definición	Capítulo sobre subprocedimientos en la publicación <i>ILE RPG Reference</i>
Módulo <code>NOMAIN</code>	"Creación de un módulo <code>NOMAIN</code> " en la página 79

Para obtener más información

Manejo de excepciones	“Manejo de excepciones en subprocedimientos” en la página 255
Especificación de procedimiento	Capítulo sobre especificaciones de procedimiento en la publicación <i>ILE RPG Reference</i>
Interfaz de procedimiento	Capítulo sobre definición de datos y prototipos en la publicación <i>ILE RPG Reference</i>
Finalización del subprocedimiento	“Volver de un subprocedimiento” en la página 159

LLamada de prototipos

Tema	Consulte
Llamada de formato libre	“Utilización de una llamada de prototipos” en la página 137
Información general	<i>ILE RPG Reference</i> , capítulo 24
Pasar parámetros	“Pasar parámetros de prototipos” en la página 139
Prototipos	Capítulo sobre definición de datos y prototipos en la publicación <i>ILE RPG Reference</i>

Parte 2. Creación y ejecución de una aplicación ILE RPG

Este apartado le proporciona la información necesaria para crear y ejecutar programas ILE RPG. En él se describe cómo:

- Entrar sentencias fuente
- Crear módulos
- Leer listados del compilador
- Crear programas
- Crear programas de servicio
- Ejecutar programas
- Pasar parámetros
- Gestionar el tiempo de ejecución
- Llamar a otros programas o procedimientos

En las páginas siguientes se tratan brevemente muchos de los términos y conceptos de Integrated Language Environment. Estos términos y conceptos se describen con mayor detalle en la publicación *ILE Concepts*.

Capítulo 5. Entrar sentencias fuente

En este capítulo se proporciona la información necesaria para entrar sentencias fuente RPG. También se describen brevemente las herramientas necesarias para realizar esta tarea.

Para entrar sentencias fuente RPG en el sistema, utilice uno de los métodos siguientes:

- De forma interactiva utilizando el SEU
- De forma interactiva utilizando CODE/400

Al principio, tal vez desee entrar las sentencias fuente en un archivo denominado QRPGLSRC. Los miembros nuevos del archivo QRPGLSRC reciben automáticamente el tipo RPGLE por omisión. Además, el archivo fuente por omisión para los mandatos ILE RPG que crean módulos y los enlazan en objetos de programa es QRPGLSRC. IBM proporciona un archivo fuente QRPGLSRC en la biblioteca QGPL. Su longitud de registro es de 112 caracteres.

Nota: puede utilizar mayúsculas y minúsculas al entrar el fuente. No obstante, el compilador ILE RPG convertirá la mayor parte del fuente a mayúsculas al compilarlo. No convertirá los literales, los datos de las matrices ni los datos de las tablas.

Creación de una biblioteca y de un archivo físico fuente

Las sentencias fuente se colocan en un miembro de un archivo físico fuente. Para poder entrar un programa, debe tener una biblioteca y un archivo físico fuente.

Para crear una biblioteca, utilice el mandato CRTLIB. Para crear un archivo físico fuente, utilice el mandato Crear Archivo Físico Fuente (CRTSRCPF). La longitud de registro recomendada para este archivo es de 112 caracteres. Esta longitud de registro tiene en cuenta la nueva estructura ILE RPG tal como se indica en la Figura 26.

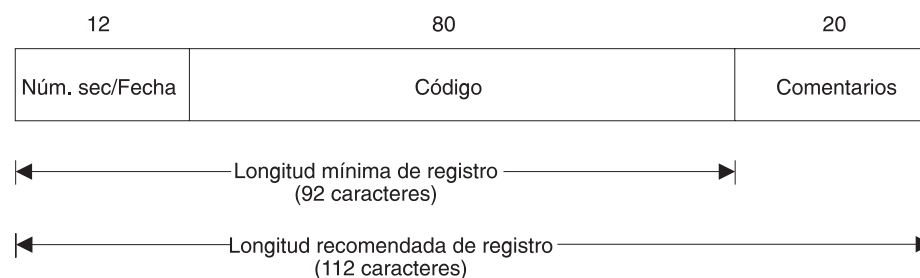


Figura 26. Desglose de la longitud de registro de ILE RPG

Puesto que el valor por omisión del sistema es de 92 caracteres para un archivo físico fuente, debe especificar explícitamente una longitud mínima de registro de 112. Si especifica una longitud inferior a 92 caracteres, el programa probablemente no se compilará, ya que puede que esté truncando el código fuente.

Para obtener más información acerca de la creación de bibliotecas y archivos físicos fuente, consulte las publicaciones *ADTS for AS/400: Source Entry Utility* y *ADTS/400: Programming Development Manager*.

Utilización del programa de utilidad para entrada del fuente (SEU)

Puede utilizar el programa de utilidad para entrada del fuente (SEU) para entrar sentencias fuente. El SEU también proporciona información sobre los valores de las distintas plantillas de especificación, así como la posibilidad de comprobar la sintaxis. Para arrancar el SEU, utilice el mandato STRSEU (programa de utilidad para entrada del fuente). Si desea conocer otras formas de arrancar y utilizar el SEU, consulte la publicación *ADTS for AS/400: Source Entry Utility*.

Si a su archivo fuente le da el nombre QRPGLSRC, el SEU automáticamente establece el tipo de fuente en RPGLE cuando inicia la sesión de edición de un miembro nuevo. De lo contrario, es preciso especificar RPGLE al crear el miembro.

Si necesita información sobre los valores que debe entrar después de escribir STRSEU, pulse F4. Aparecerá la pantalla STRSEU, que contiene una lista de parámetros y proporciona los valores por omisión. Si proporciona valores de parámetros antes de efectuar la solicitud, la pantalla aparecerá con estos valores ya colocados.

En el ejemplo siguiente, tecleará sentencias fuente para un programa que imprimirá información sobre los empleados que se encuentra en un archivo maestro. Este ejemplo le muestra cómo:

- Crear una biblioteca
 - Crear un archivo físico fuente
 - Iniciar una sesión de edición del SEU
 - Entrar sentencias fuente
1. Para crear una biblioteca denominada MIBIB, escriba:

```
CRTLIB LIB(MIBIB)
```

El mandato CRTLIB crea una biblioteca llamada MIBIB.

2. Para crear un archivo físico fuente denominado QRPGLSRC, escriba:

```
CRTSRCPF FILE(MIBIB/QRPGLSRC) RCDLEN(112)  
TEXT('Archivo físico fuente para los programas ILE RPG')
```

El mandato CRTSRCPF crea el archivo físico fuente QRPGLSRC en la biblioteca MIBIB.

3. Para iniciar una sesión de edición y crear el miembro fuente EMPRPT, escriba:

```
STRSEU SRCFILE(MIBIB/QRPGLSRC)  
SRCMBR(EMPRPT)  
TYPE(RPGLE) OPTION(2)
```

Al entrar OPTION(2) se indica que se desea iniciar una sesión para un miembro nuevo. El mandato STRSEU crea un miembro EMPRPT nuevo en el archivo QRPGLSRC en la biblioteca MIBIB e inicia una sesión de edición.

La pantalla Edición del SEU aparece tal como se muestra en la Figura 27 en la página 57. Observe que la pantalla se desplaza automáticamente de modo que la posición 6 (para el tipo de especificación) se encuentra en el borde izquierdo.

Utilización del SEU

```

=====
* NOMBRE MÓDULO:  EMPRPT
* ARCHIV RELAC:  MSTEMP  (ARCHIVO FÍSICO)
*                QSYSVRT  (ARCHIVO DE IMPRESORA)
* DESCRIPCIÓN:   Este programa imprime información de empleados
*                del archivo MSTEMP.
=====
FQSYSVRT  0    F  80      PRINTER
FMSTEMP   IP   E          K DISK
D TYPE                    S          8A
D EMPTYTYPE              PR          8A
D CODE                   1A
IEMPREC          01
C                      EVAL      TYPE = EMPTYTYPE(ETYPE)
OPRINT      H    1P              2  6
O
O          H    1P
O
O
O          12 'NOMBRE'
O          34 'NÚM. SERIE'
O          45 'DEPT'
O          56 'TIPO'
O          D    01
O          ENAME          20
O          ENUM          32
O          EDEPT         45
O          TYPE          60
* El procedimiento EMPTYTYPE devuelve una serie que representa
* el tipo de empleado indicado por el parámetro CODE.
P EMPTYTYPE      B
D EMPTYTYPE      PI          8A
D CODE           1A
C
C          SELECT
C          WHEN      CODE = 'M'
C          RETURN    'Gestor'
C          WHEN      CODE = 'R'
C          RETURN    'Normal'
C          OTHER
C          RETURN    'Desconocido'
C          ENDSL
P EMPTYTYPE      E

```

Figura 28. Fuente del miembro EMPRPT

5. Pulse F3 (Salir) para ir a la pantalla Salir. Teclee Y (Sí) para salvar EMPRPT.
Se ha salvado el miembro EMPRPT.

La Figura 29 en la página 59 muestra las DDS a las que hace referencia el fuente de EMPRPT.

```

A*****
A* DESCRIPCIÓN:  Estas son las DDS del archivo físico MSTEMP.      *
A*                Contiene un formato de registro llamado EMPREC.  *
A*                Este archivo contiene un registro para cada      *
A*                de la empresa.                                     *
A*****
A*
A      R EMPREC
A      ENUM          5  0      TEXT('NÚMERO DE EMPLEADO')
A      ENAME         20      TEXT('NOMBRE DE EMPLEADO')
A      ETYPE          1      TEXT('TIPO DE EMPLEADO')
A      EDEPT          3  0      TEXT('DEPARTAMENTO DE EMPLEADO')
A      ENHRS          3  1      TEXT('HRS SEMAN. NORMALES DE EMPLEADO')
A      K ENUM

```

Figura 29. DDS de EMPRPT

Para crear un programa a partir de este fuente utilice el mandato CRTBNDRPG, especificando DFTACTGRP(*NO).

Utilización de sentencias SQL

Se puede acceder a la base de datos DB2 UDB for iSeries® desde un programa ILE RPG intercalando sentencias SQL en el fuente del programa. Utilice las reglas siguientes para entrar las sentencias SQL:

- Teclee las sentencias SQL en la Especificación de cálculo
- Comience las sentencias SQL con el delimitador /EXEC SQL en las posiciones 7-15 (con la barra / en la posición 7)
- Puede empezar a entrar las sentencias SQL en la misma línea que el delimitador inicial
- Utilice el delimitador de línea de continuación (un signo + en la posición 7) para continuar las sentencias en las líneas siguientes
- Utilice el delimitador final /END-EXEC en las posiciones 7-15 (con la barra en la posición 7) para indicar el final de las sentencias SQL.

Nota: las sentencias SQL no pueden sobrepasar la posición 80 en el programa.

La Figura 30 muestra un ejemplo de sentencias SQL intercaladas.

```

...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
C
C                (operaciones de cálculo ILE RPG)
C
C/EXEC SQL      (delimitador inicial)
C+
C+            (líneas de continuación que contienen sentencias SQL)
C+
C  .
C  .
C  .
C/END-EXEC      (delimitador final)
C
C                (operaciones de cálculo ILE RPG)
C

```

Figura 30. Sentencias SQL en un programa ILE RPG

Utilización de sentencias SQL

Debe entrar un mandato diferente para procesar las sentencias SQL. Para obtener más información, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* del **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iseres/infocenter>.

Consulte la publicación *ADTS for AS/400: Source Entry Utility* para obtener más información sobre cómo SEU maneja la comprobación de sintaxis de sentencias SQL.

Capítulo 6. Creación de un programa con el mandato CRTBNDRPG

En este capítulo se muestra cómo crear un programa ILE utilizando el fuente RPG IV con el mandato Crear programa RPG enlazado (CRTBNDRPG). Con este mandato puede crear uno de estos dos tipos de programas ILE:

1. Programas compatibles con OPM con enlace no estático
2. Programas ILE de un solo módulo con enlace estático

La obtención de un programa del primer o del segundo tipo depende de que el parámetro DFTACTGRP de CRTBNDRPG se haya establecido en *YES o en *NO respectivamente.

La creación de un programa del primer tipo genera un programa que se comporta como un programa OPM en cuanto a los ámbitos de apertura, de alteración temporal y RCLRSC. Este alto nivel de compatibilidad se debe en parte a su ejecución en el mismo grupo de activación que los programas OPM, que es el grupo de activación por omisión.

Sin embargo, esta alta compatibilidad conlleva la imposibilidad de utilizar el enlace estático. El enlace estático hace referencia a la capacidad de llamar a procedimientos (de otros módulos o programas de servicio) y utilizar los punteros de dichos procedimientos. La imposibilidad de tener enlace estático significa que no puede:

- Utilizar la operación CALLB en su fuente
- Llamar a un procedimiento de prototipos
- Enlazar con otros módulos durante la creación de programas

La creación de un programa del segundo tipo genera un programa con características ILE como el enlace estático. Al crear el programa, puede especificar el grupo de activación en el que el programa se ejecutará y los módulos para el enlace estático. Además, puede llamar a los procedimientos desde su fuente.

Utilización del mandato CRTBNDRPG

El mandato Crear RPG Enlazado (CRTBNDRPG) crea un objeto de programa a partir del fuente RPG IV en un solo paso. También le permite enlazar otros módulos o programas de servicio mediante un directorio de enlace.

El mandato arranca el compilador ILE RPG y crea un objeto de módulo temporal en la biblioteca QTEMP. Después lo enlaza en un objeto de programa del tipo *PGM. Una vez creado el objeto de programa, se suprime el módulo temporal utilizado para crear el programa.

El mandato CRTBNDRPG es útil si desea crear un objeto de programa a partir de código fuente autónomo (que no requiere que se enlacen módulos), ya que combina los pasos de creación y enlace. Además, le permite crear un programa compatible con OPM.

Nota: si desea conservar el objeto de módulo para enlazarlo con otros módulos en un objeto de programa, debe crear el módulo utilizando el mandato

Utilización del mandato CRTBNDRPG

CRTRPGMOD. Para obtener más información, consulte el “Capítulo 7. Creación de un programa con los mandatos CRTRPGMOD y CRTPGM” en la página 77.

Puede utilizar el mandato CRTBNDRPG de forma interactiva, por lotes o desde un programa CL (Lenguaje de Mandatos). Si utiliza el mandato de forma interactiva y necesita información acerca de los valores que puede utilizar, escriba CRTBNDRPG y pulse F4 (Solicitud). Si necesita ayuda, escriba CRTBNDRPG y pulse F1 (Ayuda).

En la Tabla 5 encontrará un resumen de los parámetros del mandato CRTBNDRPG y sus valores por omisión.

Tabla 5. Parámetros del mandato CRTBNDRPG y sus valores por omisión agrupados por función

Identificación del programa	
PGM(*CURLIB/*CTLSPEC)	Determina el nombre y la biblioteca del programa creado.
SRCFILE(*LIBL/QRPGLESRC)	Identifica el archivo fuente y la biblioteca.
SRCMBR(*PGM)	Identifica el miembro de archivo que contiene las especificaciones fuente.
TEXT(*SRCMBRTXT)	Proporciona una breve descripción del programa.
Creación del programa	
GENLVL(10)	Condiciona la creación del programa a la gravedad del error (0-20).
OPTION(*DEBUGIO)	*DEBUGIO/*NODEBUGIO, determina si se generan puntos de interrupción para las especificaciones de entrada y de salida.
OPTION(*GEN)	*GEN/*NOGEN, determina si se crea el programa.
OPTION(*NOSRCSTMT)	Especifica cómo genera el compilador números de sentencia para la depuración.
DBGVIEW(*STMT)	Especifica el tipo de vista de depuración, si la hay, que se incluirá en el programa.
OPTIMIZE(*NONE)	Determina el nivel de optimización, si lo hay.
REPLACE(*YES)	Determina si el programa debe sustituir al programa existente.
BNDDIR(*NONE)	Especifica el directorio de enlace que se utilizará para la resolución de símbolos.
USRPRF(*USER)	Especifica el perfil de usuario que ejecutará el programa.
AUT(*LIBCRTAUT)	Especifica el tipo de autorización para el programa creado.
TGTRLS(*CURRENT)	Especifica el nivel de release en el que se ejecutará el objeto.
ENBPFRCOL(*PEP)	Especifica si la colección de rendimiento está habilitada.
DEFINE(*NONE)	Especifica los nombres de condición definidos antes de que se inicie la compilación.
PRFDTA(*NOCOL)	Especifica el atributo de datos de perfil de programa.
Listado del compilador	
OUTPUT(*PRINT)	Determina si hay un listado del compilador.
INDENT(*NONE)	Determina si el listado debe tener un sangrado e identifica el carácter para marcarlo.
OPTION(*XREF *NOSECLVL *SHOWCPY *EXPDDS *EXT *NOSHOWSKP *NOSRCSTMT)	Especifica el contenido del listado del compilador.
Opciones de conversión de datos	

Tabla 5. Parámetros del mandato CRTBNDRPG y sus valores por omisión agrupados por función (continuación)

CVTOPT(*NONE)	Especifica cómo se manejarán los diversos tipos de datos de los archivos descritos externamente.
ALWNULL(*NO)	Determina si el programa aceptará los valores de los campos con posibilidad de nulos.
FIXNBR(*NONE)	Determina qué datos decimales que no sean válidos debe corregir el compilador.
Consideraciones de tiempo de ejecución	
DFTACTGRP(*YES)	Identifica si este programa siempre se ejecutará en el grupo de activación por omisión OPM.
OPTION(*DEBUGIO)	*DEBUGIO/*NODEBUGIO, determina si se generan puntos de interrupción para las especificaciones de entrada y de salida.
ACTGRP(QILE)	Identifica el grupo de activación en el que debe ejecutarse el programa.
SRTSEQ(*HEX)	Especifica la tabla de secuencias de ordenación que se utilizará.
LANGID(*JOB RUN)	Se utiliza con SRTSEQ para especificar el identificador de lenguaje para la secuencia de ordenación.
TRUNCNBR(*YES)	Especifica la acción que se lleva a cabo cuando se produce un desbordamiento numérico para campos decimales empaquetados, decimales con zona y binarios en operaciones de formato fijo.
LICOPT(opciones)	Especifica opciones de código interno bajo licencia (LIC).

Vea el “Apéndice C. Los mandatos para crear” en la página 455 para consultar el diagrama de sintaxis y la descripción de los parámetros de CRTBNDRPG.

Creación de un programa para la depuración del fuente

En este ejemplo, creará el programa EMPRPT para poder depurarlo utilizando el depurador del fuente. El parámetro DBGVIEW del mandato CRTBNDRPG o CRTRPGMOD determina qué tipo de datos de depuración se crearán durante la compilación. Este parámetro proporciona seis opciones que le permiten seleccionar qué vistas desea:

- *STMT: le permite visualizar variables y establecer puntos de interrupción en ubicaciones de sentencias utilizando un listado del compilador. Con esta vista no se visualiza el fuente.
- *SOURCE: crea una vista idéntica al fuente de entrada.
- *COPY: crea una vista fuente y una vista que contiene el fuente de cualquier miembro /COPY.
- *LIST: crea una vista similar al listado del compilador.
- *ALL: crea todas las vistas anteriores.
- *NONE: no se crean datos de depuración.

El fuente de EMPRPT se muestra en la Figura 28 en la página 58.

1. Para crear el tipo de objeto:

```
CRTBNDRPG PGM(MIBIB/EMPRPT) DBGVIEW(*SOURCE) DFTACTGRP(*NO)
```

El programa se creará en la biblioteca MIBIB con el mismo nombre que el miembro fuente en el que está basado, es decir, EMPRPT. Tenga en cuenta que, por omisión, se ejecutará en el grupo de activación por omisión, QILE. Este objeto de programa se puede depurar utilizando una vista del fuente.

Utilización del mandato CRTBNDRPG

2. Para depurar el tipo de programa:

STRDBG EMPRPT

La Figura 31 muestra la pantalla que aparece tras entrar el mandato indicado anteriormente.

Visualizar fuente de módulo					
Programa:	EMPRPT	Biblioteca:	MIBIB	Módulo:	EMPRPT
1	*****				
2	* NOMBRE MÓDULO: EMPRPT				
3	* ARCHIV RELAC: MSTEMP (ARCHIVO FÍSICO)				
4	* QSYSPRT (ARCHIVO DE IMPRESORA)				
5	* DESCRIPCIÓN: Este programa imprime información de empleados				
6	* del archivo EMPMST.				
7	*****				
8	FQSYSPRT	O	F	80	PRINTER
9	FMSTEMP	IP	E		K DISK
10					
11	D TYPE		S		8A
12	D EMPTYPE		PR		8A
13	D CODE				1A
14					
15	IEMPREC		01		
					Más...
Depurar . . . _____					
F3=Fin. programa F6=Añadir/Eliminar punto int. F10=Recorrer F11=Ver variable					
F12=Reanudar F17=Observar variable F18=Trabajar con obs. F24=Más teclas					

Figura 31. Pantalla Visualizar fuente de módulo EMPRPT

En esta pantalla (Visualizar fuente de módulo) puede entrar mandatos de depuración para visualizar o cambiar los valores de los campos y establecer puntos de interrupción para controlar el flujo de programa mientras se realiza la depuración.

Para obtener más información sobre la depuración, consulte el “Capítulo 12. Depuración de programas” en la página 195.

Creación de un programa con enlace estático

En este ejemplo creará el programa CALCULAR utilizando CRTBNDRPG al que enlazará un programa de servicio durante la creación del programa.

Suponga que desea enlazar el programa CALCULAR a servicios que ha adquirido para realizar cálculos matemáticos avanzados. El directorio de enlace al que debe enlazar el fuente se denomina MATEM. Este directorio contiene el nombre de un programa de servicio que contiene los diversos procedimientos que componen los servicios.

Para crear el objeto, escriba lo siguiente:

```
CRTBNDRPG PGM(MIBIB/CALCULAR)
          DFTACTGRP(*NO) ACTGRP(GRP1) BNDDIR(MATEM)
```

El fuente se enlazará al programa de servicio especificado en el directorio de enlace MATEM en el momento de crear el programa. Ello significa que las llamadas a los procedimientos del programa de servicio se realizarán en menos tiempo que si fueran llamadas dinámicas.

Utilización del mandato CRTBNDRPG

Cuando se llama al programa, se ejecuta en el grupo de activación mencionado, que es GRP1. El valor por omisión del parámetro ACTGRP en CRTBNDRPG es QILE. Sin embargo, es recomendable que ejecute la aplicación como un grupo único para garantizar que los recursos asociados estén totalmente protegidos.

Nota: DFTACTGRP debe estar establecido en *NO para que pueda entrar un valor para los parámetros ACTGRP y BNDDIR.

Para obtener más información sobre los programas de servicio, consulte el “Capítulo 8. Creación de un programa de servicio” en la página 93.

Creación de un objeto de programa compatible con OPM

En este ejemplo, utilizará el mandato CRTBNDRPG para crear un objeto de programa compatible con OPM a partir del fuente del programa de nóminas, que se muestra en la Figura 32 en la página 66.

1. Para crear el objeto, escriba lo siguiente:

```
CRTBNDRPG PGM(MIBIB/NÓMINAS)
          SRCFILE(MIBIB/QRPGLESRC)
          TEXT('ILE RPG') DFTACTGRP(*YES)
```

El mandato CRTBNDRPG crea el programa NÓMINAS en MIBIB, el cual se ejecutará en el grupo de activación por omisión. Por omisión, se genera un listado del compilador.

Nota: el valor de DFTACTGRP(*YES) es lo que proporciona la compatibilidad con OPM. Este valor también le impide entrar un valor para los parámetros ACTGRP y BNDDIR. Además, si el fuente contiene llamadas de procedimiento enlazadas, se emite un error y finaliza la compilación.

2. Escriba uno de los mandatos CL siguientes para ver el listado que se crea:
 - DSPJOB y después seleccione la opción 4 (*Visualizar archivos en spool*)
 - WRKJOB
 - WRKOUTQ *nombre-cola*
 - WRKSPLF

Utilización del mandato CRTBNDRPG

```

*-----*
* DESCRIPCIÓN: Este programa crea una salida impresa de la paga semanal *
*               de los empleados. *
*-----*
H DATEDIT(*DMY/)
*-----*
* Definiciones de archivo *
*-----*
FTRANSACC IP E K DISK
FEMPLEADO IF E K DISK
FQSYSPRT 0 F 80 PRINTER
*-----*
* Declaraciones de variables *
*-----*
D Paga S 8P 2
*-----*
* Declaraciones de constantes *
*-----*
D Cabecera1 C 'NÚMERO NOMBRE TARIFA H-
D ORAS BONIFIC. PAGA '
D Cabecera2 C ' ' ' -
D ' ' '
*-----*
* Para cada registro del archivo de transacciones (TRANSACC), si se *
* encuentra el empleado, calcular paga de empleado e imprimir detalle. *
*-----*
C TRN_NUMBER CHAIN EMP_REC 99
C IF NOT *IN99
C EVAL (H) Pay = EMP_RATE * TRN_HOURS + TRN_BONUS
C ENDIF
*-----*
* Diseño de informe *
* -- imprimir las líneas de cabecera si está activado 1P *
* -- si se encuentra el registro (el indicador 99 está desactivado) *
* imprimir los detalles de nómina, de lo contrario imprimir un *
* registro de excepción *
* -- imprimir 'FIN DEL LISTADO' cuando LR se active *
*-----*
OQSYSPRT H 1P 2 3
0 35 'REGISTRO DE NOMINAS'
0 *DATE Y 60
0 H 1P 2 60 Cabecera1
0 H 1P 2 60 Cabecera2
0 D N1PN99 2
0 TRN_NUMBER 5
0 EMP_NAME 24
0 EMP_RATE L 33
0 TRN_HOURS L 40
0 TRN_BONUS L 49
0 Paga 60 '$ 0. '
0 D N1P 99 2
0 TRN_NUMBER 5
0 35 '** NO EN ARCHIVO DE EMPLEADOS **'
0 T LR
0 33 'FIN DEL LISTADO'

```

Figura 32. Ejemplo de un programa de cálculo de nóminas

Utilización de un listado del compilador

En este apartado se describe cómo obtener un listado y cómo utilizarlo para que le ayude a:

- Corregir errores de compilación
- Corregir errores durante la ejecución
- Proporcionar documentación para fines de mantenimiento

Vea el “Apéndice D. Listados del compilador” en la página 475 para obtener más información acerca de las distintas partes del listado y un ejemplo de listado completo.

Obtención de un listado del compilador

Para obtener un listado del compilador, especifique OUTPUT(*PRINT) en el mandato CRTBNDRPG o en el mandato CRTRPGMOD. (Este es el valor por omisión.) La especificación OUTPUT(*NONE) suprimirá un listado.

Si se especifica OUTPUT(*PRINT), se obtiene un listado del compilador que consta como *mínimo* de estas secciones:

- Prólogo (resumen de opciones de mandatos)
- Listado fuente, que incluye:
 - Mensajes de diagnóstico incorporado
 - Tabla de campos de comparación (si se utiliza el ciclo del RPG con campos de comparación)
- Mensajes de diagnóstico adicionales
- Posiciones de campo en almacenamiento intermedio de salida
- Tabla de miembros /COPY
- Datos en tiempo de compilación, que incluyen:
 - Registros y tabla de orden de clasificación alternativo o información y tabla de NLSS
 - Registros de conversión de archivos
 - Registros de matrices
 - Registros de tabla
- Resumen de mensajes
- Resumen final
- Informe de generación de código (sólo aparece si hay errores)
- Informe de enlace (sólo se aplica a CRTBNDRPG; solamente aparece si hay errores)

La siguiente información adicional se incluye en los listados del compilador si se especifica el valor apropiado en el parámetro OPTION de cualquier mandato de creación:

***EXPDDS**

Especificaciones de archivos descritos externamente (aparecen en la sección fuente del listado)

***SHOWCPY**

Registros fuente de miembros /COPY (aparecen en la sección fuente del listado)

Utilización de un listado del compilador

***SHOWSKP**

Líneas fuente excluidas por directivas de compilación condicional (aparecen en la sección fuente del listado)

***EXPDDS**

Información de campos clave (sección separada)

***XREF** Lista de referencias cruzadas (sección separada)

***EXT** Lista de referencias externas (sección separada)

***SECLVL**

Texto de mensajes de segundo nivel (aparece en la sección de resumen de mensajes)

Nota: a excepción de *SECLVL y *SHOWSKP, todos los valores anteriores reflejan los valores por omisión en el parámetro OPTION para ambos mandatos de creación. No es necesario cambiar el parámetro OPTION a menos que no desee determinadas secciones del listado o que desee incluir texto de segundo nivel.

La información que contiene un listado del compilador también depende de si se ha especificado *SRCSTMT o *NOSRCSTMT para el parámetro OPTION. Para obtener detalles sobre cómo cambia esta información, consulte *""Cabecera de fuente *NOSRCSTMT""* en la página 482 y *""Cabecera de fuente *SRCSTMT""* en la página 482.

Si se especifican palabras clave de opción de compilación en la especificación de control, las opciones del compilador vigentes aparecerán en la sección de fuente del listado.

Personalización de un listado del compilador

Puede personalizar un listado del compilador de cualquiera o de todos los modos siguientes:

- Personalizar la cabecera de página
- Personalizar el espaciado
- Sangrar operaciones estructuradas

Personalización de una cabecera de página

La información de cabecera de página incluye la línea de información sobre el producto y el título proporcionado por una directiva /TITLE. La línea de información sobre el producto incluye el aviso de copyright de la biblioteca y el compilador ILE RPG, el miembro y la biblioteca del programa fuente, la fecha y la hora en que se creó el módulo y el número de página del listado.

Puede especificar información de cabecera en el listado del compilador utilizando la directiva del compilador /TITLE. Esta directiva le permite especificar el texto que aparecerá en la parte superior de todas las páginas del listado del compilador. Esta información precederá la información normal de cabecera de página. Si la directiva es el primer registro del miembro fuente, esta información también aparecerá en la sección del prólogo.

También puede cambiar el separador de la fecha, el formato de la fecha y el separador de la hora que se utilizan en la cabecera de página y otros recuadros de información del listado. Normalmente, el compilador determina esta información según los atributos de trabajo. Para cambiar alguno de estos elementos, utilice el mandato Cambiar Trabajo (CHGJOB). Después de entrar este mandato, puede:

Utilización de un listado del compilador

- Seleccionar uno de los separadores de fecha siguientes: *SYSVAL, *BLANK, barra inclinada (/), guión (-), punto (.) o coma (,).
- Seleccionar uno de los formatos de fecha siguientes: *SYSVAL, *YMD, *MDY, *DMY o *JUL.
- Seleccionar uno de los separadores de hora siguientes: *SYSVAL, *BLANK, dos puntos (:), coma (,) o punto (.)

Estos valores se utilizan en cualquier parte del listado en que aparezca un campo de fecha o de hora.

Personalización del espaciado

Cada sección de un listado suele comenzar en una página nueva; cada página del listado comienza con la información sobre el producto, a menos que el miembro fuente contenga una directiva /TITLE. Si es así, la información sobre el producto aparece en la segunda línea y el título ocupa la primera línea.

Puede controlar el espaciado y la paginación del listado del compilador mediante la utilización de directivas de compilador /EJECT y /SPACE. La directiva /EJECT fuerza un salto de página. La directiva /SPACE controla el espaciado de las líneas en el listado. Para obtener más información sobre estas directivas, consulte la publicación *ILE RPG Reference*.

Sangrado de operaciones estructuradas

Nota: los cálculos sólo pueden sangrarse si están escritos con la sintaxis tradicional. El compilador RPG no cambia el sangrado de los cálculos de formato libre (entre /FREE y /END-FREE) del listado. Puede sangrar los cálculos de formato libre directamente en el archivo fuente.

Si las especificaciones fuente contienen operaciones estructuradas (como DO-END o IF-ELSE-END), tal vez desee que estas operaciones aparezcan sangradas en el listado fuente. El parámetro INDENT le permite especificar si habrá sangrado, así como el carácter que marcará el sangrado. Si no desea que haya sangrado, especifique INDENT(*NONE); éste es el valor por omisión. Si desea que haya sangrado, especifique dos caracteres como máximo para marcar el sangrado.

Por ejemplo, para indicar que desea sangrar las operaciones estructuradas y marcarlas con una barra vertical (|) seguida de un espacio, especifique INDENT('| ').

Si solicita el sangrado, se elimina parte de la información que aparece normalmente en el listado fuente para que se pueda realizar el sangrado. Las columnas siguientes **no** aparecerán en el listado:

- Núm de DO
- Última actualización
- PÁG./LÍNEA

Si especifica el sangrado y también una vista de depuración del listado, el sangrado no aparecerá en la vista de depuración.

La Figura 33 en la página 70 muestra parte del listado fuente generado con el sangrado. La marca de sangrado es '| '.

Utilización de un listado del compilador

Núm.	<----- Especificaciones de fuente -----><--- Comentarios --->										ID Núm.
línea1....+....2....+<----- 26 - 35 ----->....4....+....5....+....6....+....7....+....8....+....9....+..10										fuen secuenc
33	C*****										002000
34	C* MAINLINE *										002100
35	C*****										002200
36		WRITE		FOOT1						002300	
37		WRITE		HEAD						002400	
38		EXFMT		PROMPT						002500	
39	C*										002600
40		DOW		NOT *IN03						002700	
41	CSTKEY	SETLL		CMLREC2		----	20			002800	
42		IF		*IN20						002900	
43		MOVE		'1'				*IN61		003000	
44		ELSE								003100	
45		EXSR		SFLPRC						003200	
46		END								003300	
47		IF		NOT *IN03						003400	
48		IF		*IN04						003500	
49		IF		*IN61						003600	
50		WRITE		FOOT1						003700	
51		WRITE		HEAD						003800	
52		ENDIF								003900	
53		EXFMT		PROMPT						004000	
54		ENDIF								004100	
55		ENDIF								004200	
56		ENDDO								004300	
57	C*										004500
58		SETON						LR----		004600	

Figura 33. Ejemplo de parte fuente del listado con sangrado

Corrección de errores de compilación

Las secciones principales de un listado del compilador que son útiles para corregir los errores de compilación son las siguientes:

- La sección fuente
- La sección Mensajes adicionales
- La sección de tabla /COPY
- Las diversas secciones de resumen.

Los mensajes de diagnóstico incorporado, que se encuentran en la sección fuente, indican los errores que el compilador puede marcar inmediatamente. Otros errores se indican después de recibir información adicional durante la compilación. Los mensajes que indican la existencia de estos errores se encuentran en la sección fuente y en la sección Mensajes adicionales.

Para ayudarle a corregir los errores de compilación, tal vez desee incluir texto de mensajes de segundo nivel en el listado (especialmente si hace poco tiempo que trabaja con RPG). Para ello, especifique OPTION(*SECLVL) en cualquier mandato de creación. De este modo añadirá texto de segundo nivel a los mensajes listados en el resumen de mensajes.

Finalmente, recuerde que un listado del compilador es un registro escrito del programa. Por lo tanto, si encuentra errores durante la comprobación de su programa, puede utilizar el listado para comprobar que el fuente se haya codificado del modo que tenía previsto. Las partes del listado, además de las sentencias fuente, que tal vez desee comprobar son las siguientes:

- Tabla de campos de comparación
Si está utilizando el ciclo RPG con campos de comparación, puede utilizar esta tabla para comprobar que todos los campos de comparación tienen la longitud correcta y se encuentran en las posiciones correctas.
- Posiciones de almacenamiento intermedio de salida

Utilización de un listado del compilador

Contienen las posiciones inicial y final junto con el texto literal o los nombres de campo. Utilice esta información para comprobar si hay errores en las especificaciones de salida.

- Datos de tiempo de compilación

Se listan los registros y ALTSEQ FTRANS y las tablas. También se listan la información y las tablas NLSS. Las tablas y las matrices se identifican explícitamente. Utilice esta información para confirmar que ha especificado los datos de tiempo de compilación en el orden correcto y que ha especificado los valores correctos para los parámetros SRTSEQ y LANGID en el compilador.

Utilización de mensajes de diagnóstico incorporado

Hay dos tipos de mensajes de diagnóstico incorporado: apuntadores y no apuntadores. Los mensajes apuntadores indican exactamente dónde se ha producido el error. La Figura 34 muestra un ejemplo de mensajes de diagnóstico incorporado apuntadores.

Núm. línea	<----- Especificaciones de fuente ----->	<--- Comentarios --->	Do	Pág.	Fecha	ID Núm.
63	C	SETOFF	12	aabb	cccccc	003100
*RNF5051 20 a	003100	Entrada de indicador resultante no válida; toma como valor por omisión blancos.				
*RNF5051 20 b	003100	Entrada de indicador resultante no válida; toma como valor por omisión blancos.				
*RNF5053 30 c	003100	Entrada de indicadores resultantes en blanco para operación especificada.				

Figura 34. Ejemplo de mensajes de diagnóstico incorporado apuntadores

En este ejemplo, se ha situado de forma incorrecta un indicador en las posiciones 72 y 73 en lugar de en las posiciones 71 y 72 ó 73 y 74. Los tres apuntadores "aa", "bb" y "cccccc" indican las partes de la línea donde hay errores. Las columnas reales se resaltan con variables que se explican en los mensajes. En este caso, el mensaje RNF5051 indica que los campos marcados con "aa" y "bb" no contienen un indicador válido. Puesto que no hay ningún indicador válido, el compilador supone que los campos están en blanco. Sin embargo, puesto que la operación SETOFF necesita un indicador, se produce otro error, tal como indica el campo "cccccc" y el mensaje RNF5053.

Los errores se listan en el orden en el que se encuentran. Como norma general, debe intentar corregir en primer lugar los primeros errores de gravedad 30 y 40, ya que a menudo estos son la causa de otros errores.

Los mensajes de diagnóstico incorporado no apuntadores también indican la existencia de errores. Sin embargo, no se emiten inmediatamente después de la línea que contiene el error. La Figura 35 muestra un ejemplo de mensajes de diagnóstico incorporado no apuntadores.

Núm. línea	<----- Especificaciones de fuente ----->	<--- Comentarios --->	Do Pág.	Fecha	ID Núm.
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....10		Núm. Lin.	modif.	fuen secuenc
1 D FLD1	S	+5	LIKE(FLD2)		000100
2 D FLD2	S	D			000200
*RNF3479 20	1 000100	No se permite un ajuste de longitud para el campo del tipo de datos especificado.			

Figura 35. Ejemplo de mensajes de diagnóstico incorporado no apuntadores

En este ejemplo, se define FLD1 como FLD2 con una longitud de 5 bytes más. Posteriormente, FLD2 se define como una fecha, con lo cual el ajuste de longitud de la definición de FLD1 no es válido. El mensaje RNF3479 se emite apuntando a la línea de listado 1. Tenga en cuenta que también se proporciona el número de

Utilización de un listado del compilador

secuencia del SEU (000100) para ayudarle a buscar con mayor rapidez la línea del fuente que contiene el error. (El número de secuencia del SEU también puede encontrarse en la línea de listado 1.)

Utilización de mensajes de diagnóstico adicionales

La sección Mensajes de diagnóstico adicionales identifica los errores que surgen cuando se consideran una o más líneas de código juntas. Estos mensajes no aparecen en el código donde se ha producido el problema; por lo general, en el momento de la comprobación de dicha parte del fuente el compilador no sabe que se ha producido un problema. Sin embargo, siempre que sea posible, la línea del mensaje contendrá o bien el número de línea y el número de secuencia del SEU del listado o bien el número de sentencia de una línea del fuente que esté relacionada con el mensaje.

Examen de un listado del compilador utilizando el SEU

La sesión Dividir/Examinar del SEU (F15) le permite examinar un listado del compilador en la cola de salida. Puede revisar el resultado de una compilación anterior mientras efectúa las modificaciones necesarias en el código fuente.

Mientras examina el listado del compilador, puede buscar errores y corregir las sentencias fuente que los contienen. Para buscar los errores, escriba F *ERR en la línea de mandatos del SEU de la sesión de examinar. Se resaltará la línea que tenga el primer (o el siguiente) error y en la parte inferior de la pantalla se visualizará el texto de primer nivel del mismo mensaje. Puede ver el texto de segundo nivel situando el cursor sobre el mensaje y pulsando F1 (Ayuda).

Si es posible, los mensajes de error del listado indican el número de secuencia del SEU de la línea que contiene el error. El número de secuencia se encuentra justo antes del texto del mensaje.

Para obtener completa información sobre cómo examinar un listado del compilador, consulte la publicación *ADTS for AS/400: Source Entry Utility*.

Corrección de errores durante la ejecución

La sección fuente del listado también resulta de utilidad para corregir los errores durante la ejecución. Muchos mensajes de error durante la ejecución indican el número de la sentencia en la que se ha producido el error en cuestión.

Si se especifica `OPTION(*NOSRCSTMT)`, el número de línea que aparece en la parte *izquierda* del listado del compilador corresponde al número de sentencia del mensaje de error durante la ejecución. El número de ID fuente y el número de secuencia del SEU situados en la parte *derecha* del listado del compilador identifican el registro y el miembro fuente. Puede utilizarlos juntos, sobre todo si está editando el fuente utilizando el SEU, para determinar qué línea es preciso examinar.

Si se especifica `OPTION(*SRCSTMT)`, el número de sentencia que aparece en la parte *derecha* del listado del compilador corresponde al número de sentencia del mensaje de error durante la ejecución. Si la sentencia es del miembro fuente principal, coincide con la sentencia de la parte *izquierda* del listado del compilador, así como con el número de secuencia del SEU.

Si tiene un miembro `/COPY`, puede buscar el número de ID del fuente del archivo real en la tabla de miembros `/COPY` al final del listado. Para obtener un ejemplo de una tabla de miembros `/COPY`, consulte el apartado "Tabla de miembros `/COPY`" en la página 484.

Coordinación de las opciones de listado con las opciones de vista de depuración

Corregir errores durante la ejecución a menudo supone depurar un programa. Las consideraciones siguientes pueden ayudarle a depurar su programa:

- Si utiliza el depurador del fuente para depurar el programa, dispondrá de una variada gama de vistas de depuración: *STMT, *SOURCE, *LIST, *COPY, *ALL.
- Si piensa utilizar un listado del compilador como ayuda para la depuración, puede obtener uno especificando OUTPUT(*PRINT). Un listado es importante si tiene la intención de realizar la depuración utilizando una vista de sentencias (*STMT), ya que los números de sentencia para establecer puntos de interrupción son los indicados en el listado fuente. Los números de sentencia figuran en la columna titulada Número de línea si se especifica OPTION(*NOSRCSTMT) y en la columna titulada Número de sentencia si se especifica OPTION(*SRCSTMT).
- Si sabe que la depuración será larga, tal vez desee compilar el fuente con DBGVIEW(*ALL), OUTPUT(*PRINT) y OPTION(*SHOWCPY). Ello le permitirá utilizar una vista de listado o una vista del fuente e incluirá los miembros /COPY.
- Si especifica DBGVIEW(*LIST), la información que tendrá disponible durante la depuración dependerá de lo que haya especificado en el parámetro OPTION. La vista incluirá los miembros /COPY y los archivos descritos externamente sólo si especifica OPTION(*SHOWCPY *EXPDDS); estos son los valores por omisión.

Utilización de un listado del compilador para el mantenimiento

Un listado del compilador de un programa sin errores se puede utilizar como documentación cuando:

- se enseñe el programa a un programador nuevo,
- se actualice el programa en el futuro.

En cualquier caso, es aconsejable tener un listado completo, es decir, generado con OUTPUT(*PRINT) y con OPTION(*XREF *SHOWCPY *EXPDDS *EXT *SHOWSKP).

Nota: excepto para *SHOWSKP, éste es el valor por omisión de cada uno de estos parámetros en ambos mandatos de creación.

La sección Prólogo del listado es de especial importancia para el mantenimiento del programa. Esta sección le informa de lo siguiente:

- Quién compiló el módulo o el programa
- Qué fuente se utilizó para generar el módulo o el programa
- Qué opciones se utilizaron al compilar el módulo o el programa

Es posible que necesite saber las opciones de los mandatos (por ejemplo, la vista de depuración seleccionada o el directorio de enlace utilizado) cuando realice modificaciones en el programa en el futuro.

Las especificaciones siguientes para el parámetro OPTION proporcionan información adicional como se indica a continuación:

- *SHOWCPY y *EXPDDS proporcionan una descripción completa del programa, incluyendo todas las especificaciones de los miembros /COPY, y las especificaciones generadas a partir de los archivos descritos externamente.
- *SHOWSKP le permite ver las sentencias omitidas por el compilador como consecuencia de las directivas /IF, /ELSEIF, /ELSE o /EOF.

Utilización de un listado del compilador

- *XREF le permite comprobar la utilización de los archivos, campos e indicadores en el módulo o programa.
- *EXT le permite ver qué procedimiento y campos ha importado o exportado el módulo o programa. También identifica los archivos reales que se utilizaron para generar las descripciones de los archivos descritos externamente y las estructuras de datos.

Acceso al área de datos RETURNCODE

Los mandatos CRTBNDRPG y CRTRPGMOD (consulte el apartado “Utilización del mandato CRTRPGMOD” en la página 78) crean y actualizan un área de datos con el estado de la última compilación. Esta área de datos se llama RETURNCODE, tiene 400 caracteres de longitud y se encuentra en la biblioteca QTEMP.

Para acceder al área de datos RETURNCODE, especifique RETURNCODE en el factor 2 de una sentencia *DTAARA DEFINE.

El área de datos RETURNCODE tiene el formato siguiente:

Byte	Contenido y significado
1	Para CRTRPGMOD, el carácter '1' significa que se ha creado un módulo en la biblioteca especificada. Para CRTBNDRPG, el carácter '1' significa que se ha creado un módulo con el mismo nombre que el programa en QTEMP.
2	El carácter '1' significa que la compilación ha presentado anomalías a causa de errores del compilador.
3	El carácter '1' significa que la compilación ha presentado anomalías a causa de errores en el fuente.
4	No establecido. Siempre '0'.
5	El carácter '1' significa que no se ha llamado al conversor porque se había especificado OPTION(*NOGEN) en el mandato CRTRPGMOD o CRTBNDRPG o porque la compilación ha presentado anomalías antes de llamar al conversor.
6-10	Número de sentencias fuente
11-12	Nivel de gravedad del mandato
13-14	Gravedad máxima de los mensajes de diagnóstico
15-20	Número de errores detectados en el módulo (CRTRPGMOD) o en el programa (CRTBNDRPG).
21-26	Fecha de compilación
27-32	Hora de compilación
33-100	No establecido. Siempre está en blanco
101-110	Nombre del módulo (CRTRPGMOD) o del programa (CRTBNDRPG).
111-120	Nombre de la biblioteca del módulo (CRTRPGMOD) o de la biblioteca del programa (CRTBNDRPG).
121-130	Nombre del archivo fuente
131-140	Nombre de la biblioteca del archivo fuente
141-150	Nombre del miembro del archivo fuente

Acceso al área de datos RETURNCODE

151-160	Nombre del archivo del listado del compilador
161-170	Nombre de la biblioteca del listado del compilador
171-180	Nombre del miembro del listado del compilador
181-329	No establecido. Siempre está en blanco
330-334	Tiempo total de compilación transcurrido hasta la décima de segundo más cercana (o -1 si se produce un error mientras se calcula este tiempo)
335	No establecido. Siempre está en blanco
336-340	Tiempo de compilación transcurrido hasta la décima de segundo más cercana (o -1 si se produce un error mientras se calcula este tiempo)
341-345	Tiempo de conversión transcurrido hasta la décima de segundo más cercana (o -1 si se produce un error mientras se calcula este tiempo)
346-379	No establecido. Siempre está en blanco
380-384	Tiempo total de compilación de CPU hasta la décima de segundo más cercana
385	No establecido. Siempre está en blanco
386-390	Tiempo de CPU utilizado por el compilador hasta la décima de segundo más cercana
391-395	Tiempo de CPU utilizado por el conversor hasta la décima de segundo más cercana
396-400	No establecido. Siempre está en blanco

Acceso al área de datos RETURNCODE

Capítulo 7. Creación de un programa con los mandatos CRTRPGMOD y CRTPGM

El proceso de dos pasos de creación de un programa consiste en compilar el fuente en módulos utilizando el mandato CRTRPGMOD y después enlazar uno o varios objetos de módulo en un programa con el mandato CRTPGM. Con este proceso puede crear módulos permanentes. Esto, a su vez, le permite dividir una aplicación en módulos sin volver a compilar toda la aplicación. También le permite volver a utilizar el mismo módulo en distintas aplicaciones.

En este capítulo se indica cómo:

- Crear un objeto de módulo a partir del fuente RPG IV
- Enlazar módulos en un programa utilizando CRTPGM
- Leer un listado del enlazador
- Modificar un módulo o programa

Creación de un objeto de módulo

Un **módulo** es un objeto no ejecutable (del tipo *MODULE) que es la salida de un compilador ILE. Es el bloque básico de creación de un programa ILE.

Un módulo ILE RPG consta de uno o más procedimientos y de los bloques de control de archivo y almacenamiento estático que utilizan todos los procedimientos del módulo. Los procedimientos que componen un módulo ILE RPG son:

- un **procedimiento principal** opcional compuesto por el conjunto de especificaciones H, F, D, I, C y O que inician el fuente. El procedimiento principal tiene su propia semántica LR y ciclo de lógica; ninguno de los cuales resulta afectado por los de los demás módulos ILE RPG del programa.
- cero o más **subprocedimientos**, los cuales se codifican en especificaciones P, D y C. Los subprocedimientos no utilizan el ciclo RPG. Un subprocedimiento puede tener almacenamiento local disponible únicamente para que lo utilice el propio subprocedimiento.

Al procedimiento principal (si está codificado) siempre pueden llamarle otros módulos del programa. Los subprocedimientos pueden ser locales al módulo o exportados. Si son locales, únicamente pueden llamarlos otros procedimientos del módulo; si son exportados desde el módulo, pueden llamarlos todos los procedimientos del programa.

La creación de módulos consiste en la compilación de un miembro fuente y, si esta acción es satisfactoria, la creación de un objeto *MODULE. El objeto *MODULE incluye una lista de las importaciones y exportaciones a las que se hace referencia en el módulo. También contiene datos de depuración si lo solicita en la compilación.

Un módulo no se puede ejecutar solo. Debe enlazar uno o varios módulos para crear un objeto de programa (del tipo *PGM) que pueda ejecutarse. También puede enlazar uno o varios módulos para crear un objeto de programa de servicio (del tipo *SRVPGM). A continuación podrá acceder a los procedimientos que hay en los módulos enlazados mediante llamadas de los procedimientos estáticos.

Creación de un objeto de módulo

La posibilidad de combinar módulos le permite:

- Volver a utilizar partes del código. Generalmente esto crea programas más pequeños. Los programas más pequeños ayudan a mejorar el rendimiento y facilitan la depuración.
- Mantener el código compartido reduciendo las posibilidades de generar errores en otras partes del programa general.
- Gestionar programas grandes de una forma más eficaz. Los módulos le permiten dividir su antiguo programa en partes que pueden gestionarse por separado. Si es necesario ampliar el programa, sólo tendrá que volver a compilar los módulos que se han modificado.
- Crear programas en varios lenguajes en los que podrá enlazar módulos escritos en el lenguaje más adecuado para la tarea que se ha de realizar.

Para obtener más información sobre el concepto de los módulos, consulte la publicación *ILE Concepts*.

Utilización del mandato CRTRPGMOD

Puede crear un módulo utilizando el mandato Crear Módulo RPG (CRTRPGMOD). Este mandato se puede utilizar de forma interactiva, como parte de una corriente de entrada por lotes o en un programa CL (Lenguaje de Control).

Si utiliza el mandato de forma interactiva y necesita información acerca de los valores que puede utilizar, escriba CRTRPGMOD y pulse F4 (Solicitud). Si necesita ayuda, escriba CRTRPGMOD y pulse F1 (Ayuda).

En la Tabla 6 figuran los parámetros del mandato CRTRPGMOD y sus valores por omisión proporcionados por el sistema. El diagrama de sintaxis del mandato y la descripción de los parámetros se encuentran en el “Apéndice C. Los mandatos para crear” en la página 455.

Tabla 6. Parámetros del mandato CRTRPGMOD y sus valores por omisión agrupados por funciones

Identificación de módulos	
MODULE(*CURLIB/*CTLSPEC)	Determina el nombre y la biblioteca del módulo creado
SRCFILE(*LIBL/QRPGLESRC)	Identifica el archivo fuente y la biblioteca
SRCMBR(*MODULE)	Identifica el miembro de archivo que contiene las especificaciones fuente
TEXT(*SRCMBRTXT)	Proporciona una breve descripción del módulo.
Creación de módulos	
GENLVL(10)	Condiciona la creación de módulos a la gravedad del error (0-20).
OPTION(*DEBUGIO)	*DEBUGIO/*NODEBUGIO, determina si se generan puntos de interrupción para las especificaciones de entrada y salida
OPTION(*GEN)	*GEN/*NOGEN determina si se crea el módulo.
OPTION(*NOSRCSTMT)	Especifica cómo genera el compilador los números de sentencia para la depuración
DBGVIEW(*STMT)	Especifica el tipo de vista de depuración, si la hay, que se incluirá en el programa.
OPTIMIZE(*NONE)	Determina el nivel de optimización, si lo hay.
REPLACE(*YES)	Determina si el módulo debe sustituir al módulo existente.
AUT(*LIBCRTAUT)	Especifica el tipo de autorización para el módulo creado
TGTRLS(*CURRENT)	Especifica el nivel de release en el que se ejecutará el objeto.

Tabla 6. Parámetros del mandato CRTRPGMOD y sus valores por omisión agrupados por funciones (continuación)

BNDDIR(*NONE)	Especifica el directorio de enlace que se utilizará para la resolución de símbolos.
ENBPFCOL(*PEP)	Especifica si está habilitado el grupo de rendimiento
DEFINE(*NONE)	Especifica los nombres de las condiciones que se definen antes de que empiece la compilación
PRFDTA(*NOCOL)	Especifica el atributo de datos del perfil de programa
Listado del compilador	
OUTPUT(*PRINT)	Determina si hay un listado del compilador.
INDENT(*NONE)	Determina si el listado debe tener un sangrado e identifica el carácter para marcarlo.
OPTION(*XREF *NOSECLVL *SHOWCPY *EXPDDS *EXT *NOSHOWSKP *NOSRCSTMT)	Especifica el contenido del listado del compilador.
Opciones de conversión de datos	
CVTOPT(*NONE)	Especifica cómo se manejarán los diversos tipo de datos de los archivos descritos externamente.
ALWNULL(*NO)	Determina si el módulo aceptará los valores de los campos con posibilidad de nulos.
FIXNBR(*NONE)	Determina qué datos decimales no válidos debe arreglar el compilador
Consideraciones de tiempo de ejecución	
SRTSEQ(*HEX)	Especifica la tabla de secuencias de ordenación que se utilizará.
OPTION(*DEBUGIO)	*DEBUGIO/*NODEBUGIO, determina si se generan puntos de interrupción para las especificaciones de entrada y salida
LANGID(*JOBRUN)	Se utiliza con SRTSEQ para especificar el identificador de lenguaje para la secuencia de ordenación.
TRUNCNBR(*YES)	Especifica la acción a llevar a cabo cuando se produce un desbordamiento numérico para campos decimales empaquetados, decimales con zona y binarios en operaciones de formato fijo.
LICOPT(opciones)	Especifica las opciones del Código interno bajo licencia

Cuando se solicita, el mandato CRTRPGMOD crea un listado del compilador que en su mayor parte es idéntico al listado generado por el mandato CRTBNDRPG. (El listado creado por CRTRPGMOD no tiene nunca una sección de enlace.)

Si desea obtener más información sobre el uso del listado del compilador, consulte el apartado “Utilización de un listado del compilador” en la página 67. En el “Apéndice D. Listados del compilador” en la página 475 se proporciona un listado del compilador de ejemplo.

Creación de un módulo NOMAIN

En este ejemplo, se crea un objeto de módulo NOMAIN, TRANSSVC, utilizando el mandato CRTRPGMOD y sus valores por omisión. TRANSSVC contiene procedimientos de prototipos que realizan los servicios de transacción para procedimientos de otros módulos. El fuente de TRANSSVC se muestra en la Figura 36 en la página 81. Los prototipos para los procedimientos de TRANSSVC se almacenan en un miembro /COPY, como se muestra en la Figura 37 en la página 82.

1. Para crear un objeto de módulo, escriba lo siguiente:

Creación de un objeto de módulo

```
CRTRPGMOD MODULE(MYLIB/TRANSSVC) SRCFILE(MYLIB/QRPGLESRC)
```

El módulo se creará en la biblioteca MIBIB con el nombre especificado en el mandato, TRANSSVC. El fuente del módulo es el miembro fuente TRANSSVC del archivo QRPGLESRC de la biblioteca MIBIB.

Un módulo que contenga NOMAIN se enlaza a otro módulo mediante los mandatos siguientes:

- a. mandato CRTPGM
 - b. mandato CRTSRVPGM
 - c. mandato CRTBNDRPG donde el módulo NOMAIN se incluye en un directorio de enlace.
2. Una vez enlazado, este objeto de módulo se puede depurar utilizando una vista de sentencia. También se generará un listado del compilador para el módulo.
 3. Escriba uno de los mandatos CL siguientes para ver el listado del compilador.
 - DSPJOB y después seleccione la opción 4 (*Visualizar archivos en spool*)
 - WRKJOB
 - WRKOUTQ *nombre de cola*
 - WRKSPLF

```

=====
* NOMBRE MÓDULO:  TRANSSVC (Servicios de transacción)
* ARCHIVOS RELACIONADOS:  N/A
* FUENTE RELACIONADO:  TRANSRPT
* PROCEDIMIENTOS EXPORTADOS:  Trans_Inc  -- calcula las ganancias
*   de la transacción utilizando los datos de los campos de la
*   lista de parámetros. Devuelve un valor al llamador después de
*   realizar los cálculos.
*
*   Prod_Name -- recupera el nombre del producto basándose en
*   el parámetro de entrada que tiene el número del producto.
=====

* Este módulo sólo contiene subprocedimientos; es un módulo NOMAIN.
H  NOMAIN
-----
* Extraer los prototipos del miembro /COPY
-----
/COPY TRANSP
-----
* Subprocedimiento Trans_Inc
-----
P  Trans_Inc      B      EXPORT
D  Trans_Inc      PI      11P 2
D  ProdNum        10P 0   VALUE
D  Quantity       5P 0    VALUE
D  Discount       2P 2    VALUE
D  Factor         S      5P 0
*
C      SELECT
C      WHEN      ProdNum = 1
C      EVAL      Factor = 1500
C      WHEN      ProdNum = 2
C      EVAL      Factor = 3500
C      WHEN      ProdNum = 5
C      EVAL      Factor = 20000
C      WHEN      ProdNum = 8
C      EVAL      Factor = 32000
C      WHEN      ProdNum = 12
C      EVAL      Factor = 64000
C      OTHER
C      EVAL      Factor = 0
C      ENDSL
C      RETURN      Factor * Quantity * (1 - Discount)
P  Trans_Inc      E

```

Figura 36. Fuente para el miembro TRANSSVC (Pieza 1 de 2)

Creación de un objeto de módulo

```
*-----
* Subprocedimiento Prod_Name
*-----
P Prod_Name      B      EXPORT
D Prod_Name      PI      40A
D ProdNum        10P 0    VALUE
*
C                SELECT
C                WHEN     ProdNum = 1
C                RETURN   'Grande'
C                WHEN     ProdNum = 2
C                RETURN   'Super'
C                WHEN     ProdNum = 5
C                RETURN   'Supergrande'
C                WHEN     ProdNum = 8
C                RETURN   'Superextra'
C                WHEN     ProdNum = 12
C                RETURN   'Increíblemente superextra'
C                OTHER
C                RETURN   '***Desconocido***'
C                ENDSL
P Prod_Name      E
```

Figura 36. Fuente para el miembro TRANSSVC (Pieza 2 de 2)

```
* Prototipo de Trans_Inc
D Trans_Inc      PR      11P 2
D Prod           10P 0    VALUE
D Quantity       5P 0    VALUE
D Discount       2P 2    VALUE

* Prototipo de Prod_Name
D Prod_Name      PR      40A
D Prod           10P 0    VALUE
```

Figura 37. Fuente para el miembro /COPY TRANSP

Creación de un módulo para la depuración del fuente

En este ejemplo, se crea un objeto de módulo ILE RPG que se puede depurar con el depurador del fuente. El módulo TRANSRPT contiene un procedimiento principal que dirige el proceso del informe. Llama a los procedimientos de TRANSSVC a realizar ciertas tareas necesarias. El fuente de este módulo se muestra en la Figura 38 en la página 83.

Para crear un objeto de módulo, escriba lo siguiente:

```
CRTRPGMOD MODULE(MYLIB/TRANSRPT) SRCFILE(MYLIB/QRPGLESRC)
          DBGVIEW(*SOURCE)
```

El módulo se crea en la biblioteca MIBIB con el mismo nombre que el miembro fuente en el que está basado, es decir, TRANSRPT. Este objeto de módulo se puede depurar con una vista del fuente. Si desea obtener información sobre las otras vistas disponibles, consulte el apartado “Preparación de un programa para su depuración” en la página 198.

Se generará un listado del compilador para el módulo TRANSRPT.

```

=====
* NOMBRE MÓDULO:   TRANSRPT
* ARCHIVOS RELACIONADOS:  TRNSDTA (PF)
* FUENTE RELACIONADO:   TRANSSVC (Servicios de transacción)
* PROCEDIMIENTO EXPORTADO:  TRANSRPT
*   El procedimiento TRANSRPT lee cada registro de transacción
*   almacenado en el archivo físico TRNSDTA. Llama al
*   subprocedimiento Trans_Inc que realiza los cálculos y
*   devuelve un valor. A continuación, llama a Prod_Name
*   para determinar el nombre del producto. Después, TRANSRPT
*   imprime el registro de transacción.
=====
FTRNSDTA  IP  E          DISK
FQSYSPRT  0  F  80      PRINTER      OFLIND(*INOF)
/COPY QRPGL,TRANSP
* Definir la versión legible del nombre del producto como el
* valor de retorno del procedimiento 'Prod_Name'
D  ProdName  S          30A
D  Income    S          10P 2
D  Total     S          +5      LIKE(Income)
*
ITRNSREC      01
* Calcular las ganancias utilizando el subprocedimiento Trans_Inc
C          EVAL      Income = Trans_Inc(PROD : QTY : DISC)
C          EVAL      Total = Total + Income
* Localizar el nombre del producto
C          EVAL      ProdName = Prod_Name(PROD)
OQSYSPRT  H  1P          1
O          OR  OF
O
O          12 'Nombre del producto'
O          40 'Cantidad'
O          54 'Ganancias'
OQSYSPRT  H  1P          1
O          OR  OF
O
O          30 '-----+'
O          -----+
O          -----'
O          40 '-----'
O          60 '-----'
OQSYSPRT  D  01          1
O          ProdName      30
O          QTY            1  40
O          Income         1  60
OQSYSPRT  T  LR          1
O          'Total: '
O          Total          1

```

Figura 38. Fuente del módulo TRANSRPT

La DDS para el archivo TRNSDTA se muestra en la Figura 39. El miembro /COPY se muestra en la Figura 37 en la página 82.

```

A*****
A* ARCHIVOS RELACIONADOS:  TRANSRPT
A* DESCRIPCIÓN:   Este es el archivo físico TRNSDTA. Tiene un
A*               solo formato de registro llamado TRNSREC.
A* ARCHIVO TRANSACCIÓN COMPONENTES -- TRNSDTA
A      R TRNSREC
A      PROD      10S 0      TEXT('Producto')
A      QTY        5S 0      TEXT('Cantidad')
A      DISCOUNT  2S 2      TEXT('Descuento')

```

Figura 39. DDS de TRNSDTA

Ejemplos adicionales

Para ver más ejemplos de la creación de módulos, consulte los apartados siguientes:

- “Programa de servicio de ejemplo” en la página 96 para ver un ejemplo de la creación de un módulo para un programa de servicio.
- “Enlace a un programa” en la página 100. para ver un ejemplo de la creación de un módulo que se utilizará con un programa de servicio.
- “Gestión del propio almacenamiento dinámico utilizando las API enlazables ILE” en la página 122, para ver un ejemplo de la creación de un módulo para asignar almacenamiento de forma dinámica para una matriz durante la ejecución.
- “Ejemplo de fuente para ejemplos de depuración” en la página 245 para ver un ejemplo de creación de un módulo RPG y C para utilizarlo en un programa de depuración de ejemplo.

Comportamiento de los módulos ILE RPG enlazados

En ILE RPG, el *procedimiento principal* es el límite del ámbito de la semántica LR y del ciclo RPG. El *módulo* es el límite del ámbito de archivos abiertos.

En un programa ILE, puede haber varios ciclos RPG activos; para cada módulo RPG que tenga un procedimiento principal sólo hay un ciclo RPG. Los ciclos son independientes; activar LR en un procedimiento principal no afecta el ciclo de otro.

Mandatos CL relacionados

Los mandatos CL siguientes se pueden utilizar con módulos:

- Visualizar Módulo (DSPMOD)
- Cambiar Módulo (CHGMOD)
- Suprimir Módulo (DLTMOD)
- Trabajar con Módulos (WRKMOD)

Si desea más información sobre estos mandatos, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>.

Enlace de módulos a un programa

Enlace es el proceso de creación de un programa ILE ejecutable mediante la combinación de uno o varios módulos y programas de servicio opcionales y la resolución de los símbolos que se transfieren entre ellos. El código del sistema que realiza esta operación de combinación y resolución recibe el nombre de **enlazador** en el sistema iSeries.

Como parte del proceso de enlace, se debe identificar a un procedimiento como el procedimiento de arranque o **procedimiento de entrada de programa**. Cuando se llama a un programa, el procedimiento de entrada de programa recibe los parámetros de la línea de mandatos y el control inicial del programa. El código del usuario asociado con el procedimiento de entrada de programa es el procedimiento de entrada del usuario.

Enlace de módulos a un programa

Si un módulo ILE RPG contiene un procedimiento principal, contiene también implícitamente un procedimiento de entrada de programa. Por lo tanto, todo módulo ILE RPG puede especificarse como el módulo de entrada siempre y cuando no sea un módulo NOMAIN.

La Figura 40 ilustra la estructura interna de un objeto de programa. Muestra el objeto de programa TRPT, que se ha creado mediante el enlace de dos módulos TRANSRPT y TRANSSVC. TRANSRPT es el módulo de entrada.

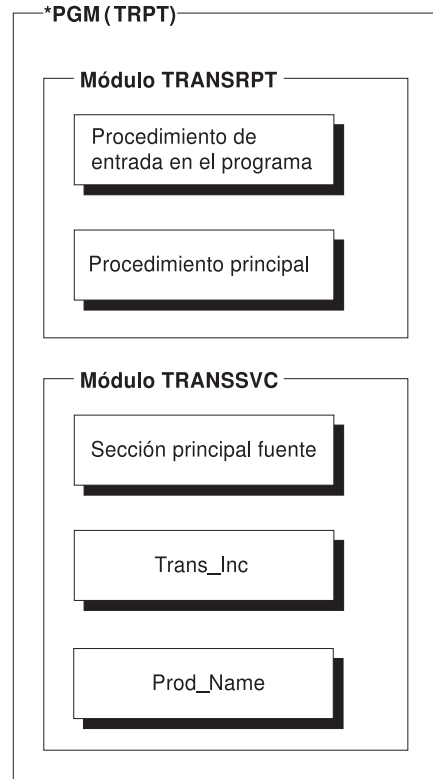


Figura 40. Estructura del programa TRPT

En un objeto enlazado, los procedimientos pueden relacionarse entre ellos mediante llamadas a procedimientos estáticos. Estas llamadas enlazadas son más rápidas que las llamadas externas. Por lo tanto, una aplicación que conste de un solo programa enlazado con varias llamadas enlazadas debería ejecutarse más rápidamente que una aplicación similar que conste de programas separados con llamadas externas entre las aplicaciones.

Además de enlazar módulos, puede enlazar éstos a programas de servicio (del tipo *SRVPGM). Los programas de servicio le permiten codificar y mantener módulos aparte de los módulos de programa. Se pueden crear rutinas comunes como programas de servicio y, si la rutina se modifica, el cambio se puede incorporar enlazando de nuevo el programa de servicio. No es preciso crear de nuevo los programas que utilizan estas rutinas comunes. Si desea obtener información sobre la creación de programas de servicio, consulte el “Capítulo 8. Creación de un programa de servicio” en la página 93.

Para obtener información acerca del proceso de enlace y del enlazador, consulte la publicación *ILE Concepts*.

Enlace de módulos a un programa

Utilización del mandato CRTPGM

El mandato Crear Programa (CRTPGM) crea un objeto de programa a partir de uno o varios módulos ya creados, y, si es necesario, de uno o varios programas de servicio. Puede enlazar módulos creados con cualquiera de los mandatos ILE de creación de módulos, que son CRTRPGMOD, CRTCMOD, CRTCLMOD y CRTCLMOD.

Nota: Los módulos y/o los programas de servicio necesarios deben haberse creado previamente utilizando el mandato CRTPGM.

Antes de crear un objeto de programa mediante el mandato CRTPGM, debe:

1. Establecer un nombre de programa.
2. Identificar los módulos y, si es necesario, los programas de servicio que desea enlazar en un objeto de programa.
3. Identificar el módulo de entrada.

Se indica el módulo que contiene el procedimiento de entrada de programa mediante el parámetro ENTMOD de CRTPGM. El valor por omisión es ENTMOD(*FIRST), lo cual significa que el módulo que contiene el primer procedimiento de entrada de programa encontrado en la lista para el parámetro MODULE es el módulo de entrada.

Suponiendo que sólo tiene un módulo con un procedimiento principal, es decir, que todos los módulos excepto uno tienen especificado NOMAIN, puede aceptar el valor por omisión (*FIRST). Alternativamente, puede especificar (*ONLY); esto le proporcionará una comprobación de que, en realidad, sólo tiene un procedimiento principal. Por ejemplo, en las dos situaciones siguientes puede especificar ENTMOD(*ONLY).

- Enlaza un módulo RPG a un módulo C sin una función principal().
- Enlaza dos módulos RPG, donde uno tiene NOMAIN en la especificación de control.

Nota: Si está enlazando más de un módulo ILE RPG con un procedimiento principal, debe especificar el nombre del módulo que desea que reciba el control cuando se llama al programa. También puede especificar *FIRST, si el módulo con un procedimiento principal precede a cualquier otro módulo con procedimientos principales de la lista especificada en el parámetro MODULE.

4. Identificar el grupo de activación que el programa utilizará.

Especifica el grupo de activación QILE mencionado si el programa no tiene requisitos especiales o si no está seguro de qué grupo se va a utilizar. En general, es recomendable ejecutar una aplicación en su propio grupo de activación. Sin embargo, tal vez desee dar un nombre al grupo de activación después de la aplicación.

Observe que el grupo de activación por omisión para CRTPGM es *NEW. Esto significa que el programa se ejecutará en su propio grupo de activación, y que éste terminará cuando lo haga el programa. Si activa o no LR, su programa tendrá una copia nueva de datos cada vez que lo llame. Para obtener más información sobre los grupos de activación, consulte el apartado “Cómo especificar un grupo de activación” en la página 112.

Para crear un objeto de programa utilizando el mandato CRTPGM, realice los pasos siguientes:

1. Entre el mandato CRTPGM.

Enlace de módulos a un programa

- Entre los valores correctos para el parámetro del mandato.

La Tabla 7 ofrece una lista de los parámetros del mandato CRTPGM y los valores por omisión. Si desea una descripción completa del mandato CRTPGM y sus parámetros, consulte el apartado *CL y API* de la categoría *Programación del iSeries 400 Information Center*, en el siguiente sitio web - <http://www.ibm.com/eserver/iseriess/infocenter>.

Tabla 7. Parámetros del mandato CRTPGM y sus valores por omisión

Grupo de parámetros	Parámetro (valor por omisión)
Identificación	PGM(<i>nombre biblioteca/nombre programa</i>) MODULE(*PGM)
Acceso de programa	ENTMOD(*FIRST)
Enlace	BNDSRVPGM(*NONE) BNDDIR(*NONE)
Tiempo de 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 haya entrado el mandato CRTPGM, el sistema realiza las acciones siguientes:

- Copia los módulos listados en lo que se convertirá el objeto programa y enlaza los programas de servicio al objeto de programa.
- Identifica el módulo que contiene el procedimiento de entrada del programa y localiza la primera importación de este módulo.
- Comprueba los módulos por el orden en el que aparecen listados y hace coincidir la primera importación con un módulo de exportación.
- Vuelve al primer módulo y localiza la importación siguiente.
- Resuelve todas las importaciones del primer módulo.
- Continúa en el módulo siguiente y resuelve todas las importaciones.
- Resuelve todas las importaciones de los módulos siguientes hasta que se hayan resuelto todas.
- Si alguna importación no se puede resolver con una exportación, el proceso de enlace termina sin crear un objeto de programa.
- Una vez que se han resuelto todas las importaciones, el proceso de enlace finaliza y se crea el objeto de programa.

Nota: Si ha especificado que una variable o procedimiento se va a exportar (mediante la palabra clave EXPORT), es posible que el nombre de la variable o el nombre del procedimiento sea idéntico al de otra variable o procedimiento de otro procedimiento del objeto de programa enlazado. En este caso, el resultado puede no ser el esperado. Consulte la publicación *ILE Concepts* para obtener información sobre cómo manejar esta situación.

Enlace de módulos a un programa

Enlace de varios módulos

En este ejemplo se muestra cómo utilizar el mandato CRTPGM para enlazar dos módulos ILE RPG en un programa TRPT. En este programa, se realiza lo siguiente:

- El módulo TRANSRPT lee cada registro de transacción del archivo TRNSDTA.
- A continuación llama al procedimiento Trans_Inc y Proc_Name del módulo TRANSSVC utilizando las llamadas enlazadas de expresiones.
- Trans_Inc calcula las entradas pertenecientes a cada transacción y devuelve el valor al llamador.
- Proc_Name determina el nombre del producto y lo devuelve
- A continuación, TRANSRPT imprime el registro de transacción.

El fuente de TRANSRPT, TRANSSVC y TRNSDTA se muestra en la Figura 38 en la página 83, Figura 36 en la página 81 y Figura 39 en la página 83, respectivamente.

1. En primer lugar cree el módulo TRANSRPT. Escriba:

```
CRTRPGMOD MODULE(MYLIB/TRANSRPT)
```

2. Después cree el módulo TRANSSVC escribiendo:

```
CRTRPGMOD MODULE(MYLIB/TRANSSVC)
```

3. Para crear el objeto de programa, escriba:

```
CRTPGM PGM(MIBIB/TRPT) MODULE(TRANSRPT TRANSSVC)  
ENTMOD(*FIRST) ACTGRP(TRPT)
```

El mandato CRTPGM crea el objeto de programa TRPT en la biblioteca MIBIB.

Tenga en cuenta que TRANSRPT se lista en primer lugar en el parámetro MODULE. ENTMOD(*FIRST) buscará el primer módulo con un procedimiento de entrada de programa. Dado que sólo uno de los dos módulos tiene un procedimiento de entrada de programa, pueden entrarse en el orden deseado.

El programa TRPT se ejecutará en el grupo de activación TRPT con nombre. El programa se ejecutará en un grupo con nombre para asegurarse de que ningún otro programa afecta sus recursos.

La Figura 41 muestra un archivo de salida creado cuando se ejecuta TRPT.

Nombre producto	Cantidad	Ganancia
-----	-----	-----
Grande	245	330.750,00
Super	15	52.500,00
Supergrande	0	,00
Superextra	123	2.952.000,00
Increíblemente superextra	15	912.000,00
Desconocido	12	,00
Total: 4.247.250,00		

Figura 41. Archivo QSYSPRT para TRPT

Ejemplos adicionales

Para ver más ejemplos de la creación de programas, consulte los apartados siguientes:

- “Enlace a un programa” en la página 100 para ver un ejemplo de enlace de un módulo y un programa de servicio.

- “Ejemplo de fuente para ejemplos de depuración” en la página 245 para ver un ejemplo de creación de un programa que consta de un módulo RPG y un módulo C.

Mandatos CL relacionados

Los mandatos CL siguientes se pueden utilizar con programas:

- Cambiar Programa (CHGPGM)
- Suprimir Programa (DLTPGM)
- Visualizar Programa (DSPPGM)
- Visualizar Referencias de Programa (DSPPGMREF)
- Actualizar Programa (UPDPGM)
- Trabajar con Programas (WRKPGM)

Si desea más información sobre estos mandatos, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>.

Utilización de un listado del enlazador

El proceso de enlace puede generar un listado que describe los recursos utilizados, los símbolos y objetos encontrados y los problemas que se han resuelto o no en el proceso de enlace. Este listado se genera como un archivo en spool del trabajo que se utiliza para entrar el mandato CRTPGM. El valor por omisión del mandato es no generar esta información, pero puede elegir el valor del parámetro *DETAIL* para generarlo con tres niveles de detalle:

- *BASIC
- *EXTENDED
- *FULL

El listado del enlazador contiene las secciones siguientes, en función del valor especificado para *DETAIL*:

Tabla 8. Secciones del listado del enlazador en función del parámetro DETAIL

Nombre de la sección	*BASIC	*EXTENDED	*FULL
Resumen de opciones del mandato	X	X	X
Tabla de resumen abreviada	X	X	X
Tabla de resumen ampliada		X	X
Listado de información del enlazador		X	X
Listado de referencias cruzadas			X
Estadísticas de enlace			X

La información que figura en este listado puede ayudarle a diagnosticar problemas si el enlace no ha sido satisfactorio o proporcionarle información acerca de lo que el enlazador ha encontrado en el proceso. Quizá desee almacenar el listado de un programa ILE en el archivo en el que almacena los módulos o el fuente de módulo de un programa. Para copiar este listado en un archivo de base de datos, puede utilizar el mandato Copiar Archivo en Spool (CPYSPLF).

Utilización de un listado del enlazador

Nota: El mandato CRTBNDRPG no creará un listado del enlazador. Sin embargo, si se producen errores de enlace durante la fase de enlace, estos errores no se anotarán en las anotaciones de trabajo, y el listado del compilador incluirá un mensaje para informarle de este hecho.

Para ver un ejemplo de un listado de enlazador básico, consulte el apartado “Ejemplo de listado del enlazador” en la página 102.

Para obtener más información sobre los listados de enlazador, consulte *ILE Concepts*.

Modificación de un módulo o programa

Puede ser necesario modificar un objeto ILE para mejorarlo o por motivos de mantenimiento. Puede identificar lo que es preciso cambiar utilizando la información de depuración o el listado del enlazador del mandato CRTPGM. A partir de esta información puede determinar qué módulo debe modificarse y, a menudo, qué procedimiento o campo se ha de cambiar.

Además, tal vez desee cambiar el nivel de optimización o de información observable de un módulo o de un programa. Esto suele ocurrir cuando desea depurar un programa o un módulo, o cuando está listo para poner un programa en producción. Estas modificaciones se pueden realizar más rápidamente y utilizan menos recursos del sistema que si se vuelve a crear el objeto en cuestión.

Para finalizar, es posible que desee reducir el tamaño de su programa una vez haya finalizado una aplicación. Los programas ILE tienen datos adicionales añadidos, los cuales los convierten en mayores que un programa OPM similar.

Cada una de estas acciones necesitan datos distintos para efectuar las modificaciones. Los recursos que necesita pueden no estar disponibles en un programa ILE.

Los apartados siguientes le indican cómo

- Actualizar un programa
- Cambiar el nivel de optimización
- Cambiar la información observable
- Reducir el tamaño del objeto

Nota: En el resto de este apartado, el término “objeto” se utilizará para hacer referencia a un módulo ILE o a un programa ILE.

Utilización del mandato UPDPGM

Por lo general, puede actualizar un programa sustituyendo los módulos que sea necesario. Por ejemplo, si añade un nuevo procedimiento a un módulo, recompila el objeto de módulo y a continuación, actualice el programa. No es necesario que vuelva a crear el programa. Esto resulta de utilidad si suministra aplicaciones a otros lugares. Sólo tendrá que enviar los módulos revisados, y el lugar receptor podrá actualizar la aplicación utilizando el mandato UPDPGM o UPDSRVPGM.

El mandato UPDPGM funciona con objetos de programa y módulo. Los parámetros de este mandato son muy parecidos a los del mandato CRTPGM. Por ejemplo, para sustituir un módulo en un programa, debe entrar el nombre del módulo en el parámetro MODULE y el nombre de la biblioteca. El mandato

Modificación de un módulo o programa

UPDPGM requiere que los módulos que se sustituyen estén en las mismas bibliotecas que cuando se creó el programa. Puede especificar la sustitución de todos los módulos o de un grupo de ellos.

El mandato UPDPGM requiere que el objeto de módulo esté presente. De este modo, es más fácil utilizar el mandato cuando ha creado el programa utilizando pasos de compilación y enlace. Puesto que el objeto de módulo ya existe, sólo tiene que especificar el nombre y la biblioteca al emitir el mandato.

Para actualizar un programa creado con el mandato CRTBNDRPG, debe asegurarse de que el módulo revisado está en la biblioteca QTEMP. Ello se debe a que el módulo temporal utilizado cuando se emitió el mandato CRTBNDRPG se creó en QTEMP. Cuando el módulo esté en QTEMP, puede emitir el mandato UPDPGM para sustituir el módulo.

Para obtener más información, consulte *ILE Concepts*.

Modificación del nivel de optimización

Optimizar un objeto es consultar el código compilado, determinar qué se puede hacer para que el programa se ejecute lo más rápido posible y hacer las modificaciones oportunas. Normalmente, cuando más alta es la petición de optimización, más se tarda en crear un objeto. En tiempo de ejecución, un programa con un nivel alto de optimización debe ejecutarse más rápidamente que el correspondiente programa o programa de servicio no optimizado.

Sin embargo, en los niveles más altos de optimización, los valores de los campos pueden no ser exactos cuando se visualizan en una sesión de depuración o después de una recuperación de excepción. Además, es posible que el depurador del fuente haya alterado las posiciones de los pasos y los puntos de interrupción en el código optimizado, ya que las modificaciones de optimización pueden eliminar o cambiar el orden de algunas sentencias.

Para garantizar que el contenido de un campo refleja su valor más reciente, especialmente tras una recuperación de excepción, puede utilizar la palabra clave NOOPT en la especificación de definición correspondiente. Para obtener más información, consulte “Consideraciones sobre optimización” en la página 260.

Para evitar este problema al depurar, puede reducir el nivel de optimización de un módulo para visualizar los campos de un modo preciso a medida que depura el programa, y después aumentar de nuevo el nivel para mejorar la eficacia del programa cuando esté listo para su producción.

Para saber cuál es el nivel de optimización actual de un objeto *programa*, utilice el mandato DSPPGM. La pantalla 3 de este mandato contiene el nivel actual. Para cambiar el nivel de optimización de un programa, utilice el mandato CHGPGM. En el parámetro del programa de optimización podrá especificar uno de los valores siguientes: *FULL, *BASIC, *NONE. Son los mismos valores que se pueden especificar en los parámetros OPTIMIZE de cualquier mandato de creación. El programa se vuelve a crear automáticamente cuando se ejecuta el mandato.

De forma similar, para determinar el nivel de optimización actual de un *módulo*, utilice el mandato DSPMOD. La página 2 de la pantalla 1 de este mandato contiene el nivel actual. Para cambiar el nivel de optimización, utilice el mandato CHGMOD. Después tendrá que crear de nuevo el programa utilizando el mandato UPDPGM o CRTPGM.

Modificación de un módulo o programa

Eliminación de la información observable

La información observable implica los tipos de datos que se pueden almacenar con un objeto y que permiten modificar éste sin volver a compilar el fuente. La adición de estos datos aumenta el tamaño del objeto. Por consiguiente, es posible que desee eliminar estos datos para reducir el tamaño del objeto. Sin embargo, una vez eliminados los datos, la información observable también se elimina. Deberá volver a compilar el fuente y crear de nuevo el programa para sustituir los datos. Estos tipos de datos son:

Datos de creación

Están representados por el valor *CRTDTA. Estos datos son necesarios para traducir el código en instrucciones de máquina. El objeto debe tener antes estos datos para poder cambiar el nivel de optimización.

Datos de depuración

Están representados por el valor *DBGDTA. Estos datos son necesarios para poder depurar un objeto.

Datos de perfil

Están representados por los valores *BLKORD y *PRCORD. Estos datos son necesarios para que el sistema pueda volver a aplicar los datos de perfil de orden de bloques y de orden de procedimientos.

Utilice el mandato CHGPGM o CHGMOD para eliminar alguno o todos los tipos de datos de un programa o módulo. La eliminación de la información observable reduce un objeto a su tamaño mínimo (sin compresión). No es posible modificar el objeto de ninguna manera excepto creándolo de nuevo. Por lo tanto, asegúrese que dispone de todo el fuente necesario para crear el programa o de que tiene un objeto de programa parecido con CRTDATA. Para volverlo a crear, debe tener autorización para acceder al código fuente.

Reducción del tamaño de un objeto

Los datos de creación (*CRTDTA) asociados con un módulo o programa ILE pueden ocupar más de la mitad del tamaño del objeto. Mediante la eliminación o la compresión de estos datos, reducirá los requisitos de almacenamiento secundario para sus programas de forma significativa.

Si elimina estos datos, asegúrese de que dispone de todo el fuente necesario para crear el programa o de que tiene un objeto de programa parecido con CRTDATA. De lo contrario, no podrá modificar el objeto.

Una alternativa es comprimir el objeto, utilizando para ello el mandato Comprimir Objeto (CPROBJ). Un objeto comprimido ocupa menos almacenamiento del sistema que un objeto que no lo esté. Si se llama a un programa comprimido, la parte del objeto que contiene el código ejecutable se descomprime automáticamente. También puede descomprimir un objeto comprimido utilizando el mandato Descomprimir objeto (DCPOBJ).

Si desea más información sobre estos mandatos CL, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>.

Capítulo 8. Creación de un programa de servicio

Este capítulo proporciona:

- Una visión general del concepto de programa de servicio
- Métodos para crear programas de servicio
- Una breve descripción del mandato CRTSRVPGM
- Un ejemplo de un programa de servicio

Visión general de los programas de servicio

Un programa de servicio es un programa enlazado (del tipo *SRVPGM) que consta de un conjunto de procedimientos a los que pueden llamar los procedimientos de otros programas enlazados.

Los programas de servicio suelen utilizarse para las funciones habituales que se invocan con frecuencia en las aplicaciones y entre diferentes aplicaciones. Por ejemplo, los compiladores ILE utilizan programas de servicio que durante la ejecución proporcionan servicios tales como funciones matemáticas y rutinas de entrada/salida. Los programas de servicio permiten volver a utilizar, simplificar el mantenimiento y reducir los requisitos de almacenamiento.

Un programa de servicio se diferencia de los demás programas en lo siguiente:

- No contiene ningún procedimiento de entrada de programa . Esto significa que no se puede llamar a un programa de servicio con la operación CALL.
- Un programa de servicio está enlazado a un programa o a otros programas de servicio mediante el enlace por referencia.

Cuando enlaza un programa de servicio a un programa normal, el contenido del programa de servicio no se copia en el programa enlazado. En su lugar, la información sobre el enlace del programa de servicio es la que se enlaza al programa. Esta operación recibe el nombre de "enlace por referencia" para distinguirla del proceso de enlace estático que se utiliza para enlazar módulos a programas.

Puesto que un programa de servicio se enlaza por referencia a un programa, se puede llamar a los procedimientos exportados del programa de servicio mediante las llamadas a los procedimientos enlazados. La llamada inicial aumenta la actividad general porque el enlace no se completa hasta que se llama al programa de servicio. Sin embargo, las llamadas siguientes a cualquiera de sus procedimientos son más rápidas que las llamadas a programas.

El conjunto de exportaciones que contiene un programa de servicio es la interfaz con los servicios que éste proporciona. Puede utilizar el mandato Visualizar programa de servicio (DSPSRVPGM) o el listado del programa de servicio para ver qué nombres de variables y de procedimientos están disponibles para utilizarlos en los procedimientos que realizan la llamada. Para ver las exportaciones asociadas con el programa de servicio PAYROLL, debe entrar:

```
DSPSRVPGM PAYROLL DETAIL(*PROCEXP *DATAEXP)
```


Métodos para crear programas de servicio

Al crear un programa de servicio, debe tener en cuenta lo siguiente:

1. Si tiene la intención de actualizar el programa en el futuro
2. Si las actualizaciones conllevarán cambios en la interfaz (es decir, en las importaciones y exportaciones utilizadas).

Si cambia la interfaz con un programa de servicio, tal vez tenga que volver a enlazar *cualquiera* de los programas enlazados al programa de servicio original. Sin embargo, si las modificaciones necesarias son compatibles con las versiones posteriores, es posible que no tenga que repetir los enlaces si ha creado el programa de servicio utilizando el lenguaje enlazador. En este caso, después de actualizar el fuente del lenguaje enlazador de modo que identifique las nuevas exportaciones, solamente tendrá que volver a enlazar los programas que las utilizan.

SUGERENCIA

Si piensa crear un módulo únicamente con subprocedimientos (es decir, con un módulo con la palabra clave NOMAIN especificada en la especificación de control) puede que desee crearlo como un programa de servicio. En un sistema, sólo es necesaria una copia de un programa de servicio y, por lo tanto, necesitará menos almacenamiento para el módulo.

Además, utilizando la palabra clave COPYRIGHT puede reservar el derecho de autor de sus programas de servicio en la especificación de control

El lenguaje enlazador le permite controlar las exportaciones de un programa de servicio. Este control puede ser muy útil si desea:

- Ocultar determinados procedimientos de programas de servicio a los usuarios de los programas de servicio
- Arreglar problemas
- Mejorar las funciones
- Reducir el efecto que las modificaciones tienen sobre los usuarios de una aplicación.

Consulte el apartado “Programa de servicio de ejemplo” en la página 96 para obtener un ejemplo de cómo se puede utilizar el lenguaje enlazador para crear un programa de servicio.

Para obtener información acerca del lenguaje enlazador, enmascarar exportaciones y otros conceptos de los programas de servicio, consulte la publicación *ILE Concepts*.

Creación de un programa de servicio utilizando CRTSRVPGM

Puede crear un programa de servicio mediante el mandato Crear Programa de Servicio (CRTSRVPGM). Cualquier módulo ILE se puede enlazar a un programa de servicio. El módulo (o los módulos) debe existir para poder crear un programa de servicio con él.

En la Tabla 9 en la página 95 figuran los parámetros del mandato CRTSRVPGM y sus valores por omisión. Si desea una descripción completa del mandato

Creación de un programa de servicio utilizando CRTSRVPGM

CRTSRVPGM y sus parámetros, consulte el apartado *CL y API* de la categoría *Programación del iSeries 400 Information Center*, en el siguiente sitio web - <http://www.ibm.com/eserver/iseries/infocenter>.

Tabla 9. Parámetros del mandato CRTSRVPGM y sus valores por omisión

Grupo de parámetros	Parámetro (valor por omisión)
Identificación	SRVPGM(<i>nombre biblioteca/nombre programa de servicio</i>) MODULE(*SRVPGM)
Acceso de programa	EXPORT(*SRCFILE) SRCFILE(*LIBL/QSRVSR) SRCMBR(*SRVPGM)
Enlace	BNDSRVPGM(*NONE) BNDDIR(*NONE)
Tiempo de ejecución	ACTGRP(*CALLER)
Varios	OPTION(*GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF) DETAIL(*NONE) ALWUPD(*YES) ALWRINZ(*NO) REPLACE(*YES) AUT(*LIBCRTAUT) TEXT(*ENTMODTXT) TGTRLS(*CURRENT) USRPRF(*USER)

Consulte el apartado “Creación del programa de servicio” en la página 99 para ver un ejemplo de utilización del mandato CRTSRVPGM.

Modificar un programa de servicio

Se puede actualizar o modificar un programa de servicio igual que un objeto de programa. En otras palabras, se puede:

- Actualizar el programa de servicio (utilizando UPDSRVPGM)
- Cambiar el nivel de optimización (utilizando CHGSRVPGM)
- Eliminar la información observable (utilizando CHGSRVPGM)
- Reducir el tamaño (utilizando CPROBJ)

Para obtener más información acerca de estos puntos, consulte el apartado “Modificación de un módulo o programa” en la página 90.

Mandatos CL relacionados

Los mandatos CL siguientes también se utilizan con los programas de servicio:

- Cambiar Programa de Servicio (CHGSRVPGM)
- Visualizar Programa de Servicio (DSPSRVPGM)
- Suprimir Programa de Servicio (DLTSRVPGM)
- Actualizar Programa de Servicio (UPDSRVPGM)
- Trabajar con Programa de Servicio (WRKSRVPGM)

Programa de servicio de ejemplo

El ejemplo siguiente muestra cómo crear el programa de servicio CVTTOHEX, que convierte las series de caracteres en su equivalente hexadecimal. Se transfieren dos parámetros al programa de servicio:

1. un campo de caracteres (InString) que se ha de convertir
2. un campo de caracteres (HexString) que contendrá el equivalente hexadecimal de dos bytes

El campo HexString se utiliza para contener el resultado de la conversión, así como para indicar la longitud de la serie que se ha de convertir. Por ejemplo, si se transfiere una serie de 30 caracteres, pero sólo quiere convertir los 10 primeros, pasaría un segundo parámetro de 20 bytes (10 multiplicado por 2). En función de la longitud de los campos pasados, el programa de servicio determina la longitud que se ha de manejar.

En la Figura 42 en la página 97 se muestra el fuente de un programa de servicio. La Figura 43 en la página 99 muestra el miembro /COPY que contiene el prototipo para CvtToHex.

La lógica básica del procedimiento contenido en el programa de servicio es la siguiente:

1. Se utilizan descriptores operativos para determinar la longitud de los parámetros que se transfieren.
2. Se determina la longitud que se ha de convertir: es la longitud más pequeña de la serie de caracteres o la mitad de la longitud del campo de la serie hexadecimal.
3. Cada carácter de la serie se convierte a su equivalente hexadecimal de dos bytes mediante la subrutina GetHex.

Tenga en cuenta que para mejorar el rendimiento de tiempo de ejecución, GetHex se codifica como una subrutina y no como un subprocedimiento. Una operación EXSR se ejecuta con más rapidez que una llamada enlazada y, en este ejemplo, se llama a GetHex muchas veces.

4. El procedimiento vuelve al llamador.

El programa de servicio utiliza descriptores operativos, que son una construcción ILE que se utiliza cuando no se conoce por anticipado la naturaleza exacta de un parámetro que se ha pasado, en este caso la longitud. Se crean los descriptores operativos en una llamada a un subprocedimiento cuando especifica el expansor de operación (D) en la operación CALLB o cuando se especifica OPDESC en el prototipo.

Para utilizar los descriptores operativos, el programa de servicio debe llamar a la API enlazable ILE, CEEDOD (Recuperar Descriptor Operativo). Esta API necesita ciertos parámetros que deben definirse para la operación CALLB. Sin embargo, es el último parámetro el que proporciona la información necesaria, es decir, la longitud. Para obtener más información sobre los descriptores operativos, consulte el apartado “Utilización de descriptores operativos” en la página 142.

```

=====
* CvtToHex - convertir la serie de entrada a hexadecimal
=====
H COPYRIGHT('(C) Copyright MyCompany 1995')
D/COPY RPGGUIDE/QRPGLE,CVTHEXPR
-----
* Parámetros de entrada principal
* 1. Entrada:  serie                carácter(n)
* 2. Salida:  serie hexadecimal    carácter(2 * n)
-----
D CvtToHex      PI                OPDESC
D  InString          16383      CONST OPTIONS(*VARSIZE)
D  HexString         32766      OPTIONS(*VARSIZE)

-----
* Prototipo para CEEDOD (Recuperar descriptor operativo)
-----
D CEEDOD        PR
D ParmNum                10I 0 CONST
D                      10I 0
D                      10I 0
D                      10I 0
D                      10I 0
D                      10I 0
D                      12A  OPTIONS(*OMIT)

* Parámetros pasados a CEEDOD
D DescType        S                10I 0
D DataType        S                10I 0
D DescInfo1       S                10I 0
D DescInfo2       S                10I 0
D InLen           S                10I 0
D HexLen          S                10I 0

-----
* Otros campos utilizados por el programa
-----
D HexDigits       C                CONST('0123456789ABCDEF')
D IntDs           DS
D  IntNum          5I 0 INZ(0)
D  IntChar         1  OVERLAY(IntNum:2)
D HexDs           DS
D  HexC1           1
D  HexC2           1
D InChar          S                1
D Pos             S                5P 0
D HexPos          S                5P 0

```

Figura 42. Fuente del programa de servicio CvtToHex (Pieza 1 de 2)

Programa de servicio de ejemplo

```

*-----*
* Use los descriptores operativos para determinar la longitud de *
* los parámetros que se han pasado. *
*-----*
C          CALLP      CEEDOD(1          : DescType : DataType :
C                               DescInfo1 : DescInfo2: Inlen   :
C                               *OMIT)
C          CALLP      CEEDOD(2          : DescType : DataType :
C                               DescInfo1 : DescInfo2: HexLen   :
C                               *OMIT)
*-----*
* Determine la longitud que se ha de manejar (un mínimo de la *
* longitud de entrada y la mitad de la longitud hexadecimal) *
*-----*
C          IF          InLen > HexLen / 2
C          EVAL        InLen = HexLen / 2
C          ENDIF

*-----*
* Cada carácter de la serie de entrada se debe convertir en *
* representación hexadecimal de 2 bytes (p. ej., '5' --> 'F5') *
*-----*
C          EVAL        HexPos = 1
C          DO          InLen          Pos
C          EVAL        InChar = %SUBST(InString : Pos :1)
C          EXSR        GetHex
C          EVAL        %SUBST(HexString : HexPos : 2) = HexDs
C          EVAL        HexPos = HexPos + 2
C          ENDDO

*-----*
* Terminado; devolver al llamador. *
*-----*
C          RETURN

*=====
* GetHex - subrutina para convertir 'InChar' a 'HexDs' *
* *
* Use la división por 16 para separa los 2 dígitos hexadecimales. *
* El cociente es el primer dígito, y el resto es el segundo. *
*=====
C          GetHex      BEGSR
C          EVAL        IntChar = InChar
C          IntNum      DIV      16          X1          5 0
C          MVR          X2          5 0
*-----*
* Use el dígito hexadecimal (más 1) para realizar la subserie de *
* la lista de caracteres hexadecimales '012...CDEF'. *
*-----*
C          EVAL        HexC1 = %SUBST(HexDigits:X1+1:1)
C          EVAL        HexC2 = %SUBST(HexDigits:X2+1:1)
C          ENDSR

```

Figura 42. Fuente del programa de servicio CvtToHex (Pieza 2 de 2)

*=====			
* CvtToHex - convertir la serie de entrada a hexadecimal			
*			
* Parámetros			
* 1. Entrada:	serie		carácter(n)
* 2. Salida:	serie hexadecimal		carácter(2 * n)
*=====			
D	CvtToHex	PR	OPDESC
D	InString	16383	CONST OPTIONS(*VARSIZE)
D	HexString	32766	OPTIONS(*VARSIZE)

Figura 43. Fuente del miembro /COPY con el prototipo para CvtToHex

Al diseñar este programa de servicio, se decidió utilizar el lenguaje enlazador para determinar la interfaz, de manera que el programa se pudiera actualizar más fácilmente en el futuro. La Figura 44 muestra el lenguaje enlazador necesario para definir las exportaciones del programa de servicio CVTTOHEX. Este fuente se utiliza en los parámetros EXPORT, SRCFILE y SRCMBR del mandato CRTSRVPGM.

```
STRPGMEXP SIGNATURE('CVTHEX')

EXPORT SYMBOL('CVTTOHEX')
ENDPGMEXP
```

Figura 44. Fuente del lenguaje enlazador de CvtToHex

El parámetro SIGNATURE de STRPGMEXP identifica la interfaz que proporcionará el programa de servicio. En este caso, la exportación identificada en el lenguaje enlazador es la interfaz. Cualquier programa que esté enlazado a CVTTOHEX utilizará esta signatura.

Las sentencias EXPORT del lenguaje enlazador identifican las exportaciones del programa de servicio. Se necesita una para cada procedimiento cuyas exportaciones desee poner a disposición del llamador. En este caso, el programa de servicio contiene un módulo que contiene un procedimiento. Por lo tanto, sólo se necesita una sentencia EXPORT.

Si desea obtener más información acerca del lenguaje enlazador y las signaturas, consulte la publicación *ILE Concepts*.

Creación del programa de servicio

Para crear el programa de servicio CVTTOHEX, siga estos pasos:

1. Cree el módulo CVTTOHEX a partir del fuente de la Figura 42 en la página 97 escribiendo lo siguiente:

```
CRTRPGMOD MODULE(MYLIB/CVTTOHEX) SRCFILE(MYLIB/QRPGLESRC)
```

2. Cree el programa de servicio utilizando el módulo CVTTOHEX y el lenguaje enlazador que se muestra en la Figura 44.

```
CRTSRVPGM SRVPGM(MYLIB/CVTTOHEX) MODULE(*SRVPGM)
EXPORT(*SRCFILE) SRCFILE(MIBIB/QRVSRVC)
SRCMBR(*SRVPGM)
```

Los tres últimos parámetros del mandato anterior identifican las exportaciones que el programa de servicio hará que estén disponibles. En este caso, se basa en el fuente que se encuentra en el miembro CVTTOHEX del archivo QRVSRVC de la biblioteca MIBIB.

Programa de servicio de ejemplo

Observe que no se necesita un directorio de enlace en este caso, ya que todos los módulos necesarios para crear el programa de servicio se han especificado con el parámetro MODULE.

El programa de servicio CVTTOHEX se creará en la biblioteca MIBIB. Se puede depurar utilizando una vista de sentencias; esto se determina mediante el parámetro DBGVIEW por omisión en el mandato CRTRPGMOD. No se genera ningún listado del enlazador.

Enlace a un programa

Para completar el ejemplo, crearemos una "aplicación" que constará del programa CVTHEXPGM que está enlazado al programa de servicio. Utiliza una serie de siete caracteres que se pasa dos veces a CVTTOHEX; una vez cuando el valor de la serie hexadecimal es 10 (es decir, convertir 5 caracteres) y la otra cuando su valor es 14, es decir, la longitud real.

Observe que el programa CVTHEXPGM sirve para mostrar la utilización del programa de servicio CVTTOHEX. En una aplicación real, el objetivo primordial de llamar a CVTTOHEX no será comprobar CVTTOHEX. Además, normalmente un programa de servicio será utilizado por muchos programas diferentes, o unos pocos programas lo utilizarán muchas veces; de lo contrario, la actividad que supone la llamada inicial no justificará su conversión en un programa de servicio.

Para crear la aplicación, siga estos pasos:

1. Cree el módulo a partir del fuente de la Figura 45 en la página 101 escribiendo lo siguiente:

```
CRTRPGMOD MODULE(MILIB/CVTHEXPGM) SRCFILE(MILIB/QRPGLESRC)
```

2. Cree el programa escribiendo

```
CRTPGM PGM(MILIB/CVTHEXPGM)
      BNDSRVPGM(MIBIB/CVTTOHEX)
      DETAIL(*BASIC)
```

Una vez creado CVTHEXPGM, éste incluirá la información referente a la interfaz que utiliza para interactuar con el programa de servicio. Es lo mismo que lo que refleja el lenguaje enlazador correspondiente a CVTTOHEX.

3. Llame al programa escribiendo:

```
CALL CVTHEXPGM
```

Durante el proceso de preparar el programa CVTHEXPGM para su ejecución, el sistema verifica que:

- El programa de servicio CVTTOHEX se encuentra en la biblioteca MIBIB
- La interfaz de uso público que utilizaba CVTHEXPGM cuando se creó sigue siendo válida en la ejecución.

Si alguna de estas condiciones no se cumple, se emite un mensaje de error.

A continuación se muestra la salida del programa CVTHEXPGM. (La serie de entrada es 'ABC123*'.)

```
Result14++++++
Result10++
C1C2C3F1F2      10 caracteres de salida
C1C2C3F1F2F35C  14 caracteres de salida
```

```

*-----*
* Programa para comprobar el programa de servicio CVTTOHEX *
* *
* 1. Use una serie de entrada de 7 caracteres *
* 2. Conviértala en una serie hexadecimal de 10 caracteres (sólo *
* se usarán los primeros 5 caracteres de entrada ya que el *
* resultado es demasiado pequeño para toda la serie de entrada*
* 3. Conviértala en una serie hexadecimal de 14 caracteres (se *
* usarán los 7 caracteres de entrada ya que el resultado es lo*
* suficientemente grande). *
*-----*
FQSYSPRT 0 F 80 PRINTER
* Prototipo para CvtToHex
D/COPY RPGGUIDE/QRPGLE,CVTHEXPR
D ResultDS DS
D Result14 1 14
D Result10 1 10
D InString S 7
D Comment S 25
C EVAL InString = 'ABC123*'

*-----*
* Pase la serie de caracteres y el campo de resultado de 10 *
* caracteres utilizando una llamada de prototipo. Los *
* descriptores operativos se pasan, como requiere el *
* procedimiento llamado CvtToHex. *
*-----*
C EVAL Comment = 'salida de 10 caracteres'
C CLEAR ResultDS
C CALLP CvtToHex(InString : Result10)
C EXCEPT

*-----*
* Pase la serie de caracteres y el campo de resultado de 14 *
* caracteres utilizando un CALLB(D). El extensor de operaciones *
* (D) creará descriptores operativos para los parámetros pasados.*
* CALLB se utiliza aquí para compararlo con el CALLP anterior. *
*-----*
C EVAL Comment = 'salida de 14 caracteres'
C CLEAR ResultDS
C CALLB(D) 'CVTTOHEX'
C PARM InString
C PARM Result14
C EXCEPT
C EVAL *INLR = *ON

OQSYSPRT H 1P
0 'Result14+++++'
OQSYSPRT H 1P
0 'Result10++'
OQSYSPRT E
0 ResultDS
0 Comment +5

```

Figura 45. Fuente del programa de prueba CVTHEXPGM

Actualización del programa de servicio

Gracias al lenguaje enlazador, el programa de servicio se ha podido actualizar y el programa CVTHEXPGM no se ha tenido que compilar de nuevo. Por ejemplo, existen dos modos de añadir un nuevo procedimiento a CVTTOHEX, dependiendo de si el nuevo procedimiento pasa al módulo existente o a uno nuevo.

Programa de servicio de ejemplo

Para añadir un nuevo procedimiento a un módulo existente, debe:

1. Añadir un nuevo procedimiento al módulo existente.
2. Recompilar el módulo cambiado.
3. Modificar el fuente del lenguaje enlazador para manejar la interfaz asociada con el nuevo procedimiento. Esto implica la adición de nuevas sentencias de exportación *a continuación* de las ya existentes.
4. Volver a crear el programa de servicio utilizando CRTSRVPGM.

Para añadir un nuevo procedimiento utilizando un módulo nuevo, debe:

1. Crear un objeto de módulo para el procedimiento nuevo.
2. Modificar el fuente del lenguaje enlazador para manejar la interfaz asociada con el nuevo procedimiento, como se ha mencionado anteriormente.
3. Enlazar el nuevo módulo al programa de servicio CVTTOHEX volviendo a crearlo.

Con cualquiera de ambos métodos, los nuevos programas pueden acceder a la nueva función. Puesto que las exportaciones antiguas están en el mismo orden, los programas existentes pueden seguir utilizándolas. No será necesario compilar de nuevo hasta que tengan que actualizarse también los programas existentes.

Para obtener más información acerca de la actualización de los programas de servicio, consulte la publicación *ILE Concepts*.

Ejemplo de listado del enlazador

La Figura 46 en la página 103 muestra un ejemplo de listado del enlazador para CVTHEXPGM. Se trata de un listado básico. Para obtener más información sobre los listados del enlazador, consulte el apartado “Utilización de un listado del enlazador” en la página 89 y la publicación *ILE Concepts*.


```

                    Crear programa
5769SS1 V4R4M0 990521                                MYLIB/CVTHEXPGM  AS400S01  07/30/99  Página 1
23:24:00
Programa . . . . . : CVTHEXPGM
Biblioteca . . . . . : MYLIB
Módulo de procedimiento de entrada de programa . . : *FIRST
Biblioteca . . . . . :
Grupo de activación . . . . . : *NEW
Opciones de creación . . . . . : *GEN *NODUPPROC *NODUPVAR *WARN *RSLVREF
Listado de detalles . . . . . : *BASIC
Permitir actualización . . . . . : *YES
Perfil de usuario . . . . . : *USER
Sustituir programa existente . . . . . : *YES
Autorización . . . . . : *LIBCRTAUT
Release de destino . . . . . : *CURRENT
Permitir reinicialización . . . . . : *NO
Texto . . . . . : *ENTMODTXT
Módulo Biblioteca Módulo Biblioteca Módulo Biblioteca
CVTHEXPGM MYLIB
Programa Programa Programa
de servicio Biblioteca de servicio Biblioteca de servicio Biblioteca
CVTTOHEX MYLIB
Directorio Directorio Directorio
de enlace Biblioteca de enlace Biblioteca de enlace Biblioteca
*NONE
Crear programa
5769SS1 V4R4M0 990521                                MYLIB/CVTHEXPGM  AS400S01  07/30/99  Página 2
23:24:00
                    Tabla de resumen abreviada
Procedimientos de entrada de programa . . . . . : 1
Símbolo Tipo Biblioteca Objeto Identificador
*MODULE MYLIB CVTHEXPGM _QRNP_PEP_CVTHEXPGM
Varias definiciones fuertes . . . . . : 0
Referencias sin resolver . . . . . : 0
***** F I N D E T A B L A D E R E S U M E N A B R E V I A D A *****
Crear programa
5769SS1 V4R4M0 990521                                MYLIB/CVTHEXPGM  AS400S01  07/30/99  Página 3
23:24:00
                    Estadísticas de enlace
Tiempo CPU de recogida de símbolos . . . . . : .016
Tiempo CPU de resolución de símbolos . . . . . : .004
Tiempo CPU de resolución de directorio de enlace . . . . . : .175
Tiempo CPU de compilación de lenguaje enlazador . . . . . : .000
Tiempo CPU de creación de listado . . . . . : .068
Tiempo CPU de creación de programa de servicio/programa . . : .234
Tiempo CPU total . . . . . : .995
Tiempo total transcurrido . . . . . : 3.531
***** F I N D E E S T A D Í S T I C A D E E N L A C E *****
*CPC5D07 - Programa CVTHEXPGM creado en la biblioteca MYLIB.
***** F I N D E L I S T A D O D E C R E A C I Ó N D E P R O G R A M A *****

```

Figura 46. Listado del enlazador básico para CVTHEXPGM

Programa de servicio de ejemplo

Capítulo 9. Ejecución de un programa

En este capítulo se muestra cómo:

- Ejecutar un programa y pasar parámetros utilizando el mandato CL CALL
- Ejecutar un programa desde una aplicación dirigida por un menú
- Ejecutar un programa utilizando un mandato creado por el usuario
- Gestionar grupos de activación
- Gestionar almacenamiento durante la ejecución.

Además, puede ejecutar un programa utilizando lo siguiente:

- El Menú del Programador. La publicación *CL Programming*, SC41-5721-04 contiene información sobre este menú.
- El mandato Arrancar gestor para el desarrollo de programación (STRPDM). La publicación *ADTS/400: Programming Development Manager* contiene información sobre este mandato.
- El programa QCMDEXC. La publicación *CL Programming* contiene información sobre este programa.
- Un lenguaje de alto nivel. El “Capítulo 10. Llamadas a programas y procedimientos” en la página 131 proporciona información acerca de la ejecución de programas en otro HLL o de la llamada de programas y procedimientos de servicio.

Ejecución de un programa utilizando el mandato CL CALL

Se puede utilizar el mandato CL CALL para ejecutar un programa (de tipo *PGM). Este mandato se puede utilizar de forma interactiva, como parte de un trabajo de proceso por lotes o incluirlo en un programa CL. Si necesita información sobre los valores que debe entrar, escriba CALL y pulse F4 (Solicitud). Si necesita ayuda, escriba CALL y pulse F1 (Ayuda).

Por ejemplo, para llamar al programa EMPRPT desde la línea de mandatos, escriba:

```
CALL EMPRPT
```

El objeto de programa especificado debe existir en una biblioteca, y ésta debe figurar en la lista de bibliotecas *LIBL. También puede especificar explícitamente la biblioteca en el mandato CL CALL del modo siguiente:

```
CALL MILIB/EMPRPT
```

Si desea más información sobre la utilización del mandato CL CALL, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>.

Después de llamar al programa, el sistema OS/400 ejecuta las instrucciones de éste.

Pasar parámetros utilizando el mandato CL CALL

Puede utilizar la opción PARM del mandato CL CALL para pasar parámetros al programa ILE cuando lo ejecute.

```
CALL PGM(nombre-programa)  
PARM(parámetro-1 parámetro-2 ... parámetro-n)
```

Ejecución de un programa utilizando el mandato CL CALL

También puede escribir los parámetros sin especificar palabras clave:

CALL biblioteca/nombre-programa (parámetro-1 parámetro-2 ... parámetro-n)

Todo valor de parámetro se puede especificar como una variable de programa CL o como:

- Una constante de serie de caracteres
- Una constante numérica
- Una constante lógica

Si pasa parámetros a un programa en el que un procedimiento ILE RPG es el procedimiento de entrada de programa, dicho programa debe tener especificado *ENTRY PLIST una vez y solamente una vez. Los parámetros siguientes (en las sentencias PARM) deben tener una correspondencia unívoca con los que se pasan el mandato CALL.

Consulte el mandato CALL en la publicación *CL Programming*, en el apartado sobre la operación de pasar parámetros entre programas, para ver una descripción completa de cómo se manejan los parámetros.

Por ejemplo, el programa EMPRPT2 requiere que se pase la contraseña correcta cuando se arranca por primera vez; de lo contrario no se ejecutará. La Figura 47 muestra el fuente.

1. Para crear el programa, escriba:
CRTBNDRPG PGM(MILIB/EMPRPT2)
2. Para ejecutar el programa, escriba:
CALL MILIB/EMPRPT2 (HELLO)

Cuando se emite el mandato CALL, se almacena el contenido del parámetro que ha pasado el mandato y el parámetro PSWORD del programa apunta a su ubicación. A continuación, el programa comprueba si el contenido de PSWORD coincide con el valor almacenado en el programa ('HELLO'). En este caso, los dos valores son iguales, por lo que el programa sigue ejecutándose.

```
*****
* NOMBRE PROG:    EMPRPT2                               *
* ARCHIV RELAC:   EMPMST  (ARCHIVO FÍSICO)                *
*                PRINT   (ARCHIVO DE IMPRESORA)           *
* DESCRIPCIÓN:    Este programa imprime información de empleados*
*                almacenada en el archivo EMPMST si la    *
*                contraseña especificada es la correcta.  *
*                Escriba "CALL library name/EMPRT2 (PSWORD)" en*
*                la línea de mandatos para ejecutar el prog., *
*                donde PSWORD es la contraseña del prog.  *
*                La contraseña de este programa es 'HELLO'. *
*****
FPRINT    0    F    80    PRINTER
FEMPMST   IP    E        K DISK
IEMPREC   01
```

Figura 47. Programa ILE RPG que requiere parámetros durante la ejecución (Pieza 1 de 2)

Ejecución de un programa utilizando el mandato CL CALL

```

*-----*
* La lista de parámetros de entrada está especificada en el prog. *
* Hay un parámetro, denominado PSWORD, que es un campo *
* de caracteres de 5 caracteres de largo. *
*-----*
C      *ENTRY      PLIST
C      PARM              PSWORD      5
*-----*
* La contraseña de este programa es 'HELLO'. El campo PSWORD *
* se comprueba para ver si contiene o no 'HELLO'. *
* Si es que no, el indicador de último registro (LR) y *IN99 *
* se activan. *IN99 controla la impresión de los mensajes. *
*-----*
C      PSWORD      IFNE      'HELLO'
C      SETON
C      ENDIF
C      LR99
C      OPRINT      H      1P      2      6
0      50 'EMPLOYEE INFORMATION'
0      H      1P
0      12 'NAME'
0      34 'SERIAL #'
0      45 'DEPT'
0      56 'TYPE'
0      D      01N99
0      ENAME      20
0      ENUM      32
0      EDEPT      45
0      ETYPE      55
0      D      99
0      16 '***'
0      40 'Contraseña no válida'
0      43 '***'

```

Figura 47. Programa ILE RPG que requiere parámetros durante la ejecución (Pieza 2 de 2)

En la Figura 48 se muestra la DDS a las que hace referencia el fuente EMPRPT2.

```

A*****
A* DESCRIPCIÓN: Esta es la DDS del archivo físico EMPMST. *
A* Contiene un formato de registro llamado EMPREC. *
A* Este archivo contiene un registro para cada *
A* empleado de la empresa. *
A*****
A*
A      R EMPREC
A      ENUM      5      0      TEXT('NÚMERO EMPLEADO')
A      ENAME      20      TEXT('NOMBRE EMPLEADO')
A      ETYPE      1      TEXT('TIPO EMPLEADO')
A      EDEPT      3      0      TEXT('DEPARTAMENTO EMPLEADO')
A      ENHRS      3      1      TEXT('HORAS TRABAJO NORMAL EMPLEADO')
A      K ENUM

```

Figura 48. DDS para EMPRPT2

Ejecución de un programa desde una aplicación dirigida por un menú

Otra forma de ejecutar un programa ILE es hacerlo desde una aplicación dirigida por menú. El usuario de la estación de trabajo selecciona una opción de un menú, la cual llama a un programa determinado. La Figura 49 en la página 108 ilustra un ejemplo de un menú de aplicaciones.

Ejecución de un programa desde una aplicación dirigida por un menú

MENÚ DE DEPARTAMENTO DE NÓMINAS

Seleccione una de las opciones siguientes:

1. Preguntar al maestro de empleados
2. Cambiar el maestro de empleados
3. Añadir nuevo empleado

Selección o mandato

==> _____

F3=Salir F4=Solicitud F9=Recuperar F12=Cancelar
F13=Asistente de información F16=Menú principal AS/400

Figura 49. Ejemplo de un menú de una aplicación

El menú que aparece en la Figura 49 lo muestra un programa de menús en el que cada opción llama a un programa ILE distinto. Puede crear un menú utilizando STRSDA y seleccionando la opción 2 ("Diseñar menús").

En la Figura 50 en la página 109 se muestra la DDS del archivo de pantalla del menú anterior. El miembro fuente se llama PAYROL y su tipo de fuente es MNUDDS. El archivo se ha creado utilizando SDA.

Ejecución de un programa desde una aplicación dirigida por un menú

```

A* Free Form Menu: PAYROL
A*
A          DSPSIZ(24 80 *DS3          -
A          27 132 *DS4)
A          CHGINPDT
A          INDARA
A          PRINT(*LIBL/QSYSPRT)
A          R PAYROL
A          DSPMOD(*DS3)
A          LOCK
A          SLNO(01)
A          CLRL(*ALL)
A          ALWROL
A          CF03
A          HELP
A          HOME
A          HLPRTN
A          1 34'MENÚ DE DEPARTAMENTO DE NÓMINAS
A          DSPATR(HI)
A          3 2'Selec. una de las opc. siguientes:'
A          COLOR(BLU)
A          5 7'1.'
A          6 7'2.'
A          7 7'3.'
A* CMDPROMPT No suprima esta especificación de DDS
A          019 2'Selección o mandato          -
A          '
A          5 11'Preguntar'
A          5 19'en'
A          5 24'empleado'
A          5 33'maestro'
A          6 11'Cambiar'
A          6 18'empleado'
A          6 27'maestro'
A          7 11'Añadir'
A          7 15'nuevo'
A          7 19'empleado'

```

Figura 50. Especificación de descripción de datos (DDS) de un menú de aplicación

La Figura 51 muestra el fuente del menú de aplicación ilustrado en la Figura 49 en la página 108 . El miembro fuente se denomina PAYROLQQ y su tipo de fuente es MNUCMD. También se ha creado utilizando SDA.

```

PAYROLQQ,1
0001 call RPGINQ
0002 call RPGCHG
0003 call RPGADD

```

Figura 51. Fuente del programa de menú

Puede ejecutar el menú escribiendo lo siguiente:

GO nombre biblioteca/PAYROL

Si el usuario entra 1, 2 ó 3 en el menú de la aplicación, el fuente de la Figura 51 llama a los programas RPGINQ, RPGCHG o RPGADD respectivamente.

Ejecución de un programa utilizando un mandato creado por el usuario

Puede crear un mandato para ejecutar un programa utilizando una definición de mandato. Una **definición de mandato** es un objeto (tipo *CMD) que contiene la definición de un mandato (que incluye el nombre del mandato, las descripciones de los parámetros y la información de comprobación de validez) e identifica el programa que realiza la función solicitada por el mandato.

Por ejemplo, puede crear un mandato llamado PAY que llame al programa PAYROL, siendo PAYROL el nombre del programa RPG que desea ejecutar. Puede entrar el mandato de forma interactiva o en un trabajo de proceso por lotes. Consulte la publicación *CL Programming* para obtener más información sobre la utilización de las definiciones de mandatos.

Respuesta a los mensajes de consulta durante la ejecución

Cuando ejecuta un programa con procedimientos ILE RPG, pueden generarse mensajes de consulta en la ejecución. Se producen cuando no hay un indicador de error o subrutina de error (*PSSR o INFSTR) para manejar la excepción en un procedimiento principal. Para que el programa continúe ejecutándose, es necesario responder a los mensajes de consulta.

Nota: Los mensajes de consulta no se emiten nunca para subprocedimientos.

Puede añadir los mensajes de consulta a una lista de respuestas del sistema para responder automáticamente a los mensajes. Las respuestas a estos mensajes se pueden especificar de forma individual o general. Esta forma de responder a los mensajes de consulta resulta especialmente adecuada para los programas de proceso por lotes que, de lo contrario, necesitarían un operador que emitiera las respuestas.

Puede añadir los siguientes mensajes de consulta ILE RPG a la lista de respuestas del sistema

Tabla 10. Mensajes de consulta ILE RPG

RNQ0100	RNQ0231	RNQ0421	RNQ1023	RNQ1235
RNQ0101	RNQ0232	RNQ0425	RNQ1024	RNQ1241
RNQ0102	RNQ0299	RNQ0426	RNQ1031	RNQ1251
RNQ0103	RNQ0301	RNQ0431	RNQ1041	RNQ1255
RNQ0104	RNQ0302	RNQ0432	RNQ1042	RNQ1261
RNQ0112	RNQ0303	RNQ0450	RNQ1051	RNQ1271
RNQ0113	RNQ0304	RNQ0501	RNQ1071	RNQ1281
RNQ0114	RNQ0305	RNQ0502	RNQ1201	RNQ1282
RNQ0115	RNQ0306	RNQ0802	RNQ1211	RNQ1284
RNQ0120	RNQ0333	RNQ0803	RNQ1215	RNQ1285
RNQ0121	RNQ0401	RNQ0804	RNQ1216	RNQ1286
RNQ0122	RNQ0402	RNQ0805	RNQ1217	RNQ1287
RNQ0123	RNQ0411	RNQ0907	RNQ1218	RNQ1299
RNQ0202	RNQ0412	RNQ1011	RNQ1221	RNQ1331
RNQ0211	RNQ0413	RNQ1021	RNQ1222	RNQ9998
RNQ0221	RNQ0414	RNQ1022	RNQ1231	RNQ9999
RNQ0222	RNQ0415			

Nota: Los mensajes de consulta ILE RPG tienen el prefijo del identificador de mensaje RNQ.

Respuesta a los mensajes de consulta durante la ejecución

Para añadir mensajes de consulta a una lista de respuestas del sistema mediante el mandato Añadir Entrada a Lista de Respuestas, entre lo siguiente:

ADDRPYLE número-secuencia id-mensaje

siendo *número-secuencia* un número comprendido entre 1 y 9999 que indica en qué posición de la lista se añade la entrada e *id-mensaje* el número de mensaje que desea añadir. Repita este mandato para cada mensaje que desee añadir.

Utilice el mandato Cambiar Trabajo (CHGJOB) (u otro mandato CL de trabajo) para indicar que un trabajo utiliza la lista de respuestas para los mensajes de consulta. Para ello, debe especificar *SYSRPLY en el atributo Respuesta a Mensaje de Consulta (INQMSGRPY).

La lista de respuestas sólo se utiliza cuando un trabajo que tiene especificado el atributo Respuesta a Mensaje de Consulta (INQMSGRPY) como INQMSGRPY(*SYSRPLY) emite un mensaje de consulta. El parámetro INQMSGRPY aparece en los mandatos CL siguientes:

- Cambiar Trabajo (CHGJOB)
- Cambiar Descripción de Trabajo (CHGJOBDD)
- Crear Descripción de Trabajo (CRTJOBDD)
- Someter Trabajo (SBMJOB).

También puede utilizar el mandato Trabajar con Entrada de Lista de Respuestas (WRKRPYLE) para modificar o eliminar entradas en la lista de respuestas del sistema. Si desea más información sobre los mandatos ADDRPLYE y WRKRPYLE, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>.

Finalización de un programa ILE

Cuando un programa ILE finaliza de forma normal, el sistema devuelve el control al llamador. Éste puede ser el usuario de una estación de trabajo o bien otro programa (como el programa de manejo de menús).

Si un programa ILE finaliza anormalmente y el programa se estaba ejecutando en un grupo de activación distinto al de su llamador, se emite el mensaje de escape CEE9901

El error *id-mensaje* ha hecho que el programa finalice.

y el control se devuelve al llamador.

Un programa CL puede supervisar esta excepción utilizando el mandato MONMSG (Supervisar Mensaje). También se puede supervisar las excepciones en otros lenguajes ILE.

Si el programa ILE se está ejecutando en el mismo grupo de activación que el del llamador y finaliza anormalmente, el mensaje emitido dependerá del motivo de la finalización del programa. Si finaliza con un error de función, se emitirá el mensaje CPF9999. Si es un procedimiento RPG el que emite la excepción, el prefijo del mensaje será RNX.

Para obtener más información sobre los mensajes de excepción, consulte el apartado “Visión general del manejo de excepciones” en la página 252.

Gestión de los grupos de activación

Un **grupo de activación** es una subestructura de un trabajo que consta de recursos del sistema (por ejemplo, almacenamiento, definiciones de compromiso y archivos abiertos) que se asignan para ejecutar uno o varios programas ILE u OPM. Los grupos de activación permiten que los programas ILE que se ejecutan en el mismo trabajo se ejecuten independientemente sin molestar entre ellos (por ejemplo, el control de compromiso y las alteraciones temporales). La idea básica es que todos los programas activados en un grupo de activación se desarrollen como una sola aplicación cooperativa.

El grupo de activación en el que se ejecutará un programa ILE se identifica al crear el programa. El grupo de activación se determina mediante el valor especificado en el parámetro ACTGRP cuando se crea el objeto de programa. (Los programas OPM siempre se ejecutan en el grupo de activación por omisión; no puede cambiar la especificación de grupo de activación de dichos programas). Una vez activado un programa ILE (tipo de objeto *PGM), permanecerá activo hasta que se suprima el grupo de activación.

El resto de esta sección le indica cómo especificar y cómo suprimir un grupo de activación. Para obtener más información sobre los grupos de activación, consulte la publicación *ILE Concepts*.

Cómo especificar un grupo de activación

Puede controlar en qué grupo de activación se ejecutará un programa ILE especificando un valor en el parámetro ACTGRP al crear el programa (mediante los mandatos CRTPGM o CRTBNDRPG) o programa de servicio (mediante el mandato CRTSRVPGM).

Nota: si utiliza el mandato CRTBNDRPG, solamente puede especificar un valor en ACTGRP si el valor de DFTACTGRP es *NO.

Puede elegir uno de los valores siguientes:

- un grupo de activación con nombre

Un grupo de activación con nombre le permite gestionar un grupo de programas de servicio y programas ILE como una sola aplicación. El grupo de activación se crea cuando se llama al primer programa que ha especificado el nombre del grupo de activación durante la creación. Después lo utilizarán todos los programas y programas de servicio que tengan especificado el mismo nombre de grupo de activación.

Un grupo de activación con nombre finaliza cuando se suprime con el mandato CL RCLACTGRP. Este mandato sólo puede utilizarse cuando el grupo de activación ya no se utiliza. Cuando finaliza, *todos* los recursos asociados con los programas y los programas de servicio del grupo de activación con nombre se devuelven al sistema.

El grupo de activación con nombre QILE es el valor por omisión del parámetro ACTGRP en el mandato CRTBNDRPG. Sin embargo, puesto que los grupos de activación están pensados para corresponderse con aplicaciones, es recomendable que especifique otro valor en este parámetro. Por ejemplo, tal vez desee que el grupo de activación tenga el nombre que la aplicación.

- *NEW

Cuando se especifica *NEW, se crea un grupo de activación nuevo cada vez que se llama al programa. El sistema crea un nombre para el grupo de activación. Este nombre es exclusivo en el trabajo.

Un grupo de activación creado con *NEW siempre finaliza cuando termina el programa con el que está asociado. Por este motivo, si desea salir del programa con LR OFF para que siga activo, no deberá especificar *NEW para el parámetro ACTGRP.

Nota: este valor no es válido para los programas de servicio. Un programa de servicio sólo se puede ejecutar en un grupo de activación con nombre o en el grupo de activación del llamador.

*NEW es el valor por omisión del parámetro ACTGRP en el mandato CRTPGM.

Si crea un programa ILE RPG con ACTGRP(*NEW), puede llamar al programa tantas veces como desee sin retornar de las llamadas anteriores. Con cada llamada, se creará una copia nueva del programa. Cada una de estas copias tendrá sus propios datos, abrirá sus archivos, etc. No obstante, debe asegurarse de que haya algún modo de finalizar las llamadas a "sí mismo"; de lo contrario, seguirá creando nuevos grupos de activación y los programas nunca retornarán.

- ***CALLER**

El programa o el programa de servicio se activará en el grupo de activación del programa de llamada. Si un programa OPM llama a un programa ILE creado con ACTGRP(*CALLER), éste se activará en el grupo de activación OPM por omisión (*DFTACTGRP).

Ejecución en el grupo de activación OPM por omisión

Cuando se crea un trabajo OS/400 el sistema crea un grupo de activación que utilizarán los programas OPM. El símbolo que se utiliza para representar este grupo de activación es *DFTACTGRP. No puede suprimir el grupo de activación OPM por omisión. Lo suprime el sistema cuando finaliza el trabajo.

Los programas OPM se ejecutan automáticamente en el grupo de activación OPM por omisión. Un programa ILE también se ejecutará en el grupo de activación OPM por omisión si se cumple una de estas condiciones:

- El programa se ha creado con DFTACTGRP(*YES) en el mandato CRTBNDRPG.
- El programa se ha creado con ACTGRP(*CALLER) en el momento de su creación y el llamador del programa se ejecuta en el grupo de activación por omisión. Observe que sólo puede especificar ACTGRP(*CALLER) en el mandato CRTBNDRPG si también se especifica DFTACTGRP(*NO).

Nota: los recursos asociados con un programa que se ejecuta en el grupo de activación OPM por omisión con *CALLER no se suprimirán hasta que finalice el trabajo.

Mantenimiento de la compatibilidad entre los programas OPM RPG/400 e ILE RPG

Si tiene una aplicación OPM que consta de varios programas RPG, puede asegurarse de que la aplicación migrada se comportará como una aplicación OPM si crea la aplicación ILE efectuando lo siguiente:

1. Convierta cada miembro fuente OPM utilizando el mandato CVTRPGSRC, y asegúrese de que convierte los miembros /COPY.

Consulte el apartado "Conversión del código fuente" en la página 432 para obtener más información.

Gestión de los grupos de activación

2. Con el mandato CRTBNDRPG, compile y enlace cada miembro fuente convertido por separado en un objeto de programa, especificando DFTACTGRP(*YES).

Para obtener más información acerca de los programas compatibles con OPM, consulte el apartado “Estrategia 1: aplicación compatible con OPM” en la página 25.

Suprimir un grupo de activación

Cuando se suprime un grupo de activación, se liberan sus recursos. Dichos recursos comprenden el almacenamiento estático y los archivos abiertos. Un grupo de activación *NEW se suprime cuando el programa con el que está asociado vuelve a su llamador.

Los grupos de activación con nombre (como QILE) son *persistentes*, ya que no se suprimen a menos que se indique explícitamente o finalice el trabajo. El almacenamiento asociado a los programas que se ejecutan en grupos de activación con nombre no se libera hasta que se suprimen dichos grupos de activación.

El almacenamiento de un programa ILE RPG creado con DFTACTGRP(*YES) se liberará cuando el programa finalice con LR o anormalmente.

Nota: El almacenamiento asociado a programas ILE que se ejecutan en el grupo de activación por omisión mediante *CALLER no se libera hasta que el usuario finalice la sesión (en el caso de un trabajo interactivo) o hasta que el trabajo finalice (en el caso de un trabajo de proceso por lotes).

Si se activan muchos programas ILE RPG (a los que se llama al menos una vez), el almacenamiento del sistema puede agotarse. Por lo tanto, debe evitar tener programas ILE que utilicen una gran cantidad de almacenamiento estático ejecutándose en el grupo de activación OPM por omisión, ya que el almacenamiento no se reclamará hasta que el trabajo finalice.

El almacenamiento asociado con un programa de servicio se reclama solamente cuando el grupo de activación con el que está asociado finaliza. Si se llama al programa de servicio en el grupo de activación por omisión, sus recursos se reclaman cuando el trabajo finaliza.

Se puede suprimir un grupo de activación con nombre utilizando el mandato RCLACTGRP. Utilice este mandato para suprimir un grupo de activación que no sea por omisión y que no se esté utilizando. Este mandato proporciona opciones para suprimir todos los grupos de activación elegibles o para suprimir un grupo de activación por nombre.

Si desea más información sobre el mandato RCLACTGRP, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>. Para obtener más información acerca del mandato RCLACTGRP y los grupos de activación, consulte la publicación *ILE Concepts*.

Mandato Reclamar Recursos

El mandato Reclamar recursos (RCLRSC) está diseñado para liberar los recursos de los programas que ya no están activos. Este mandato funciona de forma distinta según cómo se haya creado el programa. Si se trata de un programa OPM o si el programa se ha creado con DFTACTGRP(*YES), el mandato RCLRSC cerrará los archivos abiertos y liberará el almacenamiento estático.

En el caso de los programas de servicio o ILE que se han activado en el grupo de activación OPM por omisión porque se han creado con *CALLER, los archivos se cerrarán cuando se emita el mandato RCLRSC. Para los programas, el almacenamiento se reinicializará; sin embargo, no se liberará. Para los programas de servicio, el almacenamiento ni se reinicializará ni se liberará.

Nota: esto significa que si tiene un programa de servicio que se ha ejecutado en el grupo de activación por omisión y ha dejado archivos abiertos (al salir con LR desactivado) y se emite RCLRSC, cuando se vuelva a llamar al programa de servicio, parecerá que los archivos continúan abiertos, por lo que las operaciones de E/S darán un error.

En el caso de los programas ILE asociados con un grupo de activación con nombre, el mandato RCLRSC no surte *ningún* efecto. Debe utilizar el mandato RCLACTGRP para liberar recursos en un grupo de activación con nombre.

Si desea más información sobre el mandato RCLRSC, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseres/infocenter>. Para obtener más información acerca del mandato RCLRSC y los grupos de activación, consulte la publicación *ILE Concepts*.

Gestión del almacenamiento asignado dinámicamente

ILE permite gestionar directamente el almacenamiento durante la ejecución desde sus programas mediante la gestión de las áreas de almacenamiento dinámico. Un **área de almacenamiento dinámico** es un área de almacenamiento que se utiliza para las asignaciones del almacenamiento dinámico. La cantidad de almacenamiento dinámico que una aplicación necesita depende de los datos que procesan los programas y los procedimientos que utiliza el área de almacenamiento dinámico.

Para gestionar el almacenamiento dinámico, puede utilizar:

- Los códigos de operación ALLOC, REALLOC y DEALLOC
- Las funciones incorporadas %ALLOC y %REALLOC
- Las API enlazables ILE

No es necesario gestionar de forma explícita el almacenamiento durante la ejecución. Sin embargo, tal vez quiera hacerlo si desea utilizar almacenamiento asignado dinámicamente durante la ejecución. Por ejemplo, puede que desee hacerlo si no sabe con exactitud el tamaño que debe tener una matriz o una estructura de datos que aparece varias veces. Puede definir la matriz o la estructura de datos como BASED, y adquirir el almacenamiento real de la matriz o de la estructura de datos una vez que el programa haya determinado el tamaño que debe tener.

En el sistema hay disponibles dos tipos de áreas de almacenamiento dinámico: por omisión y creada por el usuario. Las operaciones de gestión del almacenamiento RPG utilizan el almacenamiento dinámico por omisión. En los apartados siguientes se describe cómo utilizar las operaciones de gestión de almacenamiento RPG con el almacenamiento dinámico por omisión, y cómo crear y utilizar su propio almacenamiento dinámico con las API de gestión de almacenamiento. Para obtener más información acerca de las áreas de almacenamiento dinámico creadas por el usuario y otros conceptos de gestión del almacenamiento en ILE, consulte la publicación *ILE Concepts*.

Gestión del área de almacenamiento dinámico por omisión utilizando operaciones RPG

La primera petición de almacenamiento dinámico en un grupo de activación da como resultado la creación de un **área de almacenamiento dinámico por omisión** en la que se realiza la asignación del almacenamiento. Las peticiones adicionales de almacenamiento dinámico se satisfacen mediante más asignaciones del área de almacenamiento dinámico por omisión. Si no hay suficiente almacenamiento en el área de almacenamiento dinámico para satisfacer la petición actual de almacenamiento dinámico, el área de almacenamiento dinámico se amplía y se asigna más almacenamiento.

El almacenamiento dinámico asignado permanecerá asignado hasta que se libere explícitamente o hasta que se descarte el área de almacenamiento dinámico. El área de almacenamiento dinámico por omisión sólo se descarta cuando finaliza el grupo de activación al que pertenece.

Todos los programas del mismo grupo de activación utilizan la misma área de almacenamiento dinámico por omisión. Si un programa accede a más almacenamiento del que tiene, puede causar problemas a otro programa. Por ejemplo, supongamos que dos programas, PGM A y PGM B, se están ejecutando en el mismo grupo de activación. Se han asignado 10 bytes para PGM A, pero este programa modifica 11 bytes. Si el byte de más estuviera asignado al programa PGM B, podrían surgir problemas en PGM B.

Puede utilizar las operaciones RPG siguientes en el área de almacenamiento dinámico por omisión:

- El código de operación ALLOC y la función incorporada %ALLOC asignan el almacenamiento en el área de almacenamiento dinámico por omisión.
- El código de operación DEALLOC libera una asignación anterior de almacenamiento dinámico desde cualquier área de almacenamiento dinámico.
- El código de operación REALLOC y la función incorporada %REALLOC cambian el tamaño del almacenamiento asignado anteriormente desde cualquier área de almacenamiento dinámico.

Nota: aunque ALLOC y %ALLOC sólo trabajan con el área de almacenamiento dinámico por omisión, DEALLOC, REALLOC y %REALLOC trabajan con el área de almacenamiento dinámico por omisión y con las áreas de almacenamiento dinámico creadas por el usuario.

La Figura 52 en la página 117 muestra un ejemplo de cómo se pueden utilizar los códigos de operaciones de gestión de memoria para crear una lista enlazada de nombres.

Gestión del almacenamiento asignado dinámicamente

```

*-----*
* Prototipos para los subprocedimientos de este módulo *
*-----*
D AddName      PR
D  name_parm   40A
D Display      PR
D Free         PR
*-----*
* Cada elemento de la lista contiene un puntero hacia el *
* nombre y un puntero hacia el siguiente elemento *
*-----*
D elem         DS      BASED(elem@)
D  name@       *
D  next@       *
D  name_len    5U 0
D nameVal      S      40A  BASED(name@)
D elemSize     C      %SIZE(elem)
*-----*
* El primer elemento de la lista está en almacenamiento estático. *
* El campo del nombre de este elemento no incluye ningún valor. *
*-----*
D first        DS
D              *  INZ(*NULL)
D              *  INZ(*NULL)
D              5U 0 INZ(0)
*-----*
* Este es el puntero al elemento actual. *
* Cuando se configura elem@ como la dirección de <first>, la *
* lista está vacía. *
*-----*
D elem@        S      *  INZ(%ADDR(first))
*-----*
* Incluya 5 elementos en la lista *
*-----*
C              DO      5
C  'Name?'      DSPLY  name      40
C              CALLP   AddName(name)
C              ENDDO
*-----*
* Muestre la lista y libérela. *
*-----*
C              CALLP   Display
C              CALLP   Free
C              EVAL    *INLR = '1'

```

Figura 52. Gestión de memoria - Crear una lista enlazada de nombres (Pieza 1 de 5)

Gestión del almacenamiento asignado dinámicamente

```
*-----*
```

```
* SUBPROCEDIMIENTOS *
```

```
*-----*
```

```
* AddName - añade un nombre al final de la lista *
```

```
*-----*
```

```
P AddName      B
```

```
D AddName      pi
```

```
D name          40A
```

```
*-----*
```

```
* Asigne un nuevo elemento para la matriz, apuntado por el *
```

```
* puntero 'next' del final actual de la lista. *
```

```
*-----*
```

```
* Antes:
```

```
*-----*
```

```
* 
```

```
* |
```

```
* | name *--->abc
```

```
* | name_len 3 |
```

```
* | next *-----|||
```

```
* |
```

```
* |
```

```
* |
```

```
*-----*
```

```
C ALLOC elemSize next@
```

```
*-----*
```

```
* 
```

```
* Después: observe que el elemento anterior es todavía el actual *
```

```
* ya que elem@ todavía está apuntando al elemento anterior *
```

```
*-----*
```

```
* 
```

```
* |
```

```
* | name *--->abc
```

```
* | name_len 3 |
```

```
* | next *----->
```

```
* |
```

```
* |
```

```
* |
```

```
*-----*
```

```
* 
```

```
* Ahora configure elem@ para que apunte el nuevo elemento *
```

```
*-----*
```

```
C EVAL elem@ = next@
```

Figura 52. Gestión de memoria - Crear una lista enlazada de nombres (Pieza 2 de 5)

Gestión del almacenamiento asignado dinámicamente

[illegible]

Figura 52. Gestión de memoria - Crear una lista enlazada de nombres (Pieza 3 de 5)

Gestión del almacenamiento asignado dinámicamente

```

*-----*
* Display - visualiza la lista                                     *
*-----*
P Display          B
D saveElem@        S          *
D dspName          S          40A
*-----*
* Guarde el puntero elem actual para que se pueda restaurar la   *
* lista después de visualizarse, y establezca el puntero de la   *
* lista al principio de la lista.                                *
*-----*
C                  EVAL      saveElem@ = elem@
C                  EVAL      elem@ = %ADDR(first)
*-----*
* Repita en bucle los elementos de la lista hasta que el siguiente*
* puntero sea *NULL                                             *
*-----*
C                  DOW       next@ <> *NULL
C                  EVAL      elem@ = next@
C                  EVAL      dspName = %SUBST(nameVal:1&gm1.name_len)
C      'Name: '      dsply      dspName
C                  ENDDO
*-----*
* Restaure el puntero de la lista a su lugar original           *
*-----*
C                  EVAL      elem@ = saveElem@
P Display          E

```

Figura 52. Gestión de memoria - Crear una lista enlazada de nombres (Pieza 4 de 5)

Gestión del almacenamiento asignado dinámicamente

```

*-----*
* Free - libera el almacenamiento utilizado por la lista      *
*-----*
P Free          B
D prv@          S          *
*-----*
* Repita en bucle los elementos de la lista hasta que el siguiente*
* puntero sea *NULL empezando por el 1er elemento real de la lista*
*-----*
C              EVAL      elem@ = %ADDR(first)
C              EVAL      elem@ = next@
C              DOW        elem@ <> *NULL
*-----*
* Libere el almacenamiento para el nombre                      *
*-----*
C              DEALLOC      name@
*-----*
* Guarde el puntero al elem@ actual
*-----*
C              EVAL      prv@ = elem@
*-----*
* Avance elem@ al siguiente elemento
*-----*
C              EVAL      elem@ = next@
*-----*
* Libere el almacenamiento para el elemento actual
*-----*
C              DEALLOC      prv@
C              ENDDO
*-----*
* Preparado para una nueva lista:
*-----*
C              EVAL      elem@ = %ADDR(first)
P Free          E

```

Figura 52. Gestión de memoria - Crear una lista enlazada de nombres (Pieza 5 de 5)

Problemas de almacenamiento dinámico

La Figura 53 en la página 122 muestra los posibles problemas asociados con una utilización indebida del almacenamiento dinámico.

Gestión del almacenamiento asignado dinámicamente

```
*..1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*-----*
* Uso indebido del almacenamiento dinámico                                     *
*-----*
D Fld1          S          25A    BASED(Ptr1)
D Ptr1          S          *
/
FREE
  Ptr1 = %ALLOC(25);
  DEALLOC Ptr1;

// A partir de este punto, no se debe acceder a Fld1 ya que el
// puntero de base Ptr1 ya no apunta al almacenamiento asignado.

SomePgm();

// Durante la llamada anterior a 'SomePgm', se han podido realizar
// varias asignaciones de almacenamiento. En cualquier caso, es muy peligroso
// realizar la siguiente asignación, ya que 25 bytes de almacenamiento se
// llenarán con 'a'. Es imposible saber qué almacenamiento se está utilizando
// para ello.

Fld1 = *ALL'a';
/END-FREE
```

Figura 53. Uso indebido del almacenamiento dinámico

De forma parecida, se pueden producir errores en los siguientes casos:

- Se puede producir un error similar si se copia un puntero antes de reasignar o desasignar el almacenamiento. Se debe tener cuidado al copiar los punteros en un almacenamiento asignado, y asegurarse de que no se utilizan una vez desasignado o reasignado el almacenamiento.
- Si se copia un puntero a un almacenamiento dinámico, la copia se puede utilizar para desasignar o reasignar el almacenamiento. En este caso, el puntero original no se debe utilizar hasta que se defina con un nuevo valor.
- Si un puntero a un almacenamiento dinámico se pasa como parámetro, el llamador puede desasignar o reasignar el almacenamiento. Una vez devuelta la llamada, si se intenta acceder al puntero se pueden producir problemas.
- Si un puntero a un almacenamiento dinámico se define en *INZSR, un RESET posterior del puntero podría hacer que el puntero se estableciera para un almacenamiento que ya no estuviera asignado.
- Otro tipo de problema podría producirse si se pierde un puntero a un almacenamiento dinámico (por ejemplo, si se ha borrado, o establecido en un nuevo puntero mediante una operación ALLOC). Una vez perdido el puntero, el almacenamiento al que apuntaba no se puede liberar. Este almacenamiento no estará disponible para su asignación, ya que el sistema no sabe que el almacenamiento ya no es direccionable.

El almacenamiento no se liberará hasta que finalice el grupo de activación.

Gestión del propio almacenamiento dinámico utilizando las API enlazables ILE

Puede aislar el almacenamiento dinámico utilizado por algunos programas y procedimientos de un grupo de activación creando uno o más almacenamientos dinámicos creados por el usuario. Para más información sobre la creación de almacenamiento dinámico creado por el usuario, consulte la publicación *ILE Concepts*.

Gestión del almacenamiento asignado dinámicamente

El ejemplo siguiente muestra cómo gestionar el almacenamiento dinámico para una matriz durante la ejecución con un almacenamiento dinámico creado por el usuario desde un procedimiento ILE RPG. En este ejemplo, los procedimientos del módulo DYNARRAY asignan almacenamiento dinámicamente para una matriz empaquetada no enlazada. Los procedimientos del módulo realizan las siguientes acciones en la matriz:

- Inicializar la matriz
- Añadir un elemento a la matriz
- Devolver el valor de un elemento
- Liberar el almacenamiento de la matriz.

DYNARRAY realiza estas acciones utilizando las tres API de almacenamiento enlazables ILE, que son CEECRHP (Crear Área de Almacenamiento Dinámico), CEEGTST (Obtener Almacenamiento) y CEEDSHP (Descartar Área de Almacenamiento Dinámico), así como el código de operación REALLOC. Si desea información específica sobre las API enlazables de gestión del almacenamiento, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iseriess/infocenter>.

La Figura 54 muestra el archivo DYNARRI de /COPY que contiene los prototipos de los procedimientos en DYNARRAY. Este archivo de /COPY es utilizado por el módulo DYNARRAY, así como por otros módulos que llaman a los procedimientos en DYNARRAY.

DYNARRAY se ha definido para utilizarlo con una matriz de decimales empaquetados (15,0). Se puede modificar fácilmente para manejar una matriz de caracteres cambiando la definición de DYNA_TYPE por un campo de caracteres.

```

*=====
* DYNARRAY :      Manejar una matriz empaquetada (15,0) no enlazada
*                  durante la ejecución. El módulo DYNARRAY contiene
*                  procedimientos para asignar la matriz, devolver o
*                  configurar un valor de matriz y desasignar la matriz.
*=====
D DYNA_TYPE      S              15P 0
D DYNA_INIT      PR
D DYNA_TERM      PR
D DYNA_SET       PR
D  Element              VALUE LIKE(DYNA_TYPE)
D  Index              5I 0  VALUE
D DYNA_GET       PR              LIKE(DYNA_TYPE)
D  Index              5I 0  VALUE

```

Figura 54. Archivo DYNARRI de /COPY que contiene los prototipos para el módulo DYNARRAY

La Figura 55 en la página 124 muestra el principio del módulo DYNARRAY que contiene la especificación del control, y las especificaciones de la definición.

Gestión del almacenamiento asignado dinámicamente

```

=====
* DYNARRAY :   Manejar una matriz empaquetada (15,0) no enlazada
*               durante la ejecución. Este módulo contiene
*               procedimientos para asignar la matriz, devolver o
*               configurar un valor de matriz y desasignar la matriz.
=====
H NOMAIN
-----
* Prototipos para los procedimientos de este módulo.
-----
/COPY DYNARRI
-----
* Interfaz a la API CEEGTST (Obtener Almacenamiento de Área de
Almacenamiento Dinámico).
* 1) HeapId = Id del almacenamiento dinámico
* 2) Size   = Número de bytes que se asignan
* 3) RetAddr= Dirección de retorno del almacenamiento asignado
* 4) *OMIT  = Parámetro de retorno. Si se especifica *OMIT aquí,
*             se recibirá una excepción desde la API si
*             no se puede satisfacer la petición.
*             Como no realizamos una supervisión, el procedimiento
*             que llama recibirá una excepción.
-----
D CEEGTST      PR
D  HeapId      10I 0      CONST
D  Size        10I 0      CONST
D  RetAddr     *
D  Feedback    12A        OPTIONS(*OMIT)
-----
* Interfaz a la API CEECRHP (Crear Área de Almacenamiento Dinámico).
* 1) HeapId = Id del almacenamiento dinámico
* 2) InitSize = Tamaño inicial del almacenamiento dinámico.
* 3) Incr     = Número de bytes que se aumenta si se debe
*             ampliar el almacenamiento dinámico.
* 4) AllocStrat = Estrategia de asignación para este almacenamiento
*                 Dinámico. Especificaremos un valor de 0 que permite
*                 al sistema elegir la estrategia óptima.
* 5) *OMIT    = El parámetro de retorno. Si se especifica *OMIT aquí,
*             se recibirá una excepción desde la API si
*             no se puede satisfacer la petición.
*             Como no realizamos una supervisión, el procedimiento
*             que llama recibirá una excepción.
-----
D CEECRHP      PR
D  HeapId      10I 0
D  InitSize    10I 0      CONST
D  Incr        10I 0      CONST
D  AllocStrat  10I 0      CONST
D  Feedback    12A        OPTIONS(*OMIT)

```

Figura 55. Variables globales y prototipos locales para DYNARRAY (Pieza 1 de 2)

Gestión del almacenamiento asignado dinámicamente

```

*-----
* Interfaz a la API CEEDSHP (Descartar Área de Almacenamiento Dinámico).
* 1) HeapId      = Id del almacenamiento dinámico
* 2) *OMIT       = El parámetro de retorno. Si se especifica *OMIT aquí,
*                 se recibirá una excepción desde la API si
*                 no se puede satisfacer la petición.
*                 Como no realizamos una supervisión, el procedimiento
*                 que llama recibirá una excepción.
*-----
D CEEDSHP          PR
D  HeapId          10I 0
D  Feedback        12A      OPTIONS(*OMIT)
*-----
* Variables globales.
*-----
D HeapVars        DS
D  HeapId          10I 0
D  DynArr@         *
*-----
* Definir la matriz dinámica. Codificamos el número de elementos
* como el máximo permitido, teniendo en cuenta que no se declarará
* ningún almacenamiento para esta definición (porque es BASED).
*-----
D DynArr          S          DIM(32767) BASED(DynArr@)
D                  LIKE(DYNA_TYPE)
*-----
* Global, para hacer un seguimiento del número actual de elementos
* en la matriz dinámica.
*-----
D NumElems        S          10I 0 INZ(0)
*-----
* Número inicial de elementos que se asignarán para la matriz,
* y número mínimo de elementos que se añadirán a la matriz
* en las siguientes asignaciones.
*-----
D INITALLOC       C          100
D SUBSALLOC       C          100

```

Figura 55. Variables globales y prototipos locales para DYNARRAY (Pieza 2 de 2)

La Figura 56 en la página 126 muestra los subprocedimientos en DYNARRAY.

Gestión del almacenamiento asignado dinámicamente

```

=====
* DYNA_INIT: Inicializar la matriz.
*
* Función: Crear un área de almacenamiento dinámico y asignar la cantidad
*          inicial de almacenamiento para la matriz durante la ejecución.
=====
P DYNA_INIT      B                      EXPORT
-----
* Variables locales.
-----
D Size           S                      10I 0
*
* Iniciar con un número predeterminado de elementos.
*
C                Z-ADD      INITALLOC      NumElems
*
* Determinar el número de bytes necesarios para la matriz.
*
C                EVAL      Size = NumElems * %SIZE(DynArr)
*
* Crear el área de almacenamiento dinámico
*
C                CALLP      CEECRHP(HeapId : Size : 0 : 0 : *OMIT)

*
* Asignar el almacenamiento y establezca el puntero de base de
* la matriz como el puntero devuelto desde la API.
*
* Observe que el código de operación ALLOC utiliza el almacenamiento
* dinámico por omisión, por lo que debemos utilizar la API CEEGTST para
* especificar otro.
C                CALLP      CEEGTST(HeapId : Size : DynArr@ : *OMIT)

*
* Inicializar el almacenamiento de la matriz.
*
C      1          DO      NumElems      I      5 0
C                CLEAR      DynArr(I)
C                ENDDO
P DYNA_INIT      E

=====
* DYNA_TERM: Terminar el manejo de matrices.
*
* Función: Suprimir el área de almacenamiento dinámico.
=====
P DYNA_TERM      B                      EXPORT
C                CALLP      CEEDSHP(HeapId : *OMIT)
C                RESET      HeapVars
P DYNA_TERM      E

```

Figura 56. Suprocedimientos de DYNARRAY (Pieza 1 de 4)

Gestión del almacenamiento asignado dinámicamente

```

=====
* DYNA_SET: Establecer un elemento de matriz.
*
* Función: Asegurar que la matriz es lo suficientemente grande para
*         este elemento, y establecer el elemento al valor proporcionado.
=====
P DYNA_SET      B      EXPORT
-----
* Parámetros de entrada para este procedimiento.
-----
D DYNA_SET      PI
D  Element      VALUE LIKE(DYNA_TYPE)
D  Index        5I 0  VALUE
-----
* Variables locales.
-----
D Size          S      10I 0
-----
* Si el usuario elige añadirlo a la matriz, deberá comprobar
* primero si la matriz es lo suficientemente grande, y si no, deberá
* aumentar el tamaño. Añada el elemento.
-----
C      Index      IFGT      NumElems
C      EXSR      REALLOC
C      ENDIF
C      EVAL      DynArr(Index) = Element
=====
* REALLOC: Reasignar subrutina de almacenamiento
*
*         Función: Aumentar el tamaño de la matriz dinámica
*         e inicializar los nuevos elementos.
=====
C      REALLOC      BEGSR

*
* Recordar el número anterior de los elementos
*
C      Z-ADD      NumElems      OldElems      5 0

```

Figura 56. Suprocedimientos de DYNARRAY (Pieza 2 de 4)

Gestión del almacenamiento asignado dinámicamente

```

*
* Calcular el nuevo número de elementos. Si el índice es mayor
* que el número actual de elementos en la matriz más la nueva
* asignación, asígnelo hasta el índice; de lo contrario,
* añada una nueva cantidad de asignación en la matriz.
*
C          IF      Index > NumElems + SUBSALLOC
C          Z-ADD   Index      NumElems
C          ELSE
C          ADD     SUBSALLOC   NumElems
C          ENDIF
*
* Calcular el nuevo tamaño de la matriz
*
C          EVAL    Size = NumElems * %SIZE(DynArr)
*
* Reasignar el almacenamiento. El nuevo almacenamiento tiene el mismo valor
* que el antiguo almacenamiento.
*
C          REALLOC  Size      DynArr@
*
* Inicializar los nuevos elementos de la matriz.
*
C      1      ADD    OldElems    I
C      I      DO     NumElems    I      5 0
C          CLEAR   DynArr(I)
C          ENDDO
C          ENDSR
P DYNA_SET      E

```

Figura 56. Suprocedimientos de DYNARRAY (Pieza 3 de 4)

```

=====
* DYNA_GET: Devolver un elemento de matriz.
*
* Función: Devolver el valor actual del elemento de matriz, si el
*          elemento está dentro del tamaño de la matriz o, en caso
*          contrario, el valor por omisión.
*=====
P DYNA_GET      B      EXPORT
*-----
* Parámetros de entrada para este procedimiento.
*-----
D DYNA_GET      PI      LIKE(DYNA_TYPE)
D Index          5I 0    VALUE
*-----
* Variables locales.
*-----
D Element      S      LIKE(DYNA_TYPE) INZ
*-----
* Si el elemento solicitado está dentro del tamaño actual de la
* matriz, devuelva el valor actual del elemento. En caso contrario,
* se puede utilizar el valor por omisión (inicialización).
*-----
C      Index      IFLE    NumElems
C          EVAL    Element = DynArr(Index)
C          ENDIF
C          RETURN   Element
P DYNA_GET      E

```

Figura 56. Suprocedimientos de DYNARRAY (Pieza 4 de 4)

La lógica del subprocedimiento es la siguiente:

Gestión del almacenamiento asignado dinámicamente

1. DYNA_INIT crea el área de almacenamiento dinámico utilizando la API enlazable ILE CEECRHP (Crear Área de Almacenamiento Dinámico), almacenando el Id de almacenamiento dinámico en una variable global HeapId. De esta forma se asigna almacenamiento del área de almacenamiento dinámico en función del valor inicial de la matriz (en este caso 100) llamando a la API enlazable ILE CEEGTST (Obtener Almacenamiento de Área de Almacenamiento Dinámico).
2. DYNA_TERM destruye el almacenamiento dinámico utilizando la API enlazable ILE CEEDSHP (Descartar Área de Almacenamiento Dinámico).
3. DYNA_SET define el valor de un elemento en la matriz.
Antes de añadir un elemento a la matriz, el procedimiento comprueba si hay suficiente almacenamiento en el área de almacenamiento dinámico. Si no, utilice el código de operación REALLOC para adquirir almacenamiento adicional.
4. DYNA_GET devuelve el valor de un elemento especificado. El procedimiento devuelve al llamador el elemento solicitado o ceros. Se devuelven ceros si el elemento solicitado no se ha almacenado en la matriz.

Para crear el módulo DYNARRAY, escriba:

```
CRTRPGMOD MODULE(MIBIB/MATRDIN) SRCFILE(MIBIB/QRPGLESRC)
```

A continuación, el procedimiento se puede enlazar con otros módulos utilizando el mandato CRTPGM o CRTSRVPGM.

La Figura 57 muestra otro módulo que comprueba los procedimientos en DYNARRAY.

```

=====
* DYNTST: Probar el programa para el módulo DYNARRAY.
=====
/COPY EXAMPLES,DYNARRI
D X          S          LIKE(DYNA_TYPE)
* Inicializar la matriz
C              CALLP    DYNA_INIT
* Definir varios elementos
C              CALLP    DYNA_SET (25 : 3)
C              CALLP    DYNA_SET (467252232 : 1)
C              CALLP    DYNA_SET (-2311 : 750)
* Recuperar varios elementos
C              EVAL     X = DYNA_GET (750)
C      '750'      DSPLY   X
C              EVAL     X = DYNA_GET (8001)
C      '8001'     DSPLY   X
C              EVAL     X = DYNA_GET (2)
C      '2'        DSPLY   X

* Limpieza
C              CALLP    DYNA_TERM
C              SETON
LR
```

Figura 57. Módulo de ejemplo utilizando procedimientos en DYNARRAY

Capítulo 10. Llamadas a programas y procedimientos

En ILE, es posible llamar a un programa o a un procedimiento. Y además, ILE RPG proporciona la capacidad de llamar a programas y procedimientos de prototipos o no de prototipos. (Un prototipo es una definición externa de la interfaz de llamada que permite que el compilador compruebe la interfaz durante la compilación).

El método recomendado para llamar a un programa o procedimiento es utilizar una llamada de prototipos. La sintaxis para llamar y pasar parámetros a los procedimientos o programas de prototipos utiliza la misma sintaxis de formato libre que se utiliza con las funciones incorporadas o en las expresiones. Por este motivo, a menudo se hace referencia a una llamada de prototipos como una llamada de formato libre.

Utilice las operaciones CALL o CALLB para llamar a un programa o procedimiento cuando:

- Tiene una interfaz de llamada extremadamente simple
- Necesita la potencia de la operación PARM con el factor 1 y el factor 2.
- Desea más flexibilidad de la que permite la comprobación de parámetros de prototipos.

En este capítulo se describe cómo:

- Llamar a un programa o procedimiento
- Utilizar una llamada de prototipos
- Pasar parámetros de prototipos
- Utilizar una llamada de formato fijo
- Volver de una programa o procedimiento
- Utilizar API enlazables de ILE
- Llamar a una rutina de gráficos
- Llamar a rutinas especiales

Visión general de las llamadas a programas o procedimientos

El proceso de programas en ILE se produce a nivel de procedimiento. Los programas ILE constan de uno o varios módulos que, a su vez, constan de uno o varios procedimientos. Un módulo ILE RPG contiene un procedimiento principal opcional y cero o más subprocedimientos. En este capítulo, el término "procedimiento" se aplica tanto a los procedimientos principales como a los subprocedimientos.

Una "llamada a programa" ILE es una forma especial de llamada a procedimiento; es decir, se trata de una llamada al procedimiento de entrada de programa. Un procedimiento de entrada de programa es el procedimiento diseñado durante la creación del programa para recibir el control cuando se llama a un programa. Si el módulo de entrada del programa es un módulo ILE RPG, el procedimiento de entrada de programa llama al procedimiento principal de dicho módulo inmediatamente después de que se llame al programa.

Esta sección contiene información general sobre:

- La llamada a programa comparada con la llamada a procedimiento

Visión general de las llamadas a programas o procedimientos

- La pila de llamadas (o cómo una serie de llamadas interactúan)
- La recurrencia
- Consideraciones acerca de cómo pasar parámetros

Llamadas a programas

Puede llamar a programas OPM o ILE mediante llamadas a programas. Una **llamada a programa** es una llamada que se hace a un objeto de programa (*PGM). El nombre del programa llamado se resuelve en una dirección durante la ejecución, justo antes de que el programa de llamada pase el control por primera vez al programa llamado. Por este motivo, las llamadas a programas suelen denominarse llamadas dinámicas.

Las llamadas a un programa ILE, a un programa EPM o a un programa OPM son ejemplos de llamadas a programas. Una llamada a una API no enlazable también es un ejemplo de llamada a programa.

Utilice la operación CALLP o las operaciones CALL y PARM para realizar una llamada a programa. Si utiliza las operaciones CALL y PARM, el compilador no puede llevar a cabo la comprobación de tipo en los parámetros, con lo cual pueden producirse errores durante la ejecución.

Cuando se llama a un programa ILE, el procedimiento de entrada de programa recibe los parámetros del programa y se le da el control inicial del programa. Además, todos los procedimientos del programa quedan disponibles para las llamadas a procedimientos.

Llamadas a procedimientos

A diferencia de los programas OPM, los programas ILE no están limitados a utilizar llamadas a programas. También pueden utilizar llamadas estáticas a procedimientos o llamadas a punteros de procedimientos para llamar a otros procedimientos. Las llamadas a procedimientos también reciben el nombre de llamadas enlazadas.

Una **llamada estática a procedimiento** es una llamada a un procedimiento ILE en la que el nombre de éste se resuelve en una dirección durante el enlace (de aquí el término "estático"). Como resultado, durante la ejecución el rendimiento cuando se utilizan llamadas estáticas a procedimientos es más alto que cuando se utilizan llamadas a programas. Las llamadas estáticas permiten el uso de descriptores operativos, la omisión de parámetros y amplían el número de parámetros que se pueden pasar (hasta 399).

Las **llamadas a punteros de procedimientos** proporcionan un modo de llamar dinámicamente a un procedimiento. Por ejemplo, puede pasar un puntero de procedimiento como un parámetro a otro procedimiento que ejecutará el procedimiento que se ha especificado en el parámetro que se ha pasado. También puede manipular matrices de nombres o direcciones de procedimientos para direccionar dinámicamente una llamada de procedimiento a diferentes procedimientos. Si el procedimiento llamado está en el mismo grupo de activación, el coste de una llamada a un puntero de procedimiento es casi el mismo que el coste de una llamada estática a un procedimiento.

Utilizando cualquier tipo de llamada a procedimiento, puede llamar a:

- Un procedimiento en un módulo independiente del mismo programa o programa de servicio ILE.

Visión general de las llamadas a programas o procedimientos

- Un procedimiento de un programa de servicio ILE diferente.

A cualquier procedimiento al que se pueda llamar mediante una llamada estática a procedimiento también se le puede llamar mediante un puntero de procedimiento.

Para obtener una lista de ejemplos de utilización de llamadas a procedimiento, consulte “Ejemplos de llamada de formato libre” en la página 139 y “Ejemplos de CALL y CALLB” en la página 154. Para obtener ejemplos de cómo utilizar punteros de procedimiento, consulte la sección del tipo de datos de puntero de procedimiento del manual *ILE RPG Reference*.

Se utiliza la operación CALLP o las operaciones CALLB y PARM para realizar una llamada a procedimiento. También puede llamar a un procedimiento de prototipos en una expresión si el procedimiento devuelve un valor. Si utiliza las operaciones CALLB y PARM, entonces el compilador no puede realizar la comprobación de tipo de los parámetros, con lo cual pueden producirse errores durante la ejecución.

Pila de llamadas

La **pila de llamadas** es una lista de las entradas de la pila de llamadas en el orden LIFO (último en entrar primero en salir). Una **entrada de pila de llamadas** es una llamada a un programa o procedimiento. Hay una pila de llamadas por cada trabajo.

Cuando se llama a un programa ILE, el primero que se añade a la pila de llamadas es el procedimiento de entrada de programa . A continuación, el sistema efectúa automáticamente una llamada a procedimiento y se añade el procedimiento del usuario asociado (el procedimiento principal). Cuando se llama a un procedimiento, sólo se añade el procedimiento del usuario (un procedimiento principal o subprocedimiento); no hay actividad general de un procedimiento de entrada de programa .

En la Figura 58 en la página 134 se muestra una pila de llamadas en la que un programa OPM llama a un programa ILE. El procedimiento principal RPG del programa ILE llama a un subprocedimiento RPG, el cual a su vez llama a un procedimiento C. Observe que en los diagramas de este manual, la entrada más reciente está situada en la parte inferior de la pila.

Visión general de las llamadas a programas o procedimientos

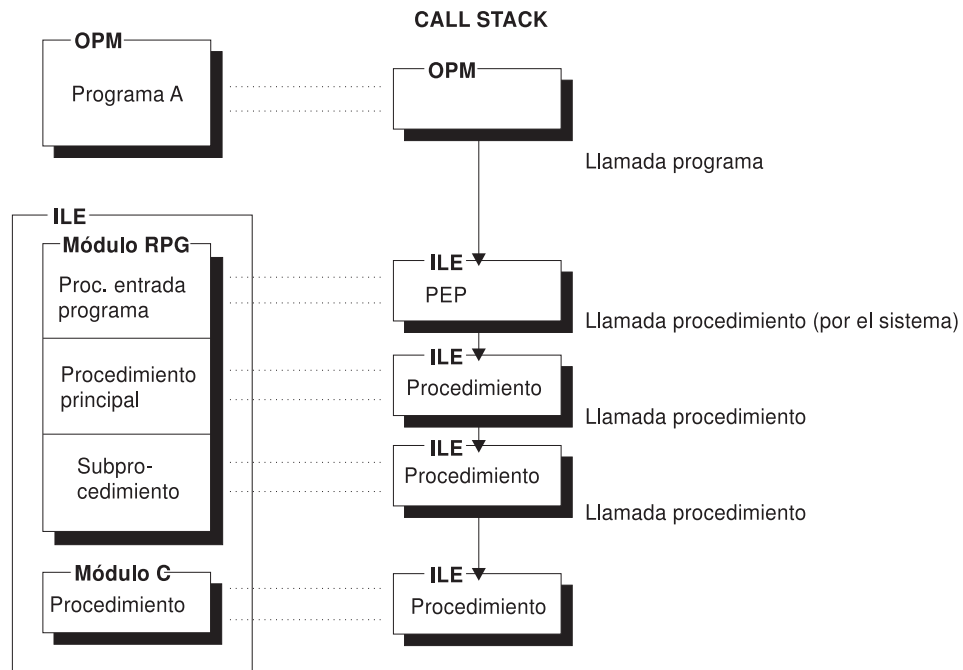


Figura 58. Llamadas a programas y procedimientos en la pila de llamadas

Nota: En una llamada a un programa, las llamadas al procedimiento de entrada de programa y al procedimiento de entrada de usuario (UEP) se producen al mismo tiempo, puesto que la llamada al UEP es automática. Por lo tanto, de ahora en adelante, los dos pasos de una llamada a un programa aparecerán juntos en los diagramas relativos a la pila de llamadas de este capítulo y posteriores.

Llamadas recurrentes

Las llamadas recurrentes sólo se permiten para subprocedimientos. Una **llamada recurrente** es una llamada en la que el procedimiento A se llama a sí mismo o llama al procedimiento B, que a su vez vuelve a llamar al procedimiento A. Cada llamada recurrente crea una nueva invocación del procedimiento que se ha de colocar en la pila de llamadas. La nueva invocación tiene un almacenamiento nuevo para todos los ítems de datos del almacenamiento automático y dicho almacenamiento no está disponible a las demás invocaciones ya que es local. (Un ítem de datos definido en un subprocedimiento utiliza almacenamiento automático a menos que se especifique la palabra clave **STATIC** para la definición.) Tenga en cuenta también que el almacenamiento automático asociado a invocaciones anteriores no resulta afectado por las invocaciones posteriores.

No puede llamarse a un procedimiento principal que esté en la pila de llamadas hasta que se devuelva al llamador. Por lo tanto, preste atención y no llame a otro procedimiento que pueda llamar a un procedimiento principal ya activo.

Intente evitar situaciones que puedan conducir accidentalmente a llamadas recurrentes. Por ejemplo, suponga que existen tres módulos, como se muestra en la Figura 59 en la página 135.

Visión general de las llamadas a programas o procedimientos

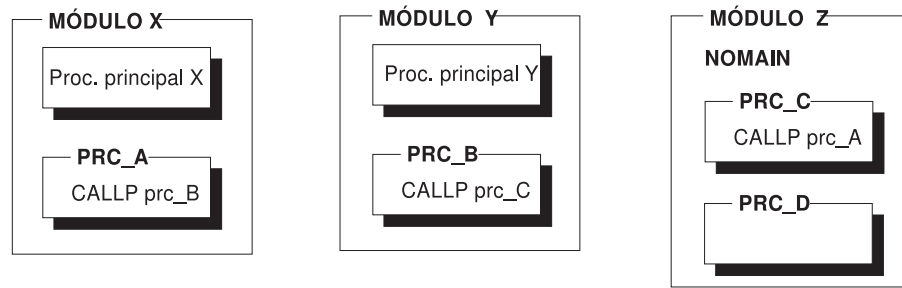
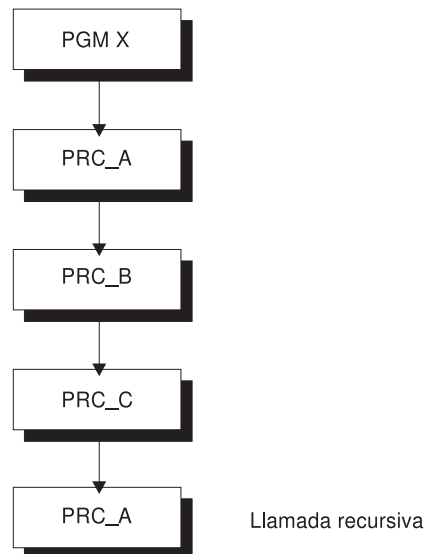


Figura 59. Tres módulos, cada uno con subprocedimientos

Está ejecutando un programa en el que un procedimiento A del módulo X llama al procedimiento B del módulo Y. Desconoce qué realiza el procedimiento B excepto que procesa algunos campos. El procedimiento B devuelve las llamadas al procedimiento C, quien a su vez llama al procedimiento A. Una vez el procedimiento C llama al procedimiento A, se ha realizado una llamada recurrente. La secuencia de pila de llamadas se muestra en la Figura 60. Tenga en cuenta que la entrada de pila de llamadas más reciente se encuentra al final.



Pila de llamadas (la última es la más reciente)

Figura 60. Pila de llamadas recurrentes que se ha de evitar

Por lo tanto aunque puede llamarse de modo recurrente a los subprocedimientos, si desconoce que está produciéndose la recurrencia puede agotar los recursos del sistema.

Atención

Las llamadas recurrentes no condicionales pueden conducir a una recurrencia infinita que puede conllevar el agotamiento de los recursos del sistema. Con una correcta programación se puede evitar la recurrencia infinita. En general, un procedimiento recurrente adecuado comienza con una prueba para determinar si se ha obtenido el resultado deseado. Si se ha obtenido, entonces el procedimiento recurrente regresa al llamador más reciente.

Consideraciones acerca de cómo pasar parámetros

Cuando diseñe una interfaz de llamada, debe realizar un número de decisiones en cuanto a cómo se pasarán los parámetros. Por otro lado, si es el llamador, entonces la mayor parte de las decisiones ya se han tomado en su nombre. La lista siguiente muestra algunas de las consideraciones acerca de cómo pasar parámetros que se han de tener en cuenta cuando se diseña una interfaz de llamada.

- Comprobación de parámetros en tiempo de compilación

La interfaz de llamada es una llamada de prototipos que se comprueba en tiempo de compilación. Esta comprobación asegura que:

- se utilicen correctamente los tipos de datos
- se pasen todos los parámetros necesarios
- sólo se pase *OMIT cuando está permitido.

- Método para pasar parámetros

Cada lenguaje de alto nivel proporciona uno o más métodos para pasar parámetros. Estos pueden ser: pasar un puntero al valor de parámetro, pasar una copia del valor o pasar el valor en sí.

- Pasar descriptores operativos

Algunas veces puede no estar seguro de cuál es el formato exacto de los datos que se le pasan. En este caso puede solicitar que se le pase un descriptor operativo para proporcionarle información adicional en relación al formato de los parámetros que le han pasado.

- Número de parámetros

En general, debe pasar el número de parámetros que espera el programa o procedimiento al que se llama. Si pasa menos parámetros de los que se espera y el programa llamado hace referencia a un parámetro para el que no se han pasado datos, recibirá un error.

- Pasar menos datos

Si pasa un parámetro y pasa muy pocos datos, puede que su aplicación no funcione correctamente. Si cambia el parámetro, puede sobregabar el almacenamiento. Y al utilizar el parámetro, puede malinterpretar el parámetro. Convirtiéndolo en un parámetro de prototipos, el compilador comprobará que la longitud sea la adecuada para el parámetro.

Si el que ha recibido la llamada ha indicado (mediante la documentación o dicho prototipo) que la longitud de un parámetro puede ser menor que la longitud máxima, puede pasar sin problemas parámetros más cortos. (Sin embargo, tenga en cuenta que el procedimiento llamado debe escribirse de modo que maneje menos datos de los necesarios.)

- Orden de evaluación

En una llamada de prototipos no se garantiza el orden de evaluación de los parámetros. Este hecho puede ser importante, si un parámetro se produce más de una vez en la lista de parámetros y si existe la posibilidades de efectos colaterales.

- Consideraciones acerca de llamadas entre lenguajes

Los diferentes lenguajes de alto nivel soportan formas diferentes de representar datos, al igual que formas diferentes de enviar y recibir datos entre programas y procedimientos. En general, debe pasar únicamente los datos cuyo tipo de datos sea común al programa o procedimiento de llamada o llamado, utilizando un método que ambos soporten.

La Tabla 11 en la página 137 asocia las consideraciones anteriores con los dos tipos de parámetros: de prototipos o no de prototipos.

Visión general de las llamadas a programas o procedimientos

Tabla 11. Opciones para pasar parámetros

Opción de parámetros	Con prototipos	Sin prototipos	Ver página
Comprobación de parámetros en tiempo de compilación	Sí		139
Pasar por referencia	Sí	Sí	140
Pasar por valor	Sí (b)		140
Pasar por referencia de sólo lectura	Sí		141
Pasar descriptores operativos	Sí (b)	Sí (b)	142
Pasar *OMIT	Sí (b)	Sí (b)	143
Omisión de parámetros de control	Sí	Sí	144
Obtener número de parámetros pasados	Sí	Sí	145
No permitir longitud de parámetro incorrecta	Sí		150
Nota: (b) se aplica únicamente a los procedimientos enlazados.			

Utilización de una llamada de prototipos

Una llamada de prototipos es una llamada para la que existe un prototipo disponible para efectuar la comprobación de parámetros. Tiene una interfaz de llamada más sencilla y ofrece más funciones. Por ejemplo, utilizando una llamada de prototipos puede llamar (con la misma sintaxis) a:

- Programas que están en el sistema durante la ejecución
- Procedimientos exportados en otros módulos o programas de servicio que estén enlazados en el mismo programa o programa de servicio
- Subprocedimientos del mismo módulo

En RPG, las llamadas de prototipos se conocen también como llamadas de formato libre. La **llamada de formato libre** hace referencia a la sintaxis de llamada en la que se especifican los argumentos de la llamada en una sintaxis de formato libre, de modo muy similar a los argumentos para las funciones incorporadas. Contrasta con la llamada de formato fijo, en la que los argumentos se colocan en especificaciones separadas. Existen dos modos de realizar una llamada de formato libre dependiendo de si se va a utilizar un valor de retorno. Si no hay un valor de retorno, utilice la operación CALLP. Si hay valor de retorno y desea utilizar el valor que se devuelve, coloque el procedimiento de prototipos en una expresión, por ejemplo, en EVAL. Si utiliza CALLP para un procedimiento que devuelve un valor, se ignorará el valor de retorno.

Nota: Sólo los procedimientos de prototipos pueden devolver valores; los programas de prototipos no pueden.

También puede codificar los paréntesis en las llamadas de procedimientos que no tengan ningún parámetro. De esta forma se pueden distinguir más fácilmente las llamadas de procedimientos de los nombres de variables escalares.

Si desea obtener información sobre el paso de parámetros de prototipo, consulte el apartado “Pasar parámetros de prototipos” en la página 139.

Utilización de una llamada de prototipos

Utilización de la operación CALLP

La operación CALLP (Llamada a un Procedimiento de prototipos) para llamar a un programa o procedimiento de prototipos escrito en cualquier lenguaje. La operación CALLP utiliza la siguiente sintaxis de factor 2 ampliado:

```
C                CALLP      NAME{ (PARM1 { :PARM2 ... } ) }
```

En los cálculos de formato libre, se puede pasar por alto CALLP si no hay expansores de operaciones. En las operaciones de formato libre, se puede utilizar uno de los siguientes formatos:

```
/free
  callp name { (parm1 { :parm2 ... } ) };
  name( {parm1 { :parm2 ... } } );
/end-free
```

Para llamar a un programa o procedimiento de prototipos siga estos pasos generales:

1. Incluya el programa o procedimiento de prototipos que se ha de llamar en las especificaciones de definición.
2. Entre el nombre del programa o procedimiento de prototipos en el campo del factor-2 ampliado, seguido por los parámetros, si los hay, entre paréntesis. Separe los parámetros con dos puntos (:). El factor 1 debe estar en blanco.

El ejemplo siguiente muestra una llamada a un conmutador de procedimiento que cambia el estado del indicador que se le pasa, en este caso *IN10..

```
C                CALLP      Switch(*in10)
```

En una llamada a programa se permite un máximo de 255 parámetros y un máximo de 399 en una llamada a procedimiento.

Puede utilizar CALLP desde cualquier parte del módulo. Si se especifica la palabra clave EXTPGM en el prototipo, la llamada será una llamada externa dinámica; de lo contrario, será una llamada a procedimiento enlazada.

Tenga en cuenta que si se utiliza CALLP para llamar a un procedimiento que devuelve un valor, dicho valor no estará disponible para el llamador. Si es necesario el valor, realice la llamada al procedimiento de prototipos en una expresión.

Realización de una llamada en una expresión

Si se ha definido un procedimiento de prototipos para que devuelva un valor, entonces debe realizar la llamada al procedimiento en una expresión si desea utilizar el valor de retorno. Utilice el nombre del procedimiento de modo que sea coherente con el tipo de datos del valor de retorno especificado. Por ejemplo, si se define un procedimiento para que devuelva un valor numérico, entonces la llamada al procedimiento en una expresión debe realizarse donde se espera un valor numérico.

La Figura 61 en la página 139 muestra el prototipo de un procedimiento CVTCHR que toma un parámetro de entrada numérico y devuelve una serie de caracteres. La Figura 62 en la página 139 muestra cómo se puede utilizar el procedimiento en una expresión.

```
* Prototipo para CVTCHR
* - devuelve una representación de caracteres del parámetro
* numérico
* Ejemplos:  CVTCHR(5) devuelve '5'
*            CVTCHR(15-124) devuelve '-109'
D  CVTCHR      PR      31A
D   NUM              30P 0  VALUE
```

Figura 61. Prototipo para CVTCHR

```
C          EVAL      STRING = 'Dirección: ' +
C                                %TRIM(CVTCHR(StreetNum))
C                                + ' ' + StreetName
* Si STREETNUM = 427 y STREETNAME = 'Mockingbird Lane', después
* de la operación EVAL STRING = 'DIRECCIÓN: 427 Mockingbird Lane'
```

Figura 62. Llamada a un procedimiento de prototipos en una expresión

Ejemplos de llamada de formato libre

Para ver ejemplos de utilización de la operación CALLP, consulte:

- Figura 22 en la página 45
- Figura 43 en la página 99
- Figura 121 en la página 246
- Figura 70 en la página 149
- Figura 134 en la página 280

Para consultar ejemplos de llamadas utilizando una expresión, consulte:

- Figura 4 en la página 11
- Figura 19 en la página 42
- Figura 38 en la página 83
- Figura 121 en la página 246

Pasar parámetros de prototipos

Cuando pasa parámetros de prototipos:

- El compilador verifica, cuando compila tanto al que realiza la llamada como al que se llama, que las definiciones de parámetros coincidan, siempre que se compilen los dos utilizando el prototipo.
- Son necesarias pocas especificaciones, ya que no necesita las operaciones PARM.

Esta sección trata sobre las diferentes opciones disponibles cuando define parámetros de prototipos y el impacto de dichas opciones en la interfaz de llamada.

Métodos para pasar parámetros

Las llamadas a programas, incluidas las llamadas a la API del sistema, requieren que los parámetros se pasen por referencia. Sin embargo, dicho requisito no existe para llamadas a procedimientos. ILE RPG permite tres métodos para pasar y recibir parámetros de prototipos

- Por referencia
- Por valor

Pasar parámetros de prototipos

- Por referencia de sólo lectura

Los parámetros que no son de prototipos sólo pueden pasarse por referencia

Pasar por referencia

El método para pasar parámetros por omisión para ILE RPG es pasarlos por referencia. Consecuentemente, no es necesario que codifique las palabras clave de la definición de parámetros para pasar parámetros por referencia. Debe pasar parámetros por referencia a un procedimiento cuando espere que el procedimiento al que se ha llamado modifique el campo pasado. Es posible que también desee pasar por referencia para mejorar el rendimiento durante la ejecución, por ejemplo, cuando pasa campos de caracteres de gran tamaño. También tenga en cuenta que los parámetros que se pasan en las llamadas a programas externos sólo pueden pasarse por referencia.

Pasar por valor

Con un procedimiento de prototipos, puede pasar un parámetro por valor en lugar de por referencia. Cuando se pasa un parámetro por valor, el compilador pasa el valor real al procedimiento llamado.

Nota: Las llamadas a programas OS/400 requieren que dichos parámetros se pasen por referencia. Consecuentemente, no puede pasar un parámetro por valor a un programa.

Pasar por valor le permite:

- Pasar literales y expresiones como parámetros.
- Pasar parámetros que no coincidan exactamente con el tipo y longitud esperados.
- Pasar una variable que, desde el punto de vista del que llama, no se modificará.

Cuando un parámetro se pasa por valor, el programa o procedimiento llamado puede cambiar el valor del parámetro, pero el llamador nunca verá el valor cambiado.

Una ventaja principal de pasar por valor es que permite una coincidencia menos rigurosa de los atributos del parámetro que se ha pasado. Por ejemplo, si la definición es para un campo numérico de tipo decimal empaquetado y longitud 5 con 2 posiciones decimales, debe pasar un valor numérico, pero este puede ser:

- Una constante o variable empaquetada, con zona o binaria, con cualquier número de dígitos y posiciones decimales.
- Una función incorporada que devuelve un valor numérico
- Un procedimiento que devuelve un valor numérico
- Una expresión numérica compleja como por ejemplo
$$2 * (\text{Mín}(\text{Long}(\text{Primer}) + \text{Long}(\text{Ultimo}) + 1) : \%size(\text{Nombre}))$$

Si el prototipo requiere una matriz de 4 elementos, el parámetro pasado puede ser:

- Una matriz con menos de 4 elementos. En este caso, los elementos restantes del parámetro recibido contendrá el valor por omisión para el tipo.
- Una matriz con 4 elementos. En este caso, cada elemento del parámetro recibido corresponderá a un elemento del parámetro pasado.
- Una matriz con más de 4 elementos. En este caso, algunos de los elementos pasados de la matriz no se pasarán al parámetro recibido.
- Un parámetro que no es una matriz. En este caso, cada elemento del parámetro recibido contendrá el valor del parámetro pasado.

Pasar parámetros de prototipos

Para pasar un parámetro por valor, especifique la palabra clave VALUE en la definición de parámetro del prototipo, como se muestra en las figuras siguientes.

```
*-----
* El procedimiento devuelve un valor entero de 10 dígitos.
* Los 3 parámetros son todos enteros de 5 dígitos pasados por
* valor.
*-----
D MiFun          PR          10I 0 EXTPROC('DO_CALC')
D                                     5I 0 VALUE
D                                     5I 0 VALUE
D                                     5I 0 VALUE
....
```

Figura 63. Prototipo para el procedimiento DO_CALC con los parámetros VALUE

```
P DO_CALC          B          EXPORT
*-----
* Este procedimiento realiza una función en 3 valores numéricos
* pasados como parámetros por valor. También devuelve un valor.
*-----
D DO_CALC          PI          10I 0
D   Term1          5I 0 VALUE
D   Term2          5I 0 VALUE
D   Term3          5I 0 VALUE
D Result          S          10I 0
C          EVAL    Result = Term1 ** 2 * 17
C                      + Term2   * 7
C                      + Term3
C          RETURN   Result * 45 + 23
P          E
```

Figura 64. Definición de interfaz de procedimiento para el procedimiento DO_CALC

Pasar por referencia de sólo lectura

Un medio alternativo de pasar un parámetro a un procedimiento de prototipos o programa de prototipos es pasarlo por referencia de sólo lectura. Pasar por referencia de sólo lectura resulta útil, si debe pasar el parámetro por referencia y sabe que el valor del parámetro no se cambiará durante la llamada. Por ejemplo, muchas API del sistema tienen longitudes o formatos de especificación de parámetros de sólo lectura.

Pasar un parámetro por referencia de sólo lectura tiene las mismas ventajas que pasarlo por valor. En especial, este método le permite pasar literales y expresiones. Sin embargo, es importante que sepa que el parámetro no se cambiará durante la llamada.

Cuando se pasa un parámetro por referencia de sólo lectura, el compilador puede copiar el parámetro en un campo temporal y pasar la dirección del temporal. Algunas condiciones que causarían esto son: el parámetro pasado es una expresión o el parámetro pasado tiene un formato diferente.

Nota: Si el programa o procedimiento llamado se compila utilizando un prototipo en un lenguaje que aplica el método de referencia de sólo lectura (ya sea ILE RPG utilizando prototipos o C), entonces no se cambiará el parámetro. Si el programa o procedimiento llamado no utiliza un prototipo, entonces el compilador no puede asegurar que no se modifique el parámetro. En este caso, la persona que define el prototipo debe prestar atención al especificar este método de pasar parámetros.

Pasar parámetros de prototipos

Para pasar un parámetro por referencia de sólo lectura, especifique la palabra clave CONST en la especificación de definición de la definición de parámetro del prototipo. La Figura 65 muestra un ejemplo de una definición de prototipo para la API ILE CEE CEETSTA (Prueba para argumento omitido).

```
*-----*
* CEETSTA (Prueba para argumento omitido) -- ILE CEE API
*   1. Distintivo de presencia                Salida  Binario(4)
*   2. Número de argumento                  Entrada  Binario(4)
*-----*
D CEETSTA          PR          EXTPROC('CEETSTA')
D Present          10I 0
D ArgNum           10I 0  CONST
D Feedback         12A 0  OPTIONS(*OMIT)
...
D HaveParm         S          10I 0
...
C                  CALLP  CEETSTA(HaveParm : 3 : *OMIT)
C                  IF    HaveParm = 1
*                  efectúe alguna acción con el tercer parámetro
C                  ENDIF
```

Figura 65. Prototipo para la API ILE CEE CEETSTA con parámetro CONST

El segundo parámetro pasado a CEETSTA puede ser un campo numérico, un literal, una función incorporada o expresión.

Utilización de descriptores operativos

En algunas ocasiones, es necesario pasar un parámetro a un procedimiento aunque el procedimiento al que se llama no sepa con exactitud cuál es el tipo de datos (por ejemplo, distintos tipos de series). En estos casos, puede utilizar **descriptores operativos** para proporcionar información descriptiva al procedimiento llamado con respecto al formato del parámetro. Esta información adicional permite que el procedimiento interprete correctamente la serie. Sólo debe utilizar descriptores operativos cuando los espera el procedimiento llamado.

Muchas API ILE enlazables esperan descriptores operativos. Si un parámetro se define como 'por descriptor', debe pasar descriptores operativos a la API. Un ejemplo es la API ILE CEE CEEDATM (Convertir segundos a indicación de la hora en caracteres). Los parámetros segundo y tercero requieren un descriptores operativos.

Nota: Actualmente, el compilador ILE RPG únicamente soporta descriptores operativos para los campos y subcampos de gráficos y caracteres. Para estructuras de datos, matrices o tablas no están disponibles los descriptores operativos. Además, los descriptores operativos no están disponibles para datos de tipo numérico de fecha, hora, indicación de la hora, puntero de base o puntero de procedimiento.

Los descriptores operativos no tienen efecto alguno sobre los parámetros que se pasan ni sobre el modo en que se pasan. Cuando a un procedimiento se le pasan descriptores operativos que no espera, dichos descriptores simplemente se ignoran.

Puede solicitar descriptores operativos para parámetros de prototipos y no de prototipos. Para parámetros de prototipos, especifique la palabra clave OPDESC en la definición de prototipos. Para parámetros no de prototipos, especifique (D) como la extensión del código de la operación CALLB. En ambos casos, el procedimiento

que realiza la llamada creará los descriptores operativos y los pasará como parámetros ocultos al procedimiento llamado. No se crearán descriptores operativos para los parámetros omitidos.

Puede recuperar información de un descriptor operativo mediante las API ILEenlazables Recuperar Información de Descriptor Operativo (CEEDOD) y Obtener Información de Descripción acerca de un Argumento de Serie (CEESGI).

Tenga en cuenta que los descriptores operativos sólo están permitidos en llamadas enlazadas. Además, para llamadas no de prototipos, el compilador emitirá un mensaje de error si se especifica una extensión de código de operación 'D' en una operación CALL.

La Figura 66 muestra un ejemplo de la palabra clave OPDESC.

```

*-----
* Len devuelve un valor entero de 10 dígitos. El parámetro
* es una serie de caracteres pasada por referencia de sólo
* lectura. Son necesarios descriptores operativos para que
* Len conozca la longitud del parámetro.

* Es necesario OPTIONS(*VARSIZE) para el parámetro pueda
* ser menor que 32767 bytes.
*-----
D Len          PR          10I 0 OPDESC
D              32767A  OPTIONS(*VARSIZE) CONST

```

Figura 66. Solicitud de descriptores operativos para un procedimiento de prototipos

En el apartado “Programa de servicio de ejemplo” en la página 96 puede encontrar un ejemplo de cómo utilizar los descriptores operativos. Este ejemplo consta de un programa de servicio que convierte series de caracteres que se pasan a dicho programa en su equivalente hexadecimal. El programa de servicio utiliza descriptores operativos para determinar la longitud de la serie de caracteres y la longitud a convertir.

Omisión de parámetros

Cuando se llama a un procedimiento, es posible que a veces se desee omitir un parámetro. Puede que no tenga importancia alguna para el procedimiento llamado. Por ejemplo, esta situación puede darse cuando llama a las API enlazables ILE. Otra razón puede ser que está llamando a un procedimiento anterior que no maneja este parámetro en particular. Si necesita omitir un parámetro en una llamada, tiene dos opciones:

- Especificar OPTIONS(*OMIT) y pasar *OMIT
- Especificar OPTIONS(*NOPASS) y no pasar el parámetro.

La diferencia primordial entre los dos métodos tiene que ver con cómo comprueba si se ha omitido un parámetro. En ambos casos, el procedimiento llamado no puede hacer referencia a un parámetro; de lo contrario, se producirán resultados imprevisibles. Por lo tanto, si se ha diseñado el procedimiento llamado para manejar diferentes números de parámetros, tendrá que comprobar el número de parámetros que se ha pasado. Si se pasa *OMIT, “contará” como un parámetro.

Pasar parámetros de prototipos

Pasar *OMIT

Puede pasar *OMIT para un parámetro de prototipos si el procedimiento llamado sabe que es posible que se pase *OMIT. En otras palabras, puede pasar *OMIT si se especifica la palabra clave OPTIONS(*OMIT) en la definición de parámetro correspondiente del prototipo. Cuando se especifica *OMIT, el compilador generará el código necesario para indicar al procedimiento llamado que se ha omitido el parámetro.

Nota: Sólo puede especificarse *OMIT para parámetros que se han pasado por referencia.

Para determinar si se ha pasado *OMIT a un procedimiento ILE RPG, utilice la función incorporada %ADDR para comprobar la dirección del parámetro en cuestión. Si la dirección es *NULL, entonces se ha pasado *OMIT. También puede utilizar la API enlazable CEETSTA (Comprobar existencia de argumento omitido). (Vea la Figura 65 en la página 142 para ver un breve ejemplo).

A continuación se facilita un ejemplo sencillo de cómo se puede utilizar *OMIT. En este ejemplo, un procedimiento llama a la API ILE CEEDOD enlazable para descomponer un descriptor operativo. La API CEEDOD espera recibir siete parámetros; sin embargo, sólo se han definido seis en el procedimiento de llamada. El último parámetro de CEEDOD (y de la mayoría de API enlazables) es el código de retorno que se puede utilizar para determinar cómo se finaliza la API. Sin embargo, el procedimiento de llamada se ha diseñado para recibir los mensajes de error como excepción en lugar de este código de retorno. Por lo tanto, en la llamada a CEEDOD, el procedimiento debe indicar que se ha omitido el parámetro para el código de retorno.

Consulte el apartado “Programa de servicio de ejemplo” en la página 96 para ver un ejemplo de cómo utilizar *OMIT.

Omitir totalmente parámetros

Otro método de omitir un parámetro es simplemente omitirlo por completo en la llamada. El procedimiento llamado debe esperar esto, lo cual significa que se le debe haber indicado en el prototipo. Para indicar que no tiene que pasarse en una llamada un parámetro de prototipos, especifique la palabra clave OPTIONS(*NOPASS) en la definición de parámetros correspondiente. Tenga en cuenta que todos los parámetros a continuación del primer *NOPASS deben especificarse también con OPTIONS(*NOPASS).

Puede especificar *NOPASS y *OMIT para el mismo parámetro, en cualquier orden, es decir, OPTIONS(*NOPASS:*OMIT) u OPTIONS(*OMIT:*NOPASS).

Como ejemplo de OPTIONS(*NOPASS), puede considerar la API del sistema QCMDEXC (Ejecutar mandato) que tiene un tercer parámetro opcional. Para permitir este parámetro, el prototipo para QCMDEXC puede escribirse como se muestra en la Figura 67 en la página 145.

*-----			
* Este prototipo para QCMDexc define tres parámetros:			
* 1- un campo de caracteres cuya longitud puede ser menor			
* de lo esperado			
* 2- cualquier campo numérico			
* 3- un campo de caracteres opcional			
*-----			
D	qcmdexc	PR	EXTPGM('QCMDexc')
D	cmd	3000A	OPTIONS(*VARSIZE) CONST
D	cmdlen	15P 5	CONST
D		3A	CONST OPTIONS(*NOPASS)

Figura 67. Prototipo para la API del sistema QCMDexc con el parámetro opcional

Comprobación del número de parámetros pasados

A veces, puede ser necesario comprobar el número de parámetros que se han pasado en una llamada. Según como se haya escrito el procedimiento, este número puede permitirle evitar las referencias a parámetros que no se han pasado. Por ejemplo, suponga que desea escribir un procedimiento que en algunas ocasiones pasará tres parámetros y en otras ocasiones pasará cuatro. Esto puede suceder cuando un campo nuevo es obligatorio. Puede escribir el procedimiento llamado de modo que procese un número u otro en función del valor que devuelva la función incorporada %PARMS. Las llamadas nuevas pueden pasar el parámetro. Las llamadas antiguos pueden permanecer sin modificar.

%PARMS no toma ningún parámetro. El valor que devuelve %PARMS también incluye cualquier parámetro para el que se haya pasado *OMIT. Para el procedimiento principal, %PARMS devuelve el mismo valor que el contenido en el campo *PARMS de un PSDS, aunque para utilizar el campo *PARMS, debe codificar también el PSDS.

Para *PARMS y %PARMS, si el número de parámetros pasado es desconocido, se devuelve el valor -1. (Para determinar el número de parámetros pasados, debe pasarse un descriptor operativo como mínimo. ILE RPG siempre pasa uno en una llamada; sin embargo, es posible que otros lenguajes ILE no lo hagan.) Si no está activo el procedimiento principal, *PARMS no tiene fiabilidad. No se recomienda hacer referencia a *PARMS desde un subprocedimiento.

Utilización de %PARMS

En este ejemplo, se ha cambiado varias veces un procedimiento FMTADDR para permitir realizar un cambio en la información de dirección de los empleados de una empresa. Tres procedimientos diferentes llaman a FMTADDR. Los procedimientos difieren únicamente en el número de parámetros que utilizan para procesar la información de empleados. Es decir, han surgido nuevos requisitos para el programa FMTADDR y, para darles soporte, se han añadido parámetros nuevos. Sin embargo, siguen soportados los procedimientos antiguos que llaman a FMTADDR y no es necesario cambiarlos o recompilarlos.

Las modificaciones en las direcciones de los empleados se pueden resumir del modo siguiente:

- Inicialmente, sólo se necesitaba la calle y el número, ya que todos los empleados vivían en la misma ciudad. Por lo tanto, la ciudad y la provincia podían proporcionarse por omisión.

Pasar parámetros de prototipos

- Más tarde, la compañía se amplió, y la información acerca de la ciudad se convirtió en variable en algunas aplicaciones de la empresa.
- Una ampliación posterior hizo que la información de la provincia también fuese variable.

El procedimiento procesa la información basándose en el número de parámetros pasados. El número puede variar entre 3 y 5, e indica al programa si deben proporcionarse los valores de ciudad y/o provincia por omisión. La Figura 68 en la página 147 muestra el fuente para este procedimiento. La Figura 69 en la página 148 muestra el fuente para el miembro /COPY que contiene el prototipo.

La lógica principal de FMTADDR es la siguiente:

1. Compruebe para ver cuántos parámetros se han pasado utilizando %PARMS. Esta función incorporada devuelve el número de parámetros pasados.
 - Si el número es mayor que 4, la provincia por omisión se sustituye por la provincia real proporcionada por el quinto parámetro, que es P_Province.
 - Si el número es mayor que 4, la ciudad por omisión se sustituye por la ciudad real proporcionada por el cuarto parámetro, que es P_City.
2. Se corrige el número de calle para imprimirlo mediante la subrutina GetStreet#.
3. Se concatena la dirección completa.
4. Volver.

```

*=====
* FMTADDR - dar formato a una dirección
*
* Parámetros de interfaz
* 1. Dirección      carácter(70)
* 2. Número de calle empaquetado(5,0)
* 3. Nombre de calle carácter(20)
* 4. Ciudad         carácter(15)  (algunos llamadores no pasan)
* 5. Provincia      carácter(15)  (algunos llamadores no pasan)
*=====
* Extraer el prototipo del miembro /COPY
/COPY FMTADDRP
DfmtAddr      PI
D Address      70
D Street#      5 0 CONST
D Street      20  CONST
D P_City       15  OPTIONS(*NOPASS) CONST
D P_Province   15  OPTIONS(*NOPASS) CONST
*-----*
* Valores por omisión de los parámetros que no se pasan.
*-----*
D City         S      15  INZ('Toronto')
D Province     S      15  INZ('Ontario')
*-----*
* Compruebe si ha pasado el parámetro de provincia. Si ha pasado,
* sustituya el valor por omisión por el del parámetro.
*-----*
C              IF      %PARMS > 4
C              EVAL    Province = P_Province
C              ENDIF
*-----*
* Compruebe si ha pasado el parámetro de ciudad. Si ha pasado,
* sustituya el valor por omisión por el del parámetro.
*-----*
C              IF      %PARMS > 3
C              EVAL    City = P_City
C              ENDIF
*-----*
* Defina 'CStreet#' para que esté en el formato de caracteres de 'Street#'*
*-----*
C              EXSR     GetStreet#
*-----*
* Dé formato a la dirección como Número de calle, ciudad, provincia *
*-----*
C              EVAL    ADDRESS = %TRIMR(CSTREET#) + ' ' +
C                      %TRIMR(CITY) + ', ' +
C                      %TRIMR(PROVINCE)
C              RETURN

```

Figura 68. Fuente para el procedimiento FMTADDR (Pieza 1 de 2)

Pasar parámetros de prototipos

```

=====
* SUBROUTINE: GetStreet#
* Obtenga el formato de caracteres del número de calle, ajustado *
* a la izquierda y con blancos a la derecha. *
=====
C      GetStreet#      BEGSR
C      MOVE      Street#      CStreet#      10
*-----*
* Busque el primer valor distinto de cero. *
*-----*
C      '0'      CHECK      CStreet#      Non0      5 0
*-----*
* Si existe un valor distinto de cero, divida en subseries el *
* número que empieza a partir del valor distinto de cero. *
*-----*
C      IF      Non0 > 0
C      SUBST(P) CStreet#:Non0 CStreet#
*-----*
* Si no hay un valor distinto de cero, utilice '0' como número de la calle*
*-----*
C      ELSE
C      MOVE(P) '0'      CStreet#
C      ENDIF
C      ENDSR

```

Figura 68. Fuente para el procedimiento FMTADDR (Pieza 2 de 2)

```

=====
* Prototipo para FMTADDR - dar formato a una dirección
*-----*
DFmtAddr      PR
D  addr      70
D  strno      5 0 CONST
D  st      20  CONST
D  cty      15  OPTIONS(*NOPASS) CONST
D  prov      15  OPTIONS(*NOPASS) CONST

```

Figura 69. Fuente para el miembro /COPY con prototipo para el procedimiento FMTADDR

La Figura 70 en la página 149 muestra el fuente para el procedimiento PRTADDR. Este procedimiento sirve para ilustrar la utilización de FMTADDR. Para su comodidad, los tres programas que llamarían a FMTADDR se combinan en este único programa. Al mismo tiempo, en este ejemplo, los datos están escritos por programa.

Dado que PRTADDR son 'tres procedimientos en uno', debe definir las estructuras de datos de dirección diferentes. Del mismo modo, existen tres partes de las especificaciones de cálculo, cada una de las cuales corresponde a los programas en cada fase. Después de imprimir la dirección, finaliza el procedimiento PRTADDR.

```

*=====
* PRTADDR - imprimir una dirección
*          Llama a FmtAddr para dar formato a la dirección
*=====
FQSYSPRT  0    F  80      PRINTER
*-----
* Prototipo de FmtAddr
*-----
DFmtAddr      PR
D  addr              70
D  strno              5  0
D  st                20
D  cty               15  OPTIONS(*NOPASS)
D  prov             15  OPTIONS(*NOPASS)
DAddress        S      70
*-----
* Stage1: Estructura de datos de dirección original
* La información variable es sólo la calle y el número.
*-----
D Stage1        DS
D  Street#1      5P 0 DIM(2) CTDATA
D  StreetNam1    20  DIM(2) ALT(Street#1)
*-----
* Stage2: Estructura de datos de dirección revisada como la
* información de la ciudad ahora variable.
*-----
D Stage2        DS
D  Street#2      5P 0 DIM(2) CTDATA
D  Addr2         35  DIM(2) ALT(Street#2)
D  StreetNam2    20  OVERLAY(Addr2:1)
D  City2         15  OVERLAY(Addr2:21)
*-----
* Stage3: Estructura de datos de dirección revisada como la
* información de la provincia ahora variable.
*-----
D Stage3        DS
D  Street#3      5P 0 DIM(2) CTDATA
D  Addr3         50  DIM(2) ALT(Street#3)
D  StreetNam3    20  OVERLAY(Addr3:1)
D  City3         15  OVERLAY(Addr3:21)
D  Province3     15  OVERLAY(Addr3:36)
*-----
* 'Prog. 1'- Añadido el uso de FMTADDR antes del parámetro ciudad. *
*-----
C          DO      2      X      5 0
C          CALLP  FMTADDR (Address:Street#1(X):StreetNam1(X))
C          EXCEPT
C          ENDDO

```

Figura 70. Fuente para el procedimiento PRTADDR (Pieza 1 de 2)

Pasar parámetros de prototipos

```

*-----*
* 'Prog. 2'- Añadido uso de FMTADDR antes del parámetro de provincia. *
*-----*
C          DO          2          X          5 0
C          CALLP      FMTADDR (Address:Street#2(X):
C                      StreetNam2(X):City2(X))
C          EXCEPT
C          ENDDO
*-----*
* 'Prog. 3'- Añadido uso de FMTADDR después de parámetro de provincia.*
*-----*
C          DO          2          X          5 0
C          CALLP      FMTADDR (Address:Street#3(X):
C                      StreetNam3(X):City3(X):Province3(X))
C          EXCEPT
C          ENDDO
C          SETON                      LR
*-----*
* Imprimir la dirección.
*-----*
OQSYSPT  E
0          Address

**
00123Bumble Bee Drive
01243Hummingbird Lane
**
00003Cowslip Street      Toronto
01150Eglinton Avenue     North York
**
00012Jasper Avenue       Edmonton      Alberta
00027Avenue Road         Sudbury        Ontario

```

Figura 70. Fuente para el procedimiento PRTADDR (Pieza 2 de 2)

Para crear estos programas, siga estos pasos:

1. Para crear FMTADDR, utilizando el fuente de la Figura 68 en la página 147, escriba:
CRTRPGMOD MODULE(MYLIB/FMTADDR)
2. Para crear PRTADDR, utilizando el fuente de la Figura 70 en la página 149, escriba:
CRTRPGMOD MODULE(MYLIB/PRTADDR)
3. Para crear el programa, PRTADDR, escriba:
CRTPGM PGM(MIBIB/PRTADDR) MODULE(PRTADDR FMTADDR)
4. Llame al programa PRTADDR. A continuación se muestra la salida:
123 Bumble Bee Drive, Toronto, Ontario
1243 Hummingbird Lane, Toronto, Ontario
3 Cowslip Street, Toronto, Ontario
1150 Eglinton Avenue, North York, Ontario
12 Jasper Avenue, Edmonton, Alberta
27 Avenue Road, Sudbury, Ontario

Pasar menos datos de los necesarios

Cuando un parámetro es de prototipos, el compilador comprobará si la longitud es la adecuada para el parámetro. Si el que ha recibido la llamada ha indicado (mediante la documentación o dicho prototipo) que la longitud de un parámetro puede ser menor que la longitud máxima, puede pasar sin problemas parámetros más cortos.

La Figura 71 muestra el prototipo para QCMDXEC, donde se ha definido el primer parámetro con OPTIONS(*VARSIZE), lo cual significa que puede pasar parámetros de longitudes diferentes para el primer parámetro. Tenga en cuenta que OPTIONS *VARSIZE sólo puede especificarse para un campo de caracteres, de gráficos o para una matriz.

*-----				
* Este prototipo para QCMDXEC define tres parámetros. El				
* primer parámetro puede pasarse con campos de caracteres				
* de longitudes diferentes, ya que se define con *VARSIZE.				
*-----				
D	qcmdexc	PR	EXTPGM('QCMDXEC')	
D	cmd	3000A	OPTIONS(*VARSIZE) CONST	
D	cmdlen	15P 5	CONST	
D		3A	CONST OPTIONS(*NOPASS)	

Figura 71. Prototipo para la API del sistema QCMDXEC con el parámetro *VARSIZE

Orden de evaluación

En una llamada de prototipos no se garantiza el orden de evaluación de los parámetros. Este hecho puede ser importante cuando se utilizan parámetros que pueden ocasionar efectos colaterales, ya que los resultados pueden no ser los que esperaba.

Un **efecto colateral** se produce si el proceso del parámetro cambia:

- El valor de un parámetro de referencia
- El valor de una variable global
- Un objeto externo, como un archivo o área de datos

Al producirse un efecto colateral, si se utiliza el parámetro en otro lugar de la lista de parámetros, el valor utilizado para el parámetro en una parte de la lista puede no ser el mismo que el valor utilizado en la otra parte. Por ejemplo, considere esta sentencia de llamada.

```
CALLP      procA (fld : procB(fld) : fld)
```

Suponga que procA tiene todos los parámetros valor y que procB tiene un parámetro referencia. Suponga también que *fld* comienza con el valor 3 y que procB modifica *fld* para que sea 5, y devuelve 10. Según el orden en que se hayan evaluado los parámetros, procA recibirá 3, 10 y 5 o posiblemente 3, 10 y 3. O posiblemente, 5, 10 y 3; o incluso 5, 10 y 5.

Resumiendo, es importante saber que pueden producirse efectos colaterales. En especial, si está proporcionando una aplicación para el uso de terceros, en la que es posible que el usuario final desconozca los detalles de algunos procedimientos, es importante asegurarse de que los valores de los parámetros pasado sean los esperados.

Llamadas entre lenguajes

Cuando se pase o reciba datos desde un programa o procedimiento escrito en otro lenguaje, es importante saber si los demás lenguajes soportan los mismo métodos para pasar parámetros y los mismo tipos de datos que ILE RPG. La Tabla 12 en la página 152 muestra los diferentes métodos para pasar parámetros que permite ILE RPG y, donde sea aplicable, cómo se codificarían en otros lenguajes ILE. La tabla también incluye el compilador OPM RPG/400 para su comparación.

Pasar parámetros de prototipos

Tabla 12. Métodos para pasar parámetros RPG

<i>Pasar por referencia</i>	
ILE RPG – prototipo	D proc PR D parm 1A C CALLP proc(fld)
ILE C	void proc(char *parm); proc(&fld);
ILE COBOL	CALL PROCEDURE "PROC" USING BY REFERENCE PARM
RPG – sin prototipos	C CALL 'PROC' C PARM FLD
CL de ILE	CALL PROC (&FLD)
<i>Pasar por valor</i>	
ILE RPG – prototipo	D proc PR D parm 1A VALUE C CALLP proc('a')
ILE C	void proc(char parm); proc('a');
ILE COBOL	CALL PROCEDURE "PROC" USING BY VALUE PARM
RPG – sin prototipos	N/A
CL de ILE	N/A
<i>Pasar por referencia de sólo lectura</i>	
ILE RPG – prototipo	D proc PR D parm 1A CONST C CALLP proc(fld)
ILE C	void proc(const char *parm); proc(&fld);
ILE COBOL	N/A ¹
RPG – sin prototipos	N/A
CL de ILE	N/A
Notas: 1. No confunda pasar referencia de sólo lectura con pasando BY CONTENT de COBOL. En términos de RPG, para pasar Fld1 por contenido, debe codificar: <div style="text-align: center;">C PARM Fld1 TEMP</div> Fld1 está protegido contra cambios, pero TEMP no. No se espera que el parámetro no se cambie.	

Para obtener información sobre los tipos de datos soportados por los diferentes lenguajes de alto nivel, consulte el manual del lenguaje correspondiente.

Consideraciones sobre las llamadas entre lenguajes

- Para garantizar que el procedimiento RPG se comunicará correctamente con el procedimiento ILE CL, codifique EXTPROC(*CL: 'procedurename') en el prototipo del procedimiento ILE CL, o en el prototipo del procedimiento RPG que llama el procedimiento ILE CL.

2. Para garantizar que el procedimiento RPG se comunicará correctamente con el procedimiento ILE C, codifique EXTPROC(*CWIDEN:'procedurename') o EXTPROC(*CNOWIDEN:'procedurename') en el prototipo del procedimiento ILE C, o en el prototipo del procedimiento RPG que llama el procedimiento ILE C. Utilice *CNOWIDEN si el fuente de ILE C contiene #pragma argument(procedure-name,nowiden) para el procedimiento; en caso contrario, utilice *CWIDEN.
3. Si desea que el procedimiento RPG sea utilizado correctamente por todos los lenguaje ILE, no especifique ningún valor especial en la palabra clave EXTPROC. En su lugar, evite los siguientes tipos para los parámetros que se pasan por valor o por valores de retorno:
 - Carácter de longitud 1 (1A o 1N)
 - UCS-2 de longitud 1 (1C)
 - Gráfico de longitud 1 (1G)
 - Flotante de 4 bytes (4F)
 - Entero de 1 byte o 2 bytes, o sin signo (3I, 3U, 5I o 5U)
4. Cuando se utiliza ILE C y otros lenguajes, se pueden declarar punteros en memoria teraespacio. ILE C requiere una opción especial de tiempo de compilación para gestionar este tipo de almacenamiento, pero ILE RPG siempre puede gestionar este almacenamiento si se compila con un release de destino de V4R4M0 o posterior. Si desea obtener más información sobre los punteros en teraespacio, consulte la publicación *ILE Concepts*, SC41-5606-05.

Utilización de las operaciones de llamada de formato fijo

La operación CALL (Llamar a un programa) se utiliza para efectuar una llamada a un programa y la operación CALLB (Llamar a un procedimiento enlazado) para efectuar una llamada a programas o procedimientos que no son de prototipos. Las dos operaciones de llamada son muy parecidas en cuanto a la sintaxis y a su uso. Para llamar a un programa o procedimiento, siga estos pasos generales:

1. Identifique el objeto al que se va a llamar en la entrada del Factor 2.
2. Opcionalmente codifique un indicador de error (posiciones 73 y 74) o un indicador LR (posiciones 75 y 76) o ambos.

Cuando un objeto llamado tiene un error, el indicador de error, si se ha especificado, se activa. De forma similar, si el objeto llamado vuelve con el indicador LR activado, éste, si se ha especificado, se activa.

3. Para pasar parámetros al objeto llamado, especifique PLIST en el campo de resultado de la operación de llamada o indique operaciones PARM inmediatamente a continuación de la operación de llamada.

Las dos operaciones transfieren el control del objeto de llamada al objeto que se llama. Después de ejecutar el objeto al que se ha llamado, el control vuelve a la primera operación que se puede procesar después de la operación de llamada del programa o procedimiento de llamada.

Se aplican las consideraciones siguientes a ambas operaciones de llamada:

- La entrada del Factor 2 puede ser una variable, un literal o una constante con nombre. Tenga en cuenta que la entrada es sensible a las mayúsculas y minúsculas.

Sólo para CALL: La entrada del Factor 2 puede ser *nombre de biblioteca/nombre de programa*, por ejemplo, MYLIB/PGM1. Si no se especifica ningún nombre de biblioteca, se utilizará la lista de bibliotecas para localizar el programa. El nombre del programa al que se llama se puede proporcionar en la ejecución especificando una variable de caracteres en la entrada del Factor 2.

Utilización de las operaciones de llamada de formato fijo

Sólo para CALLB: Para efectuar una llamada a un puntero de procedimiento, especifique el nombre del puntero de procedimiento que contiene la dirección del procedimiento al que se llamará.

- Un procedimiento puede contener varias llamadas al mismo objeto con el mismo o varios PLIST especificados.
- Cuando se llama por primera vez a un procedimiento ILE RPG (incluido el procedimiento de entrada de programa), los campos se inicializan y el procedimiento recibe el control. En las llamadas siguientes al mismo procedimiento, si éste no acabó en la llamada anterior, todos los campos, indicadores y archivos del procedimiento al que se ha llamado son los mismos que cuando este procedimiento volvió en la llamada anterior.
- El sistema registra los nombres de todos los programas a los que se llama en un procedimiento RPG. Cuando un procedimiento RPG se enlaza a un programa (*PGM), puede consultar estos nombres utilizando DSPPGMREF, aunque no puede saber qué procedimiento o módulo está realizando la llamada.

Si llama a un programa utilizando una variable, verá una entrada con el nombre *VARIABLE (y sin nombre de biblioteca).

En el caso de un módulo, puede consultar los nombres de los procedimientos a los que se llama con DSPMOD DETAIL(*IMPORT). Algunos procedimientos de esta lista serán procedimientos del sistema; sus nombres suelen contener subrayado o espacios en blanco y no debe prestarles atención.

- **Solo para CALLB:** El compilador crea un descriptor operativo que indica el número de parámetros pasados en la operación CALLB y coloca este valor en el campo *PARMS de la estructura de datos de estado de programa del procedimiento llamado. Este número incluye los parámetros que se designan como omitidos (*OMIT en la operación PARM).

Si se utiliza la extensión de operación (D) con la operación CALLB, el compilador crea también un descriptor operativo para cada campo y subcampo de caracteres y gráficos.

Para obtener más información sobre los descriptores operativos, consulte “Utilización de descriptores operativos” en la página 142.

- Existen más restricciones que se aplican al utilizar los códigos de operación CALL o CALLB. Para obtener una descripción detallada de estas restricciones, consulte la publicación *ILE RPG Reference*.

Ejemplos de CALL y CALLB

Para ver ejemplos de utilización de la operación CALL, consulte:

- El apartado “Ejemplo de fuente para ejemplos de depuración” en la página 245 para ver un ejemplo de llamada a un programa RPG.

Para ver ejemplos de utilización de la operación CALLB, consulte:

- La Figura 45 en la página 101 para ver un ejemplo de llamada a un procedimiento de un programa de servicio.
- La Figura 56 en la página 126 para ver un ejemplo de llamada a las API enlazables.
- El apartado “CLIPRIN: fuente RPG” en la página 390 para ver un ejemplo de un programa de consulta principal que llama a diversos procedimientos RPG.

Pasar parámetros utilizando PARM y PLIST

Cuando pasa parámetros utilizando llamadas de formato fijo, debe pasar parámetros utilizando las operaciones PARM y PLIST. Todos los parámetros se

Utilización de las operaciones de llamada de formato fijo

pasan por referencia. Puede especificar que se pase un descriptor operativo y también puede indicar que se omita un parámetro.

Utilización de la operación PARM

La operación PARM se utiliza para identificar los parámetros que se pasan desde o se reciben en un procedimiento. Cada parámetro se define en una operación PARM por separado. El nombre del parámetro se especifica en el campo de resultado; no es necesario que sea el mismo nombre que figura en el procedimiento de llamada o que se llama.

Las entradas de los factores 1 y 2 son opcionales e indican variables o literales cuyo valor se transfiere a o se recibe de la entrada del campo de resultado en función de si estas entradas se encuentran en el programa o procedimiento de llamada o en el programa o procedimiento llamado. La Tabla 13 muestra cómo se utilizan los factores 1 y 2.

Tabla 13. Significado de las entradas de los factores 1 y 2 en la operación PARM

Estado	Factor 1	Factor 2
En un procedimiento de llamada	Valor transferido desde la entrada del campo de resultado al volver.	Valor situado en la entrada del campo de resultado cuando se produce la llamada.
En un procedimiento llamado	Valor transferido desde la entrada del campo de resultado cuando se produce la llamada.	Valor situado en la entrada del campo de resultado al volver.

Nota: Los movimientos a la entrada del factor 1 o del campo de resultado sólo se producen cuando el procedimiento al que se llama vuelve normalmente al llamador. Si se produce un error al intentar mover datos a una de estas entradas, la operación de mover no se completa.

Si no se especifican suficientes parámetros al llamar a un procedimiento, se producirá un error cuando el procedimiento llamado utilice un parámetro no resuelto. Para evitar este error, puede:

- Comprobar %PARMS para determinar el número de parámetros pasados. Por ejemplo, si utiliza %PARMS, consulte el apartado “Comprobación del número de parámetros pasados” en la página 145.
- Especifique *OMIT en el campo de resultados de las operaciones PARM de los parámetros no pasados. A continuación, el procedimiento llamado puede comprobar si el parámetro se ha omitido verificando si el parámetro tiene el valor *NULL, utilizando %ADDR(parámetro) = *NULL. Para obtener más información, consulte el apartado “Omisión de parámetros” en la página 143.

Recuerde lo siguiente cuando especifique una operación PARM:

- Una o varias operaciones PARM deben seguir inmediatamente a una operación PLIST.
- Una o varias operaciones PARM pueden seguir inmediatamente a una operación CALL o CALLB.
- Cuando se especifica una estructura de datos de apariciones múltiples en el campo de resultado de una operación PARM, todas las apariciones de la estructura de datos se pasan como un solo campo.
- El factor 1 y el campo de resultado de una operación PARM no pueden contener un literal, un campo de consulta anticipada, una constante con nombre ni una palabra reservada de fecha del usuario.

Utilización de las operaciones de llamada de formato fijo

- Se aplican las normas siguientes a *OMIT para parámetros no de prototipos:
 - *OMIT sólo está permitido en operaciones PARM que siguen inmediatamente a una operación CALLB o en una PLIST utilizada con CALLB.
 - Los factores 1 y 2 de una operación PARM deben estar en blanco si se especifica *OMIT.
 - *OMIT no está permitido en una operación PARM que forme parte de una *ENTRY PLIST.
- Existen otras restricciones que se aplican al utilizar el código de operación PARM. Para obtener una descripción detallada de estas restricciones, consulte la publicación *ILE RPG Reference*.

Para ver ejemplos de la operación PARM, véase las figuras siguientes:

- Figura 47 en la página 106
- Figura 42 en la página 97
- Figura 133 en la página 277

Utilización de la operación PLIST

La operación PLIST:

- Define un nombre mediante el cual se puede hacer referencia a una lista de parámetros. Dicha lista se especifica en las operaciones PARM que siguen inmediatamente a la operación PLIST.
- Define la lista de parámetros de entrada (*ENTRY PLIST).

El factor 1 de la operación PLIST debe contener el nombre PLIST. Este nombre se puede especificar en el campo de resultado de una o de varias operaciones de llamada. Si la lista de parámetros es la lista de parámetros de entrada de un procedimiento al que se llama, el factor 1 debe contener *ENTRY.

En un procedimiento pueden aparecer varias PLIST. Sin embargo, sólo puede especificarse un *ENTRY PLIST y sólo en el procedimiento principal.

Para obtener ejemplos de la operación PLIST, consulte la Figura 47 en la página 106 y la Figura 133 en la página 277.

Volver de un programa o procedimiento llamado

Cuando un programa o procedimiento retorna, su entrada de pila de llamadas se elimina de la pila de llamadas. (Si se trata de un programa, se elimina también el procedimiento de entrada de programa). Un procedimiento finaliza anormalmente cuando algún elemento externo al procedimiento finaliza su invocación. Por ejemplo, esto puede suceder si un procedimiento ILE RPG X llama a otro procedimiento (por ejemplo, un procedimiento CL) que emite un mensaje de escape directamente al procedimiento de llamada X. Esto también puede producirse si el procedimiento obtiene una excepción que maneja el manejador de excepciones (un *PSSR o un indicador de error) de un procedimiento que se encuentra en una posición superior en la pila de columnas.

Debido al código de ciclos asociado con los procedimientos principales, su retorno está asociado también con ciertas rutinas de finalización. Esta sección trata sobre los diferentes modos de volver de los procedimientos principales y de los subprocedimientos y sobre las acciones que se producen con cada uno.

Volver de un procedimiento principal

Volver de un procedimiento principal hace que se produzca lo siguiente:

- Si LR está activado, se cierran los archivos y se liberan otros recursos.
- La entrada de la pila de llamadas del procedimiento se elimina de la pila de llamadas.
- Si el procedimiento de entrada de programas ha llamado al procedimiento, entonces debe eliminarse también dicho procedimiento de entrada de programas de la pila de llamadas.

Un procedimiento principal devuelve el control al procedimiento que efectúa la llamada de una de las formas siguientes:

- Con terminación normal
- Con terminación anormal
- Sin terminación.

La siguiente es una descripción de las maneras de volver de un procedimiento principal.

Para obtener una descripción detallada de en qué parte del ciclo del programa RPG se comprueban los indicadores LR, H1 a H9 y RT, véase el apartado acerca del ciclo de programa RPG en la publicación *ILE RPG Reference*.

Terminación normal

Un procedimiento principal finaliza normalmente y el control regresa al procedimiento que efectúa la llamada cuando está activado el indicador LR y los indicadores H1 a H9 no están activados. El indicador LR se puede activar:

- implícitamente, como cuando se procesa el último registro desde un archivo primario o secundario durante el ciclo de programa RPG
- explícitamente, como cuando se activa el indicador LR.

Un procedimiento principal finaliza normalmente si:

- Se procesa la operación RETURN (con factor 2 en blanco), los indicadores H1 a H9 no están activados y el indicador LR está activado.
- El indicador RT está activado, los indicadores H1 a H9 no están activados y el indicador LR está activado.

Cuando un procedimiento principal finaliza normalmente, se produce lo siguiente:

- Se realiza el movimiento del factor 2 al campo de resultado de una operación *ENTRY PARM.
- Se graban todas las matrices y las tablas que tengan un "A nombre de archivo" especificado en las especificaciones de definición y todas las estructuras de datos bloqueadas del área de datos.
- Se desbloquean todas las áreas de datos bloqueadas por el procedimiento.
- Se cierran todos los archivos que estén abiertos.
- Se establece un código de retorno para indicar al llamador que el procedimiento ha finalizado normalmente y se devuelve el control al llamador.

En la llamada siguiente al procedimiento principal, con la excepción de las variables exportadas, hay disponible una copia renovada para su proceso. (Las variables exportadas se inicializan sólo una vez, cuando se activa el programa por primera vez en un grupo de activación. Retienen su último valor asignado en una

Volver de un programa o procedimiento llamado

nueva llamada, incluso si se ha activado LR en la llamada anterior. Si desea reinicializarlas, debe restaurarlas manualmente.)

TIP

Si está acostumbrado a que la terminación con el indicador LR activado haga que se libere el almacenamiento y está trabajando en un grupo de activación, considere la posibilidad de volver sin terminar. Estos son los motivos:

- El almacenamiento no se libera hasta que el grupo de activación finaliza, por lo que la terminación con el indicador LR activado no ofrece ventaja alguna en lo referente al almacenamiento.
- El rendimiento de las llamadas aumenta si el programa no se inicializa de nuevo en cada llamada.

Probablemente, esto sólo le será útil si no necesita volver a inicializar el programa cada vez.

Terminación anormal

Un procedimiento principal finaliza anormalmente y el control vuelve al procedimiento que efectúa la llamada cuando se produce una de las acciones siguientes:

- Se toma la opción de cancelar cuando se emite un mensaje de consulta ILE RPG.
- Se procesa una operación ENDSR *CANCL en una subrutina de error *PSSR o INFSR. (Para obtener más información acerca del punto de retorno de *CANCL para las subrutinas de error *PSSR y INFSR, véase el apartado “Especificación de un punto de retorno en la operación ENDSR” en la página 274).
- Un indicador H1 a H9 está activado cuando se procesa una operación RETURN (con el factor 2 en blanco).
- Un indicador H1 a H9 está activado cuando se efectúa el proceso de último registro (LR) en el ciclo de RPG.

Cuando un procedimiento principal finaliza anormalmente, se produce lo siguiente:

- Se cierran todos los archivos que estén abiertos.
- Se desbloquean todas las áreas de datos bloqueadas por el procedimiento.
- Si el procedimiento principal ha finalizado debido a una respuesta de cancelación a un mensaje de consulta, entonces la finalización anormal la ha causado una comprobación de función. En este caso, se filtra la comprobación de función al llamador. Si ha finalizado debido a la finalización de una subrutina de error con *CANCL, entonces se emite el mensaje RNX9001 directamente al llamador. De lo contrario, el llamador verá la excepción que ha causado la finalización anormal.

En la siguiente llamada al procedimiento, habrá una copia nueva disponible para su proceso. (Para obtener más información sobre manejadores de excepción, consulte el apartado “Utilización de manejadores específicos de RPG” en la página 261.)

Volver sin terminar

Un procedimiento principal puede devolver el control al procedimiento que efectúa la llamada sin finalizar cuando están activados los indicadores LR o H1 a H9 y se ha producido una de las acciones siguientes:

- Se procesa la operación RETURN (con el factor 2 en blanco).

Volver de un programa o procedimiento llamado

- El indicador RT está activado y el control llega a la parte *GETIN del ciclo de RPG, en cuyo caso el control vuelve inmediatamente al procedimiento de llamada. (Para obtener más información sobre el indicador RT, consulte la publicación *ILE RPG Reference*)

Si llama a un procedimiento principal y vuelve sin finalizar, cuando vuelva a llamar al procedimiento, todos los campos, indicadores y archivos del procedimiento tendrán los mismo valores que tenían cuando ha salido del procedimiento. Sin embargo, existen tres excepciones:

- Esto no es así si el programa se está ejecutando en un grupo de activación *NEW, ya que el grupo de activación se suprime cuando el programa vuelve. En este caso, la próxima vez que llame al programa, éste será igual que si hubiera terminado con el indicador LR activado.
- Si comparte archivos, el estado del archivo puede ser diferente del estado que tenía cuando ha salido del procedimiento.
- Si mientras tanto se ha llamado a otro procedimiento del mismo módulo, los resultados son imprevisibles.

Puede utilizar la operación RETURN (con el factor 2 en blanco) o el indicador RT junto con el indicador LR y los indicadores H1 a H9. Debe tener en cuenta la secuencia de prueba del ciclo del programa de RPG para la operación RETURN y los indicadores RT, LR y H1 a H9. Un retorno causará una finalización si está activado el indicador LR o cualquiera de los indicadores de detención y es cierta una condiciones siguientes:

- Se efectúa una operación RETURN
- El indicador RT provocaría un retorno sin una finalización

Volver de un subprocedimiento

Un **subprocedimiento finaliza normalmente** cuando se efectúa una operación RETURN satisfactoriamente o cuando se procesa la última sentencia del procedimiento (no una operación RETURN). Sin embargo, a excepción de la eliminación del subprocedimiento de la pila de llamadas, no se lleva a cabo ninguna acción hasta que finaliza el procedimiento principal. En otras palabras, todas las acciones listadas para la finalización normal de un procedimiento principal se llevan a cabo *únicamente* para el procedimiento principal.

Un **subprocedimiento finaliza anormalmente** y el control se devuelve al procedimiento de llamada cuando se produce una excepción no manejada. Una vez más, no se llevarán a cabo más acciones hasta que finalice el procedimiento principal.

Si no se llama nunca al procedimiento principal (y por lo tanto no puede finalizar), no se cerrarán los archivos, áreas de datos, etc. Si cree que puede ocurrir esto en un subprocedimiento, debe codificar un procedimiento de finalización al que se llamará cuando finalice el subprocedimiento. Esto es ha de ser así, sobretodo si el subprocedimiento se encuentra en un módulo que tiene NOMAIN especificado en la especificación de control.

Volver utilizando API ILE enlazables

Puede finalizar normalmente un procedimiento utilizando la API ILE enlazable CEETREC. Sin embargo, esta API finalizará *todas* las entradas de la pila de llamadas que están en el mismo grupo de activación hasta el límite de control. Cuando se finaliza un procedimiento utilizando CEETREC el proceso de finalización será el *normal*, como se ha descrito anteriormente para los

Volver de un programa o procedimiento llamado

procedimientos principales y subprocedimientos. En la siguiente llamada al procedimiento, habrá una copia nueva disponible para su proceso.

Del mismo modo, puede finalizar anormalmente un procedimiento utilizando la API ILE enlazable CEE4ABN. A continuación, el proceso finalizará anormalmente como se ha descrito anteriormente.

Nota: No puede utilizar ninguna de estas API en un programa creado con DFTACTGRP(*YES), ya que en estos procedimientos no se permiten las llamadas a procedimiento.

Tenga en cuenta que si el procedimiento principal no está activo o si no existe un procedimiento principal, no se cerrará o liberará nada. En este caso, debe habilitar un manejador de excepciones ILE utilizando CEERTX. Si el manejador de excepciones está en el mismo módulo, puede cerrar los archivos, desbloquear las áreas de datos y realizar las demás acciones de finalización.

Si desea más información sobre CEETREC y CEE4ABN, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iserie/infocenter>.

Utilización de API enlazables

Existen interfaces de programación de Aplicaciones (API) enlazables disponibles para todos los lenguajes ILE. En algunos casos, proporcionan funciones adicionales además de las proporcionadas por un lenguaje ILE determinado. También son útiles para las aplicaciones de lenguaje mixto, ya que son independientes de los lenguajes de alto nivel.

Las API enlazables proporcionan una amplia gama de funciones, que incluyen las siguientes:

- Gestión de grupos de activación y flujo de control
- Gestión del almacenamiento
- Gestión de condiciones
- Servicios de mensajes
- Depurador del fuente
- Funciones matemáticas
- Gestión de llamadas
- Acceso a los descriptores operativos

Se accede a las API ILE enlazables utilizando los mismos mecanismos de llamada que utiliza ILE RPG para llamar a procedimientos, es decir, la operación CALLP o la operación CALLB. Si la API devuelve un valor y desea utilizarlo, llame a la API en una expresión. Para ver la información necesaria para definir un prototipo para una API, consulte la descripción de la API en el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/iserie/infocenter>. La Figura 72 en la página 161 muestra una 'llamada' de ejemplo a una API enlazable.

D	CEExxxx	PR	EXTPROC('CEExxxx')
D	parám1 ...		
D	...		
C		CALLP	CEExxxx(parám1 : parám2 : ... : parámn : retorno)
O			
C		CALLB	'CEExxxx'
C		PARM	parám1
C		PARM	parám2
		...	
C		PARM	parámn
C		PARM	retorno

Figura 72. Ejemplo de sintaxis de llamada para las API ILE enlazables

donde

- CEExxxx es el nombre de la API enlazable
- parám1, parám2, ... parám n se pueden omitir o son parámetros necesarios pasados a o devueltos de la API a la que se llama.
- retorno es un código de retorno que se puede omitir y que indica el resultado de la API enlazable.

Nota: No pueden utilizarse las API enlazables si se especifica DFTACTGRP(*YES) en el mandato CRTBNDRPG.

Si desea más información sobre las API enlazables, consulte el apartado *CL y API* de la categoría *Programación* del **iSeries 400 Information Center**, en el siguiente sitio web - <http://www.ibm.com/eserver/series/infocenter>.

Ejemplos de utilización de API enlazables

Para ver ejemplos de la utilización de API enlazables, consulte:

- “Programa de servicio de ejemplo” en la página 96 para ver un ejemplo de la utilización de CEEDOD
- “Gestión del propio almacenamiento dinámico utilizando las API enlazables ILE” en la página 122. para ver un ejemplo de la utilización de CEEGTST, CEEFRST y CEECZST.
- “Utilización de un manejador de condiciones” en la página 275 para ver un ejemplo de la utilización de CEEHDLR y CEEHDLU.
- “Utilización de manejadores de cancelación” en la página 282 para ver un ejemplo de la utilización CEERTX y CEEUTX.

Llamada a una rutina de gráficos

ILE RPG soporta la utilización de la operación CALL o CALLP para llamar a OS/400 Graphics, que incluye el Gestor de representación gráfica de datos (GDDM[®], un conjunto de herramientas rudimentarias de gráficos para realizar imágenes) y las Rutinas de gráficos de presentación (un conjunto de rutinas para diagramas de la empresa). El factor 2 debe contener el literal o la constante con nombre 'GDDM' (no una variable). Utilice las operaciones PLIST y PARM para pasar los parámetros siguientes:

- El nombre de la rutina de gráficos que desea ejecutar.
- Los parámetros apropiados para la rutina de gráficos especificada. Estos parámetros deben ser del tipo de datos que requiera la rutina de gráficos y no pueden tener formato flotante.

Llamada a una rutina de gráficos

El procedimiento que procesa la operación CALL no inicia ni finaliza implícitamente las rutinas de gráficos de OS/400.

Para obtener más información sobre OS/400 Graphics, las rutinas de gráficos y los parámetros, consulte la publicación *GDDM Programming Guide* y *GDDM Reference*.

Nota: Puede llamar a OS/400 Graphics utilizando la operación CALL. También puede utilizar CALLP si define un prototipo para la rutina y especifica la palabra clave EXTPGM en el prototipo. No puede utilizar la operación CALLB. No puede pasar los campos de fecha, hora, indicación de la hora o gráficos a GDDM, ni puede pasar punteros.

Llamada a rutinas especiales

ILE RPG soporta la utilización de las siguientes rutinas especiales utilizando las operaciones CALL y PARM o la operación CALLP

- Rutina de recuperación de mensajes (SUBR23R3)
- Traslado de datos de doble byte entre corchetes y supresión de caracteres de control (SUBR40R3)
- Traslado de datos de doble byte entre corchetes y adición de caracteres de control (SUBR41R3).

Nota: No puede utilizar la operación CALLB para llamar a estas subrutinas especiales. Puede utilizar CALLP si define un prototipo para las subrutinas.

Mientras la rutina de recuperación de mensajes esté soportada, es recomendable utilizar la API de mensajes QMHRTVM, que es más potente.

Del mismo modo, las rutinas SUBR40R3 y SUBR41R3 continúan solamente por motivos de compatibilidad. No se actualizarán para reflejar el nivel de soporte de gráficos proporcionado por RPG IV mediante el nuevo tipo de datos gráficos.

Consideraciones sobre las hebras múltiples

Normalmente, si se ejecuta una aplicación en múltiples hebras se puede mejorar su rendimiento. En el caso de ILE RPG, esto no es cierto en general. De hecho, el rendimiento de una aplicación de hebras múltiples puede ser menor que el de la versión de hebra única, si la seguridad en ejecución multihebra se realiza serializando los procedimientos a nivel de módulo.

La ejecución de procedimientos ILE RPG en un entorno de múltiples hebras sólo se recomienda si es necesario para otros aspectos de la aplicación (por ejemplo, cuando se escribe un programa de salida de Domino o cuando se llama a un procedimiento RPG de ejecución corta desde Java). Para programas RPG de ejecución larga llamados desde Java, recomendamos utilizar otro proceso para el programa RPG.

La palabra clave de especificación de control THREAD(*SERIALIZE) se puede especificar para ayudarle a conseguir la seguridad en ejecución multihebra para un módulo ILE RPG. Si se especifica THREAD(*SERIALIZE), se protegerán la mayoría de variables y las estructuras de control interno contra el acceso incorrecto de múltiples hebras. El módulo de seguro en ejecución multihebra se bloqueará cuando se entre un procedimiento en el módulo y se desbloqueará cuando ya no se esté ejecutando ningún procedimiento en el módulo. Este acceso serializado garantiza que sólo esté activada una hebra en un módulo, dentro de un grupo de

Consideraciones sobre las hebras múltiples

activación, en cada momento. No obstante, es responsabilidad del programador gestionar la seguridad en ejecución multihebra para el almacenamiento compartido entre módulos. Para ello, deberá añadir lógica a la aplicación para sincronizar el acceso al almacenamiento. Por ejemplo, los archivos compartidos, el almacenamiento exportado e importado, y el almacenamiento accedido por la dirección de un parámetro se pueden compartir entre módulos desde múltiples hebras. Para sincronizar el acceso a este tipo de almacenamiento, puede realizar una de las siguientes acciones, o ambas:

- Estructurar la aplicación de tal manera que no se pueda acceder simultáneamente a los recursos compartidos desde múltiples hebras.
- Si va a acceder a los recursos simultáneamente desde hebras diferentes, sincronice el acceso utilizando recursos como, por ejemplo, semáforos o mutexes. Si desea obtener más información, consulte el documento Aplicaciones multihebra, del tema Programación, en la siguiente dirección URL:
<http://www.ibm.com/eserver/series/infocenter>

Cómo compartir datos entre más de un módulo

El acceso serializado a los módulos utilizando la palabra clave de especificación de control THREAD(*SERIALIZE) garantiza el acceso secuencial a los datos globales dentro de cada módulo, pero no garantiza el acceso secuencial a los datos compartidos entre los módulos. Es responsabilidad del programador asegurarse de que sólo una hebra puede acceder a los datos compartidos cada vez.

Dos o más módulos pueden acceder a los mismos datos si:

- Se utilizan las palabras clave EXPORT/IMPORT en las especificaciones de la definición.
- Se comparten los archivos entre los módulos.
- Los datos están basados en un puntero, donde el puntero está disponible a más de un módulo.

Por ejemplo, el procedimiento A en el módulo A pasa un puntero al procedimiento B en el módulo B, y el procedimiento B guarda el puntero en una variable estática. Ahora, los dos módulos tienen acceso al almacenamiento de base al mismo tiempo que la hebra que se ejecuta en el módulo A accede al almacenamiento. Cuando vuelve el procedimiento B, otra hebra puede llamar a un procedimiento en el módulo B y acceder al almacenamiento de base. La serialización del acceso al almacenamiento estático en los módulos A y B no evitará el acceso simultáneo al mismo almacenamiento en cada módulo. A continuación se muestra un ejemplo de dos módulos que pueden acceder a los mismos datos.

Consideraciones sobre las hebras múltiples

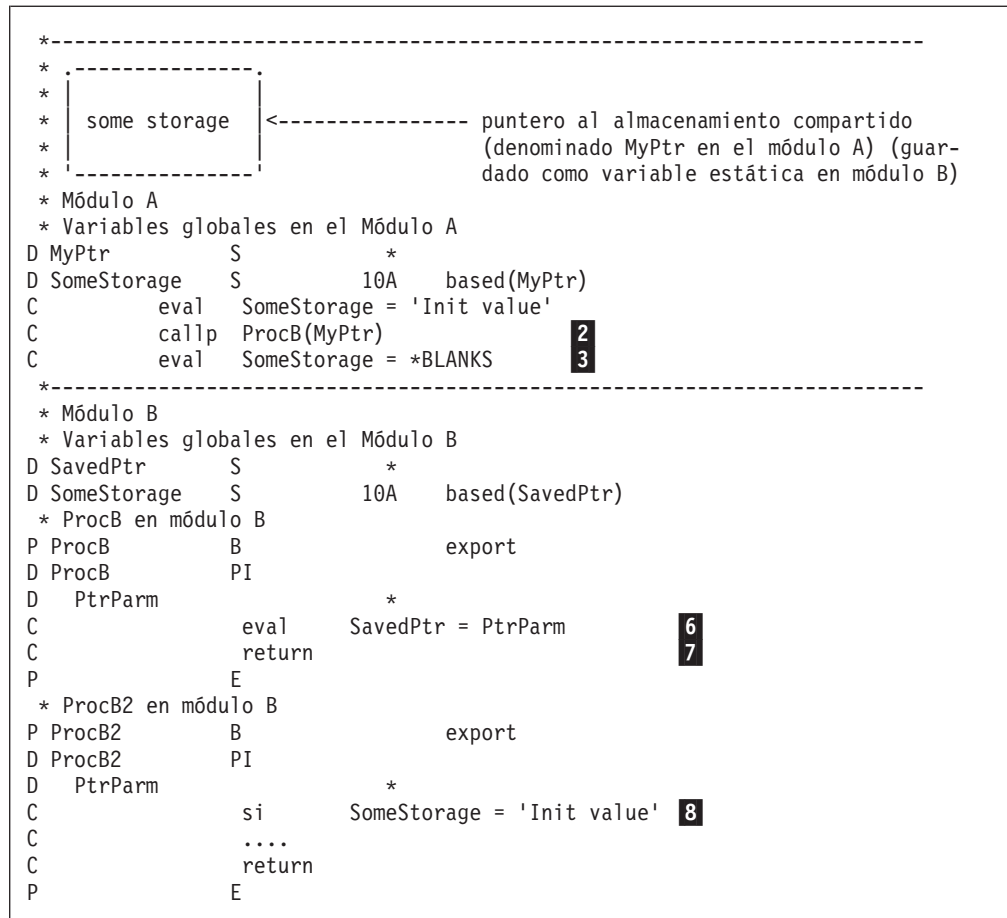


Figura 73. Ejemplo de compartimiento de datos en un entorno multihebra

Cuando el ProcA llama al ProcB (línea **2**), ninguna otra hebra puede acceder al almacenamiento apuntado por MyPtr, ya que tanto el módulo A como el módulo B están siendo usados por una hebra. El ProcB guarda el puntero en el almacenamiento estático del módulo B (línea **6**) y vuelve (línea **7**). Ahora, no hay ninguna hebra activada en el módulo B, por lo que otra hebra puede llamar al módulo B. Si otra hebra llama al ProcB2, es posible que la primera hebra procese la línea **3** antes, en el mismo momento, o después de que la segunda hebra procese la línea **8**. El orden de estos sucesos no está definido; el código utilizado para comprobar si SomeStorage = 'Init value' puede ejecutarse satisfactoriamente una vez y fallar la siguiente.

Para sincronizar el acceso a los datos compartidos, utilice lógica en el programa o las técnicas de sincronización que proporcionan C o las funciones de plataforma. Si desea obtener más información, consulte el documento Aplicaciones multihebra, del tema Programación, en la siguiente dirección URL:

<http://www.ibm.com/eserver/iseres/infocenter>

Cómo evitar los puntos muertos entre módulos

En algunos casos, puede ser necesario controlar la sincronización de los módulos utilizando otros recursos distintos de la palabra clave de especificación de control THREAD(*SERIALIZE). Por ejemplo, considere el caso donde se está llamando al mismo tiempo a dos procedimientos: PROC1 y PROC3. Aunque no hay realmente llamadas recurrentes, si el PROC1 llama al PROC4, esperará a que se desbloquee el MOD2; y si el PROC3 llama al PROC2, esperará a que se desbloquee el MOD1. Los

Consideraciones sobre las hebras múltiples

procedimientos no podrán completar sus llamadas, ya que cada módulo estará bloqueado por la hebra del otro módulo. Este tipo de problema puede producirse incluso con la serialización de llamadas a un módulo, y se denomina punto muerto.

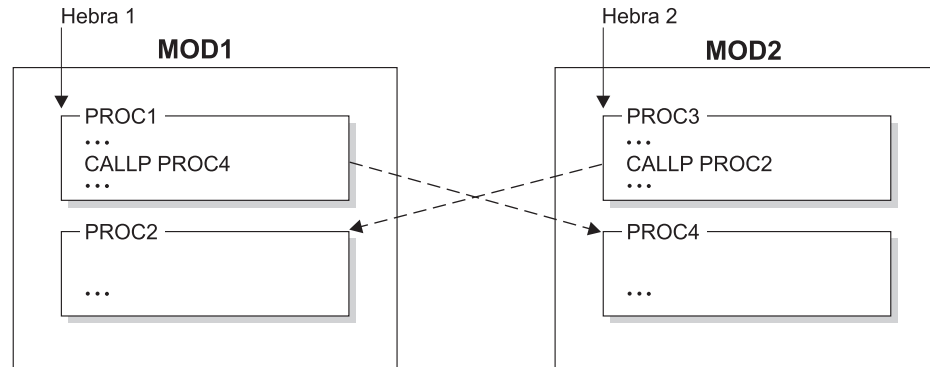


Figura 74. Ejemplo de punto muerto

Este ejemplo muestra que no se puede acceder al mismo tiempo a más de un procedimiento en el mismo módulo utilizando las técnicas de sincronización de ILE RPG.

Para evitar el problema del ejemplo anterior y garantizar aplicaciones seguras en ejecución multihebra, puede controlar la sincronización de los módulos utilizando las técnicas que proporcionan C o las funciones de plataforma. Los llamadores del PROC1 o PROC3 para cada hebra deben realizar las acciones siguientes:

1. Restringir el acceso a los módulos para todas las hebras, excepto para la hebra actual, siempre en el mismo orden (MOD1 y, a continuación, MOD2)
2. En la hebra actual, llamar a los procedimientos de los módulos (PROC1 y PROC3)
3. Renunciar al acceso a los módulos para todas las hebras en el orden inverso al del paso 1 (MOD2 y, a continuación, MOD1).

Una hebra puede restringir satisfactoriamente el acceso al MOD1. Como todos los usuarios del MOD1 y MOD2 utilizan el protocolo de acceso restringido al MOD1 y MOD2, en este orden, ninguna otra hebra podrá llamar a los procedimientos del MOD1 o MOD2 mientras la primera hebra tenga acceso restringido a los módulos. En este caso, tendrá acceso a más de un procedimiento en el mismo módulo al mismo tiempo, pero como sólo estará disponible para la hebra actual, se mantendrá el seguro en ejecución multihebra.

Este método se puede utilizar también para sincronizar el acceso al almacenamiento compartido.

Consideraciones sobre las hebras múltiples

Capítulo 11. RPG y el mundo del eBusiness

En este capítulo se describe cómo se puede utilizar ILE RPG como parte de una solución de eBusiness. Esto incluye la información siguiente

- “RPG y XML”
- “RPG y MQSeries”
- “RPG y Java” en la página 168

RPG y XML

El Lenguaje de marcación extensible (XML) es un subconjunto de SGML que está desarrollado por W3C (World Wide Web Consortium). Su objetivo es habilitar SGML genérico para que se pueda servir, recibir y procesar en la web de la misma forma que se hace con el HTML. XML ha sido diseñado para facilitar la implementación y la interoperabilidad con SGML y HTML.

Si desea obtener más información sobre XML, consulte <http://www.w3.org/XML>

XML para Lenguajes de procedimientos permite a los programas ILE C, RPG y COBOL acceder al XML de IBM para el analizador C++. Con este paquete, los programas RPG pueden crear nuevos documentos XML y analizar los existentes. Puede utilizar XML como almacenamiento de datos y mecanismo de E/S.

XML para Lenguajes de procedimientos está evolucionando continuamente. Si desea obtener la información más reciente, consulte <http://www.alphaWorks.ibm.com/tech/xml4rpg/>

XML para Lenguajes de procedimientos está disponible a través de alphaWorks, que proporciona a los primeros usuarios acceso directo a las nuevas tecnologías de “código alpha” de IBM. Puede bajar el código alpha y participar en los debates en línea con los investigadores y desarrolladores de IBM. Si desea obtener la información más reciente sobre las tecnologías alpha de IBM, consulte <http://www.alphaWorks.ibm.com/>

RPG y MQSeries

Con MQSeries, un programa puede comunicarse con otros programas de la misma plataforma o de otras utilizando el mismo producto de mensajes. MQSeries gestiona interfaces de red, garantiza la entrega, maneja los protocolos de comunicaciones y se encarga de la recuperación si hay problemas del sistema. MQSeries está disponible en más de 35 plataformas.

Si desea un ejemplo de una aplicación iSeries RPG que utilice la API MQSeries, consulte la publicación *MQSeries Application Programming Guide*, SC33-0807-10. Esta publicación también está disponible en línea, en la dirección ibm.com/software/ts/mqseries/library/manualsa/.

Introducción a Java y RPG

El lenguaje de programación Java es un lenguaje de alto nivel orientado a objetos desarrollado por Sun Microsystems. Los programas Java se pueden desarrollar utilizando el componente VisualAge para Java de WebSphere Development Studio para iSeries.

En la programación orientada a objetos, un "método" es un procedimiento programado definido como parte de una "clase", que es un grupo de métodos y variables. Desde el programa RPG se puede llamar a los métodos de Java. Aunque la mayoría de métodos de Java están escritos en Java, un método también se puede escribir en otros lenguajes de alto nivel, como RPG. Éste se conoce como "método nativo". En este apartado se incluye información sobre cómo llamar a los métodos de Java desde RPG, y sobre cómo escribir métodos nativos en RPG.

El tipo de dato de objeto y la palabra clave CLASS

Los campos que pueden almacenar objetos se declaran utilizando el tipo de datos **O**. Para declarar un campo de tipo O, codifique O en la columna 40 de la especificación D y utilice la palabra clave CLASS para proporcionar la clase del objeto. La palabra clave CLASS acepta dos parámetros:

```
CLASS(*JAVA:class_name)
```

*JAVA identifica el objeto como un objeto de Java. Class_name especifica la clase del objeto. Debe ser una constante con nombre o un literal de caracteres, y el nombre de clase debe estar totalmente calificado. El nombre de clase es sensible a mayúsculas y minúsculas.

Por ejemplo, para declarar un campo que mantenga un objeto de tipo BigDecimal:

```
D bdnm          S          0  CLASS(*JAVA:'java.math.BigDecimal')
```

Para declarar un campo que mantenga un objeto de tipo String:

```
D string        S          0  CLASS(*JAVA:'java.lang.String')
```

Tenga en cuenta que los dos nombres de clase están totalmente calificados y que las mayúsculas y minúsculas coinciden exactamente con las de la clase Java.

Los campos de tipo O no se pueden definir como subcampos de estructuras de datos. Es posible tener matrices de campos de tipo O, pero las tablas y matrices de tiempo de compilación y en tiempo de preejecución de tipo O no están permitidas.

Métodos de prototipos de Java

Como los subprocedimientos, los métodos de Java deben ser de prototipos, para poder llamarlos correctamente. El compilador ILE RPG debe conocer el nombre del método, la clase a la que pertenece, los tipos de datos de los parámetros y el tipo de datos del valor de retorno (si existe), y si el método es o no un método estático.

La palabra clave ampliada EXTPROC se puede utilizar para especificar el nombre del método y la clase a la que pertenece. Cuando se convierte un método de Java en un método de prototipos, el formato esperado de la palabra clave EXTPROC es:

```
EXTPROC(*JAVA:class_name:method_name)
```

Tanto el nombre de clase como el nombre de método deben ser constantes de tipo carácter. El nombre de clase debe ser un nombre de clase de Java totalmente

calificado y sensible a mayúsculas y minúsculas. El nombre de método debe ser el nombre del método al que se va a llamar, y es sensible a mayúsculas y minúsculas.

Utilice *JAVA cuando cree un prototipo tanto para el método escrito en Java como para el método nativo escrito en RPG. Utilice la palabra clave STATIC para indicar que el método es estático.

Definiciones y tipos de datos de Java y RPG: Los tipos de datos de los parámetros y el valor de retorno del método se especifican de la misma forma que en los subprocedimientos de prototipos, pero los tipos de datos se correlacionan en realidad con los tipos de datos de Java. En la siguiente tabla se muestra la correlación de los tipos de datos de ILE RPG desde o hacia los tipos de datos de Java.

Tipo de datos de Java	Tipo de datos de ILE RPG	Definiciones de RPG
boolean	indicador	N
byte ¹	entero	3I 0
	carácter	1A
byte[]	longitud de caracteres > 1 (Consulte la nota 3 en la página 170.)	nA
	matriz de longitud de caracteres = 1 (Consulte la nota 4 en la página 170.)	1A DIM(x)
	fecha	D
	hora	T
	indicación de la hora	Z
short	entero de 2 bytes	5I 0
char	longitud de UCS-2 = 1	1C
char[]	longitud de UCS-2 > 1 (Consulte la nota 3 en la página 170.)	nC
	matriz de longitud de UCS-2 = 1 (Consulte la nota 4 en la página 170.)	1C DIM(x)
int	entero de 4 bytes	10I 0
long	entero de 8 bytes	20I 0
float	flotante de 4 bytes	4F
double	flotante de 8 bytes	8F
cualquier objeto	objeto	O CLASS(x)
cualquier matriz	matriz de tipo equivalente (Consulte la nota 4 en la página 170.)	DIM(x)

Notas:

1. Cuando un tipo de byte de Java se convierte desde o hacia un tipo de datos de carácter (1A), se produce una conversión ASCII. Cuando un tipo de byte de Java se convierte desde o hacia un tipo de datos de entero (3I), no se produce la conversión ASCII.
2. Para las matrices de cualquier tipo en Java, puede declararse una matriz de tipo equivalente en RPG. No obstante, tenga en cuenta que no se puede utilizar una matriz de tipos de datos de longitud de caracteres mayor que 1, o longitud de UCS-2 mayor que 1.

3. Para los tipos de datos de longitud de UCS-2 mayor que 1, y longitud de caracteres mayor que 1, se permite la palabra clave VARYING. En general, se recomienda utilizar la palabra clave VARYING, ya que no se puede declarar byte[] ni char[] de Java con una longitud fija.
4. Para los tipos de datos de matriz de RPG, OPTIONS(*VARSIZE) debe estar codificado normalmente para los parámetros de matriz, ya que las matrices de Java no se pueden declarar con una longitud fija.

Los tipos de datos con zona, empaquetados, binarios y sin signo no están disponibles en Java. Si pasa un campo con zona, empaquetado, binario o con signo como parámetro, el compilador realizará la conversión correspondiente, pero esto puede producir una truncación o una pérdida de precisión.

Cuando se llama a un método, el compilador aceptará las matrices como parámetros sólo si el parámetro se convierte en un parámetro de prototipos utilizando la palabra clave DIM.

Si el valor de retorno o el parámetro de un método es un objeto, deberá proporcionar la clase del objeto codificando la palabra clave CLASS en el prototipo. El nombre de clase especificado será el del objeto que se devuelve o el del parámetro que se pasa. (Utilice la palabra clave EXTPROC para especificar la clase del método al que se llama).

Si el método al que se llama es un método estático, deberá especificar la palabra clave STATIC en el prototipo. Si el método es un constructor, deberá especificar *CONSTRUCTOR como nombre del método.

En Java, sólo se pueden pasar por valor los siguientes tipos de datos:

```
boolean
byte
int
short
long
float
double
```

Los parámetros de estos tipos deben tener la palabra clave VALUE especificada para ellos en el prototipo.

Tenga en cuenta que los objetos sólo se pueden pasar por referencia. La palabra clave VALUE no se puede especificar con el tipo O. Como las matrices son vistas como objetos en Java, los parámetros que se correlacionan con las matrices también se deben pasar por referencia. Esto incluye las matrices de byte y caracteres. Se puede utilizar la palabra clave CONST.

Ejemplos de métodos de prototipos de Java: En este apartado se ofrecen algunos ejemplos de métodos de prototipos de Java.

Ejemplo 1: La clase Integer de Java contiene un método estático denominado *toString*, que acepta un parámetro *int* y devuelve un objeto String. Esto se declara en Java como sigue:

```
static String Integer.toString(int)
```

Este método se convertirá en un método de prototipos como sigue:

```
D tostring          PR          0  EXTPROC(*JAVA:
D                               'java.lang.Integer':
D                               'toString')
```

```

D                                CLASS(*JAVA:'java.lang.String')
D                                STATIC
D    num                        10I 0 VALUE

```

La palabra clave EXTPROC identifica el método como un método de Java. También indica que el nombre del método es 'toString', y que se encuentra en la clase 'java.lang.Integer'.

El O en la columna 40 y la palabra clave CLASS indican al compilador que el método devuelve un objeto, y que la clase de este objeto es 'java.lang.String'.

La palabra clave STATIC indica que el método es un método estático, lo que significa que el objeto Integer no es necesario para llamar al método.

El tipo de datos del parámetro se especifica como 10I, que se correlaciona con el tipo de datos *int* de Java. Como el parámetro es *int*, deberá pasarse por valor, y se necesitará la palabra clave VALUE.

Ejemplo 2: La clase Integer de Java contiene un método estático denominado *getInteger*, que acepta los objetos String e Integer como parámetros, y devuelve un objeto Integer. Esto se declara en Java como sigue:

```
static Integer Integer.getInteger(String, Integer)
```

Este método se convertirá en un método de prototipos como sigue:

```

D getint          PR          0  EXTPROC(*JAVA:
D                                'java.lang.Integer':
D                                'getInteger')
D                                CLASS(*JAVA:'java.lang.Integer')
D                                STATIC
D    string        0  CLASS(*JAVA:'java.lang.String') CONST
D    num           0  CLASS(*JAVA:'java.lang.Integer') CONST

```

Este método acepta dos objetos como parámetros. O se codifica en la columna 40 de la especificación D, y la palabra clave CLASS especifica la clase de cada parámetro de objetos. Como los dos parámetros son sólo de entrada, se especifica la palabra clave CONST.

Ejemplo 3: La clase Integer de Java contiene un método denominado *shortValue*, que devuelve la representación abreviada del objeto Integer utilizado para invocar al método. Esto se declara en Java como sigue:

```
short shortValue()
```

Este método se convertirá en un método de prototipos como sigue:

```

D shortval        PR          5I 0 EXTPROC(*JAVA:
D                                'java.lang.Integer':
D                                'shortValue')

```

La palabra clave STATIC no está especificada porque el método no es un método estático. El método no toma parámetros, por lo que ninguno está codificado. Cuando se llama a este método, debe especificarse la instancia Integer como primer parámetro.

El valor de retorno se especifica como 5I, que se correlaciona con el tipo de datos *short* de Java.

Ejemplo 4: La clase Integer de Java contiene un método denominado *equals*, que acepta Object como parámetro y devuelve un booleano. Esto se declara en Java como sigue:

```
boolean equals(Object)
```

Este método se convertirá en un método de prototipos como sigue:

```
D equals          PR          N  EXTPROC(*JAVA:
D                  'java.lang.Integer':
D                  'equals')
D obj              0  CLASS(*JAVA:'java.lang.Object')
```

El valor de retorno se especifica como N, que se correlaciona con el tipo de datos boolean de Java. Como éste no es un método estático, la llamada a este método tendrá dos parámetros con el parámetro de instancia codificado primero.

Llamada a métodos de Java desde ILE RPG

En este apartado se describe cómo se llama a los métodos de Java desde los programas ILE RPG.

Si el método no es estático, se denominará "método de instancia", y se deberá codificar una instancia de objeto como primer parámetro adicional para poder llamar al método. Por ejemplo, si un método de instancia se convierte en un método de prototipos con un parámetro, deberá llamarlo con dos parámetros, siendo el primero el parámetro de instancia.

Los pasos siguientes describen la llamada de ILE RPG a un método de Java:

1. Se puede llamar a los métodos de Java utilizando los códigos de operación existentes CALLP (cuando no se espera ningún valor de retorno) y EVAL (cuando se espera un valor de retorno). Cuando el procedimiento RPG intenta hacer una llamada a un método de Java, RPG comprobará si se ha iniciado la Java Virtual Machine (JVM). Si no, RPG iniciará la JVM. El programador también puede iniciar la JVM utilizando la función JNI que se describe en el apartado "Crear la Java Virtual Machine (JVM)" en la página 184
2. Si ha iniciado la JVM o si sabe que RPG ha iniciado la JVM, puede destruirla cuando haya terminado llamando a Java. Para ello, llame a la función JNI que se describe en el apartado "Destruir la Java Virtual Machine (JVM)" en la página 180.
3. Si está utilizando sus propias clases (o clases distintas de las clases java.xxx normales), asegúrese de que la variable de entorno CLASSPATH esté configurada antes de llamar a ningún método de Java. Cuando RPG inicia la JVM, añada las clases a la variable de entorno CLASSPATH en la vía de acceso de clases estándar, para que cuando el programador utilice sus propias clases, Java pueda encontrarlas. Configure la variable de entorno CLASSPATH de forma interactiva, de la siguiente forma:

```
===>ADDENVVAR ENVVAR(CLASSPATH)
        VALUE('/myclasses/:xyzJava/classes/')
```

Los directorios deben estar separados por dos puntos.

4. Normalmente, Java realiza su propia recogida de basura, detectando los objetos que ya no son necesarios. Cuando se crean objetos llamando a constructores de Java desde el procedimiento RPG no nativo, Java no puede saber si el objeto se puede destruir, por lo que nunca lo destruirá. Si no va a necesitar más un objeto, deberá informar de ello a Java llamando a la función JNI que se describe en el apartado "Indicar a Java que ha terminado con un objeto temporal" en la página 181.

Nota: Como Java utiliza hebras, deberá estar codificada la palabra clave THREAD(*SERIALIZE) en todos los módulos que interactúen con Java.

Consulte el apartado “Codificación RPG adicional para utilizar Java” en la página 180 para obtener más información sobre las distintas funciones JNI.

Ejemplo 1

En este ejemplo, el objetivo es añadir dos valores BigDecimal a la vez. Para ello, deberá crear instancias de dos objetos BigDecimal llamando al constructor de la clase BigDecimal, deberá declarar los campos para almacenar los objetos BigDecimal, y deberá llamar al método add() en la clase BigDecimal.

```

*
* Cree un prototipo para el constructor de BigDecimal que acepte un parámetro
* String. Se devolverá un nuevo objeto BigDecimal.
* Como el constructor no cambia el parámetro String, codificaremos
* la palabra clave CONST. De esta forma será más cómodo
* llamar al constructor.
*
D bdccreate1          PR          0  EXTPROC(*JAVA:
D                               'java.math.BigDecimal':
D                               *CONSTRUCTOR)
D                               CLASS(*JAVA:'java.math.BigDecimal')
D   str                0  CLASS(*JAVA:'java.lang.String')
D                               CONST
*
* Cree un prototipo para el constructor de BigDecimal que acepte un parámetro
* double. 8F se correlaciona con el tipo de datos double de Java, y por lo tanto
* se deberá pasar por VALOR. Se devuelve un objeto BigDecimal.
*
D bdccreate2          PR          0  EXTPROC(*JAVA:
D                               'java.math.BigDecimal':
D                               *CONSTRUCTOR)
D                               CLASS(*JAVA:'java.math.BigDecimal')
D   double              8F  VALUE
*
* Defina los campos para almacenar los objetos BigDecimal.
*
D bdnum1              S          0  CLASS(*JAVA:'java.math.BigDecimal')
D bdnum2              S          0  CLASS(*JAVA:'java.math.BigDecimal')
*
* Como uno de los constructores que estamos utilizando necesita un objeto String,
* también deberemos construir uno de éstos. Cree un prototipo para el constructor de
* String que acepte una matriz de byte como parámetro. Se devolverá un
* objeto String.
*
D makestring          PR          0  EXTPROC(*JAVA:
D                               'java.lang.String':
D                               *CONSTRUCTOR)
D                               CLASS(*JAVA:'java.lang.String')
D   bytes              30A  CONST VARYING
*
* Defina un campo para almacenar el objeto String.
*
D string              S          0  CLASS(*JAVA:'java.lang.String')
*
* Cree un prototipo para el método add de BigDecimal. Éste acepta un objeto
* BigDecimal como parámetro, y devuelve un objeto BigDecimal (la suma del parámetro
* y del objeto BigDecimal utilizado para realizar la llamada).
*
D add                 PR          0  EXTPROC(*JAVA:
D                               'java.lang.BigDecimal':
D                               'add')
D                               CLASS(*JAVA:'java.math.BigDecimal')
D   bd1                0  CLASS(*JAVA:'java.math.BigDecimal') CONST
*
* Defina un campo para almacenar la suma.
*
D sum                 S          0  CLASS(*JAVA:'java.math.BigDecimal')
D
D double              S          8F  INZ(1.1)
D fld1                S          10A

```

Figura 75. Ejemplo de código RPG llamando a la clase de Java BigDecimal

Éste es el código que realiza la llamada.


```

C*
C* Llame al constructor de la clase String, para crear un objeto
C* String desde fld1. Como estamos llamando al constructor, no es
C* necesario pasar un objeto String como primer parámetro.
C*
C          EVAL      string = makestring('123456789012345678901234567890')
C*
C* Llame al constructor de BigDecimal que acepta un parámetro
C* String, utilizando el objeto String del que se ha creado una instancia.
C*
C          EVAL      bdn1 = bdcreate1(string)
C*
C* Nota: Para el objetivo del ejemplo, la construcción del
C* objeto BigDecimal anterior se ha realizado en dos pasos,
C* pero se podría haber hecho en uno, ya que hemos definido
C* el parámetro String para el primero constructor de BigDecimal
C* con la palabra clave CONST.
C*          EVAL      bdn1 = bdcreate1(
C*                  makestring('123456789012345678901234567890'))
C*
C* Llame al constructor de BigDecimal que acepta un parámetro
C* double.
C*
C          EVAL      bdn2 = bdcreate2(double)
C*
C* Añada a la vez los dos objetos BigDecimal llamando al método
C* add. El prototipo indica que add acepta un
C* parámetro, pero como add no es un método estático,
C* debemos pasar también un objeto BigDecimal para realizar
C* la llamada, y se debe pasar como primer parámetro.
C* bdn1 es el objeto que estamos utilizando para realizar
C* la llamada, y bdn2 el parámetro.
C*
C          EVAL      sum = add(bdn1:bdn2)
C* sum contiene ahora un objeto BigDecimal con el valor
C* bdn1 + bdn2.

```

Figura 76.

Ejemplo 2

Este ejemplo muestra cómo ejecutar TRIM en Java utilizando el método trim() como una alternativa a la función incorporada ILE RPG %TRIM. El método trim() en la clase String no es un método estático, por lo que no es necesario un objeto String para realizar la llamada.

```

*
* Defina un campo para almacenar el objeto String que deseamos recortar.
*
D str                      S                      0  CLASS(*JAVA:'java.lang.String')
*
* Cree un prototipo para el constructor de la clase String. El constructor
* espera una matriz de byte.
*
D makestring              PR                      0  EXTPROC(*JAVA:
D                      'java.lang.String':
D                      *CONSTRUCTOR)
D                      CLASS(*JAVA:'java.lang.String')
D parm                    10A
D
*
* Cree un prototipo para el método de String getBytes que convierte una serie en una
* matriz de byte. Después, podemos almacenar esta matriz de byte en un campo alpha.
*
D getBytes                PR                    10A  EXTPROC(*JAVA:
D                      'java.lang.String':
D                      'getBytes') VARYING
*
* Cree un prototipo para el método de String trim. No se necesita ningún parámetro,
* pero como no es un método estático, deberá utilizar un objeto String para
* llamarlo.
*
D trimstring              PR                      0  EXTPROC(*JAVA:
D                      'java.lang.String':
D                      'trim')
D fld                      S                    10A  INZ('  hello  ') VARYING

```

Figura 77. Ejemplo de código RPG utilizando el método de java trim()

La llamada se codifica de la siguiente forma:

```

C*
C* Llame al constructor de String
C*
C                      EVAL      str = makestring(fld)
C*
C* Recorte la serie llamando al método de String trim().
C* Reutilizaremos el campo str para almacenar el resultado.
C*
C                      EVAL      str = trimstring(str)
C*
C* Convierta la serie de nuevo en una matriz de byte y almacénela
C* en fld.
C*
C                      EVAL      fld = getBytes(str)

```

Figura 78. Llamada de RPG al constructor de String

Los métodos estáticos se llaman de la misma forma, aunque no es necesario un objeto para realizar la llamada. Si el método getBytes() anterior fuera estático, la llamada tendría el siguiente aspecto.

```

C                      EVAL      fld = getBytes()

```

Si el método no devuelve un valor, utilice el código de operación CALLP.

Creación de objetos

Para llamar a un método no estático, se necesita un objeto. La clase del objeto debe ser la misma que la de la clase que contiene el método. A veces puede que haya disponible un objeto, pero otras tendrá que crear una instancia de un nuevo objeto. Para ello, llame a un constructor de clases. Un constructor de clases no es un método estático ni un método de instancia, por lo que no necesita un parámetro de instancia. Cuando se crean prototipos en un constructor, se utiliza el nombre de método especial *CONSTRUCTOR.

Por ejemplo, la clase BigDecimal tiene un constructor que acepta un parámetro float.

Este constructor se convertirá en un constructor de prototipos como sigue:

```
D bdcree          PR          0  EXTPROC(*JAVA:
D                                     'java.math.BigDecimal':
D                                     *CONSTRUCTOR)
D   dnum          4F  VALUE
```

Tenga en cuenta que el parámetro se debe pasar por valor, ya que se correlaciona con el tipo de datos float de Java.

La llamada al constructor se realiza de la siguiente forma

```
D bd              PR          0  CLASS(*JAVA:
D                                     'java.math.BigDecimal')
D /free
      bd = bdcree(5.2E9);
D /end-free
```

La clase del objeto de retorno es la misma que la clase del propio constructor, por lo que la palabra clave CLASS es redundante en un constructor, aunque se puede codificar.

Métodos nativos RPG

Para definir un método nativo RPG, se debe codificar el prototipo de la misma forma que se codificaría para un método de Java normal. A continuación, escriba el subprocedimiento RPG como siempre. Debe codificar la palabra clave EXPORT en la especificación de inicio del procedimiento para el método nativo.

Debe tener los métodos nativos en un programa de servicio de la lista de bibliotecas. En la clase de Java que llama a los métodos nativos, debe tener una sentencia como la siguiente:

```
static
{
    System.loadLibrary ("MYSRVPGM");
}
```

Esto permitirá a Java encontrar los métodos nativos. Aparte de añadir *JAVA y la clase a la palabra clave EXTPROC para el prototipo de un método nativo, debe escribir el método nativo como cualquier subprocedimiento. La Figura 79 en la página 178 es un ejemplo de una clase de Java que llama a un método nativo.

```

class MyClass
{
    static
    {
        System.loadLibrary ("MYSRVPGM");
    }

    native boolean checkCust (byte custName[]);

    void anotherMethod ()
    {
        boolean found;
        // call the native method
        found = checkCust (str.getBytes());
    }
}

```

Figura 79. Clase de Java llamando a un método nativo

La Figura 80 es un prototipo de un método nativo RPG.

D checkCust	PR	N	EXTPROC(*JAVA
D			: 'MyClass'
D			: 'checkCust')
D custName		100A	VARYING CONST

Figura 80. Prototipo de método nativo RPG

El método nativo se codifica como cualquier subprocedimiento. La Figura 81 es un ejemplo de un método nativo codificado en RPG.

P checkCust	B		EXPORT
D checkCust	PI	N	
D custName		100A	VARYING CONST

```

/free
  chain custName rec;
  return %found;
/end-free

P checkCust      E

```

Figura 81. Método nativo codificado en RPG

Java llama al programa de servicio desde el grupo de activación por omisión. Si el programa de servicio se ha creado con un grupo de activación *CALLER, se ejecutará en el grupo de activación por omisión. Esto puede causar a veces algunos problemas:

- Si está depurando los métodos nativos, y desea cambiar el código, deberá finalizar la sesión y volver a iniciarla antes de que Java pueda ver la nueva versión.
- Si está llamando a otros procedimientos del programa de servicio desde otro código RPG que no se está ejecutando en el grupo de activación por omisión, no podrá compartir las variables globales entre los "procedimientos normales" y los métodos nativos. Esto puede ocurrir si un procedimiento del programa de

servicio de RPG configura varias variables globales y, a continuación, llama a la clase de Java que, a su vez, llama a un método nativo en el programa de servicio. Estos métodos nativos no podrán ver los mismos datos que el primer procedimiento configurado.

Si crea objetos de Java en los métodos nativos, por omisión serán destruidos por Java cuando vuelva el método nativo. Si desea que el objeto esté disponible después de volver el método nativo (por ejemplo, si desea utilizarlo posteriormente desde otro método nativo), deberá indicar a Java que desea realizar una referencia global, llamando al procedimiento empaquetador de JNI `getNewGlobalRef`. Cuando haya terminado con la referencia global, deberá llamar al procedimiento empaquetador de JNI `freeGlobalRef`, para que Java puede reclamar el objeto. Consulte los apartados “Indicar a Java que desea que un objeto sea permanente” en la página 182 y “Indicar a Java que ha terminado con un objeto permanente” en la página 183 para obtener más información sobre estos procedimientos empaquetadores.

Si el método nativo RPG finaliza anormalmente con una excepción no manejada, el compilador RPG lanzará una excepción a Java. La excepción será de clase `java.lang.Exception`, y tendrá la forma `RPG nnnnn`, donde `nnnnn` es el código de estado RPG.

```
try
{
    nativeMethod ();
}
catch (Exception exc)
{
    ...
}
```

Cómo obtener el parámetro de instancia en los métodos nativos no estáticos

Cuando se llama a un método nativo no estático, uno de los parámetros que Java pasa al método nativo es el objeto al que se aplica el método. Éste se denomina “parámetro de instancia”, referido como “this” en un método de Java. Dentro del propio método nativo, se puede utilizar la función incorporada `%THIS` para obtener el parámetro de instancia. No debe codificar este parámetro en la Interfaz de procedimiento.

Cómo pasar parámetros de tipo carácter desde Java a los métodos nativos

Tiene dos opciones cuando se trabaja con parámetros de tipo carácter:

- Si desea que el código de Java sea lo más sencillo posible, defina el parámetro como una serie en la declaración del método nativo de Java. El código RPG deberá recuperar el valor de la propia serie (véase “Utilización de objetos String en RPG”).
- Si desea que los datos de tipo carácter estén inmediatamente disponibles para el programa RPG, codifique el parámetro en la declaración del método nativo de Java como una matriz `byte` o `char`, y codifíquelo en el prototipo RPG como un campo de caracteres, un campo UCS-2 o uno de Fecha, Hora o Indicación de la hora. De esta forma, RPG se encargará de la conversión.

Utilización de objetos String en RPG: Si tiene un objeto String en el código RPG, puede recuperar su longitud y contenido utilizando el código de la Figura 82 en la página 180.

```

D stringBytes    PR          100A    VARYING
D
D
D
D
D stringLength  PR
D
D
D
D
D string        S
D len           S
D data          S          100A    VARYING

/free
  len = stringLength (string);
  data = stringBytes (string);
  if (len > %len(data));
    error ('Actual string was too long');
  endif;
/end-free

```

Figura 82. Recuperación de la longitud del objeto String y el contenido desde Java

Se puede definir el valor de retorno del método `getBytes` como un dato de tipo carácter de cualquier longitud, variable o no variable, y elegirla a partir de la longitud conocida de los datos de la serie de Java. También se puede definir el valor de retorno como Fecha, Hora o Indicación de la hora, si está seguro de que el objeto String tendrá el formato correcto.

De manera alternativa, puede recuperar el valor de la serie como UCS-2, llamando al método `getChars` en lugar del `getBytes`.

Codificación RPG adicional para utilizar Java

Si utiliza ILE RPG con Java, hay algunas funciones normalmente manejadas por Java que deberán ser manejadas por el código RPG. El compilador de RPG se encargará de algunas funciones, pero el programador deberá manejar otras. En este apartado se muestran algunos ejemplos de empaquetadores de RPG que realizan este trabajo, se explica cómo y cuándo se deben llamar, y se recomienda cómo manejar las excepciones de JNI.

Se describen los siguientes empaquetadores de funciones JNI:

- “Destruir la Java Virtual Machine (JVM)”
- “Indicar a Java que ha terminado con un objeto temporal” en la página 181
- “Indicar a Java que desea que un objeto sea permanente” en la página 182
- “Indicar a Java que ha terminado con un objeto permanente” en la página 183
- “Crear la Java Virtual Machine (JVM)” en la página 184
- “Obtener un puntero de entorno JNI” en la página 184

Destruir la Java Virtual Machine (JVM)

Si está seguro de que ha terminado con la JVM en el trabajo, puede destruirla. Para ello, llame al procedimiento de empaquetador de RPG `destroyJVM`.

```
CALLP  destroyJVM();
```

La Figura 83 en la página 181 contiene un código fuente de ejemplo para `destroyJVM`.

```

/*-----*/
/* destroyJVM                                     */
/*-----*/
P destroyJVM      B      EXPORT
D destroyJVM      PI      N

D rc              S      LIKE(jint)
D rpgRc           S      N
D jvm             S      *    DIM(1)
D env            S      *
D buflen         S      LIKE(jsize) INZ(%elem(jvm))
D nVMs           S      LIKE(jsize)

/free
//-----
// Observe si hay una JVM iniciada
//-----
rc = JNI_GetCreatedJavaVMs (jvm : buflen : nVMs);
//-----
// Si hay una JVM iniciada, destrúyala
//-----
if (rc = 0
and nVMs > 0);
    JavaVM_P = jvm(1);
    rc = DestroyJavaVM (jvm(1));
endif;

if (rc = 0);
    return *ON;
else;
    return *OFF;
endif;
/end-free

P destroyJVM      E

```

Figura 83. Código fuente para destroyJVM

Indicar a Java que ha terminado con un objeto temporal

Si ha creado un objeto utilizando un constructor de Java, o si ha llamado a un método de Java que ha devuelto un objeto, este objeto sólo se podrá destruir en la recogida de basura de Java cuando Java sepa que ya no necesita más el objeto. El objeto se destruye en los métodos nativos, cuando se devuelve el método nativo, pero nunca en los procedimientos RPG no nativos, a menos que se informe explícitamente a Java de que ya no se necesita más el objeto. Para ello, llame al procedimiento empaquetador de RPG freeLocalRef.

```
CALLP freeLocalRef (JNIEnv_P : string);
```

La Figura 84 en la página 182 contiene el código fuente de ejemplo para freeLocalRef.

```

/*-----*/
/* freeLocalRef                                     */
/*-----*/
P freeLocalRef...
P          B          EXPORT
D freeLocalRef...
D          PI
D env          *      VALUE
D localRef      0      CLASS(*JAVA
D                  : 'java.lang.Object')
D                  VALUE

/free
  jniEnv_P = env
  DeleteLocalRef (env : localRef)
/end-free

P freeLocalRef...
P          E

```

Figura 84. Código fuente para freeLocalRef

Nota: Para llamar a este empaquetador, se necesita el puntero de entorno JNI (que se describe más adelante en el apartado “Obtener un puntero de entorno JNI” en la página 184).

Indicar a Java que desea que un objeto sea permanente

Si tiene una referencia a un objeto de Java que se ha pasado como parámetro o se ha creado llamando a un método o constructor de Java, y desea utilizar este objeto después de volver el método nativo, deberá indicar a Java que desea que este objeto sea permanente, o “global”. Para ello, llame al procedimiento de empaquetador de RPG getNewGlobalRef y guarde el resultado en una variable global.

```
EVAL globalString = getNewGlobalRef (JNIENV_P : string);
```

La Figura 85 en la página 183 contiene el código fuente de ejemplo para getNewGlobalRef.


```

/*-----*/
/* getNewGlobalRef */
/*-----*/
P getNewGlobalRef...
P          B          EXPORT
D getNewGlobalRef...
D          PI          0    CLASS(*JAVA
D                               : 'java.lang.Object')
D env          *    VALUE
D localRef     0    CLASS(*JAVA
D                               : 'java.lang.Object')
D                               VALUE

/free
jniEnv_P = env;
return NewGlobalRef (env : localRef);
/end-free
P getNewGlobalRef...
P          E

```

Figura 85. Código fuente para getNewGlobalRef

Nota: Para llamar a este empaquetador, se necesita el puntero de entorno JNI (que se describe más adelante en el apartado “Obtener un puntero de entorno JNI” en la página 184).

Indicar a Java que ha terminado con un objeto permanente

Si ha creado una referencia global, y sabe que ya no necesita más este objeto, deberá indicar a Java que puede destruir el objeto la próxima vez que se realice la recogida de basura en Java. (El objeto sólo se destruirá si no existen otras referencias globales a él, y si no existen otras referencias dentro de Java). Para indicar a Java que ya no necesita más la referencia al objeto, llame al procedimiento de empaquetador de RPG freeGlobalRef .

```
CALLP freeGlobalRef (JNIEnv_P : globalString);
```

La Figura 86 en la página 184 contiene un código fuente de ejemplo para freeGlobalRef.

```

/*-----*/
/* freeGlobalRef                                     */
/*-----*/
P freeGlobalRef...
P          B                                EXPORT
D freeGlobalRef...
D          PI
D  env          *      VALUE
D  globalRef    0      CLASS(*JAVA
D                  : 'java.lang.Object')
D                  VALUE

/free
  jniEnv_P = env;
  DeleteGlobalRef (env : globalRef)
/end-free

P freeGlobalRef...
P          E

```

Figura 86. Código fuente para freeGlobalRef

Nota: Para llamar a este empaquetador, se necesita el puntero de entorno JNI (que se describe más adelante en el apartado “Obtener un puntero de entorno JNI”).

Crear la Java Virtual Machine (JVM)

Si la JVM no se ha creado todavía y el código RPG está listo para llamar al método de Java, RPG creará la JVM utilizando la vía de acceso de clases por omisión más la vía de acceso de clases de la variable de entorno CLASSPATH. No obstante, si el programador desea crear la JVM, puede ver un ejemplo de esta codificación en la última parte de la figura Figura 87 en la página 185.

Obtener un puntero de entorno JNI

Si necesita llamar a funciones JNI, utilice el archivo JNI de /COPY en QSYSINC/QRPGLESRC. La mayoría de funciones JNI se llaman con un puntero de procedimiento. Los punteros de procedimiento forman parte de una estructura de datos que está basada a su vez en un puntero denominado “Puntero de entorno de JNI”. Este puntero se denomina JNIEnv_P en el archivo JNI de /COPY. Para obtener este puntero, llame al procedimiento de empaquetador de JNI getJniEnv.

```
EVAL  JNIEnv_P = getJniEnv();
```

La Figura 87 en la página 185 contiene un código fuente de ejemplo para getJniEnv.

```

/*-----*/
/* getJniEnv */
/*-----*/
P getJniEnv      B      EXPORT
D getJniEnv      PI      *

D rc              S      LIKE(jint)
D rpgRc          S      N
D jvm             S      * DIM(1)
D env            S      *
D bufLen         S      LIKE(jsize) INZ(%elem(jvm))
D nVMs           S      LIKE(jsize)
D initArgs       DS      LIKEDS(JDK1_1InitArgs)
D attachArgs     DS      LIKEDS(JDK1_1AttachArgs)

/free

//-----
// Observe si hay una JVM iniciada
//-----
rc = JNI_GetCreatedJavaVMs (jvm : bufLen : nVMs);

//-----
// Si hay una JVM iniciada, adjúntela para obtener el puntero env
//-----
if (rc = 0
and nVMs > 0);
attachArgs = *ALLX'00';
JavaVM_P = jvm(1);
rc = AttachCurrentThread (jvm(1)
: env : %addr(attachArgs));

//-----
// Si no hay una JVM iniciada, iníciela con la vía de acceso de
// clases por omisión
//-----
else;
initArgs = *ALLX'00';
rc = JNI_GetDefaultJavaVMInitArgs (%addr(initArgs));
if (rc = 0);
rc = JNI_CreateJavaVM (jvm(1) : env : %addr(initArgs));
endif;
endif;

if (rc = 0);
return env;
else;
return *NULL;
endif;
/end-free

P getJniEnv      E

```

Figura 87. Código fuente para getJniEnv

Manejo de excepciones JNI

En ILE RPG, una excepción produce un mensaje de excepción. Los programas no necesitan buscar explícitamente las excepciones; en su lugar, se pueden codificar manejadores de excepciones que controlen la situación cuando se produzca una

excepción. El programador sólo deberá manejar las excepciones JNI cuando realice sus propias llamadas JNI. Cuando una llamada a una JNI produzca una excepción de Java no manejada, no aparecerá ningún mensaje de excepción. En su lugar, el programador de JNI deberá comprobar si se ha producido una excepción después de cada llamada a la función JNI. Para ello, debe llamar a la función de JNIExceptionOccurred, que devuelve un objeto Exception de Java (o el objeto null de Java que tiene un valor de 0 en el JNI). Una vez determinado si se ha producido una excepción, las únicas llamadas JNI que se pueden realizar son ExceptionClear y ExceptionDescribe. Después de llamar a ExceptionClear, ya se pueden realizar llamadas JNI de nuevo. Si realiza una llamada JNI de no excepción antes de llamar a ExceptionClear, la excepción desaparecerá, y no podrá obtener más información. RPG siempre convierte una excepción JNI en una excepción RPG (señala uno de los mensajes RNX030x, dependiendo de la función RPG que se estuviera ejecutando en ese momento).

Consejo

Incluya este tipo de código de manejo de excepciones en las versiones de los procedimientos de empaquetador JNI citadas anteriormente.

Consideraciones adicionales

Errores de tiempo de ejecución más comunes

El compilador no intentará resolver las clases en el tiempo de compilación. Si no se puede encontrar una clase durante la ejecución, aparecerá un error de tiempo de ejecución. En él se indicará que se ha recibido un objeto *UnresolvedLinkException* desde el entorno de Java.

El compilador no comprueba los tipos de parámetros en el tiempo de compilación. Si se produce un conflicto entre el prototipo y el método al que se está llamando, se recibirá un error durante la ejecución.

Consejos de depuración

Un objeto de Java se visualiza como una referencia de objeto en RPG. Esta referencia de objeto es un valor entero, que se comporta como un puntero. Las referencias normales de objeto son valores positivos, asignados en orden creciente a partir de 1. Las referencias globales, que se pueden crear utilizando la función JNI NewGlobalRef, son valores negativos. Estos valores se asignan en orden decreciente a partir del número negativo más pequeño (-2147483647).

Normalmente, estos valores no son visibles en el código RPG. No obstante, esta información puede ser de gran utilidad cuando se depura el código RPG.

Creación de objetos String en RPG

Si necesita utilizar un objeto String para pasar un método de Java, puede crearlo de la siguiente forma:

```
D newString      PR          0      EXTPROC(*JAVA
D                                     : 'java.lang.String'
D                                     : *CONSTRUCTOR)
D   value                65535A    CONST VARYING
D string           S                                like(jstring)
```

```

/free
    string = newString ('abcde');
    ...
/end-free

```

Si desea crear una serie con datos UCS-2 o datos de gráficos, utilice este código:

```

D newStringC      PR          0      EXTPROC(*JAVA
D                                     : 'java.lang.String'
D                                     : *CONSTRUCTOR)
D   value          32767C      CONST VARYING

D string          S              like(jstring)
D graphicData     S              15G
D ucs2Data        S              100C

/free
    string = newStringC (%UCS2(graphicData));
    ...
    string = newStringC (ucs2Data);
/end-free

```

Codificación JNI avanzada

El compilador RPG IV da soporte para llamar a métodos de Java y escribir métodos nativos RPG, ocultando casi toda la codificación JNI al programador de RPG. No obstante, el soporte de RPG no es necesariamente el más eficaz. Por ejemplo, este soporte convierte siempre las matrices entre RPG y Java en las llamadas, y en la entrada y salida de los métodos nativos, pero el programador puede manejar sus propias conversiones de matrices para mejorar el rendimiento.

El soporte RPG sólo proporciona acceso a los métodos de Java. Si desea acceder a los campos de una clase, deberá añadir los métodos "get" y "set" a la clase de Java, o realizar la codificación de JNI (consulte el apartado "Acceso a los campos de las clases de Java" en la página 188).

La Figura 88 es un ejemplo de una llamada de JNI en RPG.

```

/COPY JNI
D objectId      s              like(jobject)
D methodId      s              like(jmethodID)
D string        s              like(jstring)
D parms         s              like(jvalue) dim(2)

/free
    jvalue_P = %addr(parms(1));
    jvalue.i = 10;
    jvalue_P = %addr(parms(2));
    jvalue.l = string;
    CallVoidMethodA (JNIEnv_P : objectId : methodId : parms);
/end-free

```

Figura 88. Llamada de JNI en RPG

Tenga en cuenta que los punteros JNIEnv_P y jvalue_P están definidos en el archivo JNI de /COPY.

Conversión de datos de tipo carácter de Java

En Java, los datos de tipo carácter son ASCII, en lugar de EBCDIC, por lo que deberá comprobar que los nombres de clase, método y campo estén en ASCII para realizar las llamadas a funciones JNI como FindClass. Los datos de tipo carácter

que vienen de Java son ASCII. Para utilizarlos en el programa RPG, probablemente deberá convertirlos a EBCDIC. El compilador RPG maneja estas conversiones, pero si el programador está realizando las llamadas JNI, deberá realizar también las conversiones entre ASCII y EBCDIC.

Acceso a los campos de las clases de Java

RPG sólo da soporte a la llamada a métodos de Java; no da soporte al acceso a los campos de Java. Normalmente, se puede acceder a los campos a través de los métodos "get" y "set", pero también es posible acceder a los campos utilizando llamadas JNI. A continuación se proporciona un ejemplo con las llamadas JNI necesarias para acceder a los campos de una clase u objeto de Java.

Nota: Este es un ejemplo de la utilización de JNI. No es una recomendación para acceder a los campos directamente, en lugar de utilizando los métodos "get" y "set".

```

*-----
* Este ejemplo muestra cómo utilizar JNI para acceder a los campos de
* una clase o un objeto.
*
*
* Este programa crea un objeto Rectangle y accede a las variables de
* anchura y altura directamente utilizando llamadas JNI.
*
* En este caso concreto, se podrían haber utilizado los métodos
* getWidth(), getHeight(), setWidth() y setHeight() para
* acceder a estos campos, en lugar de utilizar las llamadas JNI.
*-----
H THREAD(*SERIALIZE)
/DEFINE JNI_COPY_FIELD_FUNCTIONS
/COPY JNI
/COPY JNIRPG_PR
*-----
* Clases y métodos de JAVA
*-----
D Rectangle      C      'java.awt.Rectangle'
D NewRectangle   PR      0      EXTPROC(*JAVA
: Rectangle
: *CONSTRUCTOR)
D      x      10I 0 VALUE
D      y      10I 0 VALUE
D      width   10I 0 VALUE
D      height  10I 0 VALUE
*-----
* Constantes con representaciones en ASCII de los nombres de Java
*-----
* Una forma de determinar estos valores es utilizar %UCS2 para convertir
* un valor de carácter a UCS-2, y visualizar el resultado en hex
* en el depurador.
*
* El valor de ASCII está en cada segundo byte de los caracteres UCS-2.
*
* Por ejemplo, %UCS2('abc') = X'006100620063'
*
* La representación en ASCII de 'abc' es X'616263'
*-----
D ASCII_I      C      x'49'
D ASCII_x      C      x'78'
D ASCII_y      C      x'79'
D ASCII_width  C      X'7769647468'
D ASCII_height C      X'686569676874'
*-----
* Tenga en cuenta que es 'java/awt/Rectangle', no 'java.awt.Rectangle'
* ya que JNI utiliza la barra como separador.
D ASCII_Rectangle...
D      C      X'6A61766612F6177742F52656-
D      374616E676C65'
*-----
* Manejo de cancelación
*-----
D EnableCanHdlr PR      EXTPROC('CEERTX')
D Handler      *      CONST PROCPTR
D CommArea     *      CONST OPTIONS(*OMIT)
D Feedback     12A     OPTIONS(*OMIT)
D CanHdlr      PR
D CommArea     *      CONST
*-----
* Procedimientos y variables locales
*-----
D rect      S      0      CLASS(*JAVA : Rectangle)
D x      S      10I 0
D y      S      10I 0
D rectClass S      LIKE(jclass)
D fieldId  S      LIKE(jfieldID)
D msg      S      52A
D Cleanup  PR

```

Figura 89. Utilización de JNI para acceder a los campos de clases y objetos de Java (Pieza 1 de 3)

```

*-----
* Habilite el manejar de cancelación para asegurar que se ha realizado la limpieza
*-----
C          CALLP      EnableCanHdlr (%PADDR(CanHdlr)
C                               : *OMIT : *OMIT)
*-----
* Cree un nuevo rectángulo con las coordenadas x,y (5, 15),
* anchura 100 y altura 200.
*-----
C          EVAL      rect = NewRectangle (5 : 15 : 100 : 200)
*-----
* Prepárese para llamar a las funciones JNI para acceder a campos del rectángulo
*-----
C          EVAL      JNIEnv_P = getJNIEnv ()
C          EVAL      rectClass = FindClass (JNIEnv_P
C                                         : ASCII_Rectangle)
*-----
* Llame a las funciones JNI para recuperar la anchura y la altura del rectángulo
*-----
C          eval      fieldId = GetFieldID (JNIEnv_P
C                                         : rectClass
C                                         : ASCII_width
C                                         : ASCII_I)
C          eval      width = GetIntField (JNIEnv_P
C                                         : rect
C                                         : fieldId)
C          eval      fieldId = GetFieldID (JNIEnv_P
C                                         : rectClass
C                                         : ASCII_height
C                                         : ASCII_I)
C          eval      height = GetIntField (JNIEnv_P
C                                         : rect
C                                         : fieldId)
C          eval      msg = 'The rectangle has dimensions ('
C                      + %trim(%editc(width : '1'))
C                      + ', '
C                      + %trim(%editc(height : '1'))
C                      + ')'
C      msg          dsply
*-----
* Ya no necesitamos este rectángulo, por lo que debemos indicar a Java
* que esta disponible para su reclamación.
*-----
C          callp      DeleteLocalRef(JNIEnv_P : rect)
*-----
* Llame al procedimiento de limpieza
*-----
C          CALLP      Cleanup
C          SETON
LR

```

Figura 89. Utilización de JNI para acceder a los campos de clases y objetos de Java (Pieza 2 de 3)


```

*-----
* Limpieza.
* Finalice la JVM
*-----
P Cleanup          B
C                  callp    destroyJVM()
P Cleanup          E
*-----
* Cancele el manejador. Se finaliza la JVM.
*-----
P CanHdlr          B
D CanHdlr          PI
D  CommArea                *  CONST
*-----
* Llamada anormal al procedimiento de limpieza
*-----
C                  callp    Cleanup()
P CanHdlr          E

```

Figura 89. Utilización de JNI para acceder a los campos de clases y objetos de Java (Pieza 3 de 3)

Llamada a los métodos de Java utilizando JNI en lugar de los prototipos *JAVA de RPG

Los primeros tres parámetros son siempre los mismos:

1. el puntero de entorno JNI
2. el objeto (para los métodos de instancia) o la clase (para los métodos estáticos)
3. el método

Los parámetros específicos del método se codifican después de estos tres parámetros. Esto se puede hacer de tres maneras diferentes. Por ejemplo, si el método no devuelve un valor, el tipo de retorno es "void".

CallVoidMethod:

Elija este camino si va a llamar al mismo método varias veces, ya que facilita la llamada al método. Los parámetros se espera que pasen normalmente. Para llamar a esta función JNI, el programador de RPG debe copiar el prototipo CallVoidMethod del archivo JNI de /COPY, y codificar los parámetros adicionales. Por ejemplo,

```

D CallMyMethod    PR                               EXTPROC(*CWIDEN
D                                                         : JNINativeInterface.
D                                                         CallVoidMethod_P)
D env              LIKE(JNIEnv_P) VALUE
D obj              LIKE(jobject) VALUE
D methodID         LIKE(jmethodID) VALUE
D len              LIKE(jint) VALUE
D str              LIKE(jstring) CONST
...
CallMyMethod (JNIEnv_P : objectId : methodId : 10 : string);

```

Figura 90. Código RPG de ejemplo para llamar a CallVoidMethod

CallVoidMethodA:

Elija este camino si no desea crear otro prototipo para llamar al método. Se

espera una matriz de estructuras jvalue, con cada elemento de la matriz manteniendo un parámetro. La figura Figura 88 en la página 187 anterior es un ejemplo de este procedimiento.

CallVoidMethodV:

No utilice este procedimiento en código RPG. Se espera un constructo C que es extremadamente difícil de codificar en RPG.

La función que se debe llamar dependerá del tipo de valor de retorno. Por ejemplo, si el método devuelve un entero, deberá utilizar CallIntMethodA. Para obtener la clase y los parámetros de methodID de estas funciones, utilice FindClass y GetMethodID, o GetStaticMethodID.

Nota: Cuando se llama a JNI directamente, los nombres de clase se deben especificar con una barra (/) en lugar de con un punto (.) como separador. Por ejemplo, utilice 'java/lang/String' en lugar de 'java.lang.String'.

Parte 3. Depuración y manejo de excepciones

En esta sección se describe cómo:

- Depurar una aplicación Integrated Language Environment utilizando el depurador del fuente Integrated Language Environment
- Escribir programas que manejan excepciones
- Obtener un vuelco

Capítulo 12. Depuración de programas

La depuración le permite detectar, diagnosticar y eliminar errores durante la ejecución de un programa. Puede depurar programas ILE y OPM utilizando el depurador del fuente ILE.

Este capítulo describe cómo utilizar el depurador del fuente ILE para:

- Preparar el programa ILE RPG para depurarlo
- Iniciar una sesión de depuración
- Añadir y eliminar programas de una sesión de depuración
- Ver el fuente del programa desde una sesión de depuración
- Establecer y eliminar puntos de interrupción y condiciones de observación
- Seguir los pasos de un programa
- Visualizar y cambiar el valor de los campos
- Visualizar los atributos de los campos
- Igualar un nombre abreviado a un campo, expresión o mandato de depuración

Al depurar y probar los programas, asegúrese de que su lista de bibliotecas está modificada para dirigir los programas a una biblioteca de prueba que contiene datos de prueba para que los datos reales existentes no se vean afectados.

Puede impedir que se modifiquen accidentalmente los archivos de base de datos de las bibliotecas de producción mediante la utilización de uno de los mandatos siguientes:

- Utilice el mandato Arrancar Depuración (STRDBG) y conserve el valor por omisión *NO para el parámetro UPDPROD.
- Utilice el mandato Cambiar Depuración (CHGDBG) y especifique el valor *NO del parámetro UPDPROD.
- Utilice el mandato de depuración SET en la pantalla Visualizar fuente de módulo y especifique UPDPROD NO

Consulte el capítulo acerca de la depuración en la publicación *ILE Concepts* para obtener más información sobre el depurador del fuente ILE (incluidos la autorización necesaria para depurar un programa o un programa de servicio y los efectos de los niveles de optimización).

El depurador del fuente ILE

El depurador del fuente ILE se utiliza para detectar y eliminar errores en los objetos de programas y en los programas de servicio. Mediante la utilización de los mandatos de depuración con cualquier programa ILE que contenga datos de depuración, puede:

- Ver el fuente del programa o cambiar la vista de depuración
- Establecer y eliminar puntos de interrupción y condiciones de observación
- Seguir paso a paso un número especificado de sentencias
- Visualizar o cambiar el valor de los campos, estructuras y matrices
- Igualar un nombre abreviado con un campo, expresión o mandato de depuración

El depurador del fuente ILE

Para poder utilizar el depurador del fuente, debe seleccionar una vista de depuración al crear un objeto de módulo o de programa con los mandatos CRTRPGMOD o CRTBNDRPG. Después de arrancar el depurador, puede establecer puntos de interrupción y llamar al programa.

Cuando se detiene un programa debido a un punto de interrupción o a un mandato de salto, la vista del objeto de módulo correspondiente aparece en la pantalla en el punto en que se haya detenido el programa. En este punto puede realizar otras acciones, como visualizar o cambiar los valores de los campos.

Nota: Si el programa se ha optimizado, puede seguir visualizando campos, pero sus valores pueden no ser fiables. Para garantizar que los campos o las estructuras de datos contengan los valores correctos (los actuales), especifique la palabra clave NOOPT en la especificación de definición adecuada. Para cambiar el nivel de optimización, consulte el apartado “Modificación del nivel de optimización” en la página 91.

Mandatos de depuración

Hay numerosos mandatos de depuración disponibles para utilizarlos con el depurador del fuente ILE. Los mandatos de depuración y sus parámetros se teclean en la línea de mandatos de depuración que se visualiza en la parte inferior de las pantallas Visualizar fuente de módulo y Evaluar Expresión. Estos mandatos se pueden teclear en mayúsculas, en minúsculas o en una combinación de éstas.

Nota: Los mandatos de depuración entrados en la línea de mandatos de depuración no son mandatos CL.

A continuación se listan los mandatos de depuración.

Mandato	Descripción
ATTR	Le permite visualizar los atributos de una variable. Estos atributos son el tamaño y el tipo de la variable tal como están registrados en la tabla de símbolos de depuración.
BREAK	Le permite entrar un punto de interrupción de trabajo condicional o incondicional en una posición del programa que se está probando. Utilice BREAK <i>número-de-línea</i> WHEN <i>expresión</i> para entrar un punto de interrupción de trabajo condicional.
CLEAR	Le permite eliminar los puntos de interrupción condicionales e incondicionales, o eliminar una o todas las condiciones de observación activas.
DISPLAY	Le permite visualizar los nombres y las definiciones asignadas utilizando el mandato EQUATE. También le permite visualizar un módulo fuente distinto al mostrado en ese momento en la pantalla Visualizar fuente de módulo. El objeto de módulo debe existir en el objeto de programa actual.
EQUATE	Le permite asignar una expresión, una variable o un mandato de depuración a un nombre para utilizarlo de forma abreviada.
EVAL	Le permite visualizar o cambiar el valor de una variable o visualizar el valor de las expresiones, registros, estructuras o matrices.

El depurador del fuente ILE

QUAL	Le permite definir el ámbito de las variables que aparecen en los mandatos EVAL o WATCH subsiguientes. Actualmente, no es aplicable a ILE RPG.
SET	Le permite cambiar las opciones de depuración, como por ejemplo la posibilidad de actualizar archivos de producción, especificar si las operaciones de búsqueda deben ser sensibles a mayúsculas/minúsculas o habilitar el soporte de depuración del fuente OPM.
STEP	Le permite ejecutar una o varias sentencias del procedimiento que se está depurando.
TBREAK	Le permite entrar un punto de interrupción condicional o incondicional en la hebra actual en una posición del programa que se está probando.
THREAD	Le permite visualizar la pantalla Trabajar con hebras depuradas o cambiar la hebra actual.
WATCH	Le permite solicitar un punto de interrupción cuando el contenido de una ubicación de almacenamiento especificada cambia su valor actual.
FIND	Busca hacia adelante o hacia atrás en el módulo visualizado actualmente una serie, texto o número de línea especificado.
UP	Mueve la ventana visualizada del fuente hacia el principio de la vista en proporción al espacio especificado.
DOWN	Mueve la ventana visualizada del fuente hacia el final de la vista en proporción al espacio especificado.
LEFT	Mueve la ventana visualizada del fuente hacia la izquierda el número de columnas especificado.
RIGHT	Mueve la ventana visualizada del fuente hacia la derecha el número de columnas especificado.
TOP	Sitúa la vista para que se muestre la primera línea.
BOTTOM	Sitúa la vista para que se muestre la última línea.
NEXT	Sitúa la vista para mostrar el siguiente punto de interrupción del fuente visualizado actualmente.
PREVIOUS	Sitúa la vista para mostrar el punto de interrupción anterior del fuente visualizado actualmente.
HELP	Muestra la información de ayuda en línea para los mandatos disponibles del depurador del fuente.

La ayuda en línea para el depurador del fuente ILE describe los mandatos de depuración, explica las abreviaturas permitidas y proporciona diagramas de sintaxis para cada mandato. También proporciona ejemplos en cada lenguaje ILE de visualización y cambio de variables utilizando el depurador del fuente.

Siga estos pasos para acceder a la información de ayuda en línea para ILE RPG:

1. Entre STRDBG nombre biblioteca/nombre programa, donde *nombre programa* es cualquier programa ILE que tenga datos de depuración en la biblioteca *nombre biblioteca*.
2. Entre DSPMODSRC para mostrar la vista fuente si esta pantalla no aparece a continuación del paso 1.

El depurador del fuente ILE

3. Entre PF1 (Ayuda)
4. Sitúe el cursor en EVAL y pulse Intro para visualizar la ayuda de mandatos EVAL.
5. Sitúe el cursor en Expresiones y pulse Intro para visualizar la ayuda para expresiones.
6. Sitúe el cursor en lenguaje RPG y pulse Intro para visualizar los ejemplos de lenguajes RPG.
7. Desde el panel de ayuda que aparecerá, puede seleccionar diversos temas pertenecientes a RPG, como por ejemplo, visualizar variables, visualizar tablas y visualizar estructuras de datos de aparición múltiple.

Preparación de un programa para su depuración

Un programa o un módulo debe tener datos de depuración disponibles si va a depurarlo. Puesto que los datos de depuración se crean durante la depuración, especifique si un módulo va a contener datos de depuración al crearlo utilizando CRTBNDRPG o CRTRPGMOD. Puede utilizar el parámetro DBGVIEW en cualquiera de estos mandatos para indicar qué tipo de datos (si los hay) se va a crear durante la compilación.

El tipo de datos de depuración que puede asociarse con un módulo recibe el nombre de **vista de depuración**. Puede crear una de las vistas siguientes para cada módulo que desee depurar. Son las siguientes:

- Vista del fuente raíz
- Vista del fuente COPY
- Vista del listado
- Vista de la sentencia

El valor por omisión de CRTBNDRPG y CRTRPGMOD es la creación de una vista de sentencias. Esta vista proporciona el nivel de soporte de depuración más cercano a los releases anteriores.

Si no desea que los datos de depuración se incluyan con el módulo o si desea reducir el tiempo de compilación, especifique DBGVIEW(*NONE) al crear el módulo. Sin embargo, un vuelco con formato no listará los valores de las variables del programa cuando no haya disponibles datos de depuración.

Observe también que los requisitos de almacenamiento para un módulo o un programa variarán ligeramente en función del tipo de datos de depuración incluidos en el módulo o programa. Los valores siguientes del parámetro DBGVIEW se listan en orden ascendente en función de su efecto sobre los requisitos de almacenamiento secundario:

1. *NONE
2. *STMT
3. *SOURCE
4. *COPY
5. *LIST
6. *ALL

Una vez que haya creado un módulo con datos de depuración y lo haya enlazado a un objeto de programa (*PGM), puede empezar a depurar el programa.

Preparación de un programa para su depuración

Nota: Un programa OPM debe compilarse con OPTION(*SRCDDBG) u OPTION(*LSTDBG) para poder depurarlo utilizando el depurador del fuente ILE. Para obtener más información, consulte el apartado “Inicio del depurador del fuente ILE” en la página 202.

En la siguiente tabla se resumen las vistas de depuración:

Tabla 14. Vistas de depuración

Vista de depuración	Datos de depuración	Valor del parámetro DBGVIEW
Ninguna	Sin datos de depuración	*NONE
Vista de sentencia (valor por omisión)	No se visualiza el fuente (se utilizan los números de sentencia de la sección del fuente de la lista del compilador)	*STMT
Vista del fuente raíz	Información de miembro fuente raíz	*SOURCE
Vista del fuente COPY	Información de miembro fuente raíz e información de miembros /COPY	*COPY
Vista del listado	Listado del compilador (dependiente del parámetro OPTION)	*LIST
Todas	Datos de las vistas de fuente raíz, de fuente COPY y de listado	*ALL

Creación de una vista del fuente raíz

Una **vista del fuente raíz** contiene texto del miembro fuente raíz. Esta vista no contiene ningún miembro /COPY. Además, no está disponible si el miembro fuente raíz es un archivo DDM.

Se crea una vista del fuente raíz para depurar un módulo utilizando las opciones *SOURCE, *COPY o *ALL en el parámetro DBGVIEW para los mandatos CRTRPGMOD o CRTBNDRPG cuando se crea un módulo.

El compilador crea la vista del fuente raíz mientras se compila el objeto de módulo (*MODULE). La vista del fuente raíz se crea utilizando las referencias a ubicaciones de texto en el miembro fuente raíz en lugar de copiar el texto del miembro en el objeto de módulo. Por esta razón, no debe modificar, cambiar el nombre ni mover los miembros fuente raíz entre la creación del módulo de estos miembros y la depuración del módulo creado a partir de estos miembros. Si lo hace, las vistas de estos miembros fuente pueden no ser utilizables.

Por ejemplo, para crear una vista del fuente raíz para el programa DEBUGEX cuando se utiliza CRTBNDRPG, escriba:

```
CRTBNDRPG PGM(MIBIBL/DEBUGEX) SRCFILE(MIBIBL/QRPGLESRC)
          TEXT('programa ILE RPG/400 DEBUGEX')
          DBGVIEW(*SOURCE)
```

Para crear una vista del fuente raíz para un módulo llamado DBGEX cuando se utiliza CRTRPGMOD, escriba:

```
CRTRPGMOD MODULE(MIBIBL/DBGEX) SRCFILE(MIBIBL/QRPGLESRC)
          TEXT('Módulo de entrada del programa DEBUGEX')
          DBGVIEW(*SOURCE)
```

Preparación de un programa para su depuración

Al especificar DBGVIEW(*SOURCE) con cualquier mandato de creación, se crea una vista del fuente raíz para el módulo de depuración DBGEX. Por omisión, se genera un listado del compilador con miembros /COPY y DDS ampliadas, así como otra información adicional.

Creación de una vista del fuente COPY

Una **vista del fuente COPY** contiene texto del miembro fuente raíz, así como el texto de todos los miembros /COPY ampliados en el texto del fuente. Cuando se utiliza la vista COPY, puede depurar el miembro fuente raíz del programa utilizando la vista del fuente raíz y los miembros /COPY del programa mediante la vista del fuente COPY.

La vista del miembro fuente raíz generada por DBGVIEW(*COPY) es la misma que la generada por DBGVIEW(*SOURCE). Al igual que con la vista del fuente raíz, la vista del fuente COPY no está disponible si el archivo fuente es un archivo DDM.

Puede crear una vista del fuente COPY para depurar un módulo utilizando la opción *COPY u *ALL en el parámetro DBGVIEW.

El compilador crea la vista COPY mientras se compila el objeto de módulo (*MODULE). La vista COPY se crea utilizando las referencias a ubicaciones de texto en los miembros fuente (en el miembro fuente raíz y en los miembros /COPY) en lugar de copiar el texto de los miembros en la vista. Por esta razón, no debe modificar, red denominar ni mover los miembros fuente durante el tiempo transcurrido entre la creación del objeto de módulo y la depuración del módulo creado a partir de estos miembros. Si lo hace, las vistas de estos miembros fuente pueden no ser utilizables.

Por ejemplo, para crear una vista del fuente de un programa llamado PRUEBA1 que contiene miembros /COPY, escriba:

```
CRTBNDPRPG PGM(MIBIBL/PRUEBA1) SRCFILE(MIBIBL/QRPGLESRC)
          TEXT('ILE programa RPG/400 PRUEBA1')
          DBGVIEW(*COPY)
```

Al especificar DBGVIEW(*COPY) con cualquier mandato de creación, se crea una vista del fuente raíz con miembros /COPY para el módulo de depuración PRUEBA1. Por omisión, se genera un listado del compilador. Dicho listado incluirá también miembros /COPY, ya que OPTION(*SHOWCPY) es un valor por omisión.

Creación de una vista de listado

Una **vista del listado** contiene texto parecido al del listado del compilador generado por el compilador ILE RPG. La información que figura en la vista del listado depende de si se ha especificado OPTION(*SHOWCPY), OPTION(*EXPDDS) y OPTION(*SRCSTMT) en algún mandato de creación. OPTION(*SHOWCPY) incluye los miembros /COPY en el listado y OPTION(*EXPDDS) incluye los archivos descritos externamente. OPTION(*SRCSTMT) permite depurar el objeto de programa utilizando los números de sentencia en lugar de los números de línea del listado del compilador.

Nota: Parte de la información que está disponible en el listado del compilador no aparecerá en la vista de listado. Por ejemplo, si especifica un sangrado en el listado del compilador (mediante el parámetro INDENT), el sangrado no aparecerá en la vista del listado. Si especifica OPTION(*SHOWSKP) en el listado del compilador, las sentencias que se han saltado no aparecerán en la vista de listado.

Preparación de un programa para su depuración

Se crea una vista del listado para depurar un módulo utilizando las opciones *LIST o *ALL en el parámetro DBGVIEW para los mandatos CRTRPGMOD o CRTBNDRPG cuando se crea un módulo.

El compilador crea la vista del listado mientras se genera el objeto de módulo (*MODULE). La vista del listado se crea copiando el texto de los miembros fuente adecuados en el objeto de módulo. Una vez creada, la vista del listado no depende de los miembros fuente en los que se basa.

Por ejemplo, para crear una vista del listado para un programa llamado PRUEBA1 que contiene DDS ampliadas, escriba:

```
CRTBNDRPG PGM(MIBIBL/PRUEBA1) SRCFILE(MIBIBL/QRPGLESRC)
          SRCMBR(PRUEBA1) OUTPUT(*PRINT)
          TEXT('programa ILE RPG/400 PRUEBA1')
          OPTION(*EXPDDS) DBGVIEW(*LIST)
```

Al especificar DBGVIEW(*LIST) en el parámetro DBGVIEW y *EXPDDS en el parámetro OPTION en cualquier mandato de creación, se crea una vista del listado con DDS ampliadas para depurar el fuente de PRUEBA1. Tenga en cuenta que OUTPUT(*PRINT) y OPTION(*EXPDDS) son valores por omisión.

Creación de una vista de sentencias

Una **vista de sentencias** permite depurar el objeto de módulo utilizando números de sentencia y mandatos de depuración. Puesto que el fuente no se visualizará, debe utilizar números de sentencia que aparezcan en la sección del fuente del listado del compilador. En otras palabras, para utilizar de forma eficaz esta vista, necesitará un listado del compilador. Además, los números de sentencia generados para la depuración dependen de si se ha especificado *SRCSTMT o *NOSRCSTMT para el parámetro OPTION. *NOSRCSTMT significa que los número de sentencia se asignan secuencialmente y se visualizan como números de línea en la columna situada más a la izquierda de la sección del fuente del listado del compilador. *SRCSTMT permite solicitar que el compilador utilice ID del fuente y números de secuencia SEU al generar números de sentencia para la depuración. Los números de sentencia se muestran en la columna situada más a la derecha de la sección del fuente del listado del compilador.

Se crea una vista de sentencias para depurar un módulo utilizando la opción *STMT en el parámetro DBGVIEW para los mandatos CRTRPGMOD o CRTBNDRPG cuando se crea un módulo.

Utilice esta vista cuando:

- Tenga restricciones de almacenamiento, pero no desee volver a compilar el módulo o el programa si tiene que depurarlo.
- Está enviando objetos compilados a otros usuarios y desea poder diagnosticar los problemas existentes en el código empleando el depurador, pero no quiere que dichos usuarios vean el código real.

Por ejemplo, para crear una vista de sentencias para el programa DEBUGEX utilizando CRTBNDRPG, escriba:

```
CRTBNDRPG PGM(MIBIBL/DEBUGEX) SRCFILE(MIBIBL/QRPGLESRC)
          TEXT('programa ILE RPG/400 DEBUGEX')
```

Para crear una vista de sentencias para un módulo utilizando CRTRPGMOD, escriba:

Preparación de un programa para su depuración

```
CRTRPGMOD MODULE(MIBIBL/DBGEX) SRCFILE(MIBIBL/QRPGLESRC)
      TEXT('Módulo de entrada del programa DEBUGEX')
```

Por omisión, se generan un listado del compilador y una vista de sentencias. Mediante la utilización de un listado del compilador para obtener los números de sentencia, puede depurar el programa utilizando los mandatos de depuración.

Si los valores por omisión de cualquier mandato de creación han cambiado, debe especificar explícitamente `DBGVIEW(*STMT)` y `OUTPUT(*PRINT)`.

Inicio del depurador del fuente ILE

Una vez que haya creado la vista de depuración (de sentencias, del fuente, COPY o del listado), puede empezar a depurar la aplicación. Para iniciar el depurador del fuente ILE, utilice el mandato Iniciar depuración (STRDBG). Una vez iniciado el depurador, permanecerá activo hasta que entre el mandato Finalizar depuración (ENDDBG).

Inicialmente, puede añadir un máximo de 20 objetos de programa a una sesión de depuración mediante el parámetro Programa (PGM) en el mandato STRDBG. Puede ser cualquier combinación de programas OPM o ILE. (En función de cómo se hayan compilado los programas OPM y también de los valores de entorno de depuración, es posible que pueda depurarlos utilizando el depurador del fuente ILE). Además, inicialmente puede añadir un máximo de 20 objetos de programa de servicio a una sesión de depuración mediante el parámetro Programas de servicio (SRVPGM) en el mandato STRDBG. Las normas para depurar un programa de servicio son las mismas que para depurar un programa:

- El programa o programa de servicio debe tener datos de depuración.
- Debe tener la autorización *CHANGE sobre un objeto de programa o de programa de servicio para incluirlo en una sesión de depuración.

Nota: Si depura un programa utilizando la vista COPY o la vista del fuente raíz, el código fuente debe estar en el mismo sistema que el objeto de programa que se está depurando. Además, el código fuente debe estar en una biblioteca/archivo(miembro) con el mismo nombre que tenía cuando se compiló.

Para un programa ILE, se muestra el módulo de entrada si tiene datos de depuración; de lo contrario, se muestra el primer módulo enlazado al programa ILE con datos de depuración.

Para un programa OPM, se muestra el primer programa especificado en el mandato STRDBG si tiene datos de depuración y el parámetro OPMSRC es *YES. Es decir, si un programa OPM está en una sesión de depuración, puede depurarlo utilizando el depurador del fuente ILE si se cumplen las siguientes condiciones:

1. El programa OPM se ha compilado con `OPTION(*LSTDBG)` u `OPTION(*SRCDBG)`. (Están soportados 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 ILE se ha establecido de forma que acepte programas OPM. Puede hacerlo especificando `OPMSRC(*YES)` en el mandato STRDBG. (El valor por omisión del sistema es `OPMSRC(*NO)`.)

Si no se cumplen estas dos condiciones, debe depurar el programa OPM con el depurador del sistema OPM.

Inicio del depurador del fuente ILE

Si se especifica un programa OPM sin *LSTDBG o *SRCDBG y se especifica un programa de servicio, se mostrará el programa de servicio si tiene datos de depuración. Si no hay datos de depuración, la pantalla DSPMODSRC estará vacía. Si se especifican un programa ILE y un programa de servicio, se mostrará el programa ILE.

Ejemplo de STRDBG

Para iniciar una sesión de depuración del programa de depuración de ejemplo DEBUGEX y un programa OPM RPGPGM al que se ha llamado, escriba:

```
STRDBG PGM(MIBIBL/DEBUGEX MIBIBL/RPGPGM) OPMSRC(*YES)
```

Aparece la pantalla Visualizar fuente de módulo, tal como se muestra en la Figura 91. DEBUGEX consta de dos módulos, un módulo RPG llamado DBGEX y un módulo C llamado cproc. Consulte el apartado “Ejemplo de fuente para ejemplos de depuración” en la página 245 para ver el fuente de DBGEX, cproc y RPGPGM.

Si el módulo de entrada tiene una vista de fuente raíz, COPY o vista del listado, la pantalla mostrará el fuente del módulo de entrada del primer programa. En este caso, el programa se ha creado utilizando DBGVIEW(*ALL), por lo que se muestra el fuente del módulo principal, DBGEX.

Visualizar fuente de módulo

Programa: DEBUGEX

Biblioteca: MIBIBL

Módulo: DBGEX

1

*=====

2

* DEBUGEX - Programa diseñado para ilustrar el uso del depurador del

3

* fuente ILE con el fuente ILE RPG. Proporciona un ejemplo

4

* de diferentes tipos de datos y estructuras de datos.

5

*

6

* También puede usarse para producir vuelcos formateados ejemplo.

7

*=====

8

9

*-----

10

* La palabra clave DEBUG habilita el recurso de vuelco formateado.

11

*-----

12

H DEBUG

13

14

*-----

15

* Define campos autónomos para diferentes tipos de datos ILE RPG.

Más...

Depurar . . .

F3=Fin programa

F6=Añadir/Borrar punto interrup.

F10=Saltar

F11=Ver variable

F12=Reanudar

F17=Observar variable

F18=Trab. con observación

F24=Más teclas

Figura 91. Pantalla Visualizar fuente de módulo correspondiente al programa DEBUGEX

Nota: Inicialmente, pueden añadirse un máximo de 20 objetos de programa de servicio a la sesión de depuración mediante el parámetro Programa de servicio (SRVPGM) en el mandato STRDBG. También puede añadir programas de servicio ILE a una sesión de depuración mediante la opción 1 (Añadir) de la pantalla Trabajar con lista de módulos (F14) o dejando que sea el depurador del fuente el que los añada como parte de un mandato de depuración STEP INTO.

Capítulo 12. Depuración de programas 203

Establecimiento de opciones de depuración

Después de iniciar una sesión de depuración, puede establecer o cambiar las siguientes opciones de depuración:

- Si los archivos de base de datos pueden actualizarse durante la depuración del programa. (Esta opción corresponde al parámetro UPDPDPROD del mandato STRDBG).
- Si las búsquedas de texto mediante FIND son sensibles a mayúsculas y minúsculas.
- Si los programas OPM deben depurarse mediante el depurador del fuente ILE. (Esta opción corresponde al parámetro OPMSRC).

El cambio de las opciones de depuración mediante el mandato de depuración SET afecta al valor del parámetro correspondiente, si lo hay, especificado en el mandato STRDBG. También puede utilizar el mandato Cambiar depuración (CHGDBG) para establecer las opciones de depuración. Sin embargo, la opción OPMSRC no puede cambiarse mediante el mandato CHGDBG. OPMSRC sólo puede cambiarse mediante el mandato de depuración SET.

Suponga que se encuentra en una sesión de depuración trabajando con un programa ILE y decide que también debe depurar un programa OPM que tiene datos de depuración disponibles. Para habilitar al depurador del fuente ILE para que acepte programas OPM, siga estos pasos:

1. Después de entrar STRDBG, si la pantalla actual *no* es la pantalla Visualizar fuente de módulo, escriba:
DSPMODSRC

Aparece la pantalla Visualizar fuente de módulo.
2. Escriba
SET
3. Aparece la pantalla Establecer opciones de depuración. En esta pantalla, escriba Y (Sí) en el campo *Soporte de depuración de fuente OPM* y pulse Intro para volver a la pantalla Visualizar fuente de módulo.

Ahora, puede añadir el programa OPM, utilizando la pantalla Trabajar con módulo o procesando una sentencia de llamada a dicho programa.

Adición/eliminación de programas de una sesión de depuración

Puede añadir y eliminar programas en una sesión de depuración después de haberla iniciado. Debe tener autorización *CHANGE sobre un programa para poderlo añadir o eliminar de una sesión de depuración.

Para los programas ILE, utilice la opción 1 (Añadir programa) de la pantalla Trabajar con lista de módulos del mandato DSPMODSRC. Para eliminar un programa o programa de servicio ILE, utilice la opción 4 (Eliminar programa) de la misma pantalla. Cuando se elimina un programa o un programa de servicio ILE, también se eliminan todos los puntos de interrupción del mismo. No hay límite alguno en cuanto al número de programas o programas de servicio ILE que se pueden eliminar o que puede haber en una sesión de depuración al mismo tiempo.

Para programas OPM, tiene dos opciones, dependiendo del valor especificado para OPMSRC. Si ha especificado OPMSRC(*YES), utilizando STRDBG, el mandato de depuración SET o CHGDBG, puede añadir o eliminar un programa OPM

Adición/eliminación de programas de una sesión de depuración

utilizando la pantalla Trabajar con módulo. (Tenga en cuenta que no habrá un nombre de módulo listado para un programa OPM). No hay límite alguno en cuanto al número de programas OPM que se pueden incluir en una sesión de depuración cuando se especifica OPMSRC(*YES).

Si ha especificado OPMSRC(*NO), debe utilizar los mandatos Añadir programa (ADDPGM) o Eliminar programa (RMVPGM). Sólo puede haber 20 programas OPM en una sesión de depuración al mismo tiempo cuando se especifica OPMSRC(*NO).

Nota: No puede depurar un programa OPM con datos de depuración de sesiones de depuración ILE y OPM. Si un programa OPM ya está en una sesión de depuración OPM, debe eliminarlo primero de dicha sesión antes de añadirlo a la sesión de depuración ILE o entrar en él desde una sentencia de llamada. De igual modo, si desea depurar desde una sesión de depuración OPM, debe eliminarlo primero de una sesión de depuración ILE.

Ejemplo de adición de un programa de servicio a una sesión de depuración

En este ejemplo, se añade el programa CVTTOHEX a la sesión de depuración iniciada previamente. (Consulte el apartado “Programa de servicio de ejemplo” en la página 96 para obtener una descripción del programa de servicio).

1. Si la pantalla actual *no* es la pantalla Visualizar fuente de módulo, escriba:
DSPMODSRC

Aparece la pantalla Visualizar fuente de módulo.

2. Pulse F14 (Trabajar con lista de módulos) para que aparezca la pantalla Trabajar con lista de módulos, tal como aparece en la Figura 92.
3. Para añadir el programa de servicio CVTTOHEX, en la primera línea de la pantalla, escriba: 1 (Añadir programa), CVTTOHEX en el campo *Programa/módulo* y MIBIB en el campo *Biblioteca*. Cambie el tipo de programa por omisión de *PGM a *SRVPGM y pulse Intro.
4. Pulse F12 (Cancelar) para volver a la pantalla Visualizar fuente de módulo.

Trabajar con lista de módulos				Sistema:	AS400S1
Teclee elecciones, pulse Intro.					
1=Añadir programa 4=Eliminar programa 5=Visualizar fuente de módulo					
8=Trabajar con puntos de interrupción de módulo					
Opc	Programa/módulo	Biblioteca	Tipo		
1	cvttohex	mibib	*SRVPGM		
	RPGPGM	MIBIB	*PGM		
	DEBUGEX	MIBIB	*PGM		
	DBGEX		*MODULE	Seleccionado	
	CPROC		*MODULE		
					Final
Mandato					
==>					
F3=Salir F4=Solicitud F5=Renovar F9=Recuperar F12=Cancelar					

Figura 92. Adición de un programa de servicio ILE a una sesión de depuración

Adición/eliminación de programas de una sesión de depuración

Ejemplo de eliminación de programas ILE de una sesión de depuración

En este ejemplo eliminará el programa CVTHEXPGM y el programa de servicio CVTTOHEX ILE de una sesión de depuración.

1. Si la pantalla actual *no* es la pantalla Visualizar fuente de módulo, escriba:
DSPMODSRC

Aparece la pantalla Visualizar fuente de módulo.

2. Pulse F14 (Trabajar con lista de módulos) para que aparezca la pantalla Trabajar con lista de módulos, tal como aparece en la Figura 93.
3. En esta pantalla, escriba 4 (Eliminar programa) en la línea situada junto a CVTHEXPGM y CVTTOHEX, y a continuación pulse Intro.
4. Pulse F12 (Cancelar) para volver a la pantalla Visualizar fuente de módulo.

Trabajar con lista de módulos

Sistema: AS400S1

Teclee elecciones, pulse Intro.

1=Añadir programa 4=Eliminar programa 5=Visualizar fuente de módulo

8=Trabajar con puntos de interrupción de módulo

Opc	Programa/módulo	Biblioteca	Tipo	
	*LIBL	*PGM		
4	CVTHEXPGM	MIBIB	*PGM	
	CVTHEXPG		*MODULE	
4	CVTTOHEX	MIBIB	*SRVPGM	
	CVTTOHEX		*MODULE	
	RPGPGM	MIBIB	*PGM	
	DEBUGEX	MIBIB	*PGM	
	DBGEX		*MODULE	Seleccionado
	CPROC		*MODULE	

Final

Mandato

===>

F3=Salir F4=Solicitud F5=Renovar F9=Recuperar F12=Cancelar

Figura 93. Eliminación de programas ILE de una sesión de depuración

Visualización del fuente del programa

La pantalla Visualizar fuente de módulo muestra el fuente de un objeto de programa ILE, de objeto de módulo en objeto de módulo. El fuente de un objeto de módulo ILE se puede mostrar si el objeto de módulo se ha compilado con una de las opciones de vista de depuración siguientes:

- DBGVIEW(*SOURCE)
- DBGVIEW(*COPY)
- DBGVIEW(*LIST)
- DBGVIEW(*ALL)

El fuente de un programa OPM puede mostrarse si se cumplen las siguientes condiciones:

1. El programa OPM se ha compilado con OPTION(*LSTDBG) u OPTION(*SRCDBG). (Sólo los programas RPG y COBOL pueden compilarse con *LSTDBG).

Visualización del fuente del programa

2. El entorno de depuración ILE se ha establecido de forma que acepte programas OPM, es decir, el valor de OPMSRC debe ser *YES. (El valor por omisión del sistema es OPMSRC(*NO).)

Existen dos métodos para cambiar lo que se muestra en la pantalla Visualizar fuente de módulo:

- Ir a un módulo distinto
- Cambiar la vista de un módulo

Cuando se cambia una vista, el depurador ILE del fuente efectúa una correlación con las posiciones equivalentes de la vista que desea. Cuando se cambia el módulo, la sentencia ejecutable de la vista visualizada se almacena en la memoria y se muestra cuando el módulo se visualiza de nuevo. Los números de línea que tienen puntos de interrupción se resaltan. Cuando un punto de interrupción, un paso o un mensaje hacen que el programa se detenga y se visualice la pantalla, se resalta la sentencia en la que se ha producido el punto de interrupción.

Visualización de un módulo diferente

Para cambiar el objeto de módulo que se muestra en la pantalla Visualizar fuente de módulo, utilice la opción 5 (Visualizar fuente de módulo) de la pantalla Trabajar con lista de módulos. Puede acceder a la pantalla Trabajar con lista de módulos desde la pantalla Visualizar fuente de módulo pulsando F14 (Trabajar con lista de módulos).

Si utiliza esta opción con un objeto de programa ILE, se muestra el módulo de entrada con una vista del fuente raíz, COPY o del listado (si existe). De lo contrario, se muestra el primer objeto de módulo enlazado al objeto de programa con datos de depuración. Si utiliza esta opción con un objeto de programa OPM, se muestra la vista del fuente o del listado (si está disponible).

Un método alternativo para ver otro objeto de módulo es utilizar el mandato de depuración DISPLAY. En la línea de mandatos de depuración, escriba:

```
DISPLAY MODULE nombre-módulo
```

Se muestra el *nombre-módulo* del objeto de módulo. Dicho objeto debe existir en un objeto de programa que se haya añadido a la sesión de depuración.

Por ejemplo, para pasar del módulo DBGEX de la Figura 91 en la página 203 al módulo cproc mediante la opción Visualizar fuente de módulo, siga estos pasos:

1. Para trabajar con módulos, escriba DSPMODSRC y pulse Intro. Se visualiza la pantalla Visualizar fuente de módulo.
2. Pulse F14 (Trabajar con lista de módulos) para mostrar la pantalla Trabajar con lista de módulos. La Figura 94 en la página 208 muestra una pantalla de ejemplo.
3. Para seleccionar cproc, escriba 5 (Visualizar fuente de módulo) junto a él y pulse Intro. Puesto que hay una vista de fuente raíz disponible, se mostrará como en la Figura 95 en la página 208. Si no hay disponible ningún fuente raíz, se muestra el primer objeto de módulo enlazado al objeto de programa con datos de depuración.

Visualización del fuente del programa

Trabajar con lista de módulos

Sistema: AS400S1

Teclee elecciones, pulse Intro.

1=Añadir programa 4=Eliminar programa 5=Visualizar fuente de módulo

8=Trabajar con puntos de interrupción de módulo

Opc	Programa/módulo	Biblioteca	Tipo
	*LIBL	*PGM	
	RPGPGM	MIBIB	*PGM
	DEBUGEX	MIBIB	*PGM
	DBGEX		*MODULE
5	CPROC		*MODULE
			Seleccionado

Final

Mandato

===>

F3=Salir

F4=Solicitud

F5=Renovar

F9=Recuperar

F12=Cancelar

Figura 94. Cambiar a un módulo distinto

Visualizar fuente de módulo

Programa: DEBUGEX

Biblioteca: MIBIB

Módulo: CPROC

1 #include <stdlib.h>

2 #include <string.h>

3 #include <stdio.h>

4 extern char EXPORTFLD[6];

5

6 char *c_proc(unsigned int size, char *inzval)

7 {

8 char *ptr;

9 ptr = malloc(size);

10 memset(ptr, *inzval, size);

11 printf("import string: %6s.\n",EXPORTFLD);

12 return(ptr);

13 }

Final

Depurar . . .

F3=Fin programa

F6=Añadir/Borrar punto interrupción

F10=Saltar

F11=Ver variable

F12=Reanudar

F17=Obser. variable

F18=Trab. con observ.

F24=Más teclas

Figura 95. Vista del fuente del procedimiento ILE C cproc

Cambio de la vista de un módulo

Se pueden visualizar varias vistas diferentes de un módulo ILE RPG, en función de los valores que especifique al crear el módulo. Son las siguientes:

- Vista del fuente raíz
- Vista del fuente COPY
- Vista del listado

Puede cambiar la vista del objeto de módulo que se muestra en la pantalla Visualizar fuente de módulo mediante la pantalla Seleccionar vista. Se puede acceder a la pantalla Seleccionar Vista desde la pantalla Visualizar fuente de módulo pulsando F15 (Seleccionar Vista). La pantalla Seleccionar vista se muestra en la Figura 96 en la página 209. La vista actual se lista al principio de la ventana y las demás vistas que están disponibles se muestran a continuación. Cada objeto de módulo de un objeto de programa puede tener un conjunto distinto de vistas disponibles, según las opciones de depuración utilizadas para crearlo.

Visualización del fuente del programa

Por ejemplo, para cambiar la vista del módulo de fuente raíz a listado, siga estos pasos:

1. Escriba DSPMODSRC y pulse Intro. Se visualiza la pantalla Visualizar fuente de módulo.
2. Pulse F15 (Seleccionar vista). La pantalla Seleccionar vista se muestra en la Figura 96.

Visualizar fuente de módulo

Seleccionar vista

Vista actual . . . : Vista COPY ILE RPG

Teclee opción, pulse Intro.

1=Seleccionar

Opc

Vista

1 Vista lista ILE RPG

Vista fuente ILE RPG

Vista Copy ILE RPG

F12=Cancelar

Final

Más...

Depurar . . .

F3=Fin programa

F6=Añadir/Borrar punto interrupción

F10=Saltar

F11=Ver variable

F12=Reanudar

F17=Observar variable

F18=Trab. con observación

F24=Más teclas

Figura 96. Cambio de una vista de un módulo

La vista actual se lista al principio de la ventana y las demás vistas que están disponibles se muestran a continuación. Cada módulo de un programa puede tener un conjunto distinto de vistas disponibles, según las opciones de depuración utilizadas para crearlo.

Nota: Si se crea un módulo con DBGVIEW(*ALL), la ventana Seleccionar Vista mostrará las tres vistas disponibles: fuente raíz, COPY y listado. Si el módulo no tiene miembros /COPY, la vista COPY será idéntica a la vista del fuente raíz.

3. Escriba un 1 junto a la vista del listado y pulse Intro. Aparece la pantalla Visualizar fuente de módulo, mostrando el módulo con una vista de listado.

Establecimiento y eliminación de puntos de interrupción

Puede utilizar puntos de interrupción para detener un objeto programa en un punto determinado mientras se está ejecutando. Un **punto de interrupción incondicional** detiene el objeto de programa en una sentencia determinada. Un **punto de interrupción condicional** detiene el objeto de programa cuando se cumple una condición determinada en una sentencia específica.

Existen dos tipos de puntos de interrupción: de trabajo y de hebra. Cada **hebra** de una aplicación con hebras puede tener su propio punto de interrupción de hebra en la misma posición al mismo tiempo. Tanto los puntos de interrupción de trabajo como los de hebra pueden ser incondicionales o condicionales. En general, existe un conjunto de mandatos de depuración y teclas de función para puntos de interrupción de trabajo y otro para puntos de interrupción de hebra. En el resto de

Capítulo 12. Depuración de programas 209

Establecimiento y eliminación de puntos de interrupción

esta sección sobre puntos de interrupción, la expresión punto de interrupción hace referencia tanto a trabajo como a hebra, a menos que se indique lo contrario específicamente.

Nota: Los puntos de interrupción se generan automáticamente para especificaciones de entrada y salida si se especifica la opción por omisión `OPTION(*DEBUGIO)`. Si no desea generar puntos de interrupción, especifique `OPTION(*NODEBUGIO)`.

Los puntos de interrupción se establecen antes de ejecutar el programa. Cuando el objeto de programa se detiene, aparece la pantalla Visualizar fuente de módulo. El objeto de módulo apropiado se muestra con el fuente colocado en la línea en la que se ha producido el punto de interrupción. Esta línea está resaltada. En este punto, puede evaluar campos, establecer más puntos de interrupción y ejecutar cualquier mandato de depuración.

Debe conocer las características siguientes de los puntos de interrupción antes de utilizarlos:

- Cuando se establece un punto de interrupción en una sentencia, el punto de interrupción se produce *antes* de procesarse la sentencia.
- Cuando se llega a una sentencia con un punto de interrupción condicional, la expresión condicional asociada con el punto de interrupción se evalúa *antes* de procesarse la sentencia. Si la expresión es verdadera, el punto de interrupción entra en vigor y el programa se detiene en esa línea.
- Si la línea en la que desea establecer un punto de interrupción no es una sentencia ejecutable, el punto de interrupción se establecerá en la siguiente sentencia ejecutable.
- Si se salta un punto de interrupción, dicho punto no se procesa.
- Las funciones de punto de interrupción se especifican mediante mandatos de depuración. Estas funciones son:
 - Añadir puntos de interrupción a los objetos de programa
 - Eliminar puntos de interrupción de los objetos de programa
 - Visualizar información de puntos de interrupción
 - Reanudar la ejecución de un objeto de programa después de llegar a un punto de interrupción
 - Puede tener un punto de interrupción de trabajo o de hebra en una posición especificada al mismo tiempo, pero no ambos.

Si cambia la vista del módulo después de establecer puntos de interrupción, el depurador del fuente correlaciona los números de línea de los puntos de interrupción con la nueva vista.

Si está depurando un módulo o un programa creado con una vista de sentencias, puede establecer o eliminar puntos de interrupción utilizando los números de sentencia obtenidos del listado del compilador. Para obtener más información acerca de la utilización de números de sentencia, consulte el apartado “Establecimiento y eliminación de puntos de interrupción de trabajo utilizando números de sentencia” en la página 217.

Establecimiento y eliminación de puntos de interrupción de trabajo incondicionales

Puede establecer o eliminar un punto de interrupción de trabajo incondicional utilizando:

Establecimiento y eliminación de puntos de interrupción

- F6 (Añadir/borrar punto de interrupción) o F13 (Trabajar con puntos de interrupción de módulo) en la pantalla Visualizar fuente de módulo
- El mandato de depuración BREAK para establecer un punto de interrupción de trabajo
- El mandato de depuración CLEAR para eliminar un punto de interrupción de trabajo
- La pantalla Trabajar con puntos de interrupción de módulo.

La manera más sencilla de establecer y eliminar un punto de interrupción de trabajo incondicional es utilizar F6 (Añadir/borrar punto de interrupción). La tecla de función actúa como un conmutador, por lo que eliminará un punto de interrupción de la línea en la que está el cursor si en dicha línea se había establecido un punto de interrupción.

Para eliminar un punto de interrupción incondicional de trabajo utilizando F13 (Trabajar con puntos de interrupción de módulo), pulse F13 (Trabajar con puntos de interrupción de módulo) en la pantalla Visualizar fuente de módulo. Aparecerá una lista de opciones que le permitirá establecer o eliminar puntos de interrupción. Si selecciona 4 (Borrar), se elimina un punto de interrupción de trabajo de la línea.

Un método alternativo de establecer y eliminar puntos de interrupción de trabajo incondicionales es utilizar los mandatos de depuración BREAK y CLEAR. Para establecer un punto de interrupción de trabajo incondicional utilizando el mandato de depuración BREAK, escriba:

BREAK número-línea

en la línea de mandatos de depuración. La variable *número-línea* es el número de línea de la vista visualizada actualmente del objeto de módulo en que desea establecer un punto de interrupción.

Para eliminar un punto de interrupción de trabajo incondicional utilizando el mandato de depuración CLEAR, escriba:

CLEAR número-línea

en la línea de mandatos de depuración. La variable *número-línea* es el número de línea de la vista visualizada actualmente del objeto de módulo de la que desea eliminar un punto de interrupción. Cuando se borra un punto de interrupción, también se borra para todas las hebras.

Ejemplo de establecimiento de un punto de interrupción de trabajo incondicional

En este ejemplo, establecerá un punto de interrupción de trabajo incondicional utilizando F6 (Añadir/borrar punto de interrupción). El punto de interrupción se establecerá en la primera especificación de cálculo ejecutable para poder visualizar los diversos campos y estructuras de datos.

1. Para trabajar con un módulo, escriba DSPMODSRC y pulse Intro. Se visualiza la pantalla Visualizar fuente de módulo.
2. Si desea establecer el punto de interrupción de trabajo en el módulo que se muestra, siga con el paso 3 en la página 212. Si desea establecer un punto de interrupción de trabajo en otro módulo, escriba:

DISPLAY MODULE nombre-módulo

Establecimiento y eliminación de puntos de interrupción

en la línea de mandatos de depuración, siendo *nombre-módulo* el nombre del módulo que desea visualizar.

3. Para establecer un punto de interrupción incondicional en la primera especificación de cálculo, coloque el cursor en la línea 88.
4. Pulse F6 (Añadir/borrar punto de interrupción). Si no existe un punto de interrupción en la línea 88, se establece un punto de interrupción incondicional en dicha línea, como se muestra en la Figura 97. Si en esa línea hay un punto de interrupción, se elimina.

Nota: Dado que queremos el punto de interrupción en la *primera* especificación de cálculo, podríamos haber colocado el cursor en una línea cualquiera antes del comienzo de las especificaciones de cálculo, y el punto de interrupción se hubiera colocado igualmente en la línea 88, ya que es la primera sentencia ejecutable.

Visualizar fuente de módulo

Programa: DEBUGEX Biblioteca: MIBIBL Módulo: DBGEX

84 *-----

85 * Mover 'a's a la estructura de datos DS2. Después del movimiento,

86 * la primer aparición de DS2 contiene 'a's de 10 caracteres.

87 *-----

88 C MOVE *ALL'a' DS2

89

90 *-----

91 * Cambiar la aparición de DS2 a 2 y mover 'b's a DS2, haciendo que

92 * los 10 primeros bytes sean 'a's y los segundos 10 bytes sean 'b's

93 *-----

94 C 2 OCCUR DS2

95 C MOVE *ALL'b' DS2

96

97 *-----

98 * Fld1a es un campo de recubrimiento de Fld1. Dado que Fld1 se inicializa

Más...

Depurar . . .

F3=Fin programa F6=Añadir/Borrar punto interrupción F10=Saltar F11=Ver variable

F12=Reanudar F17=Observar variable F18=Trab. con observación F24=Más teclas

Punto de interrupción añadido a línea 88.

Figura 97. Establecimiento de un punto de interrupción de trabajo incondicional

5. Después de establecer el punto de interrupción, pulse F3 (Salir) para salir de la pantalla Visualizar fuente de módulo. El punto de interrupción no se elimina.
6. Llame al programa. Cuando se llega a un punto de interrupción, el programa se detiene y se visualiza de nuevo la pantalla Visualizar fuente de módulo, con la línea que contiene el punto de interrupción resaltada. En este punto, puede seguir los pasos del programa o reanudar el proceso.

Establecimiento y eliminación de puntos de interrupción incondicionales

Puede establecer o eliminar un punto de interrupción de hebra incondicional utilizando:

- La pantalla Trabajar con puntos de interrupción de módulo
- El mandato de depuración TBREAK para establecer un punto de interrupción de hebra en la hebra actual
- El mandato de depuración CLEAR para eliminar un punto de interrupción de hebra

Establecimiento y eliminación de puntos de interrupción

Para establecer un punto de interrupción de hebra incondicional utilizando la pantalla Trabajar con puntos de interrupción de módulo:

- Escriba 1 (Añadir) en el campo *Opc*.
- En el campo *Hebra*, escriba el identificador de hebra.
- Rellene los campos restantes como si se tratara de un punto de interrupción de trabajo incondicional.
- Pulse Intro.

Nota: El campo *Hebra* se visualiza cuando la opción DEBUG del mandato SPAWN es mayor o igual a uno.

El mandato de depuración TBREAK tiene la misma sintaxis que el mandato de depuración BREAK. Mientras que el mandato de depuración BREAK establece un punto de interrupción de trabajo en la misma posición en todas las hebras, el mandato de depuración TBREAK establece un punto de interrupción de hebra en una sola hebra — la hebra actual.

La **hebra actual** es la hebra que se está depurando en ese momento. Los mandatos de depuración se emiten para esta hebra. Cuando se produce una detención de la depuración, como por ejemplo en un punto de interrupción, la hebra actual se establece en la hebra donde se ha producido la detención de la depuración. Pueden utilizarse el mandato de depuración THREAD y la pantalla Trabajar con hebras depuradas para cambiar la hebra actual.

Para eliminar un punto de interrupción de hebra incondicional, utilice el mandato de depuración CLEAR. Cuando se borra un punto de interrupción de hebra, se borra de la hebra actual.

Establecimiento y eliminación de puntos de interrupción de trabajo condicionales

Puede establecer o eliminar un punto de interrupción de trabajo condicional utilizando:

- La pantalla Trabajar con puntos de interrupción de módulo
- El mandato de depuración BREAK para establecer un punto de interrupción de trabajo
- El mandato de depuración CLEAR para eliminar un punto de interrupción

Nota: Los operadores relacionales soportados para puntos de interrupción condicionales son <, >, =, <=, >= y <> (no igual).

Una manera de establecer o eliminar puntos de interrupción de trabajo condicionales es mediante la pantalla Trabajar con puntos de interrupción de módulo. Puede acceder a dicha pantalla desde la pantalla Visualizar fuente de módulo pulsando F13 (Trabajar con puntos de interrupción de módulo). La pantalla proporciona una lista de opciones que permiten añadir o eliminar puntos de interrupción de trabajo condicionales e incondicionales. En la Figura 98 en la página 215 se muestra un ejemplo de la pantalla.

Para convertir el punto de interrupción de trabajo en condicional, especifique una expresión condicional en el campo *Condición*. Si la línea en la que desea establecer un punto de interrupción de trabajo no es una sentencia ejecutable, el punto de interrupción se establecerá en la siguiente sentencia ejecutable.

Establecimiento y eliminación de puntos de interrupción

Si se muestra una columna de hebra, antes de pulsar Intro, escriba *JOB en el campo *Hebra*.

Una vez que haya terminado de especificar todos los puntos de interrupción de trabajo, llame al programa. Puede utilizar F21 (Línea de mandatos) en la pantalla Visualizar fuente de módulo para llamar al objeto de programa desde una línea de mandatos o puede llamar al programa después de salir de la pantalla.

Cuando se llega a una sentencia con un punto de interrupción de trabajo condicional, la expresión condicional asociada con el punto de interrupción de trabajo se evalúa antes de ejecutarse la sentencia. Si el resultado es falso, el objeto de programa sigue ejecutándose. Si el resultado es verdadero, el objeto de programa se detiene y aparece la pantalla Visualizar fuente de módulo. En este punto, puede evaluar campos, establecer más puntos de interrupción y ejecutar cualquier mandato de depuración.

Un método alternativo de establecer y eliminar puntos de interrupción condicionales es utilizando los mandatos de depuración BREAK y CLEAR.

Para establecer un punto de interrupción condicional utilizando el mandato de depuración BREAK, escriba:

BREAK número-línea WHEN expresión

en la línea de mandatos de depuración. La variable *número-línea* es el número de línea de la vista visualizada actualmente del objeto de módulo en la que desea establecer un punto de interrupción y *expresión* es la expresión condicional que se evalúa cuando se encuentra el punto de interrupción. Los operadores relacionales soportados para los puntos de interrupción condicionales figuran al principio de este apartado.

En las expresiones de punto de interrupción condicional no numérico, la expresión más corta se rellena con blancos implícitamente antes de efectuar la comparación. Esta acción implícita tiene lugar antes de la conversión de NLSS (Secuencia de Ordenación de Idiomas Nacionales). Consulte el apartado "Secuencia de Ordenación de Idiomas Nacionales (NLSS)" en la página 216 para obtener más información acerca de NLSS.

Para eliminar un punto de interrupción condicional utilizando el mandato de depuración CLEAR, escriba:

CLEAR número-línea

en la línea de mandatos de depuración. La variable *número-línea* es el número de línea de la vista visualizada actualmente del objeto de módulo de la que desea eliminar un punto de interrupción.

Ejemplo de establecimiento de un punto de interrupción de trabajo condicional utilizando F13

En este ejemplo se establece un punto de interrupción de trabajo condicional utilizando F13 (Trabajar con puntos de interrupción de módulo).

1. Para establecer un punto de interrupción de trabajo condicional, pulse F13 (Trabajar con puntos de interrupción de módulo). Se visualiza la pantalla Trabajar con puntos de interrupción de módulo.
2. En esta pantalla, escriba un 1 (Añadir) en la primera línea de la lista para añadir un punto de interrupción condicional.

Establecimiento y eliminación de puntos de interrupción

- Para establecer un punto de interrupción condicional en la línea 127 cuando *IN02='1', escriba 127 en el campo *Línea* y *IN02='1' en el campo *Condición*.
- Si se muestra una columna de hebra, antes de pulsar Intro, escriba *JOB en el campo Hebra.

La Figura 98 muestra la pantalla Trabajar con puntos de interrupción de módulo después de añadir el punto de interrupción condicional.

Trabajar con puntos de interrupción de módulo		
		Sistema: TORASD80
Programa . . . :	DEBUGEX	Biblioteca . . :
Módulo . . . :	DBGEX	Tipo :
Teclee opciones, pulse Intro.		
1=Añadir 4=Borrar		
Opc	Línea	Condición
	127	*in02='1'
	88	
	102	
		Final
Mandato		
==>		
F3=Salir F4=Solicitud F5=Renovar F9=Recuperar F12=Cancelar		
Punto de interrupción añadido a línea 127.		

Figura 98. Establecimiento de un punto de interrupción de trabajo condicional

Se ha establecido un punto de interrupción de trabajo en la línea 127. La expresión se evalúa antes de ejecutar la sentencia. Si el resultado es verdadero (en el ejemplo, si *IN02='1'), el programa se detiene y aparece la pantalla Visualizar fuente de módulo. Si el resultado es falso, el programa sigue ejecutándose.

Un punto de interrupción existente siempre se sustituye por un punto de interrupción nuevo entrado en la misma posición.

- Una vez establecido el punto de interrupción, pulse F12 (Cancelar) para salir de la pantalla Trabajar con Puntos de Interrupción de Módulo. Pulse F3 (Finalizar programa) para salir del depurador ILE del fuente. El punto de interrupción no se elimina.
- Llame al programa. Cuando se llega a un punto de interrupción, el programa se detiene y se visualiza de nuevo la pantalla Visualizar fuente de módulo. En este punto, puede seguir los pasos del programa o reanudar el proceso.

Ejemplo de establecimiento de un punto de interrupción de trabajo condicional utilizando el mandato BREAK

En este ejemplo, queremos detener el programa cuando el campo de fecha BigDate tenga un determinado valor. Para especificar el punto de interrupción de trabajo condicional utilizando el mandato BREAK:

- En la pantalla Visualizar fuente de módulo, entre:

```
break 128 when BigDate='1994-09-30'
```

Se ha establecido un punto de interrupción de trabajo condicional en la línea 128.

- Después de establecer el punto de interrupción, pulse F3 (Finalizar programa) para salir del depurador ILE del fuente. El punto de interrupción no se elimina.
- Llame al programa. Cuando se llega a un punto de interrupción, el programa se detiene y se visualiza de nuevo la pantalla Visualizar fuente de módulo.

Establecimiento y eliminación de puntos de interrupción

```
Visualizar fuente de módulo
Programa: DEBUGEX      Biblioteca: MIBIBL      Módulo: DBGEX
122
123      *-----
124      * Después de la siguiente operación SETON, *IN02 = '1'.
125      *-----
126      C                      SETON
127      C                      IF      *IN02
128      C                      MOVE    '1994-09-30'  BigDate
129      C                      ENDIF
130
131      *-----
132      * Colocar un valor nuevo en la segunda celda de Arry.
133      *-----
134      C                      MOVE    4          Arry
135
136      *-----
                                           Más...
Depurar . . .  break 128 when BigDate='1994-09-30'
F3=Fin programa   F6=Añadir/Borrar punto interrupción  F10=Saltar  F11=Ver variable
F12=Reanudar      F17=Observar variable  F18=Trab. con observación  F24=Más teclas
```

Figura 99. Establecimiento de un punto de interrupción condicional utilizando el mandato **BREAK**

Secuencia de Ordenación de Idiomas Nacionales (NLSS)

Las expresiones de punto de interrupción condicional no numéricas se dividen en los dos tipos siguientes:

- Car-8: cada carácter contiene 8 bits
Corresponde a los tipos de datos RPG de carácter, fecha, hora e indicación de la hora.
- Car-16: cada carácter contiene 16 bits (DBCS)
Corresponde al tipo de datos RPG de gráficos.

NLSS sólo se aplica a las expresiones de punto de interrupción condicional no numéricas del tipo Car-8. Consulte la Tabla 15 en la página 217 para conocer las posibles combinaciones de expresiones de punto de interrupción condicional no numéricas.

La tabla de secuencia de ordenación utilizada por el depurador del fuente para las expresiones del tipo Car-8 es la especificada en el parámetro SRTSEQ de los mandatos CRTRPGMOD o CRTBNDRPG.

Si la tabla de secuencia de ordenación resuelta es *HEX, no se utiliza ninguna tabla de secuencia de ordenación. Por lo tanto, el depurador del fuente utiliza los valores hexadecimales de los caracteres para determinar la secuencia de ordenación. De lo contrario, se utiliza la tabla de secuencia de ordenación especificada para asignar pesos a cada byte antes de efectuar la comparación. Los bytes que hay entre los caracteres de desplazamiento a teclado estándar y de desplazamiento a teclado ideográfico, y los incluidos en éstos, **no** son pesos asignados. Esto difiere de la manera en que ILE RPG maneja las comparaciones; todos los caracteres, incluidos los de desplazamiento a teclado estándar e ideográfico, son pesos asignados.

Establecimiento y eliminación de puntos de interrupción

Notas:

1. La secuencia alternativa especificada por ALTSEQ (*SRC) en la especificación de control no está disponible para el depurador del fuente ILE. En su lugar, el depurador del fuente utiliza la tabla de secuencia de ordenación *HEX.
2. El nombre de la tabla de secuencia de ordenación se salva durante la compilación. En la depuración, el depurador del fuente utiliza el nombre salvado en la compilación para acceder a la tabla de secuencia de ordenación. Si la tabla de secuencia de ordenación especificada durante la compilación se resuelve como una tabla distinta de *HEX o *JOB RUN, es importante que la tabla de secuencia de ordenación *no* se altere después de iniciar la depuración. Si no se puede acceder a la tabla porque está dañada o se ha suprimido, el depurador del fuente utiliza la tabla de secuencia de ordenación *HEX.

Tabla 15. Expresiones de punto de interrupción condicional no numéricas

Tipo	Posible
Car-8	<ul style="list-style-type: none">• Campo de caracteres comparado a campo de caracteres• Campo de caracteres comparado a literal de caracteres ¹• Campo de caracteres comparado a literal hexadecimal ²• Literal de caracteres ¹ comparado a campo de caracteres• Literal de caracteres ¹ comparado a literal de caracteres¹• Literal de caracteres ¹ comparado a literal hexadecimal ²• Literal hexadecimal ² comparado a campo de caracteres ¹• Literal hexadecimal ² comparado a literal de caracteres ¹• Literal hexadecimal ² comparado a literal hexadecimal ²
Car-16	<ul style="list-style-type: none">• Campo de gráficos comparado a campo de gráficos• Campo de gráficos comparado a literal de gráficos ³• Campo de gráficos comparado a literal hexadecimal ²• Literal de gráficos ³ comparado a campo de gráficos• Literal de gráficos ³ comparado a literal de gráficos³• Literal de gráficos ³ comparado a literal hexadecimal ²• Literal hexadecimal ² comparado a campo de gráficos• Literal hexadecimal ² comparado a literal de gráficos ³
Notas: <ol style="list-style-type: none">1. El literal de caracteres tiene el formato 'abc'.2. El literal hexadecimal tiene el formato X'dígitos hex'.3. El literal de gráficos tiene el formato G'oK1K2i'. El desplazamiento a teclado ideográfico se representa como una o y el desplazamiento a teclado estándar se representa con una i.	

Establecimiento y eliminación de puntos de interrupción de trabajo utilizando números de sentencia

Puede establecer y eliminar puntos de interrupción de trabajo condicionales o incondicionales utilizando los números de sentencia que hay en el listado del compilador del módulo en cuestión. Es necesario si desea depurar un módulo que se ha creado con DBGVIEW(*STMT).

Para establecer un punto de interrupción de trabajo incondicional utilizando el mandato de depuración BREAK, escriba:

BREAK nombre-procedimiento/número-sentencia

Establecimiento y eliminación de puntos de interrupción

en la línea de mandatos de depuración. La variable *nombre-procedimiento* es el nombre del procedimiento en el que se establece el punto de interrupción. Puesto que ILE RPG permite más de un procedimiento por módulo, el *nombre-procedimiento* puede ser el nombre del procedimiento principal o uno de los subprocedimientos de un módulo. La variable *número-sentencia* es el número de sentencia del listado del compilador en la que desea establecer un punto de interrupción.

Nota: El número de sentencia del listado del fuente está indicado como Número de línea si se ha especificado `OPTION(*NOSRCSTMT)`, y como Número de sentencia si se ha especificado `OPTION(*SRCSTMT)`. Por ejemplo, la Figura 100 muestra una sección de ejemplo de un listado con `OPTION(*NOSRCSTMT)`. La Figura 101 muestra la misma sección con `OPTION(*SRCSTMT)`.

Línea	<----- Especificaciones Fuente ----->	Comentarios	Id	Núm
Número1....+....2....+----- 26 - 35 ----->....4....+....5....+....6....+....7....+....8....+....9....+....10	Src	Sec	
1 C	MOVE	'123'	BI_FLD1	000100
2 C	SETON		LR----	000200
***** FIN DE FUENTE *****				

Figura 100. Sección de ejemplo del listado con `OPTION(*NOSRCSTMT)`

Sec	<----- Especificaciones fuente ----->	Comentarios	Número
Número1....+....2....+----- 26 - 35 ----->....4....+....5....+....6....+....7....+....8....+....9....+....10	Sentencia	
000100 C	MOVE	'123'	BI_FLD1
000200 C	SETON		LR----
***** FIN DE FUENTE *****			

Figura 101. Sección de ejemplo del listado del compilador con `OPTION(*SRCSTMT)`

En este ejemplo, se utiliza una Vista de sentencia para establecer un punto de interrupción para el procedimiento TEST. Para establecer un punto de interrupción para el módulo con el listado `*NOSRCSTMT`, escriba:

```
BREAK TEST/2
```

Para establecer un punto de interrupción para el módulo con el listado `*SRCSTMT`, escriba:

```
BREAK TEST/200
```

En ambos casos, el punto de interrupción se establece en la línea
SETON LR----'.

Visualizar fuente de módulo

Programa: TEST Biblioteca: MIBIB Módulo: TEST
(Fuente no disponible).

Depurar . . . break TEST/2 Final

F3=Fin programa F6=Añadir/Borrar punto interrupción F10=Saltar F11=Ver variable
F12=Reanudar F17=Observar variable F18=trabajar con observación F24=Más teclas

Punto de interrupción añadido a sentencia 2 de procedimiento TEST.

Figura 102. Establecimiento de un punto de interrupción utilizando una Vista de sentencia

Establecimiento y eliminación de puntos de interrupción

Para todas las demás vistas de depuración, pueden utilizarse los números de sentencia además de los *números de línea* del programa en el depurador. Por ejemplo, para establecer un punto de interrupción al principio del subprocedimiento FmtCust en la Vista de listado que figura más abajo, escriba:

```
BREAK 34
```

O bien

```
BREAK FmtCust/2600
```

En ambos casos, el punto de interrupción se establece en la línea 'P FmtCust B'.

Visualizar fuente de módulo

Programa:	MIPGM	Biblioteca:	MIBIB	Módulo:	MIPGM
33	002500	*	Iniciar-procedimiento		
34	002600	P	FmtCust	B	
35	002700	D	FmtCust	PI	25A
36	002800	*	Interfaz-procedimiento (la misma que el prototipo)		
37	002900	D	FirstName		10A
38	003000	D	LastName		15A
39	003100	D	ValidRec		N
40	003200	*	Cálculos		
41	003300	C		IF	ValidRec = '0'
42	003400	C		RETURN	%TRIMR(FirstName) + ' ' + Last
43	003500	C		ENDIF	
44	003600	C		RETURN	'Last Customer'
45	003700	*	Finalizar-procedimiento		
46	003800	P		E	
47			*SALIR DE PROCEDIMIENTO PRINCIPAL		

Depurar . . . **BREAK fmtcust/2600**

Más...

F3=Fin programa F6=Añadir/Borrar punto interrupción F10=Saltar F11=Ver variable
F12=Reanudar F17=Observar variable F18=trabajar con observación F24=Más teclas
Punto de interrupción añadido a línea 34.

Figura 103. Establecimiento de un punto de interrupción utilizando números de sentencia y una vista de listado con `OPTION(*SRCSTMT)`

Para establecer un punto de interrupción de trabajo condicional utilizando el mandato de depuración `BREAK`, escriba:

```
BREAK nombre-procedimiento/número-sentencia WHEN expresión
```

en la línea de mandatos de depuración. Las variables *nombre-procedimiento* y *número-sentencia* son las mismas que las de los puntos de interrupción incondicionales. La variable *expresión* es la expresión condicional que se evalúa cuando se encuentra el punto de interrupción.

Para eliminar un punto de interrupción condicional o incondicional utilizando el mandato de depuración `CLEAR`, escriba:

```
CLEAR nombre-procedimiento/número-sentencia
```

en la línea de mandatos de depuración.

Establecimiento y eliminación de puntos de interrupción

Establecimiento y eliminación de puntos de interrupción de hebra condicionales

Puede establecer o eliminar un punto de interrupción de hebra condicional utilizando:

- La pantalla Trabajar con puntos de interrupción de módulo
- El mandato de depuración TBREAK para establecer un punto de interrupción de hebra condicional en la hebra actual
- El mandato de depuración CLEAR para eliminar un punto de interrupción de hebra condicional.

Utilización de la pantalla Trabajar con puntos de interrupción de módulo

Para establecer un punto de interrupción de hebra condicional utilizando la pantalla Trabajar con puntos de interrupción de módulo:

1. Escriba 1 (Añadir) en el campo *Opc*.
2. En el campo *Hebra*, escriba el identificador de hebra.
3. Rellene los campos restantes como si se tratara de un punto de interrupción de trabajo condicional.
4. Pulse Intro.

Para eliminar un punto de interrupción de hebra condicional utilizando la pantalla Trabajar con puntos de interrupción de módulo:

1. Escriba 4 (Borrar) en el campo *Opc* situado junto al punto de interrupción que desee eliminar.
2. Pulse Intro.

Utilización de los mandatos de depuración TBREAK o CLEAR

Para el mandato de depuración TBREAK, utilizará la misma sintaxis que para el mandato de depuración BREAK. La diferencia entre estos mandatos es que, mientras que el mandato de depuración BREAK establece un punto de interrupción de trabajo condicional en la misma posición en todas las hebras, el mandato de depuración TBREAK establece un punto de interrupción de hebra condicional en la hebra actual.

Para eliminar un punto de interrupción de hebra condicional, utilice el mandato de depuración CLEAR. Cuando se borra un punto de interrupción de hebra condicional, se borra sólo de la hebra actual.

Eliminación de todos los puntos de interrupción de trabajo y hebra

Puede eliminar todos los puntos de interrupción de trabajo y hebra , condicionales e incondicionales, de un objeto de programa que tenga un objeto módulo visualizado en la pantalla Visualizar fuente de módulo mediante el mandato de depuración CLEAR PGM. Para utilizar el mandato de depuración, escriba:

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** se utiliza para supervisar si el valor actual de una expresión o de una variable cambia durante la ejecución del programa. El establecimiento de condiciones de observación es similar al establecimiento de puntos de interrupción, con una diferencia importante:

- Las condiciones de observación detienen el programa en cuanto cambia el valor actual de una expresión o variable observada.
- Los puntos de interrupción de trabajo condicionales detienen el programa sólo si una variable cambia al valor especificado en la condición.

El depurador observa una expresión o una variable por medio del contenido de una **dirección de almacenamiento**, calculada en el momento de establecer la condición de observación. Cuando el contenido de la dirección de almacenamiento cambia con respecto al valor que tenía cuando se estableció la condición de observación o cuando se produjo la última condición de observación, el programa se detiene.

Nota: Después de registrar una condición de observación, el nuevo contenido de la ubicación de almacenamiento observada se salva como el nuevo valor actual de la expresión o variable correspondiente. La siguiente condición de observación se registrará si cambia subsiguientemente el nuevo contenido de la ubicación de almacenamiento observada.

Características de las observaciones

Antes de trabajar con ellas, debe conocer las siguientes características de las observaciones:

- Las observaciones se supervisan en todo el sistema, con un número máximo de 256 observaciones activas simultáneamente. Este número incluye las observaciones establecidas por el sistema.

Dependiendo de la utilización global del sistema, puede que el número de condiciones de observación que puede establecer en un momento determinado sea limitado. Si intenta establecer una condición de observación cuando se ha sobrepasado el número máximo de observaciones activas en el sistema, recibirá un mensaje de error y la condición de observación no se establecerá.

Nota: Si una expresión o una variable cruzan un límite de página, se utilizan dos observaciones internamente para supervisar las ubicaciones de almacenamiento. Por tanto, el número máximo de expresiones o variables que pueden observarse simultáneamente en todo el sistema va de 128 a 256.

- Las condiciones de observación sólo pueden establecerse cuando se detiene un programa bajo la depuración y la expresión o variable que debe observarse está en el ámbito. Si este no es el caso, se emite un mensaje de error cuando se solicita una observación, indicando que la entrada de pila de llamadas correspondiente no existe.
- Una vez establecida la condición de observación, la dirección de una ubicación de almacenamiento observada no cambia. Por tanto, si una observación se ha establecido en una ubicación temporal, puede producir notificaciones espurias de condición de observación.

Un ejemplo de este caso es el almacenamiento automático de un subprocedimiento ILE RPG, que puede reutilizarse después de que finalice el subprocedimiento.

Establecimiento y eliminación de condiciones de observación

Una condición de observación puede registrarse aunque la variable observada ya no esté en el ámbito. No debe presuponer que una variable está en el ámbito sólo porque se haya informado de una condición de observación.

- De ninguna manera deben solaparse dos ubicaciones de observación de un mismo trabajo. Dos ubicaciones de observación de trabajos diferentes no deben iniciarse en la misma dirección de almacenamiento; de lo contrario, se permite el recubrimiento. Si se violan estas restricciones, se emite un mensaje de error.

Nota: Los cambios efectuados en una ubicación de almacenamiento observada se pasan por alto si los ha realizado un trabajo que no es el que ha establecido la condición de observación.

- Una vez que el mandato se ha ejecutado satisfactoriamente, la aplicación se detiene si un programa de la sesión cambia el contenido de la ubicación de almacenamiento observada, y se muestra la pantalla **Visualizar fuente de módulo**.

Si el programa tiene datos de depuración, y está disponible una vista de texto del fuente, ésta se mostrará. Se resalta la línea del fuente de la sentencia que iba a ejecutarse cuando se detectó el cambio en el contenido de la ubicación de almacenamiento. Un mensaje indica qué condición de observación se ha satisfecho.

Si el programa no puede depurarse, el área de texto de la pantalla estará en blanco.

- Los programas elegibles se añaden automáticamente a la sesión de depuración si provocan la condición de observación-detención.
- Si varias condiciones de observación se cumplen en la misma sentencia del programa, sólo se informará de la primera.
- También puede establecer condiciones de observación cuando utiliza trabajos de servicio para depuración, es decir, cuando depura un trabajo desde otro trabajo.

Establecimiento de condiciones de observación

Para poder establecer una condición de observación, el programa debe estar detenido bajo depuración y la expresión o variable que desea observar debe estar en el ámbito:

- Para observar una variable global, debe asegurarse de que el programa 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 procedimiento en el que está definida la variable antes de establecer la condición de observación.

Puede establecer una condición de observación utilizando:

- F17 (Observar variable) para establecer una condición de observación para una variable en la que está situado el cursor.
- El mandato de depuración WATCH con o sin sus parámetros.

Utilización del mandato WATCH

Si utiliza el mandato WATCH, debe entrarlo como un solo 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, escriba:

WATCH

en la línea de mandatos de depuración, sin ningún parámetro.

Establecimiento y eliminación de condiciones de observación

```

Trabajar con observación
Sistema:  DEBUGGER

Teclee opciones, pulse Intro.
  4=Borrar 5=Visualizar
Opc  Núm  Variable  Dirección  Longitud
-    1    SALARY    080090506F027004  4
                                           Final

Mandato
====>
F3=Salir  F4=Solicitud  F5=Renovar  F9=Recuperar  F12=Cancelar

```

Figura 104. Ejemplo de la pantalla Trabajar con observación

La pantalla **Trabajar con observación** muestra todas las observaciones actualmente activas en la sesión de depuración. Puede borrar y visualizar las observaciones desde esta pantalla. Si selecciona la opción 5 Visualizar, la ventana **Visualizar observación**, que se muestra a continuación, visualiza información acerca de la observación activa actualmente.

```

Trabajar con observación
.....
: Visualizar observación : DEBUGGER
:
: NÚM observación .: 1 :
: Dirección .....: 080090506F027004 :
: Longitud .....: 4 :
: NÚM coincidencias: 0 :
:
: Ámbito al establecer observación:
: Programa/Biblioteca/Tipo: PAYROLL ABC *PGM :
:
: Módulo...: PAYROLL :
: Procedim : PAYROLL :
: Variable.: SALARY :
: F12=Cancelar :
:
.....
Final
```

Mandato
===>

F3=Salir F4=Solicitud F5=Renovar F9=Recuperar F12=Cancelar

Figura 105. Ejemplo de la ventana Visualizar observación

- Para especificar una variable o expresión que debe observarse, escriba:
WATCH expresión

en la línea de mandatos de depuración.

Este mandato solicita que se establezca un punto de interrupción si el valor de expresión cambia con respecto a su valor actual.

Nota: La expresión se utiliza para determinar la dirección de la ubicación de almacenamiento que debe observarse, y debe resolverse en una ubicación a la que pueda asignarse, por ejemplo:

```
%SUBSTR(X 1 5)
```

Establecimiento y eliminación de condiciones de observación

El ámbito de las variables de expresión de una observación se define mediante el mandato QUAL emitido más recientemente.

- Para establecer una condición de observación y especificar una longitud de observación, escriba:

WATCH expresión : longitud-observación

en una línea de mandatos de depuración.

Cada observación permite supervisar y comparar un máximo de 128 bytes de almacenamiento contiguo. Si se sobrepasa la longitud máxima de 128 bytes, la condición de observación no se establecerá y el depurador emitirá un mensaje de error.

Por omisión, la longitud del tipo de expresión es también la longitud de la operación observación-comparación. El **parámetro longitud de observación** altera temporalmente este valor por omisión. Determina el número de bytes de una expresión que deben compararse para determinar si se ha producido un cambio en el valor.

Por ejemplo, si se especifica un entero de 4 bytes como variable, sin el parámetro de longitud de observación, la longitud de comparación es de 4 bytes. Sin embargo, si se especifica el parámetro de longitud de observación, altera temporalmente la longitud de la expresión al determinar la longitud de observación.

Visualización de observaciones activas

Para visualizar una lista de las observaciones activas en todo el sistema y mostrar qué trabajo las ha establecido, escriba:

DSPDBGWCH

en una línea de mandatos de depuración. Este mandato abre la pantalla **Visualizar observaciones de depuración**, que se muestra a continuación.

Visualizar observaciones de depuración					
-----Trabajo-----			NÚM	LONGITUD	Sistema: DEBUGGER DIRECCIÓN
MYJOBNAME1	MYUSERPRF1	123456	1	5	080090506F027004
JOB4567890	PRF4567890	222222	1	8	09849403845A2C32
JOB4567890	PRF4567890	222222	2	2	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 106. Ejemplo de pantalla Visualizar observaciones de depuración

Nota: Esta pantalla no muestra las condiciones de observación establecidas por el sistema.

Eliminación de condiciones de observación

Las observaciones pueden eliminarse de las siguientes formas:

- El mandato CLEAR utilizado con la palabra clave WATCH finaliza selectivamente una o todas las observaciones. Por ejemplo, para borrar la observación identificada por número-observación, escriba:

Establecimiento y eliminación de condiciones de observación

CLEAR WATCH número-observación

El número de observación puede obtenerse desde la pantalla **Trabajar con observaciones**.

Para borrar todas las observaciones de la sesión, escriba:

CLEAR WATCH ALL

en una línea de mandatos de depuración.

Nota: Dado que el mandato CLEAR PGM elimina todos los puntos de interrupción en el programa que contiene el módulo que se visualiza, no tiene efecto alguno sobre las observaciones. Debe utilizar explícitamente la palabra clave WATCH con el mandato CLEAR para eliminar condiciones de observación.

- El mandato CL Finalizar depuración (ENDDBG) elimina las observaciones establecidas en el trabajo local o en un trabajo de servicio.

Nota: Se llamará automáticamente al mandato ENDDBG en situaciones anormales para asegurar que se eliminan todas las observaciones afectadas.

- La carga del programa inicial (IPL) del sistema iSeries elimina todas las condiciones de observación en todo el sistema.

Ejemplo de establecimiento de una condición de observación

En este ejemplo, observará una variable SALARY en el programa MYLIB/PAYROLL. Para establecer la condición de observación, escriba:

WATCH SALARY

en una línea de depuración, aceptando el valor por omisión para la longitud de observación.

Si el valor de la variable SALARY cambia subsiguientemente, la aplicación se detiene y aparece la pantalla **Visualizar fuente de módulo**, como muestra la Figura 107 en la página 226.

Ejemplo de establecimiento de una condición de observación

```

                                Visualizar fuente de módulo
Programa:  PAYROL      Biblioteca:  MYLIB      Módulo:  PAYROLL
52 C      eval      cnt = 1
53 C      dow      (cnt < EMPMAX)
54 C      eval      Pay_exmpt(cnt) = eflag(cnt)
55 C      eval      cnt = cnt + 1
56 C      enddo
57 C
58 C      eval      index = 1
59 C      dow      index <= cnt
60 C      if      Pay_exmpt(index) = 1
61 C      eval      SALARY = 40 * Pay_wage(index)
62 C      eval      numexmpt = numexmpt + 1
63 C      else
64 C      eval      SALARY = Pay_hours(index)*Pay_wage(index)
65 C      endif
66 C      eval      index = index + 1
67 C      enddo

                                Más...

Depurar . . .

F3=Fin programa  F6=Añadir/Borrar punto interrupción  F10=Saltar  F11=Ver variable
F12=Reanudar     F17=Observar variable  F18=Trab. con observación  F24=Más teclas
Número de observación 1 en línea 65, variable: SALARY

```

Figura 107. Ejemplo de mensaje indicando que WATCH se ha establecido satisfactoriamente

- El número de línea de la sentencia donde se ha detectado el cambio en la variable de observación está resaltado. Se trata generalmente de la primera línea ejecutable *a continuación* de la sentencia que ha cambiado la variable.
- Un mensaje indica que la condición de observación se ha satisfecho.

Nota: Si no está disponible una vista de texto, se muestra una pantalla **Visualizar fuente de módulo** en blanco, con el mismo mensaje que el anterior en el área de mensajes.

Los siguientes programas no pueden añadirse al entorno de depuración ILE:

1. Programas ILE sin datos de depuración
2. Programas OPM sólo con datos de depuración no fuente
3. Programas OPM sin datos de depuración

En los dos primeros casos, se pasa el número de sentencia detenida. En el tercer caso, se pasa la instrucción MI detenida. La información se visualiza al final de una pantalla **Visualizar fuente de módulo** en blanco, como se muestra a continuación. En lugar del número de línea, se proporciona el número de sentencia o de instrucción.

```

                                Visualizar fuente de módulo
(Fuente no disponible)
F3=Fin programa  F12=Reanudar  F14=Trabajar con lista de módulos  F18=Trabajar con observ.
F21=Entrada de mandato  F22=Entrar en  F23=Visualizar salida
Número de observación 1 en instrucción 18, variable: SALARY

```

Figura 108. Ejemplo de panel Visualizar fuente de módulo

Seguir los pasos del objeto de programa

Una vez encontrado un punto de interrupción, puede ejecutarse un número especificado de sentencias de un objeto de programa, detener el programa de nuevo y volver a la pantalla Visualizar fuente de módulo. Esta operación se realiza mediante la función de paso del depurador del fuente ILE. El objeto de programa reanuda la ejecución en la sentencia siguiente del objeto de módulo en el que el programa se ha detenido. Normalmente, se utiliza un punto de interrupción para detener el objeto de programa.

Los puntos de interrupción pueden establecerse antes de que se llame al programa y mientras se están siguiendo los pasos del programa. Los puntos de interrupción también pueden generarse automáticamente para especificaciones de entrada y salida si se especifica la opción por omisión `OPTION(*DEBUGIO)`. Si se selecciona esta opción, una función `STEP` en una sentencia `READ` se detendrá en la especificación de entrada. Puede elegir no generar puntos de interrupción para especificaciones de entrada y salida con `OPTION(*NODEBUGIO)`.

Puede entrar en un programa OPM si tiene datos de depuración disponibles y si la sesión de depuración acepta programas OPM para depuración.

Puede seguir los pasos de un objeto de programa utilizando:

- F10 (Saltar) o F22 (Entrar en) en la pantalla Visualizar fuente de módulo
- El mandato de depuración `STEP`

La manera más sencilla de seguir los pasos de un objeto de programa de sentencia en sentencia es utilizando F10 (Saltar) o F22 (Entrar en) en la pantalla Visualizar fuente de módulo. Cuando se pulsa F10 (Saltar) o F22 (Entrar en), se ejecuta la sentencia siguiente del objeto de módulo visualizado en la pantalla Visualizar fuente de módulo, y el objeto de programa se detiene de nuevo.

Nota: No puede especificar el número de sentencias que sigue paso a paso cuando se utiliza F10 (Saltar) o F22 (Entrar en). Al pulsar F10 (Saltar) o F22 (Entrar en), se realiza un solo paso.

Otro modo de seguir los pasos de un objeto de programa es utilizando el mandato de depuración `STEP`. Este mandato le permite ejecutar más de una sentencia en un solo paso. El número de sentencias a ejecutar por omisión cuando se utiliza el mandato de depuración `STEP` es 1. Para seguir los pasos de un objeto de programa utilizando el mandato de depuración `STEP`, escriba:

```
STEP número-de-sentencias
```

en la línea de mandatos de depuración. La variable *número-de-sentencias* es el número de sentencias del objeto de programa que desea ejecutar en el paso siguiente antes de que el objeto de programa se detenga nuevamente. Por ejemplo, si escribe:

```
STEP 5
```

en la línea de mandatos de depuración, se ejecutarán las cinco sentencias siguientes del objeto de programa, éste se detendrá de nuevo y se visualizará la pantalla Visualizar fuente de módulo.

Cuando en una sesión de depuración, se encuentra una sentencia de llamada a otro programa o procedimiento, puede:

- Saltar la sentencia de llamada, o bien

Seguir los pasos del objeto programa

- Entrar en la sentencia de llamada.

Una sentencia de llamada para ILE RPG incluye cualquiera de las operaciones siguientes:

- CALL
- CALLB
- CALLP
- Cualquier operación en la que haya una expresión en el campo factor 2 ampliado y la expresión contenga una llamada a un procedimiento.

Si selecciona **saltar** la sentencia de llamada, permanecerá dentro del procedimiento actual. La sentencia de llamada se procesa como un sólo paso y el cursor pasa al paso siguiente a la llamada. Saltar es la modalidad de pasos por omisión.

Si selecciona **entrar en** la sentencia de llamada, cada sentencia de la sentencia de llamada se ejecuta como un solo paso. Dependiendo del número de pasos especificado, el mandato step puede finalizar en la sentencia de llamada, en cuyo caso el fuente de la sentencia de llamada se muestra en la pantalla Visualizar fuente de módulo.

Nota: No puede saltar ni entrar en subrutinas RPG. Sin embargo, puede saltar y entrar en subprocedimientos.

Saltar sentencias de llamada

Puede saltar las sentencias de llamada utilizando:

- F10 (Saltar) en la pantalla Visualizar fuente de módulo
- El mandato de depuración STEP OVER

Puede utilizar F10 (Saltar) de la pantalla Visualizar fuente de módulo para saltar una sentencia de llamada en una sesión de depuración. Si la sentencia de llamada que se ha de ejecutar es una operación CALL a otro objeto de programa, pulsar F10 (Saltar) hará que se ejecute por completo el objeto de programa antes de que pueda volver a detenerse el objeto de programa que efectúa la llamada. Del mismo modo, si la sentencia de llamada es una operación EVAL en la que se llama a un procedimiento en la expresión, la operación EVAL se lleva a cabo por completo, incluida la llamada al procedimiento, antes de que se detenga de nuevo el programa o procedimiento que efectúa la llamada.

Como alternativa, puede utilizar el mandato de depuración STEP OVER para saltar una sentencia de llamada en una sesión de depuración. Para utilizar el mandato de depuración STEP OVER, escriba:

STEP número-de-sentencias OVER

en la línea de mandatos de depuración. La variable *número-de-sentencias* es el número de sentencias que desea ejecutar en el paso siguiente antes de que vuelva a detenerse el proceso. Si se omite esta variable, el valor por omisión es 1.

Entrar en sentencias de llamada

Puede entrar en una sentencia de llamada utilizando:

- F22 (Entrar en) desde la pantalla Visualizar fuente de módulo
- El mandato de depuración STEP INTO

Seguir los pasos del objeto programa

Puede utilizar F22 (Entrar en) de la pantalla Visualizar fuente de módulo para entrar en un programa o procedimiento llamado de una sesión de depuración. Si la sentencia siguiente que se ha de ejecutar es una sentencia de llamada a otro programa o procedimiento, pulsar F22 (Entrar en) hará que se ejecute la primera sentencia ejecutable del programa o procedimiento llamado. El programa o procedimiento llamado aparecerá en la pantalla Visualizar fuente de módulo.

Nota: El programa o procedimiento llamado debe tener datos de depuración asociados para poder aparecer en la pantalla Visualizar fuente de módulo.

Como alternativa, puede utilizar el mandato de depuración STEP INTO para entrar en una sentencia de llamada de una sesión de depuración. Para utilizar el mandato de depuración STEP INTO, escriba:

```
STEP número-de-sentencias INTO
```

en la línea de mandatos de depuración. La variable *número-de-sentencias* es el número de sentencias que desea ejecutar en el paso siguiente antes de que vuelva a detenerse el proceso. Si se omite esta variable, el valor por omisión es 1.

Si una de las sentencias que se ejecutan contiene una sentencia de llamada, el depurador entrará en el programa o procedimiento llamado. Cada sentencia del programa o procedimiento llamado se contará en el paso. Si el paso finaliza en el programa o procedimiento llamado, el programa o procedimiento llamado aparecerá en la pantalla Visualizar fuente de módulo. Por ejemplo, si escribe

```
STEP 5 INTO
```

en la línea de mandatos de depuración, se ejecutarán las cinco sentencias siguientes del objeto de programa. Si la tercera sentencia es una operación CALL a otro objeto de programa, se ejecutan las dos sentencias del objeto de programa que efectúa la llamada y las tres primeras sentencias del objeto del programa llamado.

En el ejemplo de DEBUGEX, si entra STEP INTO (o pulsa F22) mientras está en la operación EVAL que llama al procedimiento c_proc, entrará en el módulo C.

El mandato STEP INTO funciona también con el mandato CL CALL. Puede sacar partido de esto para seguir los pasos del programa después de llamarlo. Tras arrancar el depurador del fuente, desde la pantalla inicial Visualizar fuente de módulo, escriba lo siguiente

```
STEP 1 INTO
```

Así se establecerá la cuenta de paso en 1. Utilice la tecla F12 para volver a la línea de mandatos y llamar al programa. El programa se detendrá en la primera sentencia que tenga datos de depuración.

CONSEJO

Para visualizar datos inmediatamente antes o después de que se ejecute un subprocedimiento, coloque los puntos de interrupción en las especificaciones de procedimiento que inician y finalizan el subprocedimiento.

Ejemplo de cómo entrar en un programa OPM utilizando F22

En este ejemplo, se utiliza la tecla F22 (Entrar en) para entrar en el programa RPGPGM desde el programa DEBUGEX.

1. Asegúrese de que la pantalla Visualizar fuente de módulo muestra el fuente de DBGEX.

Seguir los pasos del objeto programa

2. Para establecer un punto de interrupción incondicional en la línea 102, que es la última sentencia ejecutable antes de la operación CALL, escriba Break 102 y pulse Intro.
3. Pulse F3 (Finalizar programa) para salir de la pantalla Visualizar fuente de módulo.
4. Llame al programa. El programa se detiene en el punto de interrupción 102, como se muestra en la Figura 109.

```

                                Visualizar fuente de módulo
Programa:  DEBUGEX      Biblioteca:  MIBIBL      Módulo:  DBGEX
 98      * Fld1a es un campo de recubrimiento de Fld1. Dado que Fld1 se inicializa
 99      * para 'ABCDE', el valor de Fld1a(1) es 'A'. Después de la siguiente
100      * operación MOVE, el valor de Fld1a(1) será '1'.
101      *-----
102      C              MOVE      '1'              Fld1a(1)
103
104      *-----
105      * Llamar al programa RPGPGM, que es un objeto de programa separado.
106      *-----
107      C      Plist1      PLIST
108      C              PARM              PARM1
109      C              CALL      'RPGPGM'      Plist1
110
111      *-----
112      * Llamar a c_proc, que importa ExportFld desde el procedimiento principal.
                                Más...
Depurar . . .
F3=Fin programa      F6=Añadir/Borrar punto interrupción      F10=Saltar      F11=Ver variable
F12=Reanudar          F17=Observar variable      F18=Trabajar con observación      F24=Más teclas
Punto de interrupción en línea 102.
```

Figura 109. Pantalla Visualizar fuente de módulo de DBGEX antes de entrar en RPGPGM

5. Pulse F22 (Entrar en). Se ejecuta una sentencia del programa y, a continuación, se muestra la pantalla Visualizar fuente de módulo correspondiente a RPGPGM, tal como aparece en la Figura 110 en la página 231.
En este caso, se procesa la primera sentencia ejecutable de RPGPGM (línea 13) y a continuación el programa se detiene.

Nota: No puede especificarse el número de sentencias que se siguen paso a paso cuando se utiliza F22. Al pulsar F22, se realiza un solo paso.


```

                                Visualizar fuente de módulo
Programa:  RPGPGM      Biblioteca:  MIBIB
1          *=====
2          *  RPGPGM - Programa llamado por DEBUGEX para ilustrar las funciones STEP
3          *                      del depurador del fuente ILE.
4          *
5          *  Este programa recibe un parámetro InputParm desde DEBUGEX,
6          *  lo visualiza y lo devuelve.
7          *=====
8
9          D InputParm      S              4P 3
10
11         C  *ENTRY      PLIST
12         C              PARM              InputParm
13         C  InputParm   DSPLY
14         C              SETON
                                Final
Depurar . . . _____

F3=Fin programa  F6=Añadir/Borrar punto interrupción  F10=Saltar  F11=Ver variable
F12=Reanudar    F17=Observar variable  F18=Trab. con observación  F24=Más teclas
Paso completado en línea 13.

```

Figura 110. Entrar en RPGPGM

Si el depurador del fuente ILE no se ha establecido para aceptar programas OPM, o si no hay datos de depuración disponibles, verá una pantalla Visualizar fuente de módulo en blanco con un mensaje que indica que el fuente no está disponible. (Un programa OPM tiene datos de depuración si se ha compilado con OPTION(*SRCDBG) u OPTION(*LSTDBG)).

Ejemplo de cómo entrar en un subprocedimiento

En este ejemplo, se utiliza F22 (Entrar en) para entrar en el subprocedimiento Switch, que se encuentra en el módulo DEBUGEX.

1. Asegúrese de que la pantalla Visualizar fuente de módulo muestra el fuente de DBGEX.
2. Para establecer un punto de interrupción incondicional en la línea 120, que es la última sentencia ejecutable antes de la operación CALLP, escriba Break 120 y pulse Intro.
3. Pulse F3 (Finalizar programa) para salir de la pantalla Visualizar fuente de módulo.
4. Llame al programa. El programa se detiene en el punto de interrupción 119.
5. Pulse F22 (Entrar en). Se ejecuta la sentencia de llamada y, a continuación, la pantalla se desplaza al subprocedimiento, como se muestra en la Figura 111 en la página 232. Se procesa la primera sentencia ejecutable de RPGPGM (línea 13) y, a continuación, se detiene el proceso.

Seguir los pasos del objeto programa

Visualizar fuente de módulo					
Programa:	DEBUGEX	Biblioteca:	MIBIBL	Módulo:	DBGEX
141					
142	*=====				
143	* Definir el subprocedimiento Switch.				
144	*=====				
145	P Switch	B			
146	D Switch	PI			
147	D Parm		1A		
148	*-----				
149	* Definir una variable local a efectos de depuración.				
150	*-----				
151	D Local	S	5A	INZ('aaaaa')	
152					
153	C	IF	Parm = '1'		
154	C	EVAL	Parm = '0'		
155	C	ELSE			
Depurar . . .	<hr/>				
F3=Fin programa F6=Añadir/Borrar punto interrupción F10=Saltar F11=Ver variable					
F12=Reanudar F17=Observar variable F18=Trabajar con observación F24=Más teclas					
Paso completado en línea 145.					

Figura 111. Entrar en el subprocedimiento Switch

Visualización de datos y expresiones

Puede visualizar el contenido de los campos, estructuras de datos y matrices y evaluar expresiones. Existen dos maneras de visualizar o evaluar:

- F11 (Visualizar variable)
- Mandato de depuración EVAL

Para visualizar o cambiar DS.SUBF, utilice EVAL SUBF de DS

El ámbito de los campos utilizados en el mandato EVAL puede definirse utilizando el mandato QUAL en lenguajes como ILE C. Sin embargo, este mandato no se aplica actualmente a ILE RPG.

Nota: No puede visualizar valores de retorno porque no existe un nombre externo disponible para utilizar con el mandato de depuración EVAL.

La manera más sencilla de visualizar datos o expresiones es utilizando F11 (Visualizar variable) en la pantalla Visualizar fuente de Módulo. Para visualizar un campo utilizando F11 (Visualizar variable), sitúe el cursor en el campo que desea visualizar y pulse F11 (Visualizar variable). El valor actual del campo se mostrará en la línea de mensajes situada en la parte inferior de la pantalla Visualizar fuente de módulo.

En los casos en que se evalúan estructuras, registros o matrices, el mensaje devuelto cuando se pulsa F11 (Visualizar variable) puede constar de varias líneas. Los mensajes que ocupan varias líneas se muestran en la pantalla Evaluar expresión para que sea posible ver todo el texto. Una vez que haya visto el mensaje en dicha pantalla, pulse Intro para volver a la pantalla Visualizar fuente de módulo.

Para visualizar datos utilizando el mandato de depuración EVAL, escriba:
EVAL nombre-campo

Seguir los pasos del objeto programa

en la línea de mandatos de depuración. La variable *nombre-campo* es el nombre del campo, de la estructura de datos o de la matriz que desea visualizar o evaluar. El valor se muestra en la línea de mensajes si el mandato de depuración EVAL se teclea en la pantalla Visualizar fuente de módulo; el valor se visualiza en una sola línea. De lo contrario, aparece en la pantalla Evaluar expresión.

La Figura 112 muestra un ejemplo de utilización del mandato de depuración EVAL para visualizar el contenido del subcampo LastName.

```

Visualizar fuente de módulo
Programa:  DEBUGEX      Biblioteca:  MIBIBL      Módulo:    DBGEX
61      D  LastName      10A  INZ('Jones  ')
62      D  FirstName     10A  INZ('Fred   ')
63
64      *-----
65      * Definir prototipos para los procedimientos llamados c_proc y switch
66      *-----
67      D  c_proc          PR          *  EXTPROC('c_proc')
68      D  size            10U 0  VALUE
69      D  inzval          1A  CONST
70
71      D  Switch          PR
72      D  Parm            1A
73
74      *-----
75      * Definir parámetros para llamada sin prototipos

Más...

Depurar . .  eval LastName

```

F3=Fin programa F6=Añadir/Borrar punto interrupción F10=Saltar F11=Ver variable
 F12=Reanudar F17=Observar variable F18=Trabajar con observación F24=Más teclas
LASTNAME = 'Jones'

Figura 112. Visualización de un campo utilizando el mandato de depuración EVAL

La Figura 113 en la página 234 muestra la utilización del mandato EVAL con diferentes tipos de campos RPG. Los campos se basan en el fuente de Figura 121 en la página 246. También se proporcionan más ejemplos en la ayuda en línea del depurador del fuente.

Seguir los pasos del objeto programa

Campos escalares	Definición RPG
> EVAL String STRING = 'ABCDEF'	6A INZ('ABCDEF')
> EVAL Packed1D0 PACKED1D0 = -093.40	5P 2 INZ(-93.4)
> EVAL ZonedD3D2 ZONEDD3D2 = -3.21	3S 2 INZ(-3.21)
> EVAL Bin4D3 BIN4D3 = -4.321	4B 3 INZ(-4.321)
> EVAL Int3 INT3 = -128	3I 0 INZ(-128)
> EVAL Int5 INT5 = -2046	5I 0 INZ(-2046)
> EVAL Int10 INT10 = -31904	10I 0 INZ(-31904)
> EVAL Int20 INT20 = -463972	20I 0 INZ(-463972)
> EVAL Unsigned3 UNSIGNED3 = 128	3U 0 INZ(128)
> EVAL Unsigned5 UNSIGNED5 = 2046	5U 0 INZ(2046)
> EVAL Unsigned10 UNSIGNED10 = 31904	10U 0 INZ(31904)
> EVAL Unsigned20 UNSIGNED20 = 463972	20U 0 INZ(463972)
> EVAL DBCSString DBCSSTRING = '"BBCCDD"'	3G INZ(G'~BBCCDD~')
> EVAL NullPtr NULLPTR = SYP:*NULL	* INZ(*NULL)
Campos basados	
> EVAL String STRING = 'ABCDEF'	6A INZ('ABCDEF')
> EVAL BasePtr BASEPTR = SPP:C01947001218	* INZ(%ADDR(String))
> EVAL BaseString BASESTRING = 'ABCDEF'	6A BASED(BasePtr)
Campos de Fecha, Hora e Indicación de la hora	
> EVAL BigDate BIGDATE = '9999-12-31'	D INZ(D'9999-12-31')
> EVAL BigTime BIGTIME = '12.00.00'	T INZ(T'12.00.00')
> EVAL BigTstamp BIGTSTAMP = '9999-12-31-12.00.00.000000'	Z INZ(Z'9999-12-31-12.00.00.000000')

Figura 113. Ejemplo de mandatos EVAL basados en el módulo DBGEX

Resultados inesperados al evaluar variables

Si se sorprende del valor de las variables al depurar, compruebe si se produce alguna de las siguientes situaciones:

- El módulo está optimizado. Si el módulo está optimizado, puede que el depurador no muestre el valor más actualizado de una variable. Asimismo, si cambia una variable utilizando el depurador, puede que los efectos del cambio no se reflejen en la forma en que se ejecuta el programa.
- Algunos campos de entrada no se leen desde el archivo. Normalmente, los campos de entrada que no se utilizan en el programa no resultan afectados por una operación de entrada. Si especifica la palabra clave DEBUG en la especificación de control, se leerán todos los campos de entrada.

Visualización del contenido de una matriz

La especificación de un nombre de matriz con EVAL hará que se visualice la matriz completa. Para visualizar un elemento de una matriz, especifique el índice del elemento que desea visualizar entre paréntesis.

Para visualizar un rango de elementos, utilice la siguiente notación de rango:

EVAL nombre-campo (n...m)

La variable *nombre-campo* es el nombre de la matriz, la variable *n* es un número que representa el comienzo del rango y la variable *m* es un número que representa el final del rango.

La Figura 114 muestra la utilización de EVAL con la matriz de DBGEX.

```
> EVAL Array                                3S 2 DIM(2) INZ(1.23)
  ARRAY(1) = 1.23    ** Visualizar toda la matriz **
  ARRAY(2) = 1.23
> EVAL Array(2)      ** Visualizar segundo elemento **
  ARRAY(2) = 1.23
> EVAL Array(1..2)    ** Visualizar rango de elementos **
  ARRAY(1) = 1.23
  ARRAY(2) = 1.23
```

Figura 114. Ejemplos de mandatos EVAL para una matriz

Visualización del contenido de una tabla

La utilización de EVAL sobre una tabla hace que se visualice el elemento de tabla actual. Puede visualizar toda la tabla utilizando la notación de rango. Por ejemplo, para visualizar una tabla de tres elementos, escriba:

EVAL TableA(1..3)

Puede cambiar el elemento actual utilizando la función incorporada %INDEX. Para determinar el valor del índice de la tabla, entre el mandato siguiente:

EVAL _QRNU_TABI_nombre

donde *nombre* representa el nombre de la tabla en cuestión.

La Figura 115 en la página 236 muestra la utilización de EVAL con la tabla de DBGEX.

Seguir los pasos del objeto programa

```

3      DIM(3) CTDATA
      Datos de compilación: **
> EVAL TableA      ** Mostrar valor en      aaa
TABLEA = 'aaa'      índice actual            bbb
                                                    ccc

> EVAL TableA(1)    ** Especificar índice 1 **
TABLEA(1) = 'aaa'

> EVAL TableA(2)    ** Especificar índice 2 **
TABLEA(2) = 'bbb'

> EVAL _QRNU_TABI_TableA ** Visualizar valor de índice actual **
_QRNU_TABI_TABLEA = 1

> EVAL TableA(1..3) ** Especificar toda la tabla **
TABLEA(1) = 'aaa'
TABLEA(2) = 'bbb'
TABLEA(3) = 'ccc'

> EVAL TableA=%INDEX(3) ** Cambiar índice actual a 3 **
> EVAL TableA
TABLEA = 'ccc'
```

Figura 115. Ejemplos de mandatos EVAL para una tabla

Visualización de estructuras de datos

Puede visualizar el contenido de una estructura de datos o de sus subcampos tal como lo haría con cualquier campo autónomo. Sólo tiene que utilizar el nombre de la estructura de datos después de EVAL para ver todo el contenido, o bien el nombre del subcampo para ver una parte.

Si la estructura de datos está calificada, especifique los subcampos utilizando la siguiente notación:

EVAL nombre-subcampo OF nombre-estructura-de-datos

Por ejemplo, para visualizar el subcampo NAME de la estructura de datos calificada INFO, escriba:

EVAL NAME OF INFO

Al visualizar una estructura de datos de múltiples apariciones, la especificación de EVAL con el nombre de la estructura de datos hará que se muestren los subcampos que utilizan en índice actual. Para especificar una aparición determinada, especifique el índice entre paréntesis a continuación del nombre de la estructura de datos. Por ejemplo, para visualizar el contenido de la segunda aparición de DS1, escriba:

EVAL DS1(2)

Del mismo modo, para ver el contenido de una aparición determinada de un subcampo, utilice la notación de índices.

Para determinar el valor del índice actual, entre el mandato siguiente:

EVAL _QRNU_DSI_nombre

donde *nombre* representa el nombre de la estructura de datos en cuestión.

Si un subcampo está definido como un recubrimiento de matriz de otro subcampo, para ver el contenido del subcampo de recubrimiento puede utilizar la función incorporada %INDEX para especificar la aparición, y la notación de índices para especificar la matriz.

Seguir los pasos del objeto programa

Un modo alternativo para visualizar un subcampo que es un recubrimiento de matriz es utilizar la notación siguiente:

EVAL nombre-subcampo(índice-aparición, índice-matriz)

donde la variable *nombre-subcampo* el nombre del subcampo que desea visualizar, *índice-aparición* es el número de la aparición de matriz que se visualizará e *índice-matriz* es el número del elemento que se visualizará.

La Figura 116 muestra algunos ejemplos de la utilización de EVAL con las estructuras de datos definidas en DBGEX.

```
** Tenga en cuenta que puede entrar nombre estructura datos o un nombre subcampo.
> EVAL DS3
  TITLE OF DS3 = 'Mr. '          5A  INZ('Mr. ')
  LASTNAME OF DS3 = 'Jones      ' 10A  INZ('Jones ')
  FIRSTNAME OF DS3 = 'Fred      ' 10A  INZ('Fred ')
> EVAL LastName
  LASTNAME = 'Jones '
> EVAL DS1 OCCURS(3)
  FLD1 OF DS1 = 'ABCDE'          5A  INZ('ABCDE')
  FLD1A OF DS1(1) = 'A'          1A  DIM(5) OVERLAY(FLD1)
  FLD1A OF DS1(2) = 'B'          5B 2 INZ(123.45)
  FLD1A OF DS1(3) = 'C'
  FLD1A OF DS1(4) = 'D'
  FLD1A OF DS1(5) = 'E'
  FLD2 OF DS1 = 123.45
> EVAL _QRNU_DSI_DSI ** Determinar el valor de índice actual **
  _QRNU_DSI_DSI = 1
> EVAL DS1=%INDEX(2) ** Cambiar la aparición de DS1 **
  DS1=%INDEX(2) = 2
> EVAL Fld1 ** Visualizar un subcampo **
  FLD1 = 'ABCDE' (aparición actual)
> EVAL fld1(2) (segunda aparición)
  FLD1(2) = 'ABCDE'
> EVAL Fld1a ** Visualizar un subcampo de recubrimiento de matriz **
  FLD1A OF DS1(1) = 'A' (aparición actual)
  FLD1A OF DS1(2) = 'B'
  FLD1A OF DS1(3) = 'C'
  FLD1A OF DS1(4) = 'D'
  FLD1A OF DS1(5) = 'E'
> EVAL Fld1a(2,1) ** Visualizar segunda aparición, primer elemento **
  FLD1A(2,1) = 'A'
> EVAL Fld1a(2,1..2) ** Visualizar segunda aparición, primer-segundo elementos
  FLD1A(2,1) = 'A'
  FLD1A(2,2) = 'B'
> EVAL ID_NUM OF QUALDS ** Visualizar un subcampo de una DS calificada
  ID_NUM OF QUALDS = 1100022
> EVAL ID_NUM OF LIKE_QUALDS ** Visualizar el mismo subcampo en DS diferente
  ID_NUM OF LIKE_QUALDS = 0
> EVAL COUNTRY OF LIKE_QUALDS(1) ** Un elemento de matriz de una DS calificada
  COUNTRY OF LIKE_QUALDS(1) = 'CANADA'
```

Figura 116. Utilización de EVAL con estructuras de datos

Para visualizar una estructura de datos para la cual no se han definido subcampos, debe utilizar la función de visualización de caracteres de EVAL, que se trata más adelante.

Visualización de indicadores

Los indicadores se definen como campos de caracteres de 1 byte. A excepción de los indicadores como *INLR, puede visualizar indicadores como "*INxx" o "*IN(xx)". Puesto que el sistema almacena los indicadores como una matriz, puede

Seguir los pasos del objeto programa

visualizarlos todos o sólo una parte de ellos utilizando la notación de rangos. Por ejemplo, si entra EVAL *IN, obtendrá una lista de los indicadores del 01 a 99. Para visualizar los indicadores *IN01 a *IN06, entraría EVAL *IN(1..6).

La Figura 117 muestra cada una de estas formas de utilizar los indicadores como si estuvieran definidos en DBGEX.

```
> EVAL IN02
  El identificador no existe.
> EVAL *IN02
  *IN02 = '1'
> EVAL *IN(02)
  *IN(02) = '1'
> EVAL *INLR
  *INLR = '0'
> EVAL *IN(LR)
  El identificador no existe.
> EVAL *IN(1..6)          ** Para visualizar un rango de indicadores **
  *IN(1) = '0'
  *IN(2) = '1'
  *IN(3) = '0'
  *IN(4) = '1'
  *IN(5) = '0'
  *IN(6) = '1'
```

Figura 117. Ejemplos de mandatos EVAL para una matriz

Visualización de campos como valores hexadecimales

Puede utilizar el mandato de depuración EVAL para visualizar el valor de los campos en formato hexadecimal. Para visualizar una variable en formato hexadecimal, escriba:

EVAL nombre-campo: x número-de-bytes

en la línea de mandatos de depuración. La variable *nombre-campo* es el nombre del campo que desea visualizar en formato hexadecimal. 'x' especifica que el campo se va a visualizar en formato hexadecimal. La variable *número-de-bytes* indica el número de bytes que se visualizarán. Si no se especifica ninguna longitud después de 'x', se utilizará el tamaño del campo como longitud. Siempre se visualiza un mínimo de 16 bytes. Si la longitud del campo es inferior a 16 bytes, el espacio restante *se rellena con ceros* hasta llegar a los 16 bytes.

Por ejemplo, el campo String se define como una serie de seis caracteres. Para averiguar cuál es el equivalente hexadecimal de los tres primeros caracteres, entraría lo siguiente:

```
EVAL String: x 3
Resultado:
000000    C1C2C3.. ..... - ABC.....
```

Visualización de campos en formato de caracteres

Puede utilizar el mandato de depuración EVAL para visualizar un campo en formato de caracteres. Para visualizar una variable en formato de caracteres, escriba:

EVAL nombre-campo: c número-de-caracteres

en la línea de mandatos de depuración. La variable *nombre-campo* es el nombre del campo que desea visualizar en formato de caracteres. 'c' especifica el número de caracteres que se visualizarán.

Seguir los pasos del objeto programa

Por ejemplo, en el programa DEBUGEX, la estructura de datos DS2 no tiene definido ningún subcampo. Varias operaciones MOVE mueven valores al subcampo.

Puesto que no hay subcampos definidos, no puede visualizar la estructura de datos. Por lo tanto, para ver el contenido puede utilizar la función de visualización de caracteres de EVAL.

```
EVAL DS2:C 20           Resultado:  DS2:C 20 = 'aaaaaaaaabbbbbbbbbb'
```

Visualización de datos UCS-2

El valor visualizado para campos UCS-2 se ha convertido a caracteres legibles. Por ejemplo, si un campo UCS-2 se ha establecido en %UCS2('abcde'), el valor visualizado para dicho campo será 'abcde'. Puede visualizar datos UCS-2 en cualquier campo utilizando el sufijo :u para EVAL.

Visualización de campos de longitud variable

Al utilizar EVAL fldname para un campo de longitud variable, sólo se muestra la parte de datos del campo. Si utiliza sufijos tales como :c o :x para el campo, se muestra la totalidad del campo, incluyendo la longitud. Para determinar la longitud actual de un campo de longitud variable, utilice EVAL fldname:x. La longitud corresponde a los cuatro primeros dígitos hexadecimales, en formato binario. Debe convertir este valor a formato decimal para obtener la longitud; por ejemplo, si el resultado es 003DF 1F2..., la longitud es 003D, que es $(3 * 16) + 13 = 61$.

Visualización de datos direccionados por punteros

Si desea ver a qué elemento está señalando un puntero, puede utilizar el mandato EVAL con los sufijos :c o :x. Por ejemplo, si el campo de puntero PTR1 está señalando a 10 bytes de datos de tipo carácter,

```
EVAL PTR1:c 10
```

mostrará el contenido de esos 10 bytes.

También puede mostrar el contenido en formato hexadecimal utilizando:

```
EVAL PTR1:x 10
```

Este es especialmente útil cuando los datos que direcciona el puntero no están almacenados en formato imprimible, como por ejemplo datos empaquetados o binarios.

Visualización de campos con capacidad de nulos

Puede utilizar el mandato de depuración EVAL para visualizar el indicador de nulo de un campo con capacidad de nulos. El indicador de nulo es una variable interna (similar a la variable de índice para DS de múltiples apariciones) denominada _QRNU_NULL_nombrecampo. El nombrecampo puede ser el nombre de una matriz si la matriz tiene capacidad de nulos.

Cuando el depurador visualiza un campo con capacidad de nulos, el contenido del campo se visualiza independientemente de si el campo se considera nulo. Por ejemplo, suponga que FLD1 tiene capacidad de nulos y actualmente es nulo. El resultado de EVAL _QRNU_NULL_FLD1 es '1' y EVAL FLD1 muestra el contenido actual de FLD1, aunque su indicador de nulo esté activado.

```
EVAL _QRNU_NULL_FLD1   Resultado:  _QRNU_NULL_FLD1 = '1'
EVAL FLD1              Resultado:  FLD1 = 'abcde'
```

Seguir los pasos del objeto programa

Utilización de las funciones de depuración incorporadas

Cuando se utiliza el depurador del fuente ILE, están disponibles las siguientes funciones incorporadas:

%SUBSTR

Dividir en subseries un campo de serie.

%ADDR

Recuperar la dirección de un campo.

%INDEX

Cambiar el índice de una tabla o estructura de datos de múltiples apariciones.

%VARS

Identifica el parámetro especificado como una variable.

La función incorporada %SUBSTR le permite dividir en subseries una variable de serie. El primer parámetro debe ser un identificador de serie, el segundo parámetro es la posición inicial y el tercer parámetro es el número de caracteres de un solo byte o de doble byte. Además, el segundo y el tercer parámetro deben ser literales enteros positivos. Los parámetros se delimitan con uno o varios espacios.

Utilice la función incorporada %SUBSTR para:

- Visualizar una parte de un campo de caracteres
- Asignar una parte de un campo de caracteres
- Utilizar una parte de un campo de caracteres en cualquier lado de una expresión de interrupción condicional.

La Figura 118 en la página 241 muestra algunos ejemplos de la utilización de %SUBSTR basada en el fuente de la Figura 121 en la página 246.

```

> EVAL String
  STRING = 'ABCDE '
** Visualizar los dos primeros caracteres de String **
> EVAL %substr (String 1 2)
  %SUBSTR (STRING 1 2) = 'AB'
> EVAL TableA
  TABLEA = 'aaa'
** Visualizar el primer carácter del primer elemento de la tabla **
> EVAL %substr(TableA 1 1)
  %SUBSTR(TABLEA 1 1) = 'a'
> EVAL BigDate
  BIGDATE = '1994-10-23'
** Establecer String igual a los cuatro primeros caracteres de BigDate **
> EVAL String=%substr(BigDate 1 4)
  STRING=%SUBSTR(BIGDATE 1 4) = '1994 '
> EVAL Fld1 (5 caracteres)
  FLD1 = 'ABCDE'
> EVAL String (6 caracteres)
  STRING = '123456'
** Establecer los caracteres 2-5 de String iguales a los
    cuatro primeros caracteres de Fld1 **
> EVAL %substr(String 2 4) = %substr(Fld1 1 4)
  %SUBSTR(STRING 2 4) = %SUBSTR(FLD1 1 4) = 'ABCD'
> EVAL String
  STRING = '1ABCD6'
** Sólo puede utilizar %SUBSTR en series de caracteres o gráficas. **
> EVAL %substr (Packed1D0 1 2)
  Se ha producido un error de tipo String.

```

Figura 118. Ejemplos de %SUBSTR utilizando DBGEX

Para cambiar el índice actual, puede utilizar la función incorporada %INDEX, donde el índice se especifica entre paréntesis a continuación del nombre de la función. Puede encontrar un ejemplo de %INDEX en la sección de tabla de la Figura 115 en la página 236 y de la Figura 116 en la página 237.

Nota: La función %INDEX cambiará el índice actual por el que se especifique. Por tanto, cualquier sentencia fuente que haga referencia a la tabla o la estructura de datos de múltiples apariciones que sigue a la sentencia EVAL puede estar operando con un índice distinto al que se esperaba.

Utilice la función incorporada de depuración %VARS cuando el nombre de variable entre en conflicto con los nombres de mandatos de depuración. Por ejemplo, puede utilizarse EVAL %VAR(EVAL) para evaluar una variable denominada EVAL, ya que EVAL EVAL sería un error de sintaxis.

Cambio del valor de los campos

Puede cambiar el valor de los campos utilizando el mandato EVAL con un operador de asignación (=).

El ámbito de los campos utilizados en el mandato EVAL se define utilizando el mandato QUAL. Sin embargo, no es necesario definir específicamente el ámbito de los campos contenidos en un módulo ILE RPG, ya que todos tienen un ámbito global.

Para cambiar el valor de un campo, escriba:

```
EVAL nombre-campo = valor
```

Cambio del valor de los campos

en la línea de mandatos de depuración. *Nombre-campo* es el nombre de la variable que desea cambiar y *valor* es el identificador, literal o constante que desea asignar a la variable *nombre-campo*. Por ejemplo,

```
EVAL CONTADOR=3
```

cambia el valor de *CONTADOR* a 3 y muestra

```
CONTADOR=3 = 3
```

en la línea de mensajes de la pantalla Visualizar fuente de módulo.

Utilice el mandato de depuración EVAL para asignar datos numéricos, alfabéticos y alfanuméricos a los campos. También puede utilizar la función incorporada %SUBSTR en la expresión de asignación.

Cuando se asignan valores a un campo de caracteres, se aplican las siguientes normas:

- Si la longitud de la expresión origen es menor que la longitud de la expresión destino, los datos se justifican por la izquierda en la expresión destino y las posiciones restantes se rellenan con espacios en blanco.
- Si la longitud de la expresión origen es mayor que la longitud de la expresión destino, los datos se justifican por la izquierda en la expresión destino y se truncan a la longitud de la expresión destino.

Nota: A los campos de gráficos se les puede asignar lo siguiente:

- Otro campo de gráficos
- Un literal de gráficos con el formato G'oK1K2i'
- Un literal hexadecimal con el formato X'dígitos hexadecimales'

Los campos UCS-2 deben cambiarse utilizando constantes hexadecimales. Por ejemplo, puesto que %UCS2('AB') = U'00410042', para establecer un campo UCS-2 en el formato UCS-2 de 'AB' en el depurador, utilizará EVAL ucs2 = X'00410042'.

Los campos de longitud variable pueden asignarse utilizando, por ejemplo, EVAL varfldname = 'abc'. Así se establece la parte de datos del campo en 'abc' y la parte de longitud en 3. Para establecer la parte de longitud sin cambiar los datos, determine el valor hexadecimal de la longitud (por ejemplo, 11 es X'000B') y utilice EVAL %SUBSTR(varfldname 1 2) = X'000B'.

Al asignar literales a los campos, se aplican las normas habituales de RPG:

- Los literales de caracteres deben ir entre apóstrofes.
- Los literales de gráficos deben especificarse como G'oDDDDi', siendo "o" el desplazamiento a teclado ideográfico e "i" el desplazamiento a teclado estándar.
- Los literales hexadecimales deben ir entre apóstrofes, precedidos de una 'x'.
- Los literales numéricos no deben ir entre apóstrofes.

Nota: No puede asignar una constante figurativa a un campo empleando el mandato de depuración EVAL. Las constantes figurativas no están soportadas por el mandato de depuración EVAL.

La Figura 119 en la página 243 muestra algunos ejemplos de cambio de valores de campo según el fuente de la Figura 121 en la página 246. También se proporcionan más ejemplos en la ayuda en línea del depurador del fuente.

```

** Longitud destino = Longitud origen **
> EVAL String='123456'      (6 caracteres)
  STRING='123456' = '123456'
> EVAL ExportFld           (6 caracteres)
  EXPORTFLD = 'export'
> EVAL String=ExportFld
  STRING=EXPORTFLD = 'export'
** Longitud destino < Longitud origen **
> EVAL String              (6 caracteres)
  STRING = 'ABCDEF'
> EVAL LastName            (10 caracteres)
  LASTNAME='Williamson' = 'Williamson'
> EVAL String=LastName
  STRING=LASTNAME = 'Willia'
** Longitud destino > Longitud origen **
> EVAL String              (6 caracteres)
  STRING = '123456'
> EVAL TableA              (3 caracteres)
  TABLEA = 'aaa'
> EVAL String=TableA
  STRING=TABLEA = 'aaa '
** Utilización de %SUBSTR **
> EVAL BigDate
  BIGDATE = '1994-10-23'
> EVAL String=%SUBSTR(BigDate 1 4)
  STRING=%SUBSTR(BIGDATE 1 4) = '1994 '
** Longitud destino subserie > Longitud origen subserie **
> EVAL string = '123456'
  STRING = '123456' = '123456'
> EVAL LastName='Williamson'
  LASTNAME='Williamson' = 'Williamson'
> EVAL String = %SUBSTR(Lastname 1 8)
  STRING = %SUBSTR(LASTNAME 1 8) = 'Willia'
** Longitud destino subserie < Longitud destino subserie **
> EVAL TableA
  TABLEA = 'aaa'
> EVAL String
  STRING = '123456'
> EVAL String=%SUBSTR(TableA 1 4)
  La subserie llega más allá del final de la serie.      ** Error **
> EVAL String
  STRING = '123456'

```

Figura 119. Ejemplos de cambio de los valores de campos según DBGEX

Visualización de atributos de un campo

Se puede visualizar los atributos de un campo utilizando el mandato de depuración Atributo (ATTR). Los atributos son el tamaño (en bytes) y el tipo de la variable tal como está registrado en la tabla de símbolos de depuración.

La Figura 120 en la página 244 muestra algunos ejemplos de visualización de atributos de campos según el fuente de la Figura 121 en la página 246. También se proporcionan más ejemplos en la ayuda en línea del depurador del fuente.

Igualar un nombre a un campo, expresión o mandato

```
> ATTR NullPtr
  TYPE = PTR, LENGTH = 16 BYTES
> ATTR ZonedD3D2
  TYPE = ZONED(3,2), LENGTH = 3 BYTES
> ATTR Bin4D3
  TYPE = BINARY, LENGTH = 2 BYTES
> ATTR Int3
  TYPE = INTEGER, LENGTH = 1 BYTES
> ATTR Int5
  TYPE = INTEGER, LENGTH = 2 BYTES
> ATTR Unsigned10
  TYPE = CARDINAL, LENGTH = 4 BYTES
> ATTR Unsigned20
  TYPE = CARDINAL, LENGTH = 8 BYTES
> ATTR Float4
  TYPE = REAL, LENGTH = 4 BYTES
> ATTR Float8
  TYPE = REAL, LENGTH = 8 BYTES
> ATTR Arry
  TYPE = ARRAY, LENGTH = 6 BYTES
> ATTR tablea
  TYPE = FIXED LENGTH STRING, LENGTH = 3 BYTES
> ATTR tablea(2)
  TYPE = FIXED LENGTH STRING, LENGTH = 3 BYTES
> ATTR BigDate
  TYPE = FIXED LENGTH STRING, LENGTH = 10 BYTES
> ATTR DS1
  TYPE = RECORD, LENGTH = 9 BYTES
> ATTR SpcPtr
  TYPE = PTR, LENGTH = 16 BYTES
> ATTR String
  TYPE = FIXED LENGTH STRING, LENGTH = 6 BYTES
> ATTR *IN02
  TYPE = CHAR, LENGTH = 1 BYTES
> ATTR DBCSString
  TYPE = FIXED LENGTH STRING, LENGTH = 6 BYTES
```

Figura 120. Ejemplos de visualización de atributos de campos según DBGEX

Igualar un nombre a un campo, expresión o mandato

Puede utilizar el mandato de depuración EQUATE para igualar un nombre a un campo, expresión o mandato para su uso abreviado. De este modo podrá utilizar ese nombre solo o con otra expresión. Si lo utiliza con otra expresión, el valor del nombre se determinará antes de evaluar la expresión. Estos nombres permanecen activos hasta que finaliza la sesión de depuración o se elimina el nombre.

Para igualar un nombre a un campo, expresión o mandato de depuración, escriba:
EQUATE nombre-abreviado definición

en la línea de mandatos de depuración. *Nombre-abreviado* es el nombre que desea igualar a un campo, expresión o mandato de depuración, y *definición* es el campo, expresión o mandato de depuración al que el nombre se iguala.

Por ejemplo, para definir un nombre abreviado llamado *DC* que visualice el contenido de un campo denominado *CONTADOR*, escriba:

EQUATE DC EVAL CONTADOR

en la línea de mandatos de depuración. Ahora, cada vez que escriba *DC* en la línea de mandatos de depuración, se ejecutará el mandato *EVAL CONTADOR*.

Igualar un nombre a un campo, expresión o mandato

El número máximo de caracteres que se pueden escribir en un mandato EQUATE es de 144. Si no se facilita una definición y hay un mandato EQUATE anterior que define el nombre, se eliminará la definición anterior. Si el nombre no estaba definido anteriormente, se muestra un mensaje de error.

Para ver los nombres que se han definido con el mandato de depuración EQUATE para una sesión de depuración, escriba:

```
DISPLAY EQUATE
```

en la línea de mandatos de depuración. En la pantalla Evaluar expresión se visualizará una lista de los nombres activos.

Soporte de Idiomas Nacionales de depuración del fuente para ILE RPG

Debe tener en cuenta las siguientes condiciones existentes al trabajar con el Soporte de Idiomas Nacionales de depuración del fuente para ILE RPG.

- Cuando se visualiza una vista en la pantalla Visualizar fuente de módulo, el depurador del fuente convierte todos los datos al CCSID (Identificador de Juego de Caracteres Codificado) del trabajo de depuración.
- Al asignar literales a los campos, el depurador del fuente no realizará la conversión al CCSID de los literales entre apóstrofes (por ejemplo, 'abc'). Además, los literales que aparecen entre apóstrofes son sensibles a las mayúsculas y minúsculas.

Consulte el capítulo sobre depuración en la publicación *ILE Concepts* para obtener más información acerca de las restricciones del NLS.

Ejemplo de fuente para ejemplos de depuración

La Figura 121 en la página 246 muestra el fuente para el procedimiento principal del programa DEBUGEX. La mayoría de los ejemplos y pantallas que se muestran en este capítulo se basan en este fuente. La Figura 122 en la página 249 y la Figura 123 en la página 250 muestran el fuente del programa RPGPGM llamado y y del procedimiento cproc, respectivamente.

El programa DEBUGEX está diseñado para mostrar los distintos aspectos del depurador del fuente ILE y de los vuelcos con formato ILE RPG. Puede encontrar los vuelcos de ejemplo en el capítulo siguiente.

Los pasos que se indican a continuación describen cómo se ha creado el programa DEBUGEX para utilizarlo en estos ejemplos:

1. Para crear el módulo DBGEX utilizando el fuente de la Figura 121 en la página 246, escriba:

```
CRTRPGMOD MODULE(MYLIB/DBGEX) SRCFILE(MYLIB/QRPGLESRC) DBGVIEW(*ALL)
      TEXT('Módulo principal del programa de depuración de ejemplo')
```

Se ha elegido DBGVIEW(*ALL) para mostrar las diferentes vistas disponibles.

2. Para crear el módulo C utilizando el fuente de la Figura 123 en la página 250, escriba:

```
CRTCMOD MODULE(MIBIB/cproc) SRCFILE(MIBIB/QCLESRC) DBGVIEW(*SOURCE)
      TEXT('Procedimiento C para programa de depuración de ejemplo')
```

3. Para crear el programa DEBUGEX, escriba:

```
CRTPGM PGM(MYLIB/DEBUGEX) MODULE(MYLIB/DBGEX MYLIB/CPROC)
      TEXT('Programa de depuración de ejemplo')
```

Ejemplo de fuente para ejemplos de depuración

El primer módulo, DBGEX, es el módulo de entrada de este programa. El programa se ejecutará en un grupo de activación nuevo (es decir, *NEW) cuando se le llame.

4. Para crear el programa RPG llamado utilizando el fuente de la Figura 122 en la página 249, escriba:

```
CRTBNDRPG PGM(MIBIB/RPGPGM) DFTACTGRP(*NO)
          DBGVIEW(*SOURCE) ACTGRP(*NEW)
          TEXT('Programa RPG para programa de depuración de ejemplo')
```

Se podría haber creado RPGPGM de modo que se ejecutara en el grupo de activación por omisión de OPM. Sin embargo, se ha optado por ejecutarlo en el mismo grupo de activación que DEBUGEX y, puesto que éste sólo necesita un grupo de activación temporal, se ha elegido *NEW para ambos programas.

```
*=====
*  DEBUGEX - Programa diseñado para ilustrar el uso del depurador del
*            fuente ILE con el fuente ILE RPG. Proporciona un ejemplo
*            de diferentes tipos y estructuras de datos.
*
*            También puede utilizarse para producir vuelcos format. ejemplo.
*=====
*-----*
* La palabra clave DEBUG habilita el recurso de vuelco formateado.
*-----*
H DEBUG
*-----*
* Definir campos autónomos para diferentes tipos de datos ILE RPG.
*-----*
D String          S          6A  INZ('ABCDEF')
D Packed1D0       S          5P 2 INZ(-93.4)
D ZonedD3D2       S          3S 2 INZ(-3.21)
D Bin4D3          S          4B 3 INZ(-4.321)
D Bin9D7          S          9B 7 INZ(98.7654321)
D DBCSString      S          3G  INZ(G'"BBCCDD"')
D UCS2String      S          5C  INZ(%UCS2('ucs-2'))
D CharVarying     S          5A  INZ('abc') VARYING
D Int3            S          3I 0 INZ(-128)
D Int5            S          5I 0 INZ(-2046)
D Int10           S         10I 0 INZ(-31904)
D Int20           S         20I 0 INZ(-463972)
D Unsigned3       S          3U 0 INZ(128)
D Unsigned5       S          5U 0 INZ(2046)
D Unsigned10      S         10U 0 INZ(31904)
D Unsigned20      S         20U 0 INZ(463972)
D Float4          S          4f  INZ(7.2098)
D Float8          S          8f  INZ(-129.0978652)
D DBCSString      S          3G  INZ(G'"BBCCDD"')
```

Figura 121. Fuente para el módulo DBGEX (Pieza 1 de 4). DBGEX es el módulo principal del programa DEBUGEX.

Ejemplo de fuente para ejemplos de depuración

```

*   Punteros
D NullPtr      S          *   INZ(*NULL)
D BasePtr      S          *   INZ(%ADDR(String))
D ProcPtr      S          *   ProcPtr INZ(%PADDR('c_proc'))
D BaseString   S          6A   BASED(BasePtr)
D BaseOnNull   S          10A  BASED(NullPtr)

*
D SpcPtr       S          *
D SpcSiz       C          8
*   Fecha, Hora, Indicación de la hora
D BigDate      S          D   INZ(D'9999-12-31')
D BigTime      S          T   INZ(T'12.00.00')
D BigTstamp    S          Z   INZ(Z'9999-12-31-12.00.00.000000')
*   Matriz
D Array        S          3S 2 DIM(2) INZ(1.23)
*   Tabla
D TableA       S          3    DIM(3) CTDATA
*-----*
* Definir diferentes tipos de estructuras de datos.
*-----*
D DS1          DS          OCCURS(3)
D Fld1         5A          INZ('ABCDE')
D Fld1a        1A          DIM(5) OVERLAY(Fld1)
D Fld2         5B 2        INZ(123.45)
*
D DS2          DS          10    OCCURS(2)
*
D DS3          DS
D Title        5A          INZ('Mr. ')
D LastName     10A         INZ('Jones ')
D FirstName    10A         INZ('Fred ')
*
D QUALDS       DS          QUALIFIED
D Id_Num       8S 0
D Country      20A         DIM(10)
D LIKE_QUALDS  DS          LIKEDS(QUALDS)
*-----*
* Definir prototipos para procedimientos c_proc y switch llamados
*-----*
D c_proc       PR          *   EXTPROC('c_proc')
D size         10U 0       VALUE
D inzval       1A          CONST
D Switch       PR
D Parm         1A
*-----*
* Definir parámetros para llamada sin prototipos
*   PARM1 se utiliza al llamar al programa RPGPROG.
*-----*
D PARM1        S          4P 3 INZ(6.666)
D EXPORTFLD    S          6A   INZ('export') EXPORT

```

Figura 121. Fuente para el módulo DBGEX (Pieza 2 de 4). DBGEX es el módulo principal del programa DEBUGEX.

Ejemplo de fuente para ejemplos de depuración

```

=====
* Ahora, la operación para modificar valores o llamar a otros objetos.
=====
*-----*
* Mover 'a's a la estructura de datos DS2. Después del movimiento,
* la primera aparición de DS2 contiene 10 'a's de carácter.
*-----*
C          MOVE      *ALL'a'      DS2
*-----*
* Cambiar la aparición de DS2 a 2 y mover 'b's a DS2, haciendo que
* los 10 primeros bytes sean 'a's y los segundos 10 bytes sean 'b's.
*-----*
C      2          OCCUR      DS2
C          MOVE      *ALL'b'      DS2
*-----*
* Fld1a es un campo de recubrimiento de Fld1. Dado que Fld1 se inicializa
* para 'ABCDE', el valor de Fld1a(1) es 'A'. Después de la siguiente
* operación MOVE, el valor de Fld1a(1) es '1'.
*-----*
C          MOVE      '1'          Fld1a(1)
*-----*
* Llamar al programa RPGPGM, que es un objeto de programa separado.
*-----*
C      Plist1      PLIST
C          PARM
C          CALL      'RPGPGM'      Parm1
C                          Plist1
*-----*
* Llamar a c_proc, que importa ExportFld desde el procedimiento principal.
*-----*
C          EVAL      SpcPtr = c_proc(SpcSiz : 'P')
*-----*
* Llamar a un subprocedimiento local Switch, que reserva el valor de
* un indicador.
*-----*
C          EVAL      *IN10 = '0'
C          CALLP      Switch(*in10)

```

Figura 121. Fuente para el módulo DBGEX (Pieza 3 de 4). DBGEX es el módulo principal del programa DEBUGEX.

Ejemplo de fuente para ejemplos de depuración

```

*-----*
* Después de la siguiente operación SETON, *IN02 = 1.
*-----*
C          SETON                                020406
C          IF      *IN02 = '1'
C          MOVE    '1994-09-30'  BigDate
C          ENDIF
*-----*
* Colocar un valor nuevo en la segunda celda de Array.
*-----*
C          MOVE    4          Array
*-----*
* Ahora, iniciar un vuelco formateado y volver, estableciendo en LR.
*-----*
C          DUMP
C          SETON                                LR
*-----*
* Definir el subprocedimiento Switch.
*-----*
P Switch      B
D Switch      PI
D  Parm              1A
*-----*
* Definir una variable local a efectos de depuración.
*-----*
D Local      S          5A      INZ('aaaaa')
C          IF      Parm = '1'
C          EVAL    Parm = '0'
C          ELSE
C          EVAL    Parm = '1'
C          ENDIF
P Switch      E
*-----*
* Sección de datos de compilación para Tabla.
*-----*
**
aaa
bbb
ccc

```

Figura 121. Fuente para el módulo DBGEX (Pieza 4 de 4). DBGEX es el módulo principal del programa DEBUGEX.

```

*-----*
* RPGPGM - Programa llamado por DEBUGEX para ilustrar las
*          funciones de STEP del depurador del fuente ILE.
*-----*
* Este programa recibe un parámetro InputParm desde DEBUGEX,
* lo visualiza y lo devuelve.
*-----*
D InputParm  S          4P 3
C  *ENTRY    PLIST
C          PARM              InputParm
C  InputParm DSPLY
C          SETON                                LR

```

Figura 122. Fuente del programa OPM RPGPGM

Ejemplo de fuente para ejemplos de depuración

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
extern char EXPORTFLD[6];
char *c_proc(unsigned int size, char *inzval)
{
    char *ptr;
    ptr = malloc(size);
    memset(ptr, *inzval, size );
    printf("import string: %6s.\n",EXPORTFLD);
    return(ptr);
}
```

Figura 123. Fuente para el procedimiento C cproc. cproc es llamado por DBGEX.

Capítulo 13. Manejo de excepciones

Este capítulo describe cómo funciona el manejo de excepciones de ILE RPG y cómo utilizar:

- Manejadores de excepciones
- Manejadores específicos de ILE RPG
- Manejadores de condiciones ILE
- Manejadores de cancelación

ILE RPG da soporte a los siguientes tipos de manejadores de excepciones:

- Manejadores específicos de RPG; por ejemplo, la utilización de un indicador de error, de un ampliador de código de operación 'E' o de una subrutina de error *PSSR o INFSTR.
- Manejadores de condiciones ILE, que son manejadores de excepciones escritos por el usuario que se registran durante la ejecución utilizando la API enlazable de manejador de condiciones ILE CEEHDLR.
- Manejador de cancelación ILE, que puede utilizarse cuando un procedimiento termina anormalmente.

Muchos programas utilizan algún tipo de manejo planificado de excepciones porque con ello se reduce al mínimo el número de terminaciones anormales innecesarias (es decir, las asociadas con los errores de función). Los manejadores de excepciones ILE también permiten manejar excepciones en aplicaciones de lenguaje mixto de una manera coherente.

Pueden utilizarse los manejadores de excepciones RPG para manejar la mayoría de situaciones que pueden darse en una aplicación RPG. El nivel mínimo de manejo de excepciones que RPG proporciona es la utilización de indicadores de error en determinadas operaciones. Para saber cómo utilizarlos, lea los apartados siguientes de este capítulo:

- "Manejo de excepciones de ILE RPG" en la página 254
- "Especificación de indicadores de error o del ampliador de código de operación 'E'" en la página 261
- "Utilización de una subrutina de error de archivo (INFSTR)" en la página 265
- "Utilización de un grupo MONITOR" en la página 262
- "Utilización de una subrutina de error de programa" en la página 269

Como complemento, para saber cómo funciona el manejo de excepciones ILE, lea:

- "Visión general del manejo de excepciones" en la página 252 (para conceptos generales)
- "Utilización de manejadores específicos de RPG" en la página 261
- Las secciones relativas a manejo de errores de la publicación *ILE Concepts*.

Para obtener información sobre el manejo de excepciones y el ciclo RPG, consulte la publicación *ILE RPG Reference*.

Nota: En este manual, el término "manejo de excepciones" se utiliza para hacer referencia al manejo de excepciones propiamente dicho y al manejo de

errores. Sin embargo, por coherencia con otros términos de RPG, el término "error" se utiliza en el contexto de "indicador de error" y "subrutina de error".

Visión general del manejo de excepciones

El manejo de excepciones es el proceso de:

- Examinar un mensaje de excepción que se ha emitido como resultado de un error durante la ejecución
- Opcionalmente, modificar la excepción para mostrar que se ha recibido (es decir, que se ha manejado)
- Opcionalmente, efectuar la recuperación de la excepción transfiriendo la información de excepción a una parte del código para llevar a cabo las acciones necesarias.

Cuando se produce un error durante la ejecución, se genera un mensaje de excepción. Los mensajes de excepción tienen uno de los tipos siguientes, en función del error que se produzca:

- | | |
|----------------|--|
| *ESCAPE | Indica que se ha detectado un error grave. |
| *STATUS | Describe el estado del trabajo que está realizando un programa. |
| *NOTIFY | Describe una condición que requiere una acción correctiva o una respuesta por parte del programa que efectúa la llamada. |

Error de función

Indica que se ha producido una de las tres excepciones anteriores y que no se ha manejado.

Los mensajes de excepción están asociados con las entradas de la pila de llamadas. Cada entrada de la pila de llamadas está asociada a su vez con una lista de manejadores de excepciones definidos para dicha entrada. (Consulte el apartado "Pila de llamadas" en la página 133 para obtener una descripción más amplia de la pila de llamadas).

En la Figura 124 en la página 253 se muestra una pila de llamadas en la que un programa OPM llama a un programa ILE que consta de varios módulos y, por lo tanto, de varios procedimientos. Consulte esta figura a medida que lea la información que se facilita a continuación.

En general, cuando se produce una excepción, los manejadores asociados con la entrada de la pila de llamadas tienen la oportunidad de manejar la excepción. Si ninguno de los manejadores de la lista maneja la excepción, ésta se considerará como no manejada; en este punto se realizarán las siguientes acciones por omisión para la excepción no manejada:

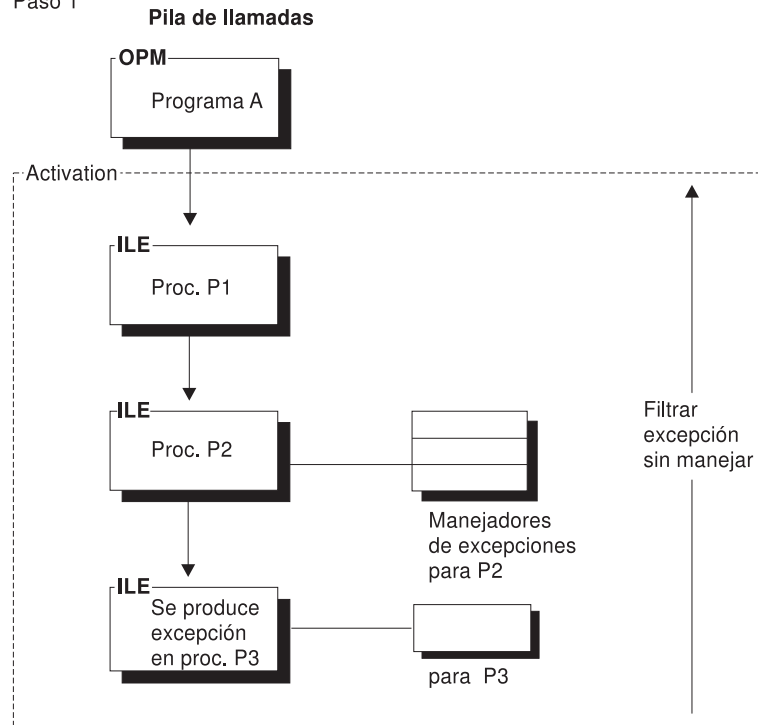
1. Si la excepción es un error de función, la entrada de la pila de llamadas se elimina de la pila.
2. La excepción se mueve (se filtra) a la entrada anterior de la pila de llamadas.
3. El proceso de manejo de excepciones comienza de nuevo para esta entrada de la pila de llamadas.

La acción de permitir que la entrada anterior de la pila de llamadas maneje una excepción recibe el nombre de **filtro**. El filtro continúa hasta que se maneja la excepción o hasta que se alcanza el límite de control. Un **límite de control** es una entrada de la pila de llamadas cuya entrada inmediatamente anterior se encuentra

Visión general del manejo de excepciones

en un grupo de activación distinto o es un programa OPM. En la Figura 124 , el Procedimiento P1 es el límite de control.

Paso 1



Paso 2

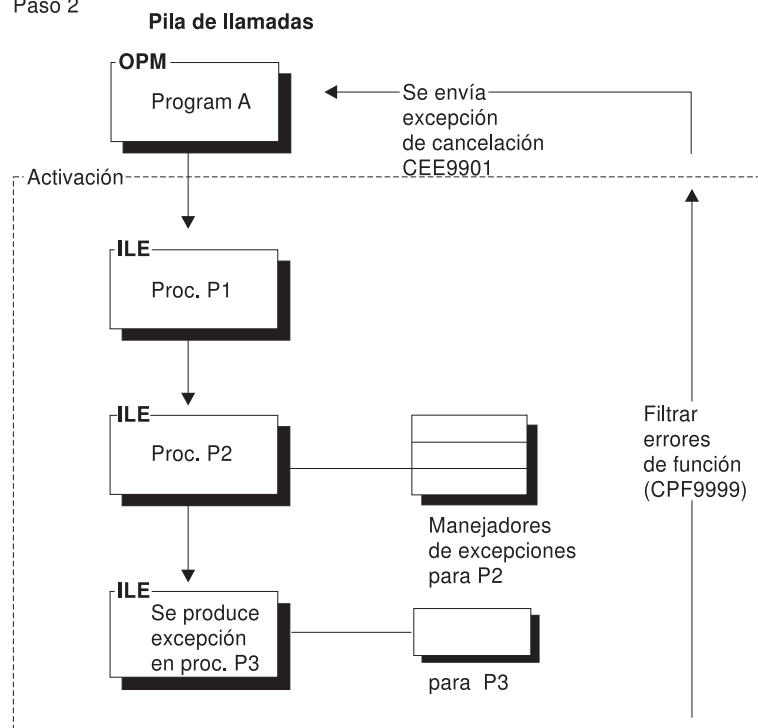


Figura 124. Pila de llamadas y filtro de mensajes de excepción

Visión general del manejo de excepciones

En OPM, el mensaje de excepción está asociado con el *programa* que está activo en la pila de llamadas. Si los manejadores de excepciones asociados no manejan la excepción, se envía un error de función a la misma entrada de la pila de llamadas que ha recibido la excepción. Si sigue sin ser manejada, la entrada se elimina y el error de función se filtra. El proceso se repite hasta que la excepción se maneja.

En ILE, un mensaje de excepción está asociado con el *procedimiento* que está activo en la pila de llamadas. Cuando se filtra la excepción, *no* se convierte en un error de función. Cada entrada de la pila de llamadas tiene la oportunidad de manejar la excepción original hasta que se alcanza el límite de control. Sólo entonces se convierte la excepción en un error de función; en este momento comienza de nuevo el proceso de la excepción empezando por el procedimiento que ha recibido la excepción. Esta vez, cada entrada de la pila de llamadas tiene la oportunidad de manejar el error de función. Si se ha alcanzado el límite de control y la excepción sigue sin ser manejada, se envía el mensaje de excepción de anomalía genérica CEE9901 al llamador del procedimiento en el límite de control. Además, se eliminarán las entradas de la pila de llamadas que no han manejado el mensaje.

Manejo de excepciones de ILE RPG

ILE RPG proporciona cuatro tipos de mecanismos de manejo de excepciones:

- Un manejador de indicador de error o de ampliador de código de operación 'E'
- Un grupo MONITOR
- Un manejador de subrutinas de error
- Un manejador de excepciones por omisión

RPG divide las excepciones en dos categorías: de programa y de archivo; con ello se determina a qué tipo de subrutina de error se llama. Algunos ejemplos de excepciones de programa son la división entre cero, el índice de matriz fuera de los límites o la raíz cuadrada (SQRT) de un número negativo. Como ejemplo de las excepciones de archivo puede citarse un tipo de registro no definido o un error de dispositivo.

Existen cinco formas distintas para indicar que RPG debe manejar una excepción: Puede:

1. Especificar un indicador de error en las posiciones 73 - 74 de las especificaciones de cálculo del código de operación adecuado.
2. Especificar el ampliador de código de operación 'E' para el código de operación adecuado.
3. Incluir el código que procesa la excepción dentro de un grupo MONITOR.
4. Codificar una subrutina de error de archivo, definida por la palabra clave INFSR en una especificación de descripción de archivo, para las excepciones de archivo. La subrutina de error de archivo sólo puede codificarse en la sección del fuente principal. No puede codificar una INFSR para un archivo que se utiliza en un subprocedimiento.
5. Codificar una subrutina de error de programa, denominada *PSSR, para las excepciones de programa. Tenga en cuenta que *PSSR es local con respecto al procedimiento en el que se codifica. Esto significa que *PSSR en un procedimiento principal manejará únicamente los errores del programa asociados con el procedimiento principal. Del mismo modo, *PSSR en un subprocedimiento sólo manejará los errores de dicho subprocedimiento.

Manejo de excepciones en un procedimiento principal

Cuando se produce una excepción en un procedimiento principal, ILE RPG efectúa lo siguiente:

1. Si en la especificación de cálculo hay un indicador de error y la excepción es una de las que se esperaba para esa operación:
 - a. El indicador se activa
 - b. La excepción se maneja
 - c. El control se reanuda con la próxima operación ILE RPG.
2. Si en la especificación de cálculo hay un ampliador de código de operación 'E' y la excepción es una de las que se esperaba para esa operación:
 - a. Se establecen los valores de retorno para las funciones incorporadas %STATUS y %ERROR.

Nota: %STATUS se establece cuando se produce una excepción, aunque no se haya especificado el ampliador 'E'.

- b. La excepción se maneja
 - c. El control se reanuda con la próxima operación ILE RPG.
3. Si no hay ningún indicador de error ni ampliador 'E' y el código que genera la excepción está en el bloque MONITOR de un grupo MONITOR, el control pasará a la sección de errores (on-error) del grupo MONITOR.
4. Si no hay ningún indicador de error ni ampliador 'E', ningún grupo MONITOR activo ha podido manejar la excepción, y
 - ha codificado una subrutina de error *PSSR y la excepción es de programa
 - o
 - ha codificado una subrutina de error INFSR para el archivo y la excepción es de E/S,

la excepción se manejará y el control se reanudará en la primera sentencia de la subrutina de error.

5. Si no se ha codificado ningún indicador de error, ampliador 'E' ni subrutina de error y ningún grupo MONITOR activo ha podido manejar la excepción, se llama al manejador de errores RPG por omisión.
 - Si la excepción *no* es un error de función, la excepción se filtrará.
 - Si la excepción es un error de función, se visualizará un mensaje de consulta. Si se elige la opción "G" o "R", el error de función se manejará y el control seguirá en el punto adecuado del procedimiento (*GETIN en el caso de 'G' o la misma especificación de cálculo que ha recibido la excepción en el caso de "R"). De lo contrario, el error de función se filtrará y el procedimiento terminará anormalmente.

Consulte el apartado "Excepciones no manejadas" en la página 257 para obtener una descripción completa del manejador RPG por omisión.

Manejo de excepciones en subprocedimientos

El manejo de excepciones en un subprocedimiento difiere de un procedimiento principal en lo siguiente:

- Dado que no puede codificar una subrutina INFSR, debe manejar los errores de archivo utilizando los indicadores de error, el ampliador de código de operación 'E' o un grupo MONITOR.
- No existe un manejador por omisión; en otras palabras, los usuarios no verán nunca un mensaje de consulta.

Visión general del manejo de excepciones

El manejo de excepciones en un subprocedimiento difiere de un procedimiento principal primordialmente en que para los subprocedimientos no se genera código de ciclo RPG. Como resultado no existe un manejador de excepciones por omisión para subprocedimientos y por lo tanto, las situaciones en las que se llamaría al manejador por omisión para un procedimiento principal corresponde a una finalización anormal del subprocedimiento. Esto significa que:

- El factor 2 de una operación ENDSR para una subrutina *PSSR de un subprocedimiento debe estar en blanco. Un factor 2 en blanco en un procedimiento principal daría como resultado que el control pasara al manejador por omisión. En un subprocedimiento, si se alcanza ENDSR, el subprocedimiento finalizará anormalmente y se señalará RNX9001 al llamador del subprocedimiento.
- Si no hay *PSSR y se produce un error de función, se elimina el procedimiento de la pila de llamadas y se filtra la excepción al llamador.
- Dado que en un subprocedimiento no se emite nunca un mensaje de consulta para un error, no tendrá acceso a la función de reintentar que hay disponible para algunos errores de E/S. Si espera errores de bloqueo de registro en un subprocedimiento, debe codificar un indicador de error o un ampliador 'E' y comprobar si el estado está relacionado con un registro que se está bloqueando.

Tenga en cuenta que PSDS e INFDS tienen ámbito de módulo. Tanto los procedimientos principales como los subprocedimientos pueden acceder a ellas.

CONSEJO

*PSSR es local con respecto al procedimiento en el que está codificado; por lo tanto, para tener una rutina de errores común, puede codificar un procedimiento de modo que maneje el error y llame al procedimiento desde cada *PSSR local.

Diferencias entre el manejo de excepciones de OPM e ILE RPG

En su mayor parte, el manejo de excepciones funciona del mismo modo en OPM RPG y en ILE RPG. La diferencia más importante reside en el área de las excepciones no manejadas.

En OPM, si se produce una excepción y no hay ningún manejador específico de RPG habilitado, se emite un mensaje de consulta. En ILE, esto sólo sucede si la excepción es un error de función. Si no lo es, la excepción se pasa al llamador del procedimiento o del programa, y las entradas superiores elegibles de la pila de llamadas tienen la oportunidad de manejar la excepción. Considere el ejemplo siguiente:

- PGM A llama a PGM B, que a su vez llama a PGM C.
- PGM B tiene un indicador de error codificado para la llamada.
- PGM C no tiene codificado ningún indicador de error ni subrutina de error *PSSR.
- PGM C produce una excepción.

En OPM, se emitiría un mensaje de error para PGM C. En ILE, la excepción se filtra a PGM B, dado que PGM C no la ha manejado. El indicador de error de PGM B se activa, lo que permite a PGM B manejar el error, y, en este proceso, PGM C termina anormalmente. No hay ningún mensaje de consulta.

Si PGM C tiene codificada una subrutina de error *PSSR, PGM C maneja la excepción y se ejecuta la subrutina de error tanto en OPM como en ILE.

Nota: Los mensajes de consulta emitidos por ILE RPG comienzan con el prefijo "RNQ", no con "RPG", como en OPM RPG

Existen ciertas diferencias de comportamiento para algunos errores determinados. Consulte el "Apéndice A. Diferencias de comportamiento entre OPM RPG/400 y ILE RPG para AS/400" en la página 423 para obtener más información.

Utilización de manejadores de excepciones

La planificación de la capacidad de manejo de excepciones de la aplicación significa tomar siguientes decisiones:

1. Decidir si utilizará el método de manejo de errores específico de RPG (es decir, indicador de error, ampliador 'E' o subrutina de error) o si escribirá una rutina de manejo de excepciones por separado que registrará mediante la API de ILE CEEHDLR. También puede elegir la utilización de ambos métodos.
2. Decidir la acción de recuperación, es decir, dónde continuará el programa el proceso si utiliza una rutina de manejo de excepciones por separado.

Además, tenga en cuenta lo siguiente al planificar los manejadores de excepciones:

- Prioridad de los manejadores
- Excepciones anidadas
- Acciones por omisión para las excepciones no manejadas
- Efecto del nivel de optimización

Prioridad del manejador de excepciones

La prioridad del manejador de excepciones cobra importancia si utiliza el manejo de errores específico del lenguaje y manejadores de condiciones ILE. En el caso de un procedimiento ILE RPG, los manejadores de excepciones tienen la prioridad siguiente:

1. Un manejador de indicador de error o de ampliador de código de operación 'E'
2. Grupo MONITOR
3. Manejador de condiciones ILE
4. Manejador de subrutinas de error de E/S (para errores de archivo) y Manejador de subrutinas de error de programa (para todos los demás errores)
5. Manejador por omisión RPG para excepciones no manejadas (únicamente para el procedimiento principal)

Excepciones anidadas

Las excepciones se pueden anidar. Una excepción anidada es aquella que se produce mientras se maneja otra excepción. Cuando esto sucede, el proceso de la primera excepción se suspende temporalmente. El manejo de excepciones comienza de nuevo con la excepción generada más recientemente.

Excepciones no manejadas

Una excepción no manejada es la que no ha sido manejada por un manejador de excepciones asociado con la entrada de la pila de llamadas que ha recibido la excepción en primer lugar. Cuando una excepción no está manejada, se produce una de las siguientes acciones:

Si el tipo de mensaje es un error de función (CPF9999) asociado con un procedimiento principal, el manejador RPG por omisión emitirá un mensaje de consulta describiendo la condición originadora.

Utilización de manejadores de excepciones

- Si elige la opción Volcar (D) o Cancelar (C), el procedimiento que recibió en primer lugar la excepción termina y el error de función se filtra al llamador.
- Si elige la opción Reintentar (R) u Obtener entrada (G), se maneja el error de función, el proceso de la excepción finaliza y el procedimiento reanuda el proceso en *GETIN (cuando se elige G) o en la operación de E/S en la que se produjo la excepción (cuando se elige R). Por ejemplo, cualquier operación de lectura se reintentará si la lectura ha resultado anómala a causa del bloqueo de registros.

En el caso de otros tipos de mensajes, la excepción se filtra hacia arriba por la pila de llamadas al llamador del procedimiento. Dicho procedimiento recibe la excepción y tiene la oportunidad de manejarla. Si no lo hace, la excepción se filtra hacia arriba por la pila de llamadas hasta llegar al límite de control, donde la excepción se convierte en un error de función, y el manejo de excepciones comienza tal como se ha descrito anteriormente.

Ejemplo de mensaje de escape no manejado

El siguiente ejemplo describe lo que sucede cuando se emite un mensaje de escape y el procedimiento que lo ha emitido no puede manejarlo. En este escenario se presupone lo siguiente:

1. Hay dos programas, PGM1 y PGM2, que se ejecutan en el mismo grupo de activación. Cada uno de ellos contiene un procedimiento, PRC1 y PRC2 respectivamente.
2. PRC1 llama a PGM2 de forma dinámica y PRC2 recibe el control.
3. El código de operación CALL en PRC1 tiene un indicador de error para la llamada.
4. No se ha codificado ningún manejador de excepciones de RPG en PRC2. Es decir, no hay ningún indicador de error codificado para la operación SUBST y tampoco hay subrutinas de error *PSSR.
5. PRC2 tiene una operación SUBST en la que la entrada del factor 1 es un número negativo.

Cuando PGM1 llama a PGM2 y se intenta la operación SUBST, se genera un mensaje de excepción RNX0100. La Figura 125 ilustra este escenario y los eventos que se producen.

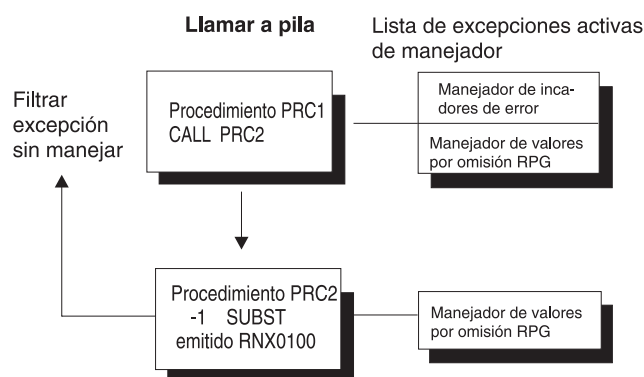


Figura 125. Escenario de mensaje de escape no manejado

Se produce lo siguiente:

Utilización de manejadores de excepciones

1. Puesto que no hay ningún indicador de error, ningún grupo MONITOR activo ni se ha codificado una subrutina de error *PSSR en la operación SUBST de PRC2, PRC2 no puede manejar el error de programa, por lo que quedará sin manejar.
2. Puesto que no se trata de un error de función, se filtra (asciende en la pila de llamadas) a PRC1.
3. PRC1 recibe (maneja) el mismo mensaje de excepción y activa el indicador de error en la operación CALL, lo que hace que PRC2 se termine.
4. El proceso continúa en PRC1 con la sentencia que sigue a la operación CALL.

Nota: Estos mismos eventos de manejo de excepciones descritos también se aplicarían a una llamada a procedimiento (operación CALLB).

Ejemplo de error de función no manejado

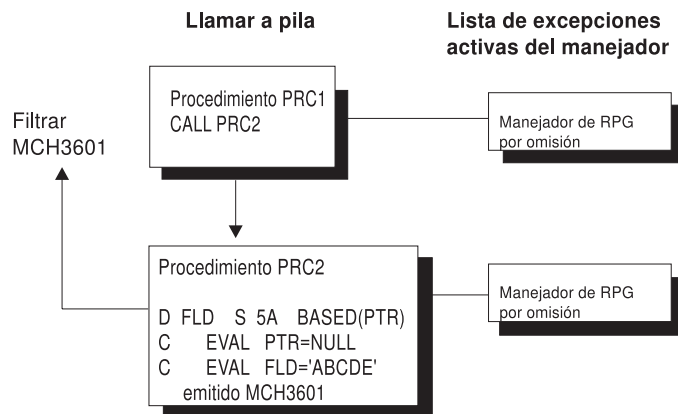
El siguiente escenario describe los eventos que se producen cuando se produce un error de función en un procedimiento principal y no se maneja. En este escenario se presupone lo siguiente:

1. Hay dos programas, PGM1 y PGM2, cada uno de los cuales contiene un procedimiento, PRC1 y PRC2 respectivamente.
2. PRC1 llama a PGM2 de forma dinámica y PRC2 recibe el control.
3. El código de operación CALL en PRC1 no tiene un indicador de error codificado.
4. No se ha codificado ningún manejador de excepciones de RPG en PRC2. Es decir, no hay ningún indicador de error, ningún grupo MONITOR ni tampoco hay subrutinas de error *PSSR.
5. PRC2 tiene un error de dirección de puntero.

Cuando PGM1 llama a PGM2, se produce un error de puntero porque el puntero de base está definido como nulo. Por lo tanto, se genera MCH1306. Se produce un error de función cuando PRC2 intenta filtrar la excepción más allá del límite de control. La Figura 126 en la página 260 ilustra este escenario y los eventos que se producen.

Utilización de manejadores de excepciones

PASO 1



PASO 2

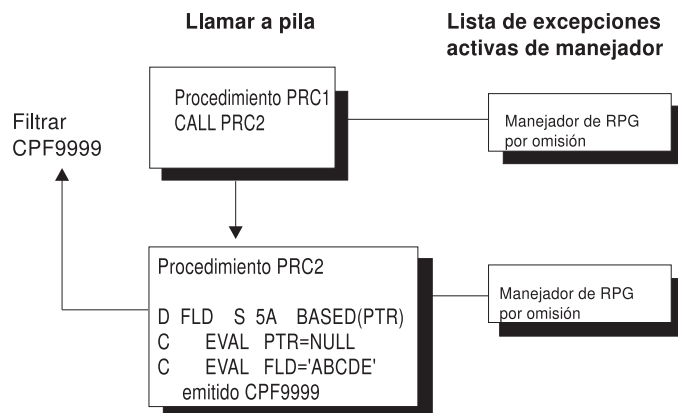


Figura 126. Escenario para error de función no manejado

Se produce lo siguiente:

1. Puesto que no hay manejadores de error en PRC2, PRC2 no puede manejar el error de función, por lo que quedará sin manejar.
2. Puesto que se trata de un error de función, se emite un mensaje de consulta en el que se describe la condición que lo ha originado.
3. Según la respuesta al mensaje de consulta, puede que PRC2 termine y la excepción se filtre a PRC1 (si la respuesta es "C") o que el proceso continúe en PRC2 (si la respuesta es "G").

Consideraciones sobre optimización

Mientras se ejecuta un programa optimizado con *FULL, el optimizador puede conservar los valores más utilizados en los registros de máquina y restaurarlos en el almacenamiento solamente en puntos definidos previamente durante el proceso normal del programa. El manejo de excepciones puede interrumpir este proceso normal y, por lo tanto, es posible que las variables de programa contenidas en los registros no se devuelvan a sus ubicaciones de almacenamiento asignadas.

Específicamente, las variables pueden no contener sus valores actuales si se produce una excepción y su recuperación se efectúa utilizando uno de los elementos siguientes:

- Grupo MONITOR
- Subrutina de error *PSSR
- Subrutina de error INFSR
- Manejador de excepción definido por usuario
- La opción Seguir ("G") desde un mensaje de consulta.
- La opción Reintentar ("R") desde un mensaje de consulta.

ILE RPG define automáticamente indicadores para que contengan sus valores actuales incluso con la optimización completa. Para garantizar que los campos o las estructuras de datos contengan los valores correctos (los actuales), especifique la palabra clave NOOPT en la especificación de definición adecuada.

Para obtener más información acerca de la palabra clave NOOPT, consulte la publicación *ILE RPG Reference*. Para obtener más información acerca de la optimización, consulte el apartado "Modificación del nivel de optimización" en la página 91.

Utilización de manejadores específicos de RPG

ILE RPG proporciona cuatro maneras de habilitar los manejadores específicos de HLL y de efectuar la recuperación de las excepciones:

1. indicadores de error o ampliador de código de operación 'E'
2. grupo MONITOR
3. subrutina de error INFSR
4. subrutina de error *PSSR

Puede obtener más información acerca del error que se haya producido codificando las estructuras de datos adecuadas y consultando los campos de estructuras de datos relacionados.

Si utiliza el ampliador 'E' en lugar de indicadores de error, el programa correspondiente y la información de errores de archivo pueden obtenerse utilizando las funciones incorporadas %STATUS y %ERROR.

En este apartado se proporcionan algunos ejemplos de la utilización de cada una de estas construcciones RPG. La publicación *ILE RPG Reference* proporciona más información acerca de las subrutinas de error *PSSR e INFSR, del código de operación EXSR y de las estructuras de datos INFDS y PSDS.

Especificación de indicadores de error o del ampliador de código de operación 'E'

Los códigos de operación que permiten un indicador de error también permiten el ampliador de código de operación 'E'. La operación CALLP también permite el ampliador 'E', aunque no permite un indicador de error. Esta proporciona dos métodos de manejo de errores ILE RPG que son esencialmente el mismo. Puede utilizarse un indicador de error o el ampliador 'E' pueden utilizarse para manejar la excepción para el mismo código de operación, pero no ambos.

Nota: Si se codifica un indicador de error o el ampliador 'E' en una operación, pero el error que se produce no está relacionado con dicha operación (por ejemplo, un error de índice de matriz en una operación CHAIN), se pasa por alto cualquier indicador de error o ampliador 'E'. El error se trata como cualquier otro error de programa.

Para habilitar el manejador de indicadores de error de RPG, especifique un indicador de error en las posiciones 73 y 74 para los códigos de operación listados en la Tabla 16 (excepto para CALLP). Si se produce una excepción en la operación, el indicador se activa, se actualiza la estructura de datos adecuada (PSDS o INFDS) y el control vuelve a la siguiente instrucción de la secuencia. A continuación puede probar el indicador para determinar la acción que se ha de realizar.

Para habilitar el manejador de ampliador de código de operación 'E', especifique una 'E' (o 'e') con cualquiera de los códigos de operación de la Tabla 16. La codificación del ampliador 'E' afecta al valor devuelto por las funciones incorporadas %ERROR y %STATUS para las excepciones. Antes de que empiece la operación, el valor devuelto por estas funciones incorporadas se establece en cero. Si se produce una excepción en la operación, los valores de retorno para estas funciones incorporadas se actualizan consecuentemente, se actualiza la estructura de datos adecuada (PSDS o INFDS) y el control vuelve a la siguiente instrucción de la secuencia. A continuación, puede utilizar estas funciones incorporadas para probar los valores devueltos y determinar la acción que debe realizarse.

Tabla 16. Códigos de operación que permiten el uso del ampliador 'E' o de un indicador de error en las posiciones 73 y 74

ACQ (e)	ADDDUR (e)	ALLOC (e)	CALL (e)
CALLB(d e)	CALLP (e m/r) ¹	CHAIN (e n)	CHECK (e)
CHECKR (e)	CLOSE (e)	COMMIT (e)	DEALLOC(e/n)
DELETE (e)	DSPLY (e)	EXFMT (e)	EXTRCT (e)
FEOD (e)	IN (e)	NEXT (e)	OCCUR (e)
OPEN (e)	OUT (e)	POST (e)	READ (e n)
READC (e)	READE (e n)	READP (e n)	READPE (e n)
REALLOC (e)	REL (e)	RESET (e)	ROLBK (e)
SCAN (e)	SETGT (e)	SETLL (e)	SUBDUR (e)
SUBST (e p)	TEST (e d/t/z)	UNLOCK (e)	UPDATE (e)
WRITE (e)	XLATE (e p)		
Notas: 1. CALLP (e m/r) es un código de operación de Factor-2 ampliado y no puede tener un indicador de errores. Sin embargo, el estado del programa y las condiciones de error pueden determinarse especificando el ampliador 'e' con este código de operación.			

Si especifica un indicador de error o un ampliador 'E' en un código de operación, puede llamar explícitamente a una subrutina de error de archivo (INFSR) o a una subrutina de error de programa (*PSSR) con la operación EXSR. Si la operación EXSR llama explícitamente a INFSR o a *PSSR y el factor 2 de la operación ENDSR está en blanco o el campo especificado tiene un valor de blanco, el control vuelve a la instrucción que sigue a la operación EXSR en la secuencia.

Utilización de un grupo MONITOR

Un grupo MONITOR realiza un manejo de errores condicional basado en el código de estado. Si se produce un error, el control pasa al grupo ON-ERROR adecuado dentro del grupo MONITOR.

Si todas las sentencias del bloque MONITOR se procesan sin errores, el control pasa a la sentencia que sigue a la sentencia ENDMON.

El grupo MONITOR puede especificarse en cualquier punto de los cálculos. Puede estar anidado dentro de grupos IF, DO, SELECT o de otro grupo MONITOR. Los grupos IF, DO y SELECT pueden estar anidados dentro de grupos MONITOR.

Si un grupo MONITOR está anidado dentro de otro grupo MONITOR, el grupo más interno se considera el primero cuando se produce un error. Si ese grupo MONITOR no maneja la condición de error, se tiene en cuenta el siguiente grupo.

Pueden utilizarse indicadores de nivel en la operación MONITOR para indicar que el grupo MONITOR forma parte de cálculos totales. A efectos de documentación, también puede especificar un indicador de nivel en una operación ON-ERROR o ENDMON, pero este indicador de nivel se pasará por alto.

Pueden utilizarse indicadores de condicionamiento en la sentencia MONITOR. Si no se satisfacen, el control pasa inmediatamente a la sentencia que sigue a la sentencia ENDMON del grupo MONITOR. Los indicadores de condicionamiento no pueden utilizarse en operaciones ON-ERROR individualmente.

Si un bloque MONITOR contiene una llamada a un subprocedimiento y éste tiene un error, el manejo de errores del subprocedimiento tendrá preferencia. Por ejemplo, si el subprocedimiento tiene una subrutina *PSSR, se llamará a ésta. El grupo MONITOR que contiene la llamada sólo se tendrá en cuenta si el subprocedimiento no puede manejar el error y la llamada falla con el estado de error en llamada 00202.

El grupo MONITOR no maneja errores que se producen en una subrutina. Si la subrutina contiene sus propios grupos MONITOR, éstos se consideran los primeros.

No se permiten las operaciones de ramificación dentro de un bloque MONITOR, pero sí están permitidas en un bloque ON-ERROR.

Una operación LEAVE o ITER dentro de un bloque MONITOR se aplica a cualquier grupo DO activo que contenga el bloque MONITOR. Una operación LEAVESR o RETURN dentro de un bloque MONITOR se aplica a cualquier subrutina, subprocedimiento o procedimiento que contenga el bloque MONITOR.

En cada sentencia ON-ERROR, debe especificar qué condiciones de error maneja el grupo ON-ERROR. Puede especificar cualquier combinación de los siguientes valores, separados mediante dos puntos:

<i>nnnnn</i>	Un código de estado
*PROGRAM	Maneja todos los códigos de estado de error de programa, de 00100 a 00999
*FILE	Maneja todos los códigos de estado de error de archivo, de 01000 a 09999
*ALL	Maneja códigos de error tanto de programa como de archivo, de 00100 a 09999. Es el valor por omisión.

Los códigos de estado situados fuera del rango de 00100 a 09999, por ejemplo los códigos de 0 a 99, no se supervisan. No puede especificar estos valores para un grupo ON-ERROR. Tampoco puede especificar ningún código de estado que no sea válido para la versión determinada del compilador que se utiliza.

Si más de un grupo ON-ERROR cubre el mismo código de estado, sólo se utiliza el primero. Por esta razón, debe especificar los valores especiales, como por ejemplo *ALL, después de los códigos de estado específicos.

Los errores que se producen dentro de un grupo ON-ERROR no son manejados por el grupo MONITOR. Para manejar los errores, puede especificar un grupo MONITOR dentro de un grupo ON-ERROR.

```
* El bloque MONITOR consta de la sentencia READ y del
* grupo IF.
* - El primer bloque ON-ERROR maneja el estado 1211, que
*   se emite para la operación READ si el archivo no está abierto.
* - El segundo bloque ON-ERROR maneja todos los demás errores de archivo.
* - El tercer bloque ON-ERROR maneja el código de estado de la operación
*   de serie 00100 y el código de estado de índice de matriz 00121.
* - El cuarto bloque ON-ERROR (que podría haber tenido un factor 2
*   de *ALL) maneja los errores no manejados por las operaciones
*   ON-ERROR específicas.
*
* Si no se produce ningún error en el bloque MONITOR, el control
* pasa de ENDIF a ENDMON.
C          MONITOR
C          READ      FILE1
C          IF        NOT %EOF
C          EVAL      Line = %SUBST(Line(i) :
C                      %SCAN('***': Line(i)) + 1)
C          ENDIF
C          ON-ERROR  1211
C          ... handle file-not-open
C          ON-ERROR  *FILE
C          ... handle other file errors
C          ON-ERROR  00100 : 00121
C          ... handle string error and array-index error
C          ON-ERROR
C          ... handle all other errors
C          ENDMON
```

Figura 127. Operación MONITOR

Utilización de una subrutina de error

Cuando escribe una subrutina de error, está haciendo dos cosas:

1. Habilitar el manejador de errores de subrutina RPG
El manejador de errores de subrutina manejará la excepción y pasará el control a la subrutina.
2. De forma opcional, especificar una acción de recuperación.
Puede utilizar la subrutina de error para efectuar acciones específicas en función del error que se haya producido o puede tener una acción genérica (por ejemplo, emitir un mensaje de consulta para todos los errores).

Las subrutinas de error están sujetas a las siguientes consideraciones:

- Puede llamar explícitamente a una subrutina de error especificando el nombre de la subrutina en el factor 2 de la operación EXSR.
- Puede controlar el punto en el que el proceso se reanuda en un procedimiento principal especificando un valor en el factor 2 de la operación ENDSR de la subrutina. En un subprocedimiento, el factor 2 de ENDSR debe estar en blanco. Utilice una operación GOTO o RETURN antes de la operación ENDSR para evitar que el subprocedimiento finalice anormalmente.

- Si se llama a una subrutina de error, el manejador de subrutinas de error RPG ya ha manejado la excepción. Por lo tanto, la llamada a la subrutina de error refleja que se ha vuelto al proceso del programa. Si se produce una excepción mientras la subrutina se está ejecutando, se llama de nuevo a la subrutina. El procedimiento entrará en un bucle a menos que codifique la subrutina de tal modo que se evite este problema.

Para ver cómo se codifica una subrutina de error para evitar un bucle de este tipo, consulte el apartado “Evitar un bucle en una subrutina de error” en la página 272.

Utilización de una subrutina de error de archivo (INFSR)

Para manejar un error o una excepción de archivo en un procedimiento principal, puede escribir una subrutina de error de archivo (INFSR). Cuando se produce una excepción de archivo:

1. La INFDS se actualiza.
2. Una subrutina de error de archivo (INFSR) recibe el control si la excepción se produce:
 - En una operación de archivo implícita (primaria o secundaria)
 - En una operación de archivo explícita que no tiene un indicador especificado en las posiciones 73 y 74.

Una subrutina de error de archivo puede manejar errores en más de un archivo.

Se aplican las restricciones siguientes:

- Si se produce una excepción de archivo durante el inicio o el fin de un programa (por ejemplo, en una apertura implícita al principio del ciclo), el control pasa al manejador de excepciones por omisión de ILE RPG y no al manejador de subrutinas de error. Por consiguiente, la subrutina de error de archivo no se procesará.
- Si se produce un error que no está relacionado con la operación (por ejemplo, un error de índice de matriz en una operación CHAIN), se pasará por alto cualquier subrutina de error INFSR. El error se trata como cualquier otro error de programa.
- INFSR no puede manejar errores en un archivo utilizado por un subprocedimiento.

Para añadir una subrutina de error de archivo al programa, siga los pasos siguientes:

1. Escriba el nombre de la subrutina después de la palabra clave INFSR en una especificación de descripción de archivo. El nombre de la subrutina puede ser *PSSR, que indica que la subrutina de error de programa recibe el control de la excepción en este archivo.
2. Opcionalmente, identifique la estructura de datos de información de archivo en una especificación de descripción de archivo mediante la palabra clave INFDS.
3. Entre una operación BEGSR en la que la entrada del factor 1 contenga el mismo nombre de subrutina que el especificado para la palabra clave INFSR.
4. Identifique un punto de retorno, si lo hay, y codifíquelo en la operación ENDSR de la subrutina. Para obtener una descripción de las entradas válidas del factor 2, consulte el apartado “Especificación de un punto de retorno en la operación ENDSR” en la página 274.
5. Codifique el resto de la subrutina de error de archivo. Aunque puede utilizarse cualquiera de las operaciones del compilador ILE RPG en la subrutina de error de archivo, no es recomendable la utilización de operaciones de E/S en el

mismo archivo en el que se ha producido el error. La operación ENDSR debe ser la última especificación para la subrutina de error de archivo.

En la Figura 128 en la página 267 se muestra un ejemplo de manejo de excepciones utilizando una subrutina de error INFSR. El programa ACTTRANS es un sencillo programa de actualización de inventario. Utiliza el archivo de transacciones TRANSACC para actualizar el archivo maestro de inventario MASPRD. Si se produce un error de E/S, se llama a la subrutina de error INFSR. Si se trata de un error de bloqueo de registro, el registro se graba en un archivo de anotaciones. De lo contrario, se emite un mensaje de consulta.

Observe que la especificación de archivo correspondiente a PRDMAS identifica la INFDS y la INFSR que se asociará con dicho archivo.

Para cada registro del archivo TRANSACC, se realiza lo siguiente:

1. Se localiza el registro adecuado del archivo maestro de productos utilizando el número de producto de la transacción.
2. Si se encuentra el registro, se actualiza la cantidad del inventario.
3. Si se produce un error en la operación UPDATE, el control se pasa a la subrutina de error INFSR.
4. Si el registro no se encuentra, el número de producto se anota en un informe de errores.

```

*=====
* TRNSUPDT: Programa sencillo de actualización de inventario.      *
* El archivo de transacción (TRANSACT) se procesa consecutivamente*
* El número de producto de la transacción se utiliza como clave   *
* para acceder al archivo maestro (PRDMAS) de forma aleatoria.     *
* 1. Si se encuentra el registro, se actualiza la cantidad         *
* del inventario.                                                  *
* 2. Si no se encuentra el registro, se imprimirá un error en      *
* un informe.                                                       *
* 3. Si el registro está bloqueado actualmente, la transacción     *
* se grabará en un archivo de cartera de pedidos de               *
* transacción que se procesará más tarde.                           *
* 4. Cualquier otro error inesperado provocará un mensaje          *
* de error de ejecución.                                            *
*=====
*-----
* Definir los archivos:                                           *
* 1) PRDMAS - Archivo maestro de productos                         *
* 2) TRANSACT - Archivo de transacción                             *
* 3) TRNBACKLG - Archivo de cartera de pedidos de transacción *
* 2) PRINT - Informe de errores                                    *
*-----
FPRDMAS    UF  E          K DISK
F
F          INFSR(PrdInfsr)
F          INFDS(PrdInfds)
FTRANSACT  IP  E          DISK
FTRNBACKLG 0  E          DISK
FPRINT     0  F  80       PRINTER
*-----
* Definir la estructura de datos de información de archivo para  *
* el archivo PRDMAS. El campo *STATUS se usa para determinar la  *
* acción a realizar.                                             *
*-----
D PrdInfds      DS
D PrdStatus     *STATUS
*-----
* Lista de excepciones esperadas.                                *
*-----
D ErrRecLock    C          CONST(1218)

```

Figura 128. Ejemplo de manejo de excepciones de archivo (Pieza 1 de 2)

```

*-----*
* Acceder al archivo maestro de productos utilizando el número *
* de producto de transacción. *
*-----*
C      TRNPRDNO      CHAIN      PRDREC      10
*-----*
* Si se encuentra el registro, se actualiza la cantidad en el *
* archivo maestro. *
*-----*
C      IF      NOT *IN10
C      SUB      TRNQTY      PRDQTY
C      UPDATE      PRDREC
*-----*
* Si no se encuentra el registro, se escribe en el informe de *
* errores. *
*-----*
C      ELSE
C      EXCEPT      NOTFOUND
C      ENDIF
C      SETON      LR
*-----*
* Rutina de manejo de errores. *
*-----*
C      PrdInfsr      BEGSR
*-----*
* Si el registro maestro está bloqueado actualmente, se graba el *
* registro de transacción en el archivo de cartera de pedidos y *
* se salta a la siguiente transacción. *
*-----*
C      PrdStatus      DSPLY
C      IF      (PrdStatus = ErrRecLock)
C      WRITE      TRNBREC
C      MOVE      '*GETIN'      ReturnPt      6
*-----*
* Si se produce un error inesperado, se emite un mensaje de *
* consulta. *
*-----*
C      ELSE
C      MOVE      *BLANK      ReturnPt
C      ENDIF
C      ENDSR      ReturnPt
*-----*
* Formato de informe de errores. *
*-----*
OPRINT      E      NOTFOUND
O      TRNPRDNO
O      29 'NOT IN PRDMAS FILE'

```

Figura 128. Ejemplo de manejo de excepciones de archivo (Pieza 2 de 2)

Cuando el control se pasa a la subrutina de error, sucede lo siguiente:

- Si el error se debe a un bloqueo de registro, el registro se graba en un archivo de cartera de clientes y el control vuelve a la parte principal con la siguiente transacción (con *GETIN como punto de retorno).
- Si el error tiene otra causa, se mueven caracteres blancos a ReturnPt. De este modo el manejador por omisión de RPG recibirá el control. La acción de recuperación en este punto dependerá de la naturaleza del error.

Observe que la comprobación de la existencia de un error de bloqueo de registro se efectúa comparando el subcampo *STATUS de la INFDS para MASPRD con el campo BloqRegErr que está definido con el valor del código de estado del bloqueo

de registros. La subrutina INFSR puede ampliarse para manejar otros tipos de errores de E/S definiendo otros errores, comprobándolos y llevando a cabo la acción adecuada.

Utilización de una subrutina de error de programa

Para manejar un error o una excepción de programa, puede escribir una subrutina de error de programa (*PSSR). Cuando se produce un error de programa:

1. Se actualiza la estructura de datos de estado del programa.
2. Si *no* se ha especificado un indicador en las posiciones 73 y 74 para el código de operación, el error se maneja y el control se transfiere a la subrutina *PSSR.
Puede transferir el control explícitamente a una subrutina de error de programa después de un error de archivo especificando *PSSR después de la palabra clave INFSR en las especificaciones de descripción de archivo.

Puede codificar *PSSR para cualquier procedimiento del módulo o para todos ellos. Cada *PSSR es local con respecto al procedimiento en el que está codificado.

Para añadir una subrutina de error *PSSR al programa, realice los pasos siguientes:

1. Opcionalmente, identifique la estructura de datos de estado de programa (PSDS) especificando una S en la posición 23 de la especificación de definición.
2. Entre una operación BEGSR con la entrada *PSSR en el factor 1.
3. Identifique un punto de retorno, si lo hay, y codifíquelo en la operación ENDSR de la subrutina. Para subprocedimientos, el factor 2 debe estar en blanco. Para obtener una descripción de las entradas válidas del factor 2, consulte el apartado “Especificación de un punto de retorno en la operación ENDSR” en la página 274.
4. Codifique el resto de la subrutina de error de programa. En la subrutina de error de programa puede utilizarse cualquiera de las operaciones del compilador ILE RPG. La operación ENDSR debe ser la última especificación de la subrutina de error de programa.

La Figura 129 en la página 270 muestra un ejemplo de una subrutina de error de programa en un procedimiento principal.

```

*-----*
* Definir partes relevantes de estructura de datos de estado de *
* programa. *
*-----*
D Psds          SDS
D Loc           *ROUTINE
D Err           *STATUS
D Parms         *PARMS
D Name          *PROC
*-----*
* AQUÍ SE ESPECIFICA EL CUERPO DEL CÓDIGO
* Se produce un error si se efectúa la división por cero.

* El control se transfiere a la subrutina *PSSR.
*-----*
* *PSSR: Subrutina de error para el procedimiento principal. Se
* busca un error de división por cero, comprobando si el
* estado es 102. Si lo es, se añade 1 al divisor y se
* continúa moviendo *GETIN a ReturnPt.
*-----*
C      *PSSR      BEGSR
C      IF          Err = 102
C      ADD         1      Divisor
C      MOVE        '*GETIN' ReturnPt      6
*-----*
* Se ha producido un error inesperado, y por tanto se mueve
* *CANCL a ReturnPt para finalizar el procedimiento.
*-----*
C      ELSE
C      MOVE        '*CANCL' ReturnPt
C      ENDIF
C      ENDSR      ReturnPt

```

Figura 129. Ejemplo de subrutina *PSSR en el procedimiento principal

La estructura de datos de estado del programa se define en las especificaciones de definición. Se especifican los subcampos predefinidos *STATUS, *ROUTINE, *PARMS y *PROGRAM y se asignan nombres a los subcampos.

La subrutina de error *PSSR se codifica en las especificaciones de cálculo. Si se produce un error de programa, ILE RPG pasa el control a la subrutina de error *PSSR. La subrutina comprueba si la excepción se ha producido a causa de una operación de división en la cual el divisor sea cero. En caso afirmativo, se añade un 1 al divisor (Divisor) y el literal '*DETC' se mueve al campo ReturnPt para indicar que el programa debe reanudar el proceso al principio de la rutina de cálculo de detalle.

Si la excepción no era una división por 0, el literal '*CANCL' se mueve al campo ReturnPt y el procedimiento finaliza.

La Figura 130 en la página 271 y la Figura 131 en la página 272 muestran cómo puede codificar subrutinas de error de programa similares en un subprocedimiento. En un ejemplo, codifica una operación GOTO y en la otra codifica una operación RETURN.


```

*-----*
* Inicio de definición de subprocedimiento
*-----*
P SubProc      B
D SubProc      PI          5P 0
...
*-----*
* Aquí se especifica el cuerpo del código, incluyendo el código *
* de recuperación.
*-----*
C      TryAgain    TAG
C      X           DIV      Divisor      Result
C                      Return    Result
*-----*
* Se produce un error si se efectúa la división por cero.

* El control se transfiere a la subrutina *PSSR.
C      *PSSR      BEGSR
*-----*
* Si es un error de división por cero, se añade 1 al divisor
* y se intenta de nuevo
*-----*
C                      IF      Err = 102
C                      ADD      1      Divisor
C                      GOTO     TryAgain
C                      ENDIF
*-----*
* Si el control alcanza ENDSR, el procedimiento fallará
*-----*
C                      ENDSR
P                      E

```

Figura 130. Ejemplo de subrutina *PSSR de subprocedimiento con GOTO

```

*-----*
* Inicio de definición de subprocedimiento
*-----*
P SubProc      B
D SubProc      PI          5P 0
...
*-----*
* Aquí se especifica el cuerpo del código, incluyendo la
* operación de división.
*-----*
C      X      DIV      Divisor      Result
C      Return      Result
*-----*
* Se produce un error si se efectúa la división por cero.

* El control se transfiere a la subrutina *PSSR.
C      *PSSR      BEGSR
*-----*
* Si es un error de división por cero, se devuelve 0 desde el
* subprocedimiento
*-----*
C      IF      Err = 102
C      RETURN      0
C      ENDIF
*-----*
* Si el control alcanza ENDSR, el procedimiento fallará
*-----*
C      ENDSR
P      E

```

Figura 131. Ejemplo de subrutina *PSSR de subprocedimiento con RETURN

Evitar un bucle en una subrutina de error

En los ejemplos anteriores, es poco probable que se produzca un error en la *PSSR y se origine un bucle. Sin embargo, según cómo se haya escrito la subrutina *PSSR, pueden aparecer bucles si se produce una excepción al procesar la *PSSR.

Una manera de evitar los bucles es estableciendo un conmutador de primera vez en la subrutina. Si no es la primera vez que se pasa por la subrutina, puede especificar un punto de retorno adecuado, como *CANCL, para la entrada del factor 2 de la operación ENDSR.

En la Figura 132 en la página 273 se muestra el programa NOLOOP, diseñado para generar excepciones con el fin de mostrar cómo se pueden evitar los bucles en una subrutina *PSSR. El programa genera dos veces una excepción:

1. En la parte principal del código, para transferir el control a la *PSSR
2. En la *PSSR, para provocar un bucle si se desea.

```

=====
* NOLLOOP: Muestra cómo evitar recursión en una subrutina *PSSR. *
=====
*-----*
* Matriz que se utilizará para causar un error *
*-----*
D Arr1          S          10A  DIM(5)
*-----*
* Generar un error de matriz fuera de límites para pasar el *
* control a *PSSR. *
*-----*
C          Z-ADD      -1      Neg1          5 0
C          MOVE      Arr1(Neg1)  Arr1(Neg1)
C          MOVE      *ON        *INLR
=====
* *PSSR: Subrutina de error para el procedimiento. Se utiliza la *
* variable InPssr para detectar la recursión en PSSR. *
* Si se detecta recursión, se cancela (*CANCL) *
* el procedimiento. *
=====
C      *PSSR      BEGSR
C          IF      InPssr = 1
C          MOVE    '*CANCL'      ReturnPt      6
C          Z-ADD    0              InPssr      1 0
C          ELSE
C          Z-ADD    1              InPssr
*
*      Ahora se genera otro error en PSSR para ver cómo la *
*      subrutina cancela el procedimiento. *
*
C          MOVE    Arr1(Neg1)  Arr1(Neg1)
*
*      Observe que las dos operaciones siguientes no se *
*      procesarán si Neg1 sigue siendo negativo. *
*
C          MOVE    '*GETIN'      ReturnPt
C          Z-ADD    0              InPssr
C          ENDIF
C          ENDSR      ReturnPt

```

Figura 132. Evitar un bucle en una subrutina de error

Para crear el programa y empezar a depurarlo utilizando el fuente de la Figura 132, escriba:

```

CRTBNDRPG PGM(MYLIB/NOLLOOP) DBGVIEW(*SOURCE)
STRDBG PGM(MYLIB/NOLLOOP)

```

Establezca un punto de interrupción en la línea de BEGSR de la subrutina *PSSR para que pueda seguir los pasos de la subrutina *PSSR.

Cuando llame al programa, sucederá lo siguiente:

1. Se produce una excepción cuando el programa intenta realizar una operación MOVE en una matriz utilizando un índice negativo. El control se pasa a la *PSSR.
2. Puesto que es la primera vez que se pasa por la subrutina *PSSR, la variable In_Pssr no está activada todavía. Para evitar que se produzca un bucle más adelante, la variable In_Pssr se activa.
3. El proceso continúa en la *PSSR con la operación MOVE después de ELSE. De nuevo, se produce una excepción, por lo que el proceso de la *PSSR comienza otra vez.

4. Esta vez, la variable In_Pssr ya se ha establecido en 1. Puesto que esto indica que la subrutina ha entrado en un bucle, el procedimiento se cancela estableciendo el campo ReturnPt en *CANCL.
5. La operación ENDSR recibe el control y el procedimiento se cancela.

El método utilizado aquí para evitar los bucles también se puede emplear en una subrutina de error INFSR.

Especificación de un punto de retorno en la operación ENDSR

Al utilizar una subrutina de error INFSR o *PSSR en un procedimiento principal, puede indicar el punto de retorno en el que el programa reanudará el proceso entrando uno de estos valores como entrada del factor 2 de la sentencia ENDSR. La entrada debe ser un literal, contante con nombre, elemento de matriz, nombre de tabla o campo de caracteres de seis posiciones cuyo valor especifique uno de los puntos de retorno siguientes.

Nota: Si los puntos de retorno se especifican como literales, deben ir entre apóstrofes y deben entrarse en mayúsculas (por ejemplo, *DETL, no *detl). Si se especifican en campos o en elementos de matriz, el valor debe ajustarse por la izquierda.

*DETL	Continuar al principio de las líneas de detalle.
*GETIN	Continuar en la rutina de obtención de registro de entrada.
*TOTC	Continuar al principio de los cálculos de totales.
*TOTL	Continuar al principio de las líneas de totales.
*OFL	continuar al principio de las líneas de desbordamiento.
*DETC	Continuar al principio de los cálculos de detalle.
*CANCL	Cancelar el proceso del programa.
Blancos	Devolver el control al manejador de excepciones por omisión de ILE RPG. Esto sucederá si el factor 2 tiene un valor de blancos y cuando el factor 2 no se especifica. Si la operación EXSR ha llamado a la subrutina y el factor 2 está en blanco, el control se devuelve a la siguiente instrucción de la secuencia.

Después de que la operación ENDSR de la subrutina INFSR o *PSSR se haya ejecutado, el compilador de ILE RPG restablece el campo o el elemento de matriz especificado en el factor 2 a blancos. Puesto que el factor 2 se establece a blancos, puede especificar el punto de retorno en la subrutina más adecuada a la excepción que se haya producido.

Si este campo contiene blancos al final de la subrutina, el manejador de excepciones por omisión de ILE RPG recibe el control después de la ejecución de la subrutina, a menos que la operación EXSR haya llamado a la subrutina INFSR o *PSSR. Si la operación EXSR ha llamado a la subrutina y el factor 2 de la operación ENDSR está en blanco, el control vuelve a la siguiente instrucción de la secuencia que sigue a la operación EXSR.

Nota: No puede especificar una entrada de factor 2 para ENDSR en un subprocedimiento. Si desea reanudar el proceso del subprocedimiento, ha de utilizar una operación GOTO a un TAG del cuerpo del subprocedimiento. Como alternativa, puede codificar una operación RETURN en *PSSR. El subprocedimiento volverá al llamador.

Manejadores de condiciones ILE

Los **manejadores de condiciones ILE** son manejadores de excepciones que se registran durante la ejecución mediante la API enlazable Registrar Manejador de Condiciones ILE (CEEHDLR). Se utilizan para manejar, filtrar o promocionar excepciones. Las excepciones se presentan a los manejadores de condiciones en forma de condiciones ILE. Puede registrar más de un manejador de condiciones ILE. Los manejadores de condiciones ILE pueden desregistrarse llamando a la API enlazable Desregistrar Manejador de Condiciones ILE (CEEHDLU).

Existen varios motivos por los que puede querer utilizar un manejador de condiciones ILE:

- Puede eludir el manejo específico del lenguaje manejando la excepción en su propio manejador.
Ello le permite proporcionar el mismo mecanismo de manejo de excepciones en una aplicación con módulos en diferentes HLL ILE.
- Puede utilizar esta API para delimitar el ámbito del manejo de excepciones a una entrada de la pila de llamadas.
El ámbito de la API enlazable CEEHDLR de ILE es la invocación que la contiene. Permanecerá en vigor hasta que la elimine del registro o hasta que el procedimiento vuelva.

Nota: Cualquier llamada a la API CEEHDLR desde un cálculo de detalle, de totales o de subrutina activará el manejador de condiciones para todo el procedimiento, incluidas las operaciones de entrada, cálculo y salida. Sin embargo, no afectará a los subprocedimientos, ni un subprocedimiento que llame a CEEHDLR afectará al procedimiento principal.

Si se llama de modo recurrente a un subprocedimiento, sólo resulta afectada la invocación que llama a CEEHDLR. Si desea que el manejador de condiciones esté activo para cada invocación, cada invocación debe llamar a CEEHDLR.

Para obtener más información sobre cómo utilizar los manejadores de condiciones de ILE, consulte la publicación *ILE Concepts*.

Utilización de un manejador de condiciones

El ejemplo siguiente muestra cómo:

1. Codificar un manejador de condiciones para manejar el error de "fuera de límites" RPG
2. Registrar un manejador de condiciones
3. Desregistrar un manejador de condiciones
4. Codificar una subrutina de error *PSSR.

Este ejemplo consta de dos procedimientos:

- RPGHDLR, que consta de un manejador de condiciones escrito por el usuario para los errores de subseries fuera de límites
- SHOWERR, que prueba el procedimiento RPGHDLR.

Mientras que SHOWERR está diseñado principalmente para mostrar el funcionamiento de RPGHDLR, los dos procedimientos combinados también son útiles para determinar cómo funciona el manejo de excepciones de ILE. Ambos procedimientos graban en QSYSPRT las "acciones" que se llevan a cabo a medida

Manejadores de condiciones ILE

que se procesan. Tal vez desee modificar estos procedimientos para simular otros aspectos del manejo de excepciones de ILE sobre los que le gustaría saber algo más.

La Figura 133 en la página 277 muestra el fuente para el procedimiento RPGHDLR. Este procedimiento define tres parámetros de procedimiento: una estructura de señales de condiciones ILE, un puntero a un área de comunicaciones entre SHOWERR y RPGHDLR, y un campo para contener las acciones posibles, que son reanudar o filtrar. (RPGHDLR no promociona ninguna excepción).

La lógica básica de RPGHDLR es la siguiente:

1. Se comprueba si se trata de un error de fuera de límites comprobando el ID de mensaje
 - Si es así, y si SHOWERR ha indicado que los errores de fuera de límites pueden pasarse por alto, el procedimiento escribe 'Manejando...' en QSYSPRT y después establece la acción en "Reanudar".
 - De lo contrario, se escribe "Filtrando" en QSYSPRT y a continuación se establece la acción en "Filtrar".
2. Volver.

```

=====
* RPGHDLR: Procedimiento de manejo de excepciones RPG.      *
* Este procedimiento hace lo siguiente:                      *
* Maneja la excepción si es el error de fuera de            *
* límites RPG (RX0100)                                       *
* de lo contrario,                                           *
* filtra la excepción                                         *
* También imprime las acciones realizadas.                  *
*                                                            *
* Nota: Se trata de un procedimiento de manejo de           *
* excepciones para el procedimiento SHOWERR.                 *
=====
FQSYSPRT  0    F 132      PRINTER

D RPGHDLR      PR
D Parm1                LIKE(CondTok)
D Parm2                *
D Parm3            10I 0
D Parm4                LIKE(CondTok)

*-----*
* Parámetros del procedimiento                                *
* 1. Entrada: Estructura de símbolos de condición            *
* 2. Entrada: Puntero al área de comunicaciones que contiene *
*      a. Un puntero a la PSDS del procedimiento manejado    *
*      b. Un indicador de si un error de serie es válido     *
* 3. Entrada: Código que identifica acciones que deben realizarse *
*      en la excepción                                        *
* 4. Salida: Condición nueva si se decide promocionar la     *
*      condición. Puesto que este manejador sólo reanuda    *
*      y filtra, se pasará por alto este parámetro.         *
*-----*
D RPGHDLR      PI
D InCondTok    LIKE(CondTok)
D pCommArea    *
D Action       10I 0
D OutCondTok    LIKE(CondTok)

```

Figura 133. Fuente del manejador de condiciones para error de subserie fuera de límites (Pieza 1 de 2)

Manejadores de condiciones ILE

```

D CondTok          DS          BASED(pCondTok)
D MsgSev           5I 0
D MsgNo            2A
D                  1A
D MsgPrefix        3A
D MsgKey           4A

D CommArea         DS          BASED(pCommArea)
D pPSDS            *
D AllowError       1N

D PassedPSDS       DS          BASED(pPSDS)
D ProcName         1 10

*
* Los códigos de acción son:
*
D Resume          C          10
D Percolate       C          20

*-----*
* Señalar al símbolo de condición de entrada *
*-----*
C          EVAL      pCondTok = %ADDR(InCondTok)

*-----*
* Si se produce error de subserie, manejar; si no, filtrar. *
* Observe que el valor de número de mensaje (MsgNo) está en hex. *
*-----*
C          EXCEPT
C          IF          MsgPrefix = 'RNX' AND
C                      MsgNo    = X'0100' AND
C                      AllowError = '1'
C          EXCEPT    Handling
C          EVAL        Action   = Resume
C          ELSE
C          EXCEPT    Perclating
C          EVAL        Action   = Percolate
C          ENDIF
C          RETURN

*=====
* Salida del procedimiento *
*=====
OQSYSPRT  E
0
0          ProcName
OQSYSPRT  E          Handling
0          'HDLR: Manejando...'
OQSYSPRT  E          Perclating
0          'HDLR: Filtrando...'

```

Figura 133. Fuente del manejador de condiciones para error de subserie fuera de límites (Pieza 2 de 2)

En la Figura 134 en la página 280 se muestra el fuente del procedimiento SHOWERR, en el que se registra el manejador de condiciones RPGHDLR.

Los parámetros del procedimiento incluyen un puntero de procedimiento a RPGHDLR y un puntero al área de comunicaciones que contiene un puntero a la PSDS del módulo y un indicador que indica si el error de serie de fuera de límites puede pasarse por alto. Además, se requiere una definición para la matriz propensa a errores ARR1 y la identificación de las listas de parámetros utilizadas por las API enlazables CEEHDLR y CEEHDLU de ILE.

La lógica básica del programa es la siguiente:

1. Se registra el manejador RPGHDLR utilizando la subrutina RegHndlr. Esta subrutina llama a la API CEEHDLR y pasa el puntero de procedimiento a RPGHDLR.
2. Se indica a RPGHDLR que el error de fuera de límites está permitido, y a continuación se genera un error de subserie de fuera de límites, luego se desactiva el indicador para que RPGHDLR no permita errores de subserie de fuera de límites inesperados.

Se llama automáticamente al manejador RPGHDLR. Éste maneja la excepción e indica que el proceso debe reanudarse en la siguiente instrucción de *máquina* que haya a continuación del error. Observe que dicha instrucción puede no estar al principio de la siguiente operación RPG.

3. Se genera un error de matriz fuera de límites.

De nuevo, se llama automáticamente a RPGHDLR. Sin embargo, esta vez no puede manejar la excepción, por lo que la filtra al siguiente manejador de excepciones asociado con el procedimiento, es decir, la subrutina de error *PSSR.

La *PSSR cancela el procedimiento.

4. Se desregistra el manejador de condiciones RPGHDLR mediante una llamada a CEEHDLU.
5. Volver.

Al igual que con el procedimiento RPGHDLR, SHOWERR escribe en QSYSPRT para mostrar lo que sucede a medida que se procesa.

Manejadores de condiciones ILE

```

=====
* SHOWERR:      Muestra el manejo de excepciones mediante un      *
*                manejador de excepciones definido por usuario.    *
=====
FQSYSPRT  0    F 132          PRINTER

*-----*
* A continuación se indican las definiciones de parámetro para la *
* API CEEHDLR. El primero es el puntero de procedimiento al      *
* procedimiento que manejará la excepción. El segundo es un      *
* puntero a un área de comunicaciones que se pasará al          *
* procedimiento de manejo de excepciones. En este ejemplo, esta  *
* área contendrá un puntero a la PSDS de este módulo y un      *
* indicador de si un error está permitido.                        *
*                                                                *
* Es necesario asegurarse de que este programa (SHOWERR) no      *
* ignora los errores manejados, y por tanto se comprobará el    *
* indicador 'Error' después de cualquier operación que pueda    *
* provocar un error que RPGHDLR "permita". También se comprobará *
* al final del programa para garantizar que no se haya pasado    *
* por alto ningún error.                                         *
*-----*
D pConHdlr      S              *  PROCPTR
D                                     INZ(%paddr('RPGHDLR'))

*-----*
* Área de comunicaciones                                         *
*-----*
D CommArea      DS              NOOPT
D  pPsds              *  INZ(%ADDR(DSPsds))
D  AllowError        1N  INZ('0')

*-----*
* PSDS                                                         *
*-----*
D DSPsds        SDS              NOOPT
D  ProcName      *PROC

*-----*
* Variables que se utilizarán para causar errores.              *
*-----*
D Arr1          S              10A  DIM(5)
D Num           S              5P  0

*-----*
* Interfaz CEEHDLR                                              *
*-----*
D CEEHDLR        PR
D  pConHdlr      *  PROCPTR
D  CommArea      *  CONST
D  Feedback      12A  OPTIONS(*OMIT)

*-----*
* Interfaz CEEHDLU                                              *
*-----*
D CEEHDLU        PR
D  pConHdlr      *  PROCPTR
D  Feedback      12A  OPTIONS(*OMIT)

```

Figura 134. Fuente para registrar un manejador de condiciones (Pieza 1 de 3)

```

*-----*
* Registrar el manejador y generar errores *
*-----*
C          EXSR      RegHndl r

*-----*
*      Generar un error de subserie *
*      Es un error "permitido" para este ejemplo (RPGHDLR maneja *
*      la excepción, permitiendo que el control vuelva a la *
*      siguiente instrucción después del error). *
*      RPGHDLR no permitirá el error a menos que el indicador *
*      "AllowError" esté activado. Esto asegura que si, por *
*      ejemplo, se añade una operación SCAN a SHOWERR más tarde, *
*      RPGHDLR no permitirá por omisión que tenga un error. *
*-----*
C          Z-ADD      -1      Num
C          EVAL      AllowError = '1'
C          SUBST      'Hello'  Examp      10
C          EVAL      AllowError = '0'

*-----*
*      El manejador ha manejado la excepción y en este punto se *
*      reanuda el control. *
*-----*
C          EXCEPT   ImBack

*-----*
*      Generar un error de matriz fuera de límites. *
*      Este no es un error "esperado" en este ejemplo. *
*-----*
C          Z-ADD      -1      Num
C          MOVE      Arr1(Num) Arr1(Num)

*-----*
*      El manejador no ha manejado la excepción, y por tanto el *
*      control no se devuelve en este punto. La excepción se *
*      filtra y el control se reanuda en la *PSSR. *
*-----*

*-----*
*      Desregistrar el manejador *
*      Nota: Si se produce una excepción antes de desregistrar *
*      el manejador, se desregistrará automáticamente cuando *
*      se cancele el procedimiento. *
*-----*
C          EXSR      DeRegHndl r
C          SETON      LR

*=====*
* RegHdlr - Llamada a la API para registrar el manejador *
*=====*
C          RegHdlr    BEGSR
C          CALLP      CEEHDLR(pConHdlr : %ADDR(CommArea) : *OMIT)
C          ENDSR

*=====*
* DeRegHdlr - Llamada a la API para desregistrar el manejador *
*=====*
C          DeRegHdlr  BEGSR
C          CALLP      CEEHDLU(pConHdlr : *OMIT)
C          ENDSR

```

Figura 134. Fuente para registrar un manejador de condiciones (Pieza 2 de 3)

Manejadores de condiciones ILE

```

=====
* *PSSR: Subrutina de error para el procedimiento
*
=====
C      *PSSR      BEGSR
C      EXCEPT
C      EXCEPT   InPssr
C      ENDSR      Cancellling
C                  '*CANCL'

=====
* Salida del procedimiento
*
=====
OQSYSPRT  E      ImBack
0
OQSYSPRT  E      InPssr
0                  'He vuelto'
OQSYSPRT  E      Cancellling
0                  'En PSSR'
OQSYSPRT  E      Cancellling
0                  'Cancelando...'

```

Figura 134. Fuente para registrar un manejador de condiciones (Pieza 3 de 3)

Si desea probar estos procedimientos, siga estos pasos:

1. Para crear el procedimiento RPGHDLR utilizando el fuente que aparece en la Figura 133 en la página 277, escriba:
CRTRPGMOD MODULE(MYLIB/RPGHDLR)
2. Para crear el procedimiento SHOWERR utilizando el fuente que aparece en la Figura 134 en la página 280, escriba:
CRTRPGMOD MODULE(MYLIB/SHOWERR)
3. Para crear el programa ERRORTTEST, escriba
CRTPGM PGM(MIBIB/ERRORTTEST) MODULE(SHOWERR RPGHDLR)
4. Para ejecutar el programa ERRORTTEST, escriba:
OVRPRTF FILE(QSYSPRT) SHARE(*YES)
CALL PGM(MYLIB/ERRORTTEST)

A continuación se muestra la salida:

```

HDLR: En manejador para SHOWERR
HDLR: Manejando...
He vuelto
HDLR: En manejador para SHOWERR
HDLR: Filtrando...
En PSSR
Cancelando...

```

Utilización de manejadores de cancelación

Los manejadores de cancelación proporcionan una importante función al permitirle obtener el control de las acciones de borrado y recuperación cuando las entradas de la pila de llamadas se terminan por una razón distinta a un retorno normal. Por ejemplo, puede que desee obtener el control cuando un procedimiento finaliza mediante la petición de sistema "2" o porque se ha respondido "C" (Cancelar) a un mensaje de consulta.

Las API enlazables ILE Registrar procedimiento de salida de usuario de finalización de entrada de pila de llamadas (CEERTX) y Procedimiento de salida de usuario de finalización de entrada de pila de llamadas (CEEUTX) proporcionan un modo de registrar dinámicamente una rutina definida por usuario para que se ejecute cuando se cancela la entrada de la pila de llamadas para la que se ha registrado. Una vez registrado, el manejador de excepciones permanece en vigor

hasta que se elimina la entrada de la pila de llamadas o hasta se llama a CEEUTX para inhabilitarlo. Para obtener más información acerca de estas API enlazables de ILE consulte la sección relativa a *CL y API* de la categoría *Programación de iSeries 400 Information Center* en este sitio Web - <http://www.ibm.com/eserver/iseriess/infocenter>.

La Figura 135 muestra un ejemplo de cómo habilitar y codificar un manejador de cancelación para un subprocedimiento. (Los manejadores de cancelación también pueden habilitarse para procedimientos principales del mismo modo.)

```

*-----
* Definir el prototipo del manejador de cancelación. Este
* procedimiento es local.
*-----
D CanHdlr          PR
D   pMsg           *
*-----
* Definir el prototipo de un subprocedimiento para habilitar el
* manejador de cancelación.
*-----
D Enabler          PR
*-----
* Definir el prototipo de un subprocedimiento para llamar a Enabler
*-----
D SubProc          PR
*-----
* Procedimiento principal. Llamar a SubProc tres veces.
*-----
C                   CALLP   SubProc
C                   CALLP   SubProc
C                   CALLP   SubProc
C                   SETON                    LR
*-----
* Procedimiento SubProc. Llamar a Enabler. Dado que esta llamada
* fallará, definir una subrutina *PSSR local para manejar el error.
*-----
P SubProc          B
C                   CALLP   Enabler
*-----
* La PSSR tiene una operación RETURN, por tanto la llamada desde el
* procedimiento principal a SubProc no fallará.
*-----
C   *PSSR          BEGSR
C   'Subproc PSSR'DSPLY
C                   RETURN
C                   ENDSR
P SubProc          E

```

Figura 135. Habilitación y codificación de un manejador de cancelación para un subprocedimiento (Pieza 1 de 3)

```

*-----
* Procedimiento Enabler. Este procedimiento habilita un manejador de
* cancelación y luego obtiene un error que provoca la cancelación de
* Enabler.
*-----
P Enabler      B
* Variables locales
D Handler      S      *   PROCPTR  INZ(%PADDR('CANHDLR'))
D Msg          S      20A
D pMsg         S      *   INZ(%ADDR(Msg))
D Zero         S      5P 0 INZ(0)
D Count        S      5I 0 INZ(0) STATIC
D Array        S      1A   DIM(2)
*-----
* Habilitar el manejador de cancelación. Cuando este procedimiento
* se cancele, se llamará al procedimiento 'CANHDLR'.
*-----
C              CALLB   'CEERTX'
C              PARM
C              PARM      Handler
C              PARM      pMsg
C              PARM      *OMIT
*-----
* Se llamará tres veces a este procedimiento. Las dos primeras veces
* se obtendrá un error mientras se habilita el manejador de cancelación.
*-----
C              EVAL    Count = Count + 1
C              SELECT
C              WHEN    Count = 1
C              EVAL    Msg = 'Dividir por cero'
C              EVAL    Zero = Zero / Zero
C              WHEN    Count = 2
C              EVAL    Msg = 'Error de serie'
C      'A'          SCAN  'ABC':Zero   Zero
*-----
* En la tercera llamada, se inhabilita el manejador de cancelación.
* El error de índice de matriz hará que el procedimiento falle,
* pero no se llamará al manejador.
*-----
C              WHEN    Count = 3
C              CALLB   'CEEUTX'
C              PARM
C              PARM      Handler
C              PARM      *OMIT
C              EVAL    Msg = 'Error de índice de matriz'
C              EVAL    Array(Zero) = 'x'
C              ENDSL
P Enabler      E

```

Figura 135. Habilitación y codificación de un manejador de cancelación para un subprocedimiento (Pieza 2 de 3)

```

*-----
* Definir el manejador de cancelación. El parámetro es un puntero al
* 'área de comunicaciones', un mensaje que debe visualizarse.
*-----
P CanHdlr      B
D CanHdlr      PI
D   pMsg                      *
*-----
* Definir un campo basado en el puntero de entrada pMsg.
*-----
D Msg          S          20A      BASED(pMsg)
*-----
* Visualizar el mensaje establecido por el procedimiento que ha
* habilitado el manejador.
*-----
C   'Cancel Hdlr 'DSPLY          Msg
P CanHdlr      E

```

Figura 135. Habilitación y codificación de un manejador de cancelación para un subprocedimiento (Pieza 3 de 3)

A continuación se indica la salida del programa CANHDLR. Observe que se llama tres veces a la *PSSR del procedimiento SubProc, pero que sólo se llama dos veces al manejador de cancelación porque se ha inhabilitado antes del tercer error.

```

DSPLY  Cancel Hdlr      Dividir por cero
DSPLY  Subproc PSSR
DSPLY  Cancel Hdlr      Error de serie
DSPLY  Subproc PSSR
DSPLY  Subproc PSSR

```

Figura 136. Salida del programa CANHDLR

Problemas cuando ILE CL supervisa los mensajes de estado y notificación

Si un procedimiento ILE CL que está en el mismo grupo de activación llama al procedimiento ILE RPG, y el que efectúa la llamada está supervisando los mensajes de estado o notificación, el llamador ILE CL puede obtener prematuramente el control debido a un mensaje de estado o de notificación que el procedimiento ILE RPG estaba intentando pasar por alto.

Por ejemplo, si el procedimiento ILE RPG graba un registro en un archivo de impresora y el archivo de impresora real tiene una longitud de registro más corta que la declarada en el procedimiento RPG, se envía el mensaje de notificación CPF4906 al procedimiento RPG. El manejo de excepciones RPG filtra este mensaje que provoca la respuesta por omisión 'I' para pasar por alto el mensaje. Esto debería permitir que la operación de salida prosiga con normalidad, y el procedimiento RPG debería continuar con la siguiente instrucción.

Sin embargo, cuando ILE CL MONMSG obtiene el control, éste pasa inmediatamente a la acción de MONMSG o a la siguiente sentencia del procedimiento ILE CL.

Nota: Para que se produzca este problema, el procedimiento que supervisa el mensaje no puede ser el llamador inmediato del procedimiento RPG.

Este problema tiene más posibilidades de producirse con un MONMSG en un llamador ILE CL, pero también puede producirse con otros lenguajes ILE que pueden supervisar mensajes de estado y notificación, incluyendo ILE RPG que utilice manejadores de condiciones ILE habilitados mediante CEEHDLR.

Si encuentra este problema, tiene dos posibles maneras de evitarlo:

1. Asegurarse de que el llamador está en un grupo de activación diferente del del procedimiento ILE RPG.
2. Habilitar un manejador de condiciones ILE en el procedimiento RPG. En el manejador, si el mensaje es uno que desea pasar por alto, indique que el mensaje debe manejarse. De lo contrario, indique que debe filtrarse.

También puede hacer que este manejador sea más genérico, y hacer que pase por alto todos los mensajes de gravedad 0 (información) y 1 (aviso).

La Figura 137 muestra un ejemplo de un manejador de condiciones ILE que pasa por alto el mensaje CPF4906.

```

*-----
* Definiciones de manejador
*-----
D Action          S          10I 0
D Token           DS
D  MsgSev          5I 0
D  MsgNo           2A
D                 1A
D  Prefix          3A
D                 4A
*-----
* Acciones
*-----
D Handle          C          10
D Percolate       C          20
*-----
* Gravedades
*-----
D Info            C          0
D Warning         C          1
D Error           C          2
D Severe          C          3
D Critical        C          4
C  *ENTRY          PLIST
C                  PARM
C                  PARM          Token
C                  PARM          dummy          1
C                  PARM          Action
*-----
* Si es CPF4906, manejar el mensaje de notificación, si no lo es, filtrar
*-----
C                  IF          Prefix = 'CPF' AND
C                  MsgNo = X'4906'
C                  EVAL          Action = Handle
C                  ELSE
C                  EVAL          Action = Percolate
C                  ENDIF
C                  RETURN

```

Figura 137. Manejador de condiciones ILE que pasa por alto el mensaje CPF4906

La Figura 138 en la página 287 muestra cómo codificar los cálculos si desea pasar por alto todos los mensajes de estado y de notificación. Los mensajes de escape y los errores de función tienen un gravedad 2 (Error) o superior.


```

*-----
* Manejar mensajes de información o aviso, de lo contrario, filtrar
*-----
C      IF      MsgSev <= Warning
C      EVAL    Action = Handle
C      ELSE
C      EVAL    Action = Percolate
C      ENDIF
C      RETURN

```

Figura 138. Cómo pasar por alto los mensajes de estado y notificación

Capítulo 14. Obtención de un vuelco

En este capítulo se describe cómo obtener un vuelco con formato ILE RPG y se proporciona un ejemplo de vuelco con formato.

Obtención de un vuelco con formato ILE RPG

Para obtener un vuelco con formato ILE RPG (es decir, una copia impresa del almacenamiento) para un procedimiento mientras éste se está ejecutando, puede realizar lo siguiente

- Codificar uno o varios códigos de operación DUMP en las especificaciones de cálculo
- Responder a un mensaje en la ejecución con la opción D o F. También es posible responder automáticamente a fin de obtener un vuelco. Consulte la descripción de “Lista de respuestas del sistema” de la publicación *CL Programming*.

El vuelco con formato incluye el contenido de los campos, el contenido de la estructura de datos, el contenido de tablas y matrices, las estructuras de datos de información de los archivos y la estructura de datos de estado de programa. El vuelco se graba en el archivo denominado QPPGMDMP. (Un vuelco anormal del sistema se graba en el archivo QPSRVDMP).

Si responde a un mensaje ILE RPG durante la ejecución con una opción F, el vuelco también incluye la representación hexadecimal de la vía de datos abierta (ODP, un bloque de control de gestión de datos).

La información de vuelco incluye los datos globales asociados con el módulo. Dependiendo de si está activo el procedimiento principal, los datos globales pueden no representar los valores asignados durante el proceso de *INZSR. Si un programa consta de más de un procedimiento, la información del vuelco con formato también reflejará la información sobre *todos* los procedimientos que estaban activos en el momento de realizar la petición de vuelco. Si un procedimiento no está activo, los valores de las variables del almacenamiento automático no serán válidos. Si un procedimiento no se ha llamado todavía, el almacenamiento estático no estará inicializado. Si se ha llamado a un procedimiento de modo recurrente, sólo se visualizará la información para la invocación más reciente.

Hay dos situaciones en las que los datos de vuelco pueden no estar disponibles:

- Si el objeto de programa se ha creado con vista de depuración *NONE. El vuelco sólo contendrá la PSDS, la información de archivo y los indicadores *IN.
- Si una sola variable o estructura requiere más de 16 MB de datos de vuelco. Esto se produce generalmente en las variables o estructuras cuyo tamaño es superior a 5 MB.

Utilización del código de operación DUMP

Puede codificar uno o más códigos de operación DUMP en los cálculos del fuente para obtener un vuelco con formato ILE RPG. Se creará un nuevo archivo de spool QPPGMDMP siempre que se produzca una operación DUMP.

Tenga en cuenta lo siguiente acerca de la operación DUMP:

Utilización del código de operación DUMP

- Para determinar si una operación DUMP provocará que se produzca un vuelco con formato, debe comprobar el ampliador de operación en la operación DUMP y la palabra clave DEBUG en la especificación de control. El vuelco con formato se producirá si se ha especificado el ampliador (A) en la operación DUMP o si se ha especificado la palabra clave DEBUG sin parámetros o con el parámetro *YES. Si no se ha especificado el ampliador (A) en la operación DUMP y tampoco se ha especificado la palabra clave DEBUG, o se ha especificado con el parámetro (*YES), se comprueba si la operación DUMP contiene errores y la sentencia se imprime en la lista, pero la operación DUMP no se procesa.
- Si la operación DUMP está condicionada, sólo se ejecutará si se cumple la condición.
- Si se elude una operación DUMP mediante una operación GOTO, la operación DUMP no se ejecutará.

Ejemplo de vuelco con formato

Las figuras siguientes muestran un ejemplo de un vuelco con formato de un módulo similar a DBGEX (consulte el apartado “Ejemplo de fuente para ejemplos de depuración” en la página 245). Para mostrar cómo se manejan los almacenamientos intermedios de datos en un vuelco con formato, se ha añadido el archivo de salida QSYSPRT.

El vuelco de este ejemplo es un vuelco con formato completo; es decir, se ha creado cuando se respondió "F" a un mensaje de consulta.

Información de estado de programa

Área de estado del programa:

Nombre del procedimiento	DBGEX2	← A
Nombre del programa	TEST	
Biblioteca	MYLIB	
Nombre del módulo	DBGEX2	
Estado del módulo	00202	B C D
Estado anterior	00000	
Sentencia con error	00000088	
Rutina RPG	RPGPGM	E
Número de parámetros		
Tipo de mensaje	MCH	F
Información adicional mensaje	4431	
Datos del mensaje		
Violación de signature de programa.		
Estado que causó RNX9001		← G
Último archivo utilizado		
Último estado del archivo		
Última operación del archivo		
Última rutina del archivo		
Última sentencia del archivo		
Nombre del último registro archivo . .		←
Nombre del trabajo	MYUSERID	
Nombre del usuario	MYUSERID	← H
Número del trabajo	002273	
Fecha en que entró en el sistema . . .	09/30/1995	
Fecha iniciado	*N/A*	
Hora iniciado	*N/A*	
Fecha de compilación	123095	
Hora de compilación	153438	
Nivel del compilador	0001	
Archivo fuente	QRPGLESRC	
Biblioteca	MYLIB	
Miembro	DBGEX2	←

Figura 139. Sección de información de estado del programa del vuelco con formato

- A** Identificación del procedimiento: nombre del procedimiento, nombre del programa y biblioteca y nombre del módulo.
- B** Código de estado actual.
- C** Código de estado anterior.
- D** Sentencia fuente del ILE RPG errónea.
- E** Rutina del ILE RPG en la que se produjo la excepción o el error.
- F** CPF o MCH para una excepción de máquina.
- G** Información acerca del último archivo utilizado en el programa antes de producirse una excepción o error. En este caso, no se ha utilizado ningún archivo.
- H** Información del programa. "*N/A*" indica cuáles son los campos para los que no hay información disponible en el programa. Estos campos sólo se actualizan si se incluyen en la PSDS.

Áreas de realimentación

Ejemplo de vuelco con formato

```

REALIMENTACIÓN ARCHIVO INFDS I
Archivo . . . . . : QSYSPRT
Archivo abierto . . . . . : YES
Archivo en EOF . . . . . : NO
Estado de archivo . . . . . : 00000
Operación de archivo . . . . . : OPEN I
Rutina de archivo . . . . . : *INIT
Número de sentencia . . . . . : *INIT
Nombre de registro . . . . . :
Identificador de mensaje . . . . . :
REALIMENTACIÓN APERTURA J
Tipo ODP . . . . . : SP
Nombre archivo . . . . . : QSYSPRT
Biblioteca . . . . . : QSYS
Miembro . . . . . : Q501383525
Archivo de spool . . . . . : Q04079N002
Biblioteca . . . . . : QSPL
Número archivo de spool . . . . . : 7
Longitud registro primario . . . . . : 80
Longitud bloque entrada . . . . . : 0
Longitud bloque salida . . . . . : 80
Clase dispositivo . . . . . : PRINTER
Líneas por página . . . . . : 66
Columnas por línea . . . . . : 132
Permitir claves duplicadas . . . . . : *N/A*
Registros a transferir . . . . . : 1
Línea desbordamiento . . . . . : 60
Incremento registro bloque . . . . . : 0
Permitir archivos compartidos . . . . . : NO
Archivo dispositivo creado con DDS . . . . . : NO
IGC o archivo con capacidad gráficos . . . . . : NO
Cuenta archivo abierto . . . . . : 1
Separar área de indicador . . . . . : NO
Almacenamientos intermedios usuario . . . . . : NO
Identificador abierto . . . . . : Q04079N002
Longitud máxima registro . . . . . : 0
ODP con ámbito para trabajo . . . . . : NO
Máximo dispositivos programa . . . . . : 1
Dispositivo programa actual definido . . . . . : 1
Nombre dispositivo . . . . . : *N
Nombre descripción dispositivo . . . . . : *N
Clase dispositivo . . . . . : '02'X
Tipo dispositivo . . . . . : '08'X

REALIMENTACIÓN DE E/S COMÚN K
Número de Transferencias . . . . . : 0
Número de Obtenciones . . . . . : 0
Número de Transferencias/Obtenciones . . . . . : 0
Número de otras E/S . . . . . : 0
Operación actual . . . . . : '00'X
Formato registro . . . . . :
Clase y tipo dispositivo . . . . . : '0208'X
Nombre dispositivo . . . . . : *N
Longitud último registro . . . . . : 80
Número de registros recuperados . . . . . : 80
Longitud último registro E/S . . . . . : 0
Cuenta bloque actual . . . . . : 0

REALIMENTACIÓN DE IMPRESORA:
Número de líneas actual . . . . . : 1
Página actual . . . . . : 1
Código de retorno principal . . . . . : 00
Código de retorno secundario . . . . . : 00

Almacenamiento intermedio de salida:
0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *
0020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *
0040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *

```

Figura 140. Sección de áreas de realimentación del vuelco con formato

- I** Esta es la sección de realimentación de archivo de la INFDS. Sólo se imprimen los campos que corresponden al tipo de archivo. El resto de las secciones de realimentación de la INFDS no se vuelcan, ya que sólo se actualizan si se han declarado en el programa.
- J** Esta es la información de realimentación de apertura de archivo correspondiente al archivo. Para obtener una descripción de los campos, consulte la sección relativa a *DB2 Universal Database para AS/400* de la categoría *Sistemas de archivos y bases de datos* de **iSeries 400 Information Center** en este sitio Web - <http://www.ibm.com/eserver/iseres/infocenter>.

Ejemplo de vuelco con formato

K Ésta es la información de realimentación de E/S común del archivo. Para obtener una descripción de los campos, consulte el sitio Web citado anteriormente.

Información para vuelco con formato completo

Vía de datos abierta:									
0000	64800000	00001AF0	00001B00	000000B0	00000140	000001C6	00000280	000002C0	* 0 F *
0020	00000530	00000000	00000000	00000380	00000000	06000000	00000000	00000000	* *
0040	00008000	00000000	003AC02B	A00119FF	000006C0	00003033	00000000	00000000	* *
0060	80000000	00000000	003AC005	CF001CB0	00000000	00000000	00000000	00000000	* *
0080	80000000	00000000	003AA024	D0060120	01900000	00010000	00000050	00000000	* & *
00A0	1F000000	00000000	00000000	00000000	E2D7D8E2	E8E2D7D9	E3404040	D8E2E8E2	* SPQSYSPRT QSYS*
00C0	40404040	4040D8F0	F4F0F7F9	D5F0F0F2					* Q04079N002QSPL & *
Realimentación apertura:									
0000	E2D7D8E2	E8E2D7D9	E3404040	D8E2E8E2	40404040	4040D8F0	F4F0F7F9	D5F0F0F2	*SPQSYSPRT QSYS Q04079N002*
0020	D8E2D7D3	40404040	40400007	00500000	D8F5F0F1	F3F8F3F5	F2F50000	00000000	*QSPL & Q501383525 *
0040	00500002	00000000	42008400	00000000	0000D5A4	00100000	00000008	00000000	* & d Nu *
0060	00000000	00000000	00000100	3C000000	0005E000	5CD54040	40404040	40400001	* *N *
0080	00000000	00001300	00000000	00000000	00010001	5CD54040	40404040	40400000	* *N *
00A0	07100000	00000000	00450045	00450045	07A10045	00450045	00700045	00450045	* *
00C0	00450045	00450045	002F0030	00040005	5CD54040	40404040	40400208	00000000	* *N *
00E0	20000000	00000000	00000000	00000000	00000000	00000001	C2200000	00059A00	* B *
0100	00000000	00000000	00000000	00000000	00000000	4040			* *
Realimentación E/S común:									
0000	00900000	00000000	00000000	00000000	00000000	00000000	00000000	00000208	* *
0020	5CD54040	40404040	40400000	00500000	00000000	00000000	00000000	00000000	**N & *
0040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	* *
0060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	* *
0080	00000000	00000000	00000000	00000000	00000000				* *
Realimentación E/S para dispositivo:									
0000	00010000	00010000	00000000	00000000	00000000	00000000	00000000	00000000	* *
0020	0000F0F0	0001							* 0000 *

Figura 141. Información proporcionada para vuelco con formato completo

La vía de datos abierta y las áreas de realimentación comunes asociadas con el archivo se incluyen en el vuelco si responde a un mensaje de consulta de ILE RPG con la opción F.

Información de datos

Ejemplo de vuelco con formato

```

VUELCO CON FORMATO ILE RPG
Nombre de módulo . . . . . : DBGEX2
Nivel de optimización . . . . . : *NONE
Indicadores de parada:
H1 '0' H2 '0' H3 '0' H4 '0' H5 '0' H6 '0' H7 '0' H8 '0' H9 '0'
Indicadores de mandato/tecla de función:
KA '0' KB '0' KC '0' KD '0' KE '0' KF '0' KG '0' KH '0' KI '0' KJ '0'
KK '0' KL '0' KM '0' KN '0' KP '0' KQ '0' KR '0' KS '0' KT '0' KU '0'
KV '0' KW '0' KX '0' KY '0'
Indicadores de nivel de control:
L1 '0' L2 '0' L3 '0' L4 '0' L5 '0' L6 '0' L7 '0' L8 '0' L9 '0'
Indicadores de desbordamiento:
OA '0' OB '0' OC '0' OD '0' OE '0' OF '0' OG '0' OV '0'
Indicadores externos:
U1 '0' U2 '0' U3 '0' U4 '0' U5 '0' U6 '0' U7 '0' U8 '0'
Indicadores generales:
01 '0' 02 '1' 03 '0' 04 '1' 05 '0' 06 '1' 07 '0' 08 '0' 09 '0' 10 '0'
11 '0' 12 '0' 13 '0' 14 '0' 15 '0' 16 '0' 17 '0' 18 '0' 19 '0' 20 '0'
21 '0' 22 '0' 23 '0' 24 '0' 25 '0' 26 '0' 27 '0' 28 '0' 29 '0' 30 '0'
31 '0' 32 '0' 33 '0' 34 '0' 35 '0' 36 '0' 37 '0' 38 '0' 39 '0' 40 '0'
41 '0' 42 '0' 43 '0' 44 '0' 45 '0' 46 '0' 47 '0' 48 '0' 49 '0' 50 '0'
51 '0' 52 '0' 53 '0' 54 '0' 55 '0' 56 '0' 57 '0' 58 '0' 59 '0' 60 '0'
61 '0' 62 '0' 63 '0' 64 '0' 65 '0' 66 '0' 67 '0' 68 '0' 69 '0' 70 '0'
71 '0' 72 '0' 73 '0' 74 '0' 75 '0' 76 '0' 77 '0' 78 '0' 79 '0' 80 '0'
81 '0' 82 '0' 83 '0' 84 '0' 85 '0' 86 '0' 87 '0' 88 '0' 89 '0' 90 '0'
91 '0' 92 '0' 93 '0' 94 '0' 95 '0' 96 '0' 97 '0' 98 '0' 99 '0'
Indicadores internos:
LR '0' MR '0' RT '0' 1P '0'
N
NOMBRE ATRIBUTOS VALOR
_QRNU_DSI_DSI INT(10) 1 '00000001'X O
_QRNU_DSI_DS2 INT(10) 2 '00000002'X
_QRNU_NULL_ARR CHAR(1) DIM(8) P
(1-2) '1' 'F1'X
(3) '0' 'F0'X
(4) '1' 'F1'X
(5-6) '0' 'F0'X
(7) '1' 'F1'X
(8) '0' 'F0'X
_QRNU_NULL_FLDNULL CHAR(1) '1' 'F1'X
_QRNU_TABI_TABLEA INT(10) 1 '00000001'X Q
ARR CHAR(2) DIM(8)
(1-3) 'AB' 'C1C2'X
(4-7) ' ' '4040'X
(8) '1' 'F1'X
ARRAY ZONED(3,2) DIM(2)
(1-2) 1.24 'F1F2F4'X
BASEONNULL CHAR(10) NOT ADDRESSABLE
BASEPTR POINTER SPP:E30095A62F001208
BASESTRING CHAR(6) 'ABCDEF' 'C1C2C3C4C5C6'X
BIGDATE DATE(10) '1994-09-30' 'F1F9F9F460F0F960F3F0'X
BIGTIME TIME(8) '12.00.00' 'F1F24BF0F04BF0F0'X
BIGTSTAMP TIMESTAMP(26) '9999-12-31-12.00.00.000000'
VALOR EN HEX 'F9F9F9F960F1F260F3F160F1F24BF0F04BF0F04BF0F0F0F0F0'X
BIN4D3 BIN(4,3) -4.321 'EF1F'X
BIN9D7 BIN(9,7) 98.7654321 '3ADE68B1'X
DBCSTRING GRAPHIC(3) ' BBCCDD ' 'C2C2C3C3C4C4'X

```

Figura 142. Sección de datos del vuelco con formato (Pieza 1 de 2)

DS1	DS	OCCURS(3)	R
OCCURRENCE(1)			
FLD1	CHAR(5)	'1BCDE'	'F1C2C3C4C5'X
FLD1A	CHAR(1)	DIM(5)	
	(1)	'1'	'F1'X
	(2)	'B'	'C2'X
	(3)	'C'	'C3'X
	(4)	'D'	'C4'X
	(5)	'E'	'C5'X
FLD2	BIN(5,2)	123.45	'00003039'X
OCCURRENCE(2)			
FLD1	CHAR(5)	'ABCDE'	'C1C2C3C4C5'X
FLD1A	CHAR(1)	DIM(5)	
	(1)	'A'	'C1'X
	(2)	'B'	'C2'X
	(3)	'C'	'C3'X
	(4)	'D'	'C4'X
	(5)	'E'	'C5'X
FLD2	BIN(5,2)	123.45	'00003039'X
OCCURRENCE(3)			
FLD1	CHAR(5)	'ABCDE'	'C1C2C3C4C5'X
FLD1A	CHAR(1)	DIM(5)	
	(1)	'A'	'C1'X
	(2)	'B'	'C2'X
	(3)	'C'	'C3'X
	(4)	'D'	'C4'X
	(5)	'E'	'C5'X
FLD2	BIN(5,2)	123.45	'00003039'X
DS2	CHAR(10)	DIM(2)	
	(1)	'aaaaaaaa'	'8181818181818181'X
	(2)	'bbbbbbbbbb'	'8282828282828282'X
DS3	DS		T
FIRSTNAME	CHAR(10)	'Fred'	'C69985844040404040'X
LASTNAME	CHAR(10)	'Jones'	'D1969585A240404040'X
TITLE	CHAR(5)	'Mr.'	'D4994B4040'X
EXPORTFLD	CHAR(6)	'export'	'85A7979699A3'X
FLDNULL	ZONED(3,1)	24.3	'F2F4F3'X
FLOAT1	FLT(4)	1.234500000000E+007	U
	VALOR EN HEX	'4B3C5EA8'X	
FLOAT2	FLT(8)	3.962745000000E+047	
	VALOR EN HEX	'49D15A640A93FCFF'X	
INT10	INT(10)	-31904	'FFFF8360'X
INT5	INT(5)	-2046	'F802'X
NEG_INF	FLT(8)	-HUGE_VAL	V
	VALOR EN HEX	'FFF0000000000000'X	
NOT_NUM	FLT(4)	*NaN	W
	VALOR EN HEX	'7FFFFFFF'X	
NULLPTR	POINTER	SYP:*NULL	
PACKED100	PACKED(5,2)	-093.40	'09340D'X
PARM1	PACKED(4,3)	6.666	'06666F'X
POS_INF	FLT(8)	HUGE_VAL	X
	VALOR EN HEX	'7FF0000000000000'X	
PROCPTR	POINTER	PRP:A00CA02EC200	Y
SPCPTR	POINTER	SPP:A026FA0100C0	
SPCSIZ	BIN(9,0)	000000008.	'00000008'X
STRING	CHAR(6)	'ABCDEF'	'C1C2C3C4C5C6'X
TABLEA	CHAR(3)	DIM(3)	
	(1)	'aaa'	'818181'X
	(2)	'bbb'	'828282'X
	(3)	'ccc'	'838383'X
UNSIGNED10	UNS(10)	31904	'00007CA0'X
UNSIGNED5	UNS(5)	2046	'07FE'X
ZONEDD3D2	ZONED(3,2)	-3.21	'F3F2D1'X
Variables locales para el subprocedimiento	SWITCH:	Z	
NOMBRE	ATRIBUTOS	VALOR	
_QRNL_PSTR_PARM	POINTER	SYP:*NULL	
LOCAL	CHAR(5)	'	'0000000000'X
PARM	CHAR(1)	NOT ADDRESSABLE	
***** F I N D E V U E L C O R P G *****			

Figura 142. Sección de datos del vuelco con formato (Pieza 2 de 2)

L

Nivel de optimización

M

Indicadores generales 1-99 y sus estados actuales ("1" es activado, "0" es desactivado). Observe que los indicadores *IN02, *IN04 y *IN06 todavía no estaban establecidos.

N

Inicio de variables de usuario, listadas por orden alfabético y agrupadas por procedimiento. Los datos locales a un subprocedimiento se almacenan en el almacenamiento automático y no están disponibles a menos que el subprocedimiento esté activo. Tenga en cuenta que se visualizan los valores hexadecimales de todas las variables. Los nombres :nt cuya longitud supere los 131 caracteres aparecen en la lista de vuelco divididos en varias

Ejemplo de vuelco con formato

líneas. La totalidad del nombre se imprimirá con los caracteres '...' al final de las líneas. Si la parte final del nombre supera los 21 caracteres, los atributos y valores se listarán empezando en la línea siguiente.

- O** Campos definidos internamente que contienen índices para estructuras de datos de varias apariciones.
- P** Campos definidos internamente que contienen los indicadores nulos para campos con capacidad de nulos.
- Q** Campos definidos internamente que contienen índices para tablas.
- R** Estructuras de datos de varias apariciones.
- S** Las estructuras de datos sin subcampos se visualizan como series de caracteres.
- T** Los subcampos de estructuras de datos se listan por *orden alfabético, no por el orden en el que están definidos*. Los espacios en blancos en las definiciones de subcampos no se muestran.
- U** Campos flotantes de 4 y 8 bytes.
- V** Indica infinito negativo.
- W** Significa que 'no es un número', indicando que el valor no es un número de coma flotante válido.
- X** Indica infinito positivo.
- Y** El atributo no distingue entre el puntero base y el puntero de procedimiento.
- Z** Los datos locales contenidos en los subprocedimientos se listan separados de la sección del fuente principal.

Parte 4. Trabajar con archivos y dispositivos

Esta sección describe cómo utilizar archivos y dispositivos en programas ILE RPG. De forma específica muestra cómo:

- Asociar un archivo a un dispositivo
- Definir un archivo (como descrito por programa o descrito externamente)
- Procesar archivos
- Acceso a archivos de base de datos
- Acceder a dispositivos conectados de forma externa
- Escribir una aplicación interactiva

Nota: el término 'programa RPG IV' hace referencia a un programa Integrated Language Environment que contiene uno o más procedimientos escritos en RPG IV.

Capítulo 15. Definición de archivos

Los archivos se utilizan a modo de enlace de conexión entre un programa y el dispositivo utilizado para E/S. Cada archivo del sistema tiene una descripción de archivo asociada que describe las características del archivo y cómo los datos asociados con el archivo se organizan en registros y campos.

Para que un programa efectúe cualquier operación de E/S, debe identificar la descripción del archivo o de los archivos a los que el programa hace referencia, qué tipo de dispositivo de E/S está utilizándose y cómo están organizados los datos. Este capítulo proporciona información general sobre:

- Asociar descripciones de archivo con dispositivos de entrada/salida
- Definición de archivos descritos externamente
- Definición de archivos descritos por programa
- Operaciones de gestión de datos

En los capítulos siguientes encontrará información acerca de cómo utilizar archivos descritos externamente y archivos descritos por programa con tipos de dispositivos diferentes.

Asociación de archivos con dispositivos de entrada/salida

El elemento clave para todas las operaciones de E/S en el iSeries es el archivo. El sistema da soporte a los tipos de archivo siguientes:

archivos de base de datos

permiten almacenar datos de forma permanente en el sistema

archivos de dispositivo

permiten acceder a dispositivos conectados externamente. En estos archivos se incluyen archivos de pantalla, de impresora, de cinta, de disquetes e ICF

archivos de salvar

se utilizan para almacenar datos salvados en disco

archivos DDM

permiten el acceso a los archivos de datos almacenados en sistemas remotos.

Cada dispositivo de E/S tiene una descripción de archivo correspondiente, de uno de los tipos que acaban de especificarse, que el programa utiliza para acceder a ese dispositivo. La asociación con el dispositivo real se efectúa cuando se procesa el archivo: se leen o graban los datos en el dispositivo cuando se utiliza el archivo durante el proceso.

RPG permite también acceder a archivos y dispositivos que no están soportados directamente por el sistema, mediante la utilización de archivos SPECIAL. Con un archivo SPECIAL, debe proporcionar un programa que maneja la asociación del nombre con el archivo y la gestión de datos del archivo. Con los otros tipos de archivo, de ello se ocupa RPG y el sistema operativo.

Para indicar al sistema operativo que descripción, o descripciones, de archivo utilizará el programa, debe especificar un *nombre de archivo* en las posiciones 7 a 16 de la especificación de descripción de archivo para cada archivo que utilice. En las

Asociación de archivos con dispositivos de entrada/salida

posiciones 36 a 42, especifique un *nombre de dispositivo* de RPG. El nombre del dispositivo define qué operaciones RPG pueden utilizarse con el archivo asociado. El nombre de dispositivo puede ser uno de los siguientes: DISK, PRINTER, WORKSTN, SEQ o SPECIAL. La Figura 143 muestra una especificación de descripción de archivo para un archivo de pantalla (WORKSTN) FILEX.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FARCHX      CF  E                      WORKSTN
```

Figura 143. Identificación de un archivo de pantalla en un programa en RPG

Observe que es el nombre del archivo, no el nombre del dispositivo (especificado en las posiciones 36 a 42), el que indica al OS/400 la descripción de archivo que contiene las especificaciones para el dispositivo real.

Los tipos de dispositivo RPG corresponden a los tipos de archivo descritos anteriormente:

Tabla 17. Correlación de tipos de dispositivo RPG con tipos de archivo iSeries.

Tipo de dispositivo RPG	Tipo de archivo iSeries
DISK	archivos de base de datos, de salvar y DDM
PRINTER	archivos de impresora
WORKSTN	archivos de pantalla, ICF
SEQ	archivos de cinta, de disquete, de salvar, de impresora y de base de datos
SPECIAL	N/D

La Figura 144 muestra la asociación del nombre de archivo ARCHX de RPG, tal como se ha codificado en la Figura 143, con la descripción de un archivo del sistema para un archivo de pantalla.

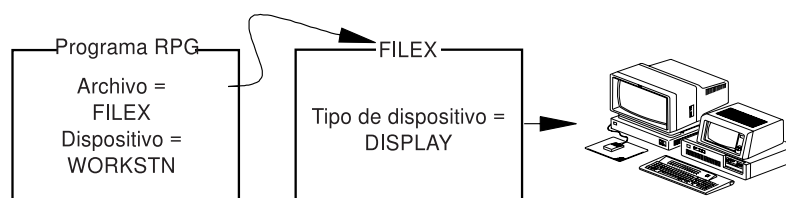


Figura 144. Asociación de un nombre de archivo con la descripción de un archivo de pantalla

en tiempo de compilación, determinadas operaciones de RPG son válidas sólo para un nombre de dispositivo específico de RPG. En este caso, la operación de RPG es dependiente del dispositivo. Un ejemplo de dependencia de dispositivo es que el código de operación EXFMT es válido sólo para un dispositivo WORKSTN.

Otros códigos de operación son independientes de dispositivo, lo que significa que pueden utilizarse con cualquier tipo de dispositivo. Por ejemplo, WRITE es una operación independiente del dispositivo.

El dispositivo SEQ

Asociación de archivos con dispositivos de entrada/salida

El dispositivo SEQ es un tipo de dispositivo independiente. La Figura 145 muestra la asociación del nombre de archivo ARCHY de RPG con la descripción de un archivo del sistema para un dispositivo secuencial. Cuando se ejecuta el programa, el dispositivo de E/S real se especifica en la descripción de ARCHY. Por ejemplo, el dispositivo podría ser PRINTER.

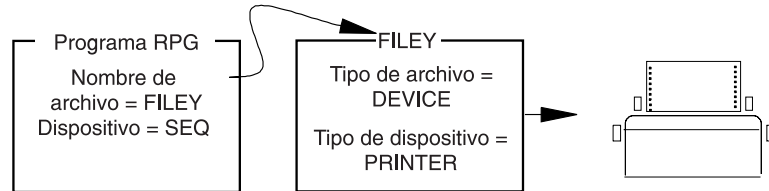


Figura 145. Asociación de un nombre de archivo con la descripción de un archivo de pantalla

Aunque el nombre y el tipo de archivo estén codificados en el programa RPG, en muchos casos puede modificar el tipo de archivo o el dispositivo utilizado en un programa sin modificar el programa. Para averiguar cómo, consulte la “Alteración temporal y redirección de la entrada y salida de archivos” en la página 313.

Denominación de archivos

En el sistema iSeries, los archivos están formados por miembros. Estos archivos se organizan en bibliotecas. La convención para la denominación de archivos es nombre de biblioteca/nombre de archivo.

En un programa ILE RPG, los nombres de archivos se identifican en las posiciones 7 a 16 en las especificaciones de descripción de archivos. Los nombres de archivos pueden tener hasta diez caracteres de longitud y deben ser exclusivos.

No califique el nombre de archivo con una biblioteca dentro de un programa. Durante la ejecución, el sistema busca el archivo en la lista de bibliotecas asociada con el trabajo. Si desea modificar el nombre o el miembro, o especificar una biblioteca determinada, puede utilizar un mandato de alteración temporal de archivos. Consulte la publicación “Alteración temporal y redirección de la entrada y salida de archivos” en la página 313 para obtener más información sobre las alteraciones temporales de archivos.

Tipos de descripciones de archivos

Al identificar la descripción de un archivo que el programa va a utilizar, debe indicar si se trata de un archivo descrito por programa o un archivo descrito externamente.

- Para un **archivo descrito por programa**, la descripción de los campos se codifica dentro del miembro fuente de RPG en las especificaciones de entrada y/o salida. La descripción del archivo ante el sistema operativo incluye información acerca de la procedencia de los datos y la longitud de los registros en el archivo.
- Para un **archivo descrito externamente**, el compilador recupera la descripción de los campos de una descripción de archivo externa que se creó mediante mandatos DDS, IDDU o SQL. Por lo tanto, no tiene que codificar las descripciones de campo en las especificaciones de entrada y/o salida dentro del miembro fuente de RPG.

La descripción externa incluye información sobre la procedencia de los datos, como la base de datos o un dispositivo específico, y una descripción de cada

Tipos de descripciones de archivos

campo y sus atributos. El archivo debe existir y ser accesible desde la lista de bibliotecas antes de compilar el programa.

Los archivos descritos externamente presentan las siguientes ventajas:

- Menor codificación de los programas. Si muchos programas utilizan un mismo archivo, puede definir los campos sólo una vez para el sistema operativo a fin de que todos los programas los utilicen. De este modo deja de ser necesaria la codificación de las especificaciones de entrada y salida para los programas RPG que utilizan archivos descritos externamente.
- Menor actividad de mantenimiento al cambiar el formato de registro del archivo. Puede actualizar los programas frecuentemente cambiando el formato de registro del archivo y recompilando a continuación los programas que utilizan los archivos sin cambiar ninguna codificación en el programa.
- Mejor documentación, ya que los programas que utilizan los mismos archivos utilizan formatos de registro y nombres de campos congruentes.
- Mejor fiabilidad. Si se especifica la comprobación de nivel, el programa en RPG avisará al usuario si hay modificaciones en la descripción externa. Consulte la "Comprobación de nivel" en la página 309 para más información.

Si se especifica un archivo descrito externamente (identificado mediante una E en la posición 22 de las especificaciones de descripción de archivo) para los dispositivos SEQ o SPECIAL, el programa RPG utiliza las descripciones de campo para el archivo, pero la interfaz con el sistema operativo es como si se tratara de un archivo descrito por programa. Los archivos descritos externamente no pueden especificar funciones dependientes de dispositivo como el control de formularios para archivos de impresora (PRINTER) porque esta información ya está definida en la descripción externa.

Utilización de archivos descritos externamente como archivos descritos por programa

Un archivo creado a partir de descripciones externas puede utilizarse como un archivo descrito por programa en el programa. Para utilizar un archivo descrito externamente como si fuera un archivo descrito por programa:

1. Especifique el archivo como descrito por programa (F en la posición 22) en la especificación de descripción de archivo del programa.
2. Describa los campos en los registros de las especificaciones de entrada y/o salida del programa.

en tiempo de compilación, el compilador utiliza las descripciones del campo en las especificaciones de entrada y/o salida. No recupera las descripciones externas.

Ejemplo de algunas relaciones habituales entre programas y archivos

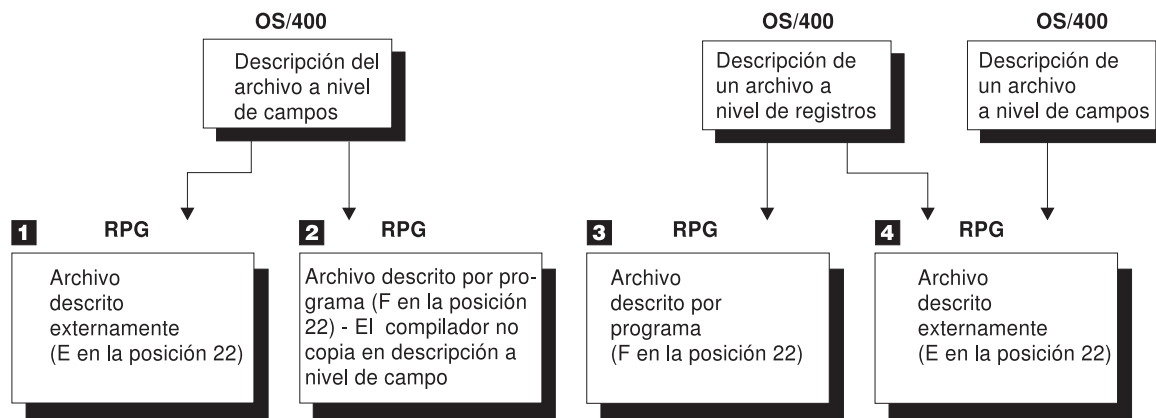


Figura 146. Relaciones habituales entre un programa en RPG y archivos en el sistema iSeries

- 1** El programa utiliza la descripción de un archivo a nivel de campo definida para el sistema operativo. Un archivo descrito externamente se identifica mediante una E en la posición 22 de las especificaciones de descripción de archivo. en tiempo de compilación, el compilador realiza la copia de la descripción externa a nivel de campo.
- 2** Un archivo descrito externamente (es decir, un archivo con una descripción externa a nivel de campo) se utiliza como un archivo descrito por programa en el programa. Un archivo descrito por programa se identifica mediante una F en la posición 22 de las especificaciones de descripción de archivos. Esta entrada indica al compilador que no realice la copia de las descripciones externas a nivel de campo. No es necesario que este archivo exista en tiempo de compilación.
- 3** Se describe un archivo para el sistema operativo sólo a nivel de registro. Los campos del registro se describen en el programa; por lo tanto, la posición 22 de las especificaciones de descripción de archivos debe incluir una F. No es necesario que este archivo exista en tiempo de compilación.
- 4** Un nombre de archivo puede especificarse en tiempo de compilación (es decir, codificarse en el miembro fuente de RPG) y otro nombre de archivo puede especificarse durante la ejecución. La E de la posición 22 de las especificaciones de descripción de archivos indica que la descripción externa del archivo ha de copiarse en tiempo de compilación. durante la ejecución, puede utilizarse un mandato de alteración temporal de archivo para que el programa acceda a un archivo distinto. Para alterar temporalmente un archivo durante la ejecución, debe asegurarse de que el nombre de los registros en ambos archivos sea el mismo. El programa RPG utiliza el nombre del formato de registro en las operaciones de entrada/salida, como por ejemplo una operación READ en la que especifica qué tipo de registro se espera. Consulte la publicación “Alteración temporal y redirección de la entrada y salida de archivos” en la página 313 para obtener más información.

Definición de archivos descritos externamente

Puede utilizar las DDS para describir archivos ante el sistema OS/400. Cada tipo de registro en el archivo se identifica mediante un nombre de formato de registro exclusivo.

Una E en la posición 22 de las especificaciones de descripción de archivo identifica un archivo descrito externamente. La entrada E indica al compilador que debe recuperar la descripción externa del archivo del sistema al compilar el programa.

La información de esta descripción externa incluye:

- Información sobre el archivo, como por ejemplo el tipo de archivo y los atributos del archivo tales como el método de acceso (por clave o por número relativo de registro).
- Descripción del formato de registro, que incluye el nombre del formato de registro y descripciones de campo (nombre, posiciones y atributos).

La información que el compilador recupera de la descripción externa se imprime en la lista del compilador siempre y cuando se haya especificado `OPTION(*EXPDDS)` en el mandato `CRTRPGMOD` o en el mandato `CRTBNDRPG` al compilar el miembro fuente. (El valor por omisión para estos dos mandatos es `OPTION(*EXPDDS)`).

La siguiente sección describe cómo utilizar una especificación de descripción de archivo para denominar o ignorar formatos de registro, y cómo utilizar especificaciones de entrada y salida para modificar descripciones externas. Recuerde que las especificaciones de entrada y de salida para archivos descritos externamente son optativas.

Cambio de nombre de formatos de registro

Muchas de las funciones que puede especificar para archivos descritos externamente (como por ejemplo la operación `CHAIN`) operan sobre el nombre de archivo o sobre un nombre de formato de registro. Por lo tanto, cada nombre de archivo y de formato de registro de un programa debe ser un nombre simbólico exclusivo.

Para cambiar un nombre de formato de registro, utilice la palabra clave `RENAME` en las especificaciones de descripción de archivo para el archivo descrito externamente, tal como aparece en la Figura 147. El formato es `RENAME(nombre anterior:nombre nuevo)`.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++++++
FMAESART   IP   E           K DISK   RENAME(FORMATOART:ARTMAE)
*
```

Figura 147. Palabra clave `RENAME` para nombres de formato de registro en un archivo descrito externamente

La palabra clave `RENAME` se utiliza habitualmente si el programa contiene dos archivos con los mismos nombres de formato de registro. En la Figura 147, se cambia el nombre del formato de registro `ITEMFORMAT` del archivo descrito externamente `ITMMSTL` y el programa pasa a utilizar el nombre `MSTITM`.

Cambio de nombres de campo

Puede cambiar parcialmente el nombre todos los campos de un archivo descrito externamente utilizando la palabra clave PREFIX en la especificación de descripción de archivo para el archivo. Puede añadir un prefijo al nombre de campo existente o puede sustituir parte del nombre de campo existente por una secuencia de caracteres. El formato es PREFIX(*cadena de prefijo*:{*núm_de_car_sustituídos*}). La Figura 148 muestra algunos ejemplos del uso de PREFIX.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++++++
* Añadir el prefijo MST a cada nombre del formato de registro
FMAESART IP E K DISK PREFIX(MAE)
*

* Cambiar el prefijo YTD a YE en cada nombre del formato de registro
FMAEVTA IP E K DISK PREFIX(YE:3)
```

Figura 148. Palabra clave PREFIX para nombres de formato de registro en un archivo descrito externamente

Ignorar formatos de registro

Si un formato de registro de un archivo descrito externamente no tiene que utilizarse en un programa, puede utilizar la palabra clave IGNORE para hacer que el programa se ejecute como si el formato de registro no existiese en el archivo. Para archivos lógicos, esto quiere decir que todos los datos asociados con este formato serán inaccesibles desde el programa. Utilice la palabra clave IGNORE en las especificaciones de descripción de archivos para el archivo descrito externamente tal como se muestra en la Figura 149 en la página 306.

El archivo debe tener más de un formato de registro y no pueden ignorarse todos ellos, por lo menos uno tiene que permanecer. A excepción de este requisito, puede ignorarse cualquier número de formatos de registro en un archivo.

Cuando se ha ignorado un formato de registro, no puede especificarse para ninguna otra palabra clave (SFILE, RENAME o INCLUDE), o incluso IGNORE.

Los nombres de formatos de registro ignorados aparecen en el listado de referencia cruzadas, pero se indica que se han ignorado.

Para indicar que debe ignorarse un formato de registro de un archivo descrito externamente, entre la palabra clave y el parámetro IGNORE(*nombre de formato de registro*) en el campo de palabra clave de las especificaciones de descripción de archivo.

La palabra clave INCLUDE también puede utilizarse para incluir sólo los nombres de formato de registro que van a utilizarse en un programa. Se excluirán el resto de formatos de registro del.

Definición de archivos descritos externamente

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++++++
*
* Suponga que el archivo MAESART contiene los formatos de registro
* siguientes NUMEMPL, NOMBRE, ADDR, TEL, WAGE. Para hacer que el
* programa se ejecute como si sólo existiesen los registros NUMEMPL
* y NOMBRE, puede utilizarse cualquiera de estos dos métodos:
*
FMAESART  UF  E          K DISK  IGNORE(ADDR:TEL:WAGE)
*
* 0:
*
FMAESART  UF  E          K DISK  INCLUDE(NUMEMPL:NOMBRE)
*
```

Figura 149. Palabra clave **IGNORE** para formatos de registro de un archivo descrito externamente

Utilización de las especificaciones de entrada para modificar una descripción externa

Puede utilizar las especificaciones de entrada para alterar temporalmente determinada información de la descripción externa de un archivo de entrada o para añadir funciones de RPG a la descripción externa. En las especificaciones de entrada, puede:

- Asignar indicadores de identificación de registro a formatos de registro tal como se muestra en la Figura 150 en la página 307.
- Cambiar el nombre de un campo tal como se muestra en la Figura 150 en la página 307.
- Asignar indicadores de nivel de control a los campos, tal como se muestra en la Figura 150 en la página 307.
- Asignar valores de campo de comparación a campos para el proceso de registros coincidentes tal como se muestra en la Figura 151 en la página 307.
- Asignar indicadores de campo tal como se muestra en la Figura 151 en la página 307.

No puede utilizar las especificaciones de entrada para alterar temporalmente las ubicaciones de campo de un archivo descrito externamente. Los campos de un archivo descrito externamente se colocan en los registros en el orden en el que aparecen listados en las especificaciones de descripción de datos. Además, las funciones dependientes de dispositivo como por ejemplo el control de formulario no son válidas en un programa RPG para archivos descritos externamente.

Nota: Puede cambiar explícitamente el nombre de un campo de una especificación de entrada, incluso cuando se especifica la palabra clave **PREFIX** para un archivo. El compilador reconocerá (y solicitará) el nombre que se *utilice* en primer lugar en su programa. Por ejemplo, si especifica el nombre con prefijo en una especificación de entrada para asociar el campo con un indicador y a continuación, intenta cambiar el nombre del campo haciendo referencia al nombre sin prefijo, obtendrá un error. Por el contrario, si en primer lugar cambia el nombre del campo con un nombre distinto al nombre con prefijo y a continuación, utiliza el nombre con prefijo en una especificación, obtendrá un error.

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *							
INombreg+++...In.....*							
IMAESTART	01	1					
I.....Campoexter+.....Campo+++++++L1M1..PoNeCe.....							
I	NUMARTI	2	ARTI		L1	3	
*							
IMAESTALM	02						
I	NUMARTI		ARTI		L1		
*							

Figura 150. Alteración temporal y adición de funciones RPG a una descripción externa

- 1** Para asignar un indicador de identificación de registro a un registro de un archivo descrito externamente, especifique el nombre del formato de registro en las posiciones 7 a 16 de las especificaciones de entrada y asigne un indicador válido de identificación de registro en las posiciones 21 y 22. Una manera habitual de utilizar las especificaciones de entrada con archivos descritos externamente es asignando indicadores de identificación de registros.

En este ejemplo, se asigna el indicador de identificación 01 al registro MAESTART y el indicador 02 al registro MAESTALM.

- 2** Para asignar un nuevo nombre a un campo en un registro descrito externamente, tiene que especificar el nombre externo del campo, ajustado por la izquierda, en las posiciones 21 a 30 de la línea de descripción del campo. En las posiciones 49 a 62, especifique el nombre que va a utilizarse en el programa.

En este ejemplo, el nombre de campo NUMARTI de ambos registros se cambia por el nombre ARTI para este programa.

- 3** Para asignar un indicador de nivel de control a un campo de un registro descrito externamente, especifique el nombre del campo en las posiciones 49 a 62 y un indicador de nivel de control en las posiciones 63 y 64.

En este ejemplo, el campo ARTI de ambos registros MAESTART y MAESTALM se especifica para ser el campo de control L1.

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *							
INombarch++SqNORiPos1+NCCPos2+NCCPos3+NCC.....*							
IREGMAE	01	1					
I.....Campoexter+.....Campo+++++++L1M1..PoNeCe.....							
I			NUMCLI		M1	1	
*							
IREGSM	02						
I			NUMCLI		M1		
I			SALDEU			98	2
*							

Figura 151. Adición de funciones de RPG a una descripción externa

- 1** Para asignar un valor de comparación a un campo de un registro descrito externamente, especifique el nombre del formato de registro en las posiciones 7 a 16 de la línea de identificación del registro. En la línea de descripción del campo, especifique el nombre del campo en las posiciones 49 a 62 y asigne un valor de nivel de comparación en las posiciones 65 y 66.

Definición de archivos descritos externamente

En este ejemplo, se ha asignado el valor de nivel de comparación M1 al campo NUMCLI de los registros REGMAE y REGSM.

- 2** Para asignar un indicador de campo a un campo en un registro descrito externamente, debe especificar el nombre del formato de registro en las posiciones 7 a 16 de la línea de identificación del registro. En la línea de descripción del campo, tiene que especificar el nombre del campo en las posiciones 49 a 62 y un indicador en las posiciones 69 a 74.

En este ejemplo, se prueba si el campo SALDEU, en el registro REGSM, es cero cuando se lee e introduce en el programa. Si el valor del campo es cero, se activa el indicador 98.

Utilización de las especificaciones de salida

Las especificaciones de salida son optativas para un archivo descrito externamente. RPG da soporte a los códigos de operación de archivos como WRITE y UPDATE que utilizan la descripción de formato de registro externo para describir el registro de salida sin necesidad de utilizar especificaciones de salida para el archivo descrito externamente.

Puede utilizar especificaciones de salida para controlar cuándo se han de grabar datos, o para especificar campos que se han de grabar. Las entradas válidas para la línea de descripción de campo de un archivo descrito externamente son los indicadores de salida (posiciones 21 a 29), el nombre del campo (posiciones 30 a 43) y un espacio en blanco final (posición 45). Las palabras y los códigos de edición para los campos grabados en un archivo descrito externamente están especificados en las DDS del archivo. Las funciones dependientes de dispositivo como por ejemplo la búsqueda de desbordamiento (posición 18) o espaciado/salto de impresión (posiciones 40 a 51) no son válidas en el programa RPG para archivos descritos externamente. El indicador de desbordamiento tampoco es válido para archivos descritos externamente. Para obtener una descripción del modo de especificar la edición en las DDS, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* de **iSeries 400 Information Center** en este sitio Web:

<http://www.ibm.com/eserver/series/infocenter>.

Si se utilizan especificaciones de salida para un archivo descrito externamente, en las posiciones 7 a 16 se especifica el nombre de formato de registro en lugar del nombre del archivo.

Si todos los campos del archivo descrito externamente deben colocarse en el registro de salida, entre *ALL en las posiciones 30 a 43 de la línea de descripción de campo. Si se especifica *ALL, no puede especificar otras líneas de descripción de campo para este registro.

Si desea colocar sólo determinados campos en el registro de salida, entre el nombre del campo en las posiciones 30 a 43. Los nombres de los campos especificados en estas posiciones tienen que ser los nombres de los campos definidos en la descripción externa del registro, a menos que se haya asignado un nuevo nombre al campo en las especificaciones de entrada. Consulte la Figura 152 en la página 309.

Debe conocer estas consideraciones para utilizar las especificaciones de salida para un archivo descrito externamente:

- En la salida de un registro actualizado, sólo se sitúan en el registro de salida que debe volverse a grabar aquellos campos especificados en las especificaciones del

Definición de archivos descritos externamente

campo de salida que cumplan las condiciones especificadas por los indicadores de salida. Los campos no especificados en las especificaciones de salida se vuelven a grabar utilizando los valores que se han leído. Este procedimiento ofrece un buen método de control en contraposición al código de operación UPDATE que actualiza todos los campos.

- En la creación de un nuevo registro, se sitúan en el registro los campos especificados en las especificaciones de campos de salida. Los campos no especificados en las especificaciones de campos de salida o que no se ajustan a las condiciones especificadas por los indicadores de salida se graban como valores por omisión, que dependen del formato de datos especificado en la descripción externa (por ejemplo: un espacio en blanco para campos de caracteres; cero para campos numéricos).

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
ONombarch++DF..N01N02N03Nomexc+++B++A++Sb+Sa+..... *
OREGART      D      20
0.....N01N02N03Campo++++++YB.Fin++PConstant/palabraedic/DTforma
0
                                *ALL 1
*
OREGVEN      D      30
0
                                NOMVEN 2
0
                                TARCOM
0
                                15      BONOS
*
```

Figura 152. Especificaciones de salida para un archivo descrito externamente

- 1** Para un archivo de actualización, todos los campos del registro se graban en el registro descrito externamente REGART utilizando los valores actuales del programa para todos los campos del registro.

Para la creación de un nuevo registro, se graban todos los campos del registro en el registro descrito externamente REGART, utilizando los valores actuales del programa para todos los campos del registro.

- 2** Para actualizar un registro, se graban los campos NOV MEN y TARCOM en el registro descrito externamente REGVEN cuando está activado el indicador 30. El campo BONOS se graba en el registro REGVEN cuando están activados los indicadores 30 y 15. Todos los demás campos del registro se graban con los valores que se han leído.

Para crear un nuevo registro, se graban los campos NOV MEN y TARCOM en el registro descrito externamente REGVEN cuando está activado el indicador 30. El campo BONOS se graba cuando están activados los indicadores 30 y 15. El resto de campos del registro se graban como valores por omisión, lo cual depende de los tipos de datos (por ejemplo: blanco para campos de caracteres; cero para campos numéricos).

Comprobación de nivel

Los programas en lenguajes de alto nivel (HLL) dependen de la recepción, durante la ejecución, de un archivo descrito externamente cuyo formato corresponda con lo que se ha copiado en el programa en tiempo de compilación. Por este motivo, el sistema proporciona una función de nivel de comprobación que asegura que el formato sea el mismo.

El compilador RPG siempre proporciona la información que la comprobación de nivel necesita cuando se utiliza un archivo DISK, WORKSTN o PRINTER descrito

Definición de archivos descritos externamente

externamente. Puede solicitar la función de comprobación de nivel en los mandatos para crear, cambiar y alterar temporalmente archivos. El valor por omisión del mandato crear archivo es solicitar la comprobación de nivel.

La comprobación de nivel se produce en base al formato de registro cuando se abre el archivo, a menos que especifique LVLCHK(*NO) al emitir un mandato de alteración temporal de archivo o de creación de archivo. Si los valores de comprobación de nivel no coinciden, se comunica el error al programa. A continuación, el programa en RPG se ocupa del error OPEN tal como se describe en la “Capítulo 13. Manejo de excepciones” en la página 251.

El programa en RPG no proporciona comprobación de nivel para archivos descritos por programa ni para archivos que utilicen los dispositivos SEQ o SPECIAL.

Para más información sobre cómo especificar la comprobación de nivel, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseriess/inforcenter>.

Definición de archivos descrito por programa

Los archivos descritos por programa son aquellos cuyos registros y campos están descritos en las especificaciones de entrada/salida del programa que utiliza el archivo. Para utilizar un archivo descrito por programa en un programa en RPG, debe:

1. Identificar el archivo en las especificaciones de descripción del archivos.
2. Si es un archivo de entrada, describir el registro y los campos en las especificaciones de entrada. El nombre del archivo especificado en las posiciones 7 a 16 en las especificaciones de entrada debe ser el mismo que el nombre correspondiente entrado en las especificaciones del archivo.
En las entradas de identificación del registro indique si desea efectuar una comprobación de secuencia de los registros dentro del archivo.
3. Entre el mismo nombre de archivo que en el paso 1 en el campo FACTOR 2 de aquellas especificaciones de cálculo que lo necesitan. Por ejemplo, las operaciones WRITE para un archivo descrito por programa necesitan un nombre de estructura de datos en el campo de resultado.
4. Si es un archivo de salida, describa el registro y los campos en las especificaciones de salida. Además debe especificar cómo va a imprimirse la salida. El nombre del archivo especificado en las posiciones 7 a 16 en las especificaciones de salida debe ser el mismo que el nombre correspondiente entrado en las especificaciones del archivo.

Un archivo descrito por programa debe existir en el sistema y estar en la lista de bibliotecas, antes de ejecutar el programa. Para crear un archivo, utilice uno de los mandatos Crear archivo, que puede encontrar en el apartado *CL and APIs* de la categoría *Programación* de **iSeries 400 Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseriess/inforcenter>.

Operaciones de gestión de datos y ILE RPG Operaciones E/S

La **Gestión de datos** es la parte del sistema operativo que controla el almacenamiento y acceso de los datos por parte de un programa de aplicación. La Tabla 18 en la página 311 muestra las operaciones de gestión de datos que proporciona el sistema iSeries y su correspondiente operación ILE RPG. También

Operaciones de gestión de datos y ILE RPG Operaciones E/S

muestra qué operaciones están permitidas para los diferentes tipos de dispositivo ILE RPG.

Tabla 18. Operaciones de gestión de datos y la operación de E/S en RPG correspondiente

Operación de gestión de datos	Operación de E/S en ILE RPG
OPEN	OPEN
READ	
Por número relativo de registro	READ, CHAIN
Por clave	READ, READE, CHAIN, archivo primario y archivo secundario
Secuencial	READ
Anterior	READP, READPE
Siguiente	READ, READE
Dispositivo solicitado	READ
WRITE-READ	EXFMT
WRITE	
Por número relativo de registro	WRITE
Por clave	WRITE, EXCEPT, archivo primario y secundario
Secuencial	WRITE, EXCEPT
FEOD	FEOD
UPDATE	
Por número relativo de registro	UPDATE, archivo primario y secundario
Por clave	UPDATE, archivo primario y secundario
DELETE	
Por número relativo de registro	DELETE, archivo primario y secundario
Por clave	DELETE, archivo primario y secundario
ACQUIRE	ACQ
RELEASE	REL
COMMIT	COMMIT
ROLLBACK	ROLBK
CLOSE	CLOSE, LR RETURN

Capítulo 16. Consideraciones generales sobre archivos

Este capítulo ofrece información acerca de los aspectos siguientes del proceso de archivos en el sistema iSeries utilizando RPG:

- alteración temporal y redirección de la entrada y salida de archivos
- bloqueo de archivos por un programa en RPG
- bloqueo de registros por un programa en RPG
- compartimiento de una vía de acceso de datos abierta
- las funciones de spooling del sistema iSeries
- utilización de SRTSEQ/ALTSEQ en un programa en RPG como alternativa a un archivo DDS

Alteración temporal y redirección de la entrada y salida de archivos

Pueden utilizarse mandatos del OS/400 para alterar temporalmente un parámetro en la descripción de archivos especificada o para redirigir un archivo durante la compilación o la ejecución. La redirección de archivos le permite especificar durante la ejecución un archivo que sustituya al archivo especificado en el programa (durante la compilación)

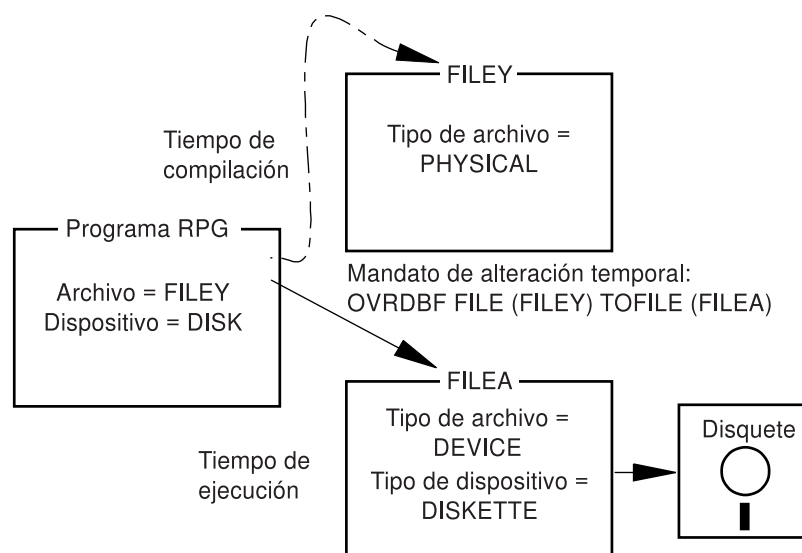


Figura 153. Ejemplo de alteración temporal de la entrada y salida de un archivo

En el ejemplo anterior, el mandato CL OVRDBF (Alterar temporalmente con archivo de base de datos) permite ejecutar el programa con un archivo de dispositivo completamente distinto al que se ha especificado durante la compilación.

Para alterar temporalmente un archivo durante la ejecución, debe asegurarse de que el nombre de los registros en ambos archivos sea el mismo. El programa RPG utiliza el nombre del formato de registro en las operaciones de entrada/salida, como, por ejemplo, una operación READ en la que especifica qué tipo de registro se espera.

Alteración temporal y redirección de la entrada y salida de archivos

No todas las redirecciones de archivos son válidas. Durante la ejecución, las comprobaciones aseguran que las especificaciones del programa RPG sean válidas para el archivo que se está procesando. El sistema OS/400 permite algunas redirecciones de archivos aunque el programa contenga elementos específicos al dispositivo. Por ejemplo, si el nombre del dispositivo RPG es PRINTER, y el archivo real al que está conectado el programa no es una impresora, el sistema OS/400 ignora las especificaciones de espaciado y salto de impresión de RPG.

Existen otras redirecciones de archivo que el sistema OS/400 no permite y que provocan la finalización del programa. Por ejemplo, si el nombre del dispositivo RPG es WORKSTN y en el programa se especifica la operación EXFMT, el programa se detiene si el archivo real al que se conecta el programa no es un archivo de pantalla o ICF.

En ILE, las alteraciones temporales están dentro del ámbito del nivel del grupo de activación, del nivel de trabajo o del nivel de llamada. Las alteraciones temporales que están dentro del ámbito del nivel del grupo de activación permanecen en vigor hasta que se suprimen, sustituyen, o hasta que finaliza el grupo de activación en el que están especificadas. Las alteraciones temporales que están dentro del ámbito del nivel de trabajo permanecen en vigor hasta que se suprimen, sustituyen, o hasta que finaliza el trabajo en el que están especificadas. Esto siempre es así independientemente del grupo de activación en el que se han especificado las alteraciones temporales. Las alteraciones temporales que están dentro del ámbito del nivel de llamada permanecen en vigor hasta que se suprimen, sustituyen, o hasta que finaliza el programa o el procedimiento en el que están especificadas.

El ámbito por omisión para alteraciones temporales es el grupo de activación. Para establecer el ámbito del nivel de trabajo, especifique OVRSCOPE(*JOB) en el mandato de alteración temporal. Para establecer el ámbito del nivel de llamada, especifique OVRSCOPE(*CALLLVL) en el mandato de alteración temporal.

Para obtener más información sobre las redirecciones de archivo válidas y las alteraciones temporales de archivos, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseriess/infocenter>.

La publicación *ILE Concepts* también incluye información sobre las alteraciones temporales y el ámbito del nivel de trabajo comparado con el ámbito del nivel de grupo de activación.

Ejemplo de redirección de entrada y salida de archivos

A continuación se muestra un ejemplo de la utilización de las alteraciones temporales de archivos durante la compilación. Suponga que desea utilizar un archivo descrito externamente para un dispositivo TAPE que no tiene una descripción de nivel de archivo. Debe:

1. Definir un archivo físico denominado FMT1 con un formato de registro que contenga la descripción de cada campo en el formato del registro. El formato de registro se define en las especificaciones de descripción de datos (DDS). Para un dispositivo de cinta, el archivo descrito externamente deberá contener sólo un formato de registro.
2. Crear el archivo denominado FMT1 mediante el mandato CL Crear archivo físico.
3. Especificar el nombre de archivo QTAPE (el nombre del archivo de dispositivo suministrado por IBM para dispositivos de cinta magnética) en el programa en

Alteración temporal y redirección de la entrada y salida de archivos

RPG. Esto identifica el archivo como descrito externamente (indicado por una E en la posición 22 de las especificaciones de descripción de archivos), y especifica el nombre de dispositivo SEQ en las posiciones 36 a 42.

4. Utilizar un mandato de alteración temporal—OVRDBF FILE(QTAPE) TOFILE(FMT1)—durante la compilación para alterar temporalmente el nombre de archivo QTAPE y utilizar en su lugar el nombre de archivo FMT1. Este mandato provoca que el compilador copie la descripción externa del archivo FMT1, que describe el formato del registro al compilador RPG.
5. Crear el programa RPG mediante el mandato CRTBNDRPG o el mandato CRTPGM.
6. Llamar al programa durante la ejecución. La alteración temporal para el archivo FMT1 no debe entrar en vigor mientras se ejecuta el programa. Si la alteración temporal está en vigor, utilice el mandato de CL DLTOVR (Suprimir alteración temporal) antes de llamar al programa.

Nota: puede que necesite utilizar el mandato CL OVRTAPF antes de llamar al programa para proporcionar la información necesaria para abrir el archivo de cintas.

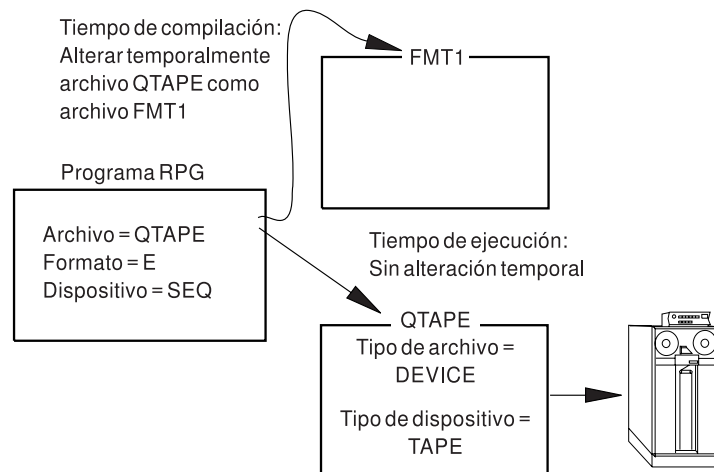


Figura 154. Ejemplo de redirección de entrada y salida de archivos

Bloqueo de archivos

El sistema OS/400 permite que un archivo que se utiliza durante la ejecución de un trabajo pueda pasarse a un estado de bloqueo (exclusivo, de permiso de lectura exclusivamente, compartido para actualización, compartido sin actualización o compartido para lectura). Los programas incluidos en un trabajo no están afectados por los estados de bloqueo de los archivos. Un estado de bloqueo de archivo sólo se produce cuando un programa de otro trabajo intenta utilizar el archivo al mismo tiempo. Puede asignarse el estado de bloqueo de archivo mediante el mandato CL ALCOBJ (Asignar objeto). Para obtener más información sobre la asignación de recursos y estados de bloqueo, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:

<http://www.ibm.com/eserver/iseres/infocenter>.

Cuando abre los archivos, el sistema OS/400 aplica los estados de bloqueo siguientes a los archivos de la base de datos :

Bloqueo de archivo

Tipo de archivo	Estado de bloqueo
Entrada	Compartido para lectura
Actualización	Compartido para actualización
Añadir	Compartido para actualización
Salida	Compartido para actualización

El estado de bloqueo de tipo compartido para lectura permite que otro usuario abra el archivo cuyo estado de bloqueo sea compartido para lectura, compartido para actualización, compartido sin actualización o de permiso de lectura exclusivamente, pero el usuario no puede especificar el uso exclusivo del archivo. El estado de bloqueo de tipo compartido para actualización permite que otro usuario abra el archivo cuyo estado de bloqueo sea compartido para lectura o compartido para actualización.

El programa en RPG establece un estado de bloqueo de permiso de lectura exclusivamente en los archivos de dispositivo. Otro usuario podrá abrir el archivo cuyo estado de bloqueo sea compartido para lectura.

Puede modificarse el estado de bloqueo que el programa en RPG ha aplicado al archivo utilizando el mandato Asignar objeto.

Bloqueo de registros

Cuando un programa lee un registro, lo lee utilizando una de estas dos modalidades: entrada o actualización. Si un programa lee un registro para actualización, se aplica un bloqueo a ese registro. Otro programa no podrá leer el mismo registro para actualizarlo hasta que el primer programa libere ese bloqueo. Si un programa lee un registro para entrada, no se aplica el bloqueo al registro. Si un registro está bloqueado por un programa, otro programa podrá leerlo para entrada.

En los programas en RPG IV, se utiliza un archivo de actualización para leer registros con el fin de actualizarlos. Si un registro se lee desde un archivo no de actualización, sólo podrá leerse para consulta. Por omisión, cualquier registro que se lea desde un archivo de actualización se leerá para su actualización. En los archivos de actualización, puede especificar que un registro se lea para entrada mediante una de las operaciones de entrada CHAIN, READ, READE, READP o REDPE y especificando una extensión de código de operación (N) en el campo del código de operación a continuación del nombre del código de operación.

Cuando un programa RPG IV bloquea un registro, este registro permanece bloqueado hasta que se produzca una de las acciones siguientes:

- se actualice el registro.
- se suprima el registro.
- se lea otro registro del archivo (ya sea para consulta o actualización).
- se realice una operación SETLL o SETGT sobre el archivo.
- se realice una operación UNLOCK sobre el archivo.
- se realice en el archivo una operación de salida definida mediante una especificación de salida que no incluya nombres de campo.

Nota: una operación de salida que añada un registro a un archivo no liberará un bloqueo de registros.

Si el programa lee un registro para actualización y este registro está bloqueado por otro programa del trabajo o por otro trabajo, la operación de lectura esperará hasta que se desbloquee el registro (excepto en el caso de archivos compartidos, consulte el apartado “Compartimiento de una vía de acceso de datos abierta”). Si el tiempo de espera sobrepasa el especificado en el parámetro WAITRCD del archivo, se produce una excepción. Si el programa no acepta esta excepción (RNQ1218), el control pasará al gestor de errores por omisión cuando se sobrepase el tiempo de espera de bloqueo de registro, y se emitirá un mensaje de consulta RNQ1218. Una de las opciones listadas para este mensaje es volver a intentar la operación en la que se excedió el tiempo de espera. Esto hará que vuelva a emitirse la operación en la que se ha sobrepasado el tiempo de espera, con lo cual el programa continuará como si el tiempo de espera excedido del bloqueo de registro no hubiera transcurrido. Observe que si el archivo tiene una INFSR especificada en la que se efectúa una operación de E/S en el archivo antes de que se pase el control al manejador de errores por omisión, pueden producirse resultados imprevistos si la operación de entrada que se vuelve a intentar es una operación secuencial, ya que es posible que se haya modificado el cursor del archivo.

Nota: los subprocedimientos no reciben mensajes de consulta y, por lo tanto, esta situación debe manejarse utilizando un indicador de error en la operación de lectura y comprobando el estado 1218 después de la lectura.

Si no se requieren cambios en un registro bloqueado, puede liberarlo de su estado de bloqueo, sin modificar el cursor del archivo, mediante la operación UNLOCK o procesando las operaciones de salida definidas mediante las especificaciones de salida que no incluyen los nombres de los campos. Estas operaciones de salida se pueden procesar mediante la salida EXCEPT, la salida de detalles o la salida de totales.

(Existen excepciones a estas reglas cuando se opera bajo el control de compromiso. Consulte el apartado “Utilización del control de compromiso” en la página 346 para obtener más información).

Compartimiento de una vía de acceso de datos abierta

Una vía de acceso de datos abierta es la vía a través de la cual se llevan a cabo todas las operaciones de entrada y salida de un archivo. Normalmente se define una vía de acceso de datos abierta independiente cada vez que se abre un archivo. Si especifica SHARE(*YES) cuando crea o altera temporalmente el archivo, la primera vía de datos abierta por el programa para el archivo se comparte con los siguientes programas que abren el archivo concurrentemente.

La posición del registro actual se mantiene en la vía de acceso de datos abierta para todos los programas que utilicen el archivo. Si se lee un registro en un programa y, a continuación, se lee un registro en otro programa llamado, el registro recuperado en la segunda lectura dependerá de si la vía de acceso de datos abierta es compartida. Si la vía de acceso de datos abierta es compartida, la posición del registro actual en el programa llamado queda determinada por la posición actual del programa que realiza la llamada. Si la vía de acceso de datos abierta no es compartida, cada programa tiene una posición independiente para el registro actual.

Si el programa mantiene un bloqueo de registros en un archivo compartido y, a continuación, llama a un segundo programa que lee el archivo compartido para actualizarlo, puede liberar el bloqueo del primer programa:

Compartimiento de una vía de acceso de datos abierta

- efectuando una operación READ sobre el archivo que el segundo programa ha actualizado, o
- utilizando la operación UNLOCK o la operación de lectura sin bloqueo.

En ILE, los archivos compartidos están dentro del ámbito del nivel de trabajo o del nivel del grupo de activación. Un programa que se ejecute en *cualquier* grupo de activación dentro de un trabajo puede compartir los archivos compartidos que están dentro del ámbito del nivel de trabajo. Los programas que se ejecutan en el mismo grupo de activación pueden compartir *sólo* los archivos compartidos que están dentro del ámbito del nivel de grupo de activación.

El ámbito por omisión para archivos compartidos es el grupo de activación. Para establecer el ámbito del nivel de trabajo, especifique OVRSCOPE(*JOB) en el mandato de alteración temporal.

ILE RPG ofrece varias mejoras en el área de ODP compartidos. Si un programa o un procedimiento efectúa una operación de lectura, otro programa o procedimiento puede actualizar el registro siempre que se haya especificado SHARE(*YES) para el archivo. Además, al utilizar archivos de múltiples dispositivos, si un programa adquiere un dispositivo, cualquier programa que comparta ODP también puede utilizar el dispositivo adquirido. Es responsabilidad del programador asegurarse de que todos los datos necesarios para efectuar la actualización estén disponibles en el programa llamado.

Compartir una vía de acceso de datos abierta mejora el rendimiento, ya que el sistema OS/400 no tiene que crear una nueva vía de acceso de datos abierta. Sin embargo, el compartir una vía de acceso de datos abierta puede causar problemas. Por ejemplo, se produce un error en los casos siguientes:

- Si un programa que comparte una vía de acceso de datos abierta intenta operaciones de archivo distintas de las especificadas por la primera apertura (por ejemplo, si intenta realizar operaciones de entrada cuando en la primera apertura sólo se especificaron operaciones de salida).
- Si un programa que comparte una vía de acceso de datos abierta para un archivo descrito externamente intenta utilizar un formato de registro que el primer programa ha ignorado.
- Si un programa que comparte una vía de acceso de datos abierta para un archivo descrito por el programa especifica una longitud de registro que supera la longitud establecida por la primera apertura.

Cuando varios archivos de un programa se alteran temporalmente para un archivo compartido durante la ejecución, el orden de apertura de los archivos es importante. Para controlar el orden de apertura de los archivos, debe utilizar una apertura controlada por el programador o utilizar un programa CL para abrir los archivos antes de llamar al programa.

Si un programa comparte la vía de acceso de datos abierta de un archivo primario o secundario, el programa deberá ejecutar los cálculos detallados del registro que se está procesando antes de llamar a otro programa que comparta la vía de acceso de datos abierta. De otra forma, si se utiliza la consulta anticipada o si la llamada se realiza en tiempo de totales, es posible que al compartir la vía de acceso de datos abierta de un archivo primario o secundario el programa llamado lea los datos de un registro equivocado del archivo.

Debe asegurarse de que cuando se abre por primera vez el archivo compartido, se especifican todas las opciones de apertura que necesitarán las aperturas posteriores

Compartimiento de una vía de acceso de datos abierta

del archivo. Si la primera vez que se abre un archivo compartido no se incluyen las opciones de apertura especificadas para las aperturas posteriores de un archivo, se enviará un mensaje de error al programa.

La Tabla 19 muestra en detalle las opciones de apertura del sistema para cada una de las opciones de apertura que puede especificar.

Tabla 19. Opciones de apertura del sistema permitidas con las opciones de apertura del usuario

Opciones de apertura del usuario de RPG	Opciones de apertura del sistema
INPUT	INPUT
OUTPUT	OUTPUT (archivo creado por programa)
UPDATE	INPUT, UPDATE, DELETE
ADD	OUTPUT (archivo existente)

Para obtener más información sobre cómo compartir una vía de acceso a datos abierta y la comparación entre el ámbito de nivel de trabajo y de grupo de activación, consulte el manual *ILE Concepts*.

Spooling

El spooling es una función del sistema que coloca los datos en un área de almacenamiento a la espera de que se procesen. El sistema iSeries permite utilizar funciones de spooling de entrada y de salida. Cada descripción de archivo de iSeries contiene un atributo de spool que determina si durante la ejecución se utilizan las funciones de spooling para el archivo. El programa en RPG no detecta si se utilizan las funciones de spooling. El dispositivo físico real desde el que se lee un archivo o en el que se graba un archivo está determinado por el lector de spool o por el transcriptor de spool. Para obtener información detallada sobre las funciones de spooling, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iseries/infocenter>.

Spooling de salida

El spooling de salida es válido para trabajos de proceso por lotes e interactivos. La descripción del archivo que el nombre de archivo especifica en el programa en RPG contiene la especificación para las funciones de spooling tal como se indica en el diagrama siguiente:

Spooling

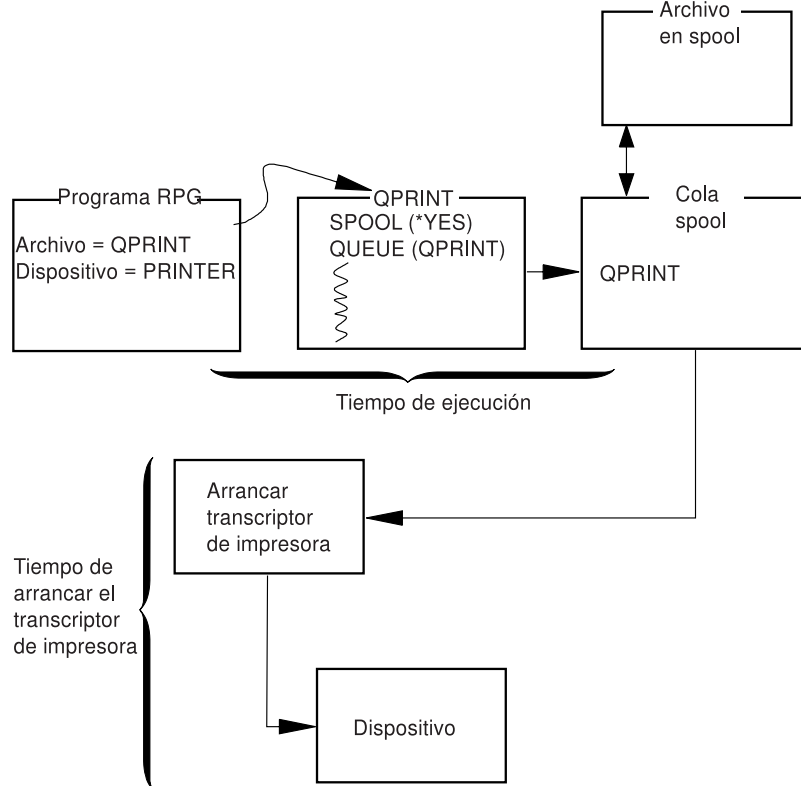


Figura 155. Ejemplo de spooling de salida

Los mandatos de alteración temporal de archivo pueden utilizarse durante la ejecución para alterar temporalmente las opciones especificadas en la descripción del archivo, como el número de copias que se han de imprimir. Además, el soporte de spooling de iSeries también le permite redirigir un archivo después de que un programa lo ejecute. Puede dirigir la misma salida impresa a un dispositivo diferente, como por ejemplo un disquete.

SRTSEQ/ALTSEQ en un programa RPG en relación con un archivo de DDS

Cuando se crea un archivo por clave utilizando SRTSEQ y LANGID, se utiliza el valor especificado para SRTSEQ al comparar las claves de caracteres en el archivo durante las operaciones CHAIN, SETLL, SETGT, READE y READPE. Al crear el programa o el módulo RPG no es necesario que especifique el mismo, u otro, valor SRTSEQ.

Cuando se especifica un valor para SRTSEQ en CRTBNDRPG o CRTRPGMOD, todas las operaciones de comparación de caracteres del programa utilizarán este valor de SRTSEQ. Este valor afecta a la comparación de *todos* los campos, incluidos los campos de claves, los campos de otros archivos y los campos declarados en el programa.

Debe decidir si desea utilizar el valor de SRTSEQ para el programa RPG en base a cómo desea que las operaciones IFxx, COMP y SORTA actúen sobre los datos de caracteres, y no en base a lo que ha especificado al crear los archivos.

Capítulo 17. Acceso a archivos de base de datos

Se puede acceder a un archivo de base de datos desde el programa asociando el nombre del archivo con el dispositivo DISK en la especificación de archivo adecuada.

Los archivos DISK de un programa ILE RPG también se asocian con archivos de gestión de datos distribuidos (DDM), que le permiten acceder a archivos en sistemas remotos como archivos de base de datos.

Archivos de base de datos

Los archivos de base de datos son objetos del tipo *FILE del sistema iSeries. Pueden ser archivos físicos o lógicos y pueden estar descritos externamente o descritos por el programa. Puede acceder a los archivos de base de datos asociando el nombre del archivo con el dispositivo DISK en las posiciones 36 a 42 de las especificaciones de descripción de archivos.

Los archivos de base de datos pueden crearse mediante los mandatos Crear archivo de OS/400. Para obtener más información sobre la descripción y creación de archivos de base de datos, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* de **iSeries 400Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseres/infocenter>.

Archivos lógicos y archivos físicos

Los archivos físicos contienen los datos reales que se almacenan en el sistema y una descripción de cómo se han de presentar o recibir los datos en un programa. Contienen sólo un formato de registro y uno o más miembros. Los registros de los archivos de base de datos pueden estar descritos por el programa o estar descritos externamente.

Un archivo físico puede tener una vía de acceso de secuencia por clave. Esto significa que los datos se presentan a un programa en una secuencia basada en uno o más campos clave del archivo.

Los archivos lógicos no contienen datos. Contienen una descripción de registros que se encuentran en uno o más archivos físicos. Un archivo lógico es una vista o representación de uno o más archivos físicos. Los archivos lógicos que contienen más de un formato reciben el nombre de archivos lógicos de **múltiples formatos**.

Si el programa procesa un archivo lógico que contiene más de un formato de registro, puede utilizar una lectura por formato de registro para establecer el formato que desea utilizar.

Archivos de datos y archivos fuente

Un **archivo de datos** contiene los datos reales o una vista de los datos. Los registros de los archivos de datos se agrupan en miembros. Todos los registros de un archivo pueden estar en un miembro o estar agrupados en miembros diferentes. La mayoría de mandatos y operaciones de base de datos por omisión presuponen que los archivos de base de datos que contienen datos tienen *sólo un miembro*. Esto significa que cuando el programa accede a los archivos de base de datos que

Archivos de base de datos

contienen datos, no es necesario especificar el nombre del miembro del archivo a menos que el archivo contenga más de un miembro. Cuando el archivo contiene más de un miembro y no se ha especificado un miembro determinado, se utiliza el primer miembro.

Generalmente, los archivos de base de datos que contienen programas fuente se componen de más de un miembro. Organizar programas fuente en miembros dentro de archivos de base de datos le permite mejorar el manejo de programas. El **miembro fuente** contiene sentencias fuente que el sistema utiliza para crear objetos programa.

Utilización de archivos DISK descritos de forma externa

Los archivos DISK descritos externamente se identifican mediante una E en la posición 22 de las especificaciones de descripción de archivos. La E indica que el compilador ha de recuperar la descripción externa del archivo desde el sistema cuando se compila el programa. Por lo tanto, debe crearse el archivo antes de compilar el programa.

La descripción externa de un archivo DISK incluye:

- Las especificaciones de formato de registro que contienen una descripción de los campos de un registro.
- Las especificaciones de vía de acceso que describen cómo han de recuperarse los registros.

Estas especificaciones provienen de las DDS del archivo y del mandato de creación de archivos de OS/400 que se ha utilizado para el archivo.

Especificaciones de formato de registro

Las especificaciones de formato de registro le permiten describir los campos de un registro así como la ubicación de los campos en un registro. Los campos están situados en el registro en el orden en que se ha especificado en las DDS. La descripción del campo normalmente incluye el nombre del campo y la longitud del campo (y el número de posiciones decimales si es un campo numérico). En lugar de especificar los atributos de campo en el formato de registro de un archivo físico o lógico, puede definirlos en un archivo de referencias de campo.

Además, pueden utilizarse las palabras clave de las DDS para:

- Especificar que no se admiten valores de clave duplicados para el archivo (UNIQUE)
- Especificar una descripción de texto para un formato de registro o para un campo (TEXT).

Para obtener una lista completa de las palabras clave de las DDS válidas para un archivo de base de datos, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** de este sitio Web: <http://www.ibm.com/eserver/iseres/infocenter>.

En la Figura 156 en la página 323 se muestra un ejemplo de las DDS para un archivo de base de datos y en la Figura 157 en la página 324 para un archivo de referencias de campos que define los atributos de los campos utilizados en el archivo de base de datos. Consulte el sitio Web anterior para obtener más información sobre los archivos de referencias de campos.

Vía de acceso

La descripción de un archivo descrito externamente contiene la vía de acceso que describe cómo han de recuperarse los registros del archivo. Los registros pueden recuperarse basándose en una vía de acceso en secuencia de llegada (sin claves) o en una vía de acceso en secuencia de clave.

La vía de acceso en secuencia de llegada se basa en el orden en el que los registros se almacenan en el archivo. Los registros se añaden al archivo uno tras otro.

Para la vía de acceso en secuencia de clave, la secuencia de registros del archivo se basa en el contenido del campo de clave que se define en las DDS del archivo. Por ejemplo, en las DDS que se muestran en la Figura 156, CUST se define como el campo de clave. La vía de acceso en secuencia de clave se actualiza siempre que se añaden y suprimen registros o cuando cambia el contenido de un campo de clave.

Para obtener una descripción completa de las vías de acceso para un archivo de base de datos descrito de forma externa, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** de este sitio Web:

<http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Las DDS de ejemplo son para el archivo lógico maestro de clientes MAECLIL. El

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.....T.Nombre++++.Lon++TDpB.....Funciones+++++*****
A** ARCHIVO MAECLIL LOGICO MAESTRO CLIENTE
A                                     UNIQUE
A          R REGCLI                  PFILE(CUSMSTP)
A                                     TEXT('Registro maestro cliente')
A          CLIE
A          NOMB
A          DIRE
A          CDAD
A          PROV
A          CP
A          CODIGO
A          TIPCLI
A          SALDO
A          SALDEP
A          ULTIMP
A          ULTIMF
A          LIMCRE
A          VENTAN
A          VTANOA
A          K CLIE
```

Figura 156. Ejemplo de las especificaciones de descripción de datos para un archivo de base de datos

archivo contiene un formato de registro REGCLI (registro maestro de clientes). Los datos de este archivo están contenidos en el archivo físico CUSMSTP, que se identifica mediante la palabra clave PFILE. La palabra clave UNIQUE se utiliza para indicar que no están permitidos los valores de clave duplicados para este archivo. El campo CLIE se identifica mediante una K en la posición 17 de la última línea como el campo de clave de este formato de registro.

Los campos de este formato de registro se listan en el orden en que aparecerán en el registro. Los atributos de los campos se obtienen del archivo físico CUSMSTP. El archivo físico, a su vez, hace referencia a un archivo de referencias de campo para

Utilización de archivos DISK descritos de forma externa

obtener los atributos de los campos. El archivo de referencias de campo se muestra en la Figura 157.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...*
A.....T.Nombre++++RLon++TDpB.....Funciones+++++*****
A**REFCPO  REFDIS  REFERENCIA DISTRIBUCION CAMPOS APLICACION
A          R DSTREF                                TEXT('Distribución ref. campos')
A* CAMPOS COMUNES UTILIZADOS COMO REFERENCIA
A          BASDAT          6  0          EDTCDE(Y) 1
A                                TEXT('Campo base datos')
A* CAMPOS UTILIZADOS POR ARCHIVO MAESTRO CLIENTES
A          CLIE          5          CHECK(MF) 2
A                                COLHDG('Número' 'Cliente')
A                                COLHDG('Nombre Cliente')
A          NOMB          20          REFFLD(NOMB) 3
A          DIRE          R          COLHDG('Dirección Cliente')
A          CDAD          R          REFFLD(NOMB) 3
A                                COLHDG('Ciudad Cliente')
A          PROV          2          CHECK(MF) 2
A                                COLHDG('Provincia')
A          CODIGO          6          CHECK(MF) 2
A                                COLHDG('Código' 'Búsqueda')
A                                TEXT('Búsqueda Núm Cliente +
A                                Código')
A          CP          5  0          CHECK(MF) 2
A                                COLHDG('Código postal' 'Código')
A          TIPCLI          1  0          RANGE(1 5) 4
A                                COLHDG('Cli' 'Tipo')
A                                TEXT('Tipo Cliente 1=Gov 2=Sch+
A                                3=Bus 4=Pvt 5=Otr')
A          SALDO          8  2          COLHDG('Cuent Pag' 'Balance') 5
A                                EDTCDE(J) 6
A          SALDEP          R          REFFLD(SALDO)
A                                COLHDG('Importe C/Cob en' 'Orden +
A                                Archivo')
A          ULTIMF          R          REFFLD(SALDO)
A                                COLHDG('Última' 'Fecha' 'Pago')
A                                TEXT('Último importe pago en C/Cob')
A          LIMCRE          R          REFFLD(Saldo)
A                                COLHDG('Límite' 'Credito')
A                                TEXT('Última fecha pago en C/Cob')
A          VENTAN          R          REFFLD(SALDO)
A                                COLHDG('Crédito' 'Límite')
A                                TEXT('Ventas Clientes este Año')
A          VTAN          R+  2          REFFLD(SALDO)
A                                COLHDG('Ventas 'Este' 'Año')
A                                TEXT('Ventas Clientes Este Año')
A          VTANOA          R+  2          REFFLD(SALDO)
A                                COLHDG('Ventas' 'Último' 'Año')
A                                TEXT('Ventas Clientes Último Año') 7
```

Figura 157. Ejemplo de un archivo de referencias de campos

Este ejemplo de un archivo de referencia de campos muestra las definiciones de los campos que utiliza el archivo MAECLIL (archivo lógico maestro de clientes) que se muestra en la Figura 156 en la página 323. El archivo de referencias de campos contiene normalmente definiciones de campos que utilizan otros archivos. En el texto que se incluye a continuación, se describen algunas de las entradas de este archivo de referencias de campos.

- 1** El campo BASDAT se edita mediante el código de edición Y, como lo indica la palabra clave EDTCDE(Y). Si este campo se utiliza en un archivo de salida descrito externamente para un programa ILE RPG, el código de edición que se utilizará es el especificado en este archivo de referencia de

Utilización de archivos DISK descritos de forma externa

campos; no puede alterarse temporalmente en el programa ILE RPG. Si el campo lo utiliza un programa ILE RPG en un archivo de salida descrito por programa, debe especificarse un código de edición para el campo en las especificaciones de salida.

- 2** La entrada CHECK(MF) especifica que el campo es un campo de relleno obligatorio cuando se entra desde una estación de trabajo. Un campo de relleno obligatorio significa que tienen que entrarse todos los caracteres del campo desde la estación de trabajo.
- 3** Los campos DIRE y CDAD comparten los mismos atributos que se han especificado para el campo NOMB, como lo indica la palabra clave REFFLD.
- 4** La palabra clave RANGE, que se especifica para el campo TIPCLI, indica que los únicos números válidos que pueden entrarse en este campo, desde una estación de trabajo, son los campos del 1 al 5.
- 5** La palabra clave COLHDG proporciona una cabecera de columna para el campo cuando se utiliza con el Programa de Utilidad Interactivo para Bases de Datos (IDU).
- 6** El campo SALDO se edita mediante el código de edición J, como lo indica la palabra clave EDTCDE(J).
- 7** Para algunos campos se facilita una descripción del texto (palabra clave TEXT). La palabra clave TEXT se utiliza a efectos de documentación y aparece en varios listados.

Claves válidas para un registro o archivo

Para una vía de acceso en secuencia de clave, puede definir uno o más campos en las especificaciones de descripción de datos (DDS) para utilizarlos como campos de clave para un formato de registro. (Sin embargo, los campos de longitud variable no pueden utilizarse como campos de clave en un programa RPG). No es necesario que todos los tipos de registro de un archivo tengan los mismos campos de clave. Por ejemplo, un registro de cabecera de pedido puede tener definido el campo PEDIDO como campo de clave y los registros de detalle del pedido pueden tener definidos los campos PEDIDO y LÍNEA como campos de clave.

La clave para un archivo viene determinada por las claves válidas para los tipos de registros de ese archivo. La clave del archivo se determina de la forma siguiente:

- Si todos los tipos de registros de un archivo tienen el mismo número de campos de clave definidos en las DDS cuyos atributos sean idénticos, la *clave para el archivo* está formada por todos los campos de la clave para los tipos de registros. (No es necesario que los campos correspondientes tengan el mismo nombre). Por ejemplo, si el archivo tiene tres tipos de registro y la clave para cada tipo de registro está formada por los campos A, B y C, la clave del archivo estará compuesta por los campos A, B y C. Es decir, la clave del archivo será igual a la clave del registro.
- Si todos los tipos de registros del archivo no tienen los mismos campos de clave, la clave para el archivo se compone de los campos de clave *comunes* a todos los tipos de registros. Por ejemplo, un archivo tiene tres tipos de registros y los campos de clave se han definido de la siguiente forma:
 - REG1 contiene el campo de clave A.
 - REG2 contiene los campos de clave A y B.
 - REG3 contiene los campos de clave A, B y C.

Utilización de archivos DISK descritos de forma externa

La clave del archivo es el campo de clave A; el campo de clave común para todos los tipos de registro.

- Si no hay ningún campo de clave común a todos los tipos de registros, no hay ninguna clave para el archivo.

En un programa ILE RPG, se puede especificar un argumento de búsqueda en determinados códigos de operación de archivos para identificar el registro que desea procesar. El programa ILE RPG compara el argumento de búsqueda con la clave del archivo o del registro y efectúa la operación especificada en el registro cuya clave coincida con el argumento de búsqueda.

Argumentos de búsqueda válidos

Puede especificar un argumento de búsqueda en las operaciones de ILE RPG CHAIN, DELETE, READE, READPE, SETGT y SETLL que especifiquen un nombre de archivo o nombre de registro.

Para una operación con un nombre de archivo, el número máximo de campos que pueden especificarse en un argumento de búsqueda es igual al número total de campos de clave válidos para la clave del archivo. Por ejemplo, si no todos los tipos de registros de un archivo contienen los mismos campos de clave, puede utilizar una lista de claves (KLIST) para especificar un argumento de búsqueda que se componga sólo del número de campos comunes a todos los tipos de registros del archivo. Si un archivo contiene tres tipos de registros, los campos de clave se definen del siguiente modo:

- REC1 contiene el campo de clave A.
- REC2 contiene los campos de clave A y B.
- REC3 contiene los campos de clave A, B y C.

El argumento de búsqueda puede ser únicamente un solo campo con atributos idénticos al campo A, ya que este campo es el único campo de clave común a todos los tipos de registro. El argumento de búsqueda no puede contener un campo de coma flotante, tener longitud variable o que permita nulos.

En una operación con un nombre de registro, el número máximo de campos que pueden especificarse en un argumento de búsqueda es igual al número total de campos de clave válidos para este tipo de registro.

Si el argumento de búsqueda está formado por un campo, puede especificarse un literal, un nombre de campo o un nombre de KLIST con un KFLD. Si el argumento de búsqueda incluye más de un campo (una clave compuesta), debe especificar KLIST con varios KFLD. Para procesar claves de valor nulo, debe utilizarse una KLIST.

Los atributos de cada campo en el argumento de búsqueda deben ser idénticos a los atributos del campo de clave del archivo o registro correspondiente. Los atributos incluyen la longitud, el tipo de datos y el número de posiciones decimales. Los atributos figuran en la tabla de datos de información de campos de claves del listado del compilador. Consulte el ejemplo en el apartado “Información sobre campos de clave” en la página 485.

En todas las operaciones de archivo CHAIN, DELETE, READE, READPE, SETGT y SETLL, también puede especificar un argumento de búsqueda que contenga menos del número total de campos válidos para el archivo o el registro. Un argumento de este tipo hará referencia a una clave parcial.

Referencia a una clave parcial

Las reglas para la especificación de un argumento de búsqueda que haga referencia a una clave parcial son las siguientes:

- El argumento de búsqueda está compuesto por campos que corresponden a los campos más a la izquierda (orden superior) de la clave para el archivo o registro.
- Sólo pueden omitirse los campos más a la derecha de la lista de claves (KLIST) para un argumento de búsqueda que haga referencia a una clave parcial. Por ejemplo, si la clave total para un archivo o registro está compuesta por los campos de clave A, B y C, los argumentos de búsqueda válidos que hacen referencia a una clave parcial son el campo A y los campos A y B.
- Cada campo del argumento de búsqueda debe ser idéntico en cuanto a atributos al campo de clave correspondiente del archivo o registro. Los atributos incluyen la longitud, el tipo de datos, el número de posiciones decimales y el formato (por ejemplo, empaquetado o con zona).
- Un argumento de búsqueda no puede hacer referencia a parte de un campo de clave.

Si un argumento de búsqueda hace referencia a una clave parcial, el archivo se coloca en el primer registro que cumpla el argumento de búsqueda, o el registro recuperado es el primer registro que cumple el argumento de búsqueda. Por ejemplo, las operaciones SETGT y SETLL sitúan al archivo en el primer registro de la vía de acceso que cumple la operación y el argumento de búsqueda. La operación CHAIN recupera el primer registro de la vía de acceso que cumple el argumento de búsqueda. La operación DELETE suprime el primer registro de la vía de acceso que cumple el argumento de búsqueda. La operación READE recupera el siguiente registro si la parte de la clave de este registro (o el registro del tipo especificado) en la vía de acceso coincide con el argumento de búsqueda. La operación READPE recupera el registro anterior si la parte de la clave de este registro (o el registro del tipo especificado) de la vía de acceso coincide con el argumento de búsqueda. Para obtener más información sobre los anteriores códigos de operación, consulte la publicación *ILE RPG Reference*.

Bloqueo y desbloqueo de registros

Por omisión, el compilador RPG desbloquea los registros de entrada y bloquea los registros de salida para mejorar el rendimiento en los archivos SEQ o DISK cuando se cumplen las siguientes condiciones:

1. El archivo está descrito por programa o, si está descrito externamente, sólo tiene un formato de registro.
2. No se utiliza la palabra clave RECNO en la especificación de descripción de archivo.

Nota: Si se utiliza RECNO, el compilador ILE RPG no permitirá bloquear registros. Sin embargo, si se trata de un archivo de entrada y se utiliza RECNO, la Gestión de datos podría bloquear registros si se ha establecido el acceso secuencial rápido. Esto significa que es posible que los registros actualizados no se vean inmediatamente.

3. Una de las condiciones siguientes es cierta:
 - a. El archivo es un archivo de salida.
 - b. Si el archivo es un archivo combinado, es una matriz o un archivo de tablas.
 - c. El archivo es un archivo sólo de entrada; no es un archivo de direcciones de registros o está procesado por un archivo de direcciones de registros; y únicamente utiliza las operaciones de archivo OPEN, CLOSE FEOD y

Utilización de archivos DISK descritos de forma externa

READ. (Es decir, no se permiten las operaciones de archivo siguientes: READE, READPE, SETGT, SETLL y CHAIN.)

El compilador RPG genera el código del programa objeto para bloquear y desbloquear todos los archivos SEQ o DISK que satisfacen las condiciones anteriores. Algunas restricciones del sistema OS/400 pueden impedir el bloqueo y desbloqueo. En dichos casos, no se mejora el rendimiento.

Puede solicitar explícitamente el bloqueo de registros, especificando la palabra clave BLOCK(*YES) en la especificación de descripción de archivo para el archivo. La única diferencia entre el bloqueo de registros por omisión y el bloqueo de registros solicitado por el usuario es que cuando se especifica BLOCK(*YES) para archivos de entrada, pueden utilizarse las operaciones SETLL, SETGT y CHAIN con el archivo de entrada (vea la condición 3c en la página 327 anterior) y continuará llevándose a cabo el bloqueo. Si no se especifica la palabra clave BLOCK y se utilizan estas operaciones, no se llevará a cabo ningún bloqueo de registros.

También puede impedir el bloqueo de registros por omisión especificando la palabra clave BLOCK(*NO) en la especificación de descripción de archivo. Si se especifica BLOCK(*NO), ni el compilador ni la gestión de datos llevarán a cabo un bloqueo de registros. Si no se especifica la palabra clave BLOCK, entonces se llevará a cabo el bloqueo por omisión tal como se ha descrito anteriormente.

La estructura de datos de información de archivos de entrada/salida y específica a los dispositivos no se actualiza después de cada operación de lectura o grabación (excepto para la información de RRN y clave en las lecturas de bloques), para aquellos archivos cuyos registros haya bloqueado y desbloqueado el compilador RPG. El área de información de retorno se actualiza cada vez que se transfiere un bloque de registros. (Para obtener más detalles sobre la estructura de datos de información de archivos, consulte la publicación *ILE RPG Reference*.)

Si impide que el archivo se bloquee y desbloquee, puede obtener información de retorno actualizada válida. Utilice uno de los métodos siguientes para impedir el bloqueo:

- Especifique BLOCK(*NO) en la especificación de descripción de archivo.
- Durante la ejecución, utilice el mandato CL OVRDBF (Alteración temporal con archivo de base de datos) con SEQONLY(*NO) especificado.

Utilización de archivos DISK descritos por programa

Los archivos descritos por programa, que se identifican mediante una F en la posición 22 de las especificaciones de descripción de archivos, pueden describirse como archivos indexados, archivos secuenciales o archivos de direcciones de registro.

Archivo indexado

Un archivo indexado es un archivo DISK descrito por programa cuya vía de acceso se crea con valores de clave. Debe crear la vía de acceso para un archivo indexado utilizando las especificaciones de descripción de datos.

Un archivo indexado se identifica mediante una I en la posición 35 de las especificaciones de descripción de archivos.

Utilización de archivos DISK descritos por programa

Los campos de clave identifican los registros de un archivo indexado. Debe especificar la longitud del campo de clave en las posiciones 29 a 33, el formato del campo de clave en la posición 34 y la posición inicial del campo de clave en la palabra clave KEYLOC de las especificaciones de descripción de archivos.

Un archivo indexado puede procesarse secuencialmente por clave, secuencialmente dentro de unos límites o aleatoriamente por clave.

Argumentos de búsqueda válidos

Para un archivo descrito por programa, un argumento de búsqueda debe ser un solo campo. Para las operaciones CHAIN y DELETE, el argumento de búsqueda debe tener la misma longitud que el campo de clave que se ha definido en las especificaciones de descripción de archivos del archivo indexado. Para el resto de las operaciones de archivo, el argumento de búsqueda puede ser un campo parcial.

Las DDS especifican los campos que han de utilizarse como campos de clave. La palabra clave KEYLOC de las especificaciones de descripción de archivos especifica la posición inicial del primer campo de clave. La entrada en las posiciones 29 a 33 de las especificaciones de descripción de archivos deben especificar la longitud de la clave tal como se ha definido en las DDS.

La Figura 158 y la Figura 159 en la página 330 muestran ejemplos de cómo utilizar las DDS para describir la vía de acceso para archivos indexados.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
A.....T.Nombre++++.Lon++TDpB.....Funciones+++++
A          R FORMATA                                PFILE(DETPED)
A                                          TEXT('Vía de acc. para arch. +
A                                          indexado)
A          CPOA          14
A          PEDIDO        5   0
A          CPOB          101
A          K PEDIDO
A*

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FORDDTLL  IP  F 118      3PIDISK  KEYLOC(15)
F*
```

Figura 158. DDS y especificación de descripción de archivo correspondiente flujo de detalle de RPG IV manejo de excepciones/errores

Debe utilizar especificaciones de descripción de datos para crear la vía de acceso para un archivo indexado descrito por programa.

En las DDS del formato de registro FORMATA del archivo lógico DETPEDL, el campo PEDIDO, que tiene una longitud de cinco dígitos, se define como el campo de clave y está en formato empaquetado. La definición de PEDIDO como el campo de clave establece el acceso por clave a este archivo. Los otros dos campos, CPOA y CPOB, describen el resto de las posiciones de este registro como campos de caracteres.

El archivo de entrada descrito por programa DETPEDL se describe en las especificaciones de descripción de archivos como un archivo indexado. Las posiciones 29 a 33 deben especificar el número de posiciones en el registro necesario para el campo de clave tal como se ha definido en las DDS: tres posiciones. La palabra clave KEYLOC especifica la posición 15 como la posición de inicio del campo de clave en el registro. Puesto que la F en la posición 22 describe

Utilización de archivos DISK descritos por programa

al archivo como descrito por programa, el ILE RPG compilador no recupera la descripción externa a nivel de campo del archivo en tiempo de compilación. Por lo tanto, debe describir los campos del registro en las especificaciones de entrada.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...*
A.....T.Nombre++++.Lon++TDpB.....Funciones+++++
A          R FORMAT                                PFILE(DETPEDP)
A          TEXT('Via de acc. para arch. +
A          Archivo)
A          CPOA          14
A          PEDIDO        5
A          ARTIC         5
A          CPOB          96
A          K PEDIDO
A          K ARTIC
```

Figura 159. (Parte 1 de 2). Utilización de las especificaciones de descripción de datos para definir la vía de acceso (clave compuesta) para un campo indexado

En este ejemplo, las especificaciones de descripción de datos definen dos campos de clave para el formato de registro FORMAT en el archivo lógico DETPEDL. Para utilizar los otros dos campos como una clave compuesta para un archivo indexado descrito por programa, los campos de clave deben ser continuos en el registro.

En las especificaciones de descripción de archivos, la longitud del campo de clave se define como 10 en las posiciones 29 a 33 (el número combinado de posiciones necesario para los campos PEDIDO y ARTIC). La posición de inicio del campo de clave se describe como 15 mediante la palabra clave KEYLOC (comenzando en la posición 44). La posición de inicio debe especificar la primera posición del primer campo de clave.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FDETPEDL IP F 120 10AIDISK KEYLOC(15)
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DNombre+++++ETDsDesde+++A/L+++IDc.Palabrasclave+++++
DCLAV          DS
D K1          1      5
D K2          6     10
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+MameIg....
C          MOVE      PEDIDO      K1
C          MOVE      ARTIC       K2
C  CLAV          CHAIN      DETPEDL          99
```

Figura 160. (Parte 2 de 2). Utilización de las especificaciones de descripción de datos para definir la vía de acceso (clave compuesta) para un archivo indexado

Cuando las DDS especifican una clave compuesta, se debe crear un argumento de búsqueda en el programa para CHAIN en el archivo. (No puede usarse una KLIST para un archivo descrito por programa). Una forma de hacerlo es creando una estructura de datos (utilizando especificaciones de definición) con subcampos iguales a los campos de claves definidos en las DDS. A continuación, en los cálculos, haga que los subcampos sean iguales al valor de los campos de claves y utilice el nombre de la estructura de datos como argumento de búsqueda para la operación CHAIN.

Utilización de archivos DISK descritos por programa

En este ejemplo, las operaciones MOVE definen los subcampos K1 y K2 con el mismo valor que los campos PEDIDO y ARTIC, respectivamente. Seguidamente, se utiliza el nombre de la estructura de datos (CLAV) como el argumento de búsqueda en la operación CHAIN.

Archivo secuencial

Los archivos secuenciales son archivos en los que el orden de los registros del archivo se basa en el orden en que los registros están colocados en el archivo (es decir, en secuencia de llegada). Por ejemplo, el décimo registro colocado en el archivo ocupa la décima posición de registro.

Los archivos secuenciales pueden procesarse al azar utilizando los números relativos de registros, o consecutivamente, según un archivo de direcciones de registros. Puede utilizar los códigos de operación SETLL o SETGT para establecer los límites del archivo.

Archivo de direcciones de registros

Puede utilizar un archivo de direcciones de registros para procesar otro archivo. Un archivo de direcciones de registros puede contener (1) registros de límites que se utilizan para procesar un archivo secuencialmente entre límites o (2) números relativos de registros que se emplean para procesar un archivo mediante números relativos de registro. El propio archivo de direcciones de registros ha de procesarse secuencialmente.

Un archivo de direcciones de registros se identifica mediante R en la posición 18 de las especificaciones de descripción de archivos. Si el archivo de direcciones de registros contiene números relativos de registros, debe haber una T en la posición 35. El nombre del archivo que ha de procesar el archivo de direcciones de registros debe indicarse en la especificación de descripción de archivos. El archivo se identifica mediante la palabra clave RAFFDATA(*nombre de archivo*).

Registros de límites

Para el proceso secuencial entre límites, el archivo de direcciones de registros contiene registros de límites. Un registro de límites contiene la clave de registro más baja y la clave de registro más alta de los registros que contiene el archivo que ha de leerse.

El formato de los registros de límites en el archivo de direcciones de registros es el siguiente:

- La clave inferior empieza en la posición 1 del registro; la clave superior va inmediatamente después de la clave inferior. No puede aparecer ningún espacio en blanco entre las claves.
- Cada uno de los registros del archivo de direcciones de registros puede contener sólo un grupo de límites. La longitud del registro debe ser mayor o igual al doble de la longitud de la clave del registro.
- La clave inferior y la clave superior en el registro de límites deben tener la misma longitud. La longitud de las claves debe ser igual a la longitud del campo de clave del archivo que debe procesarse.
- Una entrada en blanco cuya longitud a la del campo de clave del registro, provoca que el compilador de ILE RPG lea el siguiente registro en el archivo de direcciones de registro.

Utilización de archivos DISK descritos por programa

Números relativos de registros

Para el proceso de números relativos de registros, el archivo de direcciones de registros contiene números relativos de registros. Cada registro recuperado desde el archivo que está procesándose está basado en un número relativo de registro del archivo de direcciones de registros. Un archivo de direcciones de registros que contenga números relativos de registros no puede utilizarse para el proceso entre límites. Cada número relativo de registro del archivo de direcciones de registros es un campo binario de múltiples bytes en el que cada campo contiene un número relativo de registro.

Puede especificar la longitud del archivo de direcciones de registros como 4, 3 o en blanco, dependiendo del origen del archivo. Cuando utilice un archivo de direcciones de registros desde el entorno de iSeries, especifique como 4 la longitud del archivo de direcciones de registros, dado que cada campo tiene 4 bytes de longitud. Cuando utilice un archivo de direcciones de registros creado for the System/36 Environment™, especifique 3 para la longitud del archivo de direcciones de registros, puesto que cada campo tiene 3 bytes de longitud. Si deja en blanco la longitud del archivo de direcciones de registros, el compilador comprobará la longitud del registro primario durante la ejecución para determinar si debe tratarlo como un archivo de direcciones de registros de 3 o de 4 bytes.

Un valor de menos 1 (-1 o hexadecimal :XPH.FFFFFFFF:EXPH.) como número relativo de registro detiene la utilización de ese archivo de direcciones relativas de registros. El fin de archivo se produce cuando se procesan todos los registros del archivo de direcciones de registros.

Métodos para el proceso de archivos DISK

Los métodos de proceso del archivo DISK incluyen:

- Proceso consecutivo
- Proceso secuencial por clave
- Proceso al azar por clave
- Proceso secuencial entre límites.
- Proceso por número relativo de registro

La Tabla 20 en la página 333 muestra las entradas válidas para las posiciones 28, 34 y 35 de la especificación de descripción de archivos para los diversos tipos de archivos y métodos de proceso. A continuación se describe cada uno de los métodos de proceso.

Tabla 20. Métodos de proceso para archivos DISK

Método de proceso	Proceso entre límites (Pos. 28)	Tipo de dirección Tipo (Pos. 34)	Organización de archivos (Pos. 35)
Archivos descritos externamente			
<i>Con claves</i>			
Secuencialmente	En blanco	K	En blanco
Al azar	En blanco	K	En blanco
Secuencialmente entre límites (por arch. dirección registro)	L	K	En blanco
<i>Sin claves</i>			
Al azar/de forma consecutiva	En blanco	En blanco	En blanco
Archivos descritos por programa			
<i>Con claves (archivo indexado)</i>			
Secuencialmente	En blanco	A, D, G, P, T, Z o F	I
Al azar	En blanco	A, D, G, P, T, Z o F	I
Secuencialmente entre límites (por arch. dirección registro)	L	A, D, G, P, T, Z o F	I
<i>Sin claves</i>			
Al azar/de forma consecutiva	En blanco	En blanco	En blanco
Por arch. dirección registro	En blanco	En blanco	En blanco
Como arch. dirección registro (números relativos de registro)	En blanco	En blanco	T
Como archivo de límites de dirección de registro	En blanco	A, D, G, P, T, Z, F o En blanco	En blanco

Proceso consecutivo

Durante el proceso consecutivo, los registros se leen en el orden en que aparecen en el archivo.

Para archivos de entrada y salida que no utilizan funciones al azar (como SETLL, SETGT, CHAIN o ADD), el compilador del ILE RPG toma por omisión u opera como si se hubiera especificado SEQONLY(*YES) en el mandato CL OVRDBF (Alterar temporalmente con archivo de base de datos). (El compilador del ILE RPG no opera como si se hubiera especificado SEQONLY(*YES) para archivos de actualización). SEQONLY(*YES) permite que se coloquen múltiples registros en almacenamientos intermedios de la gestión de datos internos; a continuación, los registros se transfieren al compilador de ILE RPG uno a uno durante la entrada.

Si en el mismo trabajo o grupo de activación dos archivos lógicos utilizan el mismo archivo físico, y un archivo se procesa de forma consecutiva y otro por actualización al azar, puede actualizarse un registro que ya esté en el

Métodos para el proceso de archivos DISK

almacenamiento intermedio que se presenta al programa. En este caso, cuando se procese el registro desde el archivo consecutivo, el registro no reflejará los datos actualizados. Para evitar este problema, utilice el mandato CL OVRDBF y especifique la opción SEQONLY(*NO), que indica que no desea transferir múltiples registros para un archivo que se procese consecutivamente.

Para más información para el proceso sólo secuencial, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas* en **iSeries 400 Information Center** en este sitio Web:
<http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Proceso secuencial por clave

En el método de proceso secuencial por clave, los registros se leen desde el archivo en secuencia de clave.

El método de proceso secuencial por clave es válido para archivos por clave que se utilicen como archivo primario, secundario o de procedimiento completo.

Para archivos de salida o para archivos de entrada que no empleen funciones al azar (como por ejemplo SETLL, SETGT, CHAIN o ADD) y que tengan sólo un formato de registro, el compilador de ILE RPG toma por omisión u opera como si se hubiese especificado SEQONLY(*YES) en el mandato CL OVRDBF. (El compilador ILE RPG no opera como si se hubiera especificado SEQONLY(*YES) para archivos de actualización). SEQONLY(*YES) permite que se coloquen múltiples registros en almacenamientos intermedios de la gestión de datos internos; a continuación, los registros se transfieren al compilador ILE RPG uno a uno durante la entrada.

Si en el mismo trabajo dos archivos utilizan el mismo archivo físico, y uno de ellos se procesa secuencialmente y el otro se procesa al azar para actualización, podría actualizarse un registro que ya se hubiese colocado en el almacenamiento intermedio que se le ha presentado al programa. En este caso, cuando se procese el registro desde el archivo secuencial, el registro no reflejará los datos actualizados. Para evitar este problema, utilice el mandato CL OVRDBF y especifique la opción SEQONLY(*NO), que indica que no desea transferir múltiples registros para un archivo que se procese secuencialmente.

Para más información sobre el proceso sólo secuencial, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:
<http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Ejemplos de proceso secuencial por clave

Los tres ejemplos siguientes le muestran maneras diferentes de utilizar el método de proceso secuencial por clave para procesar datos.

ESPECIFICACIONES DE DESCRIPCIÓN DE DATOS (DDS): La Figura 161 en la página 335 y la Figura 162 en la página 335 muestran las especificaciones de descripción de datos (DDS) para los archivos físicos utilizados en los ejemplos. La Figura 163 en la página 335 muestra las DDS para el archivo lógico utilizado por los tres primeros ejemplos.


```

A*****
A* DESCRIPCION: Estas son las DDS para el archivo físico EMPMSTP.      *
A*               Contiene un formato de registro llamado EMPREC.      *
A*               Este archivo contiene un registro para cada empleado *
A*               de la empresa.                                         *
A*****
A*
A      R EMPREC
A      ENUM          5 0      TEXT('NÚMERO DE EMPLEADO')
A      ENAME         20      TEXT('NOMBRE DE EMPLEADO')
A      ETYPE          1      TEXT('TIPO DE EMPLEADO')
A      EDEPT          3 0      TEXT('DEPARTAMENTO DEL EMPLEADO')
A      ENHRS          3 1      TEXT('HRS SEMAN. NORMALES DEL EMPLEADO')
A      K ENUM

```

Figura 161. DDS para el archivo de base de datos MSTEMP (archivo físico)

```

A*****
A* DESCRIPCION: Estas son las DDS para el archivo físico TRWEEK.*
A*               Contiene el formato de registro RCWEEK. Este      *
A*               archivo contiene todas las entradas semanales      *
A*               en el sistema de informes de horas.                *
A*****
A*
A      R RCWEEK
A      ENUM          5 0      TEXT('NÚMERO DE EMPLEADO')
A      WEEKNO         2 0      TEXT('NÚMERO DE SEMANA DEL AÑO ACTUAL')
A      EHWK           4 1      TEXT('HORAS TRABAJADAS POR EL EMPLEADO')
A      K ENUM
A      K WEEKNO

```

Figura 162. DDS para el archivo de base de datos TRWEEK (archivo físico)

```

A*****
A* ARCHIVOS RELACIONADOS: MSTEMP      (Archivo físico)      *
A*               TRWEEK      (Archivo físico)      *
A* DESCRIPCION: Estas son las DDS para el archivo lógico EMPL1.    *
A*               Contiene dos formatos de registros llamados      *
A*               EMPREC y RCWEEK.                                  *
A*****
A      R EMPREC                                PFILE(MSTEMP)
A      K ENUM
A*
A      R RCWEEK                                PFILE(TRWEEK)
A      K ENUM
A      K WEEKNO

```

Figura 163. DDS para el archivo de base de datos EMPL1 (archivo lógico)

EJEMPLO DE PROGRAMA 1 (Secuencial por clave utilizando archivo primario): En este ejemplo, el registro maestro de empleados (EMPREC) y el registro de horas trabajadas semanalmente (RCWEEK) se incluyen en el mismo archivo lógico EMPL1. El archivo EMPL1 está definido como un archivo de entrada primario y se lee secuencialmente por clave. En las especificaciones de descripción de datos para el archivo, la clave para el registro EMPREC se ha definido como el campo ENUM (número de empleado), y la clave para el registro RCWEEK se ha definido como el campo ENUM más el campo WEEKNO (número de semanas), que es una clave compuesta.

Métodos para el proceso de archivos DISK

```

*****
*NOMBRE PROGRAMA: YTD RPT1
* ARCHIVOS REL.: EMPL1 (archivo lógico)
* PRINT (archivo de impresora)
* DESCRIPCION: Este programa muestra un ejemplo de proceso de
* registros utilizando el método secuencial por clave.
* Este programa imprime la información sobre cada
* empleado y las horas trabajadas semanalmente.
*****
FPRINT O F 80 PRINTER
FEMPL1 IP E K DISK
* A cada registro se asigna un indicador de identificación de registro;
* estos indicadores de identificación de registro se utilizan para
* controlar el proceso de los distintos tipos de registro.
IEMPREC 01
I*
IRCWEEK 02
I*

* Puesto que el archivo EMPL1 se lee secuencialmente por clave, para
* un número de empleado válido, el ENUM de un registro RCWEEK
* debe ser el mismo que el ENUM del último registro EMPREC
* recuperado. Para comprobarlo se guardan los
* ENUM del registro EMPREC en el campo EMPNO y se comparan
* con la lectura de ENUM desde los registros RCWEEK.
* Si el ENUM es válido, se establecerá *IN12. *IN12 se
* utiliza para controlar la impresión del registro RCWEEK.

C SETOFF 12
C 01 MOVE ENUM EMPNO 5 0
C*
C IF (*IN02='1') AND (ENUM=EMPNO)
C SETON 12
C ENDIF

OPRINT H 1P 2 6
O 40 'TRABAJO SEMANAL DEL EMPLEADO'
O 52 'INFORME DE HORAS'
O H 01 1
O 12 'EMPLEADO: '
O 32
O H 01 1
O 12 'NÚMERO DE SERIE: '
O 17
O 27 'DEPARTAMENTO: '
O 30
O 40 'TIPO: '
O 41
O H 01 1
O 20 'NÚMERO SEMANA'
O 50 'HORAS TRABAJADAS'
O D 12 1
O WEEKNO 18
O EHWRK 3 45

```

Figura 164. Proceso secuencial por clave, Ejemplo 1

EJEMPLO PROGRAMA 2 (secuencial por clave utilizando READ): Este ejemplo es el mismo que el ejemplo anterior, excepto que el archivo EMPL1 se ha definido como un archivo de procedimiento completo y la lectura del archivo se efectúa mediante el código de operación READ.

```

*****
*NOMBRE PROGRAMA: YTD RPT2
* ARCHIVOS REL.: EMPL1 (archivo lógico)
* PRINT (archivo de impresora)
* DESCRIPCION: Este programa muestra un ejemplo de proceso de
* registros utilizando el código de operación read.
* Este programa imprime la información sobre cada
* empleado y las horas trabajadas semanalmente.
*****
FPRINT 0 F 80 PRINTER
FEMPL1 IF E K DISK
* Los dos registros (EMPREG y RCWEEK) están en el mismo
* archivo y a cada registro se asigna un indicador de identificación
* de registro. Los indicadores de identificación de registro se utilizan
* para controlar el proceso de los diferentes tipos de registro. No pueden
* especificarse niveles de control o campos coincidentes para un
* archivo de procedimiento completo.
IEMPREG 01
I*
IRCWEEK 02
I*

* El código de operación READ lee un registro del archivo EMPL1.
* Se especifica un indicador de final de archivo en las posiciones 58 y 59.
* Si la operación READ establece el indicador de final de archivo 99,
* el programa se bifurca en el identificador EOFEND y procesa la
* rutina de final de archivo.

C SETOFF 12
C READ EMPL1 99
C 99 GOTO EOFEND
C*
C 01 MOVE ENUM EMPNO 5 0
C*
C IF (*IN02='1') AND (ENUM=EMPNO)
C SETON 12
C ENDIF

* Puesto que EMPL1 se ha definido como un archivo de procedimiento
* completo, debe establecerse el indicador *INLR para terminar el
* programa tras procesar el último registro.

C EOFEND TAG
C 99 SETON LR

```

Figura 165. Proceso secuencial por clave, Ejemplo 2 (Pieza 1 de 2)

Métodos para el proceso de archivos DISK

OPRINT	H	1P		2	6	
0						40 'TRABAJO SEMANAL DEL EMPLEADO
0						52 'INFORME DE HORAS'
0	H	01		1		
0						12 'EMPLEADO: '
0			ENAME			32
0	H	01		1		
0						12 'NÚMERO DE SERIE: '
0			ENUM			17
0						27 'DEPARTAMENTO: '
0			EDEPT			30
0						40 'TIPO: '
0			ETYPE			41
0	H	01		1		
0						20 'NÚMERO SEMANA'
0						50 'HORAS TRABAJADAS'
0	D	12		1		
0			WEEKNO			18
0			EHWRK	3		45

Figura 165. Proceso secuencial por clave, Ejemplo 2 (Pieza 2 de 2)

EJEMPLO PROGRAMA 3 (Técnica de registros coincidentes): En este ejemplo, el archivo TRSEMA se ha definido como archivo de entrada secundario. Los registros EMPREC y RCWEEK se procesan como registros coincidentes, de forma que se ha asignado el valor de registro coincidente L1 al campo ENUM de ambos registros. Los indicadores de identificación de registro 01 y 02 se han asignado a los registros para controlar el proceso para los diferentes tipos de registro.

*NOMBRE PROGRAMA: YTDRPT5									
* ARCHIVOS REL.: MSTEMP (Archivo físico)									
* TRWEEK (Archivo físico)									
* PRINT (Archivo de impresora)									
* DESCRIPCION: Este programa muestra un ejemplo de proceso de									
* registros utilizando el método de registro									
* coincidente. Este programa imprime la información									
* sobre cada empleado, las horas trabajadas									
* semanalmente y las horas extra.									

FPRINT	0	F	80			PRINTER			
FEMPMST	IP	E				K DISK			
FTRWEEK	IS	E				K DISK			
IEMPREC			01						
I						ENUM		M1	
IRCWEEK			02						
I						ENUM		M1	

Figura 166. Proceso secuencial por clave, Ejemplo 3 (Pieza 1 de 2)

C	01		Z-ADD	0		TOTHR	5	1
C	01		Z-ADD	0		TOTOVT	5	1
C	01		SETOFF					12
C*								
C	MR		IF	(*IN02='1')				
C			ADD	EHWRK		TOTHR		
C	EHWRK		SUB	ENHRS		OVT	4	111
C	11		ADD	OVT		TOTOVT		
C			SETON					12
C			ENDIF					
C	OPRINT	H	1P		2	6		
O							50	'YTD RESUMEN NÓMINA'
O		D	01		1			
O							12	'EMPLEADO: '
O			ENAME				32	
O		D	01		1			
O							12	'NÚMERO DE SERIE: '
O			ENUM				17	
O							27	'DEPARTAMENTO: '
O			EDEPT				30	
O							40	'TIPO: '
O			ETYPE				41	
O		D	02 MR		1			
O							8	'NÚMERO SEMANA'
O			WEEKNO				10	
O							32	'HORAS TRABAJADAS = '
O			EHWRK		3		38	
* Estas dos líneas de salida de detalles se procesan si *IN01 está activado								
* y no se encuentran registros coincidentes (es decir, no se encuentran								
* registros RCWEEK para dicho empleado). Obviamente, los campos de totales								
* (TOTHR y TOTOVT) son iguales a cero en este caso.								
O		D	01NMR		1			
O							70	'YTD HORAS TRABAJADAS = '
O			TOTHR		3		78	
O		D	01NMR		1			
O							70	'YTD HORAS EXTRA = '
O			TOTHR		3		78	
* Estas 2 líneas de salida de totales se procesan antes de efectuar								
* los cálculos detallados. Por lo tanto, se imprimirán los campos de								
* totales (TOTHR y TOTOVT) para el empleado en el último registro								
* recuperado si los indicadores especificados están activados.								
O		T	01 12		1			
O		OR	LR 12					
O							70	'YTD HORAS TRABAJADAS = '
O			TOTHR		3		78	
O		T	01 12		1			
O		OR	LR 12					
O							70	'YTD HORAS EXTRA = '
O			TOTOVT		3		78	

Figura 166. Proceso secuencial por clave, Ejemplo 3 (Pieza 2 de 2)

Proceso al azar por clave

Para el proceso al azar por clave, se especifica un argumento de búsqueda que identifica la clave del registro que se ha de leer en el factor 1 de las especificaciones de cálculo para la operación CHAIN. La Figura 168 en la página 341 muestra un ejemplo de un archivo DISK descrito externamente que se está procesando al azar por clave. El registro especificado puede leerse desde el archivo, bien durante el cálculo de detalle o bien durante el cálculo de totales.

Métodos para el proceso de archivos DISK

El método del proceso al azar por clave es válido para un archivo de procedimiento completo especificado como un archivo de entrada o un archivo de actualización.

Para un archivo descrito externamente, la posición 34 de la descripción de especificación de archivos debe contener una K, que indica que el archivo se procesa de acuerdo con una vía de acceso creada con claves.

Las especificaciones de descripción de datos (DDS) para el archivo especifica el campo que contiene el valor de la clave (el campo de clave). La posición 35 de la especificación de descripción de datos debe ser en blanco.

Un archivo descrito por programa debe especificarse como archivo indexado (I en la posición 35), y la posición 34 de la especificación de descripción de archivos debe contener una A, D, G, P, T o Z. La longitud del campo de clave se identifica en las posiciones 29-33 de la especificación de descripción de archivos y la posición de inicio del campo de clave se especifica en la palabra clave KEYLOC. Deben utilizarse especificaciones de descripción de datos para crear la vía de acceso para un archivo de entrada descrito por programa (vea el apartado "Archivo indexado" en la página 328).

Ejemplo de proceso al azar por clave

A continuación se muestra un ejemplo de cómo utilizar el método al azar por clave para procesar datos. La Figura 161 en la página 335 y la Figura 167 muestran las especificaciones de descripción de datos (DDS) para los archivos físicos utilizados por EMSTUPD (Figura 168 en la página 341).

```
A*****
A*  PROG. RELACIONADO:  EMSTUPD                                     *
A*  DESCRIPCIONES: Estas son las DDS para el archivo físico CHANGE. *
A*                      Contiene un formato de registro llamado CHGREC. *
A*                      Este archivo contiene nuevos datos que se utilizan *
A*                      para actualizar el archivo MSTEMP.             *
A*****
A*
A      R CHGREC
A      ENUM          5  0      TEXT('NÚMERO DE EMPLEADO')
A      NNAME         20      TEXT('NOMBRE NUEVO')
A      NTYPE          1      TEXT('TIPO NUEVO')
A      NDEPT          3  0      TEXT('NUEVO DEPARTAMENTO')
A      NNHRS          3  1      TEXT('NUEVAS HORAS SEMANALES NORMALES')
A      K ENUM
```

Figura 167. DDS para el archivo de base de datos CHANGE (archivo físico)

EJEMPLO DE PROGRAMA: En este ejemplo, el archivo MSTEMP se ha definido como un archivo de procedimiento completo de actualización. El archivo de actualización CAMBIO tiene que procesarse por claves. Las DDS para cada uno de los archivos descritos externamente (MSTEMP y CHANGE) identifican al campo ENUM como el campo de clave. Todos los procesos de lectura/actualización se controlan mediante las operaciones establecidas en las especificaciones de cálculo.

```

*****
*NOMBRE PROGRAMA: EMSTUPD
* ARCHIVOS REL.: MSTEMP (Archivo físico)
*                CHANGE (Archivo físico)
*  DESCRIPCION: Este programa muestra el proceso de registros
*                utilizando el método al azar por clave. Se utiliza
*                el código de operación CHAIN.
*                El archivo físico CHANGE contiene todos los
*                cambios efectuados al archivo MSTEMP. Su nombre de
*                formato de registro es CHGREC. Pueden existir
*                algunos campos en el CHGREC que se dejen en blanco,
*                en tal caso, no se producen cambios en dichos
*                campos.
*****
FCHANGE  IP  E          K DISK
FEMPMST  UF  E          K DISK
* Cuando se lee cada registro desde el archivo de entrada primario, CHANGE,
* se utiliza el número de empleado (ENUM) como el argumento de búsqueda
* para encadenar al registro correspondiente en el archivo MSTEMP.
* *IN03 se activará si no se encuentra ningún registro correspondiente,
* esto ocurre cuando se entra un ENUM en el registro CHGREC.
C      ENUM          CHAIN  EMPREC          03
C      03          GOTO  NEXT
C      NNAME          IFNE  *BLANK
C          MOVE      NNAME      ENAME
C          ENDIF
C      NTYPE          IFNE  *EN BLANCO
C          MOVE      NTYPE      ETYPE
C          ENDIF
C      NDEPT          IFNE  *CERO
C          MOVE      NDEPT      EDEPT
C          ENDIF
C      NNHRS          IFNE  *CERO
C          MOVE      NNHRS      ENHRS
C          ENDIF
C          UPDATE  EMPREC
C*
C      NEXT          TAG

```

Figura 168. Proceso al azar por clave de un archivo descrito externamente

Proceso secuencial entre límites

El proceso secuencial entre límites utilizando un archivo de direcciones de registros se especifica mediante una L en la posición 28 de las especificaciones de descripción de archivos y es válido para un archivo con acceso por claves.

Puede especificar el proceso secuencial entre límites para un archivo de entrada o de actualización que se designe como archivo primario, secundario o de procedimiento completo. El archivo puede estar descrito externamente o por programa (indexado). El archivo debe tener las claves en orden ascendente.

Para procesar un archivo secuencialmente entre límites desde un archivo de direcciones de registros, el programa lee:

- Un registro de límites desde el archivo de direcciones de registros
- Registros del archivo que se está procesando entre límites, con claves mayores o iguales que la clave de registro inferior y menores o iguales que la clave de registro superior en el registro entre límites. Si los dos límites suministrados por el archivo de direcciones de registros son iguales, sólo se recuperan los registros con la clave especificada.

Métodos para el proceso de archivos DISK

El programa repite este procedimiento hasta que llega al final del archivo de direcciones de registros.

Ejemplos de proceso secuencial entre límites

La Figura 169 muestra un ejemplo de un archivo indexado que se está procesando secuencialmente entre límites. La Figura 171 en la página 343 muestra el mismo ejemplo con archivos descritos externamente en lugar de archivos descrito por programa.

La Figura 161 en la página 335 muestra las especificaciones de descripción de datos (DDS) para el archivo físico utilizado por el programa ESWLIM1 (Figura 169) y ESWLIM2 (Figura 171 en la página 343).

EJEMPLO PROGRAMA 1 (Proceso secuencial entre límites): El archivo de direcciones de registros LIMITS procesa secuencialmente entre límites (L en la posición 28) MSTEMP. Cada grupo de límites del archivo de direcciones de registros se compone del número de empleado inferior y superior de los registros del archivo MSTEMP que se ha de procesar. Puesto que el archivo por clave de número de empleado (ENUM) tiene una longitud de cinco dígitos, cada grupo de límites se compone de dos claves de cinco dígitos. (Tenga en cuenta que ENUM está en formato empaquetado; por lo tanto, requiere tres posiciones en lugar de cinco).

```
*****
* NOMBRE PROGRAMA: ESWLIM1
* ARCHIVOS REL.: MSTEMP (Archivo físico)
* LIMITS (Archivo físico)
* PRINT (Archivo de impresora)
* DESCRIPCION: Este programa muestra el proceso de un archivo
* indexado secuencialmente entre límites. Este
* programa imprime información para los empleados
* cuyos números de empleado se encuentran dentro
* de los límites establecidos en el archivo LIMITS.
*****
FLIMITS IR F 6 3 DISK RAFDATA(MSTEMP)
FEMPMST IP F 28L 3PIDISK KEYLOC(1)
FPRINT O F 80 PRINTER
* Deben utilizarse especificaciones de entrada para describir los registros
* del archivo descrito por programa MSTEMP.
IEMPMST NS 01
I P 1 3 0ENUM
I 4 23 ENAME
I 24 24 ETYPE
I P 25 26 0EDEPT

* Conforme se procesa un MSTEMP dentro de cada juego de límites, se imprimen
* los correspondientes registros. El proceso del archivo MSTEMP se completa
* cuando archivo de dirección de registro LIMITS llega al final del archivo.

OPRINT H 1P 1
0 12 'NÚMERO SERIE'
0 22 'NOMBRE'
0 45 'DEPARTAMENTO'
0 56 'TIPO'
0 D 01 1
0 ENUM 10
0 ENAME 35
0 EDEPT 45
0 ETYPE 55
```

Figura 169. Proceso secuencial entre límites de un archivo descrito externamente

Métodos para el proceso de archivos DISK

EJEMPLO PROGRAMA 2 (Proceso secuencial entre límites): La Figura 170 muestra las especificaciones de descripción de datos (DDS) para el archivo LIMITS de direcciones de registros utilizado por el programa ESWLIM2 (Figura 171).

```
A*****
A* PROGRAMAS REL.: ESWLIM *
A* DESCRIPCION: Estas son las DDS para el archivo físico *
A* LIMITS. *
A* Contiene un formato de registro denominado LIMIT. *
A*****
A
A      R LIMIT
A      LOW      5 0
A      HIGH     5 0
```

Figura 170. DDS para el archivo de direcciones de registros LIMITS (archivo físico)

Este programa efectúa el mismo trabajo que el programa anterior. La única diferencia es que el archivo físico MSTEMP se ha definido como un archivo descrito externamente en un lugar de como un archivo descrito por programa.

```
*****
* NOMBRE PROGRAMA: ESWLIM2 *
* ARCHIVOS REL.: EMPMST (Archivo físico) *
* LIMITS (Archivo físico) *
* PRINT (Archivo de impresora) *
* DESCRIPCION: Este programa muestra el proceso de un *
* archivo descrito externamente secuencialmente *
* entre límites. *
* Este programa imprime información para los *
* empleados cuyos números de empleado están en *
* los límites establecidos del archivo LIMITS. *
*****
FLIMITS IR F 6 3 DISK RAFDATA(MSTEMP)
FEMPMST IP E L K DISK
FPRINT 0 F 80 PRINTER

* Las especificaciones de entrada son optativas para un archivo descrito
* externamente. En este caso, se define *IN01 como el indicador de
* identificación de registro EMPREC para controlar el
* proceso de este registro.
IEMPREC 01

OPRINT H 1P 1
0 12 'NÚMERO SERIE'
0 22 'NOMBRE'
0 45 'DEPARTAMENTO'
0 56 'TIPO'
0 D 01 1
0 ENUM 10
0 ENAME 35
0 EDEPT 45
0 ETYPE 55
0*
```

Figura 171. Proceso secuencial entre límites de un archivo descrito por programa

Proceso por número relativo de registro

El proceso de entrada o actualización al azar por número relativo de registro sólo se aplica a archivos de procedimiento completo. Mediante el código de operación CHAIN se accede al registro deseado.

Métodos para el proceso de archivos DISK

Los números relativos de registro identifican las posiciones de los registros con relación al principio del archivo. Por ejemplo, los números relativos de registro de los registros primero, quinto y séptimo son 1, 5 y 7, respectivamente.

Para un archivo descrito externamente, el proceso de entrada o actualización por número relativo de registro se determina mediante un espacio en blanco en la posición 34 de las especificaciones de descripción de archivos y el uso del código de operación CHAIN. El proceso de salida mediante un número relativo de registro se determina mediante un espacio en blanco en la posición 34 y el uso de la palabra clave RECNO en la línea de la especificación de descripción de archivos para el archivo.

Utilice la palabra clave RECNO en una especificación de descripción de archivos para especificar un campo numérico que contiene el número relativo de registro que especifica el lugar del archivo donde ha de añadirse un nuevo registro. El campo RECNO debe definirse como numérico con cero posiciones decimales. La longitud de campo debe ser suficiente para contener el número del registro más grande del archivo. Debe especificarse un campo RECNO si han de colocarse nuevos registros en el archivo utilizando especificaciones de salida o una operación WRITE.

Cuando actualice o añada un registro a un archivo por número relativo de registro, el registro debe tener ya un lugar dentro del miembro. Para una actualización, este lugar debe ser un registro existente válido; para un registro nuevo, debe tratarse de un registro suprimido.

Puede utilizar el mandato CL INZPFM para inicializar registros y utilizarlos según el número relativo de registro. El número relativo de registro actual se coloca en el campo RECNO para todas las operaciones de recuperación u operaciones que vuelvan a colocar el archivo (por ejemplo, SETLL, CHAIN o READ).

Operaciones de archivo válidas

La Tabla 21 en la página 345 muestra los códigos de operación de archivos válidos que se permiten en archivos DISK procesados mediante claves y la Tabla 22 en la página 346 muestra los mismos códigos para archivos DISK que se procesan mediante métodos sin claves. Las operaciones que se muestran en estas figuras son válidas para archivos DISK descritos externamente y archivos DISK descritos por programa.

Antes de ejecutar el programa, puede alterar temporalmente un archivo para convertirlo en otro. En concreto, puede alterar temporalmente un archivo secuencial en su programa a un archivo con claves descrito externamente. (El archivo se procesa como un archivo secuencial). También puede alterar temporalmente un archivo con claves del programa para convertirlo en otro archivo con claves, siempre que los campos de clave sean compatibles. Por ejemplo, el archivo de alteración temporal no debe tener un campo de clave menor que el especificado en el programa.

Nota: Cuando se suprime un registro de base de datos, el registro físico se marca como suprimido. Pueden suprimirse registros en un archivo si el archivo se ha inicializado con los registros suprimidos utilizando el mandato Inicializar miembro de archivo físico (INZPFM). Una vez se ha suprimido un registro, no puede leerse. Sin embargo, puede utilizar el número relativo de registro para situarse en el registro y poder escribir encima del contenido del mismo.

Operaciones de archivo válidas

Tabla 21. Operaciones de archivo válidas para métodos de proceso por clave (al azar por clave, secuencial por clave, secuencial entre límites)

Posiciones de las especificaciones de descripción					Posiciones de las especificaciones de cálculo
17	18	20	28 ¹	34 ²	26-35
I	P/S			K/A/P/G/D/T/Z/F	CLOSE, FEOD, FORCE
I	P/S	A		K/A/P/G/D/T/Z/F	WRITE, CLOSE, FEOD, FORCE
I	P/S		L	K/A/P/G/D/T/Z/F	CLOSE, FEOD, FORCE
U	P/S			K/A/P/G/D/T/Z/F	UPDATE, DELETE, CLOSE, FEOD, FORCE
U	P/S	A		K/A/P/G/D/T/Z/F	UPDATE, DELETE, WRITE, CLOSE, FEOD, FORCE
U	P/S		L	K/A/P/G/D/T/Z/F	UPDATE, DELETE, CLOSE, FEOD, FORCE
I	F			K/A/P/G/D/T/Z/F	READ, READE, READPE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F	A		K/A/P/G/D/T/Z/F	WRITE, READ, READPE, READE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F		L	K/A/P/G/D/T/Z/F	READ, OPEN, CLOSE, FEOD
U	F			K/A/P/G/D/T/Z/F	READ, READE, READPE, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F	A		K/A/P/G/D/T/Z/F	WRITE, UPDATE, DELETE, READ, READE, READPE, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
U	F		L	K/A/P/G/D/T/Z/F	READ, UPDATE, DELETE, OPEN, CLOSE, FEOD
O	Blank	A		K/A/P/G/D/T/Z/F	WRITE (añadir registros nuevos a un archivo), OPEN, CLOSE, FEOD
O	En blanco			K/A/P/G/D/T/Z/F	WRITE (carga inicial de un archivo nuevo) ³ , OPEN, CLOSE, FEOD

Notas:

1. Debe especificarse una L en la posición 28 para especificar el proceso secuencial entre límites de un archivo de entrada o actualización mediante un archivo de direcciones de registro.
2. Los archivos descritos externamente requieren una K en la posición 34; los archivos descritos por programa requieren una A, P, G, D, T, Z o F en la posición 34 y una I en la posición 35.
3. No se requiere una A en la posición 20 para la carga inicial de registros en un nuevo archivo. Si se especifica una A en la posición 20, debe especificarse ADD en las especificaciones de salida. El archivo debe haberse creado con el mandato CREAR ARCHIVO de OS/400.

Operaciones de archivo válidas

Tabla 22. Operaciones de archivo válidas para métodos de proceso sin clave (secuencial, Al azar por número de registro relativo, y consecutivo)

Posiciones de las especificaciones de descripción de archivos					Posiciones de especificaciones de cálculo
17	18	20	34	44-80	26-35
I	P/S		En blanco		CLOSE, FEOD, FORCE
I	P/S		En blanco	RECNO	CLOSE, FEOD, FORCE
U	P/S		En blanco		UPDATE, DELETE, CLOSE, FEOD, FORCE
U	P/S		En blanco	RECNO	UPDATE, DELETE, CLOSE, FEOD, FORCE
I	F		En blanco		READ, READP, SETLL, SETGT, CHAIN, OPEN, CLOSE, FEOD
I	F		En blanco	RECNO	READ, READP, SETLL, SETGT,
U	F		En blanco		READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F		En blanco	RECNO	READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
U	F	A	En blanco	RECNO	WRITE (escribir encima de un registro suprimido), READ, READP, SETLL, SETGT, CHAIN, UPDATE, DELETE, OPEN, CLOSE, FEOD
I	R		A/P/G/ D/T/Z/ F/ En blanco ¹		OPEN, CLOSE, FEOD
I	R		En blanco ²		OPEN, CLOSE, FEOD
O	En blanco	A	En blanco	RECNO	WRITE ³ (añadir registros a un archivo), OPEN, CLOSE, FEOD
O	En blanco		En blanco	RECNO	WRITE ⁴ (carga inicial de un nuevo archivo), OPEN, CLOSE, FEOD
O	En blanco		En blanco	En blanco	WRITE (ampliación o carga secuencial de un archivo), OPEN, CLOSE, FEOD
Notas: <ol style="list-style-type: none"> 1. Si la posición 34 está en blanco para un archivo de direcciones de registro entre límites, el formato de las claves en el archivo de direcciones de registro es el mismo que el formato de las claves del archivo que se procesa. 2. Un archivo de direcciones de registro que contenga números relativos de registro requiere una T en la posición 35. 3. El campo RECNO que contiene el número relativo de registro debe establecerse antes de la operación WRITE o si se especifica ADD en las especificaciones de salida. 4. No se requiere una A en la posición 20 para la carga inicial de los registros en un archivo nuevo; sin embargo, si se especifica una A en la posición 20, debe especificarse ADD en las especificaciones de salida. El archivo debe haberse creado con uno de los mandatos de creación de archivos de OS/400. 					

Utilización del control de compromiso

Esta sección describe cómo utilizar el control de compromiso para procesar operaciones de archivo como un grupo. Mediante el control de compromiso, puede asegurar uno de estos dos posibles resultados para las operaciones de archivo:

Utilización del control de compromiso

- todas las operaciones de archivo son satisfactorias (una operación de compromiso)
- ninguna de las operaciones de archivo surte efecto (una operación de retrotracción).

De esta forma, se puede procesar un grupo de operaciones como si se tratase de una unidad.

Para utilizar el control de compromiso, siga estos pasos:

- En el sistema iSeries:
 1. Prepárese para utilizar el control de compromiso. Utilice los mandatos CL CRTJRN (Crear diario), CRTJRNCV (Crear receptor de diario) y STRJRNPF (Arrancar archivo físico de diario).
 2. Comunique al sistema iSeries cuando va a comenzar y a finalizar el control de compromiso: utilice los mandatos CL STRCMTCTL (Comenzar control de compromiso) y ENDCMTCTL (Finalizar control de compromiso). Para obtener más información sobre estos mandatos, consulte el apartado *CL and APIs* de la categoría *Programación* en **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iseres/infocenter>.
- En el programa RPG:
 1. Especifique el control de compromiso (COMMIT) en las especificaciones de descripción de archivos de los archivos que desea tener bajo control de compromiso.
 2. Utilice el código de operación COMMIT (Comprometer) para aplicar un grupo de cambios a archivos bajo control de compromiso o utilice el código de operación ROLBK (Retrotraer) para eliminar el grupo de cambios pendientes en archivos bajo control de compromiso. Para obtener información sobre la función de retrotraer que realiza el sistema, consulte el manual *Backup and Recovery*.

Nota: El control de compromiso se aplica sólo a archivos de base de datos.

Inicio y final del control de compromiso

El mandato CL STRCMTCTL comunica al sistema que quiere comenzar el control de compromiso.

El parámetro LCKLVL (Nivel de bloqueo) le permite seleccionar el nivel en el que se bloquean los registros bajo el control de compromiso. Consulte el apartado “Bloqueos del control de compromiso” en la página 348 y el manual *CL Programming* para más detalles sobre los niveles de bloqueo.

Puede condicionar el control de compromiso, de forma que la decisión de si procesar, o no, un archivo bajo control de compromiso se tome durante la ejecución. Para más información, consulte el apartado “Especificación del control de compromiso condicional” en la página 351.

Cuando completa un grupo de cambios con una operación COMMIT, puede especificar una etiqueta para identificar el final del grupo. En caso de que el trabajo terminase de forma anormal, esta etiqueta identificativa se graba en un archivo, en una cola de mensajes o en un área de datos de manera que el usuario pueda conocer qué grupo de modificaciones es el último grupo que se ha terminado satisfactoriamente. Especifique este archivo, cola de mensajes o área de datos en el mandato STRCMTCTL.

Utilización del control de compromiso

Antes de llamar a un programa que procese los archivos especificados para control de compromiso, emita el mandato STRCMTCTL. Si llama a un programa que abre un archivo especificado para el control de compromiso antes de emitir el mandato STRCMTCTL, se producirá una anomalía al abrir el archivo.

El mandato CL ENDCMTCTL comunica al sistema que el grupo de activación o el trabajo ha finalizado el proceso de archivos bajo control de compromiso. Para más información sobre los mandatos STRCMTCTL y ENDCMTCTL, consulte el apartado *CL and APIs* de la categoría *Programación* en el **iSeries 400 Information Center** en este sitio Web:

<http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Bloqueos del control de compromiso

En el mandato STRCMTCTL se especifica un nivel de bloqueo, ya sea LCKLVL (*ALL), LCKLVL (*CHG) o LCKLVL (*CS). Cuando el programa esté operando bajo control de compromiso y haya procesado una operación de entrada o de salida en un registro de un archivo bajo el control de compromiso, el control de compromiso bloquea el registro de la siguiente forma:

- El programa puede acceder al registro.
- Otro programa del grupo de activación o del trabajo, que tenga este archivo bajo control de compromiso, puede leer el registro. Si el archivo es un archivo compartido, el segundo programa también puede actualizar el registro.
- Otro programa del grupo de activación o del trabajo que no tenga este archivo bajo control de compromiso no puede leer o actualizar el registro.
- Otro programa de un grupo de activación o de un trabajo distinto que tenga este archivo bajo el control de compromiso puede leer el registro si se ha especificado LCKLVL (*CHG), pero no puede leerlo si se ha especificado LCKLVL (*ALL). Con cualquiera de los niveles de bloqueo, el siguiente programa no puede actualizar el registro.
- Otro programa que no tenga este archivo bajo el control de compromiso y que no esté en el grupo de activación o en el trabajo, puede leer pero no puede actualizar el registro.
- Los bloqueos del control de compromiso son diferentes de los bloqueos normales, dependen del LCKLVL especificado y sólo pueden liberarse mediante las operaciones COMMIT y ROLBK.

Las operaciones COMMIT y ROLBK liberan los bloqueos de los registros. La operación UNLOCK no liberará los registros bloqueados utilizando el control de compromiso. Para más detalles sobre niveles de bloqueo, véase el apartado *CL and APIs* de la categoría *Programación* en **iSeries 400 Information Center** en este sitio Web: <http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Puede limitar el número de entradas que pueden bloquearse bajo el control de compromiso antes de que las operaciones COMMIT o ROLBK sean necesarias. Para más información, consulte el manual *Backup and Recovery*.

Nota: Las operaciones SETLL y SETGT bloquearán un registro en los mismos casos en los que una operación de lectura (no de actualización) bloquea un registro para el control de compromiso.

Ámbito del control de compromiso

Cuando se inicia el control de compromiso utilizando el mandato STRCMTCTL, el sistema crea una **definición de compromiso**. Una definición de compromiso contiene información relativa a los recursos que están cambiándose bajo el control

Utilización del control de compromiso

de compromiso dentro de ese trabajo. Sólo el trabajo que emitió el mandato STRCMTCTL conoce la definición de compromiso; ésta finaliza cuando se emite el mandato ENDCMTCTL.

El ámbito para la definición de compromiso indica qué programas del trabajo utilizan esa definición de compromiso. El ámbito de una definición de compromiso puede darse a nivel de grupo de activación o a nivel de trabajo.

El ámbito por omisión para una definición de compromiso se da en el grupo de activación del programa que emite el mandato STRCMTCTL, es decir, a nivel de grupo de activación. Sólo los programas que se ejecutan dentro de ese grupo de activación utilizarán esa definición de compromiso. Los programas OPM utilizarán la definición de compromiso *DFTACTGRP. Los programas ILE utilizarán el grupo de activación con el que estén asociados.

Especifique el ámbito para una definición de compromiso en el parámetro de ámbito de compromiso (CMTSCOPE) del mandato STRCMTCTL. Para obtener más información sobre el ámbito de control de compromiso dentro de ILE, consulte el apartado "Ámbito de gestión de datos" en *ILE Concepts*.

Especificación de archivos para el control de compromiso

Para indicar que un archivo DISK se ha de ejecutar bajo control de compromiso, ponga la palabra clave COMMIT en el campo de palabra clave de la especificación de descripción de archivos.

Cuando un programa especifica control de compromiso para un archivo, la especificación sólo es aplicable a las operaciones de entrada y salida que este programa realice para este archivo. El control de compromiso no es aplicable a operaciones que no sean de entrada y salida. Tampoco es aplicable a archivos en los que no se especifique control de compromiso en el programa que efectúa la operación de entrada y salida.

Cuando más de un programa acceda a un archivo como un archivo compartido, todos o ninguno de los programas deben especificar que el archivo estará bajo control de compromiso.

Utilización de la operación COMMIT

La operación COMMIT indica al sistema que ha completado un grupo de cambios en los archivos bajo control de compromiso. La operación ROLBK elimina el grupo actual de cambios en los archivos que están bajo control de compromiso. Para obtener información sobre cómo especificar estos códigos de operación y qué hace cada código de operación, consulte el manual *ILE RPG Reference*.

Si se produce una anomalía en el sistema, éste emite implícitamente una operación ROLBK. Puede comprobar la identidad del último grupo de cambios completado satisfactoriamente utilizando la etiqueta que se especificó en el factor 1 del código de operación COMMIT y el objeto de la comunicación que haya especificado en el mandato STRCMTCTL.

Al final de un grupo de activación o de un trabajo, o al emitir el mandato ENDCMTCTL, el sistema OS/400 emite un ROLBK implícito que elimina cualquier modificación desde la última operación ROLBK o COMMIT que emitió. Para asegurarse de que todas las operaciones de archivo surtan efecto, emita una operación COMMIT antes de finalizar un grupo de activación o un trabajo que opere bajo el control de compromiso.

Utilización del control de compromiso

La operación OPEN permite realizar operaciones de entrada y salida sobre un archivo, y la operación CLOSE impide que puedan realizarse operaciones de entrada y salida en un archivo. Sin embargo, las operaciones OPEN y CLOSE no afectan a las operaciones COMMIT y ROLBK. Una operación COMMIT o ROLBK afecta a un archivo, incluso después de que el archivo se haya cerrado. Por ejemplo, el programa puede incluir los pasos siguientes:

1. Emitir COMMIT (para archivos ya abiertos bajo control de compromiso).
2. Abrir un archivo especificado para el control de compromiso.
3. Realizar algunas operaciones de entrada y salida en este archivo.
4. Cerrar el archivo.
5. Emitir ROLBK.

Los cambios realizados en el paso 3 se retrotraen mediante la operación ROLBK del paso 5, aunque el archivo se haya cerrado en el paso 4. La operación ROLBK se podría emitir desde otro programa en el mismo grupo de activación o en el mismo trabajo.

Un programa no tiene que operar todos sus archivos bajo control de compromiso; si lo hace, su rendimiento podría verse afectado de forma adversa. Las operaciones COMMIT y ROLBK no surten ningún efecto sobre los archivos que no se encuentren bajo control de compromiso.

Nota: Cuando hay múltiples dispositivos conectados a un programa de aplicación, y el control de compromiso para los archivos que utiliza este programa está vigente, las operaciones COMMIT o ROLBK continúan funcionando en base al archivo y no en base al dispositivo. Puede que la base de datos se actualice con los bloques COMMIT que se hayan completado parcialmente, o que se eliminen los cambios que hayan completado otros usuarios. Es responsabilidad del usuario asegurarse de que esto no ocurra.

Ejemplo de utilización del control de compromiso

Este ejemplo muestra las especificaciones y los mandatos CL necesarios para que un programa opere bajo control de compromiso.

Para preparar la utilización del control de compromiso, emita los siguientes mandatos CL:

1. CRTJRNRCV JRNRCV (RECEPTOR)
Este mandato crea un receptor de diario con el nombre RECEPTOR.
2. CRTJRN JRN(DIARIO) JRNRCV(RECEPTOR)
Este mandato crea un diario llamado DIARIO y lo adjunta al receptor de diario llamado RECEPTOR.
3. STRJRNPF FILE(MAESTRO TRANS) JRN(DIARIO)
Este mandato dirige las entradas de diario del archivo MAESTRO y del archivo TRANS al diario DIARIO.

En el programa, especifique COMMIT para el archivo MAESTRO y el archivo TRANS:


```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FMAESTRO  UF  E    K    DISK  COMMIT
FTRANS    UF  E    K    DISK  COMMIT
F*

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+MameIg....
C          :
C          :
*
*  Utilice la operación COMMIT para completar un grupo de
*  operaciones si han sido satisfactorias o ROLBK para retrotraer las
*  modificaciones si no han sido satisfactorias.
*
C          UPDATE    REG_MAES                      90
C          UPDATE    REG_TRAN                      91
C          IF        *IN90 0 *IN91
C          ROLBK
C          ELSE
C          COMMIT
C          ENDIF

```

Figura 172. Ejemplo de utilización del control de compromiso

Para operar con el programa llamado REVISAR bajo control de compromiso, emita los mandatos:

1. STRCMTCTL LCKLVL(*ALL)

Este mandato inicia el control de compromiso con el nivel de bloqueo más alto.

2. CALL REVISE

Este mandato llama al programa REVISAR.

3. ENDCMTCTL

Este mandato finaliza el control de compromiso y provoca la ejecución de una operación de retrotracción implícita.

Especificación del control de compromiso condicional

Puede escribir un programa de forma que la decisión de abrir un archivo bajo control de compromiso se tome durante la ejecución. Sim implanta el control de compromiso condicional, no tendrá que escribir y mantener dos versiones del mismo programa: una que opere bajo control de compromiso y otra que no opere bajo control de compromiso.

La palabra clave COMMIT tiene un parámetro optativo que le permite especificar el control de compromiso condicional. Entre la palabra clave COMMIT en la sección de palabras clave de las especificaciones de descripción de archivos para un archivo o archivos determinados. El compilador ILE RPG define implícitamente un campo de caracteres de un byte con el mismo nombre que el especificado como parámetro. Si el parámetro se establece en '1', el archivo se ejecutará bajo control de compromiso.

El parámetro de la palabra clave COMMIT debe establecerse antes de abrir el archivo. Puede establecer el parámetro transfiriendo un valor cuando llama al programa o estableciendo explícitamente el parámetro en '1' en el programa.

Para aperturas compartidas, si el archivo ya está abierto, el parámetro de la palabra clave COMMIT no entra en vigor, aunque se haya establecido en '1'.

Utilización del control de compromiso

La Figura 173 es un ejemplo que muestra el control de compromiso condicional.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FMAESTRO  UF  E    K    DISK  COMMIT(INDICOMPR)
FTRANS   UF  E    K    DISK  COMMIT(INDICOMPR)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CLON01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+MameIg....
* Si COMITFLAG = '1' los archivos se abren bajo control de
* compromiso, de lo contrario no se abren.
C      *ENTRY          PLIST
C                      PARM                      INDICOMPR
C                      :
C                      :
C      *
*
* Utilice la operación COMMIT para completar un grupo de
* operaciones si han sido satisfactorias o ROLBK para retrotraer las
* modificaciones si no han sido satisfactorias. Emita únicamente
* COMMIT o ROLBK si se han abierto los archivos para control de
* compromiso (es decir, COMITFLAG = '1')
*
C                      UPDATE    REG_MAES                      90
C                      UPDATE    REG_TRAN                      91
C                      IF        INDICOMPR = '1'
C                      IF        *IN90 0 *IN91
C                      ROLBK
C                      ELSE
C                      COMMIT
C                      ENDIF
C                      ENDIF
C*
```

Figura 173. Ejemplo de utilización del control de compromiso condicional

Control de compromiso en el ciclo del programa

El control de compromiso está pensado para archivos de procedimiento completo, en los que el usuario controla la entrada y la salida. No utilice el control de compromiso con archivos primarios y secundarios, en los que el ciclo del programa en RPG es el que controla la entrada y la salida. He aquí algunas de las razones para esta recomendación:

- No puede emitir una operación COMMIT para la última salida de totales del programa.
- Es difícil la programación dentro del ciclo para la recuperación desde una condición de registro bloqueado.
- La operación ROLBK no restablece los indicadores de nivel.
- Después de una operación ROLBK, el proceso de registros coincidentes puede provocar un error de secuencia.

Archivos DDM

Los programas ILE RPG acceden a archivos de sistemas remotos a través de la **gestión de datos distribuidos (DDM)**. DDM permite a programas de aplicación de un sistema utilizar archivos almacenados en un sistema remoto como si fueran archivos de base de datos. No se requieren sentencias especiales en los programas ILE RPG para dar soporte a archivos DDM.

Un **archivo DDM** lo crea un usuario o programa en un sistema local (origen). Este archivo (cuyo tipo de objeto es *FILE) identifica un archivo que se guarda en un sistema remoto (destino). El archivo DDM proporciona la información necesaria para que un sistema local localice un sistema remoto y para acceder a los datos del archivo origen. Para más información sobre la utilización y creación de archivos DDM consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web: <http://publib.boulder.ibm.com/html/as400/infocenter.html>.

Utilización de archivos DDM anteriores a la Versión 3, Release 1

Si está utilizando un archivo DDM anterior a la Versión 3 Release 1.0, la comparación de claves no se realiza a nivel de Gestión de datos durante una operación READE o READPE, indicador EQ para SETLL, ni durante el proceso secuencial entre límites por un archivo de direcciones de registros. En su lugar, la operación READE o READPE, indicador EQ para SETLL o el proceso secuencial entre límites por un archivo de direcciones de registros compararán las claves utilizando el orden de clasificación *HEX.

Esto puede provocar resultados inesperados cuando se utilizan características DDS que provocan que más de un argumento de búsqueda coincida con una determinada clave en el archivo. Por ejemplo, si se utiliza ABSVAL en una clave numérica, tanto -1 como 1 podrían utilizarse como argumentos de búsqueda para una clave en el archivo con un valor de 1. Si se utilizara el orden de clasificación hexadecimal, un argumento de búsqueda de -1 no tendría éxito para una clave real de 1. Algunas de las características DDS que provocan que la comparación por clave sea diferente son:

- Haber especificado ALTSEQ para el archivo
- Las palabras clave ABSVAL, ZONE, UNSIGNED o DIGIT en campos de clave
- Los campos clave de Longitud variable, Fecha, Hora o Indicación de la hora
- El valor de SRTSEQ para el archivo no es *HEX
- Se especificó ALWNULL(*USRCTL) en el mandato de creación y una clave del registro o el argumento de búsqueda tiene un valor nulo (esto es válido sólo para archivos descritos externamente)

Además, si el signo de un campo numérico es diferente del signo preferido por el sistema, la comparación por clave también será diferente.

La primera vez que la comparación de claves no se realiza a nivel de Gestión de datos en un archivo DDM anterior a la Versión 3 Release 1 durante la operación READE o READPE, indicador EQ para SETLL o durante el proceso secuencial entre límites por un archivo de direcciones de registros, se emitirá un mensaje informativo (RNI2002).

Nota: El rendimiento de las operaciones de E/S para las que puede no encontrarse un registro (SETLL, CHAIN, SETGT, READE, READPE) será más lento que el rendimiento en de las operaciones anteriores a la Versión 3 Release 1.0.

Capítulo 18. Acceso a dispositivos conectados externamente

Se puede acceder a dispositivos conectados externamente desde RPG utilizando los archivos de dispositivo. Los **archivos de dispositivo** son archivos que permiten el acceso a hardware conectado externamente, como impresoras, unidades de cinta, unidades de disquete, estaciones de pantalla y otros sistemas que están conectados mediante una línea de comunicaciones.

Este capítulo describe cómo acceder a dispositivos conectados externamente utilizando nombres de dispositivos de RPG como PRINTER, SEQ y SPECIAL. Para obtener información sobre estaciones de pantalla y dispositivos ICF, consulte el “Capítulo 19. Utilización de archivos WORKSTN” en la página 371

Tipos de archivos de dispositivo

Antes de que el programa pueda leer o escribir en los dispositivos del sistema, debe crearse una descripción de dispositivo que identifique las posibilidades de hardware del dispositivo ante el sistema operativo cuando se configure el dispositivo. Un archivo de dispositivo especifica cómo puede utilizarse un dispositivo. Haciendo referencia a un archivo de dispositivo específico, el programa RPG utiliza el dispositivo de la manera en que se ha descrito al sistema. El archivo de dispositivo da formato a los datos de salida desde el programa RPG para enviarlos al dispositivo, y a los datos de entrada desde el dispositivo para enviarlos al programa RPG.

Utilice los archivos de dispositivo que aparecen en la Tabla 23 para acceder a los dispositivos asociados conectados externamente:

Tabla 23. Archivos de dispositivo de AS/400, Mandatos CL relacionados y Nombre de dispositivo de RPG

Archivo de dispositivo	Dispositivo asociado conectado externamente	Mandatos CL	Nombre de dispositivo de RPG
Archivos de impresora	Proporcionan acceso a los dispositivos de impresora y describen el formato de la salida impresa.	CRTPRTF CHGPRTF OVRPRTF	PRINTER
Archivos de cinta	Proporcionan acceso a los archivos de datos que están almacenados en dispositivos de cinta.	CRTTAPF CHGTAPF OVRTAPF	SEQ
Archivos de disquete	Proporcionan acceso a los archivos de datos que están almacenados en dispositivos de disquete.	CRTDKTF CHGDKTF OVRDKTF	DISK
Archivos de pantalla	Proporcionan acceso a dispositivos de pantalla.	CRTDSPF CHGDSPF OVRDSPF	WORKSTN
Archivos ICF	Permitir a un programa de un sistema comunicarse con un programa del mismo sistema u otro sistema.	CRTICFF CHGICFF OVRICFF	WORKSTN

El archivo de dispositivo contiene la descripción de archivo, que identifica el dispositivo que se ha de utilizar; no contiene datos.

Acceso a dispositivos de impresora

Los archivos PRINTER de los programas ILE RPG se asocian con los archivos de impresora del sistema AS/400:

Los archivos de impresora le permiten imprimir archivos de salida. Este capítulo proporciona información sobre cómo especificar y utilizar archivos de impresora en programas en ILE RPG.

Especificación de archivos PRINTER

Para indicar que desea que el programa acceda a archivos de impresora, especifique PRINTER como el nombre de dispositivo para el archivo en la especificación de descripción de archivos. Cada archivo debe tener un nombre de archivo exclusivo. Se permiten un máximo de ocho archivos de impresora por programa.

Los archivos PRINTER pueden estar descritos externamente o por programa. En un archivo PRINTER descrito externamente no se permiten los indicadores de desbordamiento OA-OG y OV, la búsqueda de desbordamiento, las entrada de espacio/salto y la palabra clave PRTCTL. Consulte la publicación *ILE RPG Reference* para obtener las entradas de especificación de salida válidas para un archivo descrito externamente.

Para un archivo PRINTER descrito externamente puede especificar la palabra clave DDS INDARA. Si intenta utilizar esta palabra clave para un archivo PRINTER descrito por programa, obtendrá un error durante la ejecución.

Puede utilizar el mandato CL CRTPRTF (Crear archivo de impresión) para crear un archivo de impresora o utilizar los nombres de archivo suministrados por IBM.

Para obtener información sobre el mandato CRTPRTF, consulte el apartado *CL and APIs* de la categoría *Programación* en **iSeries 400 Information Center** en este sitio Web: — <http://www.ibm.com/eserver/iseres/infocenter>.

Para obtener información sobre los nombres de archivo proporcionados por IBM y las DDS para archivos de impresora descritos externamente, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en el sitio Web anterior.

Los códigos de operación válidos para un archivo PRINTER son WRITE, OPEN, CLOSE y FEOD. Para obtener una descripción completa de estos códigos de operación, consulte el manual *ILE RPG Reference*.

Manejo del desbordamiento de página

Algo importante que debe tener en cuenta cuando utilice un archivo PRINTER es el desbordamiento de página. Para un archivo PRINTER descrito externamente, el manejo del desbordamiento de página es responsabilidad del usuario. Realice una de las siguientes acciones:

- Especifique un indicador, *IN01 a *IN99, como indicador de desbordamiento utilizando la palabra clave OFLIND(*indicador de desbordamiento*) en el campo de Palabras clave de las especificaciones de descripción de archivos.
- Compruebe el número de línea y el desbordamiento de página en la sección de realimentación del dispositivo de impresora en la INFDS. Consulte la publicación *ILE RPG Reference* para más información.

- Efectúe un recuento del número de líneas de salida por página.
- Compruebe la condición de excepción/error del archivo especificando un indicador en las posiciones 73 y 74 de las especificaciones de cálculo que especifican la operación de salida, o especificando una INFSR que pueda manejar el error. La INFDS contiene información detallada sobre la condición de excepción/error del archivo. Consulte el “Capítulo 13. Manejo de excepciones” en la página 251 para más información sobre el manejo de errores y excepciones.

Puede especificar un indicador, *IN01 a *IN99, para un archivo descrito externamente o descrito por programa utilizando la palabra clave OFLIND(*indicador de desbordamiento*) en las especificaciones de descripción de archivos. Este indicador se activa cuando se imprime una línea en la línea de desbordamiento, o se alcanza o sobrepasa la línea de desbordamiento durante una operación de espaciado o salto de impresión. Utilice el indicador para condicionar su respuesta a la condición de desbordamiento. El indicador no condiciona la lógica de desbordamiento de RPG como lo haría un indicador de desbordamiento (*INOA a *INOG, *INOV). La desactivación del indicador es responsabilidad del usuario.

Tanto para los archivos descritos por programa, como para los descritos externamente, el número de línea y el número de página están disponibles en la sección de realimentación de impresora de la INFDS del archivo. Para acceder a esta información, especifique la palabra clave INFDS en la especificación de archivos. En la especificación, defina el número de línea en las posiciones 367-368 y defina el número de página en las posiciones 369-372 de la estructura de datos. Tanto el campo de número de línea como el de número de página deben estar definidos en binario sin posiciones decimales. Como la INFDS se actualiza después de cada operación de salida hacia el archivo de impresora, estos campos pueden utilizarse para determinar el número de línea y el número de página actuales sin que se necesite una lógica de recuento de líneas del programa.

Nota: Si altera temporalmente un archivo de impresora por otro dispositivo, como un disco, la sección de realimentación de la impresora de la INFDS no se actualizará y la lógica de recuento de líneas no será válida.

Para un archivo PRINTER descrito por programa, las siguientes secciones se aplican a los indicadores de desbordamiento y la lógica de desbordamiento de búsqueda.

Utilización de indicadores de desbordamiento en archivos descritos por programa

Un indicador de desbordamiento (OA a OG, OV) se activa cuando se imprime o sobrepasa la última línea de una página. Un indicador de desbordamiento puede utilizarse para especificar las líneas que se han de imprimir en la página siguiente. Los indicadores de desbordamiento pueden especificarse sólo para archivos PRINTER descritos por programa y se utilizan principalmente para condicionar la impresión de líneas de cabecera. Un indicador de desbordamiento se especifica utilizando la palabra clave OFLIND de las especificaciones de descripción de archivos y puede utilizarse para condicionar operaciones en las especificaciones de cálculo (posiciones 9 a 11) y en las especificaciones de salida (posiciones 21 a 29). Si no se especifica ningún indicador de desbordamiento, el compilador RPG asigna el primer indicador de desbordamiento que no se utilice al archivo PRINTER. Los indicadores de desbordamiento también pueden especificarse como indicadores de resultado en las especificaciones de cálculo (posiciones 71 a 76).

Acceso a dispositivos de impresora

El compilador activa un indicador de desbordamiento sólo la primera vez que se produce una condición de desbordamiento en una página. Se produce una condición de desbordamiento cuando ocurre uno de los casos siguientes:

- Se imprime una línea pasada la línea de desbordamiento.
- Se pasa la línea de desbordamiento durante una operación de espaciado.
- Se pasa la línea de desbordamiento durante una operación de salto.

La Tabla 24 en la página 359 muestra los resultados de la presencia o ausencia de un indicador de desbordamiento en las especificaciones de salida y descripción de archivo.

Las siguientes consideraciones son aplicables a indicadores de desbordamiento que se utilicen en las especificaciones de salida:

- El espaciado que pasa la línea de desbordamiento activa el indicador de desbordamiento.
- El salto que pasa la línea de desbordamiento a cualquier línea de la misma página activa el indicador de desbordamiento.
- El salto, pasada la línea de desbordamiento, a cualquier línea de una nueva página no activa el indicador de desbordamiento a menos que se especifique un salto a un punto posterior a la línea de desbordamiento especificada.
- Un salto a una página nueva especificada en una línea no condicionada por un indicador de desbordamiento desactiva el indicador de desbordamiento después de que el formulario pase a la nueva página.
- Si especifica un salto a una nueva línea y la impresora se encuentra actualmente en esta línea, no se produce un salto. El indicador de desbordamiento se desactiva, a menos que la línea sobrepase la línea de desbordamiento.
- Cuando se especifica una línea OR para un registro de impresión de salida, se utilizan las entradas de espaciado y salto de la línea anterior. Si éstas difieren de la línea anterior, incorpore entradas de espaciado y salto en la línea OR.
- Los indicadores de nivel de control pueden utilizarse con un indicador de desbordamiento para que cada página contenga información sobre un grupo de control solamente. Consulte la Figura 175 en la página 360.
- Para condicionar una línea de desbordamiento, un indicador de desbordamiento puede aparecer en una relación AND o OR. Para una relación AND, el indicador de desbordamiento debe aparecer en la línea de especificación principal para que esa línea se considere una línea de desbordamiento. Para una relación OR, el indicador de desbordamiento puede especificarse en la línea principal de especificación o en la línea OR. Sólo un indicador de desbordamiento puede asociarse con un grupo de indicadores de salida. Para una relación OR sólo se utilizan para el condicionamiento de la línea de desbordamiento los indicadores de condicionamiento de la línea de especificación donde se ha especificado un indicador de desbordamiento.
- Si se utiliza un indicador de desbordamiento en una línea AND, la línea *no* es una línea de desbordamiento. En este caso, el indicador de desbordamiento se trata como cualquier otro indicador de salida.
- Cuando se utiliza el indicador de desbordamiento en una relación AND con un indicador de identificación de registro, suelen obtenerse resultados no habituales puesto que puede que el tipo de registro no se haya leído al producirse el desbordamiento. Por lo tanto, no se activa el indicador de identificación de registro y no se imprime ninguna de las líneas condicionadas por el indicador de desbordamiento y el de identificación de registro.

Acceso a dispositivos de impresora

- Un indicador de desbordamiento condiciona una línea de excepción (E en la posición 17) y los campos dentro del registro de excepción.

Tabla 24. Resultado de la presencia o ausencia de un indicador de desbordamiento

Posiciones 44-80 de las especificaciones de descripción de archivos	Posiciones 21-29 de las especificaciones de salida	Acción
Sin entrada	Sin entrada	Se utiliza el primer indicador de desbordamiento no utilizado para condicionar el salto a la página siguiente al producirse el desbordamiento.
Sin entrada	Entrada	Error en tiempo de compilación; se elimina el indicador de desbordamiento de las especificaciones de salida. Se utiliza el primer indicador de desbordamiento no utilizado para condicionar el salto a la página siguiente al producirse el desbordamiento.
OFLIND (indicador)	Sin entrada	Impresión continua; no se detecta el desbordamiento
OFLIND (indicador)	Entrada	Procesa el desbordamiento normal.

Ejemplo de impresión de cabeceras en cada página

La Figura 174 muestra un ejemplo de la codificación necesaria para imprimir cabeceras en cada página: primera página, cada página de desbordamiento, cada nueva página que se ha de iniciar debido a una modificación en los campos de control (se activa L2). La primera línea permite que se impriman las cabeceras al principio de una nueva página (salta a 06) sólo cuando se produce un desbordamiento (se activa OA y no se activa L2).

La segunda línea permite la impresión de cabeceras en la nueva página sólo al principio de un nuevo grupo de control (se activa L2). De esta forma, no se producen las cabeceras duplicadas generadas tanto por L2 como por OA que estén activadas al mismo tiempo. La segunda línea permite que se impriman las cabeceras en la primera página después de que se lea el primer registro, ya que el primer registro siempre provoca una interrupción del control (se activa L2) si en el registro se han especificado campos de control.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
ONombarch++DF..N01N02N03Nomexc+++B++A++Sb+Sa+.....
OPRINT      H      OANL2                      3 6
0.....N01N02N03Campo++++++YB.Fin++PConstant/palabraedic/DTforma
0              OR      L2
0
0                      8 'FECHA'
0                      18 'CUENTA'
0                      28 'NOMBRE'
0                      46 'SALDO'
0*
```

Figura 174. Impresión de una cabecera en cada página

Ejemplo de impresión de un campo en cada página

La Figura 175 en la página 360 muestra la codificación necesaria para la impresión de determinados campos en cada página; el salto a 06 tiene lugar bien en una

Acceso a dispositivos de impresora

condición de desbordamiento o bien en un cambio de nivel de control (L2). El indicador NL2 evita que la línea se imprima y salte dos veces en el mismo ciclo.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *  
ONombarch++DF..N01N02N03Nomexc++++B++A++Sb+Sa+.....  
OPRINT      D      OANL2                      3 6  
0           OR      L2  
0.....N01N02N03Campo++++++YB.Fin++PConstant/palabraedic/DTforma  
0                      ACCT                      8  
0*
```

Figura 175. Impresión de un campo en cada página

Utilización de la rutina de búsqueda de desbordamiento en archivos descritos por programa

Cuando no hay suficiente espacio en una página para imprimir el resto de las líneas de detalle, totales, excepción y cabecera condicionadas por el indicador de desbordamiento, puede llamarse a la rutina de búsqueda de desbordamiento. Esta rutina provoca un desbordamiento. Para determinar cuándo invocar la rutina de desbordamiento, calcule todas las situaciones de desbordamiento posibles. Al contar las líneas y los espacios puede calcular lo que sucede si se produce un desbordamiento en cada línea de detalle, totales y excepción.

La rutina de búsqueda de desbordamiento le permite alterar la lógica básica de desbordamiento de ILE RPG para evitar que se imprima sobre la perforación y le permite utilizar todo el espacio de la página que sea posible. Durante el ciclo regular del programa, el compilador sólo comprueba una vez, inmediatamente después de la salida de totales, si está activado el indicador de desbordamiento. Cuando se especifica la función de búsqueda de desbordamiento, el compilador comprueba en cada línea en la que se haya especificado la búsqueda de desbordamiento si existe desbordamiento.

La Figura 176 en la página 361 muestra el proceso normal de impresión de desbordamiento cuando se activa y desactiva la búsqueda de desbordamiento.

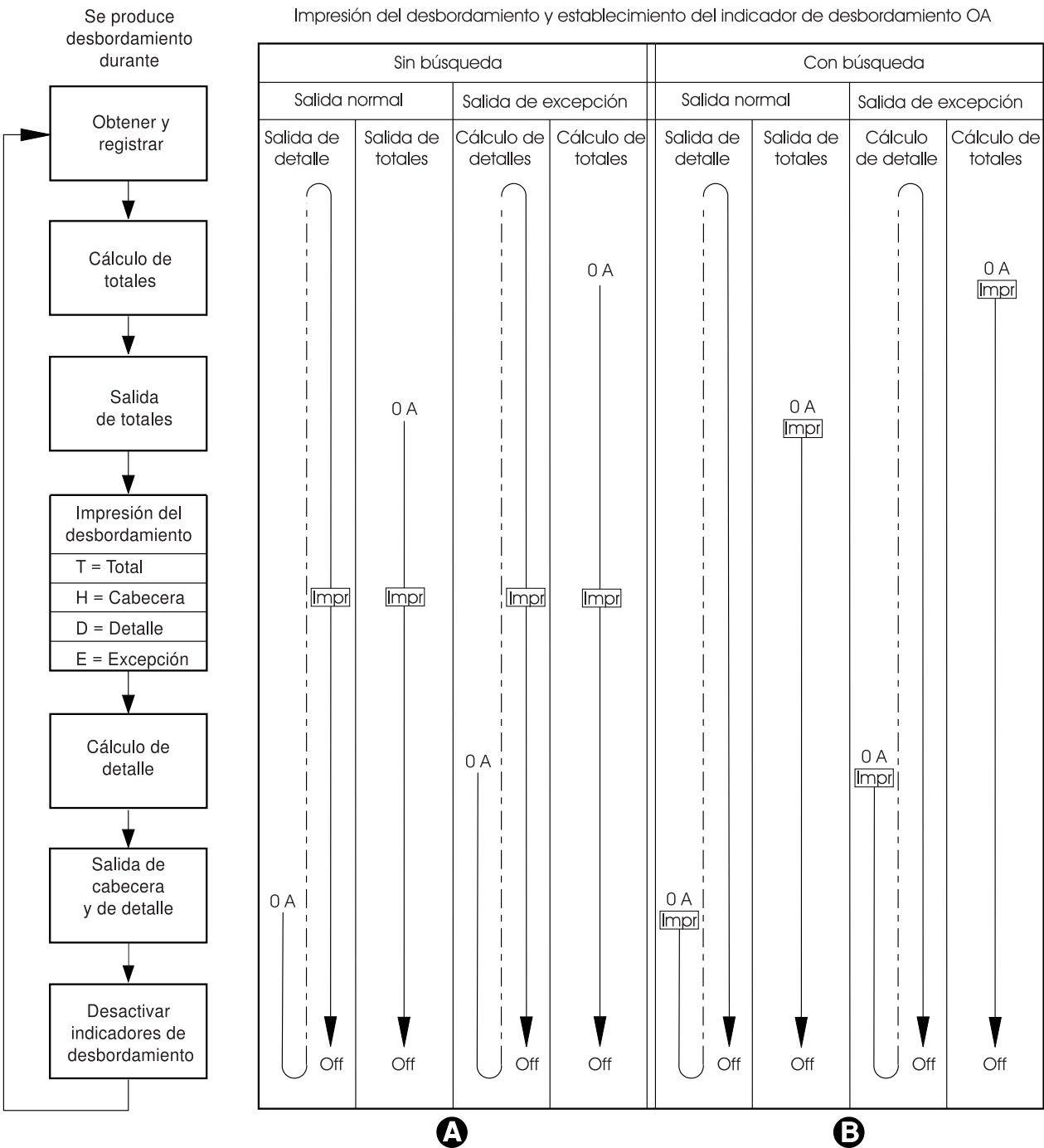


Figura 176. Impresión de desbordamiento: Valor del indicador de desbordamiento

- A** Cuando no se especifica la búsqueda de desbordamiento, las líneas de desbordamiento se imprimen tras la salida de totales. No importa cuando se produzca el desbordamiento (se activa OA), el indicador OA sigue activo a lo largo de todo el tiempo de salida de desbordamiento y se desactiva tras el tiempo de salida detallada y de cabecera.
- B** Cuando se especifica la búsqueda de desbordamiento, las líneas de desbordamiento se escriben antes de la línea de salida para la que se especificó la búsqueda de desbordamiento, siempre que el indicador de desbordamiento OA esté activo. Si se ha activado OA, sigue activo hasta

Acceso a dispositivos de impresora

el transcurso del tiempo de salida detallada y de cabecera. Las líneas de desbordamiento no vuelven a escribirse por segunda vez en el tiempo de salida de desbordamiento a menos que se detecte el desbordamiento de nuevo desde la última vez que se escribieron las líneas de desbordamiento.

Especificación de la búsqueda de desbordamiento

La búsqueda de desbordamiento se especifica mediante una F en la posición 18 de las especificaciones de salida de cualquier línea de excepción, totales o detalles para un archivo PRINTER. La rutina de búsqueda de desbordamiento no provoca que el formulario pase automáticamente a la página siguiente.

Durante la salida, se comprueban los indicadores de condicionamiento de una línea de salida para averiguar si debe escribirse la línea. Si la línea debe escribirse y se ha especificado una F en la posición 18, el compilador efectúa una prueba para determinar si se ha activado el indicador de desbordamiento. Si el indicador de desbordamiento está activo, se busca la rutina de desbordamiento y tienen lugar las siguientes acciones:

1. Sólo se realiza la comprobación de salida de las líneas de desbordamiento para el archivo en el que se ha especificado la búsqueda.
2. Se escriben todas las líneas de totales condicionadas por el indicador de desbordamiento.
3. El formulario pasa a una página nueva cuando en una línea condicionada por un indicador de desbordamiento se especifica un salto a un número de línea inferior al número de línea en el que actualmente se encuentra la impresora.
4. Se escriben las líneas de excepción, detalles y cabecera condicionadas por el indicador de desbordamiento.
5. Se escribe la línea que obtuvo la rutina de desbordamiento.
6. Se escriben las líneas de totales y de detalles que quedan por escribir para dicho ciclo de programa.

La posición 18 de cada línea OR debe incluir una F si debe utilizarse la rutina de desbordamiento para cada registro en la relación OR. La búsqueda de desbordamiento no puede utilizarse si se ha especificado un indicador de desbordamiento en las posiciones 21 a 29 de la misma línea de especificaciones. En ese caso, no se busca la rutina de desbordamiento.

Ejemplo de especificación de la búsqueda de desbordamiento

La Figura 177 en la página 363 muestra el uso de la búsqueda de desbordamiento.

*..	1	...	2	...	3	...	4	...	5	...	6	...	7	...	*
ONombarch++DF..	N01N02N03	Nomexc++++	B++A++Sb+Sa+											
OPRINTER	H	OA						3	05						
0.....	N01N02N03	Campo+++++	YB.	Fin++P	Constant/palabraedic/DTforma										
0					15	'TOTAL EMPLEADOS'									
0		TF	L1				1								
0				TOTEMPL			25								
0		T	L1				1								
0				TOTEMPL			35								
0		T	L1				1								
0				TOTEMPL			45								
0		TF	L1				1								
0				TOTEMPL			55								
0		T	L1				1								
0				TOTEMPL			65								
0		T	L1				1								
0				TOTEMPL			75								
0		T	L1				1								
0*															

Figura 177. Utilización de la búsqueda de desbordamiento

Las líneas de totales con una F codificada en la posición 18 pueden buscar la rutina de desbordamiento. Sólo lo hace si se detecta una rutina de desbordamiento con anterioridad a la impresión de una de estas líneas. Antes de procesarse la búsqueda de desbordamiento, se efectúa una comprobación para determinar si está activado el indicador de desbordamiento. Si está activado, se busca la rutina de desbordamiento, se imprime la línea de cabecera condicionada por el indicador de desbordamiento y se procesan las operaciones de totales.

Modificación de la información de control de formularios en un archivo descrito por programa

La palabra clave PRTCTL (control de impresora) le permite modificar la información sobre el control de formularios y acceder al valor de la línea actual en el programa para un archivo PRINTER descrito por programa. especifique la palabra clave PRTCTL(*nombre de estructura de datos*) en las especificaciones de descripción de archivos para el archivo PRINTER.

Puede especificar dos tipos de estructura de datos PRTCTL en el archivo fuente: una estructura de datos definida por OPM o una estructura de datos ILE. El valor por omisión es utilizar el diseño de estructura de datos ILE que se muestra en la Tabla 25. Para utilizar el diseño de estructura de datos definida por OPM, especifique PRTCTL(*nombre de estructura de datos*:*COMPAT). El diseño de estructura de datos PRTCTL de OPM se muestra en la Tabla 26 en la página 364.

La estructura de datos PRTCTL ILE debe estar definida en las especificaciones de definición. Requiere un mínimo de 15 bytes y debe contener por lo menos los cinco subcampos siguientes especificados en este orden:

Tabla 25. Diseño de una estructura de datos PRTCTL de ILE

Posiciones	Contenido de los subcampos
1-3	Un campo de caracteres de tres posiciones que contiene el valor de espaciar antes (valores válidos: blanco o 0-255)
4-6	Un campo de caracteres de tres posiciones que contiene el valor de espaciar después (valores válidos: blanco o 0-255)
7-9	Un campo de caracteres de tres posiciones que contiene el valor de saltar antes (valores válidos: blanco o 0-255)

Acceso a dispositivos de impresora

Tabla 25. Diseño de una estructura de datos PRTCTL de ILE (continuación)

Posiciones	Contenido de los subcampos
10-12	Un campo de caracteres de tres posiciones que contiene el valor de saltar después (valores válidos: blanco o 0-255)
13-15	Un campo numérico de tres dígitos sin posiciones decimales que contiene el valor del recuento de la línea actual.

La estructura de datos PRTCTL de OPM debe estar definida en las especificaciones de definición y debe contener por lo menos los cinco subcampos siguientes especificados en este orden:

Tabla 26. Diseño de una estructura de datos PRTCTL de OPM

Posiciones	Contenido de los subcampos
1	Un campo de caracteres de una posición que contiene el valor de espaciar antes (valores válidos: blanco o 0-3)
2	Un campo de caracteres de una posición que contiene el valor de espaciar después (valores válidos: blanco o 0-3)
3-4	Un campo de caracteres de dos posiciones que contiene el valor de saltar antes (valores válidos: blanco, 1-99, A0-A9 para 100-109, B0-B2 para 110-112)
5-6	Un campo de caracteres de dos posiciones que contiene el valor de saltar después (valores válidos: blanco, 1-99, A0-A9 para 100-109, B0-B2 para 110-112)
7-9	Un campo numérico de dos posiciones sin posiciones decimales que contiene el valor de recuento de línea actual.

Los valores contenidos en los cuatro primeros subcampos de la estructura de datos PRTCTL de ILE son los mismos que los permitidos en las posiciones 40 a 51 (entradas de espaciado y salto) de las especificaciones de salida. Si las entradas de espaciado/salto (posiciones 40 a 51) de las especificaciones de salida están en blanco, y los subcampos 1 a 4 también están en blanco, el valor por omisión es espaciar 1 después. Si se especifica la palabra clave PRTCTL, se utiliza sólo para los registros de salida que tienen blancos en las posiciones 40 a 51. Se puede controlar el valor de espaciado y de salto (subcampos 1 a 4) para el archivo PRINTER cambiando los valores en estos subcampos de la estructura de datos PRTCTL mientras el programa está ejecutándose.

El subcampo 5 contiene el valor de la cuenta de líneas actual. El compilador no inicializa el subcampo 5 hasta después de que se imprima la primera línea de salida. El compilador cambia entonces el subcampo 5 después de cada operación de salida al archivo.

Ejemplo de modificación de la información de control de formularios

La Figura 178 en la página 365 muestra un ejemplo de la codificación necesaria para modificar la información de control de formularios utilizando la palabra clave PRTCTL.

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++
FPRINT      0      F 132      PRINTER PRTCTL(LINE)

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DNombre+++++ETDsDesde+++A/L+++IDc.Palabrasclave+++++
DLINE      DS
D Espantes      1      3
D Espdesp      4      6
D Espantes      7      9
D Espdesp     10     12
D Linactu     13     15 0

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+MameIg....
C      EXCEPT
C 01Linactu      COMP      10      49
C 01
CAN 49      MOVE      '3'      Espdesp

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
ONombarch++DF..N01N02N03Nomexc+++B++A++Sb+Sa+.....
OPRINT      E      01
O.....N01N02N03Campo+++++YB.Fin++PConstant/palabraedic/DTforma
O      DATA      25

```

Figura 178. Ejemplo de la opción PRTCTL

En las especificaciones de descripción de archivos se especifica la palabra clave PRTCTL para el archivo PRINT. El nombre de la estructura de datos asociada es LINE.

La estructura de datos LINE se define en las especificaciones de entrada con sólo aquellos subcampos predefinidos para la estructura de datos PRTCTL. Los cuatro primeros subcampos, posiciones 1 a 12, se utilizan para facilitar la información de espaciado y salto que normalmente se especifica en las columnas 40 a 51 de las especificaciones de salida. La palabra clave PRTCTL le permite modificar estas especificaciones dentro del programa.

En este ejemplo el valor del subcampo Espdesp se cambia a 3 cuando el valor del subcampo Linactu (valor de recuento de líneas actual) es igual a 10. (Suponga que el indicador 01 se ha activado como un indicador de identificación de registros.)

Acceso a dispositivos de cinta

Utilice las especificaciones de dispositivo SEQ cuando grabe en un archivo de cinta. Para grabar registros de longitud variable en un archivo de cinta, utilice el parámetro RCDBLKFMT del mandato CL CRTTAPF o OVRTAPF. Cuando utilice el parámetro RCDBLKFMT, la longitud de cada registro que se ha de grabar a la cinta está determinada por:

- la posición final más alta especificada en las especificaciones de salida para el registro o,
- si no especifica una posición final, el compilador calcula la longitud de registro a partir de la longitud de los campos.

Lea los registros de cinta de longitud variable como si fueran registros de cualquier archivo organizado secuencialmente. Asegúrese de que la longitud de registro especificada en la especificación de descripción de archivos se ajusta al registro más largo del archivo.

Acceso a dispositivos de pantalla

Utilice archivos de pantalla para intercambiar información entre el programa y un dispositivo de pantalla como una estación de trabajo. Los archivos de pantalla se utilizan para definir el formato de la información que va a visualizarse en un pantalla, y para definir cómo el sistema va a procesar la información cuando ésta se envía a y desde la pantalla.

Consulte el “Capítulo 19. Utilización de archivos WORKSTN” en la página 371 para más información sobre cómo utilizar los archivos WORKSTN.

Utilización de archivos secuenciales

Los archivos secuenciales de un programa en ILE RPG se asocian con cualquier archivo organizado secuencialmente en el sistema AS/400, como:

- Archivo de base de datos
- Archivos de disquete
- Archivo de impresora
- Archivos de cinta

El nombre de archivo del archivo SEQ en las especificaciones de descripción de archivos hace referencia a un archivo de AS/400. La descripción de archivo del archivo de AS/400 especifica el dispositivo de E/S real; por ejemplo, cinta, impresora o disquete.

También puede utilizar los mandatos CL de alteración temporal, por ejemplo OVRDBF, OVRDKTF y OVRTAPF para especificar el dispositivo de E/S real cuando se ejecuta el programa.

Especificación de un archivo secuencial

Una especificación de dispositivo secuencial (SEQ), en las posiciones 36 a 42 de la especificación de descripción de archivos, indica que la entrada o salida está asociada a un archivo organizado secuencialmente. Consulte la Figura 179 en la página 367. El dispositivo real que ha de asociarse con el archivo mientras se ejecuta el programa puede especificarse mediante un mandato de alteración temporal del OS/400 o mediante la descripción de archivos a la que hace referencia el nombre del archivo. Si se especifica SEQ en un programa, no puede especificarse ninguna función dependiente de dispositivo como espaciar/saltar, ni tampoco CHAIN.

La figura siguiente muestra los códigos de operación permitidos para un archivo SEQ.

Tabla 27. Códigos de operación de archivo válidos para un archivo secuencial

Posiciones de especificaciones de descripción de archivo		Posiciones de especificaciones de cálculo
17	18	26-35
I	P/S	CLOSE, FEOD
I	F	READ, OPEN, CLOSE, FEOD
O		WRITE, OPEN, CLOSE, FEOD

Nota: Para un archivo secuencial no están permitidas las especificaciones de control de impresión.

Ejemplo de especificación de un archivo secuencial

La Figura 179 muestra un ejemplo de cómo especificar un archivo SEQ en un miembro fuente de ILE RPG.

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++++++
FTARITPO  IP  E          DISK
FHORAEXTR 0   F  132     SEQ
*
```

Figura 179. Dispositivo SEQ

Se ha especificado un dispositivo SEQ para el archivo HORAEXTR. Cuando se ejecuta el programa, puede utilizar un mandato de alteración temporal de OS/400 para especificar el dispositivo real (como por ejemplo impresora, cinta o disquete) que se ha de asociar al archivo mientras se ejecuta el programa. Por ejemplo, se puede especificar disquete para algunas ejecuciones del programa e impresora para otras. La descripción de archivo, a la que señala el nombre del archivo, puede especificar el dispositivo actual, en cuyo caso no es necesario utilizar un mandato de alteración temporal.

Utilización de archivos SPECIAL

El nombre de dispositivo RPG SPECIAL (posiciones 36 a 42 de las especificaciones de descripción de archivos) le permite especificar un dispositivo de entrada y/o salida al que las operaciones del ILE RPG no den soporte directamente. Las operaciones de entrada y salida para el archivo están controladas mediante una rutina escrita por el usuario. El nombre de la rutina escrita por el usuario debe identificarse en las especificaciones de descripción de archivos utilizando la palabra clave PGMNAME('nombre de programa').

ILE RPG llama a esta rutina escrita por el usuario para abrir el archivo, leer y grabar registros y cerrar el archivo. ILE RPG también crea una lista de parámetros que la rutina escrita por el usuario va a utilizar. La lista de parámetros contiene:

- parámetro de código de opción (opción)
- parámetro de estado de retorno (estado)
- parámetro de error encontrado (error)
- parámetro de área de registro (área)

El compilador del ILE RPG y la rutina escrita por el usuario acceden a esta lista de parámetros; el programa que contiene el archivo SPECIAL no puede acceder a esta lista de parámetros.

A continuación se describen los parámetros incluidos en esta lista de parámetros creada por el RPG:

Opción

El parámetro de opción es un campo de un carácter que indica que la rutina escrita por el usuario va a procesarse. Según la operación que está procesándose en el archivo SPECIAL (OPEN, CLOSE, FEOD, READ, WRITE, DELETE, UPDATE), del ILE RPG se pasa uno de los siguientes valores a la rutina escrita por el usuario:

Utilización de archivos SPECIAL

Valor que se pasa

Descripción

- | | |
|----------|---|
| O | Abrir el archivo. |
| C | Cerrar el archivo. |
| F | Forzar el final del archivo. |
| R | Leer un registro y situarlo en el área definida por el parámetro de área. |
| W | El programa del ILE RPG ha colocado un registro en el área definida por el parámetro de área en el registro; el registro debe grabarse. |
| D | Suprimir el registro. |
| U | El registro es una actualización del último registro leído. |

Estado

El parámetro de estado es un campo de caracteres de una posición que indica el estado de la rutina escrita por el usuario cuando se devuelve el control al programa ILE RPG. El estado debe incluir uno de los siguientes valores de retorno cuando la rutina escrita por el usuario devuelve el control al programa en ILE RPG:

Valor de retorno

Descripción

- | | |
|----------|---|
| 0 | Retorno normal. Se ha procesado la acción solicitada. |
| 1 | El archivo de entrada está al final del archivo y no se ha devuelto ningún registro. Si el archivo es de salida, este valor de retorno es un error. |
| 2 | No se ha procesado la acción solicitada; existe una condición de error. |

Error El parámetro de error es un campo numérico con zona de cinco dígitos con cero posiciones decimales. Si la rutina escrita por el usuario detecta un error, el parámetro de error contiene una indicación o valor que representa el tipo de error. El valor se sitúa en las primeras cinco posiciones de la posición *RECORD en la INFDS cuando el parámetro de estado contiene 2.

Área El parámetro de área es un campo de caracteres cuya longitud es igual a la longitud de registro asociada con el archivo SPECIAL. Este campo se utiliza para pasar el registro a o recibirlo desde el programa en ILE RPG

Puede añadir parámetros adicionales a la lista de parámetros creada por RPG. Especifique la palabra clave PLIST(*nombre de lista de parámetros*) en las especificaciones de descripción de archivos para el archivo SPECIAL. Consulte la Figura 180 en la página 369. Después utilice la operación PLIST en las especificaciones de cálculo para definir los parámetros adicionales.

La rutina escrita por el usuario, especificada mediante la palabra PGMNAME de las especificaciones de descripción de archivos para el archivo SPECIAL, debe contener una lista de parámetros de entrada que incluya tanto los parámetros creados por RPG, como los parámetros especificados por el usuario.

Si se especifica el archivo SPECIAL como un archivo primario, los parámetros especificados por el usuario deben inicializarse antes de leer el primer archivo

Utilización de archivos SPECIAL

primario. Puede inicializar estos parámetros con una entrada de factor 2 en las sentencias PARM, o especificando una matriz en tiempo de compilación o un elemento de matriz como un parámetro.

La Tabla 28 muestra los códigos de operación de archivo que son válidos para un archivo SPECIAL.

Tabla 28. Operaciones de archivo válidas para un archivo SPECIAL

Posiciones de especificaciones de descripción de archivo		Posiciones de especificaciones de cálculo
17	18	26-35
I	P/S	CLOSE, FEOD
C	P/S	WRITE, CLOSE, FEOD
U	P/S	UPDATE, DELETE, CLOSE, FEOD
O		WRITE, OPEN, CLOSE, FEOD
I	F	READ, OPEN, CLOSE, FEOD
C	F	READ, WRITE, OPEN, CLOSE, FEOD
U	F	READ, UPDATE, DELETE, OPEN, CLOSE, FEOD

Ejemplo de utilización de un archivo SPECIAL

La Figura 180 muestra cómo utilizar el nombre de dispositivo RPG SPECIAL en un programa. En este ejemplo, se ha asociado una descripción de archivo del archivo EXCPTN con el dispositivo SPECIAL.

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *				
FNombarch++IPEASFLreg+LonLK+AIDispos+.Palabrasclave+++++				
FEXCPTN 0 F 20 SPECIAL PGMNAME('ENSAUS')				
F PLIST(SPCL)				
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *				
DNombre+++++ETDsDesde+++A/L+++IDc.Funciones+++++				
D OUTBUF DS				
D FLD 1 20				
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *				
CL0N01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+NaMeIg...				
C	SPCL	PLIST		
C		PARM	FLD1	
C		MOVEL	'HELLO'	FLD
C		MOVE	'1'	FLD1 1
C		WRITE	EXCPTN	OUTBUF
C		MOVE	'2'	FLD1 1
C		WRITE	EXCPTN	OUTBUF
C		SETON		LR

Figura 180. Dispositivo SPECIAL

La Figura 181 en la página 370 muestra el programa ENSAUS escrito por el usuario.

Utilización de archivos SPECIAL

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
DNombre+++++ETDsDesde+++A/L+++IDc.Funciones+++++
D ERROR          S          5S 0
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... *
CL0N01Factor1+++++Operac(E)+Factor2+++++Resultado+++++Lon++D+NaMeIg...

*-----*
* Los 4 primeros parámetros son una lista de parámetros creada por ILE RPG.
* El resto se definen mediante el PLIST definido por el programador.
*-----*
C      *ENTRY      PLIST
C          PARM          OPTION          1
C          PARM          STATUS          1
C          PARM          ERROR          5 0
C          PARM          AREA          20
C          PARM          FLD1          1

*-----*
* El programa escrito por el usuario efectuará la E/S de archivo *
* de acuerdo con las opciones pasadas. *
*-----*
C          SELECT
C          WHEN      OPTION = 'O'
C* efectúa la operación OPEN
C          WHEN      OPTION = 'W'
C* efectúa la operación WRITE
C          WHEN      OPTION = 'C'
C* efectúa la operación CLOSE
C          ENDSL
C          RETURN

```

Figura 181. Programa ENSAUS escrito por el usuario

Las operaciones de E/S para el dispositivo SPECIAL las controla el programa ENSAUS escrito por el usuario. Los parámetros especificados para PLIST(SPCL), definidos por el usuario, se añaden al final de la lista de parámetros creada por el RPG para el dispositivo SPECIAL. Se puede acceder a los parámetros especificados por el programador mediante el programa ILE RPG del usuario y la rutina ENSAUS escrita por el usuario; sin embargo, sólo se puede acceder a la lista de parámetros creada por el RPG mediante la lógica interna de ILE RPG y la rutina escrita por el usuario.

Capítulo 19. Utilización de archivos WORKSTN

Las aplicaciones interactivas en el servidor iSeries normalmente están relacionadas con la comunicación con:

- Los usuarios de una o varias estaciones de trabajo a través de archivos de pantalla
- Uno o varios programas de un sistema remoto a través de archivos ICF
- Uno o varios dispositivos en un sistema remoto a través de archivos ICF.

Los **archivos de pantalla** son objetos del tipo *FILE con atributos DSPF del sistema iSeries. Utilice los archivos de pantalla para comunicarse de forma interactiva con usuarios en terminales de pantalla. Al igual que los archivos de base de datos, los archivos de pantalla puede estar descritos externamente o por programa.

Los **archivos ICF** son objetos del tipo *FILE con atributos de ICFF del sistema iSeries. Utilice archivos ICF para comunicarse (enviar y recibir datos) con otros programas de aplicación en sistemas remotos (iSeries o no iSeries). Un archivo ICF contiene los formatos de comunicación necesarios para enviar y recibir datos entre sistemas. Puede grabar programas que utilicen archivos ICFy que le permitan comunicarse (enviar y recibir datos) con otros programas de aplicación en sistemas remotos.

Cuando un archivo de un programa RPG está identificado mediante el nombre de dispositivo WORKSTN, ese programa puede comunicarse interactivamente con un usuario de estación de trabajo o utilizar la función función de comunicaciones intersistemas (ICF) para comunicarse con otros programas. En este capítulo se describe cómo utilizar:

- función de comunicaciones intersistemas (ICF)
- Archivos WORKSTN descritos externamente
- Archivos WORKSTN descritos por programa
- Archivos de dispositivos múltiples

función de comunicaciones intersistemas

Para utilizar la función ICF, defina un archivo WORKSTN en su programa que haga referencia a un archivo de dispositivo ICF. Utilice el archivo suministrado por el sistema QICDMF o un archivo que se haya creado mediante el mandatos OS/400 CRTICFF.

Codifique para ICF utilizando la ICF como un archivo del programa. La ICF es similar a un archivo de pantalla y contiene los formatos de comunicaciones requeridos para enviar o recibir datos entre sistemas.

Para más información sobre la ICF, consulte el manual *ICF Programming*.

Utilización de archivos WORKSTN descritos externamente

Un archivo WORKSTN de RPG puede utilizar un archivo de dispositivo de pantalla descrito externamente o un archivo de dispositivo ICF, el cual contiene información sobre el archivo y una descripción de los campos en los registros que han de grabarse. El archivo WORKSTN descrito externamente que se utiliza con

Utilización de archivos WORKSTN descritos externamente

mayor frecuencia es un archivo de pantalla. (Para obtener más información sobre la descripción y creación de archivos de pantalla, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseriess/infocenter>.)

Además de las descripciones de campo (como por ejemplo nombres de campo y atributos), las DDS de un archivo de dispositivo de pantalla se emplean para:

- Dar forma a la colocación del registro en la pantalla especificando las entradas de número de línea y de número de posición para cada campo y constante.
- Especificar funciones de atención tales como subrayado y campos con alta intensidad, imagen con contraste invertido o cursor parpadeante.
- Especificar comprobación de validez para los datos introducidos en la estación de trabajo de pantalla. Las funciones de comprobación de validez incluyen la detección de campos donde son necesarios datos, la detección de campos que se tienen que rellenar obligatoriamente, la detección de tipos incorrectos de datos, la detección de datos dentro de un rango específico, la comprobación de datos para una entrada válida y el proceso de la verificación de dígitos de comprobación de los módulos 10 u 11.
- Controlar funciones de gestión de pantallas, como por ejemplo si los campos han de borrarse, recubrirse o retenerse cuando se visualizan datos nuevos.
- Asociar los indicadores 01 a 99 con las teclas de atención de mandatos o con las teclas de función de mandatos. Si una tecla de función se describe como una tecla de función de mandato (CF), se devuelven al programa tanto el registro de datos (con todas las modificaciones entradas en la pantalla) como el indicador de respuesta. Si una tecla de función se describe como una tecla de atención de mandato (CA), se devuelve al programa un indicador de respuesta, pero el registro de datos permanece inalterado. De ese modo, los campos de tipo carácter, únicamente de entrada, están en blanco y los campos numéricos, únicamente de entrada, se rellenan con ceros a no ser que esos campos se hayan inicializado de un modo distinto.
- Asignar un código de edición (EDTCDE) o una palabra de edición (EDTWRD) a un campo para especificar cómo han de visualizarse los valores de los campos.
- Especificar subarchivos.

Un formato de registro de dispositivo de pantalla contiene tres tipos de campos:

- *Campos de entrada.* Los campos de entrada se transfieren desde el dispositivo al programa cuando el programa lee un registro. Los campos de entrada pueden inicializarse con un valor por omisión. Si no se ha modificado el valor por omisión, dicho valor se transfiere al programa. Los campos de entrada que no están inicializados se visualizan como blancos en los que el usuario de la estación de trabajo puede entrar datos.
- *Campos de salida.* Los campos de salida se transfieren desde el programa al dispositivo cuando el programa graba un registro en una pantalla. Los campos de salida puede suministrarlos el programa o el formato de registro del archivo de dispositivo.
- *Campos de entrada/salida (ambos).* Un campo de entrada/salida es un campo de salida que puede modificarse. Se convierte en un campo de entrada si se modifica. Los campos de entrada/salida se transfieren desde el programa cuando éste escribe un registro a una pantalla y se transfieren al programa cuando éste lee un registro desde la pantalla. Los campos de entrada/salida se emplean cuando el usuario ha de cambiar o actualizar los datos que el programa ha escrito en la pantalla.

Utilización de archivos WORKSTN descritos externamente

Si especifica la palabra clave INDARA en las DDS para un archivo WORKSTN, el programa en RPG transfiere indicadores al archivo WORKSTN en un área de indicador distinta y no en el almacenamiento intermedio de entrada/salida.

Para obtener una descripción detallada de un archivo de dispositivo pantalla descrito externamente y una lista de palabras clave de DDS válidas, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:
<http://www.ibm.com/eserver/iseriess/infocenter>.

La Figura 182 muestra un ejemplo de las DDS para un archivo de dispositivo de pantalla.

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...
AAN01N02N03T.Nombre++++RLon++TDpBLínPosFunciones+++++*****
A** CONSULTA DEL MAESTRO DE ARTICULOS
A                                     REF(REFDIS) 1
A                                     TEXT('Formato Solicitud Artículo')
A 73N61                               OVERLAY 2
A                                     CA03(98 'Fin del programa') 3
A                                     1 2'Consulta de Artículo'
A                                     3 2'Número de Artículo'
A                                     ITEM      R      I 3 15PUTRETAIN 4
A 61                                  ERRMSG('Número de Artículo no válido' 61) 5
A                                     R RESPONSE
A                                     TEXT('Formato de Respuesta')
A                                     OVERLAY 2
A                                     LOCK 6
A                                     5 2'Descripción'
A                                     5 15
A                                     5 37'Precio'
A                                     PRICE      R      5 44
A                                     7 2'Ubicación en almacén' 7
A                                     WHSLOC      R      7 22
A                                     9 2'Existencias'
A                                     ONHAND      R      9 10
A                                     9 19'Asignado' 8
A                                     ALLOC      R      9 30
A                                     9 40'Disponible'
A                                     AVAIL      R      9 51
A*

```

Figura 182. Ejemplo de las especificaciones de descripción de datos para un archivo de dispositivo de pantalla

Este archivo de dispositivo de pantalla contiene dos formatos de registro PROMPT y RESPONSE.

- 1 Los atributos de los campos de este archivo están definidos en el archivo de referencias de campos REFDIS.
- 2 Se utiliza la palabra clave OVERLAY de forma que ambos formatos de registro pueden emplearse en la misma pantalla.
- 3 La tecla de función 3 está asociada con el indicador 98, utilizado por el programador para finalizar el programa.
- 4 La palabra clave PUTRETAIN permite mantener en la pantalla el valor que se introduce en el campo ITEM. Además, se define el campo ITEM como un campo de entrada mediante la I en la posición 38. ITEM es el único campo de entrada en estos formatos de registro. El resto de campos de registro son campos de salida ya que la posición 38 para cada uno de ellos está en blanco.

Utilización de archivos WORKSTN descritos externamente

- 5** La palabra clave ERRMSG identifica el mensaje de error que se visualiza si se activa el indicador 61 en el programa que utiliza este formato de registro.
- 6** La palabra clave LOCK impide al usuario de la estación de trabajo utilizar el teclado cuando se visualiza inicialmente el formato de registro RESPONSE.
- 7** Las constantes como 'Descripción', 'Precio' y 'Ubicación en almacén' describen los campos que graba el programa.
- 8** Las entradas de línea y posición identifican donde tienen que escribirse en la pantalla los campos o constantes.

Especificación de indicadores de teclas de función en archivos de dispositivo de pantalla

Los indicadores de tecla de función KA a KN y KP a KY son válidos para un programa que contenga un archivo de dispositivo de pantalla WORKSTN si se especifica la tecla de función asociada en las especificaciones de descripción de datos (DDS).

Los indicadores de tecla de función están relacionados con las teclas de función de la siguiente forma: el indicador de tecla de función KA corresponde a la tecla de función 1, KB a la tecla de función 2... KX a la tecla de función 23 y KY a la tecla de función 24.

Las teclas de función se especifican en las DDS con la palabra clave CFxx (función de mandato) o CAxx (atención de mandato). Por ejemplo, la palabra clave CF01 autoriza la utilización de la tecla de función 1. Cuando pulsa la tecla de función 1, se activa el indicador de tecla de función KA en el programa RPG. Si especifica la tecla de función como CF01(99), se activarán en el programa RPG el indicador KA de tecla de función y el indicador 99. Si el usuario de la estación de trabajo pulsa una tecla de función que no se ha especificado en las DDS, el sistema OS/400 informa al usuario que ha pulsado una tecla incorrecta.

Si el usuario de la estación de trabajo pulsa una tecla de función determinada, se activa el indicador de tecla de función asociado en el programa RPG cuando se extraen los campos del registro (lógica de movimiento de campos) y se desactivan todos los otros indicadores de tecla de función. Si no se pulsa ninguna tecla de función, todos los indicadores de tecla de función se desactivan en tiempo de movimiento de campos. Los indicadores de tecla de función se desactivan si el usuario pulsa la tecla Intro.

Especificación de teclas de mandato en archivos de dispositivo de pantalla

Puede especificar las teclas de mandatos Ayuda, Girar hacia arriba, Girar hacia abajo, Imprimir, Borrar e Inicio en las especificaciones de descripción de datos (DDS) para un archivo de dispositivo de pantalla, con las palabras clave HELP, ROLLUP, ROLLDOWN, PRINT, CLEAR y HOME.

Un programa RPG puede procesar las teclas de mandato siempre que el compilador procese una operación READ o una operación EXFMT en un formato de registro para el que se hayan especificado en las DDS las palabras clave apropiadas. Cuando las teclas de mandato están en vigor y se pulsa una tecla de función, el sistema OS/400 devuelve el control al programa RPG. Si se especifica un indicador de respuesta en las DDS para el mandato seleccionado, se activa

Utilización de archivos WORKSTN descritos externamente

dicho indicador y se desactivan el resto de indicadores de respuesta que estén en vigor para el formato de registro y para el archivo.

Si no se especifica un indicador de respuesta en las DDS para una tecla de mandato, sucede lo siguiente

- Para la tecla IMPR sin *PGM especificado, se efectúa la función de impresión.
- Para las teclas GIRAR HACIA ARRIBA y GIRAR HACIA ABAJO utilizadas con subarchivos, el subarchivo visualizado gira hacia arriba o hacia abajo, dentro del subarchivo. Si se intenta girar más allá del inicio o del final de un subarchivo, se produce un error durante la ejecución.
- Para la tecla Impr especificada con las teclas *PGM, Girar hacia arriba y Girar hacia abajo utilizadas sin subarchivos y para las teclas Borrar, Ayuda e Inicio, se establece uno de los valores 1121-1126 de *STATUS, respectivamente, y el proceso continúa.

Proceso de un archivo WORKSTN descrito externamente

Cuando se procesa un archivo WORKSTN descrito externamente, el sistema OS/400 transforma los datos del programa al formato especificado para el archivo y visualiza los datos. Cuando se transfieren los datos al programa, estos se transforman al formato empleado por el programa.

El sistema OS/400 proporciona información para el control del dispositivo para que éste efectúe operaciones de entrada/salida. Cuando se solicita un registro de entrada del dispositivo, el sistema OS/400 emite la solicitud y, a continuación, elimina de los datos la información de control del dispositivo antes de transferir los datos al programa. Además, el sistema OS/400 puede transferir indicadores al programa que reflejan qué campos, si hay alguno, han sido alterados en el registro.

Cuando el programa solicita una operación de salida, transfiere el registro de salida al sistema OS/400. El sistema OS/400 proporciona la información de control del dispositivo necesaria para visualizar el registro. Cuando se visualiza el registro, se añade también cualquier información de constantes especificada para el formato de registro.

Cuando se transfiere un registro a un programa, los campos se disponen en el orden que se haya especificado en las DDS. El orden en el que se visualizan los campos se basa en las posiciones de pantalla (números de línea y posición) asignadas al campo dentro de las DDS. El orden en el que se especifican los campos en las DDS y el orden en el que aparecen en la pantalla no tiene que ser el mismo necesariamente.

Para obtener más información sobre el proceso de archivos WORKSTN, consulte el apartado “Operaciones válidas del archivo WORKSTN” en la página 382.

Utilización de subarchivos

En las especificaciones de descripción de datos (DDS) pueden especificarse subarchivos de un archivo de dispositivo de pantalla para permitir al usuario manejar múltiples registros del mismo tipo en la pantalla. (Consulte la Figura 183 en la página 376.) Un subarchivo es un grupo de registros que se leen o se graban en un archivo de dispositivo de pantalla. Por ejemplo, un programa lee registros de un archivo de base de datos y crea un subarchivo de registros de salida. Cuando se haya grabado el subarchivo en su totalidad, el programa envía el subarchivo completo al dispositivo de pantalla en una operación de grabación. El usuario de la estación de trabajo puede alterar los datos o entrar datos adicionales en el

Utilización de archivos WORKSTN descritos externamente

subarchivo. El programa lee a continuación todo el subarchivo desde el dispositivo de pantalla, lo transfiere al programa y procesa individualmente cada registro del subarchivo.

Los registros que desea incluir en un subarchivo se especifican en las DDS del archivo. El número de registros que se puede incluir en un subarchivo también debe especificarse en las DDS. Un archivo puede contener más de un subarchivo y pueden estar activados al mismo tiempo hasta 12 subarchivos. Dos subarchivos pueden visualizarse a la vez.

Las DDS de un subarchivo están formadas por 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 la información de los campos que se transfiere hacia o desde el archivo de pantalla bajo el control del formato de registro de control del subarchivo. El formato de registro de control del subarchivo hace que tengan lugar las operaciones físicas de lectura, grabación o de control del subarchivo. La Figura 184 en la página 378 muestra un ejemplo de las DDS para un formato de registro de subarchivo y la Figura 185 en la página 379 muestra un ejemplo de las DDS para un registro de control del subarchivo.

Para obtener una descripción del modo de utilizar las palabras clave de subarchivo, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iseries/infocenter..>

[illegible]

Figura 183. Visualización de un subarchivo

Para utilizar un subarchivo de dispositivo de pantalla en un programa en RPG debe especificar la palabra clave SFILE en las especificaciones de descripción de archivos para el archivo WORKSTN. El formato de la palabra clave SFILE es SFILE(*nombre de formato de registro:nombre de campo RECNO*). El archivo WORKSTN debe ser un archivo descrito externamente (E en la posición 22).

Utilización de archivos WORKSTN descritos externamente

Para la palabra clave SFILE debe especificar el nombre del formato de registro de subarchivo (no el formato de registro de control) y el nombre del campo que contiene el número relativo de registro que se ha de utilizar en el proceso del subarchivo.

En un programa en RPG, el proceso por número relativo de registro se define como parte de la definición SFILE. La definición SFILE implica un archivo de actualización controlado en cálculo que tenga ADD para el subarchivo. Por eso, las operaciones de archivo que son válidas para el subarchivo no dependen de la definición del archivo WORKSTN principal. Es decir, el archivo WORKSTN puede definirse como un archivo primario o un archivo controlado en cálculo.

Utilice los códigos de operación CHAIN, READC, UPDAT o WRITE con el formato de registro de subarchivo para transferir datos entre el programa y el subarchivo. Utilice los códigos de operación READ, WRITE o EXFMT con el formato de registro de control de subarchivo para transferir datos entre el programa y el dispositivo de pantalla o para procesar operaciones de control de subarchivo.

El proceso de subarchivos sigue las reglas para el proceso por número relativo de registro. El programa en RPG coloca el número relativo de registro de cualquier registro recuperado mediante una operación READC en el campo citado en la segunda posición de la palabra clave SFILE. Este campo se utiliza también para especificar el número de registro que el programa en RPG utiliza en las operaciones WRITE del subarchivo o en las operaciones de salida que utilizan ADD. El nombre de campo RECNO especificado para la palabra clave SFILE debe definirse como numérico sin ninguna posición decimal. El campo debe tener suficientes posiciones como para contener el número de registro más grande del archivo. Consulte la palabra clave SFLSIZ en el apartado *DB2 Universal Database para AS/400* de la categoría *Base de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web:

<http://www.ibm.com/eserver/iseres/infocenter>.) El código de operación WRITE y la especificación ADD de las especificaciones de salida requieren que se especifique un campo de número relativo de registro en la segunda posición de la palabra clave SFILE en la especificación de descripción de archivos.

Si un archivo WORKSTN tiene asociado un subarchivo, todas las operaciones de entrada implícitas y todas las operaciones de cálculo explícitas que hagan referencia al nombre del archivo tienen lugar para el archivo WORKSTN principal. Las operaciones que especifiquen un nombre de formato de registro que no esté designado como un subarchivo se procesan en el archivo WORKSTN principal.

Si pulsa una tecla de función determinada durante la lectura de un registro que no es de subarchivo, las lecturas posteriores de un registro de subarchivo harán que el indicador de tecla de función correspondiente se vuelva a activar aunque el indicador de tecla de función se haya desactivado entre las lecturas. Esto continuará hasta que se lea un registro que no sea de un subarchivo del archivo WORKSTN.

Utilización de archivos WORKSTN descritos externamente

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
AAN01N02N03T.Nombre++++RLon++TDpBLínPosFunciones+++++*****
A** BUSQUEDA DE NOMBRE DE CLIENTE
A                                     REF(REFDIS) 1
A                                     SFL 2
A                                     TEXT('Registro Subarchivo')
A          CLIE      R          7 3
A          NOMB      R          7 10
A          DIRE      R          7 32 3
A          CDAD      R          7 54
A          PROV      R          7 77
A*
```

Figura 184. Especificaciones de descripción de datos para un formato de registro de subarchivo

Las especificaciones de descripción de datos (DDS) para un formato de registro de subarchivo describen los registros del subarchivo:

- 1** Los atributos de los campos en el formato de registro se incluyen en el archivo de referencias de campos REFDIS tal como se especifica mediante la palabra clave REF.
- 2** La palabra clave SFL identifica el formato de registro como un subarchivo.
- 3** Las entradas de línea y posición definen la situación de los campos en la pantalla.

Utilización de subarchivos

Algunos usos clásicos de subarchivos incluyen:

- Sólo visualización. El usuario de la estación de trabajo revisa la visualización.
- Visualización con selección. El usuario solicita más información sobre uno de los elementos de la pantalla.
- Modificación. El usuario modifica uno o varios 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, pero se comprueban los registros.
- Combinación de tareas. Puede utilizarse un subarchivo como una pantalla con modificación, más la entrada de nuevos registros.

La figura siguiente muestra un ejemplo de especificaciones de descripción de datos para un formato de registro de control de subarchivo. Para ver un ejemplo de la utilización de un subarchivo en un programa en RPG, consulte el apartado “Búsqueda por código postal” en la página 402.

```

*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
AAN01N02N03T.Nombre++++RLon++TDpBLInPosFunciones+++++*****
A          R ARCTRL                      SFLCTL(SUBARCH)
A N70                      SFLCLR
A 70                      SFLDSPCTL
A 71                      SFLDSP
A                      SFLSIZ(15)
A                      SFLPAG(15)
A                      TEXT('Reg. Control Subarchivo')
A                      OVERLAY
A 71                      ROLLUP(97 'Continúa Búsqueda')
A                      CA01(98 'Fin del Programa')
A                      HELP(99 'Tecla AYUDA')
A                      1 2'Búsqueda Nombre Cliente'
A                      3 2'Código Búsqueda'
A          CODBUS    R          I 3 14PUTRETAIN
A                      5 2'Número'
A                      5 10'Nombre'
A                      5 32'Calle'
A                      5 54'Ciudad'
A                      5 76'Provincia'
A*

```

Figura 185. Especificaciones de descripción de datos para un formato de registro de control de subarchivo

El formato de registro de control de subarchivo define los atributos del subarchivo, el campo de entrada para efectuar la búsqueda, las constantes y las teclas de función. Las palabras clave que puede utilizar indican lo siguiente:

- SFLCTL indica el nombre del subarchivo asociado (SUBARCH).
- SFLCLR indica cuándo deberá borrarse el subarchivo (cuando el indicador 70 esté desactivado).
- SFLDSPCTL indica cuando debe visualizarse el registro de control del subarchivo (cuando el indicador 70 esté activado).
- SFLDSP indica cuándo va a visualizarse el subarchivo (cuando está activado el indicador 71).
- SFLSIZ indica el número total de registros a incluir en el subarchivo (15).
- SFLPAG indica el número total de registros por página (15).
- ROLLUP indica que el indicador 97 se activa en el programa cuando el usuario pulsa la tecla de Giro hacia arriba.
- HELP permite al usuario pulsar la tecla AYUDA para que se visualice un mensaje que describe las teclas de función válidas.
- PUTRETAIN permite mantener en la pantalla el valor que se ha entrado en el campo CODBUS.

Además de la información de control, el formato de registro de control del subarchivo define también las constantes a utilizar como cabeceras de columna para el formato de registro del subarchivo.

Utilización de archivos WORKSTN descritos por programa

Un archivo WORKSTN descrito por programa puede usarse con o sin un nombre de formato especificado en las especificaciones de salida. El nombre del formato, si se especifica, hace referencia al nombre de un formato de registro de las especificaciones de descripción de datos. Este formato describe:

Utilización de archivos WORKSTN descritos por programa

- Cómo se le da formato en la pantalla a la corriente de datos enviada desde un programa RPG.
- Qué datos se envían
- Qué funciones ICF se deben realizar.

Si se utiliza un nombre de formato, deben emplearse las especificaciones de entrada y de salida para describir los registros de entrada y salida.

Puede especificar PASS(*NOIND) en una especificación de descripción de archivos para un archivo WORKSTN descrito por programa. La palabra clave PASS(*NOIND) indica que el programa en RPG no transferirá indicadores para la gestión de datos en la salida ni los recibirá en la entrada adicionalmente. Es responsabilidad del usuario la transferencia de los indicadores describiéndolos como campos (con el formato *INxx, *IN o *IN(x)) en el registro de entrada o de salida. Deben especificarse en la secuencia requerida por las especificaciones de descripción de datos (DDS). Puede utilizar el listado de las DDS para determinar dicha secuencia.

Utilización de un archivo WORKSTN descrito por programa con un nombre de formato

Las siguientes especificaciones son aplicables a la utilización de un nombre de formato para un archivo WORKSTN descrito por programa.

Especificaciones de salida

En las especificaciones de salida debe especificar el nombre del archivo WORKSTN en las posiciones 7 a 16. El nombre del formato, que es el nombre del formato de registro en las DDS, se especifica como un literal en las posiciones 53 a 80 en la línea de descripción del campo siguiente. K1 a K10 deben especificarse (ajustados por la derecha) en las posiciones 47 a 51 de la línea que contenga el nombre del formato. La K que identifica la entrada como una longitud, en lugar de una posición final, y el número indica la longitud del nombre del formato. Por ejemplo, si el nombre del formato es AYUCLI, la entrada en las posiciones 47 a 51 será K6. (Se admiten ceros a la izquierda a continuación de la K). No puede condicionarse el nombre del formato (no son válidos los indicadores en las posiciones 21 a 29).

Los campos de salida deben encontrarse en el registro de salida en el mismo orden en que están definidos en las DDS. Sin embargo, no es necesario que los nombres de campo sean los mismos. Las entradas de posición final de los campos hacen referencia a la posición final del registro de salida que se transfiere del programa RPG a la gestión de datos, no a la ubicación de los campos en la pantalla.

Para transferir indicadores de la salida, efectúe una de las siguientes acciones:

- Especifique la palabra clave INDARA en las DDS para el archivo WORKSTN. No utilice la palabra clave PASS(*NOIND) en la especificación de descripción de archivos y no especifique los indicadores en las especificaciones de salida. El programa y el archivo usan un área de indicadores separada para transferir los indicadores.
- Especifique la palabra clave PASS(*NOIND) en la especificación de descripción de archivos. Especifique los indicadores en las especificaciones de salida como campos con el formato *INxx. Los campos de indicador deben preceder a otros campos en el registro de salida y deben aparecer en el orden especificado por las DDS del archivo WORKSTN. Este orden puede determinarse a partir del listado de las DDS.

Especificaciones de entrada

Las especificaciones de entrada describen el registro que el programa en RPG recibe desde la pantalla o dispositivo ICF. El nombre del archivo WORKSTN debe especificarse en las posiciones 7 a 16. Los campos de entrada deben estar ubicados en el registro de entrada en la misma secuencia en que estén definidos en las DDS. Sin embargo, los nombres de campo no necesitan ser los mismos. Las entradas de ubicación de campo hacen referencia a los campos del registro de entrada.

Para recibir indicadores en la entrada, efectúe una de las siguientes acciones:

- Especifique la palabra clave INDARA en las DDS para el archivo WORKSTN. No utilice la palabra clave PASS(*NOIND) en la especificación de descripción de archivos y no especifique los indicadores en las especificaciones de salida. El programa y el archivo usan un área de indicadores separada para transferir los indicadores.
- Especifique la palabra clave PASS(*NOIND) en la especificación de descripción de archivos. Especifique los indicadores en las especificaciones de entrada como campos con el formato *INxx. Los campos de indicador deben aparecer en el orden especificado por las DDS del archivo WORKSTN. Este orden puede determinarse a partir del listado de las DDS.

Debería asignarse un indicador de identificación de registros a cada registro del archivo para identificar el registro que se ha leído del archivo WORKSTN. En las DDS puede especificarse un campo oculto con un valor por omisión para el código de identificación de registros.

Especificaciones de cálculo

El código de operación READ es válido para un archivo WORKSTN descrito por programa que esté definido como un archivo combinado de control en cálculo. Consulte la Tabla 29 en la página 382. El nombre de archivo debe especificarse en el factor 2 de esta operación. Debe existir un formato en el dispositivo antes que pueda producirse una operación de entrada. Este requisito puede satisfacerse en un dispositivo de pantalla condicionando un registro de salida con el indicador 1P, grabando el primer formato en el dispositivo desde otro programa (por ejemplo en el programa CL). La operación EXFMT no es válida para un archivo WORKSTN descrito por programa. También puede utilizar la operación EXCEPT para grabar en un archivo WORKSTN.

Consideraciones adicionales

Cuando se utiliza un nombre de formato con un archivo WORKSTN descrito por programa, debe considerarse además lo siguiente:

- El nombre especificado en las posiciones 53 a 80 de las especificaciones de salida se supone que es el nombre de un formato de registro en las DDS que se han empleado para crear el archivo.
- Si hay una especificación Kn para un registro de salida, ésta debe usarse también para el resto de registros de salida para ese archivo. Si no se emplea una especificación Kn para todos los registros de salida a un archivo, se producirá un error durante la ejecución.

Utilización de un archivo WORKSTN descrito por programa sin nombre de formato

Cuando no se utiliza un nombre de formato de registro, un archivo de dispositivo de pantalla descrito en el programa describe un archivo que contiene una descripción de formato de registro con un campo. Los campos en el registro deben describirse dentro del programa que utiliza el archivo.

Utilización de archivos WORKSTN descritos por programa

Cuando se crea el archivo de pantalla mediante el mandato Crear archivo de pantalla, el archivo tiene los atributos siguientes:

- Puede especificarse una longitud de registro variable; por lo tanto, la longitud real del registro debe especificarse en el programa que lo utiliza. (La longitud de registro máxima permitida es el tamaño de la pantalla menos uno).
- No se transfieren indicadores ni desde, ni hacia el programa.
- No se definen indicadores de teclas de función.
- El registro se graba en la pantalla comenzando en la posición 2 de la primera línea disponible.

Archivo de entrada

En un archivo de entrada, el registro de entrada, tratado por el soporte de dispositivo OS/400 como campo de entrada único, se inicializa a blancos cuando se abre el archivo. El cursor se sitúa al principio del campo, que es la posición 2 de la pantalla.

Archivo de salida

En un archivo de entrada, el soporte del dispositivo OS/400 trata el registro de salida como una serie de caracteres que se ha de enviar a la pantalla. Cada registro de salida se graba como el siguiente registro secuencial del archivo; es decir, cada registro visualizado se sobrepone al registro visualizado anteriormente.

Archivo combinado

En un archivo combinado, el registro, tratado por el soporte de dispositivo OS/400 como un campo único, aparece en la pantalla y es tanto el registro de salida como el registro de entrada. El soporte del dispositivo inicializa el registro de entrada a blancos, y el cursor se coloca en la posición 2.

Para obtener más información sobre los archivos de dispositivo de pantalla descritos por programa, consulte el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iserics/infocenter>.

Operaciones válidas del archivo WORKSTN

La Tabla 29 muestra los códigos de operación de archivo válidos para un archivo WORKSTN.

Tabla 29. Códigos de operación de archivo válidos para un archivo WORKSTN

Posiciones de especificaciones de descripción de archivo		Posiciones de especificaciones de cálculo
17	18	26-35
I	P/S	CLOSE, ACQ, REL, NEXT, POST, FORCE
I	P/S	WRITE ¹ , CLOSE, ACQ, REL, NEXT, POST, FORCE
I	F	READ, OPEN, CLOSE, ACQ, REL, NEXT, POST
C	F	READ, WRITE ¹ , EXFMT ² , OPEN, CLOSE, ACQ, REL, NEXT, POST, UPDATE ³ , CHAIN ³ , READC ³
O	En blanco	WRITE ¹ , OPEN, CLOSE, ACQ, REL, POST

Operaciones válidas del archivo WORKSTN

Tabla 29. Códigos de operación de archivo válidos para un archivo WORKSTN (continuación)

Posiciones de especificaciones de descripción de archivo	Posiciones de especificaciones de cálculo
Notas: 1. La operación WRITE no es válida para un archivo descrito por programa que se utilice con un nombre de formato. 2. Si se utiliza la operación EXFMT, el archivo debe estar descrito externamente (una E en la posición 19 de las especificaciones de descripción de archivos). 3. Para formatos de registro de subarchivo, también son válidas las operaciones UPDATE, CHAIN y READC.	

A continuación se explican los códigos de operación EXFMT, READ y WRITE cuando se utilizan para procesar un archivo WORKSTN.

Operación EXFMT

La operación EXFMT es una combinación de una operación WRITE seguida de una operación READ para el mismo formato de registro (corresponde a la operación WRITE-READ de gestión de datos). Si define un archivo WORKSTN en las especificaciones de descripción de archivos como un archivo combinado (C en la posición 17), controlado en cálculo (F en la posición 18), que utiliza datos descritos externamente (E en la posición 22), puede usarse el código de operación EXFMT (ejecutar formato) para grabar y leer en la pantalla.

Operación READ

La operación READ es válida para un archivo combinado controlado en cálculo o para un archivo de entrada controlado en cálculo que utilice los datos descritos externamente o los datos descritos por programa. La operación READ recupera un registro de la pantalla. No obstante, debe haber un formato en el dispositivo antes que pueda producirse una operación de entrada. Esta necesidad puede satisfacerse en un dispositivo de pantalla condicionando un registro de salida con el indicador 1P, grabando el primer formato en el dispositivo desde otro programa o, si la lectura se efectúa mediante el nombre de formato de registro, empleando la palabra clave INZRCD en la descripción de registro de las DDS.

Operación WRITE

La operación WRITE graba un nuevo registro en una pantalla y es válida para un archivo combinado o un archivo de salida. Las especificaciones de salida y la operación EXCEPT también se pueden utilizar para grabar en un archivo WORKSTN. Consulte la publicación *ILE RPG Reference* para obtener una descripción completa de cada uno de estos códigos de operación.

Archivos de múltiples dispositivos

Cualquier archivo WORKSTN en RPG que tenga especificada por lo menos una de las palabras clave DEVID, SAVEIND, MAXDEV(KFILE) o SAVDS especificadas en la descripción de especificación de archivos es un archivo de múltiples dispositivos. Mediante un archivo de múltiples dispositivos, el programa puede acceder a más de un dispositivo.

Archivos de múltiples dispositivos

El programa RPG accede a los dispositivos a través de los dispositivos del programa, los cuales son un mecanismo simbólico para dirigir las operaciones a un dispositivo real. Cuando el usuario crea un archivo (utilizando las especificaciones de las DDS y mandatos tales como los de creación de archivo), debe considerar cosas tales como qué dispositivo está asociado con un dispositivo de programa, si un archivo tiene o no un dispositivo solicitante, qué formatos de registro habrán de utilizarse para pedir a los dispositivos que respondan a una operación READ por nombre de archivo y cuanto tiempo habrá de esperar una respuesta a esta operación READ. Para obtener más información sobre las opciones y requisitos para la creación de un archivo de múltiples dispositivos, consulte el capítulo sobre archivos de pantalla en el apartado *DB2 Universal Database para AS/400* de la categoría *Bases de datos y sistemas de archivos* en **iSeries 400 Information Center** en este sitio Web: <http://www.ibm.com/eserver/iseres/infocenter>. También puede consultar la información sobre los archivos ICF en el manual *ICF Programming*.

Con archivos de múltiples dispositivos se hace un uso particular de los siguientes códigos de operación:

- Además de abrir un archivo, la operación OPEN adquiere implícitamente el dispositivo que usted especifica cuando crea el archivo.
- La operación ACQ (adquirir) adquiere cualquier otro dispositivo para un archivo de múltiples dispositivos.
- La operación REL (liberar) libera un dispositivo del archivo.
- La operación WRITE, cuando se utiliza con la palabra clave de las DDS INVITE, pide a un dispositivo de programa que responda a las siguientes operaciones de leer de los dispositivos de programa solicitados. Consulte el apartado sobre invitar un dispositivo de programa en el manual *ICF Programming*.
- La operación READ efectúa tanto una operación de leer desde los dispositivos de programa solicitados como una operación de leer desde un dispositivo de programa. Cuando no hay ninguna operación NEXT en vigor, una operación de lectura en ciclo de programa o una operación READ por nombre de archivo, esperará la entrada desde cualquiera de los dispositivos a los que se les haya pedido responder (leer desde dispositivo de programa solicitado). Otras operaciones de entrada y de salida, incluyendo una operación READ por nombre de archivo después de una operación NEXT y de una operación READ por nombre de formato, efectúan una operación de leer desde un dispositivo de programa utilizando el dispositivo de programa indicado en un campo especial. (Se da nombre al campo en la palabra clave DEVID de las líneas de especificación de descripción de archivos).

Este dispositivo puede ser el utilizado en la última operación de entrada, un dispositivo especificado por el usuario o el dispositivo del programa solicitante. Consulte los apartados sobre la lectura desde dispositivos de programa invitados y sobre la lectura desde un dispositivo de programa en el manual *ICF Programming*.

- La operación NEXT especifica el dispositivo que ha de utilizarse en la siguiente operación READ por nombre de archivo o en la siguiente operación de leer en ciclo de programa.
- La operación POST coloca información en la estructura de datos de información INFDS. La información puede ser sobre un dispositivo específico o sobre el archivo. (La utilización de la operación POST no está limitada a los archivos de múltiples dispositivos).

Consulte la *ILE RPG Reference* para obtener detalles de los códigos de operación RPG.

En la especificación de descripción de archivos puede especificar varias palabras clave para controlar el proceso de archivos de múltiples dispositivos.

- La palabra clave MAXDEV indica si el archivo es un archivo de un solo dispositivo o de múltiples dispositivos.

Especifique MAXDEV(*FILE) para procesar un archivo de múltiples dispositivos tomando el máximo número de dispositivos de la definición del archivo que está procesándose. Especifique MAXDEV(*ONLY) para procesar sólo un dispositivo.

- La palabra clave DEVID le permite especificar el nombre de un dispositivo de programa al que se envían operaciones de entrada y salida.

Cuando se emite una operación de lectura desde dispositivo de programa o WRITE, el dispositivo utilizado para la operación es el dispositivo especificado como el parámetro de la palabra DEVID. Este campo se inicializa a blancos y se actualiza con el nombre del dispositivo desde el cual se ha producido la última operación satisfactoria de entrada. También puede establecerse explícitamente trasladando un valor al mismo. El código de operación ACQ no afecta al valor de este campo. Si no se especifica la palabra clave DEVID, la operación de entrada se realiza para el dispositivo desde el que tuvo lugar la última operación de entrada satisfactoria. Se utiliza un nombre de dispositivo en blanco si aún no se ha producido con éxito ninguna operación de lectura desde un dispositivo.

Cuando se emite una operación de lectura desde un dispositivo de programa o WRITE con un nombre de dispositivo en blanco, el compilador de RPG utiliza implícitamente el nombre de dispositivo del dispositivo peticionario para el programa. Si el usuario llama un programa en RPG interactivamente y adquiere un dispositivo ICF en el cual desea realizar una de estas operaciones, debe trasladar explícitamente el nombre del dispositivo ICF al nombre de campo especificado con la palabra clave DEVID antes de efectuar la operación. Si no se hace esto, el nombre de dispositivo utilizado estará en blanco (en cuyo caso se utiliza el nombre de dispositivo peticionario interactivo) o el nombre de dispositivo utilizado será el de la última operación satisfactoria de entrada. Una vez que el usuario haya realizado una operación de E/S en el dispositivo ICF, no necesita volver a modificar el valor a menos que se complete con éxito una operación de entrada con un dispositivo distinto.

- La palabra clave SAVEDS indica una estructura de datos que se ha salvado y restaurado para cada dispositivo adquirido para un archivo. La palabra clave SAVEIND indica un juego de indicadores que se ha salvado y restaurado para cada dispositivo adquirido para un archivo. Antes de una operación de entrada, se salvan el grupo en curso de indicadores y la estructura de los datos. Después de la operación de entrada, el compilador de RPG restaura los indicadores y la estructura de datos para el dispositivo asociado con la operación. El grupo de indicadores o la estructura de datos pueden ser diferentes de los que estaban disponibles antes de la operación de entrada.
- La palabra clave INFDS especifica la estructura de datos de información del archivo para el archivo WORKSTN. Puede accederse al campo de RPG *STATUS y al código de retorno mayor/menor para la operación de E/S mediante esta estructura de datos. En particular cuando se utiliza ICF, los dos campos son útiles para la detección de los errores que se han producido durante las operaciones de E/S para archivos de múltiples dispositivos.

Nota: Cuando se especifiquen estas opciones de control, se debe codificar la opción MAXDEV antes de las opciones DEVID, SAVEIND o SAVEDS.

Archivos de múltiples dispositivos

Capítulo 20. Ejemplo de una aplicación interactiva

Este capítulo ilustra algunas aplicaciones de estación de trabajo comunes y su codificación en ILE RPG.

El programa de aplicación que se muestra en este capítulo se compone de cuatro módulos. Cada módulo es un ejemplo de la utilización más frecuente de los archivos WORKSTN. El primer módulo (CLIPRIN) proporciona el menú principal del programa. Dependiendo de la selección del usuario, llama al procedimiento del módulo adecuado que proporciona la función solicitada.

Cada módulo utiliza un archivo WORKSTN para solicitar al usuario que entre y visualice información en la pantalla. Cada módulo, excepto el módulo principal CLIPRIN, también utiliza un archivo lógico que muestra una *vista* del archivo de base de datos maestro. Esta vista se compone únicamente de los campos del archivo maestro que el módulo requiere para su proceso.

Nota: Cada módulo, excepto CLIPRIN, puede compilarse como un programa autónomo, es decir, cada módulo puede utilizarse como un programa independiente.

Tabla 30. Descripción de cada uno de los módulos en el ejemplo de aplicación interactiva

Módulo	Descripción
"Consulta de menú principal" en la página 388	Un ejemplo de un programa básico de consulta de menús que utiliza el archivo WORKSTN para visualizar opciones de menú y aceptar entradas.
"Mantenimiento de archivo" en la página 391	Un ejemplo de un programa de mantenimiento que permite actualizar, suprimir, añadir y visualizar registros de clientes en un archivo maestro.
"Búsqueda por código postal" en la página 402	Un ejemplo de un programa que utiliza el proceso del subarchivo WORKSTN para visualizar todos los registros coincidentes para un código postal especificado.
"Búsqueda y consulta por nombre" en la página 410	Un ejemplo de un programa que utiliza el proceso del subarchivo WORKSTN para visualizar todos los registros coincidentes para un nombre de cliente especificado, y permite al usuario seleccionar un registro del subarchivo para visualizar la información del cliente completa.

Archivo físico de base de datos

La Figura 186 en la página 388 muestra las especificaciones de descripción de datos (DDS) para el archivo maestro de clientes. Este archivo contiene información importante sobre cada cliente, como el nombre, la dirección, el saldo de la cuenta y el número de cliente. Cada módulo que necesita información de este cliente utiliza este archivo de base de datos (o a una vista lógica del mismo).

Consulta de menú principal

```
A*****
A*NOMBRE ARCHIVO: MAECLI *
A*   PROG. REL.: MNTCLI, BUSCOD, NOMBUS *
A* ARCHIVOS REL.: MAECLIL1, MAECLIL2, MAECLIL3 (ARCHIVOS LÓGICOS)*
A*   DESCRIPCIÓN: ESTE ES EL ARCHIVO FÍSICO MAECLI. TIENE *
A*                   UN FORMATO DE REGISTRO LLAMADO REGCLIE. *
A*****
A* ARCHIVO MAESTRO CLIENTES -- MAECLI
A      R REGCLIE
A      CLIE          5  0      TEXT('NÚMERO CLIENTE')
A      NOMB          20        TEXT('NOMBRE CLIENTE')
A      DIRE1         20        TEXT('DIRECCIÓN CLIENTE')
A      DIRE2         20        TEXT('DIRECCIÓN CLIENTE')
A      CDAD          20        TEXT('CIUDAD CLIENTE')
A      PROV          2         TEXT('PROVINCIA CLIENTE')
A      CDP            5  0      TEXT('CÓDIGO POSTAL CLIENTE')
A      SALDO         10  2      TEXT('SALDO DEL DEBE')
```

Figura 186. DDS para el archivo maestro de base de datos MAECLI (archivo físico)

Consulta de menú principal

A continuación se muestra un programa de consulta sencillo que utiliza un archivo WORKSTN para visualizar opciones de menú y aceptar entradas.

MENUPRIN: DDS para un archivo de dispositivo de pantalla

La DDS para el archivo de dispositivo de pantalla MENUPRIN visualiza entradas a nivel de archivo y describe un formato de registro: HDRSCN. Las entradas a nivel de archivo definen el tamaño de la pantalla (DSPSIZ), los valores por omisión de entrada (CHGINPDFT), la tecla de impresión (PRINT) y un área de indicador diferente (INDARA).

El formato de registro HDRSCN contiene la constante CONSULTA PRINCIPAL CLIENTE, que identifica la pantalla. También contiene las palabras clave TIME y DATE, que visualizarán la fecha y la hora actuales en la pantalla. Las palabras clave CA definen las teclas de función que pueden utilizarse y asocian las teclas de función con los indicadores de un programa en RPG.

```

A*****
A*NOMBRE ARCHIVO:  MENUPRIN                      *
A*   PROG. REL.:   CLIPRIN                      *
A*   DESCRIPCIÓN:  ES EL ARCHIVO DE PANTALLA MENUPRIN. TIENE 1 *
A*                   FORMATO DE REGISTRO LLAMADO HDRSCN.      *
A*****
A                                DSPSIZ(24 80 *DS3)
A                                CHGINPDFT(CS)
A                                PRINT(QSYSPRT)
A                                INDARA
A                                R HDRSCN
A                                CA03(03 'FINAL DE CONSULTA')
A                                CA05(05 'MODO MANTENIMIENTO')
A                                CA06(06 'BÚSQ. POR MODO CÓD. POSTAL')
A                                CA07(07 'BÚSQUEDA DE MODO NOMBRE')
A                                2 4TIME
A                                DSPATR(HI)
A                                2 28'CONSULTA PRINCIPAL DE CLIENTES'
A                                DSPATR(HI)
A                                DSPATR(RI)
A                                2 70DATE
A                                EDTCDE(Y)
A                                DSPATR(HI)
A                                6 5'Pulse una de las siguientes'
A                                6 32'teclas PF'
A                                8 22'F3 Finalizar trabajo'
A                                9 22'F5 Mantener archivo de cliente'
A                                10 22'F6 Buscar cliente por código postal'
A                                11 22'F7 Buscar cliente por nombre'

```

Figura 187. DDS para el archivo de dispositivo de pantalla MENUPRIN

Además de describir las constantes, los campos, los números de línea y las posiciones horizontales para la pantalla, los formatos de registro definen también los atributos de pantalla para estas entradas.

Nota: Normalmente los atributos de campo se definen en el archivo de referencias de campo en lugar de en las DDS para un archivo. Los atributos se muestran en las DDS para que pueda observar cuales son.

Consulta de menú principal

CLIPRIN: fuente RPG

```
//*****  
// NOMB PROGRAMA: CLIPRIN *  
// ARCHIVOS REL.: MENUPRIN (DSPF) *  
// PROGRAMAS REL: MNTCLI (ILE RPG PGM) *  
// BUSCOD (ILE RPG PGM) *  
// NOMBUS (ILE RPG PGM) *  
// DESCRIPCIÓN: Es un programa de consulta principal de *  
// cliente. Insta al usuario a elegir entre *  
// las siguientes acciones: *  
// 1.Mantener (añadir, actualizar, suprimir y *  
// ver) registros de cliente. *  
// 2.Buscar registro de cliente por código postal*  
// 3.Buscar registro de cliente por nombre. *  
//*****  
  
Fmenuprin cf e workstn indds(indicadores)  
  
// Definiciones de prototipo:  
D CustMaintain pr extproc('MNTCLI')  
D SearchZip pr extproc('BUSCOD')  
D SearchName pr extproc('NOMBUS')  
  
// Definiciones de campo:  
D indicadores ds  
D exitKey n overlay(indicadores:3)  
D maintainKey n overlay(indicadores:5)  
D srchZipKey n overlay(indicadores:6)  
D srchCustKey n overlay(indicadores:7)  
  
/free  
// Mantener bucles hasta pulsar tecla de salir  
dow '1';  
// Mostrar menú principal  
exfmt hdrscn;  
  
// Realizar acción solicitada  
if exitKey;  
// Salir del programa  
leave;  
  
elseif maintainKey;  
// Mantener datos de cliente  
CustMaintain();  
  
elseif srchZipKey;  
// Buscar datos de cliente por código postal  
SearchZip();  
  
elseif srchCustKey;  
// Buscar datos de clientes por nombre de cliente  
SearchName();  
endif;  
enddo;  
  
*inlr = *on;  
/end-free
```

Figura 188. Fuente para el módulo CLIPRIN

Consulta de menú principal

Este módulo muestra el código de operación CALLB. CLIPRIN llama al módulo RPG adecuado (MNTCLI, BUSCOD o NOMBUS) dependiendo de la selección de elementos de menú del usuario.

Para crear el objeto de programa:

1. Cree un módulo para cada miembro fuente (CLIPRIN, MNTCLI, BUSCOD, NOMBUS) utilizando CRTRPGMOD.

2. Cree el programa entrando:

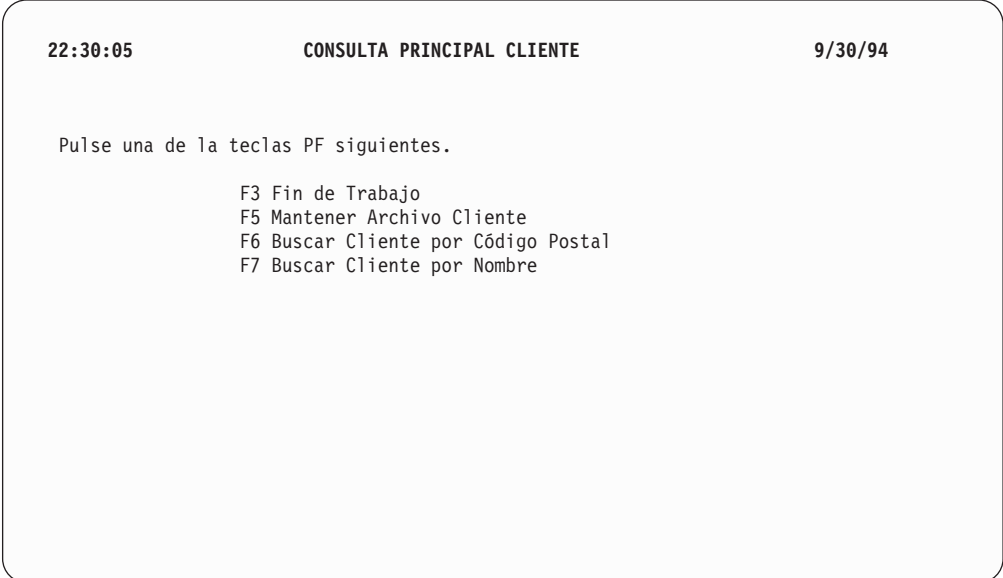
```
CRTPGM PGM(MIPROG) MODULE(CLIPRIN MNTCLI BUSCOD NOMBUS) ENTMOD(*FIRST)
```

Nota: La opción *FIRST especifica que se ha seleccionado el primer módulo de la lista, CLIPRIN, como el procedimiento de entrada del programa.

3. Llame al programa entrando:

```
CALL MIPROG
```

El "menú principal" aparecerá como en la Figura 189.



22:30:05 CONSULTA PRINCIPAL CLIENTE 9/30/94

Pulse una de la teclas PF siguientes.

F3 Fin de Trabajo
F5 Mantener Archivo Cliente
F6 Buscar Cliente por Código Postal
F7 Buscar Cliente por Nombre

Figura 189. Pantalla de solicitud Consulta Principal Cliente

Mantenimiento de archivo

A continuación se muestra un programa de mantenimiento utilizando el archivo WORKSTN. Este programa le permite añadir, suprimir, actualizar y visualizar registros del archivo maestro de clientes.

MAECLIL1: DDS para un archivo lógico

```
A*****
A*NOMBRE ARCHIVO:  MAECLIL1                      *
A* PROGRAMA REL.:  MNTCLI                          *
A* ARCHIVOS REL.:  MAECLI  (ARCHIVO FÍSICO)         *
A*  DESCRIPCIÓN:   ESTE ES EL ARCHIVO LÓGICO MAECLIL1. *
A*                  CONTIENE EL FORMATO DE REGISTRO CMLREC1. *
A*                  VISTA LÓGICA ARCHIVO MAESTRO CLIENTE (MAECLI) *
A*                  POR NÚMERO DE CLIENTE (CLIE)      *
A*****
A          R CMLREC1                                PFILE(MAECLI)
A          CLIE
A          NOMB
A          DIRE1
A          DIRE2
A          CDAD
A          PROV
A          CDP
A          K CLIE
```

Figura 190. DDS para el archivo lógico MAECLIL1

Las DDS para el archivo de base de datos utilizado por este programa describen un formato de registro: CMLREC1. Se describe cada campo del formato de registro y el campo CLIE se identifica como el campo de clave para el formato de registro.

MENUMNT: DDS para un archivo de dispositivo de pantalla

```
A*****
A*NOMBRE ARCHIVO:  NOMBRE ARCHIVO:  MENUMNT      *
A*PROGRAMAS REL.:  MNTCLIE           *
A* ARCHIVOS REL.:  MAECLIL1  (ARCHIVO LÓGICO)      *
A*  DESCRIPCIÓN:  ESTE ES EL ARCHIVO DE PANTALLA MENUMNT. TIENE *
A*                3 FORMATOS DE REGISTRO.          *
A*****
A                                REF(MAECLIL1)
A                                CHGINPDFT(CS)
A                                PRINT(QSYSVRT)
A                                INDARA
A      R HDRSCN
A                                TEXT('SOLICITAR NÚMERO CLIE')
A                                CA03(03 'FINAL DE MANTENIMIENTO')
A                                CF05(05 'MODULO ADICIÓN')
A                                CF06(06 'MODULO ACTUALIZACIÓN')
A                                CF07(07 'MODULO SUPRESIÓN')
A                                CF08(08 'MODULO VISUALIZACIÓN')
A      MODALIDAD      8A  0  1  4DSPATR(HI)
A                                1 13'MODE'
A                                DSPATR(HI)
A                                2  4TIME
A                                DSPATR(HI)
A                                2 28'MANTENIMIENTO ARCHIVO CLIENTE'
A                                DSPATR(HI RI)
A                                2 70DATE
A                                EDTCDE(Y)
A                                DSPATR(HI)
A      CLIE      R      Y  I 10 25DSPATR(CS)
A                                CHECK(RZ)
A 51  ERRMSG('CLIENTE YA EN +
A      ARCHIVO' 51)
A 52  ERRMSG('CLIENTE NO EN ARCHIVO' +
A      52)
A                                10 33'<--Entre número de cliente'
A                                DSPATR(HI)
A                                23  4'F3 Finalizar trabajo'
A                                23 21'F5 Añadir'
A                                23 34'F6 Actualizar'
A                                23 50'F7 Suprimir'
A                                23 66'F8 Visualizar'
```

Figura 191. DDS para el archivo de dispositivo de pantalla MENUMNT (Pieza 1 de 2)

Mantenimiento de archivo

R CSTINQ						TEXT('VISUALIZAR INFO CLIE')	
						CA12(12 'PANTALLA ANTERIOR')	
	MODALIDAD	8A	0	1	4	DSPATR(HI)	
				1	13	'MODE'	
						DSPATR(HI)	
				2	4	TIME	
						DSPATR(HI)	
				2	28	'MANTENIMIENTO ARCHIVO CLIENTE'	
						DSPATR(HI)	
						DSPATR(RI)	
				2	70	DATE	
						EDTCDE(Y)	
						DSPATR(HI)	
				4	14	'Cliente:'	
						DSPATR(HI)	
						DSPATR(UL)	
	CLIE	R	0	4	25	DSPATR(HI)	
	NOMB	R	B	6	25	DSPATR(CS)	
04						DSPATR(PR)	
	DIRE1	R	B	7	25	DSPATR(CS)	
04						DSPATR(PR)	
	DIRE2	R	B	8	25	DSPATR(CS)	
04						DSPATR(PR)	
	CDAD	R	B	9	25	DSPATR(CS)	
04						DSPATR(PR)	
	PROV	R	B	10	25	DSPATR(CS)	
04						DSPATR(PR)	
	CODE	R	B	10	40	DSPATR(CS)	
						EDTCDE(Z)	
04						DSPATR(PR)	
				23	2	'F12 Cancelar'	
	MODALIDAD1	8	0	23	13		
	R CSTBLD					TEXT('AÑADIR REGISTRO CLIE')	
						CA12(12 'PANTALLA ANTERIOR')	
	MODALIDAD	8	0	1	4	DSPATR(HI)	
				1	13	'MODALIDAD'	DSPATR(HI)
				2	4	TIME	
						DSPATR(HI)	
				2	28	'MANTENIMIENTO ARCHIVO CLIENTE'	
						DSPATR(HI RI)	
				2	70	DATE	
						EDTCDE(Y)	
						DSPATR(HI)	
				4	14	'Cliente:'	DSPATR(HI UL)
	CLIE	R	0	4	25	DSPATR(HI)	
				6	20	'Nombre'	DSPATR(HI)
	NOMB	R	I	6	25		
				7	17	'Dirección'	DSPATR(HI)
	DIRE1	R	I	7	25		
				8	17	'Dirección'	DSPATR(HI)
	DIRE2	R	I	8	25		
				9	20	'Ciudad'	DSPATR(HI)
	CDAD	R	I	9	25		
				10	19	'Provincia'	DSPATR(HI)
	PROV	R	I	10	25		
				10	36	'Código postal'	DSPATR(HI)
	CDP	R	Y	I	10	40	
				23	2	'F12 Cancelar adición'	

Figura 191. DDS para el archivo de dispositivo de pantalla MENU MNT (Pieza 2 de 2)

Las DDS del archivo del dispositivo de pantalla MENUMNT contienen tres formatos de registro: HDRSCN, CSTINQ y CSTBLD. El registro HDRSCN solicita el número de cliente y la modalidad de proceso. El registro CSTINQ se utiliza para

las modalidades de Actualización, Supresión y Visualización. Los campos se definen como entrada/salida (B en la posición 38). Los campos están protegidos cuando se selecciona la modalidad de Visualización o Supresión (DSPATR(PR)). El registro CSTBLDT sólo proporciona campos de entrada para un nuevo registro (I en la posición 38).

El formato de registro HDRSCN contiene la constante 'Consulta Principal Cliente'. La palabra clave ERRMSG define los mensajes que se han de visualizar si se produce un error. Las palabras clave CA definen las teclas de función que pueden utilizarse y asocian las teclas de función con los indicadores de un programa en RPG.

MNTCLI: Fuente RPG

```
//*****
// NOMBRE PROGR.: MNTCLI *
// ARCHIVOS REL.: MAECLIL1 (LF) *
// MENU MNT (DSPF) *
// DESCRIPCIÓN: Este programa muestra un programa de mantenimiento *
// de maestro de cliente que utiliza un archivo workstn. *
// Este programa permite al usuario añadir, actualizar, *
// suprimir y visualizar registros de cliente. *
// PF3 se utiliza para salir del programa. *
//*****

Fmaeclil1 uf a e k disk
Fmenu mnt cf e workstn indds(indicadores)

// Definiciones de campo:

D indicadores ds
D exitKey n overlay(indicadores:3)
D disableInput n overlay(indicadores:4)
D addKey n overlay(indicadores:5)
D updateKey n overlay(indicadores:6)
D deleteKey n overlay(indicadores:7)
D displayKey n overlay(indicadores:8)
D prevKey n overlay(indicadores:12)
D custExists n overlay(indicadores:51)
D custNotFound n overlay(indicadores:52)
// Definiciones de lista de claves:

C CSTKEY KLIST
C KFLD CLIE
```

Figura 192. Fuente para módulo MNTCLI (Pieza 1 de 5)

Mantenimiento de archivo

```
//*****  
//      MAINLINE      *  
//*****  
  
/free  
  
mode = 'DISPLAY';  
exfmt hdrscn;  
  
// Bucles hasta pulsar tecla de salir  
dow not exitKey;  
    exsr SetMaintenanceMode;  
  
    if cust <> 0;  
        if mode = 'A';  
            exsr AddSub;  
        elseif mode = 'UPDATE';  
            exsr UpdateSub;  
        elseif mode = 'DELETE';  
            exsr DeleteSub;  
        elseif mode = 'DISPLAY';  
            exsr InquirySub;  
        endif;  
    endif;  
  
    exfmt hdrscn;  
    custExists = *off; // desactivar mensajes de error  
    CustNotFound = *off;  
enddo;  
  
*inlr = *on;
```

Figura 192. Fuente para módulo MNTCLI (Pieza 2 de 5)

```
//*****
//  SUBROUTINA  - AddSub                                     *
//  OBJETIVO   - Añadir nuevo cliente a archivo             *
//*****
begsr AddSub;

    // ¿Se encuentra ya el número de cliente en el archivo?
    chain CstKey cmlrecl;
    if %found(maekl11);
        // El número de cliente ya se está utilizando
        custExists = *on;
        leavesr;
    endif;

    // Inicializar nuevos datos de cliente
    custExists = *off; // desactivar mensajes de error
    CustNotFound = *off;
    nomb = *blank;
    dire1= *blank;
    dire2= *blank;
    cdad = *blank;
    prov = *blank;
    cdp = 0;

    // Solicitar datos actualizados para este registro de cliente
    exfmt cstbld;

    // Si es correcto, añadir cliente al archivo de clientes
    if not *inl2;
        write cmlrecl;
    endif;
endsr; // final de subrutina AddSub


//*****
//  SUBROUTINA  - UpdateSub                                   *
//  OBJETIVO   - Actualizar registro maestro de clientes     *
//*****
begsr UpdateSub;

    // Buscar número de cliente
    chain cstkey cmlrecl;
    if not %found(maekl11);
        // No se ha encontrado el cliente en el archivo
        custNotFound = *on;
        leavesr;
    endif;

    // Mostrar información para este cliente
    disableInput = *off;
    exfmt cstinq;
    if not prevKey;
        // Actualizar información en archivo
        update cmlrecl;
    else;
        // Si no desea actualizar, como mínimo desbloquee
        // el registro.
        unlock maekl11;
    endif;
endsr; // final de subrutina UpdateSub;
```

Figura 192. Fuente para módulo MNTCLI (Pieza 3 de 5)

Mantenimiento de archivo

```
//*****
//  SUBROUTINA - DeleteSub
//  OBJETIVO   - Suprimir registro maestro de clientes
//*****
begsr DeleteSub;

    // Buscar número de cliente
    chain cstkey cmlrec1;
    if not %found(maeklil1);
        // No se ha encontrado el cliente en el archivo
        custNotFound = *on;
        leavesr;
    endif;

    // Mostrar información para este cliente
    disableInput = *on;
    exfmt cstinq;
    if not prevKey;
        // Suprimir registro de clientes
        delete cmlrec1;
    else;
        // Si no desea suprimir, como mínimo desbloquee
        // el registro.
        unlock csmstl1;
    endif;
endsr; // final de subrutina DeleteSub


//*****
//  SUBROUTINA - InquirySub
//  OBJETIVO   - Visualizar registro maestro de cliente
//*****
begsr InquirySub;

    // Buscar número de cliente
    chain(n) cstkey cmlrec1; // no bloquee el registro
    if not %found(maeklil1);
        // No se ha encontrado el cliente en el archivo
        custNotFound = *on;
        leavesr;
    endif;

    // Mostrar información para este cliente
    disableInput = *on;
    exfmt cstinq;
endsr; // final de subrutina InquirySub;
```

Figura 192. Fuente para módulo MNTCLI (Pieza 4 de 5)


```

//*****
//  SUBROUTINA  - SetMaintenanceMode          *
//  OBJETIVO   - Establecer modalidad de mantenimiento  *
//*****
begsr SetMaintenanceMode;
  if addKey;
    mode = 'ADD';
  elseif updateKey;
    mode = 'UPDATE';
  elseif deleteKey;
    mode = 'DELETE';
  elseif displayKey;
    mode = 'DISPLAY';
  endif;
endsr; // final de subrutina SetMaintenanceMode

/end-free

```

Figura 192. Fuente para módulo MNTCLI (Pieza 5 de 5)

Este programa mantiene un archivo maestro de clientes para adiciones, cambios y supresiones. El programa también puede utilizarse para consultas.

El programa establece la modalidad de proceso por omisión (visualización) y muestra la pantalla de solicitud de mantenimiento del cliente. El usuario de la estación de trabajo puede pulsar F3, lo cual activa el indicador 03, para solicitar el final del trabajo. El usuario también puede entrar un número de cliente y pulsar Intro para trabajar con la información de cliente. El usuario puede cambiar la modalidad de proceso pulsando la tecla F5 (AÑADIR), F6 (ACTUALIZAR), F7 (SUPRIMIR) o F8 (MOSTRAR).

Para añadir un nuevo registro al archivo, el programa utiliza el número de cliente como el argumento de búsqueda que se encadenará al archivo maestro. Si no existe el registro en el archivo, el programa visualiza la pantalla CSTBLD para permitir que el usuario entre un nuevo registro de cliente. Si el registro ya está en el archivo, se visualiza un mensaje de error. El usuario puede pulsar la tecla F12, la cual activa el indicador 12, para cancelar la operación de adición y liberar el registro. De lo contrario, para seguir con la operación de adición, el usuario entra información para el nuevo registro de cliente en los campos de entrada y graba el nuevo registro en el archivo maestro.

Para actualizar, suprimir o visualizar un registro existente, el programa utiliza el número de cliente como argumento de búsqueda para encadenar al archivo maestro. Si existe en el archivo un registro para ese cliente, el programa visualiza la pantalla de consulta del archivo de clientes CSTINQ. Si el registro no está en el archivo, se visualiza un mensaje de error. Si la modalidad de proceso es visualizar o suprimir, los campos de entrada están protegidos contra modificación. De lo contrario, para seguir con el registro del cliente, el usuario puede entrar nueva información en los campos de entrada del registro de cliente. El usuario puede pulsar la tecla F12, la cual activa el indicador 12, para cancelar la actualización y liberar el registro. La modalidad de visualización libera automáticamente el registro cuando se pulsa la tecla Intro.

En la Figura 193 en la página 400, el usuario de la estación de trabajo responde a la solicitud entrando el número de cliente 00007 para visualizar el registro de cliente.

Mantenimiento de archivo

MODALIDAD VISUALIZACIÓN	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
22:30:21		
00007 <---Entre número de Cliente		
F3 Fin de Trabajo F5 Añadir F6 Actualizar F7 Suprimir F8 Visualizar		

Figura 193. Pantalla de solicitud en modalidad de visualización 'Mantenimiento Archivo Cliente'

Puesto que existe el registro de cliente para el número de cliente 00007 en el Archivo Maestro, los datos se visualizan como se muestra en la Figura 194.

MODALIDAD VISUALIZACIÓN	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
22:31:06		
Cliente:	00007	
Miguel García		
Entenza, 16		
Suite 1702		
Barcelona		
BCN	08015	
F12 Cancelar VISUALIZACION		

Figura 194. Pantalla en modalidad de visualización 'Mantenimiento Archivo Cliente'

El usuario de la estación de trabajo responde a la solicitud de adición entrando un número de cliente nuevo tal como se muestra en la Figura 195 en la página 401.

MODALIDAD ADICIÓN 22:31:43	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
<p>00012 <---Entre Número de Cliente</p>		
<p>F3 Fin de Trabajo F5 Añadir F6 Actualizar F7 Suprimir F8 Visualizar</p>		

Figura 195. Pantalla de solicitud en modalidad de adición 'Mantenimiento Archivo Cliente'

En la Figura 196 se añade un nuevo cliente al Archivo Maestro de Clientes.

MODALIDAD ADICIÓN 22:32:04	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
<p><u>Cliente:</u> 00012</p>		
<p>Nombre JUAN PEREZ Dirección AVDA. DIAGONAL, 201 Dirección Ciudad BARCELONA Provincia BCN Código postal 08028</p>		
<p>F12 Cancelar Adición</p>		

Figura 196. Pantalla de solicitud en modalidad de adición 'Mantenimiento Archivo Cliente'

El usuario de la estación de trabajo responde a la solicitud de supresión entrando un número de cliente tal como se muestra en la Figura 197 en la página 402.

Mantenimiento de archivo

MODALIDAD SUPRESIÓN 22:32:55	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
00011 <--Entre Número Cliente		
F3 Fin de Trabajo F5 Añadir F6 Actualizar F7 Suprimir F8 Visualizar		

Figura 197. Pantalla de solicitud en modalidad de supresión 'Mantenimiento Archivo Cliente'

El usuario de la estación de trabajo responde a la solicitud de actualización entrando un número de cliente tal como se muestra en la Figura 198.

MODALIDAD ACTUALIZACIÓN 22:33:17	MANTENIMIENTO ARCHIVO CLIENTES	9/30/94
00010 <--Entre Número Cliente		
F3 Fin de Trabajo F5 Añadir F6 Actualizar F7 Suprimir F8 Visualizar		

Figura 198. Pantalla de solicitud en modalidad de actualización 'Mantenimiento Archivo Cliente'

Búsqueda por código postal

A continuación se muestra el proceso del subarchivo WORKSTN (sólo pantalla). Los subarchivos se utilizan para visualizar todos los registros coincidentes para un código postal especificado.

MAECLIL2: DDS para un archivo lógico

```
A*****
A*NOMBRE ARCHIVO: MAECLIL2 *
A*PROGRAMAS REL.: BUSCOD *
A* ARCHIVOS REL.: MAECLI (ARCHIVO FÍSICO) *
A* DESCRIPCIÓN: ESTE ES EL ARCHIVO LÓGICO MAECLIL2. *
A* CONTIENE UN FORMATO DE REGISTRO LLAMADO CMLREC2. *
A* VISTA LÓGICA DE ARCHIVO MAESTRO DE CLIENTES (MAECLI) *
A* POR CÓDIGO POSTAL DE CLIENTE (COD) *
A*****
A R CMLREC2 PFILE(MAECLI)
A CDP
A NOMB
A SALDO
A K CDP
```

Figura 199. DDS para archivo lógico MAECLIL2

Las DDS para el archivo de base de datos utilizado por este programa describen un formato de registro CMLREC2. El archivo lógico MAECLIL2 especificado por distrito postal se basa en el archivo físico MAECLI, tal como indica la palabra clave PFILE. El formato de registro generado por el archivo lógico incluirá solamente los campos especificados en el archivo lógico DDS. Se excluirán el resto de campos.

MENUBCOD: DDS para un archivo de dispositivo de pantalla

```

A*****
A*NOMBRE ARCHIVO:  MENUBCOD                                     *
A*PROGRAMAS REL.:  BUSCOD                                       *
A* ARCHIVOS REL.:  MAECLIL2   (ARCHIVO LÓGICO)                 *
A*  DESCRIPCIÓN:  ESTE ES EL ARCHIVO DE PANTALLA MENUBCOD. TIENE 6 *
A*                                     FORMATOS DE REGISTRO.      *
A*****
A                                     REF(MAECLIL2)
A                                     CHGINPDFT(CS)
A                                     PRINT(QSYSPT)
A                                     INDARA
A                                     CA03(03 'FINAL DE TRABAJO')
A          R CABECERA
A                                     OVERLAY
A          2 4TIME
A          DSPATR(HI)
A          2 28'BÚSQUEDA DE CLIENTE POR CÓDIGO POSTAL
A          DSPATR(HI RI)
A          2 70DATE
A          EDTCDE(Y)
A          DSPATR(HI)
A          R PIE1
A          23 6'INTRO - Continuar'
A          DSPATR(HI)
A          23 29'F3 - Finalizar trabajo'
A          DSPATR(HI)
A          R PIE2
A          23 6'INTRO - Continuar'
A          DSPATR(HI)
A          23 29'F3 - Finalizar trabajo'
A          DSPATR(HI)
A          23 47'F4 - REARRANCAR CÓDIGO POSTAL'
A          DSPATR(HI)
A          R SOLICIT
A          OVERLAY
A          4 4'Entrar código postal'
A          DSPATR(HI)
A          CDP      R      Y      I      4 19DSPATR(CS)
A          CHECK(RZ)
A          61      ERRMSG('NO ENCONTRADO CÓD POSTAL' +
A          61)
A          R SUBARCH
A          NOMB      R      9 4
A          SALDO      R      9 27EDTCDE(J)
A          R SUBCTL
A          SFLCTL(SUBARCH)
A          55      SFLCLR
A          55      SFLCLR
A          N55      SFLDSPCTL
A          N55      SFLDSP
A          SFLSIZ(13)
A          SFLPAG(13)
A          ROLLUP(95 'GIRO ARRIBA')
A          OVERLAY
A          CA04(04 'REARRANCAR CÓDIGO POSTAL')
A          4 4'Código postal'
A          CDP      R      0 4 14DSPATR(HI)
A          7 4'Nombre cliente'
A          DSPATR(HI UL)
A          7 27'Saldo C/D'
A          DSPATR(HI UL)

```

Figura 200. DDS para el archivo de dispositivo de pantalla MENUBCOD

Búsqueda por código postal

Las DDS para el archivo del dispositivo de pantalla MENUBCOD contienen seis formatos de registro: CABECERA, PIE1, PIE2, SOLICIT, SUBARCH y SUBCTL.

El formato de registro SOLICIT solicita al usuario que entre un código postal. Si no se encuentra el código postal en el archivo, se visualiza un mensaje de error. El usuario puede pulsar la tecla F3, que activa el indicador 03, para finalizar el programa.

El formato del registro SUBARCH debe estar definido inmediatamente antes del formato de registro de control de subarchivo SUBCTL. El formato de registro de subarchivo, que está definido mediante la palabra clave SFL, describe cada campo del registro y especifica la ubicación donde va a aparecer el primer registro en la pantalla (aquí, en la línea 9).

El formato del registro de control del subarchivo contiene las siguientes palabras clave exclusivas:

- SFLCTL identifica este formato como formato de registro de control y da un nombre al formato de registro de subarchivo asociado.
- SFLCLR describe cuándo van a borrarse los registros existentes del subarchivo (cuando está activado el indicador 55). Se necesita esta palabra clave para pantallas adicionales.
- SFLDSPCTL indica cuándo va a visualizarse el formato del registro de control del subarchivo (cuando está desactivado el indicador 55).
- SFLDSP indica cuando va a visualizarse el subarchivo (cuando está desactivado el indicador 55).
- SFLSIZ especifica el tamaño total del subarchivo. En este ejemplo, el tamaño del subarchivo es de 13 registros que se visualizan en las líneas 9 a 21.
- SFLPAG define el número de registros de una página. En este ejemplo, el tamaño de la página es el mismo que el tamaño del subarchivo.
- ROLLUP indica que el indicador 95 se activa en el programa cuando se utiliza la función de giro hacia arriba.

La palabra clave OVERLAY define este formato de registro de control de subarchivo como un formato de recubrimiento. El formato de este registro puede escribirse sin que el sistema OS/400 borre antes la pantalla. La tecla F4 es válida para repetir la búsqueda con el mismo distrito postal. (Esta utilización de la tecla F4 permite una forma de giro hacia abajo).

Búsqueda por código postal

BUSCOD: Fuente RPG

```
//*****
//PROGRAMAS REL.: BUSCOD
// ARCHIVOS REL.: MAECLIL2 (ARCHIVO LÓGICO)
//                MENUBCOD (ARCHIVO WORKSTN)
// DESCRIPCIÓN: Este programa muestra un programa de búsqueda de maestro
//                de clientes que utiliza el proceso del subarchivo workstn.
//                Este programa solicita el código postal al usuario
//                y muestra los registros maestros de clientes por
//                código postal.
//                Puede utilizarse la tecla de firo arriba para observar
//                otra página. PF3 se utiliza para salir del programa.
//*****
Fmaec1il2 if e          k disk
Fmenubcod cf e          workstn sfile(subarch:recnum)
F                                indds(indicadores)

// Definiciones de campo:
D recnum          s          5p 0
D recordFound     s          n

D indicadores      ds
D  exitKey         n          overlay(indicadores:3)
D  restartKey      n          overlay(indicadores:4)
D  sflClear        n          overlay(indicadores:55)
D  zipNotFound     n          overlay(indicadores:61)
D  rollupKey       n          overlay(indicadores:95)

// Definiciones de lista de claves:
C  cstkey          klist
C  kfld            zip
```

Figura 201. Fuente para módulo BUSCOD (Pieza 1 de 3)


```
//*****
//  MAINLINE                                     *
//*****

/free

// Escribir menú inicial
write piel;
write cabecera;
exfmt solicit;

// bucles hasta pulsar PF03
dow not exitKey;
  setll cstkey cmlrec2;
  recordFound = %equal(maectl12);
  if recordFound;
    exsr ProcessSubfile;
  endif;

// Salir del bucle si se pulsó PF03 en la pantalla del subarchivo
if exitKey;
  leave;
endif;

// Si se ha pulsado PF04, repetir búsqueda con el mismo
// código postal.
if restartKey;
  iter;
endif;

// Solicitar código postal nuevo.
if not recordFound;
  // Si no encontrado un código postal, no volver a escribir cabecera
  // y pié
  write piel;
  write cabecera;
endif;
zipNotFound = not recordFound;
exfmt prompt;
enddo;

*inlr = *on;
```

Figura 201. Fuente para módulo BUSCOD (Pieza 2 de 3)

Búsqueda por código postal

```
//*****
//  SUBROUTINA - ProcessSubfile                                *
//  OBJETIVO   - Procesar subarchivo y visualizarlo           *
//*****
begsr ProcessSubfile;

    // Mantener bucles mientras se pulsa la tecla de giro arriba
    dou not rollupKey;
    // ¿Existe más información que podamos añadir al subarchivo?
    if not %eof(maeklil2);
        // Borrar y rellenar subarchivo con datos de cliente
        exsr ClearSubfile;
        exsr FillSubfile;
    endif;

    // Escribir subarchivo y esperar respuesta
    write pie2;
    exfmt subctl;
enddo;

endsr; // final de subrutina ProcessSubfile


//*****
//  SUBROUTINA - FillSubfile                                    *
//  OBJETIVO   - Rellenar subarchivo con código postal especificado *
//              coincidente en registros de cliente.             *
//*****
begsr FillSubfile;

    // Bucle por los registros de clientes con código postal especificado
    recnum = 0;
    dou %eof(menubcod);
        // Leer registro siguiente con código postal especificado
        reade zip cmlrec2;
        if %eof(maeklil2);
            // Si no hay más registros, esto es todo
            leavesr;
        endif;

        // Añadir información sobre este registro al subarchivo
        recnum = recnum + 1;
        write subarch;
    enddo;
endsr; // final de subrutina FillSubfile;


//*****
//  SUBROUTINA - ClearSubfile                                    *
//  OBJETIVO   - Borrar registros de subarchivo                *
//*****
begsr ClearSubfile;

    sflClear = *on;
    write subctl;
    sflClear = *off;

endsr; // final de subrutina ClearSubfile

/end-free
```

Figura 201. Fuente para módulo BUSCOD (Pieza 3 de 3)

Las especificaciones de descripción de archivos identifican el archivo de disco que va buscarse y el archivo del dispositivo de pantalla que va a utilizarse (MENUBCOD). La palabra clave SFILE para el archivo WORKSTN identifica el formato de registro (SUBARCH) que va a utilizarse como subarchivo. El campo de número relativo de registro (RECNUM) especificado controla el registro al que se accede dentro del subarchivo.

El programa visualiza el formato de registro SOLICIT y espera la respuesta del usuario de la estación de trabajo. La tecla F3 activa el indicador 03, que controla el final del programa. El código postal (CDP) se utiliza para situar el archivo MAECLIL2 mediante la operación SETLL. Tenga en cuenta que el nombre del formato de registro CMLREC2 se utiliza en la operación SETLL en lugar del nombre del archivo MAECLIL2. Si no se encuentra ningún registro, se visualiza un mensaje de error.

La subrutina SFLPRC maneja el proceso para el subarchivo: lo borra, llena y visualiza. El subarchivo está preparado para solicitudes adicionales en la subrutina SFCLR. Si está activado el indicador 55, no se produce ninguna acción en la pantalla, pero se borra el área del almacenamiento principal para los registros del subarchivo. La rutina SFLFIL rellena el subarchivo con registros. Se lee un archivo del registro MAECLIL2. Si el código postal es el mismo, se incrementa el recuento de registros (RECNUM) y se graba el registro en el subarchivo. Esta subrutina se repite hasta que esté relleno el subarchivo (indicador 21 en la operación WRITE) o se produzca un final de archivo en el archivo MAECLIL2 (indicador 71 en la operación READE). Cuando se rellena el subarchivo o se produce un fin de archivo, la operación EXFMT graba el subarchivo en la pantalla mediante el formato de control registro del subarchivo. El usuario revisa la pantalla y decide:

- Finalizar el programa pulsando la tecla F3.
- Volver a arrancar el distrito postal pulsando la tecla F4. No se visualiza el formato de registro SOLICIT y se visualiza el subarchivo comenzando por el mismo código postal.
- Rellenar otra página pulsando las teclas de GIRO ARRIBA. Si se produce un final de archivo en el archivo MAECLIL2, vuelve a visualizarse la página actual; de lo contrario, se borra el subarchivo y se visualiza la página siguiente.
- Continuar con otro código postal, pulsando la tecla INTRO. Se visualiza el formato de registro SOLICIT. El usuario puede entrar un código postal o finalizar el programa.

En la Figura 202 en la página 410, el usuario entra un código postal como respuesta a la solicitud.

Búsqueda por código postal

22:34:38

BÚSQUEDA DE CLIENTE POR CÔD. POSTAL

9/30/94

Entre Código Postal 11201

INTRO - Continuar

F3 - Fin de trabajo

Figura 202. Pantalla de solicitud 'Búsqueda de Cliente por Código Postal'

El subarchivo se escribe en la pantalla tal como se muestra en la Figura 203 .

22:34:45

BÚSQUEDA DE CLIENTE POR CÓDIGO POSTAL

9/30/94

Código postal 11201

<u>Nombre de cliente</u>	<u>Saldo C/D</u>
Jorge Sanz	300.000
Miguel García	150.000
Alicia Coll	5.000

INTRO - Continuar

F3 - Fin de trabajo

F4 - REARRANCAR CÓDIGO POSTAL

Figura 203. Pantalla 'Búsqueda Cliente por Código Postal'

Búsqueda y consulta por nombre

A continuación se muestra el proceso del subarchivo WORKSTN (pantalla con selección). Los subarchivos se utilizan para visualizar todos los registros coincidentes para un nombre de cliente determinado, después de lo cual al usuario se le permite efectuar una selección del subarchivo, como como visualizar información adicional del cliente.

MECLIL3: DDS para un archivo lógico

```

A*****
A*NOMBRE ARCHIVO:  MAECLIL3                      *
A*PROGRAMAS REL.:  NOMBUS                        *
A* ARCHIVOS REL.:  MAECLIE                        *
A*  DESCRIPCIÓN:  ESTE ES EL ARCHIVO LÓGICO MAECLIL3. TIENE *
A*                  UN FORMATO DE REGISTRO LLAMADO REGCLI.  *
A*                  VISTA LÓGICA DE ARCHIVO MAESTRO DE CLIENTES *
A*                  (MAECLI) POR NOMBRE (NOMB)              *
A*****
A          R REGCLI                                PFILE(MAECLI)
A          K NOMB
A*
A*****
A* NOTA: PUESTO QUE EL FORMATO DE REGISTRO DEL ARCHIVO FÍSICO (MAECLI) *
A*      TIENE EL MISMO NOMBRE DE FORMATO DE REGISTRO, NO ES NECESARIO *
A*      UN LISTADO DE CAMPOS EN ESTE ARCHIVO DDS.              *
A*****

```

Figura 204. DDS para el archivo lógico MAECLIL3

Las DDS para el archivo de base de datos utilizado en este programa definen un formato de registro que lleva por nombre REGCLI e identifican el campo NOMB como campo de clave.

Búsqueda y consulta por nombre

MENUBNOM: DDS para un archivo de dispositivo de pantalla

```
A*****
A*NOMBRE ARCHIVO:  MENUBNOM3
A*PROGRAMAS REL.:  NOMBUS
A* ARCHIVOS REL.:  MAECLIL3  (ARCHIVO LÓGICO)
A*  DESCRIPCIÓN:  ESTE EL EL ARCHIVO DE PANTALLA MENUBNOM. TIENE 7
A*                  FORMATOS DE REGISTRO.
A*****
A                  REF(MAECLIL3)
A                  CHGINPDFT(CS)
A                  PRINT(QSYSPRT)
A                  INDARA
A                  CA03(03 'FINAL DE TRABAJO')
A      R HEAD
A                  OVERLAY
A                  2  4TIME
A                  DSPATR(HI)
A                  2  25'BÚSQUEDA DE CLIENTE Y CONSULTA POR NOMBRE'
A                  DSPATR(HI UL)
A                  2  70DATE
A                  EDTCDE(Y)
A                  DSPATR(HI)
A      R PIE1
A                  23  6'INTRO - Continuar'
A                  DSPATR(HI)
A                  23  29'F3 - Finalizar trabajo'
A                  DSPATR(HI)
A      R PIE2
A                  23  6'INTRO - Continuar'
A                  DSPATR(HI)
A                  23  29'F3 - Finalizar trabajo'
A                  DSPATR(HI)
A                  23  47'F4 - Rearrancar nombre'
A                  DSPATR(HI)
A      R SOLICIT
A                  OVERLAY
A                  5  4'Entrar nombre de búsqueda'
```

Figura 205. DDS para el archivo de dispositivo de pantalla MENUBNOM (Pieza 1 de 2)

Búsqueda y consulta por nombre

A									DSPATR(HI)
A		SRCNAM	R		I	5	23	REFFLD(NAME MAECLIL3)	
A								DSPATR(CS)	
A		R SUBARCH						SFL	
A								CHANGE(99 'CAMPO MODIFICADO')	
A		SEL		1A	B	9	8	DSPATR(CS)	
A								VALUES(' ' 'X')	
A		CDP	R			0	9	54	
A		CLIE	R			0	9	43	
A		NOMB	R			0	9	17	
A		R SUBCTL						SFLCTL(SUBARCH)	
A								SFLSIZ(0013)	
A								SFLPAG(0013)	
A	55							SFLCLR	
A	N55							SFLDSPCTL	
A	N55							SFLDSP	
A								ROLLUP(95 'GIRO ARRIBA')	
A								OVERLAY	
A								CF04(04 'REARRANCAR NOMBRE BÚSQUEDA')	
A						5	4	'Nombre de búsqueda'	
A		SRCNAM	R			0	5	17	REFFLD(NAME MAECLIL3)
A								DSPATR(HI)	
A						7	6	'Seleccionar'	
A								DSPATR(HI)	
A						8	6	'X'	Nombre cliente '
A								DSPATR(HI)	
A								DSPATR(UL)	
A						8	42	'Número	Código postal '
A								DSPATR(HI)	
A								DSPATR(UL)	
A		R CUSDSP							
A								OVERLAY	
A						6	25	'Cliente'	
A		CLIE		5S	00	6	35	DSPATR(HI)	
A						8	25	'Nombre'	
A		NOMB		20A	0	8	35	DSPATR(HI)	
A						10	25	'Dirección'	
A		DIRE1		20A	0	10	35	DSPATR(HI)	
A		DIRE2		20A	0	11	35	DSPATR(HI)	
A						13	25	'Ciudad'	
A		CDAD		20A	0	13	35	DSPATR(HI)	
A						15	25	'Provincia'	
A		PROV		2A	0	15	35	DSPATR(HI)	
A						15	41	'Código postal'	
A		CDP		5S	00	15	50	DSPATR(HI)	
A						17	25	'Saldo C/D'	
A		SALDO		10Y	20	17	42	DSPATR(HI)	
A								EDTCDE(J)	

Figura 205. DDS para el archivo de dispositivo de pantalla MENUBNOM (Pieza 2 de 2)

Las DDS para el archivo del dispositivo de pantalla MENUBNOM contienen siete formatos de registro: CABECERA, PIE1, PIE2, SOLICIT, SUBARCH, SUBCTL y CUSDSP.

El formato de registro SOLICIT solicita al usuario que entre un código postal y un nombre de búsqueda. Si no se realiza ninguna entrada, la visualización comienza al principio del archivo. El usuario puede pulsar la tecla F3, que activa el indicador 03, para finalizar el programa.

El formato del registro SUBARCH debe estar definido inmediatamente antes del formato de registro de control de subarchivo SUBCTL. El formato de registro de

Búsqueda y consulta por nombre

subarchivo definido con la palabra clave SFL describe cada campo del registro y especifica la ubicación de la pantalla donde aparecerá el primer registro (aquí en la línea 9).

El formato del registro de control del subarchivo SUBCTL contiene las siguientes palabras clave exclusivas:

- SFLCTL identifica este formato como formato de registro de control y da un nombre al formato de registro de subarchivo asociado.
- SFLCLR describe cuándo van a borrarse los registros existentes del subarchivo (cuando está activado el indicador 55). Se necesita esta palabra clave para pantallas adicionales.
- SFLDSPCTL indica cuándo va a visualizarse el formato del registro de control del subarchivo (cuando está desactivado el indicador 55).
- SFLDSP indica cuando va a visualizarse el subarchivo (cuando está desactivado el indicador 55).
- SFLSIZ especifica el tamaño total del subarchivo. En este ejemplo, el tamaño del subarchivo es de 13 registros que se visualizan en las líneas 9 a 21.
- SFLPAG define el número de registros de una página. En este ejemplo, el tamaño de la página es el mismo que el tamaño del subarchivo.
- ROLLUP indica que el indicador 95 se activa en el programa cuando se utiliza la función de giro hacia arriba.

La palabra clave OVERLAY define este formato de registro de control de subarchivo como un formato de recubrimiento. El formato de este registro puede escribirse sin que el sistema OS/400 borre antes la pantalla. La tecla F4 es válida para repetir la búsqueda del mismo nombre. (Esta utilización de la tecla F4 permite una forma de giro hacia abajo).

El formato de registro CUSDSP visualiza la información de los clientes seleccionados.

NOMBUS: Fuente RPG

```

//*****
// NOMBRE Progr.: NOMBUS
// ARCHIVOS REL.: MAECLIL3 (ARCHIVO LÓGICO)
//                MENUBNOM (ARCHIVO WORKSTN)
// DESCRIPCIÓN: Este programa muestra un programa de búsqueda de maestro
//                de clientes que utiliza el proceso del subarchivo workstn.
//                Este programa solicita al usuario el nombre del
//                cliente y lo utiliza para situar el archivo maeclil3
//                mediante la operación setll. A continuación, visualiza los
//                registros mediante subarchivos.
//                Para rellenar otra página, pulse la tecla de giro arriba
//                Para visualizar los detalles del cliente, entre 'X' junto
//                a dicho cliente y pulse Intro.
//                Para salir del programa, pulse PF3.
//*****

Fmaeclil3 if e          k disk
Fmenubnom cf e          workstn sfile(subarchivo:recnum)
F                                     indds(indicadores)

// Definiciones de campo:
D recnum          s          5p 0

D indicadores      ds
D  exitKey          n      overlay(indicadores:3)
D  restartKey       n      overlay(indicadores:4)
D  sflClear         n      overlay(indicadores:55)
D  rollupKey        n      overlay(indicadores:95)

// Definiciones de lista de claves:
C  cstkey           klist
C                   kfld          srcnam
C  zipkey           klist
C                   kfld          name

```

Figura 206. Fuente para el módulo NOMBUS (Pieza 1 de 4)

Búsqueda y consulta por nombre

```
//*****
//  MAINLINE                                     *
//*****

/free

write piel;
write cabecera;
exfmt solicit;

// bucle hasta que se pulse la tecla de salir
dow not exitKey;
  setll cstkey regcli;
  exsr ProcessSubfile;
  exsr DisplayCustomerDetail;

// Salir del bucle si se pulsa tecla salir en pantalla de subarchivo
if exitKey;
  leave;
endif;

// Repetir bucle si se pulsa tecla rearmar en pantalla de subarchivo
if restartKey;
  iter;
endif;

write piel;
write cabecera;
exfmt solicit;

enddo;

*inlr = *on;

//*****
//  SUBROUTINA - ProcessSubfile                     *
//  OBJETIVO   - Procesar subarchivo y visualizar   *
//*****
begsr ProcessSubfile;

// Mantener bucles mientras se pulsa la tecla de giro arriba
dou not rollupKey;
  // ¿Existe más información que podamos añadir al subarchivo?
  if not %eof(maectl13);
    // Borrar y rellenar subarchivo con datos de cliente
    exsr ClearSubfile;
    exsr FillSubfile;
  endif;

  // Escribir subarchivo y esperar respuesta
  write pie2;
  exfmt subctl;
enddo;

endsr; // final de subrutina ProcessSubfile
```

Figura 206. Fuente para el módulo NOMBUS (Pieza 2 de 4)

```

//*****
//  SUBROUTINA - FillSubfile                                *
//  OBJETIVO   - Rellenar subarchivo                        *
//*****
begsr FillSubfile;

    // Bucle por los registros de clientes con código postal especificado
    numreg = 0;
    dou %eof(menubnom);
        // Leer registro siguiente con código postal especificado
        read regclie;
        if %eof(maectl13);
            // Si no hay más registros, esto es todo
            leavesr;
        endif;

        // Añadir información sobre este registro al subarchivo
        numreg = numreg + 1;
        sel = *blanco;
        write subarchivo;
    enddo;

endsr; // final de subrutina FillSubfile;

//*****
//  SUBROUTINA - ClearSubfile                                *
//  OBJETIVO   - Borrar registros de subarchivo            *
//*****
begsr ClearSubfile;

    sflClear = *on;
    write subctl;
    sflClear = *off;

endsr; // final de subrutina ClearSubfile

```

Figura 206. Fuente para el módulo NOMBUS (Pieza 3 de 4)

Búsqueda y consulta por nombre

```
//*****  
// SUBROUTINA - DisplayCustomerDetail *  
// OBJETIVO - Visualizar registros de cliente seleccionados *  
//*****  
begsr DisplayCustomerDetail;  
  
// Bucle por todo el registro modificado en subarchivo  
readc subarch;  
dow not %eof(menubnom);  
    // Rearrancar la visualización de registros de cliente solicitados  
    restartKey = *on;  
    // Buscar registro de clientes y visualizarlo  
    chain zipkey regclie;  
    exfmt cusdsp;  
  
    // Salir de bucle si se pulsa la tecla de salir  
    if exitKey;  
        leave;  
    endif;  
  
    readc subarch;  
enddo;  
  
endsr; // fin de subrutina ChangeSubfile  
  
/end-free
```

Figura 206. Fuente para el módulo NOMBUS (Pieza 4 de 4)

Las especificaciones de descripción de archivos identifican el archivo de disco que va buscarse y el archivo del dispositivo de pantalla que va a utilizarse (MENUBNOM). La palabra clave SFILE para el archivo WORKSTN identifica el formato de registro (SUBARCH) que va a utilizarse como subarchivo. El campo de número relativo de registro (RECNUM) especifica el registro al que se accede dentro del subarchivo.

El programa visualiza el formato de registro SOLICIT y espera la respuesta del usuario de la estación de trabajo. La tecla F3 activa el indicador 03, que controla el final del programa. El nombre (NOMB) se utiliza como la clave para que la operación SETLL sitúe en el archivo MAECLIL3. Tenga en cuenta que el nombre del formato de registro REGCLIE se utiliza en la operación SETLL en lugar del nombre del archivo MAECLIL3.

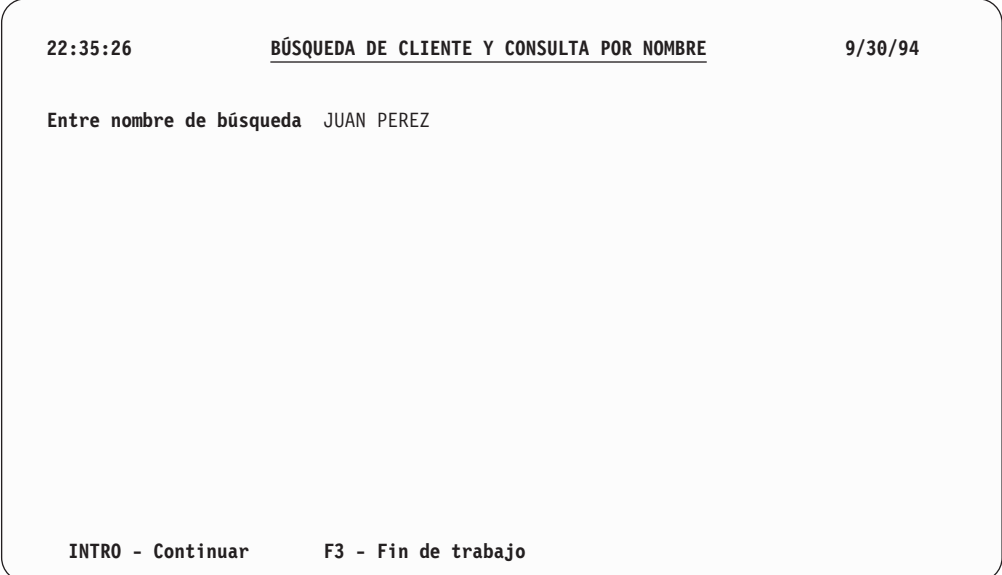
La subrutina SFLPRC maneja el proceso para el subarchivo: lo borra, llena y visualiza. El subarchivo está preparado para solicitudes adicionales en la subrutina SFCLR. Si está activado el indicador 55, no se produce ninguna acción en la pantalla, pero se borra el área del almacenamiento principal para los registros del subarchivo. La rutina SFLFIL rellena el subarchivo con registros. Se lee un registro del archivo MAECLIL3, se incrementa el recuento de registros (RECNUM) y se graba el registro en el subarchivo. Esta subrutina se repite hasta que esté lleno el subarchivo (indicador 21 en la operación WRITE) o se produzca un final de archivo en el archivo MAECLIL3 (indicador 71 en la operación READE). Cuando se rellena el subarchivo o se produce un fin de archivo, la operación EXFMT graba el subarchivo en la pantalla mediante el formato de control registro del subarchivo. El usuario revisa la pantalla y decide:

- Finalizar el programa pulsando la tecla F3.
- Rearrancar el subarchivo pulsando la tecla F4. No se visualiza el formato de registro SOLICIT y se visualiza el subarchivo comenzando por el mismo nombre.

Búsqueda y consulta por nombre

- Rellenar otra página pulsando las teclas de GIRO ARRIBA. Si se produce un final de archivo en el archivo MAECLIL3, vuelve a visualizarse la página actual; de lo contrario, se borra el subarchivo y se visualiza la página siguiente.
- Visualizar los detalles del cliente entrando X y pulsando INTRO. El usuario puede volver a continuación a la pantalla SOLICIT pulsando INTRO, volver a visualizar el subarchivo pulsando la tecla F4 o finalizar el programa pulsando la tecla F3.

En la Figura 207, el usuario responde a la solicitud inicial entrando un nombre de cliente.



22:35:26 BÚSQUEDA DE CLIENTE Y CONSULTA POR NOMBRE 9/30/94

Entre nombre de búsqueda JUAN PEREZ

INTRO - Continuar F3 - Fin de trabajo

Figura 207. Pantalla de solicitud 'Búsqueda y Consulta de Cliente por Nombre'

El usuario solicita más información entrando una X tal como se muestra en la Figura 208 en la página 420.

Búsqueda y consulta por nombre

22:35:43	<u>BÚSQUEDA DE CLIENTE Y CONSULTA POR NOMBRE</u>		9/30/94
Nombre de búsqueda JUAN PEREZ			
Seleccionar			
"X"	Nombre de cliente	Número	Código postal
X	JUAN PEREZ	00012	08019
	JUAN PEREZ	00209	08016
INTRO - Continuar F3 - Fin de trabajo F4 - Rearrancar nombre			

Figura 208. Pantalla de información 'Búsqueda y Consulta de Cliente por Nombre'

En la Figura 209 aparece la información detallada para el cliente seleccionado. En este punto el usuario selecciona la tecla de función adecuada para continuar o finalizar la consulta.

23:39:48	<u>BÚSQUEDA DE CLIENTE Y CONSULTA POR NOMBRE</u>		9/30/94
Cliente 00012			
Nombre JUAN PEREZ			
Dirección AVDA. DIAGONAL 201			
Ciudad BARCELONA			
Provincia BCN		Código postal 08028	
Saldo C/D		.00	
INTRO - Continuar F3 - Fin de trabajo F4 - Rearrancar nombre			

Figura 209. Pantalla de información detallada 'Búsqueda y Consulta de Cliente por Nombre'

Parte 5. Apéndices

Apéndice A. Diferencias de comportamiento entre OPM RPG/400 y ILE RPG para AS/400

Las listas que aparecen a continuación muestran las diferencias de funcionamiento entre el compilador OPM RPG/400 y ILE RPG.

Compilación

1. Si se especifica CVTOPT(*NONE) en OPM RPG, todos los campos descritos externamente que sean de un tipo al que RPG no dé soporte, o que tengan atributos que RPG no soporte, serán ignorados. Si se especifica CVTOPT(*NONE) en ILE RPG, todos los campos descritos externamente se incorporarán al programa con el mismo tipo que se especificó en la descripción externa.
2. En RPG IV no existe ninguna dependencia entre DATEDIT y DECEDIT en la especificación de control.
3. En relación a los mandatos de crear de ILE RPG (CRTBNDRPG y CRTRPGMOD):
 - El parámetro IGNDECERR del mandato CRTRPGPGM se ha sustituido por el parámetro FIXNBR en los mandatos de crear de ILE RPG. IGNDECDDTA ignora cualquier error en los datos decimales y continúa con la siguiente instrucción de la máquina. En algunos casos, esto puede provocar que los campos se actualicen con valores incorrectos y a veces imprevisibles. FIXNBR corrige los datos de una manera predecible antes de que se utilicen.
 - Existe un parámetro nuevo, TRUNCNBR, para controlar si se permite el desbordamiento numérico.
 - No existen características o mandatos de informe automático en RPG IV.
 - No es posible solicitar del compilador un listado MI.
4. En un listado de compilador, los número de línea comienzan por 1 y aumentan en 1 por cada línea de origen o de especificaciones generadas, cuando se especifica el valor por omisión OPTION(*NOSRCSTMT). Si se especifica OPTION(*SRCSTMT), se imprimirán los números de secuencia en vez de los números de línea. Los ID de fuente son numéricos, es decir, ya no existen más números de línea AA000100 para los miembros /COPY o las DDS ampliadas.
5. RPG IV precisa que todas las directivas del compilador aparezcan *antes* que los datos de tiempo de compilación, incluida /TITLE. Cuando RPG IV encuentra una directiva /TITLE, la tratará como si fueran datos. (El RPG III trata las especificaciones /TITLE como directivas del compilador en cualquier punto del fuente.)

La Ayuda para la conversión eliminará las especificaciones /TITLE que encuentre en los datos de tiempo de compilación.
6. ILE RPG es más riguroso al detectar solapamientos de campos en las estructuras de datos. Para algunas operaciones de cálculo con operandos solapados, ILE RPG emite un mensaje mientras que el compilador OPM no.
7. En ILE RPG no puede utilizarse la palabra NOT como un nombre de variable. NOT es una palabra especial que se utiliza como un operador en expresiones.
8. Durante el tiempo de compilación, la fuente se lee utilizando el CCSID del archivo fuente principal, mientras que para OPM RPG la fuente se lee utilizando el CCSID del trabajo.

Ejecución

1. RPG IV no da soporte a la operación FREE.
2. Puede que aparezcan ciertos mensajes MCH en las anotaciones de trabajo que no aparecen bajo OPM (por ejemplo, MCH1202). La aparición de estos mensajes no indica un cambio en el funcionamiento del programa.
3. Si utiliza la API QMHSNDPM para enviar mensajes desde el programa, puede que necesite añadir 1 al parámetro de desplazamiento de la pila para permitir la presencia del procedimiento de entradas al programa en la pila. Esto sucederá únicamente si el procedimiento ILE es el procedimiento de entrada de usuario, y si se ha utilizado el valor especial '*' para la cola de mensajes de llamada y un valor mayor que 0 para el desplazamiento de la pila.
4. ILE RPG no interpreta los códigos de retorno distintos a 0 o 1 para las llamadas a programas o procedimientos que finalizan sin una excepción.
5. Cuando el manejador para cancelar de un programa ILE RPG recibe control, establecerá en 2 el código de retorno del sistema. El manejador para cancelar de un programa OPM RPG no modifica el valor del código de retorno del sistema.
6. Cuando se detecta una recurrencia, el OPM de RPG/400 visualiza el mensaje de consulta RPG8888. El ILE RPG indica el mensaje de escape RNX8888; no se visualiza ningún mensaje de consulta para esta condición. Tenga en cuenta que esto únicamente se aplica a procedimientos principales. Se permite la recurrencia para subprocedimientos.
7. Si se producen errores de datos decimales durante la inicialización de un subcampo decimal con zona o de un subcampo decimal empaquetado, es posible que los valores de restablecimiento (los valores que se utilizan para restaurar el subcampo con la operación RESET) no sean válidos. Por ejemplo, puede que no se haya inicializado el subcampo o que lo haya recubierto otro subcampo inicializado de un tipo diferente. Si se intenta una operación RESET para dicho subcampo, entonces se producirá un error de datos decimales en OPM RPG/400. Sin embargo, se completará satisfactoriamente una operación RESET en el mismo subcampo de ILE RPG; después de la operación RESET, el subcampo tendrá el mismo valor no válido. Como resultado, los intentos de utilizar el valor obtendrán un error de datos decimales.
8. En ILE RPG, las posiciones 254-263 de la estructura de datos del estado del programa (PSDS) contienen el nombre de usuario del trabajo emisor. En OPM RPG, estas posiciones reflejan el perfil de usuario actual. El perfil de usuario actual en ILE RPG se encuentra en las posiciones 358-367.

Depuración y manejo de excepciones

1. La operación DEBUG no está soportada en RPG IV.
2. Cuando se utiliza el depurador de fuentes de ILE, no se pueden utilizar identificadores RPG, nombres de subrutina o puntos en el ciclo, tales como *GETIN y *DETC, para establecer puntos de interrupción.
3. Tanto RPG como ILE RPG normalmente dejan los errores de función en las anotaciones de trabajo. Sin embargo, en ILE RPG, si se ha codificado un indicador de error, un extensor 'E' o una rutina de error *PSSR, no aparecerá el error de función.

Es aconsejable que elimine los códigos que supriman los errores de función, ya que la presencia del indicador, el extensor 'E', o *PSSR impedirán que se produzcan errores de función.

4. El rendimiento de la llamada de activación de LR se mejorará enormemente si no existe ninguna PSDS, o una PSDS que no supere los 80 bytes, ya que parte

Diferencias entre OPM RPG/400 e ILE RPG

de la información para completar la PSDS pasados los 80 bytes es difícil de obtener. Si no se codifica la PSDS, o es demasiado corta para contener la fecha y la hora en que se inició el programa, estos dos valores no estarán disponibles en un vuelco formateado. Todos los demás valores de la PSDS estarán disponibles, sin importar la longitud de la PSDS.

5. El prefijo para los mensajes de consulta de ILE RPG es RNQ, así que si utiliza la lista de respuesta por omisión deberá añadir entradas de RNQ que sean similares a las entradas de RPG.
6. En OPM, si un programa CL llama a un programa RPG seguido de MONMSG, y el programa RPG recibe una notificación o mensaje acerca del estado, el CL MONMSG no manejará la notificación o mensaje acerca del estado. Si se está intentando llamar a ILE RPG desde ILE CL y ambos se encuentran en el mismo grupo de activación, ILE CL MONMSG manejará la notificación o mensaje acerca del estado y el procedimiento RPG se parará inmediatamente sin que se emita un mensaje de error RPG. Para obtener más información consulte el apartado "Problemas cuando ILE CL supervisa los mensajes de estado y notificación" en la página 285.
7. Cuando se visualiza una variable utilizando el depurador de fuentes de ILE, se obtendrán resultados poco fidedignos si:
 - el programa ILE RPG utiliza un archivo descrito externamente y
 - la variable está definida en el archivo de base de datos pero no se hace referencia a ella en el programa ILE RPG.
8. Si su programa RPG III tiene un problema de discrepancia de parámetros (por ejemplo, pasa un parámetro de longitud 10 a un programa que espera un parámetro de longitud 20, y el programa llamado cambia los 20 bytes), su programa experimentará un problema de corrupción de almacenamiento. Este problema no siempre puede provocar un error si el almacenamiento que se ha dañado no es importante para la ejecución del programa.
Cuando este programa se convierte a RPG IV, el diseño del almacenamiento puede ser diferente para que el programa utilice el almacenamiento dañado. Esto puede provocar una excepción inesperada, por ejemplo la excepción MCH3601 en una operación de archivo como es SETLL. Si se producen errores extraños que parecen no estar relacionados con la aplicación, deberá comprobar los parámetros de todas las operaciones de llamada para asegurarse de que todos los parámetros tienen la longitud correcta.

E/S

1. En ILE RPG se puede leer un registro en un archivo abierto para actualizar y que se ha creado o alterado temporalmente con SHARE(*YES), y a continuación actualizar este registro bloqueado en otro programa que haya abierto el mismo archivo para actualizar.
2. No se puede modificar el indicador MR con las operaciones MOVE o SETON. (RPG III sólo impide la utilización de SETON con MR.)
3. La entrada de Tipo de archivo de las especificaciones de archivo ya no condiciona el tipo de operaciones de E/S que deben estar presentes en las especificaciones de cálculo.
Por ejemplo, en RPG III, si define un archivo como un archivo de actualización, entonces debe haber una operación UPDAT en el programa. Este ya no es el caso en RPG IV. Sin embargo, la definición de archivos todavía debe ser coherente con las operaciones de E/S presentes en el programa. Por lo tanto, si tiene una operación UPDATE en el fuente, el archivo deberá estar definido como un archivo de actualización.

Diferencias entre OPM RPG/400 e ILE RPG

4. ILE RPG permite la agrupación por bloques de registros incluso cuando se especifica la palabra clave COMMIT en la especificación de descripción del archivo.
5. En RPG IV, un archivo abierto para actualización se abrirá también con posibilidad de supresión. No es precisa ninguna operación DELETE para que tenga posibilidad de supresión.
6. En RPG IV, no es necesario codificar un número real para el número de dispositivos que utilizará un archivo de múltiples dispositivos. Si especifica MAXDEV(*FILE) en una especificación de descripción de archivos, entonces el número de las áreas de salvar que se creen para SAVDS y SAVEIND se basa en el número de dispositivos que puede manejar el archivo. (Las palabras clave SAVDS, SAVEIND, y MAXDEV en la especificación de una descripción de archivo RPG IV corresponden a las opciones SAVDS, IND, y NUM en la línea de continuación de la especificación de una descripción de archivo RPG III, respectivamente).

En ILE RPG, el número total de dispositivos de programa que el programa puede adquirir tiene que ser igual al número máximo de dispositivos definido en el archivo de dispositivo. OPM RPG/400 lo permite mediante la opción NUM.

7. En ILE RPG, los códigos de operación ACQ y REL pueden utilizarse con archivos de dispositivo únicos.
8. En ILE RPG, el número relativo de registro y los campos de clave de la sección de información de retorno específica de base de datos de INFDS se actualizan en cada operación de entrada al realizar lecturas por bloques.
9. Cuando se produce un error de restricción referencial en OPM RPG/400, el código de estado se establece en "01299" (error de E/S). En ILE RPG, el código de estado se establece en "01022", "01222", o "01299", dependiendo del tipo de error de restricción referencial que se produzca.
 - Si la gestión de datos no puede asignar un registro debido a un error de restricción referencial, se emitirá un mensaje de notificación CPF502E. ILE RPG establecerá el código de estado en "01222" y OPM RPG/400 lo establecerá en "01299".

Si no aparece un indicador de error, un extensor 'E', o la subrutina de error INFSR, ILE RPG emitirá el mensaje de consulta RNQ1222 y OPM RPG/400 emitirá el mensaje de consulta RPG1299. La diferencia principal entre estos dos mensajes es que RNQ1222 permite reintentar la operación.
 - Si la gestión de datos detecta un error de restricción referencial que le ha hecho emitir un mensaje de notificación CPF503A, CPF502D, o CPF502F, ILE RPG establecerá el código de estado en "01022" y OPM RPG/400 lo establecerá en "01299".

Si no aparece un indicador de error, un extensor 'E', o una subrutina de error INFSR, ILE RPG emitirá el mensaje de consulta RNQ1022 y OPM RPG emitirá el mensaje de consulta RPG1299.
 - Todos los errores de restricción referencial detectados por la gestión de datos que motiven que la gestión de datos emita un mensaje de escape serán la causa de que tanto OPM RPG como ILE RPG establezcan el código de estado en "01299".
10. En ILE RPG, la sección de información de retorno específica de base de datos de INFDS se actualiza independientemente del resultado de la operación de E/S. En OPM RPG/400, esta sección de información de retorno no se actualiza si se produce la condición de registro no encontrado.
11. ILE RPG se basa más que OPM RPG/400 en el manejo de errores de la gestión de datos. Esto significa que en algunos casos aparecerán ciertos mensajes de

Diferencias entre OPM RPG/400 e ILE RPG

error en las anotaciones de trabajo de un programa ILE RPG, pero no en un programa OPM RPG/400. Algunas de las diferencias que observará en el manejo de errores son:

- Al realizar una operación UPDATE en un registro de un archivo de base de datos que no ha sido bloqueado por una operación de entrada anterior, tanto ILE RPG como OPM RPG/400 establecen el código de estado en "01211". ILE RPG detecta esta situación cuando la gestión de datos emite un mensaje de notificación CPF501B y lo coloca en las anotaciones de trabajo.
 - Al manejar archivos WORKSTN e intentar realizar una operación de E/S en un dispositivo que no ha sido adquirido ni definido, tanto ILE como OPM RPG establecerán el estado en "01281". ILE RPG detecta esta situación cuando la gestión de datos emite un mensaje de escape CPF5068 y lo coloca en las anotaciones de trabajo.
12. Al realizar una operación READE, REDPE (READPE en ILE), SETLL en un archivo de base de datos, o al realizar procesos secuenciales entre límites mediante un archivo de direcciones de registros, el OPM RPG/400 efectúa comparaciones de clave utilizando el orden de clasificación *HEX. Esto puede dar resultados diferentes de los que se esperaban al utilizar características DDS que hacen que más de un argumento de búsqueda coincida con una clave dada en el archivo.
- Por ejemplo, si se utiliza ABSVAL en una clave numérica, tanto -1 como 1 serían satisfactorios como argumentos de búsqueda para una clave del archivo con el valor 1. Mediante la orden de clasificación hexadecimal, el argumento de búsqueda -1 no será satisfactorio para la clave real 1.
- ILE RPG realiza comparaciones de clave utilizando la orden de clasificación *HEX sólo para los archivos DDM anteriores a V3R1. Consulte la publicación "Utilización de archivos DDM anteriores a la Versión 3, Release 1" en la página 353 para obtener más información.
13. ILE RPG permite que el archivo destino y el archivo origen, especificado para las matrices antes de su ejecución y para las tablas, sea diferente. En OPM RPG, ambos nombres de archivo deben ser iguales; si son diferentes se emitirá el mensaje de diagnóstico QRG3038.
14. Cuando se especifica la conversión de un archivo controlado por RAF, los resultados al utilizar ILE RPG pueden ser distintos a los de OPM RPG/400, dependiendo de la tabla de conversión. Esto es debido a que la secuencia de operaciones es diferente. En OPM RPG/400 la secuencia es: recuperar registro, convertir y comparar; en ILE RPG la secuencia es: convertir, comparar y recuperar registro.

Datos DBCS en campos de tipo carácter

1. En OPM RPG/400, la posición 57 (Comprobación de transparencia) de la especificación de control le permite especificar si el compilador RPG/400 debe buscar los caracteres DBCS, los literales de caracteres y las constantes. Si especifica que el compilador debe explorar los literales transparentes y si un literal de tipo carácter que empieza por un apóstrofo seguido por un desplazamiento a teclado ideográfico presenta anomalías en la comprobación de transparencia, el literal se vuelve a analizar como un literal que no es transparente.

En ILE RPG, no existe ninguna opción en la especificación de control para especificar si el compilador debe realizar la comprobación de transparencia en literales de tipo carácter. Si un literal de tipo carácter contiene un carácter de control de desplazamiento a teclado ideográfico, con independencia de la posición del carácter de desplazamiento a teclado ideográfico dentro del literal

Diferencias entre OPM RPG/400 e ILE RPG

de tipo carácter, el carácter de desplazamiento a teclado ideográfico indica el comienzo de los datos DBCS. El compilador comprobará lo siguiente:

- Si existe un desplazamiento a teclado estándar que se corresponda con cada desplazamiento a teclado ideográfico (es decir, los caracteres de control de desplazamiento a teclado estándar e ideográfico deben estar equilibrados)
- Si existe un número par (dos como mínimo) entre el desplazamiento a teclado estándar y el desplazamiento a teclado ideográfico
- La ausencia de un desplazamiento a teclado ideográfico intercalado en los datos DBCS

Si no se cumplen las condiciones anteriores, el compilador emitirá un mensaje de diagnóstico y el literal no se volverá a analizar. Como resultado, si existen literales de tipo carácter en los programas OPM RPG que no superen la comprobación de transparencia realizada por el compilador OPM RPG, esos programas tendrán errores de compilación en ILE RPG.

2. En OPM RPG/400, si hay dos apóstrofes consecutivos entre caracteres de control de desplazamiento a teclado ideográfico y a teclado estándar dentro de un literal de tipo carácter, los dos apóstrofes consecutivos se considerarán como un único apóstrofo si el literal de tipo carácter no es un literal transparente. El literal de tipo carácter no será un literal transparente si:
 - El literal de tipo carácter no empieza por un apóstrofo seguido de un desplazamiento a teclado ideográfico
 - El literal de tipo carácter falla la comprobación de transparencia que realiza el compilador
 - El usuario no ha especificado que el compilador deba realizar una comprobación de transparencia

En ILE RPG, si hay dos apóstrofes consecutivos entre caracteres de control de desplazamiento a teclado ideográfico y a teclado estándar dentro de un literal de tipo carácter, los apóstrofes no se considerarán como un único apóstrofo. Un par de apóstrofes dentro de un literal de tipo carácter sólo se considerarán como un apóstrofo único si no están entre un carácter de control de desplazamiento a teclado ideográfico y otro a teclado estándar.

3. En ILE RPG, si desea evitar la comprobación de literales para caracteres de desplazamiento a teclado ideográfico (es decir, que no desea que un carácter de desplazamiento a teclado ideográfico se interprete como tal), deberá especificar todo el literal como un literal hexadecimal. Por ejemplo, si tiene un literal 'AoB' donde 'o' representa un carácter de control de desplazamiento a teclado ideográfico, deberá codificar este literal como X'C10EC2'.

Apéndice B. Utilización de la ayuda para la conversión de RPG III a RPG IV

Los diseños de las especificaciones fuente de RPG IV difieren notablemente de los diseños de RPG III del entorno System/38™ y de OPM RPG/400. Por ejemplo, se han modificado las posiciones de las entradas de las especificaciones así como los tipos de especificaciones disponibles. Los diseños de las especificaciones de RPG IV no son compatibles con los diseños anteriores. Para aprovechar al máximo las características de RPG IV, se deberán convertir los miembros fuente RPG III y RPG/400 de las aplicaciones al formato fuente de RPG IV.

Nota: Los tipos válidos de miembros fuente que puede convertir son RPG, RPT, RPG38, RPT38, SQLRPG y la que está en blanco. La ayuda para la conversión no da soporte a la conversión de RPG36, RPT36, ni la de otros tipos de miembros fuente que no sean RPG.

Si tiene prisa y desea comenzar ya, diríjase al apartado “Conversión del código fuente” en la página 432 y siga las indicaciones generales.

Visión general de la conversión

Para convertir programas fuente al formato fuente de RPG IV llame a la ayuda para la conversión utilizando el mandato CL Convertir fuente RPG (CVTRPGSRC). La ayuda para la conversión convierte:

- Un solo miembro
- Todos los miembros de un archivo físico fuente
- Todos los miembros con un prefijo de nombre de miembro común dentro del mismo archivo

Para disminuir la probabilidad de que se produzcan problemas de conversión, opcionalmente puede incluir los miembros /COPY en el código fuente convertido. Para mayor comodidad al leer el código, también tiene la opción de incluir plantillas de especificaciones en el código fuente convertido.

La ayuda para la conversión convierte cada miembro fuente línea a línea. Después de cada conversión de miembros, actualiza un archivo de anotaciones cronológicas si ha especificado un archivo de anotaciones cronológicas en el mandato. También es posible obtener un informe de la conversión que incluya información como, por ejemplo, los errores de conversión, las sentencias /COPY, las operaciones CALL y el estado de la conversión.

La ayuda para la conversión presupone que el código fuente está libre de errores de compilación. Si es así, entonces convertirá de forma satisfactoria la mayor parte del código fuente. En algunos casos, puede haber un pequeño fragmento del código que es posible que deba convertir manualmente. Algunos de estos casos los identifica la ayuda para la conversión. Otros no se detectan hasta que el usuario intente compilar el fuente convertido. Para ver cuáles puede identificar la ayuda para la conversión, puede ejecutar la ayuda para la conversión utilizando el miembro no convertido como entrada, y especificar un informe de conversión pero ningún miembro de salida. Para obtener información sobre los tipos de codificación que no pueden convertirse, consulte el apartado “Resolución de problemas de conversión” en la página 448.

Consideraciones sobre los archivos

La ayuda para la conversión opera en miembros de archivos. Esta sección presenta información sobre los diferentes aspectos de los archivos que deben tenerse en cuenta cuando se utiliza la ayuda para la conversión.

Tipos de miembros fuente

La Tabla 31 lista los distintos tipos de miembros fuente, indica si el tipo de miembro se puede convertir y también indica el tipo de miembro fuente de salida.

Tabla 31. Tipos de miembros fuente y su estado de conversión

Tipo de miembro fuente	¿Conversión?	Tipo de miembro convertido
RPG	Sí	RPGLE
RPG38	Sí	RPGLE
RPT	Sí	RPGLE
RPT38	Sí	RPGLE
'blank'	Sí	RPGLE
RPG36	No	N/A
RPT36	No	N/A
SQLRPG	Sí	SQLRPGLE
Cualquier otro tipo	No	N/A

Si el tipo de miembro fuente es 'blank', la ayuda para la conversión supondrá que tiene un tipo de miembro RPG. Si el tipo de miembro fuente está en blanco para un miembro fuente del generador automático de informes, deberá asignar al miembro el tipo de miembro fuente correcto (RPT o RPT38) antes de convertirlo. Si lo hace, la ayuda para la conversión ampliará automáticamente el miembro fuente del generador automático de informes para que se pueda convertir adecuadamente. La expansión es necesaria ya que el ILE RPG no da soporte a los miembros fuente del generador automático de informes.

Para obtener más información sobre cómo convertir miembros fuente de un generador automático de informes, consulte “Conversión de miembros fuente del generador automático de informes” en la página 440.

Longitud de registro de archivos

La longitud de registro recomendada para el archivo físico fuente convertido es de 112 caracteres. Esta longitud de registro tiene en cuenta la estructura RPG IV, tal como se muestra en la Figura 210 en la página 431. La longitud de registro recomendada de 112 caracteres corresponde también a la cantidad máxima de información que cabe en una línea de un listado de compilador.

Visión general de la conversión

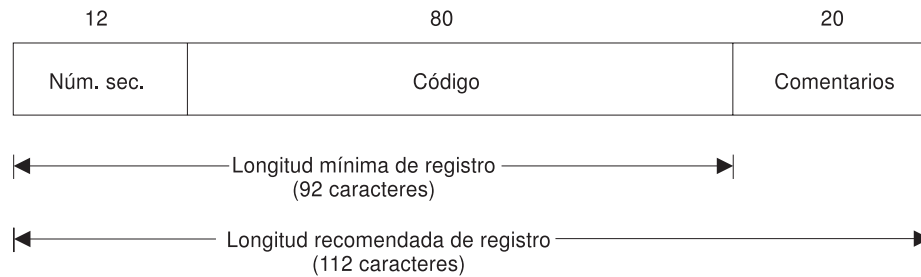


Figura 210. RPG IV Desglose de la longitud de registro

Si el archivo fuente convertido tiene una longitud de registro menor que 92 caracteres, entonces se emitirá un mensaje de error y se detendrá la conversión. Esto se debe a que la longitud de registro no es lo bastante larga para contener los 80 caracteres permitidos para el código fuente y por tanto es probable que se pierda parte del código.

Nombres de archivo y miembro

El miembro no convertido y el miembro para la salida convertida sólo pueden tener el mismo nombre si se encuentran en archivos o bibliotecas distintos.

El nombre del miembro o miembros fuente convertido(s) depende de si convierte un miembro o varios. Si convierte un miembro, el valor por omisión es dar al miembro fuente convertido el mismo nombre que el miembro no convertido. Es posible, desde luego, especificar un nombre distinto para el miembro de salida. Si está convirtiendo todos los miembros fuente de un archivo, o un grupo de ellos, utilizando un nombre genérico, se dará automáticamente a los miembros el mismo nombre que a los miembros fuente no convertidos.

Tenga en cuenta que es opcional especificar el nombre de archivo, de biblioteca y de miembro para la salida convertida. Si no especifica ninguno de estos nombres, la salida convertida se colocará en el archivo QRPGLSRC y tendrá un nombre de miembro igual al nombre del miembro no convertido. (Se buscará en la lista de bibliotecas el archivo QRPGLSRC).

El archivo de anotaciones cronológicas

La ayuda para la conversión utiliza un archivo de anotaciones cronológicas para proporcionar seguimientos de auditoría del estado de cada conversión de miembro fuente. Examinando el archivo de anotaciones cronológicas, puede determinar el estado de las conversiones anteriores. Puede acceder al archivo de anotaciones cronológicas con un programa escrito por el usuario para otros procesos, como por ejemplo la compilación y el enlace de programas.

Si se especifica que debe actualizarse un archivo de anotaciones cronológicas, el formato del registro deberá coincidir con el formato del archivo de base de datos "modelo" QARNCVTLG, de la biblioteca QRPGL, suministrado por IBM. La Figura 217 en la página 448 muestra las DDS de este archivo. Utilice el siguiente mandato CRTDUPOBJ para crear una copia de esta plantilla en su propia biblioteca, a la que se hace referencia como MIBIB. Es posible que desee poner al archivo de anotaciones cronológicas el nombre QARNCVTLG, ya que este es el nombre por omisión del archivo de anotaciones cronológicas que se utiliza en la Ayuda para la Conversión.

```
CRTDUPOBJ OBJ(QARNCVTLG) FROMLIB(QRPGL) OBJTYPE(*FILE)
          TOLIB(MIBIB) NEWOBJ(QARNCVTLG)
```

Visión general de la conversión

Debe tener autorización de gestión de objetos, operativa y de adición sobre el archivo de anotaciones cronológicas al que accede la ayuda para la conversión.

Para obtener información sobre cómo utilizar el archivo de anotaciones cronológicas, consulte el apartado “Utilización del archivo de anotaciones cronológicas” en la página 447.

Requisitos de la herramienta de ayuda para la conversión

Para utilizar la ayuda para la conversión necesita tener la autorización siguiente:

- *autorización USE para el mandato CVTRPGSRC
- *autorización USE sobre la biblioteca que contiene el archivo fuente y los miembros fuente
- *autorización CHANGE sobre la nueva biblioteca que contendrá el archivo fuente y los miembros fuente convertidos
- autorización de gestión de objetos, operativa y de adición sobre el archivo de anotaciones cronológicas utilizado por la ayuda para la conversión

Además de los requisitos de autorización sobre objetos, puede haber requisitos de almacenamiento adicionales. Cada programa fuente convertido suele ser aproximadamente un 25% mayor al tamaño del programa antes de la conversión. Para utilizar la ayuda para la conversión necesita suficiente almacenamiento para almacenar los archivos fuente convertidos.

Funciones que no realiza la ayuda para la conversión

- La ayuda para la conversión no soporta la conversión del formato RPG IV de nuevo a los formatos RPG III o RPG/400.
- El compilador RPG IV no soporta la conversión automática de los miembros fuente RPG III o RPG/400 al formato fuente RPG IV *en tiempo de compilación*.
- La ayuda para la conversión no soporta la conversión de programas fuente RPG II al formato fuente de RPG IV. Sin embargo, se puede utilizar en primer lugar la **ayuda para la conversión de RPG II a RPG III** y a continuación la ayuda para la conversión de RPG III a RPG IV.
- La ayuda para la conversión no realiza cambios técnicos en el código fuente, excepto cuando son necesarios (por ejemplo, el número de indicadores de condicionamiento).
- La ayuda para la conversión no crea archivos. Para ejecutarla, deberán existir el archivo de anotaciones cronológicas y el archivo de salida.

Conversión del código fuente

Esta sección explica cómo convertir programas fuente al formato RPG IV. Trata sobre el mandato CVTRPGSRC, el cual inicia la ayuda para la conversión y sobre cómo utilizarlo.

>Para convertir el código fuente al formato RPG IV, siga estos pasos generales:

1. Si utiliza un área de datos como una especificación de control, debe crear una nueva área de datos en el formato RPG IV. Consulte el capítulo sobre especificaciones de control en la publicación *ILE RPG Reference* para obtener más información.
2. Cree un archivo de anotaciones cronológicas, si es necesario.

A menos que especifique LOGFILE(*NONE), debe existir un archivo de anotaciones cronológicas al que acceda la ayuda para la conversión. Si no tiene

ninguno, puede crearlo utilizando el mandato CRTDUPOBJ. Para obtener más información, consulte los apartados “El archivo de anotaciones cronológicas” en la página 431 y “Utilización del archivo de anotaciones cronológicas” en la página 447.

3. Cree el archivo para los miembros fuente convertidos.

La ayuda para la conversión no creará ningún archivo. Debe crear el archivo de salida para el fuente convertido antes de ejecutar el mandato CVTRPGSRC. El nombre recomendado y la longitud de registro para el archivo de salida es QRPGLSRC y 112 caracteres respectivamente. Para obtener información adicional acerca de los archivos, consulte el apartado “Consideraciones sobre los archivos” en la página 430.

4. Convierta el fuente utilizando el mandato CVTRPGSRC.

Necesita entrar el nombre del archivo y del miembro que se va a convertir. Si acepta los valores por omisión, obtendrá un miembro convertido el archivo QRPGLSRC. El nombre del miembro se corresponderá con el nombre del miembro fuente no convertido. Los miembros /COPY no se ampliarán en el miembro fuente convertido, a menos que sea de tipo RPT o RPT38. Se generará un informe sobre la conversión.

Consulte la publicación “El mandato CVTRPGSRC” para obtener más información.

5. Compruebe si existen errores en el archivo de anotaciones cronológicas o en el informe de errores. Para obtener más información, consulte el apartado “Análisis de la conversión” en la página 444.

6. Si existen errores, corríjlos y vaya al paso 4.

7. Si no hay errores, cree el programa. Para obtener información sobre cómo crear programas ILE RPG, consulte el “Capítulo 6. Creación de un programa con el mandato CRTBNDRPG” en la página 61.

8. Si el miembro fuente convertido todavía tiene problemas de compilación, estos probablemente se deban a que su miembro fuente primario contiene directivas del compilador /COPY. Tiene dos opciones para corregir esta situación:

- Vuelva a convertir el miembro fuente especificando EXPCPY(*YES) para ampliar los miembros de copia en el miembro fuente convertido.
- Corrija manualmente el resto de los errores utilizando el listado del compilador como guía.

Consulte el apartado “Resolución de problemas de conversión” en la página 448 para obtener más información.

9. Cuando el miembro fuente convertido se haya compilado satisfactoriamente, vuelva a probar el programa antes de transferirlo de nuevo a producción.

El mandato CVTRPGSRC

Para convertir el fuente RPG III o RPG/400 al nuevo formato RPG IV, utilice el mandato CVTRPGSRC para iniciar la ayuda para la conversión. La Tabla 32 muestra los parámetros de los mandatos basados en su función.

Tabla 32. Parámetros de CVTRPGSRC y sus valores por omisión agrupados por función

Identificación del programa	
FROMFILE	Identifica el nombre de biblioteca y archivo del fuente RPG que se va a convertir
FROMMBR	Identifica qué miembros fuente se van a convertir
TOFILE(*LIBL/QRPGLSRC)	Identifica el nombre de biblioteca y archivo de la salida convertida

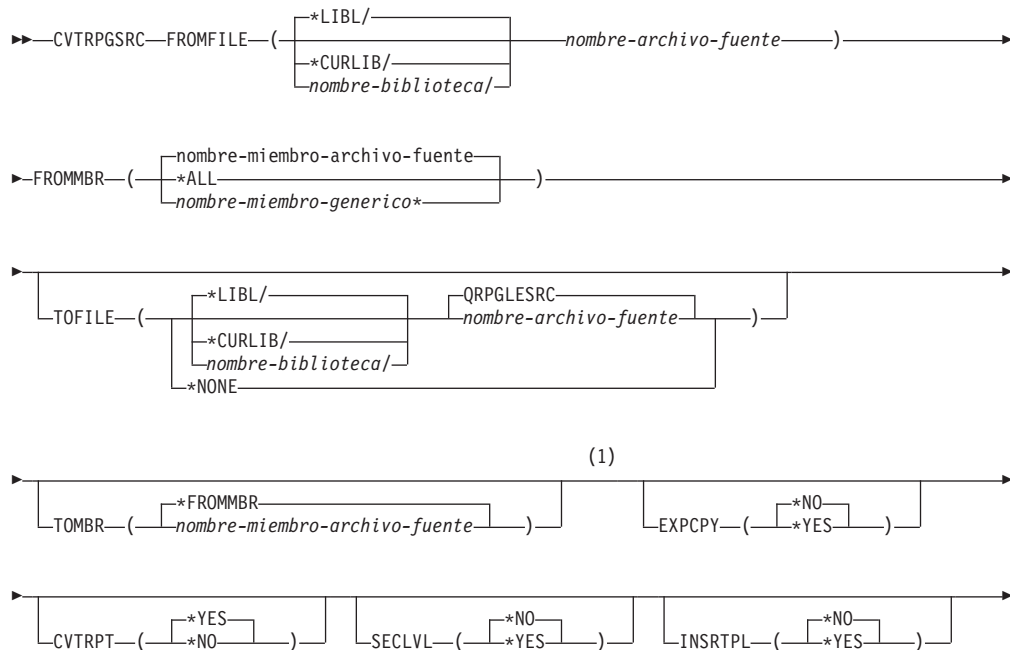
Conversión del código fuente

Tabla 32. Parámetros de CVTRPGSRC y sus valores por omisión agrupados por función (continuación)

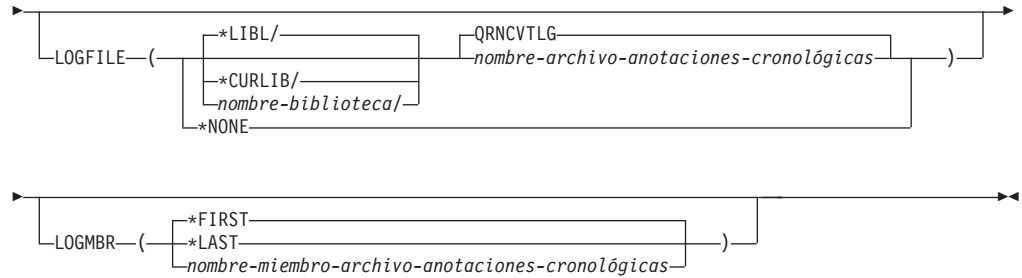
TOMBR(*FROMMBR)	Identifica los nombres de los miembros de archivo del fuente convertido
Proceso de conversión	
TOMBR	Si se especifica *NONE, no se salvará ningún miembro de archivo
EXPCPY(*NO)	Determina si se incluyen las sentencias /COPY en la salida convertida
INSRTPL(*NO)	Indica si las plantillas de especificaciones se deben incluir en la salida convertida
Resultado de conversión	
CVTRPT(*YES)	Determina si se debe producir un informe de conversión
SECLVL(*NO)	Determina si se debe incluir texto de mensaje de segundo nivel
LOGFILE(*LIBL/QRNCVTLG)	Identifica el archivo de anotaciones cronológicas para el informe de auditoría
LOGMBR(*FIRST)	Identifica qué miembro del archivo de anotaciones cronológicas se debe utilizar para el informe de auditoría

La sintaxis del mandato CVTRPGSRC se muestra a continuación.

Trabajo: B,I Pgm: B,I REXX: B,I Exec



Conversión del código fuente



Notas:

- 1 Todos los parámetros anteriores a este punto pueden especificarse por posición

Los parámetros y sus valores posibles aparecen a continuación del diagrama de sintaxis. Si necesita indicaciones, teclee CVTRPGSRC y pulse F4. Aparecerá la pantalla CVTRPGSRC, que lista los parámetros y suministra los valores por omisión. Para obtener una descripción de un parámetro de esta pantalla, coloque el cursor sobre el parámetro y pulse F1. También se dispone de la ayuda general acerca de los parámetros pulsando F1 sobre cualquier parámetro y pulsando a continuación F2.

FROMFILE

Especifica el nombre del archivo fuente que contiene el código fuente de RPG III o RPG que se va a convertir y la biblioteca donde está almacenado el archivo fuente. Este es un parámetro necesario; no hay ningún nombre de archivo por omisión.

nombre-archivo-fuente

Entre el nombre del archivo fuente que contenga el(los) miembro(s) fuente que hay que convertir.

*LIBL

El sistema busca en la lista de bibliotecas para encontrar la biblioteca donde está almacenado el archivo fuente.

*CURLIB

La biblioteca actual se utiliza para buscar el archivo fuente. Si no se ha especificado una biblioteca actual, se utilizará la biblioteca QGPL.

nombre-biblioteca

Entre el nombre de la biblioteca donde está almacenado el archivo fuente.

FROMMBR

Especifica el nombre del miembro (o de los miembros) que se va a convertir. Este es un parámetro necesario; no hay ningún nombre de miembro por omisión.

Los tipos válidos de miembros fuente que se van a convertir son RPG, RPT, RPG38, RPT38, SQLRPG y los blancos. El mandato Convertir fuente RPG no soporta los tipos de miembro fuente RPG36, RPT36 y otros tipos de miembro fuente que no sean RPG (por ejemplo, CLP y TXT).

nombre-miembro-archivo-fuente

Entre el nombre del miembro fuente que se va a convertir.

Conversión del código fuente

****ALL***

El mandato convierte todos los miembros del archivo fuente especificado.

nombre-miembro-generico*

Entre el nombre genérico de los miembros que tengan el mismo prefijo en sus nombres, seguido de un '*' (asterisco). El mandato convierte todos los miembros que tienen el nombre genérico en el archivo fuente especificado. Por ejemplo, especificar FROMMBR(PR*) dará como resultado la conversión de todos los miembros cuyos nombres empiecen por 'PR'.

(Consulte la publicación CL Programmer's Guide para obtener más información acerca del nombre genérico).

TOFILE

Especifica el nombre del archivo fuente que contiene los miembros fuente convertidos y la biblioteca donde está almacenado el archivo fuente convertido. Ha de existir el archivo fuente convertido, que deberá tener una longitud de registro de 112 caracteres: 12 para el número y la fecha de secuencia, 80 para el código y 20 para los comentarios.

QRPGLESRC

El archivo fuente por omisión QRPGLESRC contiene el miembro (o miembros) fuente convertido.

****NONE***

No se genera ningún miembro convertido. Se ignora el valor del parámetro TOMBR. Se debe especificar también CVTRPT(*YES), de lo contrario la conversión finalizará inmediatamente.

Esta característica permite encontrar algunos problemas potenciales sin necesidad de crear el miembro fuente convertido.

nombre-archivo-fuente

Entre el nombre del archivo fuente convertido que contenga el miembro (o los miembros) fuente convertido.

El nombre del archivo fuente TOFILE tiene que ser distinto del nombre del archivo fuente FROMFILE, si el nombre de la biblioteca TOFILE es el mismo que el de la biblioteca FROMFILE.

***LIBL**

El sistema busca en la lista de bibliotecas para encontrar la biblioteca donde está almacenado el archivo fuente convertido.

****CURLIB***

La biblioteca actual se utiliza para buscar el archivo fuente convertido. Si no se ha especificado una biblioteca actual, se utilizará la biblioteca QGPL.

nombre-biblioteca

Entre el nombre de la biblioteca donde está almacenado el archivo fuente convertido.

TOMBR

Especifica el nombre del miembro (o miembros) fuente convertido del archivo fuente convertido. Si el valor especificado en el parámetro FROMMBR es *ALL o genérico*, TOMBR deberá ser igual a *FROMMBR.

***FROMMBR**

El nombre de miembro especificado en el parámetro FROMMBR se utiliza como nombre del miembro fuente convertido. Si se especifica FROMMBR(*ALL), se convertirán todos los miembros fuente de FROMFILE. Los miembros fuente convertidos tienen los mismos nombres

Conversión del código fuente

que los miembros fuente originales. Si se especifica un nombre genérico en el parámetro FROMMBR, se convertirán todos los miembros fuente especificados que tengan el mismo prefijo en sus nombres. Los miembros fuente convertidos tienen los mismos nombres que los miembros fuente genéricos originales.

nombre-miembro-archivo-fuente

Entre el nombre del miembro fuente convertido. Si el miembro no existe, será creado.

EXPCPY

Especifica si el miembro o miembros /COPY se han expandido en el miembro fuente convertido. Sólo se deberá especificar EXPCPY(*YES) si se tienen problemas de conversión que estén relacionados con los miembros /COPY.

Nota: Si el miembro es de tipo RPT o RPT38, EXPCPY(*YES) o EXPCPY(*NO) no surtirá efecto ya que el programa generador automático de informes siempre ampliará los miembros /COPY.

*NO

No expande el(los) miembro(s) de archivo /COPY en el fuente convertido.

*YES

Expande el(los) miembro(s) de archivo /COPY en el fuente convertido.

CVTRPT

Especifica si se ha imprimido un informe de conversión.

*YES

El informe de conversión se ha imprimido.

*NO

El informe de conversión no se ha imprimido.

SECLVL

Especifica si se ha imprimido texto de segundo nivel en el informe de conversión de la sección de resumen de mensajes.

*NO

No se ha imprimido texto de mensaje de segundo nivel en el informe de conversión.

*YES

Se ha imprimido texto de mensaje de segundo nivel en el informe de conversión.

INSRTPL

Especifica si las plantillas de especificación de ILE RPG (H-, F-, D-, I-, C- y/o la plantilla de especificación O), se han insertado en el miembro (o miembros) fuente convertido. El valor por omisión es *NO.

*NO

No se ha insertado una plantilla de especificación en el miembro fuente convertido.

*YES

Se ha insertado una plantilla de especificación en el miembro fuente convertido. Se inserta cada plantilla de especificación al comienzo de la sección de especificación apropiada.

LOGFILE

Especifica el nombre del archivo de anotaciones cronológicas que se utiliza para hacer un seguimiento de la información sobre la conversión. A menos que

Conversión del código fuente

se especifique *NONE, deberá haber un archivo de anotaciones cronológicas. El archivo ya deberá existir y tendrá que ser un archivo físico de datos. Puede crear el archivo de anotaciones cronológicas utilizando el mandato CPYF con el archivo QARNCVTLG "Desde objeto" de la biblioteca QRPGL y el archivo QRNCVTLG "Objeto nuevo" de su biblioteca.

QRNCVTLG

El archivo de anotaciones cronológicas por omisión QRNCVTLG se utiliza para contener la información sobre la conversión.

*NONE

La información sobre la conversión no se escribe en un archivo de anotaciones cronológicas.

nombre-archivo-anotaciones-cronológicas

Entre el nombre del archivo de anotaciones cronológicas que se va a utilizar para hacer un seguimiento de la información sobre la conversión.

*LIBL

El sistema busca en la lista de bibliotecas para encontrar la biblioteca donde está almacenado el archivo de anotaciones cronológicas.

nombre-biblioteca

Entre el nombre de la biblioteca donde está almacenado el archivo de anotaciones cronológicas.

LOGMBR

Especifica el nombre de miembro del archivo de anotaciones cronológicas que se utiliza para hacer un seguimiento de la información sobre la conversión. La nueva información se añade a los datos existentes en el miembro del archivo de anotaciones cronológicas especificado.

Si el archivo de anotaciones cronológicas no contiene miembros, se creará un miembro que tenga el mismo nombre que el archivo de anotaciones cronológicas.

*FIRST

El mandato utiliza el primer miembro del archivo de anotaciones cronológicas especificado.

*LAST

El mandato utiliza el último miembro del archivo de anotaciones cronológicas especificado.

nombre-miembro-archivo-anotaciones-cronológicas

Entre el nombre de miembro del archivo de anotaciones cronológicas que se utiliza para hacer un seguimiento de la información sobre la conversión.

Conversión de un miembro mediante los valores por omisión

Puede beneficiarse de los valores por omisión suministrados en el mandato CVTRPGSRC. Simplemente entre:

CVTRPGSRC FROMFILE(nombre de archivo) FROMMBR(nombre de miembro)

Esto provocará la conversión del miembro fuente especificado. La salida se colocará en el archivo QRPGLSRC en la biblioteca de la lista de bibliotecas que contenga este archivo. Los miembros /COPY no se ampliarán, ni se insertará ninguna plantilla de especificaciones, y se generará el informe de conversión. Se actualizará el archivo de anotaciones cronológicas QRNCVTLG.

Nota: ya tienen que existir los archivos QRPGLSRC y QRNCVTLG.

Conversión de todos los miembros de un archivo

Es posible convertir todos los miembros de un archivo físico fuente especificando FROMMBR(*ALL) y TOMBR(*FROMMBR) en el mandato CVTRPGSRC. La ayuda para la conversión intentará convertir todos los miembros del archivo especificado. Si algún miembro presenta alguna anomalía en la conversión, el proceso de conversión continuará.

Por ejemplo, si desea convertir todos los miembros fuente del archivo QRPGSRC al archivo QRPGLSRC, deberá entrar:

```
CVTRPGSRC  FROMFILE(RPGANT/QRPGSRC)
           FROMMBR(*ALL)
           TOFILE(RPGNUE/QRPGSRC)
           TOMBR(*FROMMBR)
```

Este mandato convierte todos los miembros fuente de la biblioteca RPGANT del archivo físico fuente QRPGSRC. Los nuevos miembros se crean en la biblioteca RPGNUE del archivo físico fuente QRPGLSRC.

Si prefiere guardar todos los fuentes (fuente DDS, fuentes RPG, etc.) en el mismo archivo, también puede convertir los miembros fuente RPG en un solo paso, especificando FROMMBR(*ALL). La ayuda para la conversión sólo convertirá los miembros con un tipo de RPG válido (consulte la Tabla 31 en la página 430).

Conversión de algunos miembros de un archivo

Si necesita convertir sólo algunos miembros que están en un archivo físico fuente y dichos miembros comparten un prefijo común en el nombre de miembro, puede convertirlos especificando el prefijo seguido de un * (asterisco).

Por ejemplo, si desea convertir todos los miembros con el prefijo PAY, deberá entrar:

```
CVTRPGSRC  FROMFILE(RPGANT/QRPGSRC)
           FROMMBR(PAY*)
           TOFILE(RPGNUE/QRPGSRC)
           TOMBR(*FROMMBR)
```

Este mandato convierte todos los miembros fuente de la biblioteca RPGANT del archivo físico fuente QRPGSRC. Los nuevos miembros se crean en la biblioteca NEWRPG del archivo físico fuente QRPGLSRC.

Realización de una conversión de prueba

Puede realizar una ejecución de prueba en los miembros fuente que le puedan ocasionar problemas a la hora de convertirlos. Entonces obtendrá un informe sobre la conversión del miembro fuente convertido que puede identificar ciertos errores de conversión.

Por ejemplo, para realizar una conversión de prueba en el miembro fuente PAYROLL, teclee:

```
CVTRPGSRC  FROMFILE(RPGANT/QRPGSRC)
           FROMMBR(NOMINAS)
           TOFILE(*NONE)
```

El parámetro TOMBR debe especificarse como *FROMMBR. Sin embargo, como éste es el valor por omisión, no necesita especificarlo a menos que haya modificado

Conversión del código fuente

el valor por omisión. El parámetro CVTRPT debe especificarse como *YES — este es también el valor por omisión. Si no lo hace, entonces la conversión se detendrá inmediatamente.

La utilización del parámetro TOFILE(*NONE) evita que la ayuda para la conversión genere un miembro convertido, pero le permite generar un informe de conversión. Para obtener más información acerca del informe de conversión, consulte el apartado “Análisis de la conversión” en la página 444.

Obtención de informes de conversión

La ayuda para la conversión normalmente genera un informe de conversión cada vez que se emite el mandato. El nombre del archivo en spool corresponde al nombre de archivo especificado en el parámetro TOFILE. Si intenta convertir un miembro que ya existe o que tiene un tipo de miembro al que no se da soporte, se imprimirá un mensaje en las anotaciones de trabajo que indique que estos miembros no se han convertido. También se actualiza el archivo de anotaciones cronológicas, si se ha solicitado, de modo que refleje que no se ha llevado a cabo la conversión. Sin embargo, el informe no contendrá información relacionada con estos miembros.

El informe de conversión incluye la información siguiente:

- Opciones del mandato CVTRPGSRC
- Sección del fuente que incluye:
 - errores o avisos de conversión
 - operaciones de llamada
 - directivas /COPY
- Resumen de mensajes
- Resumen final

Los mensajes de errores de conversión proporcionan sugerencias sobre cómo corregir el error. Además, se indican las operaciones CALL y las directivas /COPY del fuente no convertido para ayudarle a identificar las distintas partes de la aplicación que esté convirtiendo. En general, debe convertir todos los componentes RPG de una aplicación al mismo tiempo.

Si no desea un informe de conversión, especifique CVTRPT(*NO).

Conversión de miembros fuente del generador automático de informes

Cuando se detecta un miembro fuente del generador automático de informes (de tipo RPT o RPT38) en un programa fuente RPG III o RPG/400 OPM, la ayuda para la conversión llama al mandato CRTRPTPGM para ampliar el miembro fuente y a continuación lo convierte. (Esto se debe a que ILE RPG no da soporte al generador automático de informes).

El programa generador automático de informes genera un archivo en spool cada vez que lo llama la ayuda para la conversión. Es posible que desee revisar este archivo para ver si se ha producido algún error en la ampliación del generador automático de informes, ya que estos errores no aparecerán en el informe sobre la conversión.

En especial, es posible que desee comprobar si existe algún mensaje de error en el archivo en spool del generador automático de informes que indique que no se han

encontrado miembros /COPY. La ayuda para la conversión no sabrá si faltan estos archivos. Sin embargo, sin estos archivos, es posible que no pueda convertir el fuente satisfactoriamente.

Nota: Si el tipo de miembro fuente del miembro que va a convertirse no es RPT o RPT38 y el miembro es un miembro fuente del generador automático de informes, deberá asignar al miembro el tipo de miembro fuente correcto (RPT o RPT38) antes de convertirlo, de lo contrario pueden producirse errores de conversión.

El generador automático de informes da soporte a los datos de tiempo de compilación en los miembros /COPY. RPG IV no les da soporte. Si mantiene datos de tiempo de compilación en los miembros /COPY para que varios programas puedan utilizar los datos, considere la posibilidad de mover los datos de tiempo de compilación a un espacio de usuario y de acceder a ellos mediante las API del espacio de usuario.

Conversión de miembros fuente con SQL incluido

Cuando se convierte código que contiene SQL y el código SQL continúa en varias líneas, puede ocurrir lo siguiente:

- Si hay líneas de continuación pero la columna 74 está en blanco, simplemente se copiará la línea en el miembro ILE.

Nota: Esto podría ser un problema si la columna 74 es un carácter en blanco dentro de una serie de caracteres.

- Si la columna 74 no está en blanco, todos los códigos SQL desde esa línea hasta /END-EXEC estarán concatenados y serán copiados en el miembro ILE, rellorando las 80 columnas. Si ocurre esto:
 - Se ignorarán los comentarios de la columna 75 en adelante.
 - Se copiarán en el miembro ILE las líneas de comentarios intercalados (C*) antes de copiar el código concatenado.
 - Se pueden producir problemas si se subdividen los literales DBCS.

Si no desea que se produzca esta concatenación y reasignación de formato, asegúrese de que la columna 74 esté en blanco.

Inserción de plantillas de especificaciones

Como las especificaciones fuente para RPG IV son nuevas, puede que desee insertar plantillas de especificaciones en el fuente convertido. Para insertar plantillas, especifique INSRTPL(*YES) en el mandato CVTRPGSRC. El valor por omisión es INSRTPL(*NO).

Conversión de fuente desde un archivo de datos

La ayuda para la conversión convertirá el fuente desde un archivo de datos. Como los archivos de datos generalmente no tiene números de secuencia, la longitud de registro mínima del archivo para colocar la salida convertida es de 80 caracteres. (Consulte la Figura 210 en la página 431). La longitud de registro recomendada es de 112 caracteres para un archivo de datos.

Nota: si el archivo de datos tiene números de secuencia, debe eliminarlos antes de ejecutar la ayuda para la conversión.

Ejemplo de conversión de fuente

El ejemplo muestra un miembro fuente RPG III de ejemplo que se va a convertir a RPG IV. La Figura 211 muestra el fuente de la versión RPG III.

```

H                                     TSTPGM
FFILE1 IF E                           COMM1
FQSYSPRT 0 F 132 OF LPRINTER
LQSYSPRT 60FL 560L
E          ARR1 3 3 1                COMM2
E          ARR2 3 3 1
IFORMAT1
I          NOMANT                     NOMBRE
I* COMENTARIO SOBRE ESTRUCTURA DE DATOS
IDS1      DS
I
I          1 3 FIELD1
I* COMENTARIO DE CONSTANTE CON NOMBRE
I          'XYZ' C                     CONST1
I          4 6 ARR1                     COMM3
C          ARR1,3 DSPLY
C          READ FORMAT1                01
C          NAME DSPLY
C          SETON                        LR
C          EXCPTOUTPUT
OQSYSPRT E 01      OUTPUT
O          ARR2,3 10
**
123
**
456

```

Figura 211. Fuente RPG III para PRUEBA1

Para convertir este fuente, entre:

```

CVTRPGSRC FROMFILE(MIBIB/QRPGSRC) FROMMBR(PRUEBA1)
          TOFILE(MILIB/QRPGLESRC) INSRTPL(*YES)

```

El fuente convertido se muestra en la Figura 212 en la página 443.

```

1 .....H*unciones+++++++Comentarios+++++
2      H DFTNAME(TSTPGM)
3 .....F*ombarchi+IPEASFRlen+LKlen+AIDispos+.Funciones+++++++Comentarios+++++
4      FFILE1      IF      E      DISK      COMM1
5      FQSYSPRT    0      F  132      PRINTER OFLIND(*INOF)
6      F      FORMLEN(60)
7      F      FORMOFL(56)
8 .....D*Nombre+++++++ETDsDesde+++A/L+++IDc.Funciones+++++++Comentarios+++++
9      D ARR2      S      1      DIM(3) CTDATA PERRCD(3)
10     D* COMENTARIO SOBRE ESTRUCTURA DE DATOS
11     D DS1      DS
12     D FIELD1      1      3
13     D ARR1      4      6
14     D      DIM(3) CTDATA PERRCD(3)      COMM2
15     D* COMENTARIO DE CONSTANTE CON NOMBRE
16     D CONST1      C      CONST('XYZ')      COMM3
17 .....I*ombarchi+SqNORiPos1+NCCPos2+NCCPos3+NCC.....Comentarios+++++
18 .....I*.....Ext_field+Fmt+SPFrom+To+++DcField+++++++L1M1FrP1MnZr.....Comentarios+++++
19     IFORMAT1
20     I      OLDNAME      NAME
21 .....C*0N01Factor1+++++++Opcode(E)+Factor2+++++++Result+++++++Len+++D+HiLoEq....Comentarios+++++
22     C      ARR1(3)      DSPLY      FORMAT1      01
23     C      READ
24     C      NAME      DSPLY
25     C      SETON      LR
26     C      EXCEPT      OUTPUT
27     OQSYSPRT    E      OUTPUT      01
28     O      ARR2(3)      10
29 **CTDATA ARR1
30 123
31 **CTDATA ARR2
32 456

```

Figura 212. Fuente convertido (RPG IV) para PRUEBA1

Tenga en cuenta lo que sigue a continuación, acerca del fuente convertido:

- Los nuevos tipos de especificación son H (control), F (archivo), D (definición), I (entrada), C (cálculo) y O (salida); deben estar en este orden.

El fuente convertido contiene plantillas de especificaciones para los nuevos tipos, ya que se ha especificado INSRTPL(*YES) en CVTRPGSRC.

- La especificaciones de control, de archivo y definición están orientadas a palabras clave. Consulte las líneas 2, 4 - 7, y 9 - 16.
- El miembro ILE tiene un nuevo tipo de especificación o definición. Se utiliza para definir campos, matrices y constantes con nombre autónomos, así como estructuras de datos.

En este ejemplo,

- ARR2 está definido como una matriz autónoma (línea 9)
- La estructura de datos DS1 está definida como una estructura de datos con dos subcampos FIELD1 y ARR1 (líneas 11 - 14)
- La constante CONST1 está definida como una constante (línea 16)

Las especificaciones de entrada (I) se utilizan ahora solamente para definir registros y campos de un archivo. Consulte las líneas 19 - 20.

- Las especificaciones de extensión (E) se han eliminado. Las matrices y tablas se definen ahora utilizando las especificaciones de definición.
- Las entradas del archivo de direcciones de registros (RAF) de las especificaciones de extensión se han sustituido por la palabra clave RAFDATA de la especificación de descripción de archivos.

Ejemplo de conversión de fuente

- Las especificaciones del contador de líneas se han eliminado. Se han sustituido por las palabras clave FORMLEN y FORMOFL de la especificación de descripción de archivos. Consulte las líneas 6 y 7.
- Se han ampliado todos los tipos de especificaciones para permitir nombres de 10 caracteres para los campos y los archivos.
- En RPG IV, las estructuras de datos (que se definen utilizando especificaciones de definición) deben preceder a las especificaciones de entrada.

Tenga en cuenta que en el fuente convertido, se ha movido la estructura de datos DS1 (línea 11) para que preceda la especificación que contiene la información de FORMAT1 (línea 19).

- En RPG III, las constantes con nombre pueden aparecer en medio de una estructura de datos. Esto no se permite en RPG IV.

En el fuente convertido, se ha movido CONST1 (línea 16) para que vaya detrás de la estructura de datos DS1 (línea 11).

- Si se mueve una especificación, también se mueven los comentarios que la precedan.

En el fuente convertido, se han movido los comentarios por encima de CONST1 y DS1 con las especificaciones que les seguían.

- En RPG III, para definir una matriz como un subcampo de estructura de datos, se debe definir tanto la matriz como el subcampo de estructura de datos con el mismo nombre. Esta definición doble no está permitida en RPG IV. En lugar de eso, se deben especificar los atributos de matriz al definir los subcampos utilizando la nueva sintaxis de palabras claves.

En este ejemplo, ARR1 se define dos veces en la versión OPM, pero se ha fusionado en una sola definición en el fuente convertido. Consulte las líneas 13 y 14.

Si se fusionan las especificaciones de las matrices de RPG III se puede cambiar el orden las definiciones de matriz. Si las matrices cuyo orden ha cambiado son matrices de tiempo de compilación, es posible que la carga de datos de la matriz resulte afectada. Para solucionar este problema, RPG IV proporciona un formato de palabra clave para los registros **. Después de ** debe entrar una de las palabras clave FTRANS, ALTSEQ o CTDATA. Si la palabra clave es CTDATA, se debe entrar el nombre de la matriz o de tabla en las posiciones 10 - 19.

En este ejemplo, la matriz ARR2 pasa a preceder a la matriz ARR1, debido a que se han fusionado las dos especificaciones RPG III para ARR2. La ayuda para la conversión ha insertado las palabras clave y los nombres de las matrices en los registros ** convertidos, lo cual asegura la carga correcta de los datos de tiempo de compilación. Consulte las líneas 29 y 31.

- Tenga en cuenta que se ha modificado la sintaxis de las matrices. La notación ARR1,3 de RPG III es ARR1(3) en RPG IV. Consulte la línea 28.

Análisis de la conversión

La ayuda para la conversión proporciona dos maneras de analizar los resultados de la conversión. Estas maneras son:

- el informe de errores de conversión
- el archivo de anotaciones cronológicas

Utilización del informe de conversión

La ayuda para la conversión genera un informe sobre la conversión si se especifica el parámetro CVTRPT(*YES) en el mandato CVTRPGSRC. El nombre del archivo en spool es el mismo que el nombre de archivo especificado en el parámetro TOFILE.

El informe sobre la conversión consta de cuatro partes:

1. opciones del mandato CVTRPGSRC
2. sección del fuente
3. resumen de mensajes
4. resumen final

La primera parte del listado incluye un resumen de las opciones de mandato utilizadas por CVTRPGSRC. La Figura 213 muestra el resumen del mandato para una conversión de ejemplo.

5769RG1 V4R4M0 990521 RN	IBM ILE RPG	AS400S01	12/30/99 20:41:35	Página 1
Mandato	CVTRPGSRC			
Emitido por	DAVE			
Desde archivo	QRPGSRC			
Biblioteca	MIBIB			
Desde miembro	REPORT			
A archivo	QRPGLESRC			
Biblioteca	MIBIB			
A miembro	*FROMMBR			
Archivo de anotaciones	*NONE			
Biblioteca				
Anotar miembro	*FIRST			
Ampliar miembros de copia	*NO			
Imprimir informe de conversión	*YES			
Incluir texto de segundo nivel	*YES			
Insertar plantilla de especific.	*YES			

Figura 213. Resumen del mandato de un informe de conversión de ejemplo

La sección fuente incluye las líneas que contienen mensajes informativos, de aviso o de error asociados con los mismos. Estas líneas presentan un asterisco (*) en la columna 1 para facilitar la revisión en SEU. El resumen de mensajes contiene los tres tipos de mensajes.

Dos mensajes informativos que pueden tener un interés especial son:

- RNM0508 — señala las sentencias /COPY
- RNM0511 — señala las operaciones CALL

Todos los miembros /COPY de un programa deben convertirse para que el programa ILE RPG correspondiente compile sin errores. De forma similar, puede que desee convertir todos los miembros relacionados mediante CALL a la vez. Utilice esta parte del informe como ayuda para identificar estos miembros. La Figura 214 en la página 446 muestra la sección del fuente para la conversión de ejemplo.

Análisis de la conversión

```
5769RG1 V4R4M0 990521 RN      IBM ILE RPG      AS400S01      12/30/99 20:41:35      Página  2
Desde el archivo . . . . . : MYLIB/QRPGSRC(REPORT)
Al archivo . . . . . : MYLIB/QRPGLESRC(REPORT)
Anotar archivo . . . . . : *NONE
                               I n f o r m e  d e  c o n v e r s i ó n
Secuencia <----- Especificaciones de fuente -----><----- Comentarios -----> Página
Número  ....1....+...2....+...3....+...4....+...5....+...6....+...7....+...8....+...9....+...10....+...11....+...12 Línea
000002 C          CALL      PROG1
*RNM0511 00 se ha encontrado el código de operación CALL.
000003 C/COPY COPYCODE
*RNM0508 00 Encontrada directiva del compilador /COPY.
000004 C          FREE      PROG2
*RNM0506 30 Código de operación FREE no soportado en RPG IV.

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

Figura 214. Sección del fuente del informe de conversión de ejemplo

El resumen de mensajes del listado muestra los diferentes mensajes que se han emitido. Si especifica SECLVL(*YES), aparecerán los mensajes de segundo nivel en el resumen de mensajes. La Figura 215 muestra la sección de mensajes para la conversión de ejemplo, incluidos los mensajes de segundo nivel.

```
5769RG1 V4R4M0 990521 RN      IBM ILE RPG      AS400S01      12/30/99 20:41:35      Página  2
                               R e s u m e n   d e   m e n s a j e s
ID mens. Gv Número Texto mensaje
*RNM0508 00      1 Encontrada la directiva del compilador /COPY.
                  Causa . . . . . : Para que este fuente RPG IV pueda compilar
                                correctamente, asegúrese de que todos los miembros fuente /COPY
                                incluidos en este miembro fuente se hayan convertido también a
                                RPG IV.
                  Recuperación . . : Asegúrese de que todos los miembros fuente
                                /COPY se hayan convertido antes de compilar en RPG IV. En algunos
                                casos, pueden producirse problemas al intentar convertir y
                                compilar miembros fuente que utilizan la directiva del compilador
                                /COPY. Si se produce esta situación, especifique *YES para el
                                parámetro EXPCPY en el mandato CVTRPGSRC para ampliar el miembro(s)
                                /COPY dentro el fuente convertido. Para obtener más información, consulte
                                ILE RPG para AS/400 Programmers Guide.
*RNM0511 00      1 Encontrado el código de operación CALL.
                  Causa . . . . . : Se han identificado las especificaciones de RPG
                                que contienen códigos de operación CALL porque el usuario puede
                                desear llevar a cabo lo siguiente:
                                -- cambiar el código de la operación CALL a CALLB para
                                beneficiarse del enlace estático
                                -- convertir todos los programas de una aplicación a RPG IV.
                  Recuperación . . : Convierta el código de la operación CALL
                                a CALLB si desea beneficiarse del enlace estático,
                                o convierta el programa llamado a RPG IV si desea
                                convertir todos los programas de una aplicación.
*RNM0506 30      1 El código de operación FREE no está soportado en RPG IV.
                  Causa . . . . . : El programa RPG III o RPG/400 contiene
                                el código de operación FREE, que no está soportado en RPG IV.
                  Recuperación . . : Elimine la operación FREE y sustitúyala por
                                código alternativo para que la lógica de programación no resulte
                                afectada antes de compilar el fuente convertido.

* * * * * F I N   D E   R E S U M E N   D E   M E N S A J E S   * * * * *
```

Figura 215. Resumen de mensajes del informe de conversión de ejemplo

El resumen final del listado proporciona estadísticas sobre los mensajes y registros. En las anotaciones de trabajo también se coloca un mensaje de estado final. La Figura 216 en la página 447 muestra la sección de mensajes para la conversión de ejemplo.

R e s u m e n f i n a l		
Total de mensajes:		
Información (00)	:	2
Aviso (10)	:	0
Error grave (30+)	:	1

Total	:	3
Totales de fuente:		
Registros originales leídos	:	3
Registros convertidos escritos	:	4
Mensaje de mayor gravedad emitido :	:	30
***** FIN DE RESUMEN FINAL *****		
***** FIN DE CONVERSIÓN *****		

Figura 216. Resumen final del informe de conversión de ejemplo

Utilización del archivo de anotaciones cronológicas

Examinando el archivo de anotaciones cronológicas, puede observar el resultado de las conversiones. El archivo de anotaciones cronológicas se actualiza después de cada operación. Hace un seguimiento de:

- Los miembros fuente y los nombres de sus bibliotecas
- Los nombres de archivos fuente convertidos y los nombres de sus bibliotecas
- Se ha encontrado un error de la gravedad más alta

Por ejemplo, si no se encuentra ningún error, el estado de la conversión se establece en 0. Si se encuentran errores graves, el estado se establece en 30.

Si intenta convertir un miembro con un tipo de miembro no soportado o un miembro que ya exista, entonces la conversión no se efectuará ya que esto es un error grave (gravedad 40 o superior). Se añadirá un registro al archivo de anotaciones cronológicas cuyo estado de conversión estará establecido en 40. TOFILE, TOMBR y TO LIBRARY se establecerán en blanco para indicar que no se ha generado TOMBR (ya que no se ha realizado la conversión).

El archivo de anotaciones cronológicas es un archivo de base de datos físico, descrito externamente. Se proporciona un "modelo" de este archivo en la biblioteca QRPGL del archivo QARNCVTLG. Tiene un formato de registro llamado QRNCVTLG. Todos los nombres de campo tienen una longitud de seis caracteres y siguen el convenio de denominación LGxxxx, donde xxxx describe los campos. La Figura 217 en la página 448 muestra las DDS para este archivo.

Utilice el siguiente mandato CPYF para crear una copia de este modelo en su propia biblioteca, mencionada aquí como MILIB. Es posible que desee poner al archivo de anotaciones cronológicas el nombre QRNCVTLG, ya que este es el nombre por omisión del archivo de anotaciones cronológicas que se utiliza en la ayuda para la conversión.

```
CPYF FROMFILE(QRPGL/QARNCVTLG) TOFILE(MILIB/QRNCVTLG)
      CRTFILE(*YES)
```

Resolución de problemas de conversión

A	R	QRNCVTFM		
A		LGCENT	1A	COLHDG('CVT' 'CENT')
A				TEXT('Siglo conversión: 0-20th 1-+
A				21st')
A		LGDATE	6A	COLHDG('CVT' 'FECHA')
A				TEXT('Fecha conversión: formato Y+
A				YMMDD')
A		LGTIME	6A	COLHDG('CVT' 'HORA')
A				TEXT('Hora conversión: formato H+
A				HMMSS')
A		LGSYST	8A	COLHDG('CVT' 'SIST')
A				TEXT('Nombre del sistema en el que +
A				se ejecuta la conversión')
A		LGUSER	10A	COLHDG('CVT' 'USUARIO')
A				TEXT('Nombre del perfil de usuario +
A				en que se ejecuta la conversión')
A		LGFRFL	10A	COLHDG('DESDE' 'ARCHIVO')
A				TEXT('Desde archivo')
A		LGFRLB	10A	COLHDG('DESDE' 'BIB')
A				TEXT('Desde biblioteca')
A		LGFRMR	10A	COLHDG('DESDE' 'MIEMB')
A				TEXT('Desde miembro')
A		LGFRMT	10A	COLHDG('FMBR' 'TIPO')
A				TEXT('Desde tipo de miembro')
A		LGTOFL	10A	COLHDG('HASTA' 'ARCHIVO')
A				TEXT('Hasta archivo')
A		LGTOLB	10A	COLHDG('HASTA' 'BIB')
A				TEXT('Hasta biblioteca')
A		LGTOMR	10A	COLHDG('HASTA' 'MIEMB')
A				TEXT('Hasta miembro')
A		LGTOMT	10A	COLHDG('TMBR' 'TIPE')
A				TEXT('Hasta tipo de miembro')
A		LGLGFL	10A	COLHDG('ANOT' 'ARCHIVO')
A				TEXT('Archivo de anotaciones')
A		LGLGLB	10A	COLHDG('ANOT' 'BIB')
A				TEXT('Biblioteca de anotaciones')
A		LGLGMR	10A	COLHDG('ANOT' 'MIEMB')
A				TEXT('Miembro de anotaciones')
A		LGCEXP	1A	COLHDG('COP' 'EXP')
A				TEXT('Copiar miembro ampliado: Y=Sí, +
A				N=No')
A		LGERRL	1A	COLHDG('CONV' 'INF')
A				TEXT('Informe conversión impreso: Y=+
A				Sí, N=No')
A		LGSECL	1A	COLHDG('SEG' 'NIV')
A				TEXT('Texto segundo nivel impreso: Y=+
A				Sí, N=No')
A		LGINSR	1A	COLHDG('INTR' 'PLANT')
A				TEXT('Plant. introducida: Y=Sí, N=N+
A				o')
A		LGSTAT	2A	COLHDG('CONV' 'EST')
A				TEXT('Estado de conversión')
A		LGMRDS	50A	COLHDG('MIEMB' 'DESC')
A				TEXT('Descripción de miembro')

Figura 217. DDS para el modelo de archivo de anotaciones cronológicas QARNCVTLG de la biblioteca QRPGL

Resolución de problemas de conversión

Los problemas de conversión pueden deberse a una o varias de las razones siguientes:

- El fuente RPG III presenta errores de compilación
- RPG IV no soporta ciertas características del lenguaje RPG III

Resolución de problemas de conversión

- Existen una o varias directivas del compilador /COPY en el fuente RPG III
- La utilización de estructuras de datos descritos externamente
- Diferencias de comportamiento de tiempo de ejecución entre OPM e ILE

Las secciones siguientes explican cada una de estas áreas.

Errores de compilación en código RPG III existente

La ayuda para la conversión da por supuesto que se está intentando convertir un programa RPG III válido, es decir, un programa sin errores de compilación. Si no ocurre esto, entonces pueden producirse resultados no previstos durante la conversión. Si cree que su programa contiene errores de compilación, compílelo primero utilizando el compilador de RPG III y corrija los errores antes de realizar la conversión.

Características de RPG III no soportadas

Algunas de las características del lenguaje RPG III *no* reciben soporte en RPG IV. Las más importantes son:

- La función del generador automático de informes
- El código de operación FREE
- El código de operación DEBUG

Dado que no se da soporte al generador automático de informes, la ayuda para la conversión ampliará automáticamente estos programas (es decir, llamará al generador automático de informes) antes de realizar la conversión, si el tipo es RPT o RPT38.

Debe sustituir los códigos de operación FREE o DEBUG por una lógica equivalente antes o después de la conversión.

Si especifica la opción CVTRPT(*YES) en el mandato CVTRPGSRC, recibirá un informe sobre la conversión que identificará la mayor parte de estos tipos de problemas.

Para obtener más información sobre cómo convertir miembros del generador automático de informes, consulte el apartado “Conversión de miembros fuente del generador automático de informes” en la página 440. Para obtener más información sobre las diferencias entre RPG III y RPG IV, consulte el “Apéndice A. Diferencias de comportamiento entre OPM RPG/400 y ILE RPG para AS/400” en la página 423.

Utilización de la directiva del compilador /COPY

En algunos casos los errores no se encontrarán hasta que compile el fuente RPG IV convertido. Los errores de conversión de este tipo suelen estar relacionados con la utilización de la directiva del compilador /COPY. Estos errores pertenecen a dos categorías: problemas de fusión y problemas sensibles al contexto. A continuación explicamos por qué se producen estos problemas y cómo se pueden solucionar.

Problemas de fusión

Debido a las diferencias entre los lenguajes RPG III y RPG IV, la ayuda para la conversión debe cambiar el orden de ciertas sentencias del fuente. Un ejemplo de este cambio de orden se muestra en el apartado “Ejemplo de conversión de fuente” en la página 442 para el miembro fuente PRUEBA1 de RPG III. Si se compara la ubicación de la estructura de datos DS1 en la Figura 211 en la página 442 y en la

Resolución de problemas de conversión

Figura 212 en la página 443, observará que la estructura de datos DS1 se ha movido de modo que precede al formato de registro FORMAT1.

Ahora suponga que el miembro RPG III PRUEBA1 se ha dividido en dos miembros, PRUEBA2 y COPYDS1, donde la estructura de datos DS1 y la constante con nombre CONST1 están en un miembro de copia COPYDS1. Este miembro de copia se incluye en el fuente PRUEBA2. La Figura 218 y la Figura 219 muestran el fuente para PRUEBA2 y COPYDS1 respectivamente.

```

H                                     TSTPGM
FFILE1  IF  E                                     COMM1
FQSYSPRT 0  F      132      0F      LPRINTER
LQSYSPRT 60FL 560L
E                                     ARR1  3  3  1      COMM2
E                                     ARR2  3  3  1
IFORMAT1
I                                     NOMANT      NOMBRE
/COPY COPYDS1
C      ARR1,3      DSPLY
C      READ FORMAT1      01
C      NAME      DSPLY
C      SETON      LR
C      EXCPTOUTPUT
OQSYSPRT E  01      OUTPUT
O      ARR2,3      10
**
123
**
456
```

Figura 218. Fuente RPG III para PRUEBA2

```

I* COMENTARIO SOBRE ESTRUCTURA DE DATOS
IDS1      DS
I
I* COMENTARIO DE CONSTANTE CON NOMBRE
I      'XYZ'      C      CONST1      COMM3
I      4  6  ARR1
```

Figura 219. Fuente RPG III para COPYDS1

En esta situación, la ayuda para la conversión convertiría correctamente el miembro PRUEBA2 y el miembro de copia COPYDS1. Sin embargo, cuando se incluya el miembro de copia durante la compilación, se insertará por debajo de FORMAT1 porque ahí es donde se encuentra situada la directiva /COPY. Como resultado, todas las líneas fuente del miembro de copia COPYDS1 obtendrán el error de " el registro fuente está fuera de secuencia". En RPG IV, las especificaciones de definición deben preceder a las especificaciones de entrada.

Tenga en cuenta que la ayuda para la conversión no ha podido mover la directiva /COPY por encima del formato FORMAT1 porque se desconoce el contenido del miembro /COPY.

Existen dos métodos para corregir este tipo de problema:

1. Utilizar la opción EXPCPY(*YES) del mandato CVTRPGSRC para incluir todos los miembros /COPY en el miembro fuente RPG IV convertido.

Resolución de problemas de conversión

Este método es fácil y funcionará la mayoría de las veces. Sin embargo, incluir los miembros /COPY en cada miembro fuente reduce la capacidad de mantenimiento de su aplicación.

2. Corregir manualmente el código después de la conversión, utilizando la información del listado de compilador de ILE RPG y la publicación *ILE RPG Reference*.

Otros ejemplos de este tipo de problema incluyen:

- Especificaciones de línea y archivos de direcciones de registros
En RPG III la especificación de contador de líneas y el archivo de direcciones de registro de la especificación de extensión se cambian por las palabras clave (RAFDATA, FORMLEN y FORMOFL) de la especificación de descripción de archivo. Si el contenido de un miembro /COPY contiene únicamente las especificaciones de contador de líneas y/o el archivo de direcciones de registro de la especificación de extensión, pero no la especificación de descripción de archivo correspondiente, la ayuda para la conversión no sabrá dónde insertar las palabras clave.
- Matrices de especificación de extensión y subcampos de estructura de datos
Como mencionamos en el apartado “Ejemplo de conversión de fuente” en la página 442, en RPG IV no está permitido definir una matriz autónoma y un subcampo de estructura de datos con el mismo nombre. Por lo tanto, como se muestra en el ejemplo PRUEBA1 (Figura 212 en la página 443), la ayuda para la conversión debe fusionar estas dos definiciones. Sin embargo, si la matriz y el subcampo de estructura de datos no están en el mismo miembro fuente (es decir, uno o los dos se encuentran en un miembro /COPY) esta fusión no puede efectuarse y se generará un error de tiempo de compilación.
- Matrices de tiempo de compilación fusionadas y registros de datos de tiempo de compilación (**)
Como se muestra en el ejemplo PRUEBA1 (Figura 212 en la página 443), si se fusionan las matrices de tiempo de compilación con las definiciones de subcampo de estructura de datos, la carga de datos de las matrices puede resultar afectada. Para resolver este problema, los datos de las matrices de tiempo de compilación se cambiarán al nuevo formato **CTDATA, si como mínimo se fusiona una matriz de tiempo de compilación. Sin embargo, si las matrices y los datos no residen en el mismo archivo fuente (es decir, uno o los dos se encuentran en un miembro COPY), la denominación de registros de datos de tiempo de compilación con el formato **CTDATA no podrá ejecutarse correctamente.

Problemas sensibles al contexto

En RPG III, hay ocasiones en que es imposible determinar el tipo de especificaciones de un miembro /COPY sin el contexto de las especificaciones del miembro fuente primario que lo rodean. Este problema se produce en dos casos:

- En subcampos de estructura de datos o en campos de archivos descritos por programa

Resolución de problemas de conversión

```
I* Si el miembro fuente de RPG III contiene sólo las sentencias de fuente
I* que describen los campos FIELD1 y FIELD2 que aparecen a continuación, la
I* ayuda para la conversión no sabrá bien cómo convertirlas. Estas
I* sentencias pueden ser campos de estructura de datos (que se convierten a
I* especificaciones de definición) o campos de archivos
I* descritos por programa (que se convierten en
I* especificaciones de entrada).
I                                1    3 FIELD1
I                                4    6 FIELD2
```

Figura 220. Archivo /COPY de RPG III con sólo campos de entrada

- Al redenominar un campo de estructura de datos descritos externamente o un campo de archivo descrito externamente

```
I* Si el miembro fuente de RPG III contiene sólo la sentencia fuente
I* que describe el campo CHAR que aparece a continuación, la ayuda para la
I* conversión no sabrá bien cómo convertirla. Esta sentencia puede ser una
I* redenominación de un campo de estructura de datos descritos externamente
I* (que se convierte en una especificación de definición) o
I* una redenominación de campo de un archivo descrito
I* externamente (que se convierte en una especificación de
I* entrada).
I          CHARACTER                      CARC
```

Figura 221. Fuente RPG III con un campo redenominado

En las dos instancias anteriores, se presupone una estructura de datos y se generan las especificaciones de definición. También se genera un bloque de comentarios que contiene el código de la especificación de entrada. Por ejemplo, la ayuda para la conversión convertirá el fuente de la Figura 220 al código que se muestra en la Figura 222. Si se necesita el código de la especificación de entrada, suprima las especificaciones de definición y borre los asteriscos de las especificaciones de entrada correspondientes.

```
D* Si el miembro fuente en RPG III contiene sólo los campos
D* de descripción de sentencias fuente FIELD1 y FIELD2 que
D* aparecen a continuación, la ayuda para la conversión no está
D* segura de cómo convertirlos. Estas sentencias pueden ser
D* campos de estructura de datos (que se convierten en
D* especificaciones de definición) o campos de archivo
D* descritos por programa (que se convierten en especificaciones
D* entrada).
D FIELD1                1    3
D FIELD2                4    6
I*                      1    3 FIELD1
I*                      4    6 FIELD2
```

Figura 222. Fuente RPG IV después de convertir fuente con sólo los campos de entrada

Recuerde que existen dos manera de corregir estos tipos de problemas. Utilice la opción EXPCPY(*YES) del mandato CVTRPGSRC, o corrija manualmente el código después de la conversión.

Utilización de estructuras de datos descritos externamente

Existen dos problemas que es posible que deba arreglar manualmente aunque haya especificado la opción EXPCPY(*YES) en el mandato CVTRPGSRC.

- La fusión de una matriz con un subcampo DS descrito externamente

Resolución de problemas de conversión

- La redenominación e inicialización de un subcampo DS descrito externamente

Estos problemas están relacionados con la utilización de estructuras de datos descritos externamente.

Dado que estos problemas causarán errores de tiempo de compilación, puede utilizar la información del listado del compilador de ILE RPG y el manual *ILE RPG Reference* para corregirlos.

Fusión de una matriz con un subcampo DS descrito externamente

Como se ha mencionado anteriormente, en RPG IV no se puede definir una matriz autónoma y un subcampo de estructura de datos con el mismo nombre. En general, la ayuda para la conversión fusionará estas dos definiciones. Sin embargo, si el subcampo está en una estructura de datos descritos externamente, esta fusión no se manejará y se le solicitará que corrija manualmente el miembro fuente convertido.

Por ejemplo, el campo ARRAY en la Figura 223 se incluye dos veces en la Figura 224. Se incluye una vez como una matriz autónoma y una vez en la estructura de datos descritos externamente EXTREC. Cuando se convierte, el fuente RPG IV generado se muestra en la Figura 225. Este código no se compilará ya que ARRAY está definido dos veces. Para corregir este problema, suprima la matriz autónoma y añada un subcampo con las palabras clave para la estructura de datos DSONE, como se muestra en la Figura 226 .

A	R RECORD		
A	CHARACTER	10	
A	ARRAY	10	

Figura 223. DDS para la estructura de datos externos

E		ARRAY	10	1	
IDSONE	E DSEXTREC				
C	CHAR	DSPLY			
C		SETON			LR

Figura 224. Fuente RPG III que utiliza una estructura de datos externos con una matriz

D ARRAY	S	1	DIM(10)	
D DSONE	E DS		EXTNAME(EXTREC)	
C CHAR				
C				LR

Figura 225. Fuente RPG IV con dos definiciones para la matriz

D DSONE	E DS		EXTNAME(EXTREC)	
D ARRAY	E		DIM(10)	
C CHAR				
C				LR

Figura 226. Fuente RPG IV corregido con una sola definición para la matriz

Resolución de problemas de conversión

Redenominación e inicialización de un subcampo DS descrito externamente

En RPG III, al redenominar e inicializar un campo en una estructura de datos descrita externamente, tenía que utilizar dos líneas de fuente, como se muestra para el campo CHAR en la Figura 227. El fuente convertido también contiene dos líneas de fuente, como se muestra en la Figura 228. Esta utilización de dos líneas de fuente para un campo ocasionará un error de tiempo de compilación, ya que el campo CHAR se ha definido dos veces. Para corregir este código, deberá combinar las palabras clave del campo CHAR en una sola línea, como se muestra en la Figura 229, donde los campos clave INZ y EXTFLD se han combinado y sólo se muestra una instancia en el campo CHAR.

IDSONE	E DSEXTREC	
I	CHARACTER	CHAR
I I	'XYZ'	CHAR
C	CHAR	DSPLY
C	SETON	LR

Figura 227. Fuente RPG III con subcampo externo redenominado e inicializado

D DSONE	E DS	EXTNAME(EXTREC)
D CHAR	E	EXTFLD(CHARACTER)
D CHAR	E	INZ('XYZ')
C CHAR	DSPLY	
C	SETON	LR

Figura 228. Fuente RPG IV con dos definiciones para el subcampo redenominado

D DSONE	E DS	EXTNAME(EXTREC)
D CHAR	E	EXTFLD(CHARACTER) INZ('XYZ')
C CHAR	DSPLY	
C	SETON	LR

Figura 229. Fuente RPG IV corregido con una sola definición

Diferencias de tiempo de ejecución

Si tiene matrices anteriores a la ejecución que se solapan en estructuras de datos, el orden de carga de estas matrices durante la ejecución puede ser diferente en RPG III y en RPG IV. Esta diferencia de orden puede ocasionar que los datos de la sección solapada difieran. El orden en el que se cargan las matrices es el orden en el que se encuentran en el fuente. Este orden puede haberse cambiado cuando se han fusionando las matrices con los subcampos durante la conversión.

En general, debe evitar situaciones en las que una aplicación conste de programas OPM e ILE que estén divididos a través del grupo de activación por omisión de OMP y de un grupo de activación con nombre. Al producir una división entre estos dos grupos de activación, se mezcla el comportamiento de OPM con el comportamiento de ILE y el resultado puede ser difícil de prever. Consulte el “Capítulo 3. Estrategias de creación de programas” en la página 25 o la publicación *ILE Concepts*, para obtener más información.

Apéndice C. Los mandatos para crear

Esta sección proporciona información acerca de:

- Utilización de los mandatos de CL
- Diagrama de sintaxis y descripción de CRTBNDRPG
- Diagrama de sintaxis y descripción de CRTRPGMOD

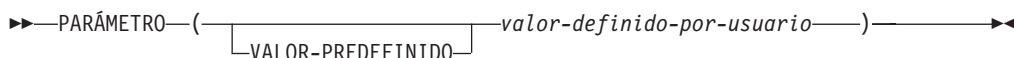
Para obtener información sobre los mandatos Crear programa y Crear programa de servicio, consulte la sección *CL y las API* de la categoría *Programación* en **iSeries 400 Information Center** en el sitio web <http://www.ibm.com/eserver/iseriess/infocenter>.

Utilización de los mandatos de CL

Los **mandatos**, **parámetros** y **palabras clave del lenguaje de control (CL)** se pueden entrar tanto en mayúsculas como en minúsculas. En el diagrama de sintaxis se muestran en mayúsculas (por ejemplo, PARÁMETRO, VALOR-PREDEFINIDO). Las variables aparecen en letras minúsculas en cursiva (por ejemplo, *valor-definido-de-usuario*). Las variables son nombres o valores definidos por el usuario.

Cómo interpretar diagramas de sintaxis

Los diagramas de sintaxis de este manual utilizan las convenciones siguientes:



Lea el diagrama de sintaxis de izquierda a derecha y de arriba a abajo, siguiendo la ruta de la línea.

El símbolo ▶▶ indica el principio del diagrama de sintaxis.

El símbolo →◀ indica el final del diagrama de sintaxis.

El símbolo → indica que la sintaxis de la sentencia continúa en la línea siguiente.

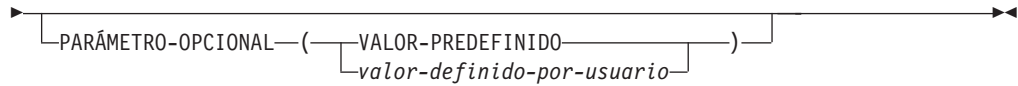
El símbolo ▶— indica la continuación de una sentencia desde la línea anterior.

El símbolo —(—)— indica que el parámetro o valor debe entrarse en el paréntesis.

Los **parámetros necesarios** aparecen en la línea base y deben entrarse. Los **parámetros opcionales** aparecen por debajo de la línea de base y no es necesario entrarlos. En la muestra siguiente, en necesario entrar el PARÁMETRO-NECESARIO y un valor para el mismo, pero no es necesario entrar el PARÁMETRO-OPCIONAL o un valor para el mismo.



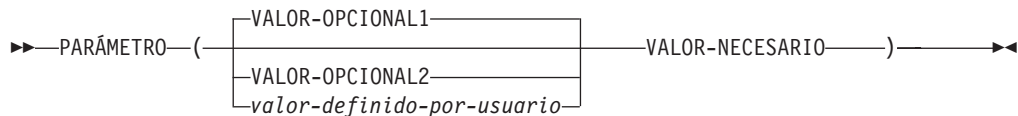
Lectura de diagramas de sintaxis



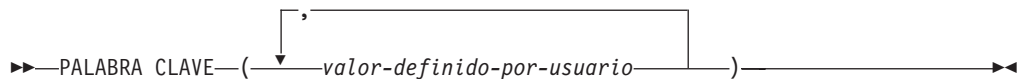
Los **valores por omisión** aparecen por encima de la línea de base y no es necesario entrarlos. Se utilizan cuando no se especifica un parámetro. En la muestra siguiente, se puede entrar el VALOR-POR-OMISIÓN, OTRO-VALOR-PREDEFINIDO, o nada. Si no se entra nada, se presupone el VALOR-POR-OMISIÓN.



Los **valores opcionales** se indican por medio de una línea en blanco. La línea en blanco indica que no es necesario entrar un valor del primer grupo (VALOR-OPCIONAL1, VALOR-OPCIONAL2, *valor-definido-por-usuario*). Por ejemplo, basándose en la sintaxis siguiente, el usuario podría entrar PALABRA CLAVE(VALOR-NECESARIO).



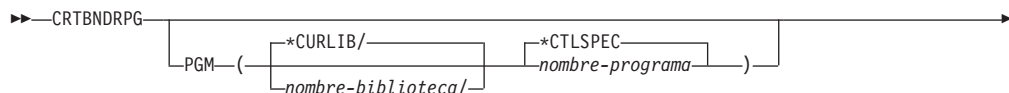
Los **valores repetidos** se pueden especificar para algunos parámetros. La coma (,) de la muestra siguiente indica que cada *valor-definido-por-usuario* debe separarse con una coma.



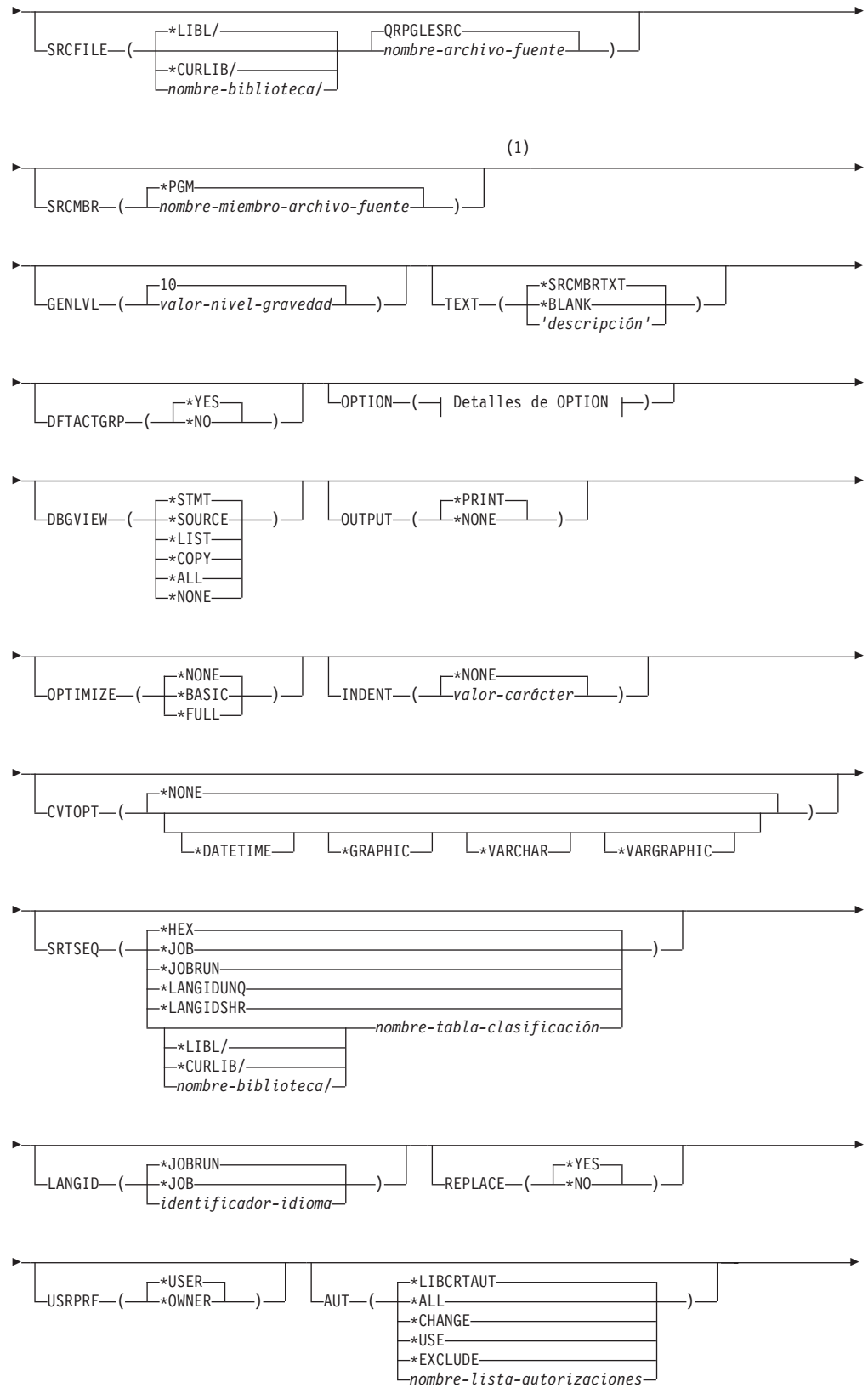
Mandato CRTBNDRPG

El mandato Crear RPG Enlazado (CRTBNDRPG) realiza las tareas combinadas de los mandatos Crear Módulo RPG (CRTRPGMOD) y Crear Programa (CRTPGM) creando un objeto de módulo temporal desde el código fuente y a continuación creando el programa objeto. Una vez que se ha creado el programa objeto, CRTBNDRPG suprimirá el objeto de módulo que haya creado. El diagrama de sintaxis completo para el mandato CRTBNDRPG se muestra a continuación.

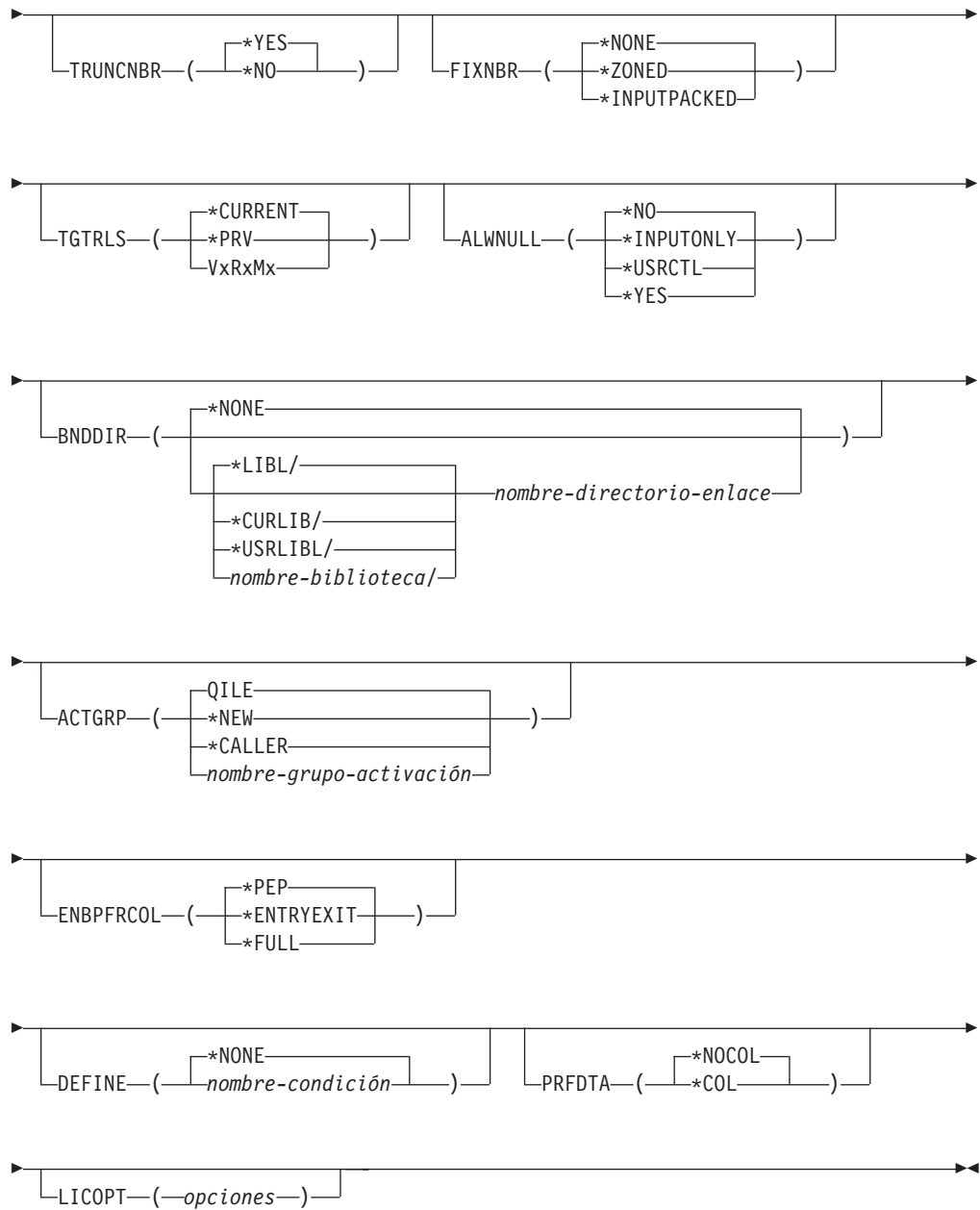
Job: B,I Pgm: B,I REXX: B,I Exec



Mandato CRTBNDRPG



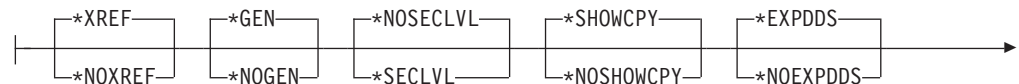
Mandato CRTBNDRPG

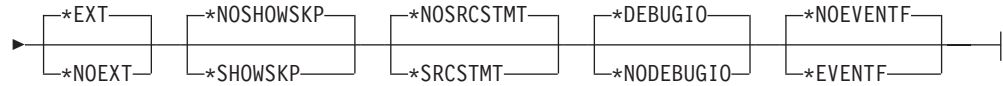


Notas:

- 1 Todos los parámetros anteriores a este punto pueden especificarse por posición

Detalles de OPCIÓN:





Descripción del mandato CRTBNDRPG

Los parámetros, palabras clave y variables del mandato CRTBNDRPG se listan a continuación. La misma información está disponible en línea. Entre el nombre del mandato en una línea de mandatos, pulse PF4 (Solicitud) y a continuación pulse PF1 (Ayuda) para cualquier parámetro sobre el que desee información.

PGM

Especifica el nombre del programa y el nombre de biblioteca del programa objeto (*PGM) que se está creando. El nombre del programa y el nombre de la biblioteca tienen que estar de acuerdo con las convenciones de denominación de iSeries. Si no se especifica ninguna biblioteca, el programa creado se almacena en la biblioteca actual.

*CTLSPEC

El nombre para el programa compilado se toma del nombre especificado en la palabra clave DFTNAME de la especificación de control. Si no se especifica el nombre de programa en la especificación de control, y el miembro fuente procede de un archivo de base de datos, se utilizará como nombre de programa el nombre de miembro, especificado por el parámetro SRCMBR. Si el fuente no procede de un archivo de base de datos, el nombre de programa tomará por omisión RPGPGM.

nombre-programa

Entrar el nombre del programa objeto.

*CURLIB

El programa objeto creado se almacena en la biblioteca actual. Si no se ha especificado una biblioteca actual, se utilizará QGPL.

nombre-biblioteca

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

SRCFILE

Especifica el nombre del archivo fuente que contiene el miembro fuente de ILE RPG que se va a compilar y la biblioteca donde está ubicado el archivo fuente. La longitud recomendada del archivo físico fuente es de 112 caracteres: 12 para el número y la fecha de secuencia, 80 para el código y 20 para los comentarios. Este es el fragmento máximo de fuente que se muestra en el listado de compilador.

QRPGLESRC

El archivo fuente por omisión QRPGLESRC contiene el miembro fuente de ILE RPG que se va a compilar.

nombre-archivo-fuente

Entrar el nombre del archivo fuente que contenga el miembro fuente de ILE RPG que se va a compilar.

*LIBL

El sistema busca en la lista de bibliotecas para encontrar la biblioteca donde está almacenado el archivo fuente. Este es el valor por omisión.

Mandato CRTBNDRPG

***CURLIB**

La biblioteca actual se utiliza para buscar el archivo fuente. Si no se ha especificado una biblioteca actual, se utilizará QGPL.

nombre-biblioteca

Entrar el nombre de la biblioteca donde está almacenado el archivo fuente.

SRCMBR

Especifica el nombre del miembro del archivo fuente que contiene el programa fuente de ILE RPG que se va a compilar.

***PGM**

Se utiliza el nombre especificado por el parámetro PGM como nombre de miembro del archivo fuente. El programa objeto compilado tendrá el mismo nombre que el miembro del archivo fuente. Si el parámetro PGM no especifica ningún nombre de programa, el mandato utilizará el primer miembro creado en el archivo fuente, o añadido al mismo, como nombre del miembro fuente.

nombre-miembro-archivo-fuente

Entrar el nombre del miembro que contenga el programa fuente de ILE RPG.

GENLVL

Controla la creación del programa objeto. El programa objeto se crea si todos los errores encontrados durante el proceso de compilación poseen un nivel de gravedad menor que, o igual al del nivel de gravedad generacional especificado.

10 No se generará un programa objeto si se tienen mensajes con un nivel de gravedad superior a 10.

valor-nivel-gravedad

Entrar un número del 0 al 20, ambos inclusive. No se generará el programa objeto para los errores de gravedad superior a 20.

TEXT

Permite entrar texto que describa brevemente el programa y su función. El texto aparece cuando se visualiza la información del programa.

***SRCMBRTXT**

Se utiliza el texto del miembro fuente.

***BLANK**

No aparece texto.

'descripción'

Entrar el texto que describa brevemente la función de las especificaciones del fuente. El texto puede tener un máximo de 50 caracteres y debe estar entre apóstrofes. Los apóstrofes no son parte de la serie de 50 caracteres. Los apóstrofes no son necesarios si se está entrando el texto en la pantalla de solicitud.

DFTACTGRP

Especifica si el programa creado tiene la finalidad ejecutarse siempre en el grupo de activación por omisión.

***YES**

Cuando se llama a este programa, siempre se ejecutará en el grupo de activación por omisión. El grupo de activación por omisión es el grupo de activación donde se ejecutan todos los programas del modelo de programa original (OPM).

Especificar DFTACTGRP(*YES) permite a los programas ILE RPG comportarse como programas OPM en las áreas de ámbito de alteración temporal, ámbito abierto y RCLRSC.

El enlace estático de ILE no está disponible cuando se crea un programa con DFTACTGRP(*YES). Esto significa que no se puede utilizar el parámetro BNDDIR ni el ACTGRP al crear este programa. Además, cualquier operación de llamada en el fuente debe llamar a un programa y no a un procedimiento.

DFTACTGRP(*YES) es útil al intentar mover una aplicación de programa a programa hasta ILE RPG.

***NO**

El programa se asocia con el grupo de activación especificado por el parámetro ACTGRP. El enlace estático está permitido cuando se especifica *NO.

Si se especifica ACTGRP(*CALLER) y un programa que se esté ejecutando en el grupo de activación por omisión llama a este programa, éste se comportará según la semántica de ILE en las áreas de compartimento de archivo, ámbito de archivo y RCLRSC.

DFTACTGRP(*NO) es útil cuando se tiene intención de sacar provecho de los conceptos de ILE, por ejemplo, ejecutar en un grupo de activación con nombre, o enlazar con un programa de servicio.

OPTION

Especifica las opciones que se deben utilizar cuando se ha compilado el miembro fuente. Se puede especificar cualquiera de las opciones, o todas ellas, en cualquier orden. Separe las opciones con uno o varios espacios en blanco. Si una opción se ha especificado varias veces, se utilizará la última de ellas.

***XREF**

Genera un listado de referencias cruzadas (cuando es apropiado) para el miembro fuente.

***NOXREF**

No se genera un listado de referencias cruzadas.

***GEN**

Se crea un programa objeto si el nivel de gravedad más alto que ha devuelto el compilador no sobrepasa la gravedad especificada en la opción GENLVL.

***NOGEN**

No se crea un programa objeto.

***NOSECLVL**

No se imprime un mensaje de texto de segundo nivel en la línea siguiente al mensaje de texto de primer nivel.

***SECLVL**

Se imprime un mensaje de texto de segundo nivel en la línea siguiente al mensaje de texto de primer nivel de la sección de resumen de mensajes.

***SHOWCPY**

Se muestran los registros fuente de los miembros incluidos por la directiva del compilador /COPY.

***NOSHOWCPY**

No se muestran los registros fuente de los miembros incluidos por la directiva del compilador /COPY.

Mandato CRTBNDRPG

*EXPDDS

Se muestra la ampliación de los archivos descritos externamente en la información de campo del listado y de la clave de visualización.

*NOEXPDDS

No se muestra la ampliación de los archivos descritos externamente en la información de campo del listado o de la clave de visualización.

*EXT

Se muestra la lista de procedimientos externos y de campos a los que se hecho referencia durante la compilación en el listado.

*NOEXT

No se muestra la lista de procedimientos externos y de campos a los que se ha hecho referencia durante la compilación en el listado.

*NOSHOWSKP

No se muestran las sentencias ignoradas de la parte fuente del listado. El compilador ignora las sentencias como resultado de las directivas /IF, /ELSEIF y /ELSE.

*SHOWSKP

Se muestran todas la sentencias de la parte fuente del listado, independientemente de si el compilador se las ha saltado.

*NOSRCSTMT

Los números de línea del listado se asignan secuencialmente; estos números se utilizan al depurar utilizando números de sentencia. Los números de línea se muestran en la columna situada más a la izquierda del listado. Los números de secuencia de los ID y SEU del fuente se muestran en las dos columnas situadas más a la derecha del listado.

*SRCSTMT

Los números de sentencia para la depuración se generan utilizando los números de secuencia de SEU y los ID de fuente de la manera siguiente:

$$\text{Número_sentencia} = \text{ID_fuente} * 1000000 + \text{número_secuencia_SEU_fuente}$$

Los números de secuencia de SEU se muestran en la columna situada más a la izquierda del listado. Los números de sentencia se muestran en la columna situada más a la derecha del listado; estos números se utilizan al depurar utilizando números de sentencia.

Nota: Cuando se especifica OPTION(*SRCSTMT), todos los números de secuencia de los archivos fuente deben contener valores numéricos válidos. Si hay números de secuencia duplicados en el mismo archivo fuente, el depurador podrá comportarse de manera imprevisible y los números de sentencia para los mensajes de diagnóstico o las entradas de referencias cruzadas podrán no tener sentido.

*DEBUGIO

Se generan puntos de interrupción para todas las especificaciones de entrada y salida.

*NODEBUGIO

No se generan puntos de interrupción para las especificaciones de entrada y salida.

*NOEVENTF

No se crea un archivo de eventos para que lo utilice CoOperative Development Environment/400 (CODE/400). CODE/400 utiliza este

archivo para proporcionar información de retorno de errores integrada con el editor de CODE/400. Un archivo de eventos se crea normalmente al crear un módulo o programa desde dentro de CODE/400.

***EVENTF**

Se crea un archivo de eventos para que lo utilice CoOperative Development Environment/400 (CODE/400). El archivo de eventos se crea como un miembro en el archivo EVFEVENT de la biblioteca donde se almacena el módulo creado o programa objeto. Si el archivo EVFEVENT no existe, se creará automáticamente. El nombre del miembro del archivo de eventos es el mismo que el del objeto que se está creando.

CODE/400 utiliza este archivo para proporcionar información de retorno de errores integrada con el editor de CODE/400. Un archivo de eventos se crea normalmente al crear un módulo o programa desde dentro de CODE/400.

DBGVIEW

Especifica qué nivel de depuración hay disponible para el programa objeto compilado, y qué vistas del fuente hay disponibles para la depuración a nivel de fuente.

***STMT**

Permite depurar el programa objeto utilizando los números de línea o los números de sentencia del listado de compilador. Los números de línea se muestran en la columna situada más a la izquierda de la sección fuente del listado de compilador cuando se especifica OPTION(*NOSRCSTMT). Los números de sentencia se muestran en la columna situada más a la derecha de la sección fuente del listado de compilador cuando se especifica OPTION(*SRCSTMT).

***SOURCE**

Genera la vista del fuente para depurar el programa objeto compilado. Esta vista no está disponible si el miembro fuente raíz es un archivo DDM. Además, si se realizan cambios en cualquier miembro fuente después de la compilación y antes de intentar depurar el programa, es posible que las vistas de esos miembros fuente no se puedan aprovechar.

***LIST**

Genera la vista del listado para depurar el programa objeto compilado. La información contenida en la vista del listado depende de si se ha especificado *SHOWCPY, *EXPDDS y *SRCSTMT en el parámetro OPTION.

Nota: La vista del listado no mostrará ningún sangrado que pueda haberse solicitado utilizando el opción Sangrar.

***COPY**

Genera las vistas del fuente y de copia para depurar el programa objeto compilado. La vista del fuente para esta opción es la misma que se ha generado para la opción *SOURCE. La vista de copia es una vista para depurar que tiene incluidos todos los miembros fuente /COPY. Estas vistas no estarán disponibles si el miembro fuente raíz es un archivo DDM. Además, si se realizan cambios en cualquier miembro fuente después de la compilación y antes de intentar depurar el programa, es posible que las vistas de esos miembros fuente no se puedan aprovechar.

***ALL**

Genera las vistas del listado, del fuente y de copia para depurar el

Mandato CRTBNDRPG

programa objeto compilado. La información contenida en la vista del listado depende de si se especifica *SHOWCPY, *EXPDDS y *SRCSTMT en el parámetro OPTION.

***NONE**

Inhabilita todas las opciones de depuración para depurar el programa objeto compilado.

OUTPUT

Especifica si se ha generado un listado de compilador.

***PRINT**

Genera un listado de compilador que consta del fuente del programa ILE RPG y de todos los mensajes de tiempo de compilación. La información contenida en el listado depende de si se ha especificado *XREF, *SECLVL, *SHOWCPY, *EXPDDS, *EXT, *SHOWSKP y *SRCSTMT en el parámetro OPTION.

***NONE**

No se genera el listado de compilador.

OPTIMIZE

Especifica el nivel de optimización, si lo hay, del programa.

***NONE**

El código generado no se optimiza. Este es el más rápido en términos de tiempo de conversión. Permite visualizar y modificar variables mientras están en modalidad de depuración.

***BASIC**

Se realiza parte de la optimización en el código generado. Esto permite visualizar las variables de usuario pero no modificarlas, mientras el programa está en modalidad de depuración.

***FULL**

Posibilita una optimización que genera el código más eficiente. El tiempo de conversión es el más largo. En la modalidad de depuración, las variables de usuario pueden que no se modifiquen pero que se visualicen aunque los valores presentados no sean valores actuales.

INDENT

Especifica si las operaciones estructuradas deben sangrarse en el listado fuente para su mejor legibilidad. También especifica los caracteres que se utilizan para marcar las cláusulas de la operación estructurada.

Nota: Los sangrados que se soliciten aquí no se reflejarán en la vista de depuración del listado que se crea al especificar DBGVIEW(*LIST).

***NONE**

Las operaciones estructuradas no se sangrarán en el listado fuente.

valor-carácter

El listado fuente se sangra para las cláusulas de la operación estructurada. La alineación de sentencias y cláusulas se marca utilizando los caracteres que elija el usuario. Se puede elegir cualquier serie de caracteres de hasta 2 caracteres de longitud. Si desea utilizar un espacio en blanco en su serie de caracteres, deberá cerrar la serie entre comillas simples.

Nota: Es posible que el sangrado no aparezca como se esperaba, si hay errores en el programa.

CVTOPT

Especifica la manera en que el compilador de ILE RPG maneja los tipos de datos gráficos, de fecha, de hora y de indicación de la hora, así como los tipos de datos de longitud variable que se recuperan de archivos de base de datos descritos externamente.

***NONE**

Ignora los tipos de datos de base de datos de longitud variable y utiliza los tipos de datos gráficos, de indicación de la hora, de hora y de fecha del RPG nativo.

***DATETIME**

Especifica que los tipos de datos de base de datos de indicación de la hora, de hora y de fecha se van a declarar como campos de caracteres de longitud fija.

***GRAPHIC**

Especifica que los tipos de datos gráficos del juego de caracteres de doble byte (DBCS) se van a declarar como campos de caracteres de longitud fija.

***VARCHAR**

Especifica que los tipos de datos de tipo carácter de longitud variable se van a declarar como campos de caracteres de longitud fija.

***VARGRAPHIC**

Especifica que los tipos de datos gráficos del juego de caracteres de doble byte (DBCS) de longitud variable se van a declarar como campos de caracteres de longitud fija.

SRTSEQ

Especifica la tabla de secuencia de ordenación que se va a utilizar en el programa fuente ILE RPG.

***HEX**

No se utiliza ninguna tabla de secuencia de ordenación.

***JOB**

Se utiliza el valor SRTSEQ para el trabajo cuando se cree *PGM.

***JOB RUN**

Se utiliza el valor SRTSEQ para el trabajo cuando se ejecute *PGM.

***LANGIDUNQ**

Se utiliza una tabla de pesos exclusivos. Este valor especial se utiliza conjuntamente con el parámetro LANGID para determinar la tabla de secuencia de ordenación apropiada.

***LANGIDSHR**

Se utiliza una tabla de pesos compartidos. Este valor especial se utiliza conjuntamente con el parámetro LANGID para determinar la tabla de secuencia de ordenación apropiada.

nombre-tabla-clasificación

Entrar el nombre calificado de la tabla de secuencia de ordenación que se va a utilizar con el programa.

***LIBL**

El sistema busca en la lista de bibliotecas para encontrar la biblioteca donde está almacenada la tabla de secuencia de ordenación.

Mandato CRTBNDRPG

***CURLIB**

La biblioteca actual se utiliza para buscar la tabla de secuencia de ordenación. Si no se ha especificado una biblioteca actual, se utilizará QGPL.

nombre-biblioteca

Entre el nombre de la biblioteca donde está almacenada la tabla de secuencia de ordenación.

Si desea utilizar los parámetros SRTSEQ y LANGID para determinar el orden de clasificación alternativo, deberá especificar también ALTSEQ(*EXT) en la especificación de control.

LANGID

Especifica el identificador de idioma que se utilizará cuando la secuencia de ordenación sea *LANGIDUNQ y *LANGIDSHR. El parámetro LANGID se utiliza conjuntamente con el parámetro SRTSEQ para seleccionar la tabla de secuencia de ordenación.

***JOB RUN**

Se utiliza el valor LANGID asociado con el trabajo cuando se ejecute el programa RPG.

***JOB**

Se utiliza el valor LANGID asociado con el trabajo cuando se cree el programa RPG.

identificador-idioma

Utilizar el identificador del idioma especificado; por ejemplo, FRA para francés y DEU para alemán.

REPLACE

Especifica si se crea un programa nuevo cuando ya existe un programa del mismo nombre en la biblioteca especificada (o que se da por supuesta). El módulo intermedio creado durante el proceso del mandato CRTBNDRPG no está sujeto a las especificaciones de REPLACE y tienen un REPLACE(*NO) implícito contra la biblioteca QTEMP. El módulo intermedio se suprime una vez que el mandato CRTBNDRPG ha completado el proceso.

***YES**

Se crea un programa nuevo en la biblioteca especificada. El programa existente del mismo nombre en la biblioteca especificada se traslada a la biblioteca QRPLOBJ.

***NO**

No se crea un programa nuevo si un programa del mismo nombre ya existe en la biblioteca especificada. El programa existente no se sustituye, se visualiza un mensaje y se detiene la compilación.

USRPRF

Especifica el perfil de usuario que ejecutará el programa objeto creado. El perfil del propietario del programa o del usuario del programa se utiliza para ejecutar el programa y para controlar qué objetos puede utilizar el programa (incluyendo la autorización que el programa tiene para cada objeto). Este parámetro no se actualiza si el programa ya existe. Para cambiar su valor, se debe suprimir el programa y recompilar utilizando el nuevo valor (o, si existen los objetos *MODULE constitutivos, se puede elegir invocar el mandato CRTPGM).

***USER**

El programa se ejecuta bajo el perfil de usuario del usuario del programa.

***OWNER**

El programa se ejecuta bajo el perfil de usuario del usuario del programa y del propietario. El conjunto colectivo de autorización sobre objeto en ambos perfiles de usuario se utiliza para buscar objetos y acceder a ellos mientras el programa se está ejecutando. El usuario del programa será el propietario de los objetos que se creen durante la ejecución del programa.

AUT

Especifica la autorización concedida a los usuarios que no tienen autorización específica sobre el objeto, que no figuran en la lista de autorizaciones, y cuyo grupo de usuarios no tiene autorización específica sobre el objeto. La autorización se puede modificar para todos los usuarios o para usuarios específicos después de que se haya creado el programa con los mandatos de CL Otorgar autorización sobre el objeto (GRTOBJAUT) o Revocar autorización sobre el objeto (RVKOBJAUT). Para obtener más información sobre estos mandatos, consulte la sección sobre *CL y las API* de la categoría *Programación* en **iSeries 400 Information Center** del sitio Web - <http://www.ibm.com/eserver/iseres/infocenter>.

***LIBCRTAUT**

La autorización de uso público para el objeto se toma de la palabra clave CRTAUT de la biblioteca destino (la biblioteca que contiene el objeto). El valor se determina al crear el objeto. Si el valor CRTAUT para la biblioteca cambia después de crear el objeto, el nuevo valor no afectará a ningún objeto existente.

***ALL**

Autorización para todas las operaciones sobre el programa objeto, excepto para aquellas de uso exclusivo para el propietario o las que controla la autorización de gestión de lista de autorizaciones. El usuario puede controlar la existencia del programa objeto, especificar esta seguridad para el mismo, cambiarlo y realizar funciones básicas sobre él, pero no puede transferir su propiedad.

***CHANGE**

Proporciona toda la autorización de datos y la autorización para realizar todas las operaciones en el objeto de programa excepto las limitadas al propietario o controladas por la autorización de objeto y la autorización de gestión de objetos. El usuario puede modificar el objeto y realizar funciones básicas sobre el mismo.

***USE**

Proporciona autorización operacional del objeto y autorización de lectura; es decir, autorización para operaciones básicas sobre el objeto de programa. Se evita que el usuario modifique el objeto.

***EXCLUDE**

Se evita que el usuario acceda al objeto.

nombre de la lista de autorizaciones

Entre el nombre de una lista de autorizaciones de usuarios y autorizaciones a los que se ha añadido el programa. Esta lista de autorizaciones protegerá al objeto de programa y la autorización pública para el objeto de programa se establecerá en *AUTL. Cuando se emite el mandato CRTBNDRPG, la lista de autorizaciones debe existir en el sistema.

Nota: Utilice el parámetro AUT para reflejar los requisitos de seguridad de su sistema. En el manual *iSeries Security Reference* se describen los recursos de seguridad disponibles.

Mandato CRTBNDRPG

TRUNCNBR

Especifica si el valor truncado se pasa al campo de resultados o si se genera un error cuando se produce un desbordamiento numérico durante la ejecución del programa.

Nota: La opción TRUNCNBR no es válida para los cálculos realizados dentro de expresiones. (En el campo Extended-Factor 2 aparecen expresiones). Si se produce un desbordamiento para estos cálculos, siempre se producirá un error. Además, siempre se señala un desbordamiento cuando el valor que se asigna a un campo sin signo o íntegro está fuera de alcance.

*YES

Se ignora el desbordamiento numérico y el valor truncado se pasa al campo de resultados.

*NO

Cuando se detecta un desbordamiento numérico, se genera un error de tiempo de ejecución con el código de error RNX0103.

FIXNBR

Especifica si el compilador fija los datos decimales que no son válidos.

*NONE

Indica que los datos decimales que no son válidos provocarán errores de datos decimales durante el tiempo de ejecución si se utilizan.

*ZONED

El compilador fijará los datos decimales con zona que no son válidos en la conversión a datos empaquetados. Los blancos en campos numéricos serán tratados como ceros. Se comprobará la validez de cada dígito decimal. Si un dígito decimal no es válido, se sustituirá por cero. Si un signo no es válido, el signo se forzará a un código de signo positivo de 'F' hex. Si el signo es válido, pasará a una 'F' hex de signo positivo o a una 'D' hex de signo negativo, según corresponda. Si los datos empaquetados resultantes no son válidos, no se fijarán.

*INPUTPACKED

Indica que si se encuentran datos decimales empaquetados que no son válidos durante el proceso de especificaciones de entrada, la variable interna se establecerá en cero.

TGTRLS

Especifica el nivel de release del sistema operativo en el que pretende utilizar el objeto que se está creando. En los ejemplos para los valores *CURRENT y *PRV, y cuando se especifica el valor del *release destino*, se utiliza el formato VxRxMx para especificar el release, donde Vx corresponde a la versión, Rx al release y Mx se refiere al nivel de modificación. Por ejemplo, V2R3M0 se refiere a la versión 2, release 3, nivel de modificación 0.

Los valores válidos para este parámetro cambian en cada release. Los valores posibles son:

*CURRENT

El objeto debe utilizarse en el release del sistema operativo que actualmente se encuentra en ejecución en su sistema. Por ejemplo, si V2R3M5 se está ejecutando en el sistema, *CURRENT significa que pretende utilizar el objeto en un sistema con V2R3M5 instalado. También puede utilizar el objeto en un sistema con cualquier release posterior del sistema operativo instalado.

Nota: Si V2R3M5 se está ejecutando en el sistema y el objeto debe utilizarse en un sistema con V2R3M0 instalado, especifique TGTRLS(V2R3M0) y no TGTRLS(*CURRENT).

***PRV**

El objeto debe utilizarse en el release anterior con el nivel de modificación 0 del sistema operativo. Por ejemplo, si en el sistema se está ejecutando V2R3M5, *PRV significa que pretende utilizar el objeto en un sistema con V2R2M0 instalado. También puede utilizar el objeto en un sistema con cualquier release posterior del sistema operativo instalado.

release de destino

Especifica el release en el formato VxRxMx. Puede utilizar el objeto en un sistema con el release especificado o con cualquier release posterior del sistema operativo instalado.

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

Nota: La versión actual del mandato puede dar soporte a opciones que no estén disponibles en releases anteriores del mandato. Si el mandato se utiliza para crear objetos que deben utilizarse en un release anterior, se procesará mediante el compilador apropiado para ese release y no se reconocerán las opciones a las que no se da soporte. El compilador no emitirá necesariamente ningún aviso referente a las opciones que no puede procesar.

ALWNULL

Especifica cómo se permitirá al módulo RPG de ILE utilizar registros que contengan campos de capacidad nula desde archivos de base de datos descritos externamente.

***NO**

Especifica que el módulo RPG de ILE no procesará registros con campos de valor nulo desde archivos descritos externamente. Si intenta recuperar un registro con valores nulos, el módulo RPG de ILE no podrá acceder a ningún dato del registro y se producen errores de correlación de datos.

***INPUTONLY**

Especifica que el módulo RPG de ILE puede leer satisfactoriamente registros con campos de capacidad nula que contienen valores nulos desde archivos de base de datos descritos externamente sólo de entrada. Cuando se recupera un registro que contiene valores nulos, no se producen errores de correlación de datos y los valores por omisión de la base de datos se colocan en aquellos campos que contienen valores nulos. El módulo no puede:

- utilizar campos de clave con capacidad nula
- crear o actualizar registros que contienen campos con capacidad nula
- determinar si un campo con capacidad nula es realmente nulo mientras el módulo está en ejecución
- establecer un campo de capacidad nula como nulo.

***USRCTL**

Especifica que el módulo RPG de ILE puede leer, escribir y actualizar registros con valores nulos desde archivos de base de datos descritos

Mandato CRTBNDRPG

externamente. Los registros con claves nulas pueden recuperarse utilizando operaciones con clave. El módulo puede determinar si un campo de capacidad nula es realmente nulo y puede establecer un campo de capacidad nula como nulo para la salida o actualización. El programador es el responsable de garantizar que los campos que contienen valores nulos se utilizan correctamente dentro del módulo.

***YES**

Igual que *INPUTONLY.

BNDDIR

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

***NONE**

No se especifica ningún directorio de enlace.

nombre del directorio de enlace

Especifica el nombre del directorio de enlace que se utiliza en la resolución de símbolos.

El nombre del directorio puede calificarse con uno de los siguientes valores de biblioteca:

***LIBL**

El sistema examina la lista de bibliotecas en búsqueda de la biblioteca donde se ha almacenado el directorio de enlace.

***CURLIB**

Se busca la biblioteca actual para el trabajo. Si no se especifica ninguna biblioteca como la biblioteca actual para el trabajo, se utiliza la biblioteca QGPL.

***USRLIBL**

Sólo se buscan las bibliotecas en la parte del usuario de la lista de bibliotecas del trabajo.

nombre de la biblioteca

Especifica el nombre de la biblioteca que se debe buscar.

ACTGRP

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

QILE

Cuando se llama este programa, se activa en el grupo de activación llamado QILE.

***NEW**

Cuando se llama este programa, se activa en un grupo de activación nuevo.

***CALLER**

Cuando se llama este programa, se activa en el grupo de activación del llamador.

nombre del grupo de activación

Especifica el nombre del grupo de activación que se utilizará cuando se llame este programa.

ENBPFRCOL

Especifica si está habilitada la acumulación del rendimiento.

***PEP**

Las estadísticas de rendimiento se recogen sólo a la entrada y salida del procedimiento de entrada del programa. Esto es válido para el procedimiento de entrada de programa real de un programa, no para el procedimiento principal de los módulos dentro del programa. Éste es el valor por omisión.

***NEW**

Cuando se llama este programa, se activa en un grupo de activación nuevo.

***ENTRYEXIT**

Las estadísticas de rendimiento se recogen a la entrada y salida de todos los procedimientos del programa.

***FULL**

Las estadísticas de rendimiento se recogen a la entrada y salida de todos los procedimientos. Además, las estadísticas se recogen antes y después de cada llamada de un procedimiento externo.

DEFINE

Especifica los nombres de condición que se han definido antes de que empiece la compilación. El uso del parámetro DEFINE(nombre de condición) es equivalente a la codificación de la directiva /DEFINE nombre de condición en la primera línea del archivo fuente.

***NONE**

No se define ningún nombre de condición. Éste es el valor por omisión.

nombre de condición

Pueden especificarse hasta 32 nombres de condición. Cada nombre puede tener hasta 50 caracteres de longitud. Los nombres de condición se considerarán definidos al inicio de la compilación.

PRFDTA

Especifica el atributo de datos de definición del perfil del programa para el programa. La definición del perfil de programa es una técnica avanzada de optimización que se utiliza para reordenar los procedimientos y el código del interior de los procedimientos basándose en datos estadísticos (datos de definición de perfil).

***NOCOL**

No se permite que este programa recoja datos de definición de perfil. Este es el valor por omisión.

***COL**

Se permite que el programa recoja datos de definición de perfil. Puede especificarse *COL sólo si el nivel de optimización del módulo es *FULL y al compilar con un release de destino *CURRENT.

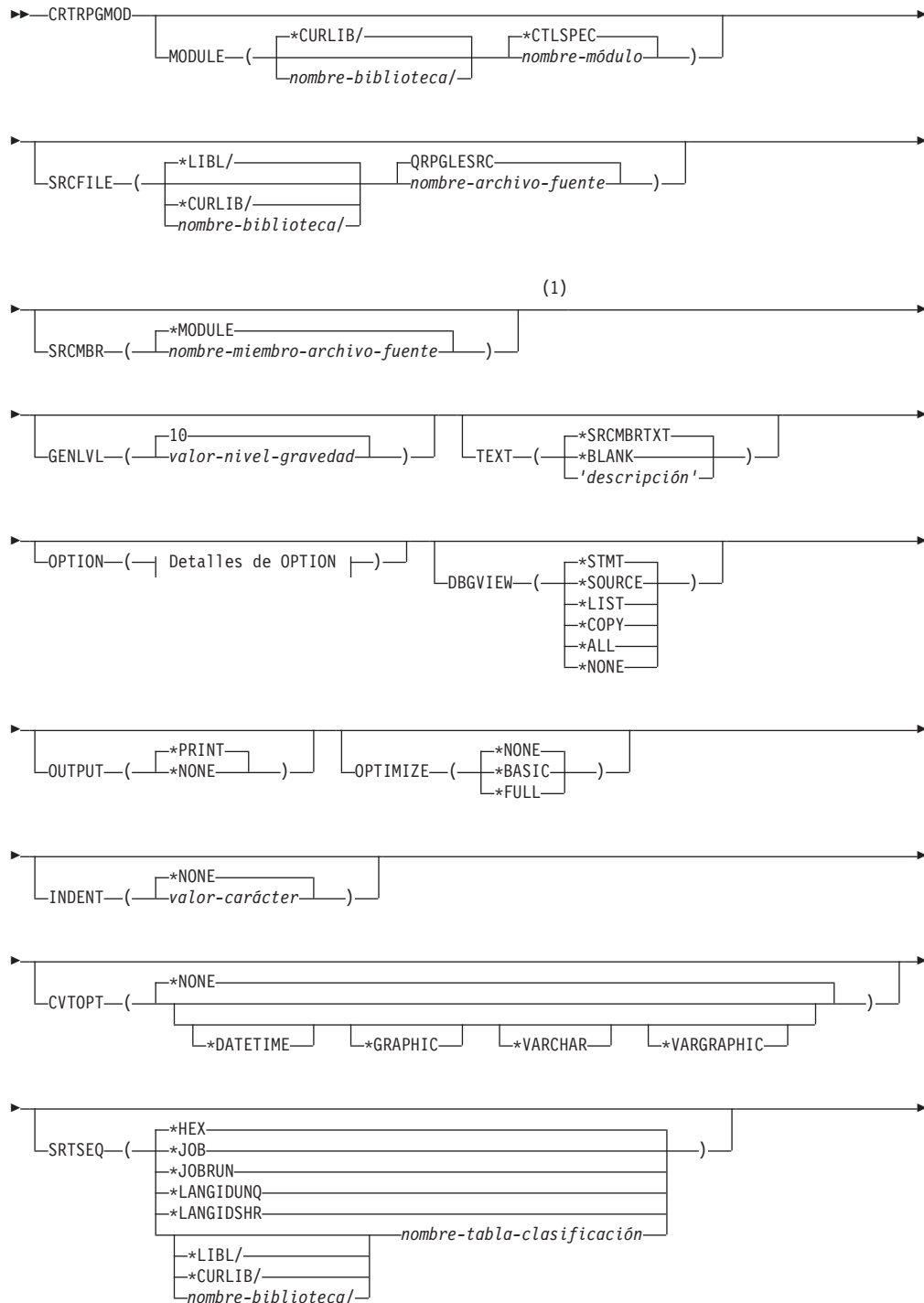
LICOPT

Especifica una o varias opciones de tiempo de compilación de Código interno con licencia. Este parámetro permite la selección de opciones de tiempo de compilación individuales y está diseñada para el programador avanzado que entiende las potenciales ventajas y desventajas de cada tipo seleccionado de opción del compilador.

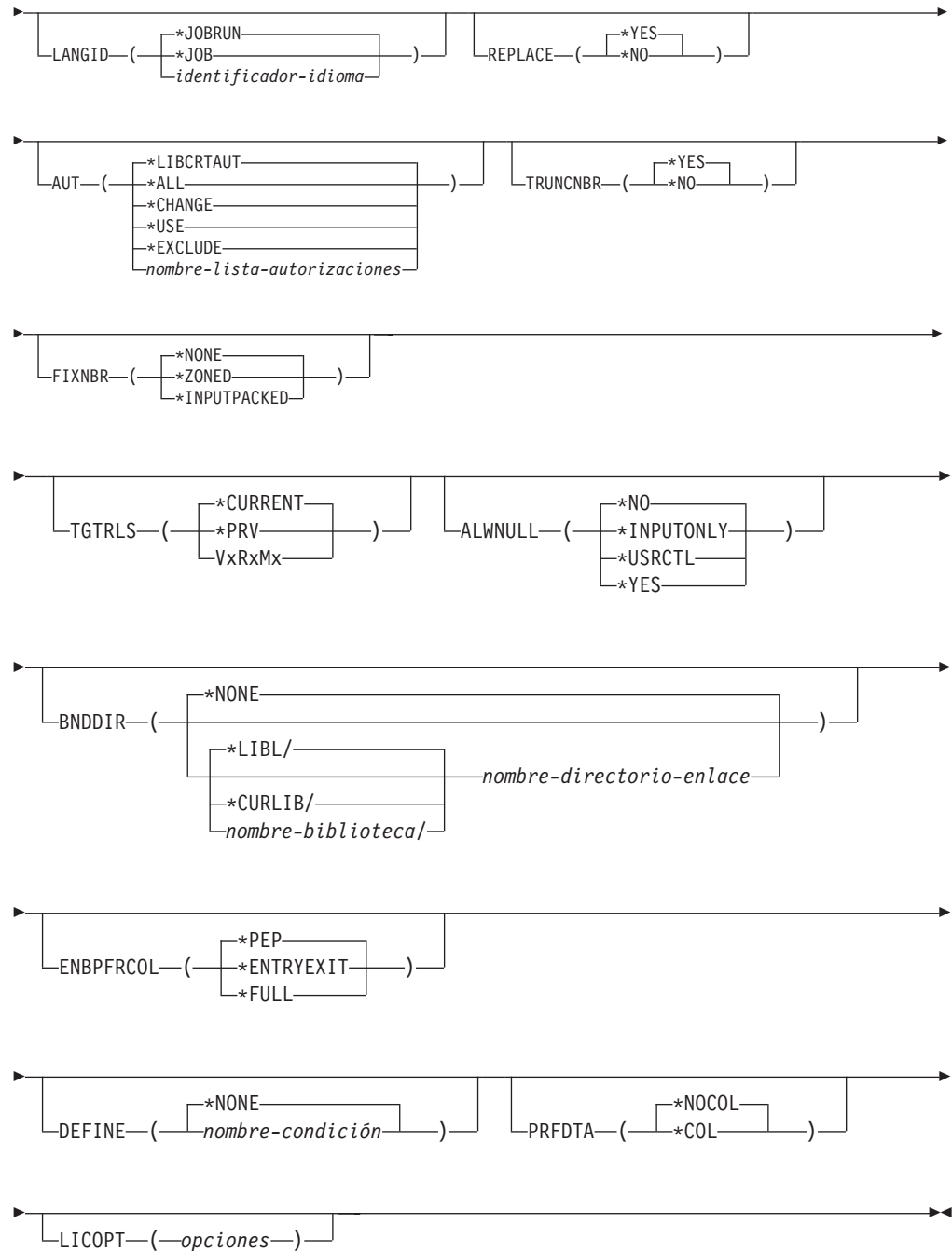
Mandato CRTRPGMOD

El mandato Crear módulo de RPG (CRTRPGMOD) compila el código fuente del ILE RPG para crear un objeto de tipo módulo (*MODULE). El diagrama de sintaxis completo para el mandato CRTRPGMOD se muestra a continuación.

Job: B,I Pgm: B,I REXX: B,I Exec

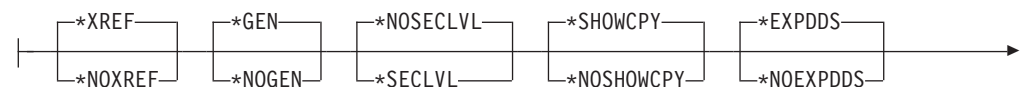


Mandato CRTRPGMOD

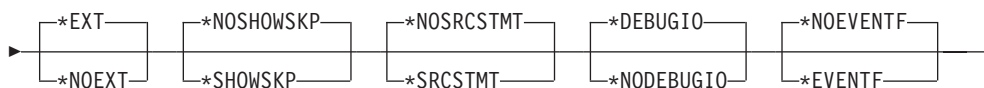


Notas:

- 1** Todos los parámetros anteriores a este punto pueden especificarse por posición

Detalles de OPTION:

Mandato CRTRPGMOD



Descripción del mandato CRTRPGMOD

Para obtener una descripción de los parámetros, opciones y variables del mandato CRTRPGMOD, consulte la descripción correspondiente del mandato CRTBNDRPG. Se corresponden con exactitud, excepto que los de CRTRPGMOD hacen referencia a módulos y no a programas. (Cuando examine las descripciones CRTBNDRPG, recuerde que CRTRPGMOD no tiene los parámetros siguientes: ACTGRP, DFTACTGRP, USRPRF).

También se puede obtener una descripción de CRTRPGMOD en línea. Entre el nombre del mandato en una línea de mandatos y pulse PF4 (Solicitud) y a continuación, pulse PF1 (Ayuda) para cualquier parámetro sobre el que desee información.

Apéndice D. Listados del compilador

Los listados del compilador le proporcionan información sobre si es correcto el código con respecto a la sintaxis y la semántica del lenguaje RPG IV. Los listados están diseñados para ayudarle a corregir los errores mediante un editor de fuente, así como para ayudarle en la depuración de un módulo. Este apartado le indica cómo interpretar un listado del compilador ILE RPG. Consulte el apartado “Utilización de un listado del compilador” en la página 67 para obtener más información sobre el uso de un listado.

Para obtener un listado del compilador, especifique OUTPUT(*PRINT) en el mandato CRTRPGMOD o en el mandato CRTBNDRPG. (Éste es el valor por omisión.) La especificación OUTPUT(*NONE) suprimirá un listado.

La Tabla 33 resume las especificaciones de palabra clave y su información de listado del compilador asociada.

Tabla 33. Secciones del listado del compilador

Sección del listado ¹	OPTION ²	Descripción
Prólogo		Resumen de opciones de mandatos
Listado del fuente		Especificaciones del fuente
Mensajes de diagnóstico en línea		Errores contenidos en una línea del fuente
miembros /COPY	*SHOWCPY	Registros fuente de miembros /COPY
Declaraciones que se han saltado	*SHOWSKP	Líneas fuente que las directivas de compilación condicional han excluido.
Archivos descritos externamente	*EXPDDS	Especificaciones generadas
Tabla de campos de comparación		Longitudes que coinciden basadas en campos coincidentes
Mensajes adicionales de diagnóstico		Errores que ocupan más de una línea del fuente
Posiciones de campo en el almacenamiento intermedio de salida		Posiciones de inicio y fin de campos de salida descritos por el programa
Tabla de miembros /COPY		Lista de miembros /COPY y sus nombres externos
Datos de tiempo de compilación		Registros fuente de compilación
Orden de clasificación alternativa		Tabla y registros de ALTSEQ o tabla e información de NLSS
Conversión de archivos		Registros de conversión de archivos
Matrices		Registros de matrices
Tablas		Registros de tabla
Información de campos de clave	*EXPDDS	Atributos de campos de clave
Referencia cruzada	*XREF	Referencias de archivos y registros, y de campos e indicadores
Referencias externas	*EXT	Lista de campos y procedimientos externos a la que se hace referencia durante la compilación.
Resumen de mensajes		Lista de mensajes y número de veces que se producen
Texto de segundo nivel	*SECLVL	Texto de segundo nivel de mensajes

Listados del compilador

Tabla 33. Secciones del listado del compilador (continuación)

Sección del listado ¹	OPTION ²	Descripción
Resumen final		Total de registros de fuente y mensajes y mensaje final de compilación
Errores de generación de código ³		Los errores (si los hay) que se producen durante la fase de generación de códigos.
Sección de enlaces ³		Errores (si los hay) que se producen durante la fase de enlace para el mandato CRTBNDRPG
Notas:		
<ol style="list-style-type: none">1. La información que figura en la sección del listado depende de si se ha especificado *SRCSTMT o *NOSRCSTMT para el parámetro OPTION. Para obtener más detalles sobre el modo en que cambia esta información, consulte los apartados ""Cabecera de fuente *NOSRCSTMT"" en la página 482 y ""Cabecera de fuente *SRCSTMT"" en la página 482. *SRCSTMT le permite solicitar que el compilador utilice los números de orden SEU y los ID de fuente al generar números de sentencia para la depuración. De lo contrario, los números de sentencia se asocian con los números de línea del listado y los números se asignan secuencialmente.2. La columna OPCIÓN indica qué valor especificar en el parámetro OPCIÓN para obtener esta información. Una entrada en blanco significa que la información siempre aparecerá si se especifica OUTPUT(*PRINT).3. Las secciones que contienen los errores de generación de código y los errores de enlace aparecen sólo si existen errores. No existe ninguna opción para suprimir estas secciones.		

Lectura de un listado del compilador

El texto siguiente contiene una discusión y un ejemplo de cada sección del listado del compilador. Las secciones se presentan en el orden en el que aparecen en el listado.

Prólogo

La sección del prólogo resume los parámetros del mandato y su valor tal como los procesó el analizador de mandatos CL. Si se especificó *CURLIB o *LIBL, se lista el nombre de la biblioteca real. También se indica en el prólogo el efecto de las alteraciones temporales. La Figura 230 en la página 477 muestra cómo interpretar la sección del prólogo del listado para el programa MYSRC, que se ha compilado utilizando el mandato CRTBNDRPG.

5769RG1 V4R4M0 990521 RN	IBM ILE RPG	MYLIB/MYSRC	1a	AS400S01	98/07/27 12:58:46	Página 1
Mandato	CRTBNDRPG					
Emitido por.	MIIDUSUA					
Programa	MYSRC		2			
Biblioteca	MYLIB					
Texto 'descripción'.	Texto especificado en el mandato					
Miembro fuente	MYSRC		3			
Archivo fuente	QRPGLSRC			4		
Biblioteca	MYLIB					
CCSID	37					
Texto 'descripción'.	Texto especificado en el miembro fuente					
Última modificación.	98/07/27 12:50:13					
Nivel de gravedad de generación.	10					
Grupo de activación por omisión.	*NO					
Opciones del compilador.	*XREF *GEN *SECLVL *SHOWCPY				5	
	*EXPDDS *EXT *SHOWSKP *NOSRCSTMT					
	*DEBUG *NOEVENTF					
Vistas de depuración	*ALL					
Salida	*PRINT					
Nivel de optimización.	*NONE					
Sangrado del listado fuente	' '		6			
Tipo de opciones de conversión	*NONE					
Orden de clasificación	*HEX					
Identificador de lenguaje.	*JOB RUN					
Sustituir programa	*YES					
Perfil de usuario	*USER					
Autorización	*LIBCRTAUT					
Truncar numérico	*YES					
Arreglar numérico.	*ZONED *INPUTPACKED					
Release de destino	*CURRENT					
Permitir valores nulos	*NO					
Directorio de enlace	BNDDIRA BNDDIRB					
Biblioteca	CMDLIBA CMDLIBB					
Grupo de activación.	CMDACTGRP					
Definir nombres de condición	ABC 7					
	DEF					
Permitir colección rendimiento	*PEP					
Datos de definición de perfil.	*NOCOL					

Figura 230. Ejemplo de prólogo para CRTBNDRPG

1 Cabecera de página

La información de la cabecera de página incluye la línea de información sobre el producto y el texto proporcionado por una directiva /TITLE. El apartado “Personalización de un listado del compilador” en la página 68 describe cómo puede personalizar la cabecera de página y el espaciado del listado del compilador.

2 Módulo o programa

El nombre del objeto de módulo creado (si utiliza CRTRPGMOD) o el nombre del objeto de programa creado (si utiliza CRTBNDRPG)

3 Miembro fuente

El nombre del miembro fuente desde el que se recuperaron los registros fuente (puede ser distinto de **2** si se utilizaron mandatos de alteración temporal).

4 Fuente

El nombre del archivo realmente utilizado para suministrar los registros fuente. Si se altera temporalmente el archivo, se utiliza el nombre del fuente que lo altera temporalmente.

5 Opciones del compilador

Las opciones del compilador vigentes en el momento de la compilación, tal como se especifiquen en el mandato CRTRPGMOD o el mandato CRTBNDRPG.

6 Marca de sangrado

El carácter utilizado para marcar operaciones estructuradas en la sección del fuente del listado.

7 Definir nombres de condición

Especifica los nombres de condición vigentes antes de leer el fuente.

Listados del compilador

Sección de fuente

La sección del fuente muestra los registros que comprenden las especificaciones fuente del ILE RPG. Los registros del miembro fuente raíz siempre se muestran. Si también se especifica `OPTION(*EXPDDS)`, entonces la sección de fuente muestra registros generados desde archivos descritos externamente y los marca con un '=' en la columna junto al número de línea. Estos registros no se muestran si está especificado `*NOEXPDDS`. Si se especifica `OPTION(*SHOWCPY)`, entonces también muestra los registros de los miembros /COPY especificados en el fuente, y los marca con un '+' en la columna junto al número de línea. Estos registros no se muestran si se ha especificado `*NOSHOWCPY`.

La sección del fuente también muestra el proceso de compilación condicional. Se imprimen todas las líneas con directivas `/IF`, `/ELSEIF`, `/ELSE` y `/ENDIF` y las líneas de fuente seleccionadas por los grupos `/IF` y se les asigna un número de línea del listado. Si se ha especificado `OPTION(*SHOWSKP)`, muestra todas las sentencias que han sido excluidas por las directivas `/IF`, `/ELSEIF` y `/ELSE`, y las marca mediante un '-----' en la columna junto a la sentencia. Los números de línea del listado no se incrementan para las líneas que se han excluido. Todas las sentencias que se han saltado se imprimen tal como se especificaron, pero no se interpretan de ningún modo. Por ejemplo, una sentencia excluida con una directiva `/EJECT` no provoca un salto de página. Del mismo modo, las directivas del compilador `/SPACE`, `/TITLE`, `/COPY` y `/EOF` se ignoran si aparecen en líneas excluidas. Estas sentencias no se muestran si se especifica `OPTION(*NOSHOWSKP)` por omisión; en su lugar, se imprime un mensaje con el número de líneas que se han excluido.

La sección del fuente identifica cualquier error de sintaxis del fuente, e incluye una tabla de campos de comparación cuando es necesario.

Si se especifica `OPTION(*NOSRCSTMT)`, los números de línea se imprimen secuencialmente en el lado izquierdo del listado indicando los números de línea del fuente compilados. Los números de orden SEU e ID de fuente se imprimen en el lado derecho del listado indicando los miembros y los registros del fuente respectivamente. Por ejemplo, la Figura 231 muestra una sección del listado con una sentencia `/COPY` en la línea 35. En el miembro fuente raíz, la siguiente línea es una operación `DOWEQ`. En el listado, sin embargo, la operación `DOWEQ` está en la línea 39. Las tres líneas que intervienen que aparecen en el listado son del miembro fuente `/COPY`.

Línea	Especificaciones de fuente										Comentarios	Sec	fuen	Id	Número
Número	1	2	3	4	5	6	7	8	9	10					
34	C														001500
35	C	/COPY	MYCPY												001600
		-----*													
		* Nombre de miembro RPG. . . : MYCPY													5
		* Nombre externo : RPGGUIDE/QRPGLESRC(MYCPY)													5
		* Última modificación. . . . : 98/07/24 16:20:04													5
		* Texto 'descripción' . . . : Texto en miembro de copia													5
		-----*													
36	C	Azul(1)	DSPLY												5000100
37	C	Verde(4)	DSPLY												5000200
38	C	Rojos(2)	DSPLY												5000300
39	C	*in20	doweq												001700

Figura 231. Ejemplo de sección del listado con `OPTION(*NOSRCSTMT)`

Si se especifica `OPTION(*SRCSTMT)`, los números de orden se imprimen en el lado izquierdo del listado indicando los números de orden SEU. Los números de sentencia se imprimen en el lado derecho del listado. La información del número de sentencia es idéntica a la información del número de orden SEU y el ID de

fuente. Por ejemplo, la Figura 232 muestra una sección del listado con una sentencia /COPY con el número de orden 001600. La siguiente línea del miembro fuente raíz es igual a la línea con el siguiente número de orden en el listado: el número de orden 001700. A las tres líneas que intervienen se les asignan los números de orden SEU del miembro fuente /COPY. Los números de sentencia correspondientes se generan a partir de los números de orden SEU e ID de fuente de los miembros fuente /COPY y raíz.

La Figura 233 en la página 480 muestra toda la sección de fuente para MYSRC con

Sec	<----- Especificaciones de fuente -----><----- Comentarios ----->										Sentencia
Número1....+....2....+<----- 26 - 35 ----->....4....+....5....+....6....+....7....+....8....+....9....+...10										Número
001500 C	MOVE	'123'	BI_FLD1								001500
001600 C/COPY MYCPY										971104 001600	

* Nombre de miembro RPG. . . : MYCPY										5	
* Nombre externo : RPGGUIDE/QRPGLESRC(MYCPY)										5	
* Última modificación. . . : 98/07/24 16:20:04										5	
* Texto 'descripción' . . . : Texto en miembro de copia										5	

000100+C	Azul(1)	DSPLY								5000100	
000200+C	Verde(4)	DSPLY								5000200	
000300+C	Rojos(2)	DSPLY								5000300	
001700 C	*in20	doweq	*OFF								001700

Figura 232. Ejemplo de sección del listado con OPTION(*SRCSTMT)

OPTION(*NOSRCSTMT) especificado.

Listados del compilador

5769RG1	V4R4M0	990521	RN	IBM ILE RPG	MYLIB/MYSRC	AS400S01	98/07/28 14:21:00	Página	2		
1a											
Línea	----- Especificaciones de fuente -----><----- Comentarios-->						Do	Página	Cambio	Ord	fuen
Número1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....10	Núm	Línea	Fecha	Id	Número					
Listado fuente											
1	H DFTACTGRP(*NO) ACTGRP('Srcactgrp')	CCSID(*GRAPH:*SRC)				980727			000100		
2	H OPTION(*NODEBUGIO)					980727			000200		
3	H BNDDIR('SRCLIB1/BNDDIR1' : 'SRCLIB2/BNDDIR2' : 'nom.ext')					971104			000300		
4	H ALTSEQ(*SRC)					971104			000400		
5	H FIXNBR(*ZONED)					980728			000500		
6	H TEXT('Texto especificado en la especificación de control')					971104			000600		
-----*											
* Opciones del compilador vigentes:											
-----*											
* Texto 'descripción'. :											
* Texto especificado en la especificación de control											
* Nivel de gravedad de generación. : 10											
* Grupo de activación por omisión. : *NO											
* Opciones del compilador. :											
* *XREF *GEN											
* *SECLVL *SHOWCPY											
* *EXPDDS *EXT											
* *SHOWSKP *NOSRCSTMT											
* *NODEBUGIO *NOEVENTF											
* *NONE											
* Sangrado del listado fuente. . . : ' '											
* Tipo de opciones de conversión. : *NONE											
* Orden de clasificación. : *HEX											
* Identificador del lenguaje. . . : *JOB RUN											
* Perfil del usuario. : *USER											
* Autorización. : *LIBCRTAUT											
* Truncar numérico. : *YES											
* Arreglar numérico. : *ZONED *INPUTPACKED											
* Permitir valores nulos. : *NO											
* Directorio enlace desde mandato : BNDDIRA BNDDIRB											
* Biblioteca. : CMDLIBA CMDLIBB											
* Directorio de enlace desde fuente: BNDDIR1 BNDDIR2											
* Biblioteca. : SRCLIB1 SRCLIB2											
* "nom.ext"											
* *LIBL											
* Grupo de activación. : Srcactgrp											
* Permitir colección rendimiento. : *PEP											
* Datos definición de perfil. . . : *NOCOL											
-----*											
7	FInFile	IF	E	DISK			971104		000700		
-----*											
* Nombre RPG Nombre externo											
* Nombre archivo. : INFILE MYLIB/INFILE											
* Formato(s) de registro. . . : INREC INREC											
-----*											
8	FKEYL6	IF	E	K DISK			971104		000800		
-----*											
* Nombre RPG Nombre externo											
* Nombre archivo. : KEYL6 MYLIB/KEYL6											
* Formato(s) registro. . . : REC1 REC1											
* REC2 REC2											
-----*											
9	FOutfile	O	E	DISK			971104		000900		
-----*											
* Nombre RPG Nombre externo											
* Nombre archivo. : OUTFILE MYLIB/OUTFILE											
* Formato(s) registro. . . : OUTREC OUTREC											
-----*											
10	D Azul	S	4	DIM(5) CTDATA PERRCD(1)			971104		001000		
11	D Verde	S	2	DIM(5) ALT(Azul)			971104		001100		
12	D Rojo	S	4	DIM(2) CTDATA PERRCD(1)			980727		001200		
13	D DSEXT1	E DS	100	PREFIX(BI) INZ(*EXTDFT)			980727		001300		
14	D FLD3	E		INZ('111')			980727		001400		

Figura 233. Ejemplo de parte fuente del listado (Pieza 1 de 3)

	-----											4		1
	* Estructura de datos. . . . : DSEXT1											*		1
	* Prefijo. : BI : 0											*		1
	* Formato externo : REC1 : MYLIB/DSEXT1											*		1
	* Texto del formato. . . . : Descripción del formato del registro											*		1
	-----											*		1
5	15=D	BI_FLD1	5A	EXTFLD (FLD1)	descripción FLD1	1000001								
	16=D			INZ (*BLANCO)		1000002								
	17=D	BI_FLD2	10A	EXTFLD (FLD2)	descripción FLD2	1000003								
	18=D			INZ (*BLANCO)		1000004								
	19=D	BI_FLD3	18A	EXTFLD (FLD3)	descripción FLD3	1000005								
	20=D			INZ ('111')		1000006								
	21=I	INREC											2000001	
	-----											*		2
	* Formato de registro RPG. . . : INREC											*		2
	* Formato externo : INREC : MYLIB/INFILE											*		2
	-----											*		2
	22=I		A	1	25	FLDA								
	23=I		A	26	90	FLDB								
	24=I	13488	*VAR	C	91	112	UCS2FLD							
	25=I	IREC1											2000003	
	-----											*		2000004
	* Formato de registro RPG. . . : REC1											*		2000001
	* Formato externo : REC1 : MYLIB/KEYL6											*		3
	-----											*		3
	26=I		*ISO-D	1	10	FLD12								
	27=I		A	11	13	FLD13								
	28=I		A	14	17	FLD14								
	29=I		A	18	22	FLD15								
	30=I	13488	C	23	32	FLDC								
	31=I	13488	*VAR	C	33	44	FLDCV							
	32=I	835	G	45	54	FLDG								
	33=I	IREC2											3000008	
	-----											*		4000001
	* Formato de registro RPG. . . : REC2											*		4
	* Formato externo : REC2 : MYLIB/KEYL6											*		4
	-----											*		4
	34=I		*ISO-D	1	10	FLD22								
	35=I		A	11	13	FLD23								
	36=I		A	14	17	FLD24								
	37=I		A	18	22	FLD25								
Línea	<----- Especificaciones de fuente-----><----- Comentarios ----->										Sec fuen			
Número1....+....2....+<-----	26 - 35	----->....4....+....5....+....6....+....7....+....8....+....9....+....10	Id	Número									
38	C	MOVE	'123'	BI_FLD1	971104	001500								
39	C	COPY MYCPY											001600	
	-----											6		5
	* Nombre de miembro RPG. . . : MYCPY											*		5
	* Nombre externo : MYLIB/QRPGLESRC(MYCPY)											*		5
	* Última modificación. . . . : 98/07/24 16:20:04											*		5
	* Texto 'descripción'. . . . : Texto especificado en miembro Copy											*		5
	-----											*		
7	40+C	Azul(1)	DSPLY											5000100
	41+C	Verde(4)	DSPLY											5000200
	42+C	Rojo(2)	DSPLY											5000300
8	43	C	*in20	doweq	*OFF								001700	
	44	C		READ	InRec	----20								001800
	45	C		if	NOT *in20								001900	
	46	C	FLDA	DSPLY								002000		
	47	C		endif								002100		
	48	C		enddo								002200		
	49	C		write	outrec								002300	
	50	C		SETON								002400		
	47	C	DEFINE ABC								971104	002500		
	51	C	IF DEFINED(ABC)								971104	002600		
	52	C	MOVEL	'x'	Y	10						002700		
	54	C	MOVEL	'x'	Z	10						002800		
	55	C	ELSE								971104	002900		
10	-----	C	MOVEL	' '	Y	10						971104		
	-----	C	MOVEL	' '	Z	10						971104		
	56	C	C/ENDIF								971104	003200		

Figura 233. Ejemplo de parte fuente del listado (Pieza 2 de 3)

Listados del compilador

Línea	<----- Especificaciones de fuente ----->	<----- Comentarios----->	Do	Página	Cambio	Ord	fuen
Número1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10	Núm	Línea	Fecha	Id	Número	
57=0	OUTREC					6000001	
	-----					6	
	* Formato de registro RPG. . : OUTREC					6	
	* Formato externo : OUTREC : MYLIB/OUTFILE					6	
	-----					6	
58=0	FLDY	100A CHAR	100			6000002	
59=0	FLDZ	132A CHAR	32			6000003	
60=0	GRAPHFLD	156G GRPH	12 835			6000004	
	* * * * * F I N D E L F U E N T E * * * * *						

Figura 233. Ejemplo de parte fuente del listado (Pieza 3 de 3)

1a Cabecera de fuente *NOSRCSTMT

La cabecera de fuente que se muestra en el ejemplo anterior se generó con OPTION(*NOSRCSTMT) especificado.

Número de línea

Comienza por 1 y aumenta en 1 por cada registro fuente o registro generado. Utilice este número al depurar utilizando números de sentencia.

Línea de regla

Esta línea se ajusta cuando se especifica el sangrado.

Número Do

Identifica el nivel de las operaciones estructuradas. Este número no aparecerá si se solicita el sangrado.

Línea de página

Muestra las 5 primeras columnas del registro fuente.

Id de fuente

Identifica el fuente (/COPY o DDS) del registro. Se puede utilizar para los miembros /COPY para obtener el nombre de miembro externo de la tabla de miembros /COPY.

Número de orden (en lado derecho del listado)

Muestra el número de orden del SEU del registro de un miembro de un archivo físico fuente. Muestra un número de incremento para los registros de un miembro /COPY o registros generados desde las DDS.

1b Cabecera de fuente *SRCSTMT

Cuando se especifica OPTION(*SRCSTMT), la cabecera de fuente pasa a:

Ord	<----- Especificaciones de fuente ----->	<----- Comentarios----->	Do	Página	Modificación	Sentencia
Número1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10	Núm	Línea	Fecha	Número	

La Línea de regla, el Número Do y la Línea de página no se modifican.

Número de orden (en lado izquierdo del listado)

Muestra el número de orden del SEU del registro de un miembro de un archivo físico fuente. Muestra un número de incremento para los registros de un miembro /COPY o registros generados desde las DDS.

Número de sentencia

Muestra el número de sentencia generado desde el número de ID fuente y el número de orden SEU del siguiente modo:

$\text{num_sent} = \text{ID_fuente} * 1000000 + \text{núm_orden_SEU_fuente}$

Utilice este número al depurar utilizando números de sentencia.

2 Opciones del compilador vigentes

Identifica las opciones del compilador vigentes. Se visualiza cuando se especifican palabras clave de opciones de compilador en la especificación de control.

3 Información de archivo/registro

Identifica el archivo descrito externamente y los registros que contiene.

4 Información de las DDS

Identifica desde qué archivo descrito externamente se extrae la información del campo. Muestra el valor del prefijo, si se especifica. Muestra el texto de registro de formato, si se especifica en las DDS.

5 Especificaciones generadas

Muestra las especificaciones generadas desde las DDS, indicadas mediante un '=' junto al Número de línea. Muestra hasta 50 caracteres de texto de campo, si se especifica en las DDS. Muestra el valor inicial tal como se especifica mediante la palabra clave INZ en la especificación de definición. Si se especifica INZ(*EXTDFT) para subcampos de estructura de datos descritos externamente, se visualiza el valor por omisión de las DDS. Los valores por omisión que son demasiado largos para ajustarse en una línea se truncan y se les añade el sufijo '...'.

6 Información de miembro /COPY

Identifica qué miembro /COPY se utiliza. Muestra el texto del miembro, si existe. Muestra la fecha y hora en que se modificó el miembro por última vez.

7 Registros de miembro /COPY

Muestra los registros del miembro /COPY, indicados mediante un '+' junto al Número de línea.

8 Sangrado

Muestra cómo aparecen las operaciones estructuradas cuando solicita que se marquen.

9 Utilización de indicador

Muestra la posición de los indicadores no utilizados, cuando se utiliza un indicador.

10 Utilización de OPTION(*SHOWSKP)

Muestra dos sentencias excluidas mediante una directiva /IF, que se indican con '-----' junto a las sentencias. Si se ha especificado OPTION(*NOSHOWSKP), estas dos sentencias se sustituirán por: LÍNEAS EXCLUIDAS: 2.

Mensajes de diagnóstico adicionales

La sección Mensajes de diagnóstico adicionales lista los mensajes del compilador que indican errores que ocupan más de una línea. Cuando es posible, los mensajes indican el número de línea y el número de orden del fuente en el que aparece el error. Figura 234 en la página 484 muestra un ejemplo.

Listados del compilador

```
      M e n s a j e s   d e   d i a g n ó s t i c o   a d i c i o n a l e s
ID mens. Gv Número Ord Texto mensaje
*RNF7066 00      8 000800 No se utiliza el formato de registro REC1 para entrada o salida.
*RNF7066 00      8 000800 No se utiliza el formato de registro REC2 para entrada o salida.
*RNF7086 00     60 000004 El RPG maneja la agrupación por bloques para el archivo INFILE. La INFDS sólo se actualiza
                        cuando se transfieren bloques de datos.
*RNF7086 00     60 000004 El RPG maneja la agrupación por bloques para el archivo OUTFILE. La INFDS sólo se actualiza
                        cuando se transfieren bloques de datos.
      * * * * *   F I N   D E   M E N S A J E S   A D I C I O N A L E S   D E   D I A G N Ó S T I C O   * * * * *
```

Figura 234. Ejemplo de mensajes de diagnóstico adicionales con `OPTION(*NOSRCSTMT)`

Si se especifica `OPTION(*SRCSTMT)`, los mensajes sólo tendrán el número de sentencia indicado. La Figura 235 muestra un ejemplo.

```
      M e n s a j e s   d e   d i a g n ó s t i c o   a d i c i o n a l e s
Id mens.Gv      Sentencia Texto mensaje
*RNF7066 00      000800 El formato de registro REC1 no se utiliza para entrada o salida.
*RNF7066 00      000800 El formato de registro REC2 no se utiliza para entrada o salida.
*RNF7086 00     6000004 El RPG maneja la agrupación por bloques para el archivo INFILE. La INFDS sólo se actualiza
                        cuando se transfieren bloques de datos.
*RNF7086 00     6000004 El RPG maneja la agrupación por bloques para el archivo OUTFILE. La INFDS sólo se actualiza
                        cuando se transfieren bloques de datos.
      * * * * *   F I N   D E   M E N S A J E S   A D I C I O N A L E S   D E   D I A G N Ó S T I C O   * * * * *
```

Figura 235. Ejemplo de mensajes de diagnóstico adicionales con `OPTION(*SRCSTMT)`

Posiciones del almacenamiento intermedio de salida

Las posiciones de campo de la tabla de posiciones del almacenamiento intermedio de salida se incluyen en el listado cuando el fuente contiene especificaciones de salida descritas por programa. Para cada variable o literal que sea una salida, la tabla contiene el número de línea de la especificación de campo de salida y las posiciones de inicio y fin dentro del almacenamiento intermedio de salida. Los literales que sean demasiado largos para la tabla se truncan y se les añade el sufijo '...' sin ningún apóstrofo final (por ejemplo, 'literal-demasiado-lar...'). La Figura 236 muestra un ejemplo de una tabla de posiciones del almacenamiento intermedio de salida.

```
      P o s i c i o n e s   A l m a c e n a m i e n t o   I n t e r m e d i o   d e   S a l i d a
Núm   Pos   Pos   Pos   Campo o constante
Línea Inic. Final
  58    1   100 FLDY
  59   101   132 FLDZ
  60   133   156 GRAPHFLD
      * * * * *   F I N   D E   P O S I C I Ó N   A L M A C E N A M I E N T O   I N T E R M E D I O   D E   S A L I D A   * * * * *
```

Figura 236. Tabla de posiciones del almacenamiento intermedio de salida

Tabla de miembros /COPY

La tabla de miembros /COPY identifica los miembros /COPY especificados en el fuente y lista sus nombres externos. Puede hallar el nombre y ubicación de un miembro utilizando el número de ID del fuente. La tabla también es útil como un registro de qué miembros fuente utiliza el módulo/programa. La Figura 237 muestra un ejemplo.

```
      M i e m b r o s   / C o p y
Línea Fuente nombre RPG <----- Nombre externo -----> CCSID <- Último cambio ->
Número Id      Biblioteca Archivo Miembro      Datos      Tiempo
  39    5 MYCPY      MYLIB      QRPGLSRC MYCPY      37  98/07/24 16:20:04
      * * * * *   F I N   D E   M I E M B R O S   / C O P Y   * * * * *
```

Figura 237. Ejemplo de tabla de miembros /COPY

Datos de tiempo de compilación

La sección sobre Datos en tiempo de compilación incluye información sobre las tablas ALTSEQ o NLSS y sobre las tablas y matrices. En este ejemplo, existe un orden de clasificación alternativo y dos matrices, tal como se muestra en la Figura 238.

61 **		Datos en tiempo de Compilación	971104	003300

		* Datos de tabla de orden de clasificación alternativo: *		

62	ALTSEQ	1122ACAB4B7C36F83A657D73	971104	003400
Línea	<----- Registros de datos ----->		Cambiar	Ord Fuent
Número	...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10		Fecha	Id Número

		* Tabla de orden de clasificación alternativo: *		
		* Número de caracteres con un orden alterado : 6 1 *		
		* 2 0_1_2_3_4_5_6_7_8_9_A_B_C_D_E_F_ *		
		* 0 0 *		
		* 1 . 22 3 1 *		
		* 2 2 *		
		* 3 3 *		
		* 4 4 *		
		* 5 5 *		
		* 6 . . . F8 6 *		
		* 7 7 *		
		* 8 8 *		
		* 9 9 *		
		* A . . 65 A *		
		* B . . . 7C B *		
		* C AB C *		
		* D 73 D *		
		* E E *		
		* F F *		
		* 0_1_2_3_4_5_6_7_8_9_A_B_C_D_E_F_ *		

63	**		971104	003500

		* Matriz . . . : AZUL 4 Matriz alternativa . . . : VERDE *		

64	1234ZZ		971104	003600
65	ABCDYY		971104	003700
66	5432XX		971104	003800
67	EDCBWW		971104	003900
68	ABCDEF		0980728	004000
69	**		971104	00410

		* Matriz . . . : ROJA *		

70	3861		971104	00420
71	TJKL		971104	00430
		* * * * F I N D A T O S T I E M P O C O M P I L A C I O N * * * * *		

Figura 238. Ejemplo de sección de datos de tiempo de compilación

1 Número total de caracteres alterados

Muestra el número de caracteres cuyo orden de clasificación se ha alterado.

2 Caracter para alterar

Las filas y columnas de la tabla identifican en conjunto los caracteres que se han de alterar. Por ejemplo, el nuevo valor para el carácter 3A es 65, se encuentra en la columna 3_ y la fila _A.

3 Orden alternativo

El nuevo valor de clasificación hexadecimal del carácter seleccionado.

4 Información de matriz/tabla

Identifica el nombre de la matriz o tabla para la que espera los datos el compilador. El nombre de la matriz alternativa también aparece, si está definido.

Información sobre campos de clave

La sección sobre Información de campos de clave muestra información sobre los campos de clave de cada archivo por clave. También muestra la información sobre

Listados del compilador

las claves que sean comunes para múltiples registros (es decir, las claves comunes). La Figura 239 muestra un ejemplo.

I n f o r m a c i ó n d e c a m p o s d e c l a v e			
Archivo	Nombre campo interno	Nombre campo externo	Atributos
2 CLVL6			
Teclas comunes:			
		DATE	*ISO- 10
		CHAR	3
REC1			
	FLD12	DATE	*ISO- 10
	FLD13	CHAR	3
	FLD15	CHAR	5
	FLDC	UCS2	5 13488
	FLDCV	VUC2	5 13488
	FLDG	GRPH	5 835
REC2			
	FLD22	DATE	*ISO- 10
	FLD23	CHAR	3
* * * * * F I N I N F O R M A C I O N C A M P O S C L A V E * * * * *			

Figura 239. Ejemplo de información de campos de clave

Tabla de referencias cruzadas

La tabla de referencias cruzadas contiene como mínimo tres listas:

- archivos y registros
- campos globales
- indicadores

Además, contiene campos locales que utiliza cada subprocedimiento. Utilice esta tabla para comprobar dónde se utilizan los archivos, campos e indicadores dentro del módulo/programa.

Tenga en cuenta que el mensaje informativo RNF7031, que se emite cuando no se hace referencia a un identificador, sólo aparecerá en la sección de referencias cruzadas del listado y en el resumen de mensajes. No aparece en la sección fuente del listado.

Los nombres con longitudes superiores a 122 caracteres aparecerán en la sección de referencias cruzadas de la división del listado en múltiples líneas. Se imprimirá todo el nombre con los caracteres '...' al final de las líneas. Si el segmento final del nombre tiene una longitud superior a los 17 caracteres, se listarán los atributos y números de línea empezando por la línea siguiente. La Figura 240 en la página 487 muestra un ejemplo del módulo TRANSRPT, que tiene dos procedimientos.

En este ejemplo, la tabla de referencias cruzadas muestra los números de línea para cada referencia. Si se especifica OPTION(*SRCSTMT), en lugar de OPTION(*NOSRCSTMT), se mostrarán los números de sentencia para cada referencia y el listado de referencias cruzadas podrá extenderse más allá de las primeras 80 columnas del listado.

Referencias cruzadas					
Referencias de archivo y registro:					
Archivo	Dispositivo	Referencias (D=Definida)			
Registro					
CUSTFILE	DISK	8D			
CUSTREC		0	44		
*RNF7031 CUSTRPT	DISK	9D			
ARREARS		0	60	79	
Referencias de campos globales:					
Campo	Atributos	Referencias (D=Definida M=Modificada)			
*INZSR	BEGSR	63D			
AMOUNT	P(10,2)	56M	83	95	
CITY	A(20)	53D	132		
CURDATE	D(10*ISO-)	42D	64M	92	
CUSTNAME	A(20)	50D	122		
CUSTNUM	P(5,0)	49D	124		
DUEDATE	A(10)	57M	84	91	
NOMBRE DE PROCEDIMIENTO EXTREMADAMENTE EXTENSO QUE REQUIERE MÁS DE UNA LÍNEA EN LA REFERENCIA CRUZADA A PESAR DE QUE... SE UTILIZA TODA LA LÍNEA HASTA LA COLUMNA 132 PARA IMPRIMIR EL NOMBRE...					
	I(5,0)	9D			
FMTCUST	PROTOTYPE	35D	59	113	114
	PROTOTYPE	134			
INARREARS	A(1)	30D	58	85	86
	PROTOTYPE	101			
LONG_FLOAT	F(8)	7D	11M	12M	
NUMT0CHAR	A(31)	22D	124	130	
	PROTOTYPE				
RPTADDR	A(100)	59	82		
RPTNAME	C(100)	59	81		
	CCSID(13488)				
RPTNUM	P(5,0)	80			
SHORT_FLOAT	F(4)	8D	10M		
*RNF7031 STATE	A(2)	54D			
STREETNAME	A(20)	52D	131		
STREETNUM	P(5,0)	51D	130		
ESTE_NOMBRE_NO_ES TAN LARGO...					
	A(5)	7D			
UDATE	S(6,0)	64			
*RNF7031 ZIP	P(5,0)	55D			
Referencias de campo INARREARS:					
Campo	Atributos	Referencias (D=Definida M=Modificada)			
DAYSlate	I(10,0)	88D	92M	94	
DATEDUE	D(10*ISO-)	89D	91M	92	
Referencias de campo FMTCUST:					
Campo	Atributos	Referencias (D=Definida M=Modificada)			
NAME	A(100)	115D	122M		
	BASED(_QRNL_PST+)				
ADDRESS	A(100)	116D	130M		
	BASED(_QRNL_PST+)				
Referencias de indicador:					
Indicador		Referencias (D=Definida M=Modificada)			
*RNF7031 01		44D			
***** FIN DE REFERENCIA CRUZADA *****					

Figura 240. Ejemplo de tabla de referencias cruzadas con OPTION(*NOSRCSTMT)

Lista de referencias externas

La sección sobre Referencias Externas lista los procedimientos y campos externos que se requieren de un módulo o están disponibles para otros módulos en tiempo de enlace. Esta sección se muestra siempre que el fuente contenga procedimientos de enlace estático, campos importados o campos exportados.

La parte sobre el procedimiento de enlace estático contiene el nombre del procedimiento, y las referencias al nombre de la operación CALLB o de la función incorporada %PADDR, o el nombre de un procedimiento de enlace de prototipo llamado por CALLP o dentro de una expresión.

Las partes de archivos importados y exportados contienen el nombre del campo, la dimensión si es una matriz, el atributo del campo y la referencia de la definición. La Figura 241 en la página 488 muestra un ejemplo.

Listados del compilador

Referencias Externas			
Procedimientos enlazados estáticamente:			
Procedimiento		Referencias	
PROTOTYPED		2	2
PADDR PROC		4	
CALLB PROC		6	
Campos importados:			
Campo	Atributos	Definido	
IMPORT_FLD	P(5,0)	3	
Campos exportados:			
Campo	Atributos	Definido	
EXPORT_ARR(2)	A(5)	2	
***** FIN DE REFERENCIAS EXTERNAS *****			

Figura 241. Ejemplo de referencias externas

Resumen de mensajes

El resumen de mensajes contiene totales ordenador por la gravedad de los errores que se han producido. Si se especifica OPTION(*SECLVL), también proporciona texto de mensajes de segundo nivel. La Figura 242 muestra un ejemplo.

Resumen de mensajes			
ID mens.	Gv	Número	Texto mensaje
*RNF7031	00	16	No se incluye una referencia al nombre o indicador. Causa : El campo, subcampo, TAG, estructura de datos, PLIST, KLIST, subrutina, indicador o prototipo se definen en el programa, pero no se incorporan referencias. Recuperación . . . : Se hace referencia al ítem, o se elimina del programa. Volver a compilar.
*RNF7066	00	2	No se utiliza el formato de registro del archivo descrito externamente. Causa : Existe un nombre de formato de registro para un archivo descrito externamente que no se utiliza en una operación de entrada o salida válida. Recuperación . . . : Utilizar el nombre del formato de registro del archivo descrito externamente para entrada o salida, o especificar el nombre como un parámetro para la palabra clave IGNORE. Volver a compilar.
*RNF7086	00	2	El RPG maneja la agrupación por bloques para el archivo. La INFDS sólo se actualiza cuando se transfieren bloques de datos. Causa : El RPG especifica MLTRCD(*YES) en el UFCB (Bloque de control del archivo de usuario). Los registros se transfieren entre el RPG y la administración de datos en bloques. Las posiciones 241 a final de la INFDS (Estructura de datos de información de archivo) sólo se actualizan cuando se lee o se graba un bloque de registros. Recuperación . . . : Si se necesita esta información después de cada lectura o grabación de un registro, especifique el mandato OVRDBF para el archivo con SEQONLY(*NO).
***** FIN DE RESUMEN DE MENSAJES *****			

Figura 242. Ejemplo de resumen de mensajes

Resumen final

La sección sobre el resumen final proporciona las estadísticas finales sobre los mensajes y sobre el fuente También especifica el estado de la compilación. La Figura 243 en la página 489 muestra un ejemplo.

R e s u m e n f i n a l		
Total de mensajes:		
Información (00) :	20
Aviso (10) :	0
Error (20) :	0
Error grave (30+) :	0

Total :	20
Totales del fuente:		
Registros :	71
Especificaciones :	55
Registros de datos :	8
Comentarios :	0
***** FIN DE RESUMEN FINAL *****		
Programa MYSRC ubicado en biblioteca MYLIB. 00 nivel de gravedad más alto. Creado el 98/07/28 a las 14:21:03.		
***** FIN DE COMPILACION *****		

Figura 243. Ejemplo de resumen final

Errores de generación de código y de enlace

Tras la sección sobre el resumen final, hallará una sección con los errores de generación de código y/o los errores de enlace.

La sección sobre errores de generación de código aparecerá sólo si se producen errores mientras el compilador está generando código para el objeto de módulo. Generalmente, esta sección no aparecerá. La sección sobre errores de enlace aparecerá siempre que haya mensajes producidos durante la fase de enlace del mandato CRTBNDRPG. Un error común es dejar de especificar la ubicación de *todos* los procedimientos externos y los campos a que se hacía referencia en el fuente en el momento en que se emitió el mandato CRTBNDRPG.

Avisos

Esta información ha sido desarrollada para los productos y servicios que se ofrecen en EEUU. Puede que IBM no ofrezca los productos, servicios o funciones que aparecen en este documento en otros países. Póngase en contacto con su representante local de IBM y solicítele información sobre los productos y servicios disponibles actualmente en su zona. Cualquier referencia a un producto, programa o servicio de IBM no supone ni implica que sólo puedan utilizarse los productos, programas o servicios de IBM. En su lugar puede utilizarse cualquier producto, programa o servicio que no infrinja ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente pendientes que abarquen el tema descrito en este documento. El suministro de este documento no le concede ninguna licencia sobre estas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

El siguiente párrafo no es válido para el Reino Unido o cualquier otro país donde dichas cláusulas sean inconsistentes con la ley local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, YA SEA IMPLÍCITAS O EXPLÍCITAS, INCLUSIVE, SIN LIMITARSE A ELLO, LAS GARANTÍAS IMPLÍCITAS DE NO-INCUMPLIMIENTO, COMERCIALIZACIÓN O ADECUACIÓN A UN FÍN DETERMINADO. Algunos estados no permiten la renuncia de garantías explícitas o implícitas en determinadas transacciones, por lo tanto, puede que esta cláusula no le afecte.

Esta información podría incluir imprecisiones técnicas o errores tipográficos. Periódicamente se introducen cambios en la información que aquí se incluye; estos cambios se incorporarán en las nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin aviso.

Cualquier referencia incluida a sitios Web que no pertenezcan a IBM se proporciona sólo por comodidad y no se entenderá bajo ningún concepto como una aprobación de dichos sitios Web. Los materiales de dichos sitios Web no forman parte de los materiales para este producto IBM y el uso de dichos sitios Web corre a su riesgo.

IBM puede utilizar o distribuir cualquier información que usted proporcione del modo que considere apropiado sin incurrir en ninguna obligación hacia usted.

Los poseedores de una licencia de este programa que deseen obtener información al respecto con el fin de permitir: (i) el intercambio de información entre programas creados de manera independiente y otros programas (incluido éste) y (ii) la utilización mutua de la información que se ha intercambiado, deben dirigirse a la siguiente dirección:

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, que en algunos casos incluirán el pago de una cuota.

IBM proporciona el programa bajo licencia descrito en esta información y todo el material bajo licencia disponible para el mismo bajo las condiciones del Contrato del cliente de IBM, el Contrato de licencia del programa internacional de IBM o cualquier contrato equivalente entre IBM y el usuario.

Esta información contiene ejemplos de datos e informes que se utilizan diariamente en operaciones comerciales. Para ilustrarlas de la manera más completa posible, los ejemplos incluyen nombres de individuos, compañías, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones utilizados por una empresa comercial real es pura coincidencia.

LICENCIA DE DERECHOS DE REPRODUCCIÓN:

Esta información contiene programas de aplicación modelo en lenguaje fuente, que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas modelo de la forma deseada sin pagar a IBM para desarrollar, utilizar, comercializar o distribuir programas de aplicación según la interface de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas modelo. Estos ejemplos no se han verificado a conciencia bajo todas las condiciones. Por lo tanto, IBM no puede garantizar o aludir a la fiabilidad, operabilidad o funcionamiento de estos programas. Puede copiar, modificar y distribuir estos programas modelo de la forma deseada sin pagar a IBM para desarrollar, utilizar, comercializar o distribuir programas de aplicación según la interface de programación de aplicaciones de IBM.

Información de la interfaz de programación

Esta publicación está diseñada para ayudarle a crear programas utilizando el RPG IV fuente. Esta publicación contiene información sobre la Interface de programación de uso general y ayuda asociada que proporciona el compilador ILE RPG.

Las interfaces de programación de uso general permiten al cliente escribir programas que soliciten o reciban los servicios del compilador ILE RPG.

Marcas registradas y marcas de servicio

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

400	IBM
AFP	IBMLink
Application System/400	Integrated Language Environment
AS/400	MQSeries
AS/400e	Operating System/400
C/400	OS/400

COBOL/400	PROFS
DB2	RPG/400
e (Stylized)	System/36
@server	System/38
FORTRAN/400	VisualAge
GDDM	WebSphere
iSeries	

Domino es una marca registrada de Lotus Development Corporation en los Estados Unidos o/y otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos o/y otros países.

Microsoft, Windows, Windows NT y el logo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos o/y otros países.

UNIX es una marca comercial registrada en los Estados Unidos o/y otros países y X/Open Company Limited es el único que otorga las licencias.

Otros nombres de compañías, productos y servicios pueden ser marcas registradas o marcas de servicio de terceros.

Las marcas registradas y las marcas sin registrar se señalan mediante ® y ™ respectivamente.

Bibliografía

Para obtener información adicional sobre temas relacionados con la programación en ILE RPG en el sistema iSeries, consulte las publicaciones de IBM iSeries siguientes:

- *ADTS/400: Application Development Manager User's Guide*, SC09-2133-02, describe la creación y gestión de proyectos definidos para la característica Gestor de desarrollo de aplicaciones, así como la utilización del programa para desarrollar aplicaciones.
 - *ADTS/400: Programming Development Manager*, SC09-1771-00, proporciona información sobre cómo utilizar el gestor del desarrollo de programación (PDM) para trabajar con listas de bibliotecas, objetos, miembros y opciones definidas por el usuario para realizar con facilidad operaciones como copiar, suprimir y renombrar. Contiene ejemplos e información de consulta para ayudar al usuario en el aprendizaje de PDM. Las operaciones y las teclas de función que se utilizan de forma más habitual se explican a través de ejemplos.
 - La *ADTS for AS/400: Source Entry Utility*, SC09-2605-00 proporciona información sobre cómo utilizar la Application Development ToolSet Source Entry Utility (SEU) para crear y editar miembros fuente. Este manual explica cómo iniciar y finalizar una sesión de SEU y cómo utilizar las múltiples características de este editor de texto de pantalla completa. Este manual contiene ejemplos que ayudan tanto a los nuevos usuarios como a los usuarios con experiencia a realizar diferentes tareas de edición, desde utilizar los mandatos en línea más simples a utilizar solicitudes predefinidas para lenguaje de alto nivel y formatos de datos.
 - *Application Display Programming*, SC41-5715-00 proporciona información sobre:
 - Utilizar las DDS para crear y mantener pantallas para aplicaciones;
 - Crear y trabajar con archivos de pantalla en el sistema;
 - Crear información de ayuda en línea;
 - Utilizar UIM para definir paneles y diálogos para una aplicación;
 - Utilizar grupos de paneles, registros o documentos.
 - *Backup and Recovery*, SC41-5304-05 proporciona información sobre cómo establecer y gestionar lo siguiente:
 - Creación de diarios, protección de la vía de acceso y control de compromiso
 - Agrupaciones de almacenamiento auxiliares del usuario (ASP)
 - Protección de disco (paridad de dispositivo, duplicación de disco y suma de comprobación)
- Proporciona información de rendimiento sobre copia de seguridad del soporte magnético y operaciones de salvar y restaurar. También incluye temas avanzados de recuperación y copia de seguridad, como la utilización del soporte para salvar en tiempo de actividad, salvar y restaurar a un release diferente y consejos y técnicas de programación.
- *CL Programming*, SC41-5721-04 proporciona una amplia descripción de la programación en iSeries en la que se incluye una descripción general de objetos y bibliotecas, programación en CL, control de flujo y comunicaciones entre programas, trabajar con objetos en programas CL y creación de programas en CL. También se incluye una explicación sobre mensajes predefinidos e improvisados y manejo de mensajes, definición y creación de mandatos y menús definidos por el usuario, pruebas de aplicación entre las que se incluyen modalidad de depuración, puntos de interrupción, rastreos y funciones de pantalla.
 - *Communications Management*, SC41-5406-02 proporciona información sobre la gestión del trabajo en un entorno de comunicaciones, el estado de las comunicaciones, el rastreo y diagnóstico de problemas de comunicaciones, el manejo y recuperación de errores, el rendimiento e información de almacenamiento de velocidad de línea específica y de subsistema.
 - *GDDM Programming Guide*, SC41-0536-00 proporciona información sobre cómo utilizar el gestor de visualización de datos gráficos (GDDM) del OS/400 para grabar programas de aplicación de gráficos. Incluye muchos programas de ejemplo e información para

ayudar a los usuarios a entender cómo se integra el producto en el sistema de proceso de datos.

- *GDDM Reference*, SC41-3718-00 proporciona información sobre cómo utilizar el gestor de visualización de datos gráficos (GDDM) del OS/400 para grabar programas de aplicación de gráficos. Este manual proporciona descripciones detalladas de todas las rutinas de gráficos disponibles en GDDM. También proporciona información sobre las interfaces de lenguaje de alto nivel para GDDM.
- *ICF Programming*, SC41-5442-00 proporciona la información necesaria para grabar programas de aplicación que utilizan comunicaciones de iSeries y la función de comunicaciones entre sistemas de OS/400 (OS/400-ICF). También contiene información sobre palabras clave de especificaciones de descripción de datos (DDS), formatos suministrados por el sistema, códigos de retorno, soporte de transferencia de archivos y ejemplos de programas.
- *IDDU Use*, SC41-5704-00 describe cómo utilizar el programa de utilidad de definición de datos interactiva (IDDU) del iSeries para describir diccionarios de datos, archivos y registros al sistema. Incluye:
 - Una introducción a los conceptos de definición de datos y archivos del sistema
 - Una introducción a la utilización de IDDU para describir los datos utilizados en consultas y documentos
 - Tareas representativas relacionadas con la creación, mantenimiento y utilización de diccionarios de datos, archivos, formatos de registro y campos.
 - Información avanzada sobre cómo utilizar IDDU para trabajar con campos creados en otros sistemas e información sobre recuperación de errores y prevención de problemas.
- *ILE C/C++ Programmer's Guide*, SC09-2712-02 proporciona información sobre cómo desarrollar aplicaciones mediante el lenguaje C de ILE. Incluye información sobre creación, ejecución y depuración de programas. También incluye consideraciones de programación para llamadas de programa de interlenguaje y procedimientos, local, excepciones de manejo, base de datos y archivos descritos externamente y de dispositivo. También se proporcionan algunos consejos sobre cómo mejorar el rendimiento. En el apéndice se incluye

información sobre el código fuente de migración desde de EPM o desde del sistema a C de ILE.

- *ILE COBOL Programmer's Guide*, SC09-2540-02 proporciona información sobre cómo grabar, compilar, enlazar de forma lógica, ejecutar, depurar y mantener programas COBOL de ILE en el sistema iSeries. Proporciona información de programación sobre cómo llamar a otros programas COBOL/400 de ILE y COBOL que no son de ILE, compartir datos con otros programas, utilizar punteros y manejar excepciones. También describe cómo efectuar operaciones de entrada y salida en dispositivos conectados externamente, archivos de base de datos, archivos de pantalla y archivos ICF.
- *ILE Concepts*, SC41-5606-05 explica los conceptos y la terminología pertenecientes a la arquitectura del Entorno de lenguajes integrados (ILE) del programa bajo licencia de OS/400. Se incluyen temas sobre la creación de módulos, el enlace, la ejecución de programas, la depuración de programas y el manejo de excepciones.
- *ILE RPG Reference*, SC09-2508-03 proporciona información sobre el lenguaje de programación RPG de ILE. Este manual describe, para cada una de las posiciones y palabras clave, las entradas válidas para todas las especificaciones de RPG IV, y proporciona una descripción detallada de todos los códigos de operación y funciones incorporadas. Este manual también contiene información sobre el ciclo lógico de RPG, matrices y tablas, las funciones de edición e indicadores.
- *ILE RPG Reference Summary*, SX09-1315-02 proporciona información sobre los lenguajes de programación RPG III y RPG IV. Este manual contiene tablas y listas para todas las especificaciones y operaciones en ambos lenguajes. Se proporciona una clave para correlacionar especificaciones y operaciones de RPG III con especificaciones y operaciones de RPG IV.
- *Printer Device Programming*, SC41-5713-04 proporciona información que le ayudará a entender y controlar la impresión. Proporciona información específica sobre elementos y conceptos de impresión del sistema iSeries, el soporte de impresión y de archivos de impresora para operaciones de impresión, y la conectividad de impresoras. Incluye consideraciones sobre la utilización de sistemas personales, otras funciones de impresión como Business Graphics Utility (BGU), funciones

avanzadas de impresión AFP), y ejemplos de trabajo con elementos de impresión del sistema iSeries, como mover archivos de salida en spool de una cola de salida a otra distinta. También incluye un apéndice de mandatos del lenguaje de control (CL) utilizado para gestionar la carga de impresión de trabajos. También se proporcionan los fonts disponibles que pueden utilizarse con el sistema iSeries. Las tablas de sustitución proporcionan una referencia cruzada de fonts que pueden sustituirse si las impresoras conectadas no dan soporte a los fonts especificados en la aplicación.

- *iSeries Security Reference*, SC41-5302-04 le indica cómo puede utilizarse el soporte de seguridad del sistema para proteger el sistema y los datos contra la utilización por parte de otros usuarios que carecen de la autorización adecuada, para proteger los datos contra la destrucción o deterioro intencionado o accidental, para mantener la información de seguridad actualizada y para configurar la seguridad del sistema.
- *Software Installation*, SC41-5120-05 proporciona 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 también está pensado para usuarios que ya tienen un sistema iSeries con una release instalada y desean instalar una nueva release.
- *System Operation*, SC41-4203-00 proporciona información sobre cómo manejar mensajes, trabajar con salidas de trabajos e impresoras, establecer comunicaciones de dispositivos, trabajar con funciones de soporte, realizar la limpieza del sistema, etc.
- *Tape and Diskette Device Programming*, SC41-5716-01 proporciona información para ayudar al usuario a desarrollar y dar soporte a programas que utilizan unidades de cinta y disquetes para E/S. Incluye información sobre archivos de dispositivo y descripciones de dispositivos de cinta y disquete.
- | • *Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More* proporciona indicaciones y consejos para los programadores del sistema iSeries que desean aprovechar al máximo el IV de RPG y el Entorno de lenguajes integrados (ILE). Puede acceder a esta publicación desde el sitio Web Redbooks de IBM:
| <http://www.redbooks.ibm.com/>

Índice

Caracteres Especiales

%ADDR, función incorporada de depuración 240
%ADDR (Obtener dirección de variable) parámetros omitidos 144
%INDEX, función incorporada de depuración 240
%PARMS (Devolver número de parámetros) comprobación del número de parámetros 145
%SUBSTR, función incorporada de depuración cambiar valores 242 ejemplos 240
%VARS, función incorporada de depuración 240
*CALLER 113
*CANCL 274
/COPY, sentencia en un informe de conversión 445 problemas de conversión 440, 449 tabla en listado del compilador 484 vista de depuración COPY 200
*DETC 274
*DETL 274
*ENTRY PLIST 156
*EXTDFT ejemplo 479 en listado del compilador 483
*GETIN 274
*JOB secuencia de ordenación, SRTSEQ 465
*JOB RUN identificador de idioma, LANGID 466 secuencia de ordenación, SRTSEQ 465
*NEW 112
*OFL 274
*OMIT 143, 144
*TOTC 274
*TOTL 274
*USER perfil de usuario, USRPRF 466

A

ACTGRP, parámetro CRTBNDRPG, mandato 62 especificar 112 mandato CRTBNDRPG 470 mandato CRTPGM 86 utilizar 64
activación de programa 112
actualización del programa de servicio 101

Actualizar programa (UPDPGM), mandato utilizar 90
ADTS 14
almacenamiento dinámico 115
alteración temporal de descripción externa 306
alteraciones temporales de archivo 306 discusión general 313, 344 ejemplo 314 indicado en un listado del compilador 477
ALWNULL, parámetro CRTBNDRPG, mandato 62 mandato CRTBNDRPG 469 mandato CRTRPGMOD 78
ámbito de archivos 84
análisis de la conversión 444
Añadir entrada de lista de respuestas (ADDRPLYE), mandato añadir a lista de respuestas del sistema 111
añadir objetos a una sesión de depuración 204
API Desregistrar Manejador de Condiciones ILE (CEEHDLU) 275
API enlazables CEE4ABN 160 CEECRHP (Crear área de almacenamiento dinámico) 22 CEECRHP (Crear Área de Almacenamiento Dinámico) 123 CEECZST (Reasignar almacenamiento) 23 CEEDSHP (Descartar área de almacenamiento dinámico) 23 CEEDSHP (Descartar Área de Almacenamiento Dinámico) 123 CEEFRST (Liberar almacenamiento) 23 CEEGTST (Obtener Almacenamiento de Área de Almacenamiento Dinámico) 23, 123 CEEHDLR (Registrar Manejador de Condiciones ILE) 275 CEEHDLU (Desregistrar Manejador de Condiciones ILE) 275 CEERTX (Registrar procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282 CEETREC 159 CEETSTA (Comprobar argumento omitido) 144 CEEUTX (Procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282 convenio de llamada 160 Crear área de almacenamiento dinámico (CEECRHP) 22

API enlazables (*continuación*)
Crear Área de Almacenamiento Dinámico (CEECRHP) 123
Descartar área de almacenamiento dinámico (CEEDSHP) 23
Descartar Área de Almacenamiento Dinámico (CEEDSHP) 123 descripción 160 ejemplo de codificación 160 Liberar almacenamiento (CEEFRST) 23
Obtener almacenamiento de área de almacenamiento dinámico (CEEGTST) 23
Obtener Almacenamiento de Área de Almacenamiento Dinámico (CEEGTST) 123
Obtener Información de Descripción acerca de un Argumento de Serie (CEESGI) 143
pasar descriptores operativos a 142
Reasignar almacenamiento (CEECZST) 23
Recuperar Información de Descriptor Operativo (CEEDOD) 143
visión general 22
volver de un procedimiento 159
API Registrar Manejador de Condiciones ILE (CEEHDLR) 275
aplicaciones de hebras bloquear y desbloquear procedimientos 164 consideraciones acerca de la codificación 162 depurar 209 visión general 23
Application Development Manager 14
Application Development ToolSet 14
Application Dictionary Services 15
archivo alterar temporalmente 306 bloqueo 315 claves válidas 325 compartimiento 317 consideraciones generales 299 convenios para la denominación 301 dependencia de dispositivo 299 descrito externamente 299 descrito por programa 299, 310 diagramas de proceso archivo secuencial 367 archivo SPECIAL 369 archivo WORKSTN 383 diferencias entre ILE RPG y OPM RPG/400 425
DISK 321
DISK descrito de forma externa 322 independencia de dispositivo 299 indexado 328 nombre alterar temporalmente 306

- archivo (*continuación*)
 - descrito externamente 299
 - descrito por programa 310
 - opciones de apertura 317
 - PRINTER 356
 - redireccionamiento 301
 - SEQ 331, 366
 - WORKSTN 371
- archivo, alteraciones temporales 306
 - discusión general 313, 344
 - ejemplo 314
 - indicado en un listado del compilador 477
- archivo combinado 382
- archivo de anotaciones cronológicas
 - acerca de 431
 - DDS para 447
 - utilizar 447
- archivo de auditoría 431
- archivo de base de datos
 - archivo de datos 321
 - archivos físicos y lógicos 321
 - descripción del nivel del campo 321
 - descripción del nivel del registro 321
 - discusión general 321
 - miembro fuente 322
- archivo de cintas 331
- archivo de comunicaciones ICF 371
- archivo de datos, convertir fuente desde 441
- archivo de direcciones de registros
 - con números relativos de registro 331
 - con registros de límites 331
 - número relativo de registro 331
 - problemas de conversión 443, 451
 - secuencial entre límites 331
- archivo de eventos para CODE/400 462
- archivo de múltiples dispositivos
 - WORKSTN 383
- archivo de referencias de campos, ejemplo de 324
- archivo de salida 382
- archivo descrito externamente
 - adición a descripción externa 304
 - alteración temporal 306
 - archivos físicos y lógicos 321
 - cambio de nombre de formato de registro 304
 - cambio de nombres de campo 305
 - como archivo WORKSTN 371, 375
 - como descrito por programa 302
 - definición 301
 - especificaciones 304
 - especificaciones de descripción de archivo para 304
 - especificaciones de formato de registro 322
 - especificaciones de salida para 308
 - ventajas 299
 - vía de acceso 323
- archivo descrito por programa
 - archivos físicos y lógicos 321
 - argumentos de búsqueda válidos 329
 - como archivo DISK 328
 - como archivo WORKSTN 379, 380, 381
- archivo descrito por programa (*continuación*)
 - definición 301
- archivo DISK
 - códigos de operación de archivo permitidos
 - para métodos de proceso por clave 346
 - para métodos de proceso sin clave 346
 - descripción general 321
 - descrito externamente
 - como descrito por programa 302
 - descripción general 322
 - ejemplos 323
 - especificaciones de formato de registro 322
 - vía de acceso 323
 - descrito por programa
 - archivo de direcciones de registros 331
 - archivo indexado 328
 - archivo secuencial 331
 - proceso 332
 - especificaciones de formato de registro 322
 - métodos de proceso
 - proceso al azar por clave 340
 - proceso consecutivo 333
 - proceso por número relativo de registro 343
 - proceso secuencial entre límites 341
 - proceso secuencial por clave 334
 - visión general 332
- archivo físico 321
- archivo físico fuente, crear 55
- archivo indexado
 - argumentos de búsqueda válidos 329
 - descripción general 328
 - vía de acceso 328
- archivo lógico
 - general 321
 - múltiples formatos 321
- archivo PRINTER
 - acceder a valor de la línea actual 363
 - códigos de operación de archivo permitidos 356
 - desbordamiento de página 356
 - indicadores de desbordamiento 356
 - lógica de búsqueda de desbordamiento 360
 - modificar control de formularios 363
 - número máximo de archivos permitidos en programa 356
 - PRTCTL (control de impresora) 363
- archivo secuencial 331
- archivo SEQ
 - códigos de operación de archivo permitidos 367
 - descripción general 366
 - diagrama de proceso 367
 - ejemplo 367
 - longitud variable 366
 - restricciones 366
- archivo SPECIAL
 - discusión general 367, 369
- archivo SPECIAL (*continuación*)
 - operaciones de archivo válidas 369
 - supresión de registros de 369
- archivo WORKSTN
 - códigos de operación de archivo permitidos con 382
 - definición 371
 - descrito externamente
 - proceso 375
 - descrito por programa
 - archivo combinado 382
 - archivo de entrada 382
 - archivo de salida 382
 - con nombre de formato 380
 - consideraciones 381
 - especificaciones de cálculo 381
 - especificaciones de entrada 381
 - especificaciones de salida 380
 - general 379
 - sin nombre de formato 381
 - descritos externamente 371
 - ejemplo de programa de búsqueda y consulta 410
 - ejemplos 387
 - indicadores de tecla de función con 374
 - múltiples dispositivos 383
 - proceso 382
 - programa de consulta de ejemplo 388
 - programa de ejemplo de proceso de subarchivo 402
 - programa de mantenimiento de datos de ejemplo 391
 - subarchivos
 - ejemplos 378
 - formato de registro 375
 - formato de registro de control 375
 - para archivo de dispositivo de pantalla 375
 - utilizaciones de 378
 - utilizar 371
 - archivos de dispositivo
 - archivos de estación de trabajo 371
 - archivos DISK 321
 - archivos PRINTER 356
 - archivos SEQ 366
 - dependencia de dispositivo 299
 - discusión general 355
 - independencia de dispositivo 299
 - múltiples dispositivos 383
- área de almacenamiento dinámico
 - área de almacenamiento dinámico por omisión 116
 - definición 115
 - ejemplo 122
- área de almacenamiento dinámico por omisión 116
- área de datos RETURNCODE 74
- áreas de datos
 - RETURNCODE 74
- argumento de búsqueda
 - archivo descrito externamente
 - descripción 326
 - referencia a una clave parcial 327
 - válido 326

- argumento de búsqueda (*continuación*)
 - archivo descrito por programa 329
- asignar almacenamiento para una matriz durante la ejecución 122
- ATTR, mandato de depuración
 - definición 196
 - ejemplo 243
 - utilizar 243
- AUT, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 467
 - mandato CRTRPGMOD 78
- autorización para mandatos ix
- ayuda para el diseño de pantallas (SDA) 108

B

- bibliografía 495
- biblioteca, crear 55
- biblioteca de pruebas, utilizar 204
- bloqueo
 - archivo 315
 - autónomo 316
 - bajo control de compromiso 348
 - lectura sin bloqueo 316
 - reintento en tiempo de espera excedido 316
 - tiempo de espera excedido del bloqueo del registros 316
 - UNLOCK 316
- bloqueo de archivos 315
- bloqueo de registros 316
- bloqueo/desbloqueo de registros 327
- BNDDIR en CRTBNDRPG, parámetro
 - CRTBNDRPG, mandato 62
 - enlace estático 64
 - mandato CRTBNDRPG 470
 - mandato CRTRPGMOD 78
- BREAK, mandato de depuración
 - definición 196
 - ejemplo 215
 - utilizar 211, 214, 218
- bucle, evitarlo en una subrutina de error 272

C

- cabeceras de página 68
- cambiar la vista de depuración de un módulo 208
- Cambiar módulo (CHGMOD), mandato 91
 - eliminar la información observable 92
- cambiar nivel de optimización de un programa o módulo 91
- Cambiar programa (CHGPGM), mandato
 - eliminar la información observable 92
 - parámetros de optimización 91
- Cambiar programa de servicio (CHGSRVPGM), mandato 101
- cambiar valores de campos al depurar 241
- cambio de nombre de campos 305

- cambio de nombre de formato de registro 304
- cambio de nombres de campo 305
- campo
 - cambiar el valor al depurar 241
 - igualar a un nombre al depurar 244
 - mantener valores actuales al depurar 196
 - visualizar al depurar
 - como valores hexadecimales 238
 - en formato de caracteres 238
 - en formato de longitud variable 239
 - en formato UCS-2 239
 - utilizar EVAL 232
 - visualizar atributos de al depurar 243
- carpetas, listado
 - básico 102
 - como recurso de mantenimiento 90
 - crear 89
 - determinar exportaciones en el programa de servicio 93
 - secciones de 89
- CCSID
 - indicado en listado del compilador 480
- CEE4ABN 160
- CEECRHP (Crear área de almacenamiento dinámico), API
 - enlazable 22
- CEECRHP (Crear Área de Almacenamiento Dinámico), API
 - enlazable 123
- CEECSZST (Reasignar almacenamiento), API
 - enlazable 23
- CEEDOD (Recuperar información del descriptor operativo) 96
 - descriptores operativos 143
 - ejemplo 143
- CEEDSHP (Descartar área de almacenamiento dinámico), API
 - enlazable 23
- CEEDSHP (Descartar Área de Almacenamiento Dinámico), API
 - enlazable 123
- CEEFRST (Liberar almacenamiento), API
 - enlazable 23
- CEEGTST (Obtener almacenamiento de área de almacenamiento dinámico), API
 - enlazable 23
- CEEGTST (Obtener Almacenamiento de Área de Almacenamiento Dinámico), API
 - enlazable 123
- CEEHDLR (Registrar Manejador de Condiciones ILE) 275
- CEEHDLU (Desregistrar Manejador de Condiciones ILE) 275
- CEERTX (Registrar procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282
- CEESGI (Obtener Información de Descripción acerca de un Argumento de Serie) 143
- CEETREC 159
- CEETSTA (Comprobar argumento omitido) 144

- CEEUTX (Procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282
- ciclo del programa
 - control de compromiso 352
 - descripción general 4
 - lógica de búsqueda de desbordamiento 360
 - último ciclo 6
- CL, mandatos
 - ADDRPLYE 111
 - Añadir Programa (ADDPGM) 205
 - autorización ix
 - CALL 105
 - Cambiar Módulo (CHGMOD) 91
 - CRTRPGMOD 78
 - CVTRPGSRC 433
 - CHGPGM 92
 - DSPMOD 154
 - DSPPGMREF 154
 - Eliminar Programa (RMVPGM) 205
 - Finalizar Depuración (ENDDBG) 202
 - Iniciar depuración (STRDBG) 202, 204
 - leer diagramas de sintaxis 455
 - mandato CRTPGM 86
 - mandatos adicionales de programa de servicio 95
 - mandatos utilizados con frecuencia 13
 - MONMSG 285
 - RCLACTGR 112
 - RCLRSC 114
 - relacionados con el programa 89
 - relacionados con módulos 84
 - UPDPGM 90
 - utilizar 455
 - Visualizar fuente de módulo (DSPMODSRC) 204, 205, 206, 207
 - WRKRPLYE 111
- CL de ILE
 - como lenguaje ILE 19
 - como módulo en programa ILE 30
 - en aplicación avanzada 32
 - en aplicación en lenguaje mixto 31
 - llamada a un programa ILE RPG 31
 - llamada a un programa RPG 27
 - manejo inesperado de excepciones de estado y notificación 285
 - método para pasar parámetros 151
- clave
 - compuesta 327
 - para un registro o un archivo 325
 - parcial 327
- clave parcial 327
- claves válidas
 - para archivo 325
 - para registros 325
- CLEAR, mandato de depuración
 - definición 196
 - eliminar todos 220
 - utilizar 211, 214, 219
- código de operación ALLOC (asignar almacenamiento) 116
- código de operación CALL (llamada a un programa)
 - en un informe de conversión 445

- código de operación CALL (llamada a un programa) *(continuación)*
 - utilizar 153
- código de operación CALLB (llamada a un procedimiento enlazado)
 - llamar a programas 153
 - utilizar 153
- código de operación CALLP (llamada a un programa o procedimiento de prototipos)
 - utilizar 138
- código de operación COMMIT (compromiso)
 - con múltiples dispositivos 349
 - consideraciones del sistema 349
 - control de compromiso 349
- código de operación DEALLOC (liberar almacenamiento) 116
- código de operación DUMP (vuelco de programa)
 - obtener un vuelco con formato 289
 - utilizar 289
- código de operación ENDSCR (finalización de subrutina)
 - especificar un punto de retorno 274
- código de operación EXFMT (grabar/a continuación, leer formato) 383
- código de operación FREE (desactivar un programa) 448
- código de operación PARM (identificar parámetros) 106
 - *OMIT 143, 144
 - reglas para especificación 155
 - utilizar 155
- código de operación PLIST (identificar una lista de parámetros) 106
 - *ENTRY PLIST 156
 - utilizar 156
- código de operación REALLOC (reasignar almacenamiento con nueva longitud) 116
- código de operación RETURN (devolver al llamador)
 - función en terminación anormal 158
 - función en terminación normal 157
 - volver sin terminar 158
- códigos de estado
 - errores de gestión de datos 426
- códigos de operación 382
 - permitido con archivo DISK 346
 - permitido con archivo PRINTER 356
 - permitido con archivo secuencial 366
 - permitido con archivo SPECIAL 369
 - permitir ampliador 'E' 262
 - permitir indicadores de error 262
 - visión general 6
- compartimiento de archivo 317
- compartimiento de una vía de acceso de datos abierta para un archivo 317
- compatibilidad con OPM, mantenimiento 65, 113
- compilación
 - crear módulos 77
 - diferencias entre ILE RPG y OPM RPG/400 423
 - en ILE 19
 - utilizar el mandato CRTBNDRPG 61
- compilador, listado
 - coordinar opciones de listado con opciones de vista de depuración 73
 - corregir errores de compilación 70
 - corregir errores durante la ejecución 72
 - especificar el formato de 68
 - examinar utilizando SEU 72
 - información por omisión 67
 - lectura 475
 - listado de ejemplo 476
 - mensajes de diagnóstico adicionales 72
 - mensajes de diagnóstico incorporado 71
 - obtener 67
 - sangrar operaciones estructuradas 69
 - secciones de 67, 476
 - utilizar 67
 - utilizar como documentación 73
- comportamiento de los módulos ILE RPG enlazados 84
- comprimir un objeto 92
- comprobación, nivel 309
- comprobación de nivel 309
- comprobación de secuencia en especificaciones de entrada 310
- Comprobar argumento omitido (CEETSTA) 144
- comprobar el número de parámetros pasados 145
- comunicación
 - acceso a otros programas y sistemas 371
- condicionamiento de salida
 - indicadores de desbordamiento 357
- consejos sobre el rendimiento
 - llamada a programa 158
 - llamada para activar LR 424
- consideraciones de rendimiento
 - subrutinas contra subprocedimientos 96
- consultar nombres de programas/procedimientos llamados 154
- control de compromiso 346
 - ámbito 348
 - bloqueos 348
 - condicional 351
 - ejemplo 350
 - en el ciclo del programa 352
 - especificación de archivos 349
 - inicio y final 347
 - operación COMMIT 349
- control de compromiso condicional, especificación 351
- conversión, análisis 444
- conversión de código manual 448
- conversión de prueba, realizar 439
- convertir a RPG IV
 - análisis de la conversión 444
 - archivo de anotaciones cronológicas 431
 - consideraciones sobre los archivos 430
 - conversión 432
- convertir a RPG IV *(continuación)*
 - convertir algunos miembros de archivo 439
 - convertir fuente desde un archivo de datos 441
 - convertir miembros fuente con SQL incluido 441
 - convertir miembros fuente del generador automático de informes 440
 - convertir todos los miembros de un archivo 439
 - ejemplo 442
 - longitud de registro de archivos 430
 - mandato CVTRPGSRC 433
 - nombres de archivo y miembro 431
 - obtener informes de conversión 440
 - problemas de conversión 448
 - realizar una conversión de prueba 439
 - requisitos 432
 - restricciones 432
 - tipos de miembros fuente válidos 430
 - utilizar el archivo de anotaciones cronológicas 447
 - utilizar un informe de errores de conversión 445
 - visión general 429
- CoOperative Development Environment/400 (CODE/400)
 - archivos de eventos 462
 - descripción 15
- coordinar opciones de listado con opciones de vista de depuración 73
- corregir errores de compilación 70
- corregir errores durante la ejecución 72
- creación de programas
 - compatible con OPM
 - crear 25
 - estrategia que debe evitar 34
- consideraciones acerca de la codificación 49, 50
- ejemplos de 63, 64, 65, 88
- estrategias para 25
 - aplicación ILE utilizando CRTRPGMOD 30
 - compatible con OPM 25
 - estrategia que debe evitar 34
 - mandato CRTPGM 86
 - utilizando CRTBNDRPG 27
 - utilizar CRTRPGMOD y CRTPGM 77
 - utilizar proceso de un solo paso 61
- Crear área de almacenamiento dinámico (CEECRHP), API enlazable 22
- Crear Área de Almacenamiento Dinámico (CEECRHP), API enlazable 123
- Crear módulo RPG (CRTRPGMOD), mandato
 - agrupación de parámetros por función 78
 - descripción de parámetros 474
 - diagrama de sintaxis 472
 - e ILE 20
 - ejemplos 99, 100

Crear módulo RPG (CRTRPGMOD),
 mandato (*continuación*)
 estrategia de creación de
 programas 30
 utilizar 78
 valores por omisión 79
 valores por omisión de
 parámetros 78
 Crear programa (CRTPGM), mandato 30
 acciones del sistema 87
 crear un programa 77
 e ILE 20
 ejemplos 100
 enlazar varios módulos 88
 parámetros 87
 utilizar 86
 Crear programa de servicio
 (CRTSRVPGM), mandato
 e ILE 20
 ejemplo 99
 parámetros 94
 Crear programa RPG enlazado
 (CRTBNDRPG), mandato
 área de datos RETURNCODE 74
 coordinar opciones de listado con
 vista de depuración 73
 crear programas 61
 descripción de parámetros 459
 diagrama de sintaxis 456
 e ILE 20
 ejemplos
 programa compatible con
 OPM 65
 programa con enlace estático 64
 programa para depuración del
 fuente 63
 estrategia de creación de
 programas 25, 27
 parámetros agrupados por
 función 62
 RETURNCODE, área de datos 74
 utilizar 61
 valores por omisión de
 parámetros 62
 crear programas
 compatible con OPM
 crear 25
 estrategia que debe evitar 34
 consideraciones acerca de la
 codificación 49, 50
 ejemplos de 63, 64, 65, 88
 estrategias para 25
 aplicación ILE utilizando
 CRTRPGMOD 30
 compatible con OPM 25
 estrategia que debe evitar 34
 mandato CRTPGM 86
 utilizando CRTBNDRPG 27
 utilizar CRTRPGMOD y
 CRTPGM 77
 utilizar proceso de un solo
 paso 61
 crear programas de servicio
 acerca de 93
 métodos 94
 crear un archivo físico fuente 55
 crear un listado del enlazador 89

crear un módulo
 utilizar CRTRPGMOD 78
 utilizar valores por omisión de
 CRTRPGMOD 79
 visión general 77
 crear un programa con el mandato
 CRTBNDRPG 61
 crear una biblioteca 55
 crear una vista de depuración
 COPY 200
 fuente raíz 199
 listado 201
 sentencia 201
 CVTOPT, parámetro
 CRTBNDRPG, mandato 62
 Mandato CRTBNDRPG 465
 mandato CRTRPGMOD 78
 CVTRPT, parámetro 437, 440, 445

D

datos de bases de datos
 actualizar al depurar 204
 datos de depuración
 crear 198
 efecto sobre el tamaño de objetos 198
 eliminar de un módulo 92
 ninguno 198
 DBCS
 consideraciones sobre depuración
 NLSS 216
 en RPG IV campos de tipo
 carácter 427
 DBGVIEW, parámetro
 coordinar con opciones de listado 73
 CRTBNDRPG, mandato 62
 Mandato CRTBNDRPG 463
 mandato CRTRPGMOD 78
 preparar un programa para
 depuración 198
 utilizar 63
 valores para ver fuente 206
 DEFINE, parámetro
 CRTBNDRPG, mandato 62
 mandato CRTBNDRPG 471
 mandato CRTRPGMOD 78
 definición de mandato 110
 depuración, vista
 cambiar al depurar 208
 definición 198
 fuente COPY 200
 fuente raíz 199
 listado 201
 sentencia 201
 valor por omisión 201
 depuración del fuente
 actualizar archivos de
 producción 204
 añadir un objeto a una sesión 204
 cambiar de módulo al depurar 207
 cambiar valores de campos 241
 consideraciones sobre NLSS 216
 coordinar con opciones de listado 73
 crear un programa para
 depuración 63
 descripción general 195

depuración del fuente (*continuación*)
 diferencias entre ILE RPG y OPM
 RPG/400 424
 efectos de la optimización 91, 196
 eliminar un objeto de una
 sesión 204, 206
 establecer condiciones de
 observación 221
 establecer opciones de
 depuración 204
 establecer y eliminar puntos de
 interrupción 209
 funciones incorporadas
 %ADDR 240
 %INDEX 240
 %SUBSTR 240
 %VARS 240
 cambiar valores utilizando
 %SUBSTR 242
 descripción general 240
 ejemplos 240
 iniciar el depurador del fuente 202
 límite de programas OPM en sesión
 de depuración 205
 normas para asignar valores
 utilizando EVAL 242
 obtener un vuelco con formato 289
 preparar un programa 198
 resultados inesperados 234
 seguir los pasos 227
 Soporte de Idiomas Nacionales 245
 ver fuente 206
 ver nombres abreviados 245
 visión general 22
 visualizar atributos de 243
 visualizar campos como valores
 hexadecimales 238
 visualizar campos en formato de
 caracteres 238
 visualizar campos en formato de
 longitud variable 239
 visualizar campos en formato
 UCS-2 239
 visualizar datos direccionados por
 punteros 239
 visualizar datos y expresiones 232
 visualizar el contenido de una
 matriz 235
 visualizar el contenido de una
 tabla 235
 visualizar estructuras de datos de
 múltiples apariciones 236
 visualizar indicadores 237
 depurador del fuente ILE
 descripción 195
 inicio 202
 mandatos de depuración 196
 depurar
 actualizar archivos de
 producción 204
 añadir un objeto a una sesión 204
 cambiar de módulo al depurar 207
 cambiar valores de campos 241
 consideraciones sobre NLSS 216
 coordinar con opciones de listado 73
 crear un programa para
 depuración 63

- depurar (*continuación*)
 - descripción general 195
 - diferencias entre ILE RPG y OPM RPG/400 424
 - efectos de la optimización 91, 196
 - eliminar un objeto de una sesión 204, 206
 - establecer condiciones de observación 221
 - establecer opciones de depuración 204
 - establecer y eliminar puntos de interrupción 209
 - funciones incorporadas
 - %ADDR 240
 - %INDEX 240
 - %SUBSTR 240
 - %VARS 240
 - cambiar valores utilizando %SUBSTR 242
 - descripción general 240
 - ejemplos 240
 - iniciar el depurador del fuente 202
 - límite de programas OPM en sesión de depuración 205
 - normas para asignar valores utilizando EVAL 242
 - obtener un vuelco con formato 289
 - preparar un programa 198
 - resultados inesperados 234
 - seguir los pasos 227
 - Soporte de Idiomas Nacionales 245
 - ver fuente 206
 - ver nombres abreviados 245
 - visión general 22
 - visualizar atributos de 243
 - visualizar campos como valores hexadecimales 238
 - visualizar campos en formato de caracteres 238
 - visualizar campos en formato de longitud variable 239
 - visualizar campos en formato UCS-2 239
 - visualizar datos direccionados por punteros 239
 - visualizar datos y expresiones 232
 - visualizar el contenido de una matriz 235
 - visualizar el contenido de una tabla 235
 - visualizar estructuras de datos de múltiples apariciones 236
 - visualizar indicadores 237
- desbloqueo/bloqueo de registros 327
- desbordamiento
 - indicadores 357
 - página 356
- desbordamiento de página, en archivo PRINTER 356
- Descartar área de almacenamiento dinámico (CEEDSHP), API enlazable 23
- Descartar Área de Almacenamiento Dinámico (CEEDSHP), API enlazable 123
- descomprimir un objeto 92

- descripción de parámetros
 - Mandato CRTBNDRPG 459
 - mandato CRTRPGMOD 474
 - mandato CVTRPGSRC 435
- descriptores operativos
 - definición 142
 - ejemplo 96
- DETAIL, parámetro
 - crear un listado del enlazador 89
- DETC 274
- detectar errores en un programa 195
- DETL 274
- DFTACTGRP en CRTBNDRPG, parámetro
 - CRTBNDRPG, mandato 62
 - descripción 460
 - ejecutar en OPM por omisión 113
 - utilizar 61, 64, 65
- diagnosticar errores en un programa 195
- diagrama de flujo
 - lógica de búsqueda de desbordamiento 360
- diagramas de sintaxis
 - interpretar 455
 - Mandato CRTBNDRPG 456
 - mandato CRTRPGMOD 472
 - mandato CVTRPGSRC 434
- diferencias de compatibilidad entre OPM RPG/400 y ILE RPG 423
- diferencias de comportamiento entre OPM RPG/400 y ILE RPG 423
- Diferencias de E/S entre ILE RPG y OPM RPG/400 425
- diferencias entre OPM e ILE RPG
 - diferencias de comportamiento 423
 - manejo de excepciones 256
- diferentes vistas de un módulo 208
- directivas del compilador
 - cambiar una cabecera del listado 68
- DISPLAY, mandato de depuración
 - definición 196
 - utilizar 207
 - ver nombres abreviados 245
- dispositivos
 - WORKSTN 371
- documentación de programas 73

E

- ejecución, gestionar el almacenamiento en tiempo de 115
- ejecutar un programa
 - desde una aplicación dirigida por un menú 107
 - diferencias entre ILE RPG y OPM RPG/400 424
 - en el grupo de activación OPM por omisión 113
 - utilización de un mandato creado por el usuario 110
 - utilizando el mandato CL CALL 105
 - visión general 105
- ejemplos
 - aplicación interactiva 387
 - compilación
 - ejemplo de listado del enlazador 102

- ejemplos (*continuación*)
 - compilación (*continuación*)
 - enlazar varios módulos 88
 - programa compatible con OPM 65
 - programa con enlace estático 64
 - programa de servicio 96
 - programa para depuración del fuente 63
 - convertir a RPG IV
 - algunos miembros de un archivo 439
 - conversión de ejemplo 442
 - realizar una conversión de prueba 439
 - todos los miembros de un archivo 439
 - depurar
 - añadir un programa de servicio a una sesión 205
 - cambiar la vista de depuración de un módulo 208
 - cambiar valores de campos 242
 - eliminar programas de una sesión 206
 - establecer opciones de depuración 204
 - establecer un punto de interrupción condicional 214
 - establecer un punto de interrupción incondicional 211
 - fuentes para ejemplos de depuración 245
 - utilizar %SUBSTR para visualizar valores de campos 240
 - ver un módulo distinto en una sesión de depuración 207
 - visualizar atributos de un campo 243
 - visualizar campos como valores hexadecimales 238
 - visualizar campos en formato de caracteres 238
 - visualizar campos en formato de longitud variable 239
 - visualizar campos en formato UCS-2 239
 - visualizar datos direccionados por punteros 239
 - visualizar el contenido de una matriz 235
 - visualizar el contenido de una tabla 235
 - visualizar estructuras de datos de múltiples apariciones 236
 - visualizar indicadores 237
- E/S
 - consulta por código postal y búsqueda por nombre 410
 - mantenimiento de datos 391
 - proceso de subarchivo 402
 - programa de consulta 388
 - gestionar el propio almacenamiento dinámico 122
 - llamada de programa/procedimiento
 - comprobar el número de parámetros pasados 145

- ejemplos (*continuación*)
 - utilizar parámetros omitidos 96
- manejar excepciones
 - error de función no manejado 259
 - evitar un bucle en una subrutina de error 272
 - manejador de cancelación 282
 - mensaje de escape no manejado 258
 - subrutina de error *PSSR 269
 - subrutina de error de archivo 266
 - utilizar un manejador de cancelación 283
 - utilizar un manejador de condiciones 275
- módulo con múltiples procedimientos 45
- pasar parámetros utilizando el mandato CL CALL 106
- programa ILE RPG de ejemplo 7
- subprocedimientos 40
- creación de un módulo NOMAIN 79
- eliminar errores en un programa 195
- eliminar la información observable 92
- eliminar objetos de una sesión de depuración 204
- eliminar puntos de interrupción
 - acerca de 209
 - puntos de interrupción de hebra condicionales 220
 - puntos de interrupción de hebra incondicionales 212
 - puntos de interrupción de trabajo condicionales 213
 - puntos de interrupción de trabajo incondicionales 210
 - todos 220
 - utilizar números de sentencia 217
- ENBPFRCOL, parámetro
 - CRTBNDRPG, mandato 62
 - mandato CRTBNDRPG 470
 - mandato CRTRPGMOD 78
- enlazar
 - definición 84
 - después de modificar un módulo 90
 - módulos a un programa 84
 - programa de servicio a programa 100
- enlazar varios módulos 88
- ENTMOD, parámetro 86
- Entorno de lenguajes integrados (ILE)
 - consideraciones sobre las llamadas entre lenguajes 152
 - crear programas 19
 - en vigor
 - programa compatible con OPM 26
 - programa utilizando CRTBNDRPG 28
 - estrategias de creación de programas 25, 27, 30
 - estructura interna de programa 85
 - familia de compiladores de ILE 19
 - finalización de un programa ILE 111
 - gestión de programas 21
 - llamada a programas 21
- Entorno de lenguajes integrados (ILE) (*continuación*)
 - llamadas entre lenguajes 151
 - visión general 19
- entrada
 - archivo 382
- EQUATE, mandato de depuración
 - definición 196
 - ejemplo 244
 - utilizar 244
- error de función
 - definición 252
 - no manejado 259
- error de función no manejado 259
- errores
 - archivo 254
 - corregir compilación 70
 - corregir durante la ejecución 72
 - programa 254
- errores de compilación, corregir 70
- errores de enlace del listado del compilador 489
- errores de generación de código del listado del compilador 489
- errores durante la ejecución, corregir con un listado del compilador 72
- especificaciones
 - archivo descrito externamente 304
 - descripción de 3
 - descripción de archivo 304
 - formato de registro 322
 - orden 3
 - tipos 3
- especificaciones de cálculo
 - archivo WORKSTN descrito por programa 381
 - descripción general 3
- especificaciones de control
 - consideraciones para la conversión 432
 - descripción general 3
 - ejemplo 8
- especificaciones de definición
 - descripción general 3
- especificaciones de descripción de archivo
 - control de compromiso 349
 - descripción general 3
 - para archivos descritos externamente 304
- especificaciones de entrada
 - descripción general 3
- especificaciones de extensión
 - problemas de conversión 443, 451
- especificaciones de salida
 - archivo WORKSTN descrito por programa 380
 - con descripciones externas 308
 - descripción general 3
 - ejemplo 10
- especificar el formato del listado del compilador 68
- especificar indicadores de error 261
- especificar un grupo de activación 112
- especificar un punto de retorno 274
- establecer opciones de depuración 204
- establecer pasos al depurar
 - en un programa 227
- establecer pasos al depurar (*continuación*)
 - en un programa o procedimiento 228
 - un programa o procedimiento 228
- establecer puntos de interrupción
 - acerca de 209
 - ejemplo 211, 214
 - puntos de interrupción de hebra condicionales 220
 - puntos de interrupción de hebra incondicionales 212
 - puntos de interrupción de trabajo condicionales 213
 - puntos de interrupción de trabajo incondicionales 210
 - utilizar números de sentencia 217
- estrategias para crear programas ILE 25
- estructura de datos de estado del programa
 - ejemplo 155, 269
 - utilizar en una subrutina de error 269
- estructura de datos de información de archivos
 - ejemplo 266
 - utilizar en una subrutina de error 265
- estructuras de datos
 - apariciones múltiples
 - visualizar al depurar 236
 - subcampos
 - problemas de conversión 453
 - visualizar al depurar 236
 - utilizar mandato de depuración EVAL 236
- EVAL, mandato de depuración
 - cambiar valores 241
 - contenido de una matriz 235
 - contenido de una tabla 235
 - definición 196
 - ejemplo 233, 242
 - en formato de caracteres 238
 - en formato de longitud variable 239
 - en formato UCS-2 239
 - indicadores 237
 - normas para asignar valores 242
 - utilizar 232
 - visualizar estructuras de datos 236
- evitar un bucle en una subrutina de error 272
- examinar un listado del compilador utilizando SEU 72
- excepción
 - anidadas, 257
 - supervisar durante la ejecución 111
- excepción/error de archivos
 - definición 254
 - ejemplo 266
 - utilizar una subrutina INFSR 265
- excepción/errores del programa
 - definición 254
 - ejemplo 269, 275
 - evitar un bucle 272
 - utilizar una subrutina *PSSR 269
- excepciones anidadas 257
- excepciones/errores, manejar
 - consideraciones generales 257

- excepciones/errores, manejar (*continuación*)
 - consideraciones sobre optimización 260
 - diferencias entre ILE RPG y OPM RPG/400 256, 424
 - especificar un punto de retorno 274
 - específico de RPG 254
 - evitar un bucle 272
 - filtro 252
 - indicadores de error 261
 - manejador de cancelación 282
 - manejador de condiciones 275
 - no manejadas 257
 - palabra clave NOOPT 260
 - subrutina de error *PSSR 269
 - subrutina de error/excepción de archivo (INFSR) 265
 - tipos de 251
 - utilizar el ampliador 'E' 261
 - visión general 252
 - visión general de subrutina de error/excepción 264
- excepciones no manejadas 257
- EXPCPY, parámetro 437
- EXPORT, palabra clave
 - nombres duplicados 87
- expresiones
 - devolver valores 138

F

- filtro de una excepción 252
- final del control de compromiso 347
- finalizar un programa o procedimiento
 - después de llamada al sistema 111
 - terminación anormal 158
 - terminación normal 157
 - utilizando API enlazables 159
 - visión general, volver de 156
 - volver sin terminar 158
- FIND, mandato de depuración 197
- FIXNBR, parámetro
 - CRTBNDRPG, mandato 62
 - mandato CRTBNDRPG 468
 - mandato CRTRPGMOD 78
- formato de caracteres
 - CCSID de carácter
 - indicado en listado del compilador 480
 - visualizar al depurar 238
- formato de entero
 - parámetro TRUNCNBR 468
- formato de entero sin signo
 - parámetro TRUNCNBR 468
- formato de gráfico
 - CCSID de gráfico
 - indicado en listado del compilador 480
 - consideraciones sobre depuración NLSS 216
 - normas para asignar valores
 - utilizando EVAL 242
- formato de longitud variable
 - visualizar al depurar 239
- formato de registro
 - cambio de nombre 304

- formato de registro (*continuación*)
 - especificaciones para archivos descritos externamente 322
 - ignorar 305
 - para un subarchivo 376
- formato de registro de control, subarchivo 375
- formato del listado del compilador, especificar 68
- formato UCS-2
 - CCSID UCS-2
 - indicado en listado del compilador 480
 - visualizar al depurar 239
- FROMFILE, parámetro 435
- FROMMBR, parámetro 435, 439
- fuelle desde un archivo de datos, convertir 441
- función de comunicaciones intersistemas (ICF) 371
- función incorporada %ALLOC 116
- función incorporada %REALLOC 116
- funciones del sistema
 - spooling 319
- funciones incorporadas
 - %ADDR 144
- funciones JNI, empaquetadores para las 180

G

- GDDM 161
- GENLVL, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 460
 - mandato CRTRPGMOD 78
- gestión de almacenamiento
 - almacenamiento dinámico 115
 - asignar durante la ejecución 122
 - gestionar durante la ejecución 115
- gestión de datos distribuidos (DDM)
 - archivos 352
- gestión de programas 21
- gestionar almacenamiento asignado dinámicamente 115
- gestionar almacenamiento durante la ejecución 115
- gestionar el área de almacenamiento dinámico por omisión utilizando operaciones RPG 116
- gestionar grupos de activación 112
- Gestor de Representación Gráfica de Datos (GDDM) 161
- grupo de activación
 - *CALLER 113
 - ejecutar en OPM por omisión 113
 - especificar 113
 - *NEW 86, 159
 - especificar 112
 - finalizar 112
 - con nombre 86
 - especificar 112
 - supresión 112
 - definición 112
 - función en manejo de excepciones 252
 - gestionar 112

- grupo de activación (*continuación*)
 - identificar 86, 112
 - OPM por omisión 113
 - QILE 86, 112
 - supresión 114
- grupo de activación con nombre 112
- grupo de activación OPM por omisión 25, 34
 - ejecutar en 113
- grupo de activación por omisión 25, 34, 113
 - ejecutar en 113
- grupo MONITOR 262
- grupo ON-ERROR 262

- identificar un grupo de activación 112
- ignorar formato de registro 305
- igualar un nombre a un campo, expresión o mandato 244

ILE C

- como lenguaje ILE 19
- en aplicación avanzada 32
- en aplicación en lenguaje mixto 31
- fuelle del módulo en ejemplo de depuración 249
- método para pasar parámetros 151

ILE COBOL

- como lenguaje ILE 19
- método para pasar parámetros 151

ILE RPG

- comportamiento de los módulos
 - enlazados 84
- convertir a 429
- diagrama de lógica 5
- diferencias de comportamiento entre OPM RPG/400 423
- operaciones de gestión de datos 310
- programa de ejemplo 7
- tipos de dispositivo a los que se da soporte 355
- visión general del lenguaje RPG IV 3
- visión general del manejo de excepciones 254
- incluir vista del fuelle, crear 200
- INDENT, parámetro 200
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 464
 - mandato CRTRPGMOD 78
- indicador de retorno (RT)
 - utilizado para finalizar un programa o procedimiento 157, 158
- indicador de último registro (LR)
 - utilizado para finalizar un programa o procedimiento 157, 158
- indicadores
 - como indicadores de error 261
- desbordamiento
 - con archivo PRINTER 356
- descripción general 356
- ejemplos 360
- lógica de búsqueda de desbordamiento 360
- presencia o ausencia de 358
- relación con el ciclo del programa 360

- indicadores (*continuación*)
 - valor de 360
 - error 261
 - parada (H1-H9)
 - utilizado para finalizar un programa o procedimiento 157, 158
 - retorno (RT)
 - utilizado para finalizar un programa o procedimiento 157, 158
 - tecla de función (KA-KN, KP-KY)
 - con archivo WORKSTN 374
 - último registro (LR)
 - descripción general 6
 - utilizado para finalizar un programa o procedimiento 157, 158
 - utilizar 6
 - visualizar al depurar 237
- indicadores 01-99
 - en un ejemplo de vuelco con formato 295
 - visualizar al depurar 237
- indicadores de desbordamiento
 - con archivo PRINTER 356
 - condicionamiento de salida 357
 - descripción general 357
 - ejemplos 360
 - lógica de búsqueda de desbordamiento 360
 - presencia o ausencia de 358
 - relación con el ciclo del programa 360
 - valor de 360
- indicadores de error
 - especificar 261
- indicadores de parada (H1-H9)
 - utilizado para finalizar un programa o procedimiento 157, 158
- indicadores resultantes (01-99, H1-H9, OA-OG, OV, L1-L9, LR, U1-U8, KA-KN, KP-KY, RT)
 - como indicadores de error 261
- información en línea
 - para depurador del fuente ILE 197
 - para los mandatos de crear 459
- información observable de módulo 92
- información sobre campos de clave del listado del compilador 485
- informes de conversión
 - obtener 440
 - secciones de 445
 - utilizar 445
- Iniciar depuración (STRDBG),
 - mandato 202
 - parámetro Actualizar archivos de producción (UPDPDPROD) 204
- iniciar el depurador del fuente ILE 202
- inicio del control de compromiso 347
- insertar plantillas de
 - especificaciones 441
- INSRTPL, parámetro 437, 441
- interfaz de programación de aplicaciones (API)
 - llamar a una no enlazable 132
 - QMHSNDPM 424

- interfaz de programación de aplicaciones (API) (*continuación*)
 - Recuperar Mensaje (QMHRVTM), API 162
- interrupción de control
 - ejemplo 359

J

- Java 180
 - llamada a RPG desde Java 177
 - llamar a Java desde RPG 172
 - métodos nativos 177
- Java Virtual Machine (JVM) 180
 - juego de caracteres de doble byte
 - consideraciones sobre depuración NLSS 216
 - en RPG IV campos de tipo carácter 427

L

- LANGID, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 466
 - mandato CRTRPGMOD 78
- leer registro siguiente
 - con subarchivo WORKSTN 377
- leer un registro 383
- lenguaje de carpetas
 - ejemplo 99
 - razones para utilizar 94
- lenguajes, ILE 19
- liberación de un registro bloqueado 317
- Liberar almacenamiento (CEEFRST), API
 - enlazable 23
- liberar recursos de programas ILE 114
- LICOPT, parámetro
 - mandato CRTBNDRPG 471
- límite de control 252
- lista de parámetros
 - creada por PARM 156
 - identificar 136
 - reglas para especificación 156
- lista de referencias externas del listado del compilador 487
- lista de respuestas de mensajes
 - añadir a 110
 - cambiar 111
- lista de respuestas del sistema
 - añadir a 110
 - cambiar 111
- listado de carpetas
 - básico 102
 - como recurso de mantenimiento 90
 - crear 89
 - determinar exportaciones en el programa de servicio 93
 - secciones de 89
- listado de referencias cruzadas 486
- listado del compilador
 - coordinar opciones de listado con opciones de vista de depuración 73
 - corregir errores de compilación 70
 - corregir errores durante la ejecución 72

- listado del compilador (*continuación*)
 - especificar el formato de 68
 - examinar utilizando SEU 72
 - información por omisión 67
 - lectura 475
 - listado de ejemplo 476
 - mensajes de diagnóstico
 - adicionales 72
 - mensajes de diagnóstico
 - incorporado 71
 - obtener 67
 - sangrar operaciones estructuradas 69
 - secciones de 67, 476
 - utilizar 67
 - utilizar como documentación 73
- LOGFILE, parámetro 437
- LOGMBR, parámetro 438
- longitud de registro de archivos,
 - consideraciones acerca de la conversión 430
- longitud de registro de un archivo,
 - consideraciones acerca de la conversión 430

LL

- llamada de programa/procedimiento
 - devolver valores 138
 - en ILE 21
- llamada de formato libre 138
- llamadas de formato fijo 153
- llamadas entre lenguajes 151
- llamadas estáticas 132
- llamadas recurrentes 134
- llamar a API enlazables 160
- llamar a gráficos 161
- llamar a procedimientos 132
- llamar a programas 132
- llamar a rutinas especiales 162
- métodos para pasar parámetros 139
- pila de llamadas 133
- terminación anormal de programa o procedimiento 158
- terminación normal de programa o procedimiento 157
- utilizar la operación CALL 153
- utilizar la operación CALLB 153
- visión general 131
- volver de un programa o procedimiento llamado 156
- volver sin terminar 158
- llamada de prototipos
 - orden de evaluación de parámetros 151
- llamada estática a procedimiento 132
- llamadas, pila 133, 252
- llamadas a punteros de procedimientos 132
- llamadas dinámicas 21, 132
- llamadas entre lenguajes 151
- llamadas estáticas 22, 132
- llamar
 - Java desde RPG 172
 - RPG desde Java 177
- llamar a programas/procedimientos
 - devolver valores 138
 - en ILE 21

- llamar a programas/procedimientos (continuación)
 - llamada de formato libre 138
 - llamadas de formato fijo 153
 - llamadas entre lenguajes 151
 - llamadas estáticas 132
 - llamadas recurrentes 134
 - llamar a API enlazables 160
 - llamar a gráficos 161
 - llamar a procedimientos 132
 - llamar a programas 132
 - llamar a rutinas especiales 162
 - métodos para pasar parámetros 139
 - pila de llamadas 133
 - terminación anormal de programa o procedimiento 158
 - terminación normal de programa o procedimiento 157
 - utilizar la operación CALL 153
 - utilizar la operación CALLB 153
 - visión general 131
 - volver de un programa o procedimiento llamado 156
 - volver sin terminar 158
- llamar a una rutina de gráficos 161

M

- mandato Borrar 374
- mandato CL CALL
 - ejecutar un programa 105
 - ejemplo de pasar parámetros 106
 - pasar parámetros 105
- mandato creado por el usuario, ejecutar un programa RPG 110
- mandato CRTRPTPGM (crear programa generador automático de informes)
 - convertir miembros del generador automático de informes 440
- mandato CVTRPGSRC (Convertir RPG fuente)
 - descripción de parámetros 435
 - diagrama de sintaxis 434
 - ejemplo 439
 - utilizar los valores por omisión de mandatos 438
 - valores por omisión del parámetro 433
- mandato editar fuente (STRSEU) 56
- mandato Finalizar depuración (ENDDBG) 202
- mandato Visualizar módulo (DSPMOD) 154
- mandato Visualizar programa (DSPPGM)
 - determinar nivel de optimización 91
- mandato Visualizar programa de servicio (DSPSRVPGM) 93
- mandatos de depuración
 - ATTR 243
 - CLEAR 211
 - descripción general 196
 - DISPLAY 207
 - EQUATE 244
 - EVAL 232, 241
 - igualar a un nombre al depurar 244
 - STEP 227
 - STEP INTO 228

- mandatos de depuración (continuación)
 - STEP OVER 228
 - WATCH 221
- manejador de cancelación 251
 - CEERTX (Registrar procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282
 - CEEUTX (Procedimiento de salida de usuario de finalización de entrada de pila de llamadas) 282
 - ejemplo 283
 - utilizar 282
- manejador de condiciones 251
 - ejemplo 275
 - filtrar una excepción 276
 - llamadas recurrentes 275
 - registrar 275
 - visión general 275
- manejador de excepciones
 - específico de RPG 254, 261
 - prioridad de 257
- manejador de excepciones por omisión, RPG 254
- manejo de excepción/error
 - grupo MONITOR 262
- manejo de excepciones/errores
 - consideraciones generales 257
 - consideraciones sobre optimización 260
 - diferencias entre ILE RPG y OPM RPG/400 256, 424
 - especificar un punto de retorno 274
 - específico de RPG 254
 - evitar un bucle 272
 - excepciones de JAVA 185
 - filtro 252
 - indicadores de error 261
 - manejador de cancelación 282
 - manejador de condiciones 275
 - no manejadas 257
 - palabra clave NOOPT 260
 - subrutina de error *PSSR 269
 - subrutina de error/excepción de archivo (INFSR) 265
 - tipos de 251
 - utilizar el ampliador 'E' 261
 - visión general 252
 - visión general de subrutina de error/excepción 264
- mantenimiento de compatibilidad con OPM 65, 113
- matriz
 - carga 454
 - matrices en tiempo de preejecución 454
 - problemas de conversión 453
 - visualizar al depurar 235
- matriz de tiempo de ejecución
 - asignar almacenamiento durante la ejecución 122
- matriz dinámica
 - asignar almacenamiento durante la ejecución 122
- matriz o tabla de tiempo de compilación
 - sección del listado del compilador 485
- MCH3601 425

- memoria teraespacio 153
- mensaje de escape no manejado 258
- mensajes
 - consulta
 - responder 110
 - diagnóstico adicional 72
 - diagnóstico incorporado 71
 - excepción
 - ejemplo 258
 - no manejada 257
 - tipos de 252
- mensajes de consulta
 - lista de 110
 - responder 110
- mensajes de consulta durante la ejecución, responder 110
- mensajes de escape
 - definición 252
 - no manejado 258
- mensajes de excepción
 - filtro 252
 - manejados inesperadamente por CL MONMSG 285
 - no manejada 257
 - tipos de 252
- métodos de Java 172
- métodos de proceso
 - al azar por clave 339
 - archivo WORKSTN 375, 382
 - consecutivo 333
 - número relativo de registro 343
 - para archivo DISK 332
 - secuencial entre límites 341
 - secuencial por clave 334
 - sin clave 344
 - sólo secuencial 334, 344
- migrar a ILE RPG 429
- modificar un módulo 90
- modificar un programa 90
- modificar un programa de servicio 95
- MODULE, parámetro 86
 - Mandato CRTBNDRPG 459
 - mandato CRTRPGMOD 78
- módulo
 - acerca de 77
 - cambiar al depurar 207
 - comportamiento de los ILE RPG enlazados 84
 - creación de un módulo NOMAIN 79
 - crear 77
 - determinar el módulo de entrada 86
 - diferentes vistas de depuración 208
 - efecto de los datos de depuración sobre el tamaño 198
 - eliminar la información observable 92
 - enlazar a un programa 84
 - enlazar varios 88
 - información del listado de vuelco 289
 - mandato CRTRPGMOD 78
 - mandatos CL relacionados 84
 - modificación del nivel de optimización 91
 - modificar y volver a enlazar 90
 - preparar para depuración 198
 - reducir tamaño 92

módulo (*continuación*)
 relación con el programa 85
 sustituir en un programa 90
 ver fuente al depurar 206
 visión general del módulo de
 múltiples procedimientos 35
 módulo de entrada 30, 86
 módulo libre de ciclos 79
 módulo NOMAIN
 consideraciones acerca de la
 codificación 50
 crear 79
 módulos, creación
 utilizar CRTRPGMOD 78
 utilizar valores por omisión de
 CRTRPGMOD 79
 visión general 77
 MQSeries 167
 múltiples dispositivos conectados a
 programa de aplicación 350

N

nombre de dispositivo, función del 300
 nombre de formato 380
 nombre del programa
 parámetro *FROMMBR 436
 nombres largos
 en listado del compilador 486
 NOOP, palabra clave
 mantener valores actuales al
 depurar 196
 nivel de optimización de
 programa 91
 y manejo de excepciones 260
 NOT
 Diferencias de funcionamiento entre
 ILE RPG y RPG/400 423
 número de página, en archivo
 PRINTER 356
 número relativo de registro 332

O

Obtener almacenamiento de área de
 almacenamiento dinámico (CEEGETST),
 API enlazable 23
 Obtener Almacenamiento de Área de
 Almacenamiento Dinámico (CEEGETST),
 API enlazable 123
 Obtener Información de Descripción
 acerca de un Argumento de Serie
 (CEESGI) 143
 obtener informes de conversión 440
 obtener un listado del compilador 67
 OFL 274
 operaciones de archivo
 permitido con archivo DISK 346
 permitido con archivo PRINTER 356
 permitido con archivo secuencial 367
 permitido con archivo SPECIAL 369
 permitido con archivo
 WORKSTN 382
 operaciones de archivo válidas
 archivo SPECIAL 369
 operaciones de gestión de memoria
 código de operación ALLOC (asignar
 almacenamiento) 116
 código de operación DEALLOC
 (liberar almacenamiento) 116
 código de operación REALLOC
 (reasignar almacenamiento con
 nueva longitud) 116
 función incorporada %ALLOC 116
 función incorporada
 %REALLOC 116
 operaciones de llamada
 consultar nombres de procedimientos
 llamados 154
 DSPPGMREF 154
 llamada de formato libre 138
 llamadas de formato fijo 153
 llamar a programas 153
 rutinas especiales 162
 utilizar 138
 operaciones estructuradas
 sangrar 69
 operativos, descriptores
 definición 142
 ejemplo 96
 optimización
 consideraciones acerca del manejo de
 excepciones 260
 definición 91
 efecto sobre los campos al
 depurar 196
 nivel de
 comprobar 91
 de un objeto, cambiar 91
 OPTIMIZE, parámetro
 CRTBNDRPG, mandato 62
 Mandato CRTBNDRPG 464
 mandato CRTRPGMOD 78
 OPTION, parámetro
 coordinar con opciones de vista de
 depuración 73
 coordinar listado y opciones de vista
 de depuración 200
 CRTBNDRPG, mandato 62
 Mandato CRTBNDRPG 461
 mandato CRTRPGMOD 78
 utilizar 67, 73
 OPTIONS, palabra clave
 *NOPASS 144
 *OMIT 144
 orden de clasificación
 consideraciones sobre
 depuración 216
 efecto en los parámetros SRTSEQ 320
 tabla ALTSEQ del listado del
 compilador 485
 orden de clasificación alternativo
 consideraciones sobre
 depuración 216
 orden de evaluación
 en llamada de prototipos 151
 OUTPUT, parámetro
 CRTBNDRPG, mandato 62
 Mandato CRTBNDRPG 464
 mandato CRTRPGMOD 78
 utilizar 67

P

palabra clave IGNORE 305
 palabra clave PREFIX 305
 palabra clave RENAME 304
 palabras clave
 *OMIT 144
 DDS 321
 EXPORT 87
 NOOPT 91, 260
 para archivo de dispositivo de
 pantalla
 CLEAR 374
 HELP 374
 HOME 374
 PRINT 374
 ROLLDOWN 374
 ROLLUP 374
 para línea de continuación 321
 CLEAR 374
 HELP 374
 HOME 374
 PRINT 374
 ROLLDOWN 374
 ROLLUP 374
 palabras clave de especificación de
 control
 palabras clave de opción de
 compilación
 ejemplo del listado del
 compilador 476
 palabras reservadas
 *CANCL 274
 *DETC 274
 *DETL 274
 *GETIN 274
 *OFL 274
 *TOTC 274
 *TOTL 274
 parámetro de área para SPECIAL
 PLIST 368
 parámetro de estado de retorno 368
 parámetros
 comprobar número de los que se han
 pasado 145
 descriptores operativos 142
 especificar 155
 omitidos 143
 pasar 136
 pasar utilizando el mandato CL
 CALL 105
 requisitos de coincidencia de tipo de
 datos 142
 parámetros, descripciones
 Mandato CRTBNDRPG 459
 mandato CRTRPGMOD 474
 mandato CVTRPGSRC 435
 parámetros, tabla
 CRTBNDRPG, mandato 62
 mandato CRTRPGMOD 78
 mandato CVTRPGSRC 433
 parámetros omitidos 143
 *OMIT 144
 partes de un programa ILE RPG 7
 pasar parámetros
 comprobar número de los que se han
 pasado 145
 descriptores operativos 142

- pasar parámetros (*continuación*)
 - ejemplo 106
 - métodos para lenguajes ILE 151
 - parámetros omitidos 143
 - pasar menos datos 150
 - por referencia 140
 - por referencia de sólo lectura 141
 - por valor 140
 - requisitos de coincidencia de tipo de datos 142
 - utilizando el mandato CL CALL 105
 - utilizar PARM 155
 - utilizar PLIST 156
 - visión general 136
- perceptibilidad 92
- PGM, parámetro
 - CRTBNDRPG, mandato 62
- pila de llamadas 133, 252
- plantillas de especificaciones, insertar 441
- plantillas, insertar especificaciones 441
- posiciones decimales
 - especificaciones de entrada
 - archivo WORKSTN descrito por programa 381
 - con descripciones externas 306
- posiciones del almacenamiento
 - intermedio de salida, en el listado del compilador 484
- preparar un programa para depuración 198
- prevención de impresión sobre agujeros 360
- PRFDTA, parámetro
 - CRTBNDRPG, mandato 62
 - eliminar la información observable 92
 - mandato CRTBNDRPG 471
 - mandato CRTRPGMOD 78
- probar puntos de interrupción 210
- procedimiento
 - información de vuelco 289
 - llamada a puntero de procedimiento 132
 - llamada estática a procedimiento 132
 - llamar 131
 - pasar parámetros 136
 - saltar 228
 - terminación anormal 158
 - terminación normal 157
 - volver de 156
 - volver sin terminar 158
- procedimiento de entrada de programa (PEP)
 - definición 84
 - determinar 86
 - y la pila de llamadas 133
- procedimiento de entrada de usuario (UEP)
 - función en el programa 85
 - y la pila de llamadas 133
- Procedimiento de salida de usuario de finalización de entrada de pila de llamadas (CEEUTX) 282
- procedimiento principal
 - ámbito de archivos 84
- procedimiento principal (*continuación*)
 - consideraciones acerca de la codificación 50
 - visión general 35
 - volver de 157
- proceso al azar por clave
 - discusión general 339
 - ejemplo 340
- proceso consecutivo 333
- proceso de creación de programa de dos pasos 77
- proceso de creación de un programa de un solo paso 61
- proceso por clave
 - archivo de límites de direcciones de registros 331
 - archivo indexado 328
 - secuencial entre límites 341
 - vía de acceso 323
- proceso por número relativo de registro 343
- proceso secuencial entre límites
 - discusión general 341
 - ejemplos 342
- proceso secuencial por clave
 - discusión general 334
 - ejemplos 334
- proceso sin clave 344
- proceso sólo secuencial 333, 334
- programa
 - actualización 90
 - cambiar al depurar 207
 - compatible con OPM
 - efecto de ILE 26
 - ejemplo 26
 - estrategia de creación de programas 25, 34
 - método de creación 25
 - diferentes vistas de depuración 208
 - efecto de los datos de depuración sobre el tamaño 198
 - ejecución utilizando un mandato creado por el usuario 110
 - ejecutar 105
 - ejecutar desde una aplicación dirigida por menú 107
 - ejecutarse en el grupo de activación OPM por omisión 113
 - ejemplo 7
 - eliminar la información observable 92
 - enlazar módulos 84
 - entrar en 228
 - entrar fuente 55
 - entrar sentencias SQL 59
 - establecer condiciones de observación 221
 - estructura interna 85
 - finalizar 111
 - ILE avanzado 32
 - lenguaje mixto 31
 - lenguaje único 31
 - efecto de ILE 28
 - liberar recursos 114
 - llamada utilizando la operación CALLP 138
 - llamar 131, 132
- programa (*continuación*)
 - llamar utilizando expresiones 138
 - llamar utilizando la operación CALL 153
 - mandatos CL relacionados 89
 - modificación del nivel de optimización 91
 - modificar 90
 - múltiples módulos
 - estrategia de creación general 30
 - pasar parámetros 136
 - preparar para depuración 198
 - procedimiento de entrada de programa 84
 - reducir tamaño 92
 - saltar 228
 - seguir los pasos 227
 - terminación anormal 158
 - terminación normal 157
 - ver fuente al depurar 206
 - volver de 156
 - volver sin terminar 158
- programa, activación 112
- programa, ciclo
 - control de compromiso 352
 - descripción general 4
 - lógica de búsqueda de desbordamiento 360
 - primer ciclo 5
 - último ciclo 6
- programa de generación automática de informes
 - convertir a ILE RPG 440
- programa de lenguaje de control (CL)
 - como módulo en programa ILE 30
 - en aplicación compatible con OPM 25
 - mandatos utilizados con frecuencia 13
 - mandatos utilizados con ILE RPG 13
- programa de servicio
 - actualización 101
 - añadir a una sesión de depuración 204
 - crear 93
 - ejemplo 96
 - ejemplo de listado del enlazador 102
 - en aplicación avanzada 32
 - enlazar con CRTBNDRPG 64
 - lenguaje enlazador 99
 - mandatos CL relacionados 95
 - métodos para crear 94
 - modificar 95
 - razones para utilizar 93
 - reclamar recursos 114
- programa de utilidad para entrada del fuente (SEU) 55
 - entrar fuente 56
 - examinar un listado del compilador 72
- programa fuente
 - conversión a ILE RPG 432
 - convertir algunos miembros 439
 - convertir miembros fuente del generador automático de informes 440
 - convertir todos los miembros 439

- programa fuente (*continuación*)
 - entrada en el sistema 55
 - entrar sentencias SQL 59
 - longitud de registro, al convertir 430
 - nombres de archivo y miembro durante la conversión 431
 - tipos de miembros fuente durante la conversión 430
- programa o procedimiento de prototipos llamada de prototipos 36
- programa/procedimiento, fin después de llamada al sistema 111
- terminación anormal 158
- terminación normal 157
- utilizando API enlazables 159
- visión general, volver de 156
- volver sin terminar 158
- programas, gestión 21
- programas de servicio, creación
 - acerca de 93
 - métodos 94
- prototipo
 - descripción 36
 - utilizar 137
- prototipos de Java, Métodos de 168
- PRTCTL (control de impresora)
 - ejemplo 364
 - información general 363
- punteros
 - en memoria teraespacio 153
- punto de interrupción condicional
 - definición 209
 - establecer 214
 - establecer y eliminar para hebra 220
 - establecer y eliminar para trabajo 213
 - utilizar números de sentencia 217
- punto de interrupción incondicional
 - definición 209
 - establecer 211
 - establecer y eliminar para hebra 212
 - establecer y eliminar para trabajo 210
 - utilizando números de sentencia 217
- punto de reanudación 274
- puntos de interrupción condicional
 - establecer y eliminar para hebra 220
 - establecer y eliminar para trabajo 213
- eliminar todos 220
- establecer utilizando números de sentencia 217
- establecer y eliminar 209
- incondicional
 - establecer y eliminar para hebra 212
 - establecer y eliminar para trabajo 210
- probar 210
- puntos de retorno, especificar en ENDSR 274

Q

- QUAL, mandato de depuración
 - definición 197
 - ILE RPG 241

R

- realizar una conversión de prueba 439
- realizar una conversión rápida 438
- Reasignar almacenamiento (CEECSZST), API enlazable 23
- Reclamar grupo de activación (RCLACTGRP), mandato
 - grupos de activación con nombre 112
 - suprimir grupos de activación 114
- Reclamar recursos (RCLRSC), mandato
 - para liberar almacenamiento 114
 - programa compatible con OPM 26
 - programa ILE 28
- RECNO, palabra clave
 - con proceso por número relativo de registro 343
- recuperar desbordamiento
 - descripción general 360
 - lógica 360
- Recuperar información del descriptor operativo (CEEDOD) 96
 - descriptores operativos 143
 - ejemplo 143
- recurrencia
 - llamada a manejadores de condiciones 275
 - llamadas recurrentes 50, 134
- redirección, archivo
 - definición 301
 - descripción general 301
- reducir el tamaño del objeto 92, 198
- Registrar procedimiento de salida de usuario de finalización de entrada de pila de llamadas (CEERTX) 282
- registro
 - bloqueo 316
 - claves válidas 325
 - liberación 317
 - límites 331
- registro de entrada
 - desbloqueo 327
- registro de salida
 - bloqueo 327
- registros de límites 323
- registros de longitud variable 366
- reintento en un tiempo de espera excedido de bloqueo de registro 317
- REPLACE, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 466
 - mandato CRTRPGMOD 78
- requisitos de la ayuda para la conversión 432
- responder a mensajes de consulta durante la ejecución 110
- restricciones de conversión de códigos 448
- resumen de mensajes del listado del compilador 488

- resumen final del listado del compilador 488
- retorno desde un programa llamado 156
- RPG IV
 - características de RPG III no soportadas 449
 - conversión a 25, 27
 - convertir a 429
 - diferencias de comportamiento entre RPG III 423
 - visión general 3
- Rutinas de Gráficos de Presentación (PGR) 161
- rutinas especiales, llamar a 162

S

- salida
 - especificaciones
 - archivo WORKSTN descrito por programa 380
- sangrar operaciones estructuradas en el listado del compilador 69
- sección de fuente del listado del compilador 478
- sección de mensajes de diagnóstico adicionales del listado del compilador 483
- sección del prólogo del listado del compilador 476
- SECLVL, parámetro 437
- SET, mandato de depuración
 - definición 197
- SETLL
 - excepción MCH3601 425
- sin datos de depuración 198
- soporte de gráficos 161
- Soporte de Idiomas Nacionales (NLS) de depurador del fuente 245
- soporte para valores nulos
 - visualizar campos con capacidad de nulos 239
- spooling 319
- spooling de salida 319
- SQL DB2 para AS/400
 - entrar sentencias SQL 59
- SRCFILE, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 459
 - mandato CRTRPGMOD 78
- SRCMBR, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 460
 - mandato CRTRPGMOD 78
- SRTSEQ, parámetro
 - consideraciones sobre depuración 216
 - CRTBNDRPG, mandato 62
 - efecto en las comparaciones de clave 320
 - Mandato CRTBNDRPG 465
 - mandato CRTRPGMOD 78
- STEP, mandato de depuración
 - definición 197
 - en 228
 - saltar 228
- STRSEU (editar fuente), mandato 56

- subarchivos
 - códigos de operación de archivo
 - permitidos con 377
 - descripción general 375, 377
 - descripciones 375
 - ejemplos 378
 - formato de registro 375
 - formato de registro de control 375
 - utilizaciones de 378
- subcampos
 - para estructura de datos de estado del programa 290
 - para estructura de datos de información de archivo 291, 293
 - para PRTCTL 363
- subprocedimientos
 - ámbito de archivos 84
 - consideraciones acerca de la codificación 50
 - datos locales en listado de vuelco 296
 - depurar 231
 - ejemplo 10
 - entrar en 228
 - flujo de lógica 6
 - información en el listado del compilador 487
 - saltar 228
 - visión general 35
 - volver de 159
- SUBR23R3 (recuperación de mensajes) 162
- SUBR40R3 (manipulación de variables Caracteres de doble byte) 162
- SUBR41R3 (manipulación de variables Caracteres de doble byte) 162
- subrutina de condición de excepción/error del programa
 - descripción 269
 - ejemplo 269
- subrutina de excepción/error de archivos (INFSR)
 - descripción 265
 - ejemplo 266
 - especificaciones para 265
- subrutinas
 - error 264
 - error de archivo (INFSR) 265
 - error de programa (*PSSR) 269
 - evitar un bucle 272
 - llamar a rutinas SUBR 162
- subrutinas de error
 - evitar un bucle 272
 - para errores de archivo 265
 - programa 269
 - utilizar 264
- subserie de literal de gráficos o de caracteres
 - %SUBSTR incorporada de depuración ILE 240
- sugerencias de programación
 - creación de un módulo NOMAIN 94
 - establecer puntos de interrupción de subprocedimiento 229
- suprimir un grupo de activación 114
- sustituir módulos en un programa 90

T

- tabla
 - visualizar al depurar 235
- tabla de parámetros
 - CRTBNDRPG, mandato 62
 - mandato CRTRPGMOD 78
 - mandato CVTRPGSRC 433
- tablas de resumen
 - códigos de operación de archivo permitidos con
 - DISK 344
 - PRINTER 356
 - secuencial 367
 - SPECIAL 369
 - WORKSTN 382
 - proceso de archivo secuencial 367
 - proceso de archivo SPECIAL 369
- TBREAK, mandato de depuración
 - definición 197
 - utilizar 212, 220
- tecla de mandato Ayuda 374
- tecla de mandato Girar hacia abajo 374
- tecla de mandato Girar hacia arriba 374
- tecla de mandato Imprimir 374
- tecla de mandato Inicio 374
- teclas de atención de mandatos (CA) 372
- teclas de función
 - con archivo WORKSTN 374
 - indicadores 374
- teclas de función de mandatos (CF) 372
- teclas de mandatos especiales 374
- terminación anormal de programa o procedimiento 158
- terminación normal de programa o procedimiento 157
- TEXT, parámetro
 - CRTBNDRPG, mandato 62
 - Mandato CRTBNDRPG 460
 - mandato CRTRPGMOD 78
- TGTRLS, parámetro
 - CRTBNDRPG, mandato 62
 - mandato CRTBNDRPG 468
 - mandato CRTRPGMOD 78
- THREAD, mandato de depuración
 - definición 197
 - utilizar 213
- tipos de manejadores de excepciones 251
- tipos de miembros fuente, conversión de 430
- TOFILE, parámetro 436, 439
- TOMBR, parámetro 436, 439
- TOTC 274
- TOTL 274
- Trabajar con entrada de lista de respuestas (WRKRPLYE), mandato
 - cambiar una lista de respuestas del sistema 111
- traspasar una excepción
 - utilizar un manejador de condiciones 276
- TRUNCNBR, parámetro
 - CRTBNDRPG, mandato 62
 - mandato CRTBNDRPG 468
 - mandato CRTRPGMOD 78

U

- USRPRF en CRTBNDRPG, parámetro CRTBNDRPG, mandato 62
- Mandato CRTBNDRPG 466

V

- valor de retorno
 - devolver la utilización de expresión 138
- valores hexadecimales, visualizar al depurar 238
- variable local
 - en vuelco con formato 296
- ver fuente al depurar 206
- vía de acceso
 - ejemplo de 329
 - para archivo DISK descrito externamente 323
 - para archivo indexado 328
- vía de acceso en secuencia de clave 323
- vía de acceso en secuencia de llegada 323
- vía de datos abierta
 - compartimiento 317
- vista de depuración
 - cambiar al depurar 208
 - definición 198
 - fuentes COPY 200
 - fuentes raíz 199
 - listado 201
 - sentencia 201
 - valor por omisión 201
- vista de la sentencia
 - crear 201
 - utilizar para depurar 217
- vista del fuente raíz, crear 199
- vista del listado, crear 201
- VisualAge RPG 15
- visualizar atributos de un campo 243
- visualizar datos y expresiones al depurar 232
- Visualizar fuente de módulo (DSPMODSRC), mandato 204, 205, 206, 207
- Visualizar referencias de programa (DSPPGMREF), mandato 154
- volver a enlazar 90
- volver de un procedimiento principal 157
- volver de un procedimiento principal llamado 156
- volver de un subprocedimiento 159
- volver sin terminar 158
- volver utilizando API ILE enlazables 159
- vuelco, con formato 289
- vuelco con formato 289

W

- WATCH, mandato de depuración
 - definición 197
 - ejemplo 225
 - establecer condiciones 221

X

XML 167



Número de Programa: 5722-WDS

Printed in Denmark by IBM Danmark A/S

SC10-3607-00

