http://www.teknodatips.com.ar

Teknoda - Notas técnicas – Tips de AS400 - iSeries- System i -

Tip Nro. 44

(Lo nuevo, lo escondido, o simplemente lo de siempre pero bien explicado)

Nuevas sentencias de control en programación CL

Tema: Programación CL
Utilidad: Conocer y aplicar las nuevas estructuras de control, tanto iterativas como de selección, que se pueden utilizar en la programación CL
Nivel: Intermedio.
Versión: V5R3 o superior

Lista de Tips publicados hasta la fecha:

Resumen ejecutivo e Introducción

En cada nueva versión del OS/400 (i5/OS) IBM incorpora nuevos comandos CL y modifica otros para otorgarles funcionalidad que no tenían, por ejemplo, con el agregado de parámetros nuevos.

Sin embargo, desde la V5R3 del i5/OS se incorporaron cambios y mejoras sin precedentes con respecto a las **capacidades de programación** del Lenguaje de Control (CL), que la comunidad AS/400 (iSeries , System i) había estado esperando por mucho tiempo.

Entre las nuevas capacidades en la programación, es importante destacar la incorporación de **nuevas sentencias de control**, tanto iterativas como de selección.

En el presente tip, detallaremos con algunos ejemplos estas nuevas sentencias iterativas, que incluyen: **DOWHILE, DOUNTIL y DOFOR** (nuevas sentencias que reemplazan al conocido GOTO) y, por otro lado, la sentencia **SELECT** como nueva sentencia de control de selección, complementando la utilización de la sentencia IF.

Nuevas sentencias iterativas estructuradas

Existen tres nuevas estructuras para ciclos desde V5R3: **DOWHILE**, **DOUNTIL**, y **DOFOR**. Todas ellas controlan un cuerpo de comandos terminados con la sentencia **ENDDO**.

Sentencia DOWHILE

La sentencia **DOWHILE** acepta un parámetro, palabra clave **COND**, que permite especificar el mismo tipo de condición que se establece en la ya conocida sentencia IF.

Esta sentencia iterativa ejecuta los comandos incluidos en el cuerpo del ciclo DOWHILE **mientras la condición sea verdadera** y cuando deja de serlo, el control pasa al siguiente comando de la sentencia ENDDO asociada.

La sentencia **DOWHILE** es una sentencia iterativa que evalúa la condición lógica **al principio** del ciclo, por lo tanto no hay garantía de que el código dentro del ciclo se ejecute alguna vez.

El siguiente es un ejemplo de un programa CL sencillo que utiliza la sentencia DOWHILE:

```
PGM
DCL         VAR(&ENDJOB) TYPE(*CHAR) LEN(1) VALUE('0')
RTVJOBA     ENDSTS(&ENDJOB) /* Recupera el estado de +
              cancelación del job*/
DOWHILE     COND(&ENDJOB = '0') /* Mientras No se establezca la +
                                   cancelación del job  (&ENDJOB = '1') */
            CALL        PGM(TEKPRU/FUNDAM)
            CALL        PGM(TEKVEN/PROGRPG1)
RTVJOBA     ENDSTS(&ENDJOB) /* Vuelve a recuperar el estado de +
                               cancelación del job */
ENDDO
ENDPGM
```

En el código anterior, se establece un ciclo que invoca al programa FUNDAM de la biblioteca TEKPRU y al programa PROGRPG1 en TEKVEN, **mientras** no se produzca un EOJ. Si la devolución de la variable &ENDJOB en la primera sentencia RTVJOBA, devolviera un '1', significa que se ha establecido una cancelación del job donde está ejecutándose el programa. Por lo tanto las sentencias dentro del loop no se ejecutarían ni una sola vez, alcanzando en este caso la sentencia ENDPGM.

Sentencia DOUNTIL

Como en la sentencia DOWHILE, la sentencia **DOUNTIL** también acepta un parámetro, palabra clave **COND**, que permite especificar el mismo tipo de condición que se establece en la ya conocida sentencia IF.

Al usar la sentencia **DOUNTIL**, la condición lógica indicada en el parámetro COND, se evalúa **después** de ejecutar el grupo de comandos CL especificados dentro del loop, y:

- Si la condición es **verdadera**, no se ejecuta el grupo de comandos indicados en el cuerpo del ciclo, se deja el loop y el procesamiento continúa con la sentencia siguiente a la sentencia ENDDO asociada.
- Si la condición es **falsa**, continúa procesándose el grupo de comandos dentro del cuerpo del loop, comenzando con el primero de ese grupo.

Por lo tanto, al usar la sentencia **DOUNTIL**, el grupo de comandos CL dentro del cuerpo del loop se ejecutará **al menos una vez.**

Lo siguiente especifica el mismo ejemplo de código anterior, pero en este caso utilizando la sentencia **DOUNTIL**, en vez de DOWHILE:

```
PGM
DCL         VAR(&ENDJOB) TYPE(*CHAR) LEN(1) VALUE('0')
DOUNTIL     COND(&ENDJOB = '1' ) /* Hasta que se establezca la +
                                    cancelación del job */
            CALL        PGM(TEKPRU/FUNDAM)
            CALL        PGM(TEKVEN/PROGRPG1)
 RTVJOBA     ENDSTS(&ENDJOB) /* Vuelve a recuperar el +
                  estado de cancelación del job */
ENDDO
ENDPGM
```

Se puede observar que al usar la sentencia **DOUNTIL**, los dos programas invocados en el cuerpo del loop, se ejecutarán **al menos una vez**. Después de cada iteración, se evalúa el valor de la variable &ENDJOB. Si &ENDJOB tiene un valor de '1', entonces el programa continuará **después** de la sentencia ENDDO, alcanzando en este caso la sentencia ENDPGM.

Sentencia DOFOR

La tercera estructura iterativa que se detalla es la sentencia **DOFOR**. Esta sentencia es similar en funcionalidad a los códigos de operación DO y FOR de la programación RPG.

La sentencia **DOFOR** requiere para su ejecución, una variable de control, **que debe ser de tipo *INT** o ***UINT**, (no de tipo *DEC), para ser definida en el parámetro **VAR** . (Los tipos de variables ***INT y *UINT** son nuevos tipos de variables agregadas a la programación CL en la V5R3.)

DOFOR también requiere los parámetros **FROM** y **TO**, que denotan los valores de comienzo y de límite para la variable de control. Estos valores pueden ser constantes, expresiones o variables.

Existe un parámetro opcional, llamado **BY**, que permite identificar el **incremento** a la variable de control antes de cada iteración. Para ciclos descendentes, se deberá ingresar un valor negativo para este parámetro, y un valor 0 (cero) o positivo para ciclos ascendentes. No se permiten variables para este parámetro y el valor por default es 1.

El siguiente es un programa ejemplo con el uso de la sentencia **DOFOR**:

```
PGM
DCL         VAR(&VAR) TYPE(*INT)
DCL         VAR(&POS) TYPE(*DEC) LEN(10) VALUE(1)
DOFOR       VAR(&VAR) FROM(1) TO(10)
       CHGVAR      VAR(%SST(*LDA &POS 2)) VALUE(&VAR)
       CHGVAR      VAR(&POS) VALUE(&POS + 50)
       DSPDTAARA   DTAARA(*LDA)
ENDDO
ENDPGM
```

En el código anterior, se puede notar que en cada paso de la iteración (desde el 1 al 10) se le asigna el valor de &VAR al Area de Datos local (*LDA) en la posición indicada por la variable &POS que se incrementa en 50 cada vez.

Comandos LEAVE e ITERATE asociadas a las sentencias iterativas

Se pueden utilizar los nuevos comandos LEAVE e ITERATE, asociados con las sentencias de control iterativas detalladas previamente, para alterar el comportamiento de los ciclos DOWHILE, DOUNTIL y DOFOR.

El **comando LEAVE, dentro del loop,** indica finalizar el proceso, "salir" del loop y pasar el control al comando siguiente al ENDDO del loop asociado (por default, si no se utiliza una etiqueta en el mandato).

En cambio, el **comando ITERATE,** interrumpe el procesamiento de los comandos, pasa el control al final del loop y evalúa la condición para determinar si las sentencias dentro del cuerpo del loop se ejecutarán o no nuevamente.

Además, ambas sentencias soportan el uso del parámetro **Etiqueta de Mandato** (palabra clave **CMDLBL**) y permite "saltar" a otros loops anidados en el mismo programa. Si no se provee una etiqueta, las sentencias LEAVE o ITERATE se refieren al ciclo más interno. En caso contrario, se referirán a las etiquetas del grupo DOWHILE, DOUNTIL o DOFOR activo.

La sentencias LEAVE e ITERATE **sólo** están permitidas dentro de un grupo DOWHILE, DOUNTIL o DOFOR.

La estructura de selección SELECT

Además de las sentencias de control iterativas mencionadas en los párrafos anteriores, la V5R3 incorporó una **nueva estructura de selección**, que está implementada con los comandos: **SELECT, WHEN, OTHERWISE y ENDSELECT.**

La **estructura SELECT** es una estructura "IF" más especializada, que define un conjunto mutuamente excluyente de condiciones lógicas a evaluar y, con cada condición que se evalúa, existe un comando asociado a ejecutar si la condición evaluada es verdadera.

Qué especifican los comandos que intervienen en la estructura:

- **SELECT** comienza la estructura y **ENDSELECT** la finaliza. Estas sentencias no aceptan parámetros.
- Un bloque SELECT debe contener al menos una sentencia **WHEN**, y en cambio, puede contener o no un comando **OTHERWISE**.
- El grupo WHEN es idéntico en estructura a la sentencia IF: el primer parámetro, COND, es una expresión lógica que se evalúa a verdadero o falso. El segundo parámetro, THEN, es opcional e indica una única acción a realizarse si la condición evaluada es verdadera. Se puede utilizar un grupo DO/ENDDO para ejecutar más de un comando, como en la sentencia IF.
- El sistema ejecuta el **primer WHEN** cuya condición se evalúa a verdadero, ignorando los WHEN subsiguientes y el comando OTHERWISE que pudieran aparecer más abajo en la estructura.
- **OTHERWISE** es idéntico a ELSE en su estructura y define el/los comandos a ser ejecutados si **ninguna** de las condiciones sobre las sentencias WHEN en el bloque SELECT es verdadera. Tambien en este caso, se puede utilizar un grupo DO/ENDDO para ejecutar más de un comando.

El siguiente es un ejemplo de un programa que utiliza el **grupo SELECT**, y el uso de la sentencia **DOWHILE**:

```
PGM
DCL        VAR(&LGL) TYPE(*LGL) VALUE('1')
DCLF       FILE(TEKTCL/PANTAPRU)
DOWHILE    COND(&LGL)    /* true */
SNDRCVF
   SELECT
        WHEN       COND(&OPCION = 1) THEN(DSPLIB LIB(TEKTCL))
        WHEN       COND(&OPCION = 2) THEN(WRKSPLF)
        WHEN       COND(&OPCION = 3) THEN(DSPSYSVAL SYSVAL(QTIME))
        WHEN       COND(&OPCION = 4) THEN(DSPSYSVAL SYSVAL(QDAY))
        WHEN       COND(&OPCION = 5) THEN(LEAVE)
        WHEN       COND(&IN03 = '1') THEN(LEAVE)
   ENDSELECT
 ENDDO
 ENDPGM
```

Para concluir, se puede decir que a medida que se vayan conociendo las sentencias DOWHILE, DOUNTIL y DOFOR, va a implicar que de a poco se vaya reemplazando el uso de la sentencia GOTO, utilizada tradicionalmente en la programación CL como sentencia de control de flujo iterativa. Podría ocurrir lo mismo con el grupo SELECT.

Para tener en cuenta:

- La V5R3 es release con las mejoras más importantes realizadas al compilador CL desde las realizadas en el compilador ILE CL en V3R1.

Con respecto a los anidamientos en las sentencias mencionadas, existen algunas limitaciones:

- Se pueden anidar hasta 10 niveles de sentencias IF. Esto no cambió con respecto a los releases anteriores.
- Se pueden anidar hasta 25 niveles de comandos DO, DOWHILE, DOUNTIL y DOFOR. Esto no significa tener 25 niveles de cada uno de los tipos de DO, sino 25 niveles de todas las operaciones "DO….".
- Se pueden tener hasta 25 niveles de anidamiento del comando SELECT
- Los comandos DOWHILE, DOUNTIL, DOFOR y el grupo SELECT sólo son válidos en procedimientos CL.

Existen otras mejoras en la programación CL incorporadas **desde la V5R3** que incluye:

- nuevos tipos de variables, por ejemplo *INT y *UINT, y **en V5R4** un tipo de variable *PTR (**variable tipo Pointer** – Puntero)
- el manejo de **subrutinas** en programas CL en la V5R4
- Desde la **V5R3** en los programas CL, se pueden declarar **hasta 5 archivos**, eliminando la restricción de manejar "sólo uno" como en las versiones anteriores de OS/400.

Dudas o consultas a : nsalmun@teknoda.com