Programación Lógica

Daniel del Valle [2019-09-01]

Bucles

Un bucle o lazo (loop) es un segmento de un algoritmo o programa, cuyas instrucciones se repiten un número determinado de veces mientras se cumple una determinada condición. Se debe establecer un mecanismo para determinar las tareas repetitivas. Este mecanismo es una condición que puede ser verdadera o falsa y que se comprueba una vez a cada paso o iteración del bucle. Un bucle consta de tres partes:

- Decisión
- Cuerpo del bucle
- Salida del bucle

Bucles anidados

En un algoritmo pueden existir varios bucles. Los bucles pueden ser anidados o independientes. Los bucles son anidados cuando están dispuestos de tal modo que unos son interiores a otros.

CONTADORES [constante]

Los procesos repetitivos son la base del uso de las computadoras. En estos procesos se necesitan normalmente contar los sucesos o acciones internas del bucle, como pueden ser los elementos de un fichero, el numero de iteraciones a realizar por el bucle, etc. Una forma de controlar un bucle es mediante un contador.

Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante en cada iteración. El contador puede ser positivo(incrementos, uno en uno) o negativo (decremento, uno en uno). En el ejemplo el contador cuenta desde 1 a 10 y deja de contar cuando la variable cont. Toma el valor 51 y se termina el bucle.

ACUMULADOR [variable]

Un acumulador o TOTALIZADOR es una variable cuya misión es almacenar cantidades variables resultantes de sumas sucesivas. Realiza la misma función que un contador con la diferencia de que el incremento o decremento de cada suma es variable en lugar de constante como en el caso del contador. Se representa por la instrucción: `variable_1 = variable_1 + variable_2.`

INTERRUPTORES [variable binaria]

Un interruptor o CONMUTADOR (SWITCH), a veces se les denomina INDICADOR, BANDERA (FLAG). Es una variable que puede tomar diversos valores a lo largo de la ejecución del programa y que permite comunicar información de una parte a otra del mismo. Los interruptores pueden tomar dos valores diferentes 1 y 0. (De ahí su nombre de interruptor, "encendido" o "apagado", "abierto" o "cerrado").

SI *bandera* ENTONCES
    Acción_1
SINO
    Acción_2
FIN_SI

Si bandera es igual a 1, se ejecuta la *acción_1*, y si bandera es igual a 0, se ejecuta la *acción_2*.

Teorema de la programación estructurada

En mayo de 1966 Böhm y Jacopini demostraron que un programa propio puede ser escrito utilizando solamente tres tipos de estructura de control:

1. Secuenciales
2. Selectivas
3. Repetitivas*

ESTRUCTURA SECUENCIAL

La estructura secuencial es aquélla en la que una acción sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. La estructura secuencial tiene una entrada y una salida.

```
Acción 1 > Acción 2 > Acción 3
```

ESTRUCTURA SELECTIVAS

La especificación formal de algoritmo tiene realmente utilidad cuando el algoritmo requiere una descripción más complicada que una lista sencilla de instrucciones. Éste es el caso cuando existe un número de posibles alternativas resultantes de la evaluación de una determinada condición. Las estructura selectivas se utilizan para tomar decisiones lógicas; de ahí que se suelan denominar también estructuras de decisión o alternativas. En la estructura selectiva se evalúa una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas. La representación de una estructura selectiva se hace con palabra en pseudocódigo, con una figura geométrica en forma de rombo. Las estructuras selectivas o alternativas pueden ser: 1) Simples, 2) Dobles, 3) Múltiples.

Alternativa simple

La estructura alternativa simples si_entonces, ejecuta una determinada acción cuando se cumple una determinada condición. La selección si_entonces evalúa la condición:

- Si la condición es verdadera, entonces ejecuta la acción (simple o contar de varias acciones)
- Si la condición es falsa, entonces no hace nada.

SI *condición* ENTONCES
    Acción por verdadero
FIN_SI

Observe que las palabras del pseudocódigo SI y FIN_SI se alinean verticalmente indentando (sangrado) la acción o bloque de acciones.

Alternativa doble

La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. Si la condición es verdadera, se ejecuta la acción_1 y si es falsa, se ejecuta la acción_2.

SI *condición* ENTONCES
    Acción_1
SINO
    Acción_2
FIN_SI

Observe que en el pseudocódigo las acciones que dependen de entonces y sino están sangradas en relación con las palabras SI y FIN_SI; este procedimiento aumenta la legibilidad de la estructura y es el medio idóneo para representar algoritmos.

Alternativa múltiple

Con frecuencia, es necesario que existan más de dos elecciones posibles. Por ejemplo: en la resolución de la ecuación de segundo grado existen tres posibles alternativas o camino a seguir, según que el discriminante sea negativo, nulo o positivo. Este problema, se podría resolver por estructuras alternativas simples o dobles, anidadas o en cascada; sin embargo, este método si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y naturalmente de legibilidad. La estructura de decisión múltiple evaluará una expresión que podrá tomar n valores distintos: 1, 2, 3, . . . , n. Según que elija uno de estos valores en la condición, se realizará una de las n acciones, o lo que es igual, el flujo del algoritmo seguirá un determinado camino entre los n posibles.

SEGÚN_SEA
    E1: Acciones 1
    E2: Acciones 21
    Acciones 22
    En: Acciones n
    OTRO: E<>
FIN_SEGUN

La estructura de decisión múltiple en pseudocódigo se puede representar de diversas formas, pudiendo ser las acciones, simples o compuestas.

ESTRUCTURAS REPETITIVAS

Las computadoras están especialmente diseñadas para todas aquellas aplicaciones en las cuales una operación o conjunto de ellas deben repetirse muchas veces. Un tipo muy importante de estructura es el algoritmo necesario para repetir una o varias acciones un número determinado de veces. Un programa que lee una lista de números puede repetir la misma secuencia de mensajes el usuario e instrucciones de lectura hasta que todos los números de un fichero se lean. Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones. Por ejemplo: Si se desea sumar una lista de números escritos desde teclado. El medio conocido hasta ahora es leer los números y añadir sus valores a una variable SUMA que contenga las sucesivas sumas parciales. La variable SUMA se hace igual a cero y a continuación se incrementa en el valor del número cada vez que uno de ellos se lea. El algoritmo que resuelve este problema es:

```
Inicio
    suma = 0
    leer número
    suma =suma
    leer número
    suma =suma
    ......
    .......
fin
```

y así sucesivamente para cada número de la lista. En otras palabras, el algoritmo repite muchas veces las acciones. Las dos principales preguntas a realizarse en el diseño de un bucle son: ¿Qué contiene el bucle? y ¿Cuántas veces se debe repetir? Cuando se utiliza un bucle para sumar una lista de números, se necesita saber cuántos números se han de sumar. Para ello necesitaremos conocer algún medio para detener el bucle. Existen dos procedimientos para contar el número de iteraciones, usar una variable que se inicializa a la cantidad de números que se desea y a continuación se decrementa en uno cada vez que el bucle se repite, o bien inicializar la variable en cero o en uno, e ir incrementando en uno a cada iteración hasta llegar al número deseado. Para detener la ejecución de los bucles se utiliza una condición de parada. Los tres casos generales de estructuras repetitivas dependen de la situación y modo de la condición. La condición se evalúa tan pronto se encuentra en el algoritmo y su resultado producirá los tres tipos de estructuras citadas.

1. La condición de salida del bucle se realiza al principio del bucle (estructura mientras)
2. La condición de salida se origina al final del bucle; el bucle se ejecuta hasta que se verifique una cierta condición.
3. La condición de salida se realiza con un contador que cuenta el número de iteraciones.

Estructura MIENTRAS

La estructura repetitiva mientras es aquella en la que el bucle se repite mientras se cumple una determinada condición. La representación gráfica es:

```
Mientras condición
    Acciones
    ............
    ..............
Fin_mientras
```

Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión booleana). Si se evalúa falsa, ninguna acción se toma y el programa prosigue en la siguiente instrucción del bucle. Si la expresión booleana es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión booleana. Este proceso se repite una y otra vez mientras la expresión (condición) sea verdadera.

Estructura REPETIR

Existen muchas situaciones en las que se desea que un bucle se ejecute al menos una vez antes de comprobar la condición de repetición. En la estructura mientras si el valor de la expresión booleana es inicialmente falsa, el cuerpo del bucle no se ejecutará; Por ello se necesitan otros tipos de estructuras repetitivas. La estructura repetir se ejecuta hasta que se cumpla una condición determinada que se comprueba al final del bucle.

```
Repetir
    Acciones
Hasta_que condición
```

El bucle repetir hasta_que se repite mientras el valor de la expresión booleana de la condición sea falsa, justo la opuesta de la sentencia mientras. Con una estructura repetir, el cuerpo del bucle se ejecuta siempre al menos una vez. Cuando una instrucción repetir se ejecuta, la primera cosa que sucede es la ejecución del bucle y a continuación se evalúa la expresión booleana resultante de la condición. Si se evalúa como falsa, el cuerpo del bucle se repite y la expresión booleana se evalúa una vez. Después de cada iteración del cuerpo del bucle, la expresión booleana se evalúa; si es verdadera, el bucle termina y el programa sigue en la siguiente instrucción a hasta_que.

Estructura DESDE / PARA

En muchas ocasiones se conoce de antemano el número de veces que desean ejecutar las acciones de un bucle. En estos casos en el número de iteraciones es fija, se debe usar la estructura desde o para. La estructura desde ejecuta las acciones del cuerpo del bucle un número especificado de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.

```
Desde Vc=Vi hasta Vf Hacer
    Acciones
Fin_Desde
```

Donde:
Vc= Variable de Control.
Vi= Valor Inicial.
Vf= Valor Final.

La estructura desde comienza con un valor inicial de la variable índice y las acciones especificadas se ejecutan a menos que el valor inicial sea mayor que el valor final. La variable índice se incrementa en uno y si este nuevo valor no excede al final, se ejecutan de nuevo las acciones. Por consiguiente, las acciones específicas en el bucle se ejecutan para cada valor de la variable índice desde el valor inicial hasta el valor final con el incremento de uno en uno. El incremento de la variable índice siempre es 1 si no se indica expresamente lo contrario. Es posible que el incremento sea distinto de uno, positivo o negativo. La variable índice o de control normalmente será de tipo entero y es normal emplear como nombre las letras: I, J, K. El formato de la estructura desde varia si se desea un incremento distinto a 1, bien positivo, bien negativo (decremento).

```
Desde I = vi hasta vf Paso inc {inc incremento}..
        Dec {dec decremento}..
    Acciones..
    ..............
    ..............
fin_desde..
```

Si el valor inicial de la variable índice es menor que el valor final, los incrementos deben ser positivos, ya que en caso contrario la secuencia de acciones no se ejecutaría. De igual modo si el valor inicial es mayor que el valor final, el incremento debe de ser en este caso negativo, es decir, decremento. Al incremento se le suele denominar también paso.

Elementos básicos de un programa

En programación se debe separar la diferencia entre el diseño del algoritmo y su implementación en un lenguaje específico. Por ello se debe distinguir claramente entre los conceptos de programación y el medio en que ellos se implementan en un lenguaje específico. Sin embargo, una vez que se comprendan los conceptos de programación y cómo utilizarlos, la enseñanza de un nuevo lenguaje es relativamente fácil. Los lenguajes de programación tienen elementos básicos que se utilizan como bloques constructivos, así como reglas para las que esos elementos se combinan. Estas reglas se denominan sintaxis del lenguaje. Solamente las instrucciones sintácticamente correctas pueden ser interpretadas por la computadora y los programas que contengan errores de sintaxis son rechazados por la máquina. Los elementos básicos constitutivos de un programa o algoritmo son:

= Palabras reservadas
= Identificadores
= Caracteres especiales
= Constantes
= Variables
= Expresiones
= Instrucciones

Además de estos elementos básicos existen otros elementos que forman parte de los programas, cuya comprensión y funcionamiento será vital para el correcto diseño de un algoritmo y naturalmente la codificación del programa.

= Bucles
= Contadores
= Acumuladores
= Interruptores
= Estructuras
   1. Secuenciales
   2. Selectivas
   3. Repetitivas

El amplio conocimiento de todos los elementos de programación y el modo de su integración en los programas constituyen las técnicas de programación que todo buen programador debe conocer.