aprenderaprogramar.com

Herramientas de programación: contadores y su control. Valor inicial y final en bucles for next, do while, etc. (CU00157A)

Sección: Cursos

Categoría: Curso Bases de la programación Nivel I

Fecha revisión: 2024

Autor: Mario R. Rancel

Resumen: Entrega nº 56 del Curso Bases de la programación Nivel I

24

HERRAMIENTAS PARA LA PROGRAMACIÓN

Llamaremos herramienta a una parte del código cuyo fin es apoyar una construcción más amplia y que resulta efectiva en distintos programas. Un martillo es una herramienta que nos será útil para construir una casa de madera, resultando además efectivo para otro tipo de construcciones como una casa de bloques, un garaje, un edificio, un centro comercial, etc. Así herramientas podrían ser: una variable, un bucle, una serie de instrucciones ... que podemos usar en distintos programas para cumplir un fin.

CONTADORES

Ya nos hemos adentrado en el concepto de contador al hablar de la instrucción *Desde ... Siguiente*. Allí lo definíamos como un elemento que toma valores ordenadamente, y decíamos que entre los programadores es habitual usar letras como *i, j, k, m, n.* Lo expuesto, en referencia a la variable de control de un bucle *Desde ... Siguiente*, es válido. Queremos aquí extender el concepto de contador a cualquier estructura, instrucción o parte de un programa que se nos ocurra. Usar una variable como contador nos permitirá saber la cantidad de veces que ha ocurrido algo, y esto a su vez nos servirá para tomar decisiones, mostrar mensajes al usuario, comenzar o finalizar procesos, así como para el control de bucles.

El control de un bucle *Mientras ... Hacer* se realizaría de la siguiente manera:

```
[Pseudocódigo aprenderaprogramar.com]
Mientras i < límite Hacer
        Instrucción 1 Instrucción 2
        i = i + 1
Repetir
```

Por supuesto se pueden introducir variantes en la condición (*i <= límite*, *i > límite*, *i >= límite* ...) o en el contador (*i = i + 2*, *i = i + 3*, *i = i + 15*, *i = i – 1*, *i = i – 2*, *i = i – 15*, etc.). Es decir, el contador puede funcionar sumando o restando, contando hacia adelante o hacia atrás, y de uno en uno, de dos en dos o lo que se desee.

Se ha de tener en cuenta que una mala definición del contador, límite y progresión puede dar lugar a un bucle infinito. Por ejemplo si *límite > i*, progresión es *i = i – 1* y condición es *i < límite*, el bucle no se acabará nunca ya que la progresión da lugar a que la variable *i* siempre siga siendo menor que el límite.

Ejemplo:

```
Mientras i < 10 Hacer
        Mostrar "Mensaje"
        i = i – 1
Repetir
```

Si la variable contador entra con valor *i = 0*, tras la primera pasada adoptará el valor *– 1*, a la segunda *– 2*, y sucesivamente *– 3, – 4,* ... no llegándose a verificar nunca la condición de salida del bucle (*i >= 10*).

Un aspecto a tener en cuenta es el valor inicial del contador en el momento de entrar en el bucle, que será cero o vacío si no ha intervenido en el programa, o el valor que tenga derivado de su intervención anterior si ya ha aparecido. Supongamos un programa del tipo:

```
[Ejemplo aprenderaprogramar.com]
Desde i = 1 hasta 10
        Leer Nombre(i)
        Mostrar "Bienvenido", Nombre(i)
Siguiente
[Aquí irán distintos procesos]
Mostrar "A continuación la despedida"
Mientras i <= 10 Hacer
        Mostrar "Adiós", Nombre(i)
        i = i + 1
Repetir
```

En este caso se da la bienvenida a 10 nombres leídos desde un archivo, se muestra "A continuación la despedida", y sin embargo no se muestra ningún mensaje de Adiós. ¿Por qué? La intención del programador es clara pero "olvidó" que la variable *i* tenía un valor anterior. En este caso, derivado de su uso en el bucle de bienvenida, tras el cual el valor de *i* había quedado establecido en *11* (ver el apartado *Desde ... Siguiente* / Valor del contador). Un "olvido" de este tipo puede dar lugar a un mal funcionamiento de todo un programa, errores, bloqueos, etc.

La solución pasa o bien por estar seguros de antemano de que el valor de entrada de la variable es válido, o bien por fijar la variable a un valor deseado justo antes de entrar en el bucle. En este caso haríamos:

```
Mostrar "A continuación la despedida"
i = 1
Mientras i <= 10 Hacer
        Mostrar "Adiós", Nombre(i)
        i = i + 1
Repetir
```

Para la instrucción *Hacer … Repetir Mientras*, todo lo comentado es válido. La escritura sería del tipo:

```
[Pseudocódigo aprenderaprogramar.com]
Hacer
        Instrucción 1
        Instrucción 2
        i = i + 1
Repetir Mientras i < límite
```

Un uso de los contadores habitual aunque no vinculado al control de flujos, consiste en emplearlos para controlar la cantidad de elementos de una lista que cumplen unas condiciones. Por ejemplo:

```
i = 1: j = 0
Mientras i <= 100 Hacer
        Leer Dato(i)
        Si Dato(i) < 50 Entonces
                j = j + 1
        FinSi
        i = i + 1
Repetir
```

Se extraen 100 datos almacenados en un array contando a través de la variable *i*, y se cuenta de entre esos datos cuántos son menores de 50 a través de la variable *j*. Nótese que la variable *i* se inicia en uno para empezar a leer a partir del dato 1, mientras que la variable *j* se inicia en cero para empezar a contar a partir del cero. En caso de que ningún dato sea inferior a 50 el contador *j* queda con el valor cero.

En este caso hemos usado una variable para controlar el bucle y otra para contar elementos que cumplen una condición. ¿Es posible hacer las dos cosas con una única variable? Posible sí es, aunque podemos calificarlo como “delicado”. El problema radica en que si el contador del bucle depende de una condición y esa condición por un motivo u otro no se cumple nunca... lo ya sabido: bucle infinito. Así que en principio, como a tantas cosas, pongámosle la etiqueta de “no recomendable”. Su uso sólo sería acertado en una situación perfectamente controlada.

Supongamos el caso:

```
[Ejemplo aprenderaprogramar.com]
Inicio
    i = 0 : A = 1 : B = 100
    Mientras i <= 100 Hacer
        Desde j = A hasta B – 1 Hacer
            Leer Dato(j)
            Si Dato(j) < 50 Entonces
                i = i + 1
            FinSi
        Siguiente
        A = B : B = B + 100
    Repetir
Fin
```

Aparte de contener un bucle anidado en otro, observamos que la variable *i* se está usando al mismo tiempo como controladora del bucle a modo de contador y como contador del número de elementos de una serie cuyo valor es inferior a 50. No es incorrecto pero sí delicado. En caso de no existir suficiente número de datos inferior a 50 no se produciría la salida del bucle.
El algoritmo actúa de la siguiente manera: entra en un bucle controlado por el valor *i,* que en ese momento vale cero. Seguidamente entra en otro bucle donde realiza *99* iteraciones (desde *A* hasta *B – 1*, esto es, de *1* hasta *100 – 1*). En cada iteración extrae una variable de un array que suponemos guardado en un fichero y lo analiza. Si su valor es menor de *50* lo contabiliza en el contador *i*, que a su vez, recordemos, controla el bucle externo. Una vez finalizadas las *99* iteraciones *A* adquiere el valor de *B* y *B* se incrementa en *100* unidades. Se vuelve a la entrada del bucle exterior y se analiza si *i* es menor o igual que *100.* Ser igual a 100 tras la primera pasada será imposible porque para *99* iteraciones el valor máximo de *i* es *99*. En la siguiente pasada, se extraen los siguientes *100* datos del array y se repite el proceso.

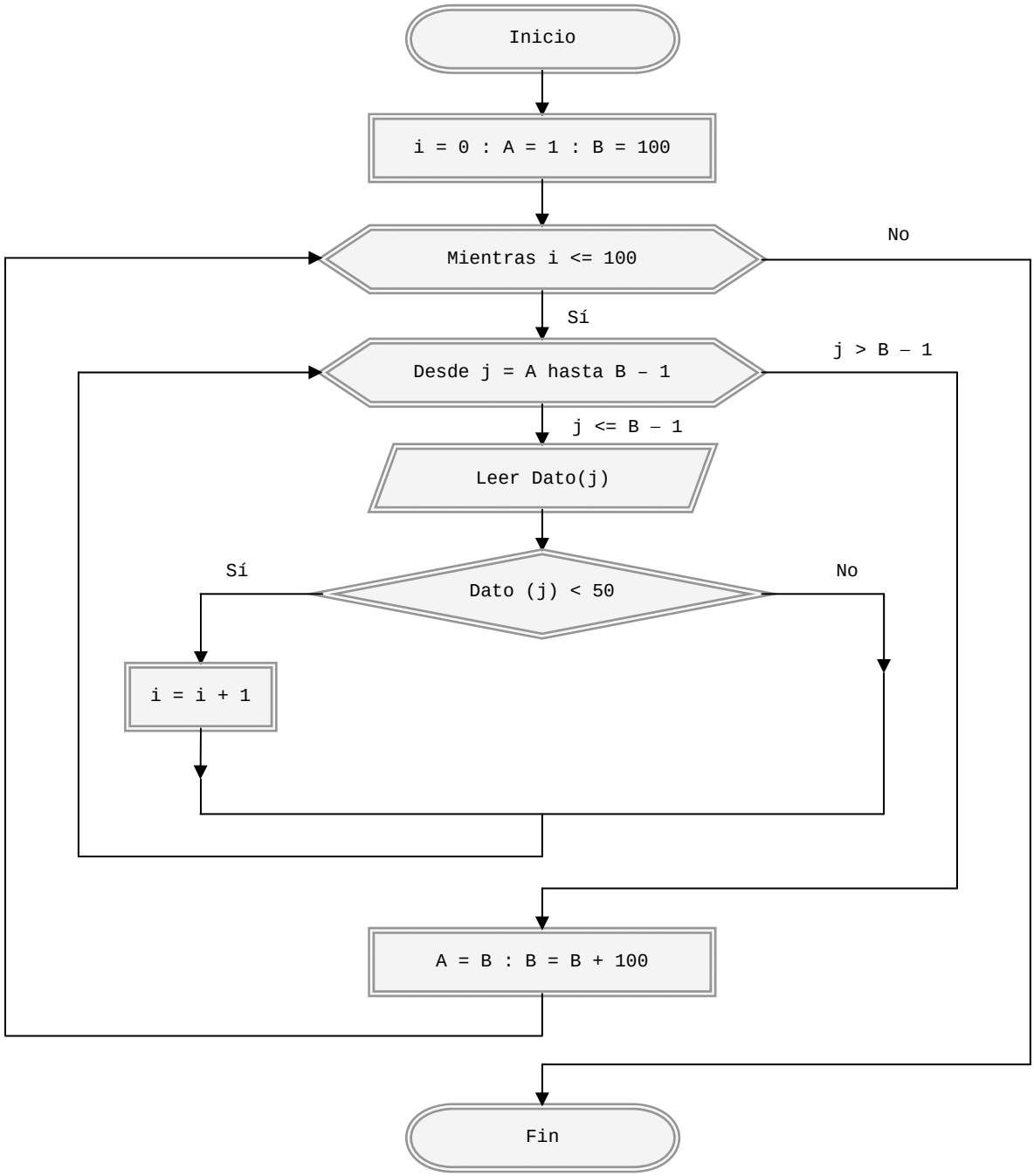
Estamos suponiendo que el número de datos contenido en el array no es limitante (cifra muy elevada), lo cual sería otra cuestión a contemplar. Centrándonos en el valor de la variable *i* de control del bucle, de lo expuesto se deduce que:

- El valor mínimo de *i* con el que se podrá salir del bucle es *i = 101.*
- El número mínimo de pasadas del bucle para salir será: *2*.
- El número máximo de pasadas para salir será: indeterminado.

Y es en este “indeterminado” donde apreciamos algo indeseable. Porque podría darse el caso de que haya un número elevado de datos con valor inferior a *50*, en cuyo caso la salida del bucle sería rápida. Podría darse el caso de un número reducido de datos con valor inferior a *50*, en cuyo caso la salida del bucle sería lenta (se alarga el tiempo necesario por tener que procesar muchos más datos). Y por último, podría darse el caso de que el número de datos con valor inferior a *50* fuera nulo o escasísimo. Si es nulo el bucle no termina nunca y el ordenador “se bloquea”. Si es escasísimo (necesidad de procesar una gran cantidad de datos) habrá un bloqueo aparente que puede ser prácticamente equivalente a un bloqueo total.
La solución a estos casos vendría por:

- Diseño de algoritmos eficientes y que eviten “indeterminaciones”.
- Introducción de mecanismos de seguridad que eviten bloqueos. Los estudiaremos más adelante.

Veamos el diagrama de flujo:



Próxima entrega: CU00158A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59