Estructuras de repetición

Sitio: Agencia de Aprendizaje a lo largo de la Vida
Curso: Tecnicas de Programación 1° F
Libro: Estructuras de repetición

Imprimido por: MARIO DAVID GONZALEZ BENITEZ
Día: sábado, 7 de septiembre de 2024, 23:04

Tabla de contenidos

1. Introducción

Las estructuras de repetición son las llamadas estructuras cíclicas, iterativas o de bucles. Permiten ejecutar un conjunto de instrucciones de manera repetida (o cíclica) mientras que la expresión lógica a evaluar se cumpla, es decir, que su valor sea verdadero.

¡Te invitamos a conocer estos elementos fundamentales de la programación!

2. ¿Qué es una estructura de repetición?

Son estructuras donde una o un conjunto de órdenes o sentencias deben cumplirse más de una vez.

Cuando se conoce con exactitud la cantidad de veces que las órdenes se van a ejecutar, se está ante la presencia de un “ciclo exacto o definido”, por ejemplo: circulando por la calle con nuestro vehículo, el cual tiene neumáticos convencionales de 4 tuercas, hemos pinchado una rueda y debemos cambiarla por la de auxilio. Tenemos que quitar el neumático. Para poder llevarlo a cabo, necesitamos quitar las 4 tuercas que lo sujetan. Es decir, necesitamos repetir el mismo proceso exactamente 4 veces, una por cada tuerca.

Ahora bien…. cuando esta cantidad no se conoce, es necesario buscar algún valor clave que termine la repetición sino nuestro algoritmo quedaría ciclando indefinidamente y entonces “el ciclo es condicional”.

Conozcamos una situación de la vida cotidiana: Preparar ñoquis.

Supongamos que estamos preparando unos ricos ñoquis para el almuerzo, ponemos agua en la olla, una vez que comienza a hervir, debemos introducir la pasta en la olla. Pero estemos atentos a que no se nos pasen, debemos ir mirándolos y apenas suban a la superficie hay que quitarlos. Es decir, aquí podríamos detectar el mismo proceso (ir mirando los ñoquis) mientras no suban a la superficie. Una vez que esto ocurra, debemos dejar de mirar para ya poder servirlos.

Retomemos

Los ciclos permiten ejecutar una serie de acciones en forma reiterada (iterativa). De acuerdo a la forma en que se decide construir la repetición podemos clasificarlas en las siguientes variantes:

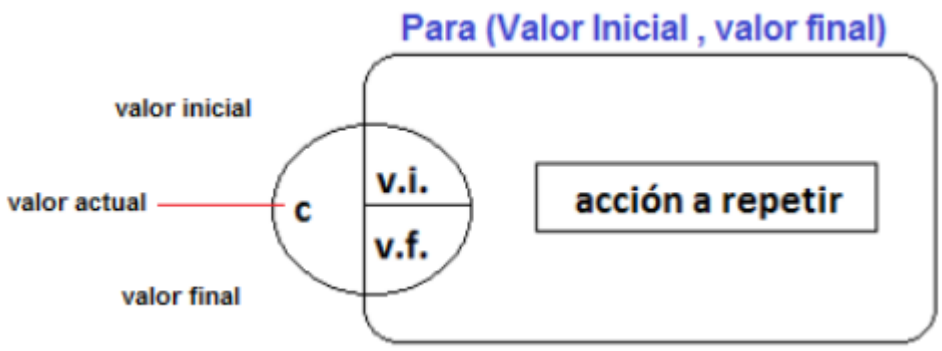
- Ciclo exacto:
  - Tomando un rango de valores inicial - final, se repite el ciclo **Para** cada valor intermedio dentro de ese rango elegido
- Ciclos condicionales:
  - **Mientras** se cumpla la condición **hacer** (MIENTRAS – HACER).
  - **Hacer** al menos una vez y repetir **mientras** se cumpla la condición (HACER – MIENTRAS).

Como característica principal observamos que las estructuras de repetición necesitan SIEMPRE de una variable que va modificando su valor DENTRO del ciclo y es evaluada en su proposición lógica (incluso en el ciclo PARA).

2.1. Estructura de repetición PARA

Esta estructura es usada para repeticiones donde se sabe la cantidad de repeticiones que deben realizarse antes de ingresar al ciclo. Más adelante veremos que puede tener matices y mejoras pero nunca se puede modificar el valor inicial (v.i.) ni el valor final (v.f.) una vez que estamos dentro del ciclo.

Utiliza una variable (c) que va modificándose dentro de la misma estructura (tampoco la debemos modificar dentro del ciclo) y que es evaluada en la proposición lógica mientras no alcance el valor final.



Veamos un ejemplo de estructura de repetición para:

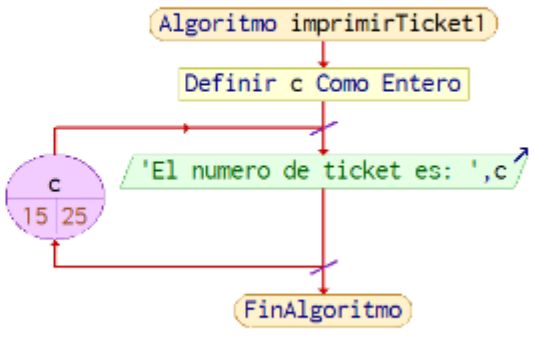
Se necesita realizar un programa que imprima 10 tickets por pantalla. Tomando valores de numeración del 15 al 25. Realizar prueba de escritorio para los dos ejemplos.

En el pseudocódigo

Vemos que definimos una variable llamada c del tipo entero y en el ciclo Para se le asigna un valor inicial (que para nuestro ejemplo es 15) y le indicamos que llegue al número 25.

Luego le indicamos que vaya "contando" o pasando entre ese rango de valores inicial y final "Con Paso 1", es decir, de 1 en 1. Si omitimos "Con Paso", por defecto el programa entenderá que queremos hacerlo de 1 en 1.

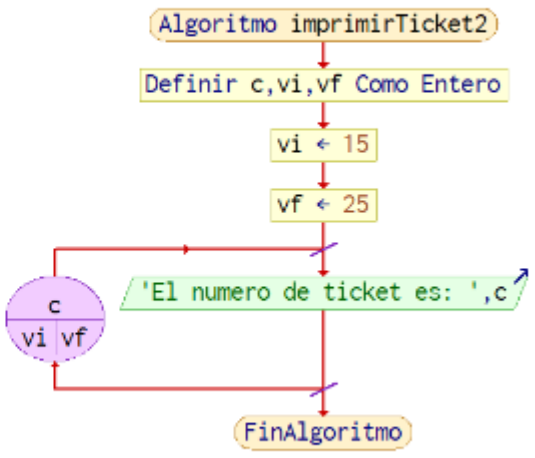
En este primer ejemplo utilizamos el rango con valores constantes entre 15 y 25

En pseudocódigo	En un diagrama de flujo
Algoritmo imprimirTicket1 Definir c Como Entero Para c = 15 Hasta 25 Con Paso 1 Hacer Escribir 'El numero de ticket es: ',c FinPara FinAlgoritmo	

También podemos tener ese rango definido en variables.

Para nuestro ejemplo llamaremos a esas variables vi y vf (valor inicial y valor final respectivamente).

En pseudocódigo	En un diagrama de flujo

`Algoritmo imprimirTicket2`
`  Definir c,vi,vf Como Entero`
`  vi = 15`
`  vf = 25`
`  Para c = vi Hasta vf Hacer`
`  Escribir 'El numero de ticket es: ',c`
`  FinPara`
`FinAlgoritmo`
	

¿Qué ventaja creés que podemos obtener entre los 2 ejemplos mostrados?

Pensemos que si lo tenemos como valores constantes nuestro programa siempre mostrará tickets desde el 15 al 25.

Pero ¿en el ejemplo de las variables no?

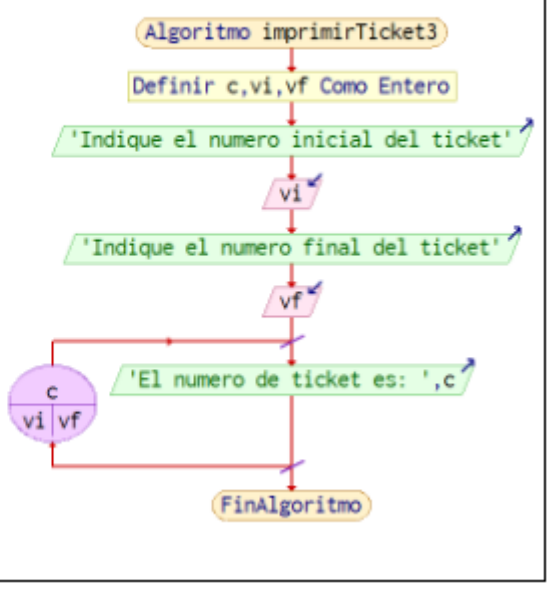
Claro, también hará lo mismo…..

Y ¿entonces?

Pero ahora tenemos la posibilidad de pedirle al usuario qué rango de tickets desea imprimir

Veamos cómo quedaría el ejemplo:

En pseudocódigo	En un diagrama de flujo

`Algoritmo imprimirTicket3`
`  Definir c,vi,vf Como Entero`
`  Escribir "Indique el numero inicial del ticket"`
`  Leer vi`
`  Escribir "Indique el numero final del ticket"`
`  Leer vf`
`  Para c←vi Hasta vf Hacer`
`  Escribir 'El numero de ticket es: ',c`
`  FinPara`
`FinAlgoritmo`
	

Otro ejemplo:

Ingresar 10 números enteros. Calcular e informar la cantidad de positivos, la cantidad de negativos y la cantidad de ceros.

```
Proceso Ejemplo1
    Definir Cp, Cn, Cc, num1, num Como Entero;
    Cp = 0;
    Cn = 0;
    Cc = 0;
    Para num1← 1 Hasta 10 Con Paso 1 Hacer
        Escribir "Ingrese un numero:";
        Leer num;
        Si num > 0 Entonces
            Cp = Cp + 1;
        SiNo
            Si num < 0 Entonces
                Cn = Cn + 1;
            SiNo
                Cc = Cc + 1;
            FinSi
        FinSi
    FinPara
    Escribir "La cantidad de numeros positivos es:", Cp;
    Escribir "La cantidad de numeros negativos es:", Cn;
    Escribir "La cantidad de ceros es:", Cc;
FinProceso
```

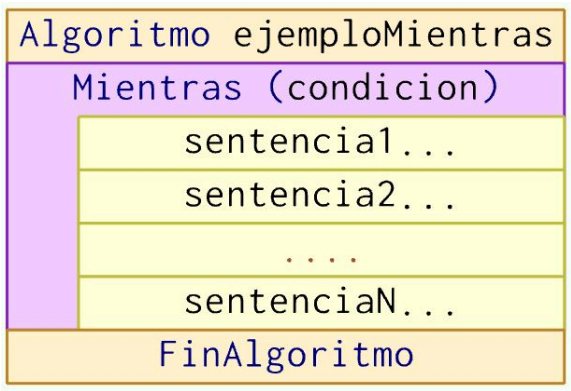
2.2. Estructura de repetición (mientras - hacer)

Esta estructura de repetición evalúa la proposición lógica antes de comenzar el ciclo de repetición por lo que si la evaluación (proposición lógica) no es verdadera la iteración no se llevará a cabo.

> En otras palabras, se puede decir que dada una condición x, la acción z se podrá ejecutar 0 o n veces, dependiendo del grado de verdad de x.

Funciona repitiendo su objetivo mientras la expresión de la condición sea cierta o verdadera. Cuando esta es falsa, el bucle se detiene.

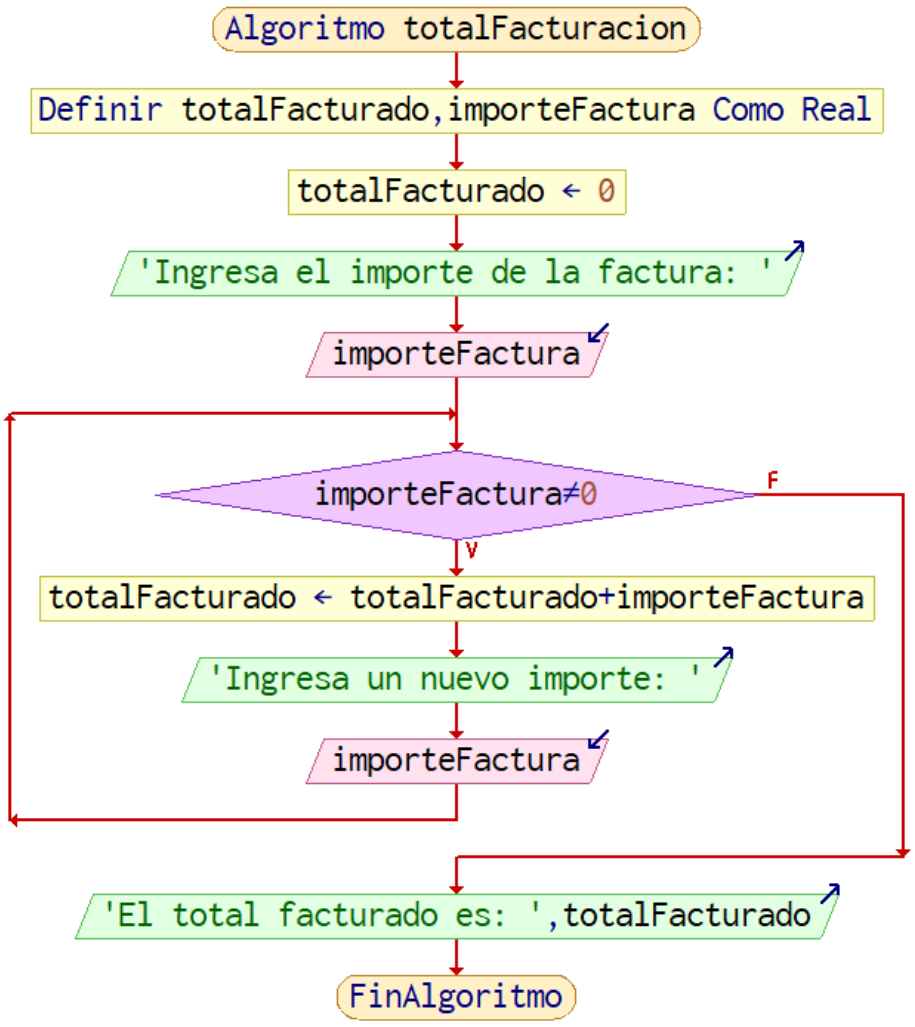
El valor de la condición se comprueba al principio del bucle. Esto significa que si la condición es falsa al iniciarse, el bucle no se ejecutará ni siquiera una vez. Por eso decimos que este ciclo se va a repetir de 0 a N veces. No sabemos cuántas veces será ese N ya que dependemos de una condición lógica.



Veamos un ejemplo.

En un comercio se desea obtener el total facturado en el día mediante el ingreso del importe de las ventas realizadas. Para indicar el fin del día se ingresa como importe de la venta el valor cero (0).

```
Algoritmo totalFacturacion
    Definir totalFacturado, importeFactura Como Real
    totalFacturado = 0
    Escribir "Ingresa el importe de la factura: "
    Leer importeFactura
    Mientras importeFactura ≠ 0 Hacer
        totalFacturado = totalFacturado + importeFactura
        Escribir "Ingresa un nuevo importe: "
        Leer importeFactura
    FinMientras
    Escribir "El total facturado es: ", totalFacturado
FinAlgoritmo
```



En este ejemplo vemos cómo se hace lo que llamaremos una “lectura anticipada” al comienzo del ciclo, que hace que al momento de llegar a evaluar la condición tenga un valor cargado en la variable (importeFactura) a ser evaluada.

¿Qué pasaría si el usuario como primer importeFacturado ingresara el valor 0 (cero)?

Sencillamente no ingresará al ciclo ya que evalúa la condición (importeFacturado != 0) le dará falso y, como vimos, cuando la condición es falsa, no ingresa al ciclo.

Si el importeFacturado no es 0 (cero), ingresa al ciclo y comenzamos a acumular el importeFacturado.

Acumular implica sumar al valor que tenía un nuevo valor ingresado.

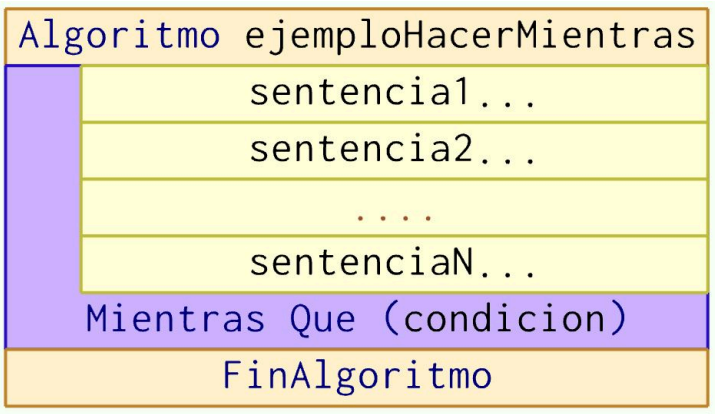
Tengamos en cuenta que la variable que oficia de acumulador, debe ser inicializada previamente.

2.3. Estructura de repetición (hacer - mientras)

Esta estructura de repetición a diferencia de la anterior siempre ejecuta al menos una vez las instrucciones que contiene porque la evaluación de la proposición lógica (condición) se hace al final de la primera repetición.

> Por eso podemos decir que se ejecuta desde 1 a N veces., (dependiendo el grado de verdad de lo que se evalúa).

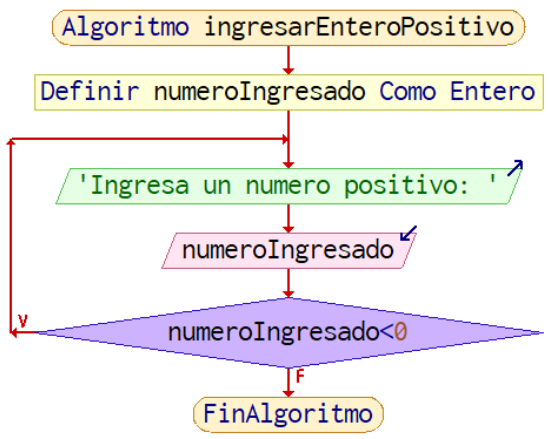
Es similar al anterior pero tiene la particularidad de ejecutar el proceso en primer término y luego evaluar la condición. De esta manera la cantidad mínima de veces que se ejecuta el proceso es una, mientras que en el caso anterior el proceso puede no ejecutarse si la condición es falsa la primera vez que se evalúa.



Veamos un ejemplo.

Se necesita pedir un número entero positivo al usuario tantas veces como sea necesario hasta que se cumpla.

```
Algoritmo ingresarEnteroPositivo
    Definir numeroIngresado Como Entero
    Repetir
        Escribir "Ingresa un numero positivo: "
        Leer numeroIngresado
    Mientras Que numeroIngresado < 0
FinAlgoritmo
```



En síntesis:

Tenemos 2 tipos de ciclos, exacto y condicional.

- El exacto (Para) nos permite ciclar exactamente una determinada cantidad de veces.
- Los ciclos condicionales dependen de una condición lógica, que se cumpla para seguir ciclando.
- Dentro de los condicionales tenemos el ciclo Mientras que primero evalúa la condición lógica y de ser verdadera ingresa al ciclo para ejecutar las sentencias y,
- el ciclo Hacer Mientras que al menos una vez ejecuta lo que hay dentro del ciclo y luego evalúa la condición lógica y de ser verdadero vuelve a realizar las repeticiones.