Algoritmos: Núcleo temático 6

Germán Alberto Angarita Henao
Instructor SENA
Centro de Comercio y Turismo
Armenia (Q)
2021

Estructuras de repetición

- Ciclos condicionados al inicio
- Ciclos condicionados al final

Introducción a estructuras de repetición

Es muy común encontrar en la práctica algoritmos cuyas operaciones se deben ejecutar un número repetido de veces. Si bien las instrucciones son las mismas, los datos sobre los que se opera pueden variar.

Tipos de estructuras de repetición

Todo ciclo debe terminar de ejecutarse luego de un número finito de veces, por lo que es necesario en cada iteración del mismo, evaluar las condiciones necesarias para decidir si se debe seguir ejecutando o si debe detenerse. En todo ciclo, siempre debe existir una condición de parada o fin de ciclo la cual puede estar al inicio o al final. Por lo cual se tienen:

- Ciclos condicionados al inicio.
- Ciclos condicionados al final.

Ciclos condicionados al inicio

Las estructuras de repetición condicionadas al inicio tienen la particularidad de preguntar si cumplen con cierta condición antes de entrar en el ciclo. Estas estructuras de repetición se dividen en:

- Para-Hasta
- Mientras

Ciclo Para

La estructura de repetición Para conocida comúnmente como FOR, es la adecuada para utilizar en un ciclo que se ejecutará un número definido de veces. Este tipo de estructura está presente en todos los lenguajes de programa orientados a objetos y estructurados.

Pseudocódigo	Diagrama de flujo

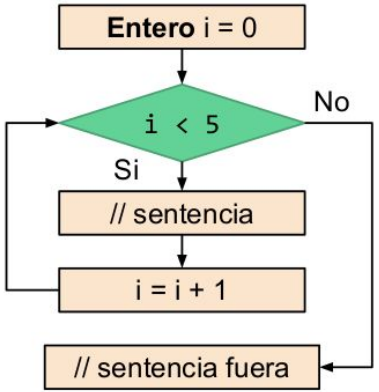
Entero i
 	
Para i = 0 **Hasta** 5 **Incremento** 1
  // sentencia dentro
FinPara
  // sentencia fuera
	

Tabla 1. Representación del ciclo para con pseudocódigo y diagramas de flujo

Ejemplo de un problema usando ciclo Para

Programe un algoritmo que sume los números del 1 al 1000.

```
Algoritmo SumaDeLosPrimerosNumeros

   Entero sumaDeNumeros

   sumaDeNumeros = sumarPrimerosNumeros( )

   imprimir( sumaDeNumeros )

FinAlgoritmo
```

```
Funcion Entero sumarPrimerosNumeros( )

   Entero suma = 0, i

   Para i = 1 Hasta 1000 Incremento 1
      suma = suma + i
   FinPara

   Retorne suma

FinFuncion
```

Ejercicio

Programe un algoritmo que sume los números pares del 1 al 700.

Ciclo Mientras

La estructura de repetición `Mientras` conocida comúnmente como `While`, es la es- tructura adecuada para utilizar en un ciclo cuando no sabemos el número de ve- ces que éste se ha de repetir.

Pseudocódigo	Diagrama de flujo

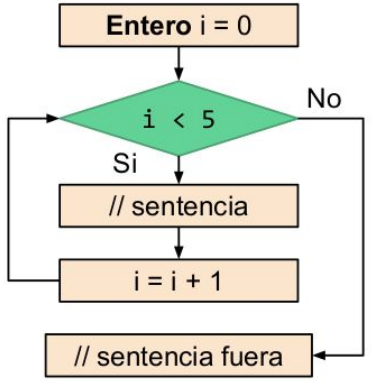
Entero i
 	
Mientras i < 5
// sentencia dentro
i = i + 1
FinMientras
// sentencia fuera
	

Tabla 2. Representación del ciclo mientras con pseudocódigo y diagramas de flujo

Ejemplo de un problema usando ciclo Mientras

Programe un algoritmo que sume los 100 números negativos más pequeños.

```
Algoritmo SumaDeLosNegativosPequenos

  Entero sumaDeNegativos

  sumaDeNegativos = sumarNegativosPequenos( )

  imprimir( sumaDeNegativos )

FinAlgoritmo
```

```
Funcion Entero sumarNegativosPequenos( )

   Entero suma = 0, i = -1

   Mientras i >= -100
      suma = suma + i
      i = i - 1
   FinPara

   Retorne suma

FinFuncion
```

Ejercicio

Programe un algoritmo que dado un número entero devuelva su factorial. Revuelva el problema usando el ciclo mientras.

Ciclos condicionados al final

Las estructuras de repetición condicionadas al final tienen la particularidad de ejecutar primero las sentencias dentro del ciclo y luego preguntar si cumple la condición para continuar, por lo cual ejecuta siempre mínimo una vez las instrucciones dentro del ciclo. La estructura de repetición que se usa para este tipo de ciclos es:

- Haga-MientrasQue

Ciclo Haga-MientrasQue

La estructura de repetición `Haga-MientrasQue` conocida comúnmente como `do-While`, es la estructura adecuada para utilizar en un ciclo cuando sabe que se ejecutará mínimo una vez.

Pseudocódigo	Diagrama de flujo

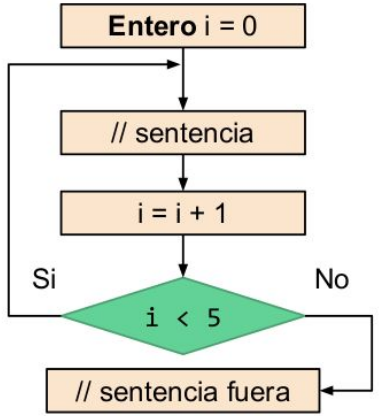
Entero i
 	
Haga
  // sentencia dentro
  i = i + 1
MientrasQue i < 5
  // sentencia fuera
	

Tabla 3. Representación del ciclo haga-mientrasque con pseudocódigo y diagramas de flujo

Ejemplo de un problema usando ciclo Haga-MientrasQue

Programe un algoritmo que lea dos números enteros y devuelva la suma entre los elementos leídos (incluyendolos). Realice la respectiva prueba de escritorio.

```
Algoritmo SumaDeIntervalo

  Entero menor, mayor, sumaIntervalo

  leer ( menor )

  leer ( mayor )

  sumaIntervalo = sumarIntervalo(menor,mayor)

  imprimir( sumaIntervalo )

FinAlgoritmo
```

```
Funcion Entero sumarIntervalo( Entero min, Entero max )

   Entero suma = 0, i = min

   Haga
      suma = suma + i
      i = i + 1
   Mientras i <= max

   Retorne suma

FinFuncion
```

Ejercicio

Programe un algoritmo que lea un número entero y devuelva el número que hace referencia a esa posición en la serie fibonacci. Ejemplo, si se tiene:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

y se desea obtener la posición 7, el resultado sería: 8

si se desea obtener la posición 10, el resultado sería: 34.

Haga la respectiva prueba de escritorio

Material adaptado de presentaciones pertenecientes a Ingeniería de Sistemas y Computación Universidad del Quindío - Einer Zapata, ezapata@uniquindio.edu.co - Carlos A. Flórez, caflorez@uniquindio.edu.co