

Fundamentos de la Programación

Repasemos... ¿QUÉ ES UNA ESTRUCTURA REPETITIVA ?

Durante el proceso de creación de programas, es muy común encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras de algoritmos que permiten repetir una o varias acciones, un número determinado de veces.

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.

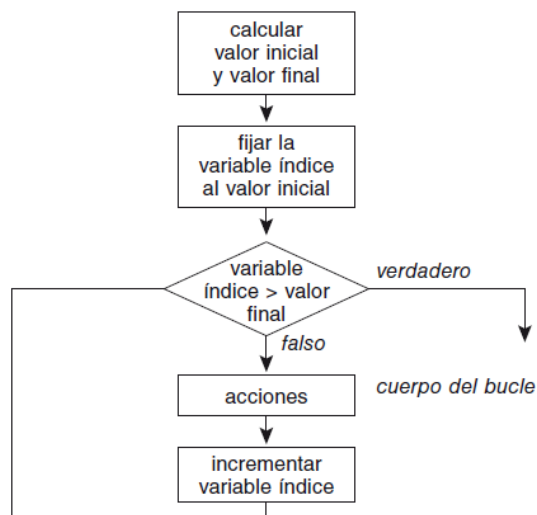
Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuando deja de repetirse.

Hay distintos tipos de bucles:

- Mientras
- Hacer Mientras
- Para

Estructura Para

La **estructura Para** (denominada "for" en varios lenguajes de programación, como C, C++, Java, Python y JavaScript, entre otros.) **te permite ejecutar un conjunto de acciones para cada paso de un conjunto de elementos**. Generalmente, consta de tres componentes: inicialización, finalización e incremento. Comienza con un valor inicial de una variable llamada índice y las acciones especificadas se ejecutan un número determinado de veces, hasta que el valor del índice alcanza el valor final. La variable índice se incrementa en uno en cada iteración y si este nuevo valor no supera al valor final, las acciones se ejecutan nuevamente. En resumen, **las acciones dentro del bucle se ejecutan para cada valor del índice desde el valor inicial hasta el valor final, con un incremento de uno en uno**.




Para `variable_numerica` \leftarrow `valor_inicial` **Hasta** `valor_final` **Con Paso** `paso` **Hacer**
`secuencia_de_acciones`
Fin Para

Aquí hay algunos puntos clave para entender cómo funciona esta estructura:

1. **Inicialización de la variable índice:** Antes de iniciar el bucle, se establece el valor inicial de la variable índice. Esto se hace generalmente utilizando la palabra clave "Desde" seguida del valor inicial. Por ejemplo: Desde $i = 1$. Recuerda, en el caso de PSeINT, definir la variable índice como entero.
2. **Condición de continuación del bucle:** Se evalúa una condición en cada iteración para determinar si el bucle debe continuar ejecutándose. Si esta condición es verdadera, el bucle continúa; si es falsa, el bucle termina. Por ejemplo: Hasta que $i > 10$.
3. **Incremento o decremento de la variable índice:** En cada iteración, la variable índice se incrementa o decrementa según sea necesario. Esto se hace generalmente utilizando la palabra clave "Con Paso" seguida del valor del incremento o decremento. Por ejemplo: Con Paso 1 para un incremento de 1, o Con Paso -1 para un decremento de 1.
4. **Variable índice de control:** La variable índice se utiliza para controlar el número de iteraciones del bucle y suele tener nombres como "i", "j" o "k".

Es importante tener en cuenta que el valor inicial de la variable índice debe ser menor o mayor que el valor final, dependiendo de si se utiliza un incremento o decremento. Además, el paso debe ser positivo si el valor inicial es menor que el valor final, y negativo si es mayor.

 **TIP:** En la estructura PARA tenemos el control absoluto de las repeticiones, ya que podemos determinar la inicialización, límite y aumento del valor de i.

Ejemplo:

```
1  Algoritmo EjemploPara
2
3  Definir i Como Entero
4
5  Para i ← 1 Hasta 10 Con Paso 1 Hacer
6      .....
7      Escribir "La tabla del 2 es: " i * 2
8      .....
9  Fin Para
10
11
12 FinAlgoritmo
13
```

PSelnt - Ejecutando proceso EJEMPLOPARA


```
*** Ejecución Iniciada. ***
La tabla del 2 es: 2
La tabla del 2 es: 4
La tabla del 2 es: 6
La tabla del 2 es: 8
La tabla del 2 es: 10
La tabla del 2 es: 12
La tabla del 2 es: 14
La tabla del 2 es: 16
La tabla del 2 es: 18
La tabla del 2 es: 20
*** Ejecución Finalizada. ***
```

Estructura Repetitivas anidadas

En la programación, al igual que con las estructuras condicionales, **es posible anidar estructuras repetitivas**. Los bucles anidados son una técnica poderosa que nos permite ejecutar conjuntos de instrucciones de manera repetida dentro de otros conjuntos de instrucciones repetidas. Imagina que tienes dos bucles, uno dentro del otro. Esto quiere decir que podemos diseñar por necesidad que una estructura se encuentre dentro de otra.

Para empezar, el primer bucle se inicializa y se ejecuta según su condición, y dentro de este, otro bucle, anidado, también se inicializa y ejecuta de manera independiente.

Estos bucles anidados son comunes en algoritmos de búsqueda y ordenación, donde necesitamos comparar elementos entre sí en diferentes niveles de anidamiento. Sin embargo, es importante tener en cuenta que la anidación excesiva de bucles puede aumentar la complejidad y el tiempo de ejecución de un programa. Por lo tanto, es fundamental utilizarlos con precaución y optimizarlos según sea necesario para garantizar un rendimiento óptimo de nuestro código.

 **TIP:** Es muy importante asegurarse de evitar el solapamiento. El solapamiento en estructuras de programación ocurre cuando una estructura de código, como un bucle, una función o una condición, se inicia dentro de otra estructura pero no finaliza dentro de ella, sino que se extiende fuera de su alcance. Esto puede generar errores de lógica, dificultar la lectura y el mantenimiento del código, y producir comportamientos inesperados en el programa.

Ejemplo:

```
Algoritmo BuclesAnidados
  Definir i,j como Entero
  // Bucle externo
  Para i ← 0 Hasta 2 Con Paso 1 Hacer
    Escribir "Bucle externo: i = ", i
    // Bucle interno
    Para j ← 0 Hasta 1 Con Paso 1 Hacer
      Escribir "  Bucle interno: j = ", j
    FinPara
  FinPara
FinAlgoritmo
```

Explicación del Ejemplo de Bucles Anidados

1. Bucle Externo:

- Para i ← 0 Hasta 2 Con Paso 1 Hacer: Este bucle externo controla la variable i, que toma valores desde 0 hasta 2 (inclusive) en pasos de 1.

2. Acciones del Bucle Externo:

- Escribir "Bucle externo: i = ", i: Imprime el valor actual de i.

3. Bucle Interno:

- Para j ← 0 Hasta 1 Con Paso 1 Hacer: Este bucle interno controla la variable j, que toma valores desde 0 hasta 1 (inclusive) en pasos de 1.

4. Acciones del Bucle Interno:

- Escribir " Bucle interno: j = ", j: Imprime el valor actual de j, con una sangría adicional para mostrar que está dentro del bucle externo.