# UNIDAD 3
# ESTRUCTURAS REPETITIVAS

# UD3. ESTRUCTURAS REPETITIVAS

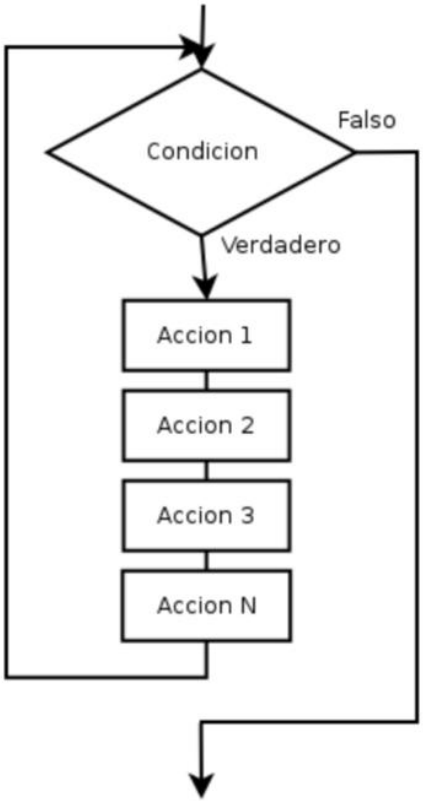
## 1. INTRODUCCIÓN

Las **instrucciones repetitivas (o bucles)** son aquellas que **permiten variar o alterar la secuencia normal de ejecución de un programa** haciendo posible que un grupo de operaciones (acciones) se repita un número determinado o indeterminado de veces, dependiendo del cumplimiento de una condición.

Veremos tres tipos: Bucle Mientras (WHILE), Bucle Hacer-Hasta (DO-WHILE) y Bucle Para (FOR)

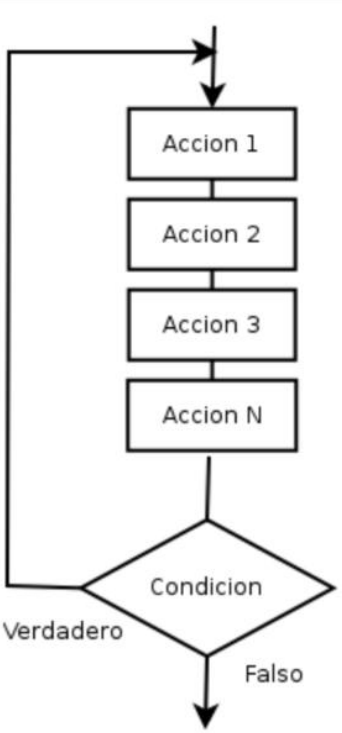
## 2. ESTRUCTURA MIENTRAS (WHILE)

En la estructura Mientras o "**WHILE**" el bloque de acciones **se repite mientras la condición sea cierta**, evaluándose siempre la condición antes de entrar en el bucle, por ello es posible que las acciones no se ejecuten nunca.

| Pseudocódigo | Ordinograma |
|---|---|
| Mientras Condición hacer<br>Acción 1<br>Acción 2<br>...<br>Acción N<br>FinMientras<br><br>**CODIGO JAVA:**<br><br>`while (condicion) {`<br>`  accion1();`<br>`  accion2();`<br>`  // ...`<br>`  accionN();`<br>`}` |  |

## 3. ESTRUCTURA HASTA (DO-WHILE)

En la estructura hasta o "**DO-WHILE**", el bloque de instrucciones **se repite mientras que la condición sea cierta**, y la condición se evalúa al final del bloque por lo que siempre se ejecutarán al menos una vez el bloque de instrucciones.

| Pseudocódigo | Ordinograma |
|---|---|
| Repetir<br>Acción 1<br>Acción 2<br>...<br>Acción N<br>Mientras Condición.<br><br>**CODIGO JAVA:**<br><br>`do { accion1(); accion2(); // ... accionN(); } while (condicion);` |  |

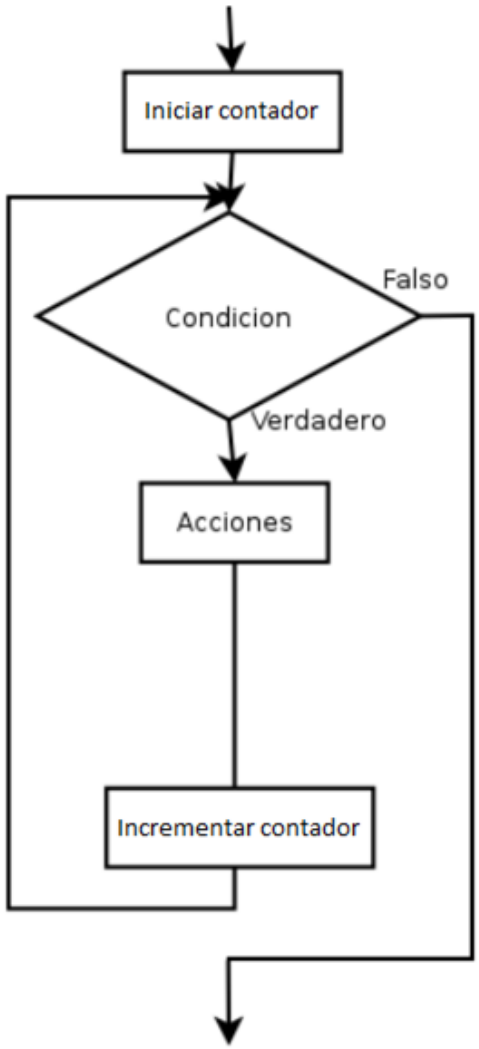
## 4. ESTRUCTURA PARA (FOR)

En la estructura Para o “FOR” se conoce de antemano el número de veces que se ejecutará el bloque de instrucciones.

> El bloque de acciones **se repite mientras que la condición sea cierta**, **evaluándose** siempre la condición **antes de entrar en el bucle**, por ello es posible que las acciones no se ejecuten nunca.

Esta explicación es idéntica a la del bucle WHILE, pero un bucle FOR debe cumplir las siguientes características:

1. La variable contador se inicializa con un valor inicial.
2. La condición siempre debe ser: variable_contador <= valor_final
3. En cada interacción, la variable contador se incrementa en un determinado valor.

| Pseudocódigo | Ordinograma |
|---|---|
| Para Var_Cont de ValorInicial a ValorFinal con Incremento = n<br>Acción 1<br>Acción 2<br>...<br>Acción N<br>FinPara<br><br>**CODIGO JAVA:**<br><br>`for (int varCont = valorInicial; varCont <= valorFinal; varCont += n) {`<br>`  accion1();`<br>`  accion2();`<br>`  // ...`<br>`  accionN();`<br>`}` |  |

## 5. FORMAS DE ACABAR UN BUCLE

Uno de los peligros que se corren cuando se escribe un bucle es que éste no acabe nunca, lo que se denomina **bucle infinito**, para no cometer este error grave debemos recordar que las condiciones de los bucles deben poder cambiar dentro del bucle, es decir que si por ejemplo utilizamos una variable comparada con una constante, dicha variable debe poder cambiar de valor dentro del bucle.

> Las estructuras repetitivas deben incluir un mecanismo para que éstas se acaben.

Algunos de los métodos más usados para esa labor son los siguientes:

1. **Cuando sabemos el número de veces que se va a repetir la estructura, utilizaremos un contador.**

   Por ejemplo, en un enunciado del tipo: "imprimir la tabla del 7", sabemos que el proceso va desde 1 a 10, por lo tanto, usaremos un contador.

   cont = 1
   Mientas cont <=10
   Escribir cont * 7
   cont = cont + 1
   FinMientras

```
int cont = 1;
while (cont <= 10) {
    System.out.println(cont * 7);
    cont++;
}
```

2. **Preguntando si queremos seguir en el bucle.**

   Por ejemplo, en un ejercicio del tipo "introducir N alumnos y hallar su media", debemos preguntar si queremos introducir más alumnos:

   . . .
   seguir="s"
   Mientras ((seguir="s") o (seguir="S"))
   . . .
   Escribir "Introducir más alumnos?"
   Leer seguir
   FinMientras
   . . .

```java
Scanner sc = new Scanner(System.in);

String seguir = "s";
while (seguir.equalsIgnoreCase("s")) {
    // ... aquí tus acciones (pedir nombre/nota, acumular, etc.)

    System.out.print("¿Introducir más alumnos? (s/n): ");
    seguir = sc.nextLine().trim();
}
```

3. **Usando un valor centinela.**

   Así, en el caso del siguiente problema: “Introducir N notas hasta introducir un 10”:

   . . .

   Leer nota

   Mientras (nota <> 10)

   . . .

   Leer nota

   FinMientras

   . . .

```java
Scanner sc = new Scanner(System.in);
int nota;

System.out.print("Introduce nota (10 para terminar): ");
nota = Integer.parseInt(sc.nextLine().trim());

while (nota != 10) {
    // ... aquí procesas la nota (acumular, contar, etc.)
    System.out.print("Introduce nota (10 para terminar): ");
    nota = Integer.parseInt(sc.nextLine().trim());
}
```

4. **Usando un interruptor que tomará el valor lógico true o false.**

   En un ejemplo como “Repetir ciertas instrucciones mientras la condición sea cierta”:

   . . .

   Mientras (SW = Verdadero)

   . . .

   FinMientras

   . . .

```java
boolean SW = true;
while (SW) {
    // ... tus instrucciones
    // en algún momento, cambia la condición:
    // SW = condicionSigueSiendoVerdadera();
    // o para salir:
    // SW = false;
}
```

# 6. ELEMENTOS AUXILIARES

Los elementos auxiliares son variables que realizan funciones específicas dentro de un programa, y por su gran utilidad, frecuencia de uso y peculiaridades, conviene hacer un estudio separado de las mismas.

## *6.1 Contadores*

Si vamos a repetir una acción un número determinado de veces y esa variable se va a incrementar siempre en una cantidad constante, se denomina **contador**. Sería útil llamarla algo así como CONT, CONTA, CONTADOR... Si tuviéramos varios contadores dentro de un programa podríamos llamarlos CONT1, CONT2...

Se utilizan en los siguientes casos:

- Para contabilizar el número de veces que es necesario repetir una acción (variable de control de un bucle).
- Para contar un suceso particular solicitado por el enunciado del problema. Un contador debe inicializarse a un valor inicial (normalmente a cero) e incrementarse cada vez que ocurra un suceso.

## *6.2 Acumuladores*

Si por el contrario, dicho objeto se va **incrementando de forma variable** se denomina **acumulador**. Deberemos llamarla ACU, ACUM, ACUMULA, ACUMULADOR, SUMA, ... u otra palabra significativa.

Se utiliza en aquellos casos en que se desea obtener el total acumulado de un conjunto de cantidades, siendo inicializado con un valor cero.

También en ocasiones hay que obtener el total acumulado como producto de distintas cantidades, en este caso se inicializará a uno.

Por ejemplo: imprimir la suma de N edades.

## *6.3 Interruptores*

Por último, tenemos ciertas variables que pueden **tomar dos valores: cierto o falso**. Se les denomina **interruptores o switches** y su función es que ciertas instrucciones se ejecuten mientras tenga un valor determinado. A las variables de este tipo las denominaremos SW.

Se utiliza para:

- Recordar que un determinado suceso a ocurrido o no en un punto determinado del programa, y poder así realizar las decisiones oportunas.
- Hacer que dos acciones diferentes se ejecuten alternativamente dentro de un bucle.

Por ejemplo: introducir N edades y acabar al introducir un 99.

## 7. EJEMPLOS

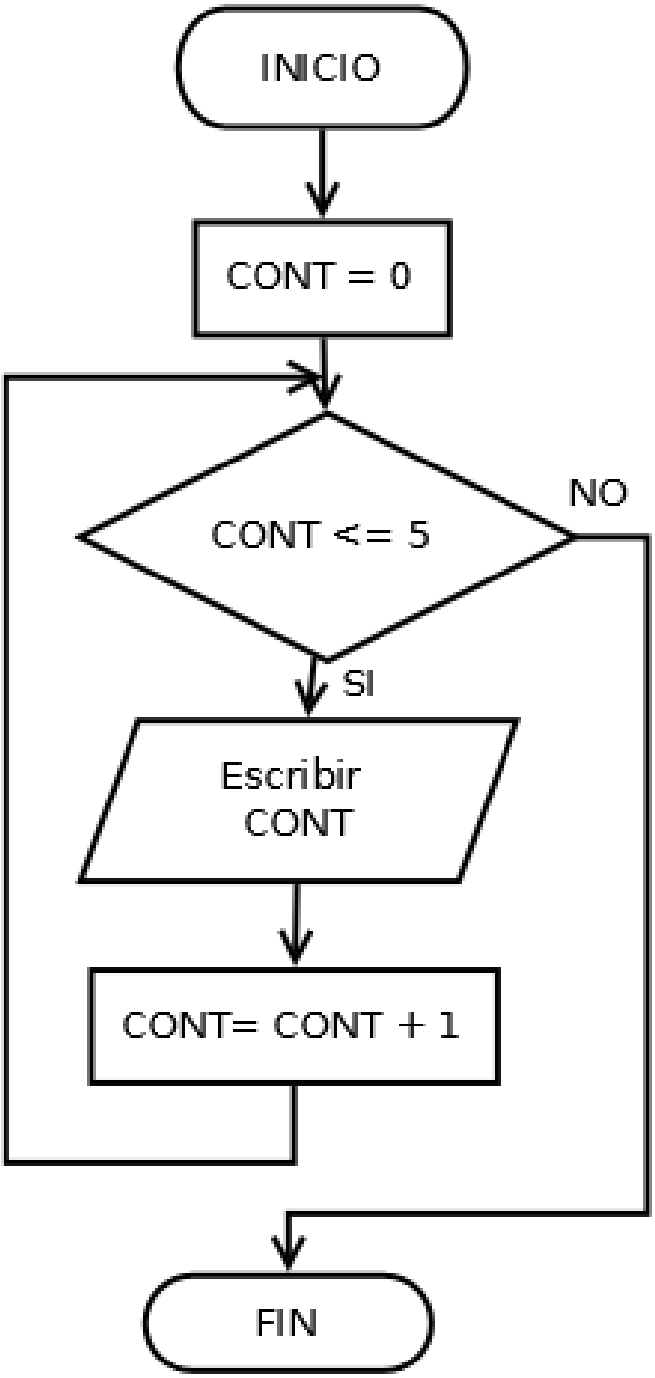
### *Ejemplo 1*

En el primer ejemplo vamos a ver cómo crear un programa que muestre por pantalla desde el número 0 hasta el 5.

Para un caso como éste, es interesante utilizar la estructura PARA (FOR) ya que conocemos el numero de veces que se va a ejecutar el bloque de instrucciones. En este caso 6 porque vamos desde el 0 hasta el 5.

Tenemos que tener en cuenta que:

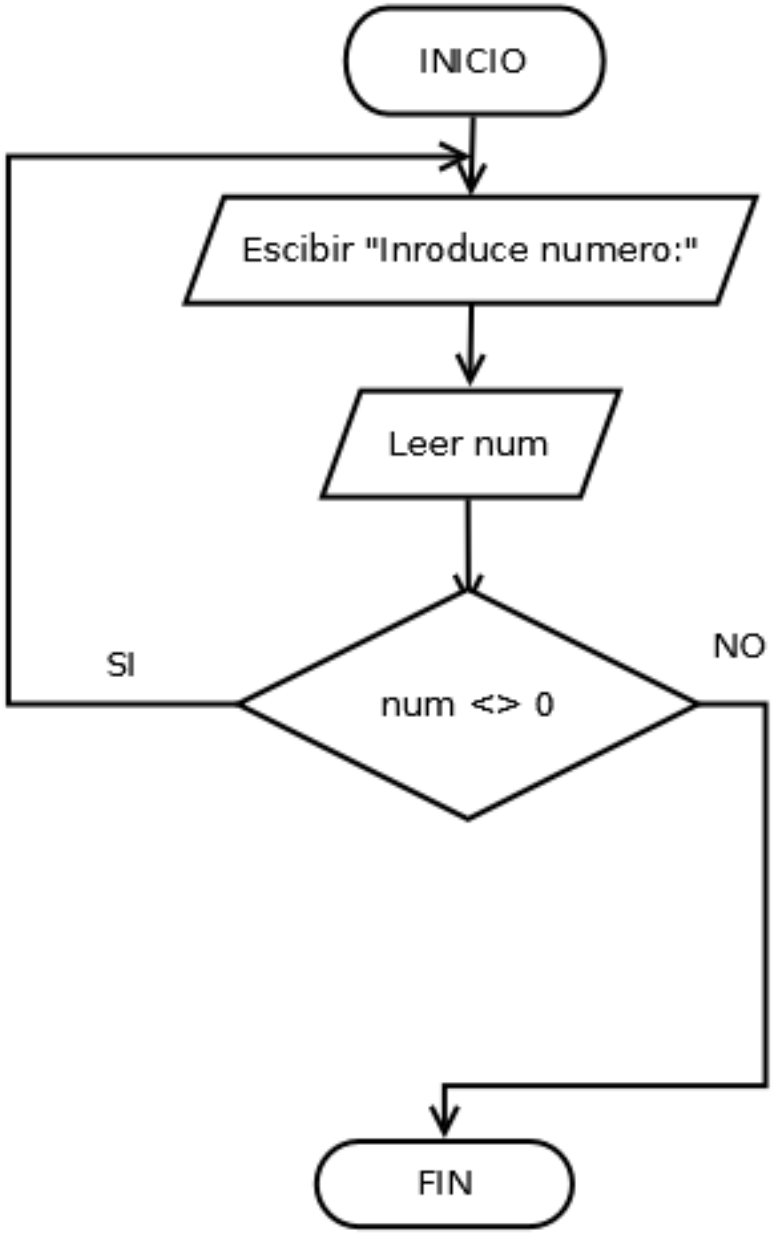
- El bloque se repite mientras la condición sea cierta.
- La condición se evalúa siempre antes de entrar en el bucle.
- La variable contador se inicializa con el valor inicial. En este caso 0
- La condición es: cont<=valor_final
- En cada iteración la variable contador se incrementa en un determinado valor. En este caso 1
- El orden del bucle será:
    1. La inicialización debe estar siempre inmediatamente por encima de la condición y de la flecha de retorno del bucle. (Ver ejemplo).
    2. La condición (<=)
    3. El incremento del contador.


INICIO
CONT = 0
CONT <= 5
NO
SI
Escribir
CONT
CONT= CONT + 1
FIN

## *Ejemplo 2*

En el segundo ejemplo vamos a crear un sencillo programa que pida por teclado al usuario números hasta que éste introduzca un 0.

Para este caso se utilizaría una estructura HASTA (DO-WHILE) ya que primero le pedimos al usuario el número y después comprobamos si es distinto de 0.



### *Ejemplo 3*

En el tercer ejemplo vamos a crear un sencillo programa que calcule la suma de los tres primeros números (del 1 al 3).

En este caso se utiliza un acumulador para obtener el resultado del sumatorio.