# Práctica 4. ESTRUCTURAS DE CONTROL ITERATIVAS

OBJETIVO:
- ✓ Conocer y manejar con habilidad los distintos tipos de estructuras de control iterativas

*No olvides que antes de pasar a la fase de diseño deberemos **analizar** exhaustivamente **el problema***

Las **sentencias de control iterativas** son aquellas que nos permiten variar o alterar la secuencia normal de ejecución de un programa haciendo posible que un grupo de acciones se ejecute más de una vez de forma consecutiva. Este tipo de instrucciones también recibe el nombre de **bucle**.

En los bucles se suelen utilizar algunas variables para unas tareas específicas, tales como **contadores, acumuladores** o **interruptores**.

- Contadores:

Un contador no es más que una variable destinada a contener un valor que se irá incrementando o decrementando en una cantidad fija. Se suelen utilizar para el control de procesos repetitivos.

- Acumuladores:

Un acumulador es una variable destinada a contener distintas cantidades provenientes de los resultados obtenidos en operaciones aritméticas previamente realizadas de manera sucesiva, lo que nos permitirá obtener el total acumulado de dichas cantidades. A diferencia de los contadores, no controlan los procesos repetitivos. Su inicialización depende de en qué operación matemática van a ser utilizados.

- Interruptores (switches):

Los interruptores, también denominados conmutadores o indicadores, son variables que pueden tomar dos únicos valores considerados como lógicos y opuestos entre sí a lo largo de todo el programa (0 o 1, 1 o –1, Verdadero o Falso, on/off, etc.).

Su objetivo es recordar en un determinado lugar del programa una ocurrencia o suceso acaecido o no con antelación, o hacer que dos acciones diferentes se ejecuten alternativamente en un proceso repetitivo. También deben ser inicializados. No se debe abusar de su utilización cuando no sea necesario.

**Ejercicio Resuelto 1.** **Implementa un programa que lea dos números naturales desde teclado. A continuación el programa debe contar y sumar los números que hay entre ellos. Por último debe imprimir en pantalla esa información.**

```
#include <iostream>
using namespace std;

main () {
     int i, cont, suma, num1, num2, aux;

     cout << "Introduce un número:";
     cin >> num1;
     cout << "Introduce otro número:";
     cin >> num2;
     if (num1>num2){
         aux=num1;
         num1=num2;
         num2=aux;
     }
     cont=0; suma=0;
     for (i=num1+1; i<num2; i++){
         cont++;
         suma = suma+i;
     }
     cout << "Hay " << cont << " números entre el " << num1 << " y el " << num2;
     cout << "\nLa suma es " << suma;
}
```

**Ejercicio Resuelto 2.** **Implementa un programa que lea un número natural por teclado y calcule y visualice el número formado por las mismas cifras pero en sentido inverso.**

```
#include <iostream>
using namespace std;

main () {
     int num;

     do {
             cout << "Introduce un número natural: ";
             cin >> num;
     } while (num <= 0);

     cout << "El número con las cifras en orden inverso es: ";
     while (num > 0) {
             cout << num%10;
             num = num/10;
     }
}
```

**Ejercicio Resuelto 3.** **Implementa un algoritmo que visualice por pantalla la siguiente figura, preguntando al usuario el número de estrellas que tiene que contener la línea final.**

```
*
**
***
****
*****
******
```

```
#include <iostream>
using namespace std;
main() {
   int n, fil, col;

   cout <<"Introduce el número de estrellas de la línea final:";
   cin >> n;
   for(fil=1; fil<=n; fil++){
       for(col=0; col<fil; col++){
           cout << '*';
       }
       cout << endl;   //fin de linea
   }
}
```