

# Estructuras de Control de Repetición

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

# Bucle

Un **bucle** es una estructura de control que **repite instrucciones**.

Un bucle entonces nos permite repetir un bloque de instrucciones determinado hasta que se cumpla cierta condición.

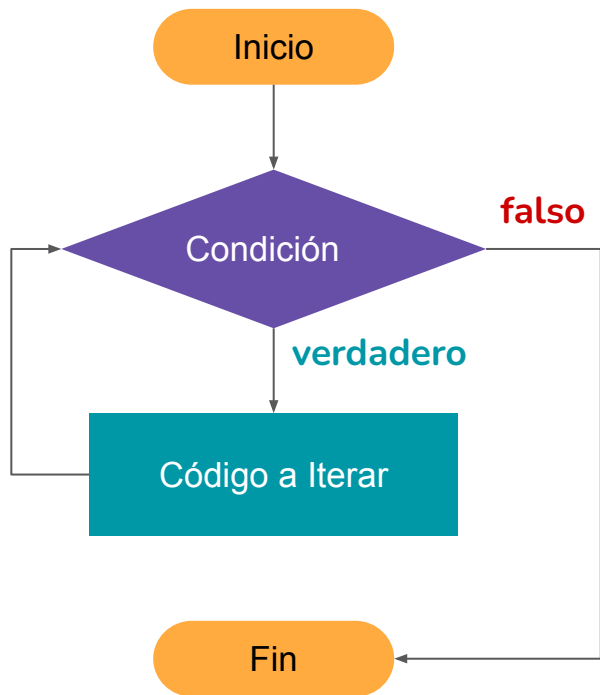
Cada repetición se suele llamar iteración.



# Ciclo While

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



## Ciclo While

La idea principal del ciclo while es: **MIENTRAS** se cumpla la condición **REALIZAR** estas acciones. Cuando la condición deje de cumplirse salimos del bucle y continúa el flujo del programa.

Muy importante: En el ciclo while la condición es lo primero que se evalúa, antes de ejecutar el código a iterar.

# Sintaxis: Ciclo while

Usamos la palabra reservada **while**, seguida de la condición entre paréntesis **()** y finalmente colocamos el código que se repetirá entre llaves **{}**

```
while (condicion) {  
    // codigo a ejecutar  
}
```

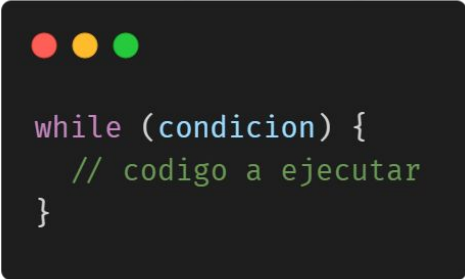
**Importante:** Necesitamos en el código a iterar insertar una variable de control que nos permita salir eventualmente el ciclo while. En caso contrario nuestro programa se quedará ciclado “infinitamente”.

# Ejemplos: Ciclo while

Usamos la palabra reservada **while**, seguida de la condición entre paréntesis **()** y finalmente colocamos el código que se repetirá entre llaves **{}**

```
> var index = 0;
  while(index < 11) {
    console.log(index);
    index++;
  };
```

Index++;  
Index = index + 1;  
Index -- ; => index = index - 1;



```
while (condicion) {
  // codigo a ejecutar
}
```

# Ejemplo #1: Ciclo while

```
while (condicion) {  
  // codigo a ejecutar  
}
```

```
> // Ejemplo #1  
  // Imprimir números del 0 al 10 en consola.  
var index = 0;  
while(index < 11) {  
  console.log(index);  
  index++;  
};
```

Resultado:

0

1

2

3

4

5

6

7

8

9

10

## Ejemplo #2: Ciclo while

```
while (condicion) {  
  // codigo a ejecutar  
}
```

```
> // Guarda un arreglo de valores introducidos  
// Si el usuario no introduce un valor, termina el ciclo.  
var arreglo = [];  
var userInput;  
while (!(userInput=="")) {  
  userInput = prompt("Ingresa cualquier carácter");  
  arreglo.push(userInput);  
}  
console.log("Introduciste estos valores: " + arreglo);
```

Resultado:

d

f

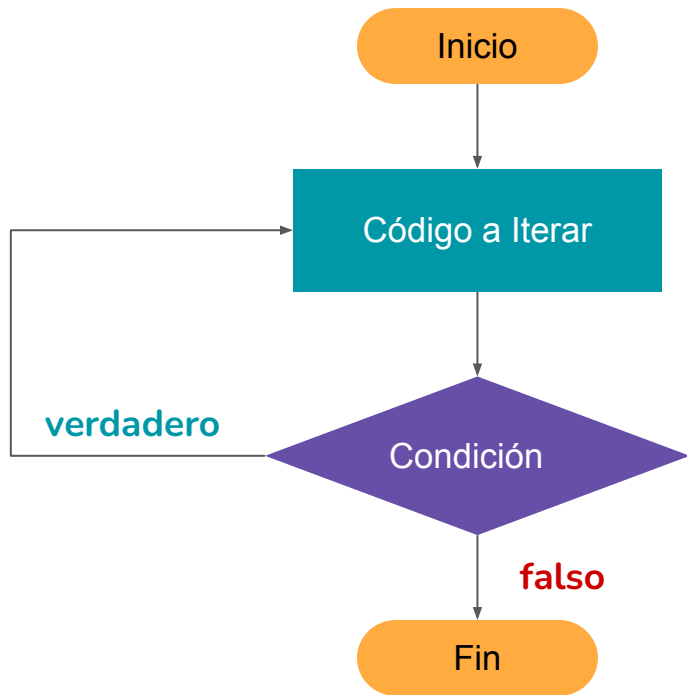
g

a

Introduciste estos valores: d,f,g,a,

¿Qué detalle encuentras?  
¿Cómo lo solucionarías?





## Ciclo Do While

Variante del ciclo While puro, con la diferencia que la primera vez siempre se ejecuta el código y posteriormente evalúa la condición para ver si se vuelve a ejecutar.

# Sintaxis: Ciclo do while

Usamos la palabra reservada **do**, seguido del el código que se repetirá entre llaves **{}**, seguido de la palabra reservada **while** y finalmente la condición a evaluar en cada iteración entre paréntesis **()**.

```
do {  
    // código a ejecutar  
}  
while (condicion);
```

# Ejemplo: Ciclo do while

```
do {  
    // código a ejecutar  
}  
while (condicion);
```

```
> // Ejemplo #1  
// Conteo de números del 1 al 10 en consola  
var contador = 0;  
do {  
    contador++;  
    console.log('Conteo:' + contador);  
} while (contador < 10);
```

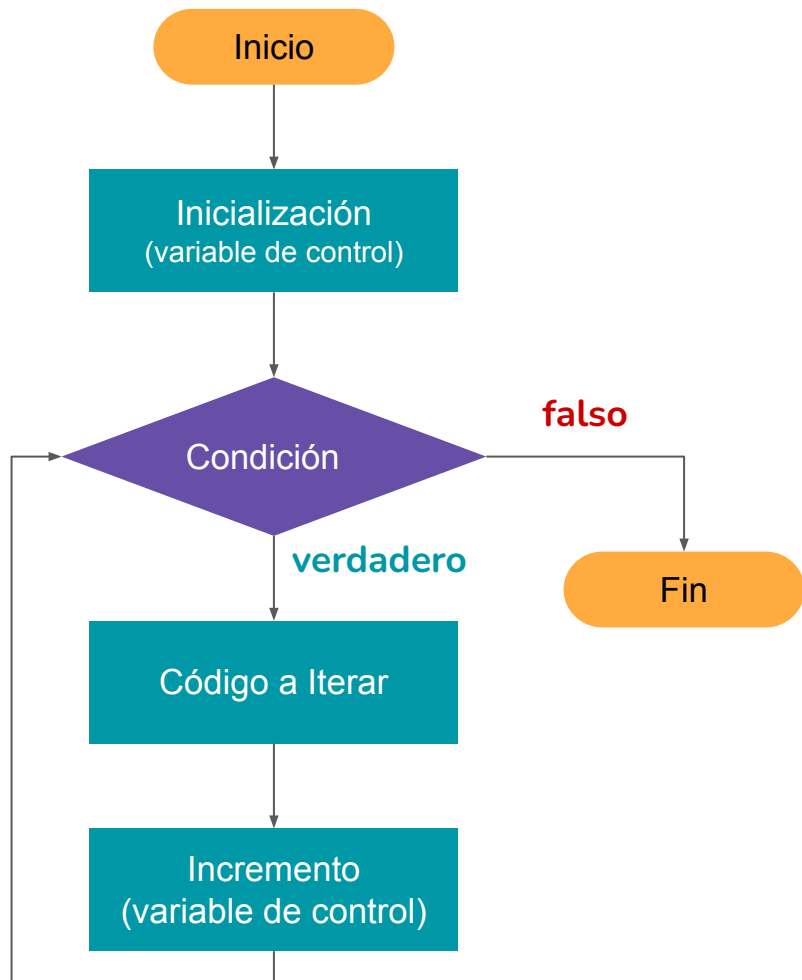
## Resultado:

```
Conteo:1  
Conteo:2  
Conteo:3  
Conteo:4  
Conteo:5  
Conteo:6  
Conteo:7  
Conteo:8  
Conteo:9  
Conteo:10
```

# Ciclo For

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



## Ciclo for

Un **bucle for** es un bucle que **repite** el bloque de instrucciones **un número predeterminado de veces**.

# Contadores y acumuladores

En muchos programas se necesitan variables que cuenten cuántas veces ha ocurrido algo (contadores) o que acumulen valores (acumuladores).



# Contador

Se entiende por contador una variable que lleva la cuenta del número de veces que se ha cumplido una condición.

```
> // Del 1 al 10 ¿Cuántos números son múltiplos de 2?  
var contador = 0;  
for (var index = 1; index <= 10; index++) {  
  if (index % 2 == 0) {  
    contador = contador + 1;  
    console.log(`${index} es múltiplo de 2`);  
  }  
}  
console.log(`De 0 a 10 existen ${contador} múltiplos de 2`);
```

2 es múltiplo de 2

4 es múltiplo de 2

6 es múltiplo de 2

8 es múltiplo de 2

10 es múltiplo de 2

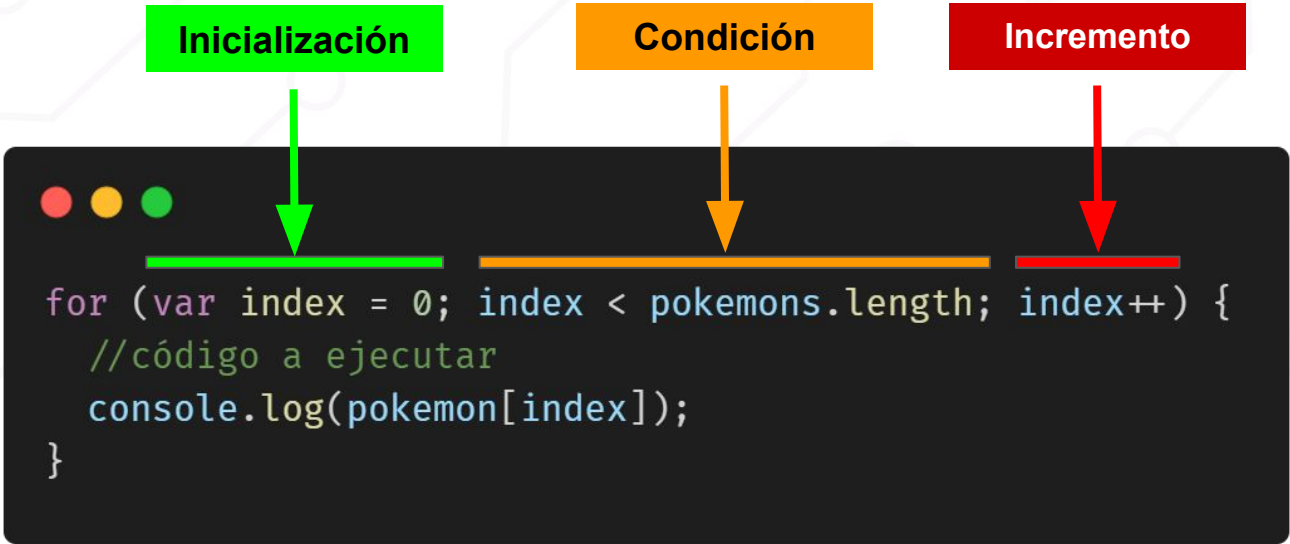
De 0 a 10 existen 5 múltiplos de 2

# Sintaxis Ciclo for

Inicialización

Condición

Incremento



```
for (var index = 0; index < pokemons.length; index++) {  
  //código a ejecutar  
  console.log(pokemon[index]);  
}
```

The diagram illustrates the three components of a for loop syntax with color-coded arrows and underlines. A green arrow points to the initialization part 'var index = 0', which is underlined in green. An orange arrow points to the condition part 'index < pokemons.length', which is underlined in orange. A red arrow points to the increment part 'index++', which is underlined in red. The code is displayed in a dark-themed editor window with standard window control buttons (red, yellow, green) in the top-left corner.

**Inicialización:** De la variable que llevará el conteo de cuantas veces se iterara.

**Condición:** Mientras la condición se cumpla, se ejecutará el código dentro de las llaves {}.

**Incremento:** Se ejecuta después de cada iteración, normalmente se coloca un **contador** que incremente en 1 la variable de inicialización.



# Acumulador

Se entiende por acumulador:

Una **variable** que **acumula** el resultado de una operación.

```
> var acumulador = 0;  
  for (var index = 0; index <= 4; index++) {  
    acumulador = acumulador + index;  
    console.log(acumulador);  
  }
```

0
1
3
6
10

For

VS

while

# ¿Cuándo usar *While* y cuándo *For*?

No existen reglas fijas, pero una buena recomendación para escoger entre ambas es el caso de si conozco o no el número de iteraciones que voy a realizar:

- Usamos el **ciclo for** para iterar un **arreglo**.
- Usamos el **ciclo for** cuando sabemos que el código a iterar debería ejecutarse **n veces**.
- Usamos el **ciclo while** para la variable que nos permite leer un archivo.
- Usamos el **ciclo while** para preguntar por entradas del usuario (user input).
- Usamos el **ciclo while** cuando el incremento de valor en iteración es algún valor no estándar.

También es importante mencionar que conforme adquiramos más habilidades podríamos usar estructuras de iteración más avanzadas diferentes a for y while.