

Introducción a los Ciclos

Víctor Armando Jaramillo Moreno

22 Febrero 2025

1 ¿Qué son los Bucles-Ciclos?

Los bucles o tambien llamados ciclos permiten ejecutar un bloque de código varias veces sin necesidad de escribirlo repetidamente. Son muy útiles cuando queremos repetir una acción hasta que se cumpla una condición.

Existen tres tipos principales de bucles en C++:

- **while**: Ejecuta un bloque de código mientras se cumpla una condición.
- **do-while**: Ejecuta el código al menos una vez y luego sigue mientras la condición se cumpla.
- **for**: Permite repetir un bloque de código un número específico de veces.

2 El Bucle while

El bucle **while** sigue ejecutándose mientras una condición sea verdadera. Su estructura es:

```
while (condición) {  
    // Bloque de código a repetir  
}
```

Ejemplo:

```
int x = 1;  
while (x <= 5) {  
    cout << "Número: " << x << endl;  
    x++; // Incrementa x para evitar un bucle infinito  
}
```

2.1 Cuidado con los Bucles Infinitos

Si olvidamos actualizar la variable de control, el bucle puede repetirse para siempre, causando un problema conocido como bucle infinito:

```
int x = 1;
while (x > 0) {
    cout << "Este es un bucle infinito" << endl;
}
```

3 El Bucle do-while

Este tipo de bucle es similar al **while**, pero siempre ejecuta el código al menos una vez, porque la condición se verifica después de la primera ejecución.

```
do {
    // Bloque de código a repetir
} while (condición);
```

Ejemplo:

```
int x = 1;
do {
    cout << "Número: " << x << endl;
    x++;
} while (x <= 5);
```

4 El Bucle for

El bucle **for** es ideal cuando sabemos cuántas veces queremos ejecutar el código. Su estructura es:

```
for (inicio; condición; actualización) {
    // Bloque de código a repetir
}
```

4.1 Explicación de sus componentes

- **Inicio:** Se ejecuta una sola vez antes de que el bucle comience. Generalmente se usa para inicializar una variable de control.
- **Condición:** Se evalúa antes de cada iteración. Si es verdadera, el bucle sigue ejecutándose; si es falsa, el bucle termina.
- **Actualización:** Se ejecuta al final de cada iteración y normalmente se usa para modificar la variable de control.

Ejemplo:

```
for (int i = 1; i <= 5; i++) {
    cout << "Número: " << i << endl;
}
```

4.2 Ejemplos adicionales

Ejemplo 1: Imprimir los primeros 10 números pares

```
for (int i = 2; i <= 20; i += 2) {  
    cout << i << " ";  
}
```

Ejemplo 2: Contar hacia atrás desde 10 hasta 1

```
for (int i = 10; i >= 1; i--) {  
    cout << i << " ";  
}
```

Ejemplo 3: Imprimir la tabla de multiplicar del 5

```
int numero = 5;  
for (int i = 1; i <= 10; i++) {  
    cout << numero << " x " << i << " = " << numero * i << endl;  
}
```

5 Anidación de Bucles

Podemos usar bucles dentro de otros bucles para tareas más complejas, como contar hacia atrás dentro de un ciclo:

```
for (int i = 1; i <= 3; i++) {  
    for (int j = 10; j > 0; j--) {  
        cout << j << " ";  
    }  
    cout << "Lanzamiento " << i << " completado." << endl;  
}
```

Ejemplo de salida si el usuario ingresa 3:

```
10 9 8 7 6 5 4 3 2 1 Lanzamiento 1 completado.  
10 9 8 7 6 5 4 3 2 1 Lanzamiento 2 completado.  
10 9 8 7 6 5 4 3 2 1 Lanzamiento 3 completado.
```

6 Equivalencia de Bucles

Podemos transformar un `for` en un `while` si lo reescribimos de esta manera:

```
int i = 0;  
while (i < 10) {  
    cout << i+1 << endl;  
    i++;  
}
```

```
for (int i = 1; i <= 10; i++) {  
    cout << "Número: " << i << endl;  
}
```

7 Conclusión

Los bucles son fundamentales en la programación porque permiten ejecutar tareas repetitivas de manera eficiente. Elige **while** cuando no sepas cuántas veces se ejecutará el código, **do-while** cuando necesites que el código se ejecute al menos una vez, y **for** cuando conozcas el número exacto de repeticiones.