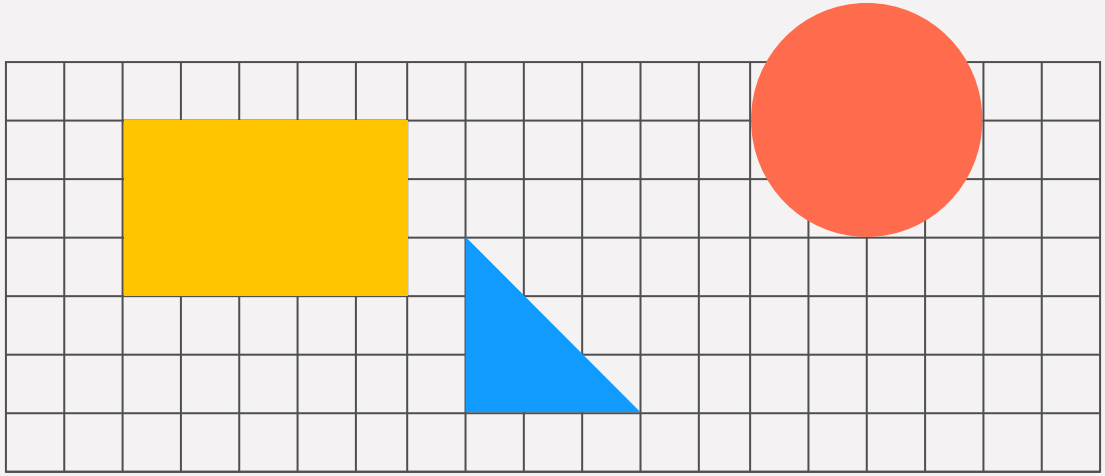
Unidad 3
La Estructura de Iteración



Sentencia for

Sentencia for

La sentencia `for` se usa para ejecutar un bloque de código **un número determinado de veces**. A diferencia de `while`, donde el control de la condición es más manual, `for` tiene una estructura más organizada con **tres partes**:

1. **Inicialización** → Se ejecuta una sola vez antes del primer ciclo.
2. **Condición** → Se evalúa antes de cada iteración; si es `true`, el ciclo continúa.
3. **Actualización** → Se ejecuta después de cada iteración para modificar la variable de control.

Sintaxis de for

```
for (inicialización; condición; actualización)
{
    // Código que se ejecuta en cada iteración mientras la condición sea verdadera
}
```

▶ Contador del 1 al 5

```c
int main()
{
    for (int i = 1; i ≤ 5; i++)
    { // Se inicializa i en 1, el ciclo sigue mientras i ≤ 5, se incrementa i en cada iteración
        printf("Número: %d\n", i);
    }

    return 0;
}
```

- **Inicialización:** `int i = 1;` → Se declara e inicializa `i` en 1.
- **Condición:** `i <= 5;` → Mientras `i` sea menor o igual a 5, el ciclo continúa.
- **Actualización:** `i++` → Se incrementa `i` en 1 después de cada iteración.

Errores comunes con for

No modificar la variable de control

Error - Bucle infinito

```
for (int i = 1; i <= 5; ) { // Falta el incremento
    printf("%d\n", i);
}
```

Errores comunes con for

Condición incorrecta

Error - No entra al ciclo

```
 for (int i = 10; i < 5; i++) { // i empieza en 10, pero nunca es menor
que 5
    printf("Esto nunca se imprimirá.\n");
}
```

Sentencia while

Sentencia while

La sentencia while permite repetir un bloque de código mientras una condición sea verdadera. Se usa cuando no sabemos exactamente cuántas veces se repetirá el ciclo, ya que depende de la evaluación de la condición.

Sintaxis de while

```
while (condición) {
    // Código que se ejecuta mientras la condición sea verdadera
}
```

Contador del 1 al 5

```c
int main()
{
  int contador = 1; // Inicialización de la variable

  while (contador ≤ 5)
  { // Condición: Mientras contador sea menor o igual a 5
    printf("Número: %d\n", contador);
    contador++; // Incrementa el contador en 1
  }

  return 0;
}
```

- **Se inicializa `contador` en 1**.
- **El `while` evalúa si `contador <= 5`**. Si es cierto, ejecuta el bloque dentro del ciclo.
- **Se imprime el valor de `contador`**.
- **Se incrementa `contador` en 1** (`contador++`).
- **Se vuelve a evaluar la condición**. Si sigue siendo verdadera, repite el ciclo.
- Cuando `contador` llega a 6, **la condición es falsa y el `while` termina**.

Errores comunes al usar while

Ciclo infinito

Si la variable dentro del while no cambia dentro del ciclo, la condición nunca será falsa y el programa quedará atrapado en un bucle infinito.

```
 int x = 1;
while (x <= 5) {
    printf("Número: %d\n", x);
    // Falta incrementar x, el ciclo será infinito
}
```

Errores comunes al usar while

Condición incorrecta

Asegurarse de que la condición permita al while ejecutarse al menos una vez si es necesario.

```
 int x = 10;
while (x < 5) { // x nunca será menor que 5
    printf("Esto nunca se imprimirá\n");
}
```

Sentencia do-while

Sentencia do-while

La sentencia `do-while` es una estructura de control de flujo en C que **ejecuta un bloque de código al menos una vez** y luego sigue ejecutándolo **mientras una condición sea verdadera**.

A diferencia del `while`, donde primero se evalúa la condición antes de ejecutar el código, en `do-while` el código **se ejecuta al menos una vez** antes de comprobar la condición.

Sintaxis de do-while

```
do {
    // Código a ejecutar al menos una vez
} while (condición);
```

```c
int main()
{
  int numero;

  do
  {
    printf("Ingrese un número mayor que 10: ");
    scanf("%d", &numero);
  } while (numero ≤ 10); // Si el número es 10 o menor, vuelve a pedirlo

  printf("Número aceptado: %d\n", numero);

  return 0;
}
```

- Se declara la variable `numero`.
- Dentro del `do`, se muestra un mensaje y se pide al usuario un número.
- **Si el número es menor o igual a 10, se repite el proceso**.
- **Si el número es mayor a 10, el bucle termina** y se muestra el mensaje final.

Diferencia entre while y do-while

Característica	while	do-while
¿Cuándo se evalúa la condición?	Antes de entrar al bucle	Después de ejecutar el bloque
¿Se ejecuta al menos una vez?	No, si la condición es `false` desde el inicio, nunca entra al ciclo	Sí, siempre se ejecuta al menos una vez
Uso recomendado	Cuando puede ser que el bloque **nunca** se ejecute	Cuando se necesita que el bloque **se ejecute al menos una vez**

Ejemplo comparativo

```c
// Usando while
int x = 5;
while (x > 10) {
    printf("Esto no se imprimirá\n");
}
```

```c
// Usando do-while
int y = 5;
do {
    printf("Esto se imprimirá al menos una vez\n");
} while (y > 10);
```

Errores comunes con do-while

Olvidar la actualización de la variable de control

```c
int i = 1;
do {
    printf("%d\n", i);
} while (i <= 5); // ¡Ciclo infinito porque i nunca cambia!
```