Loops

Universidad Católica Boliviana

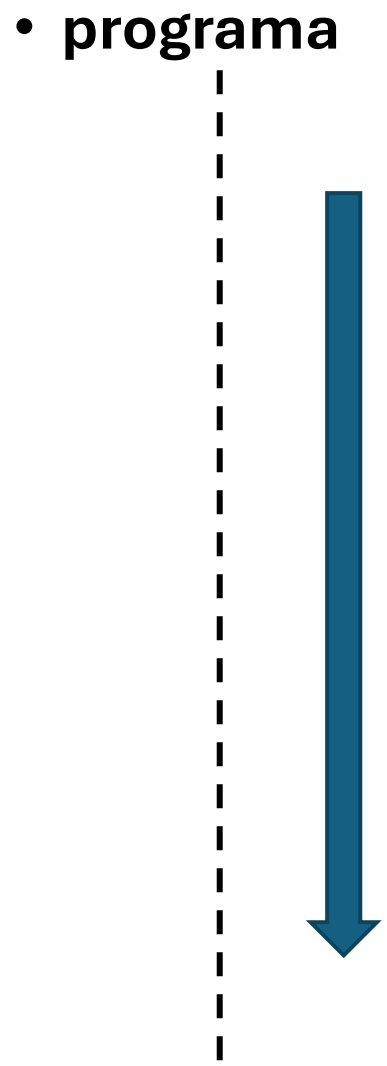
MSc, José Jesús Cabrera Pantoja

Outline

- While loop
- Do-while
- For loop
- Control del loop

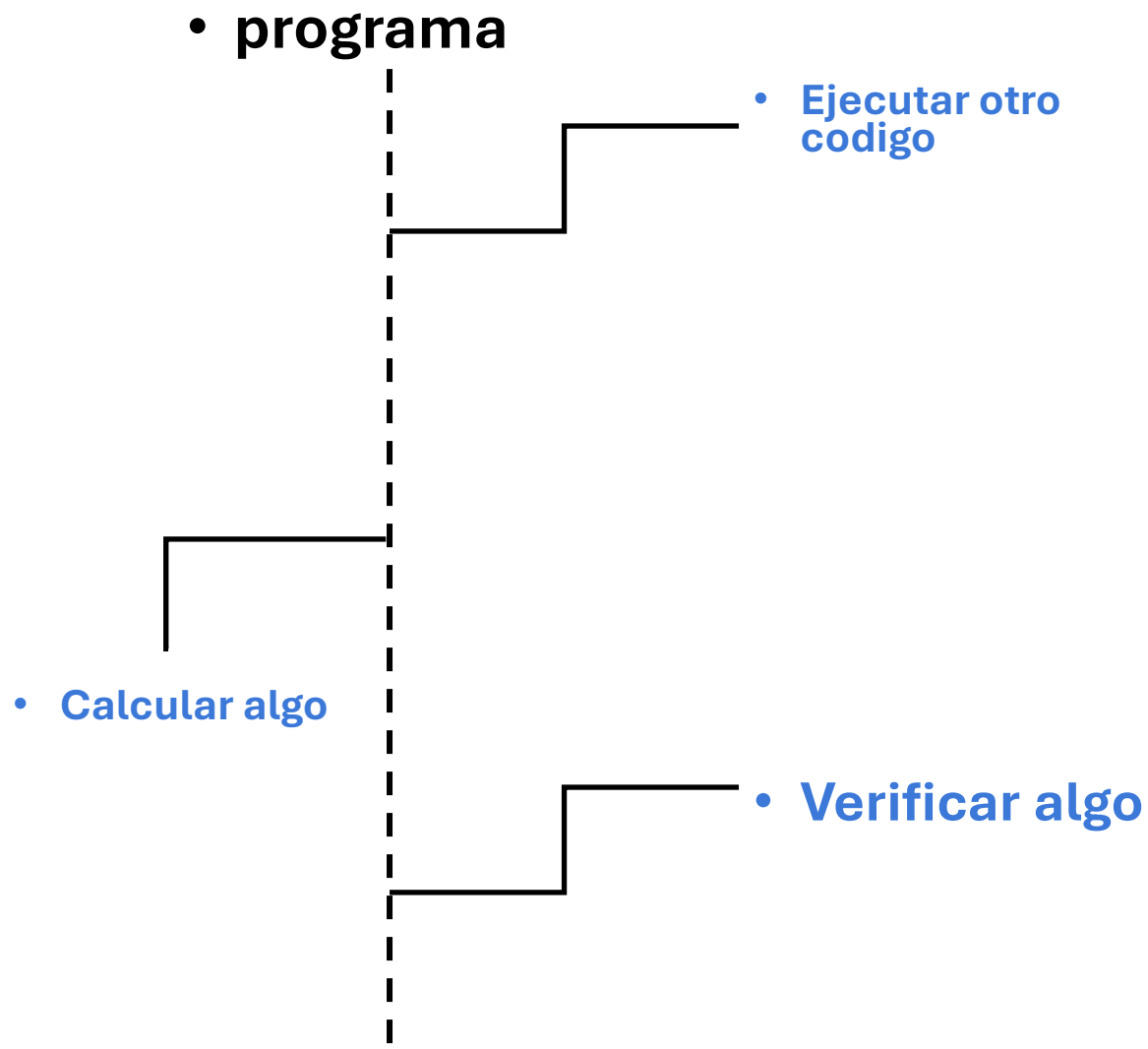
Control de flujo

- Comienza por el principio, dijo el rey, muy gravemente, y continúa hasta que llegues al final: Entonces detente. **“Lewis Carroll, El maravilloso mundo de Alicia”**



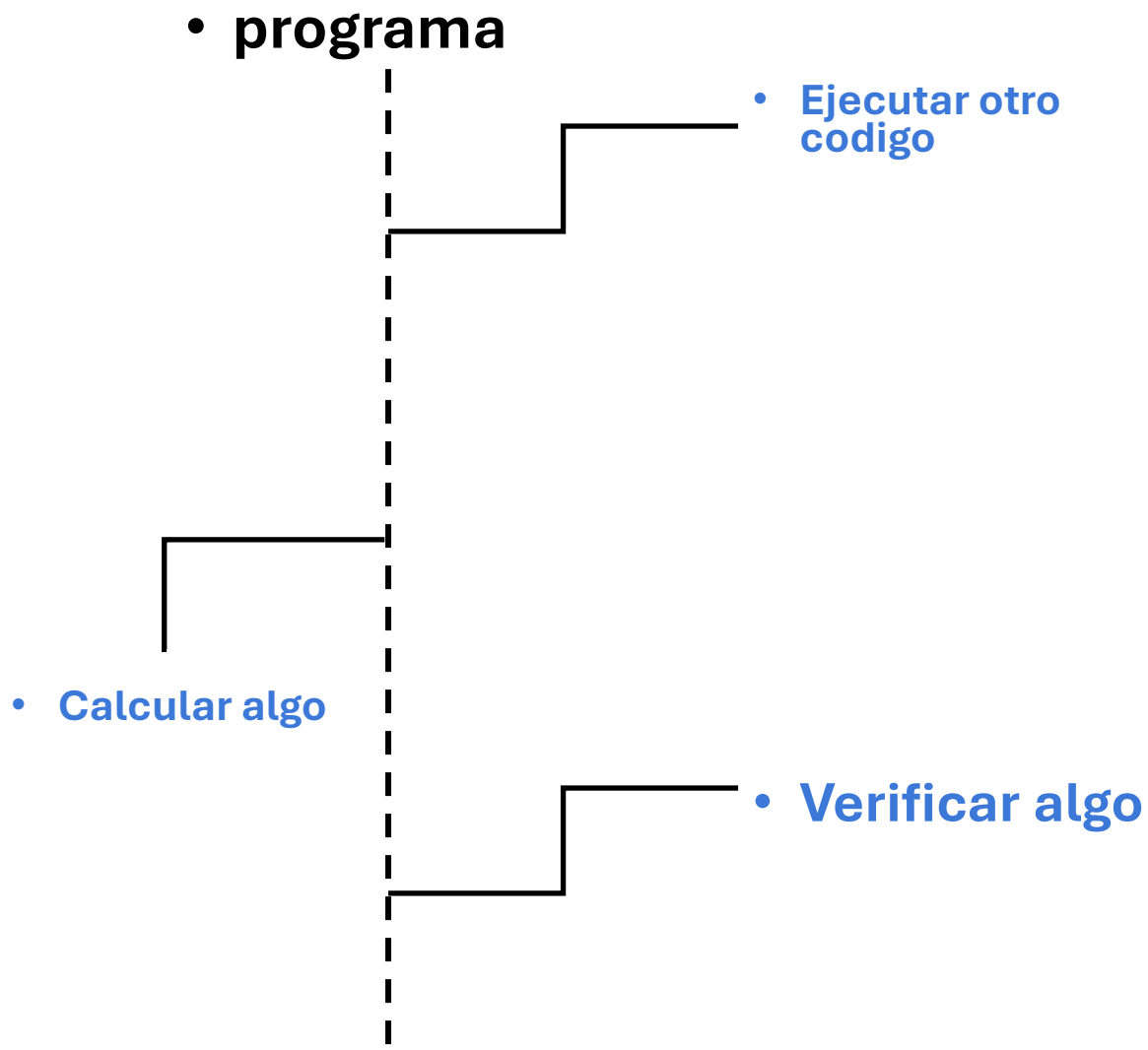
Control de flujo

- Aquí es donde entran en juego las declaraciones condicionales como **IF**, **IF-ELSE** y **IF-ELSE-IF**.



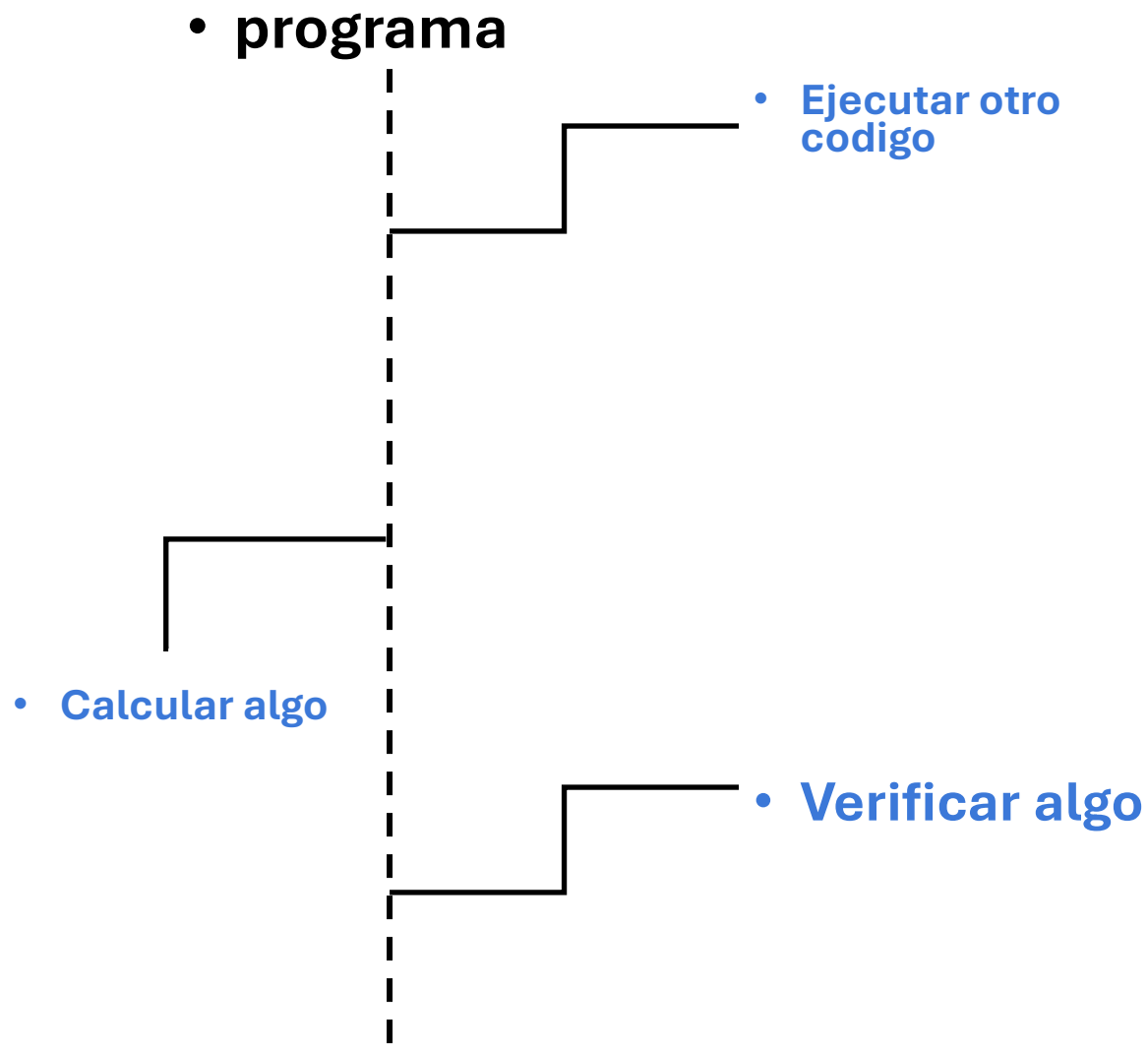
Control de flujo

- Otra forma es el operador:
- **SWITCH**



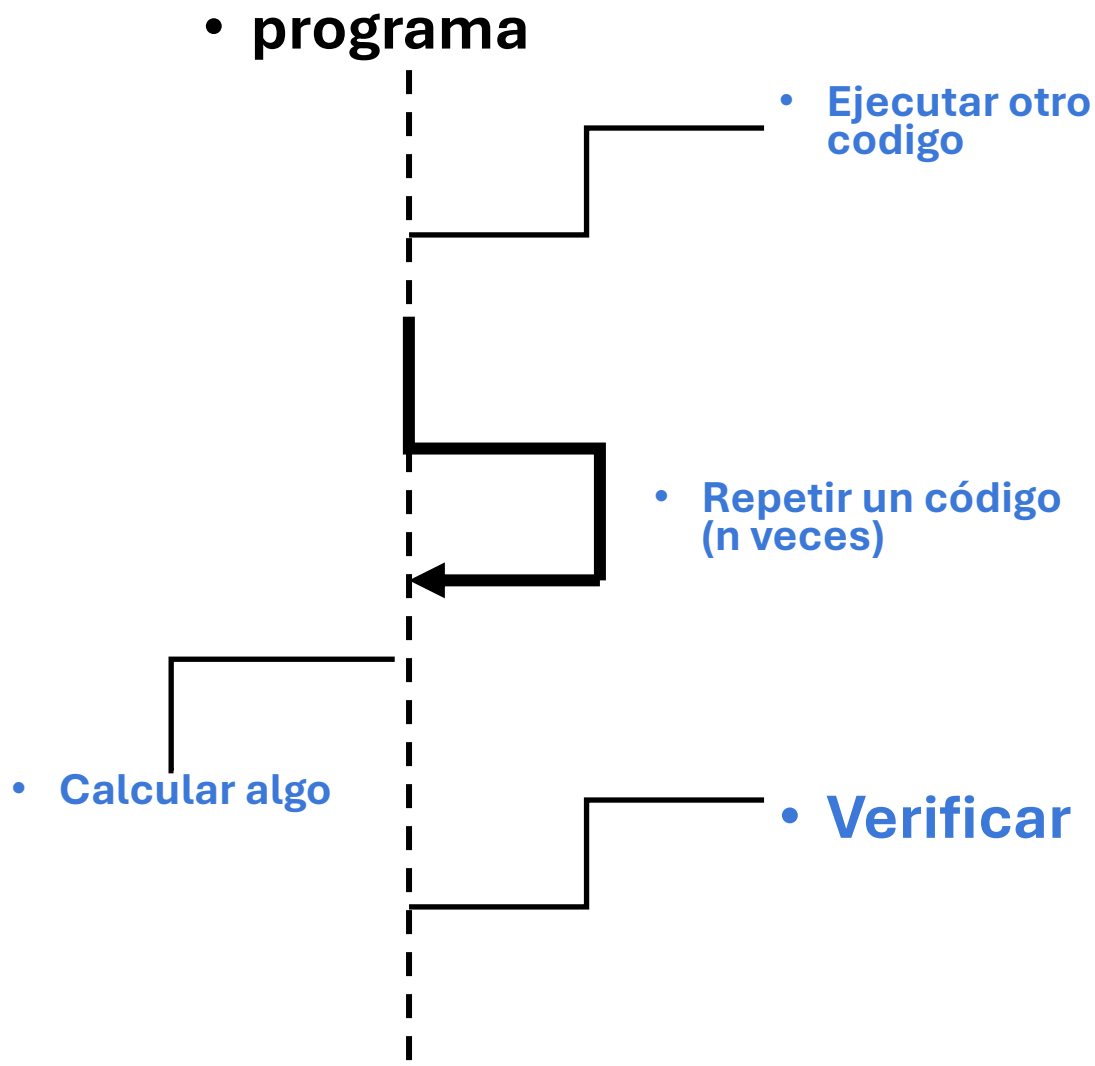
Control de flujo

- En la vida real, a menudo realizamos tareas específicas repetidamente hasta que logramos un objetivo en particular.



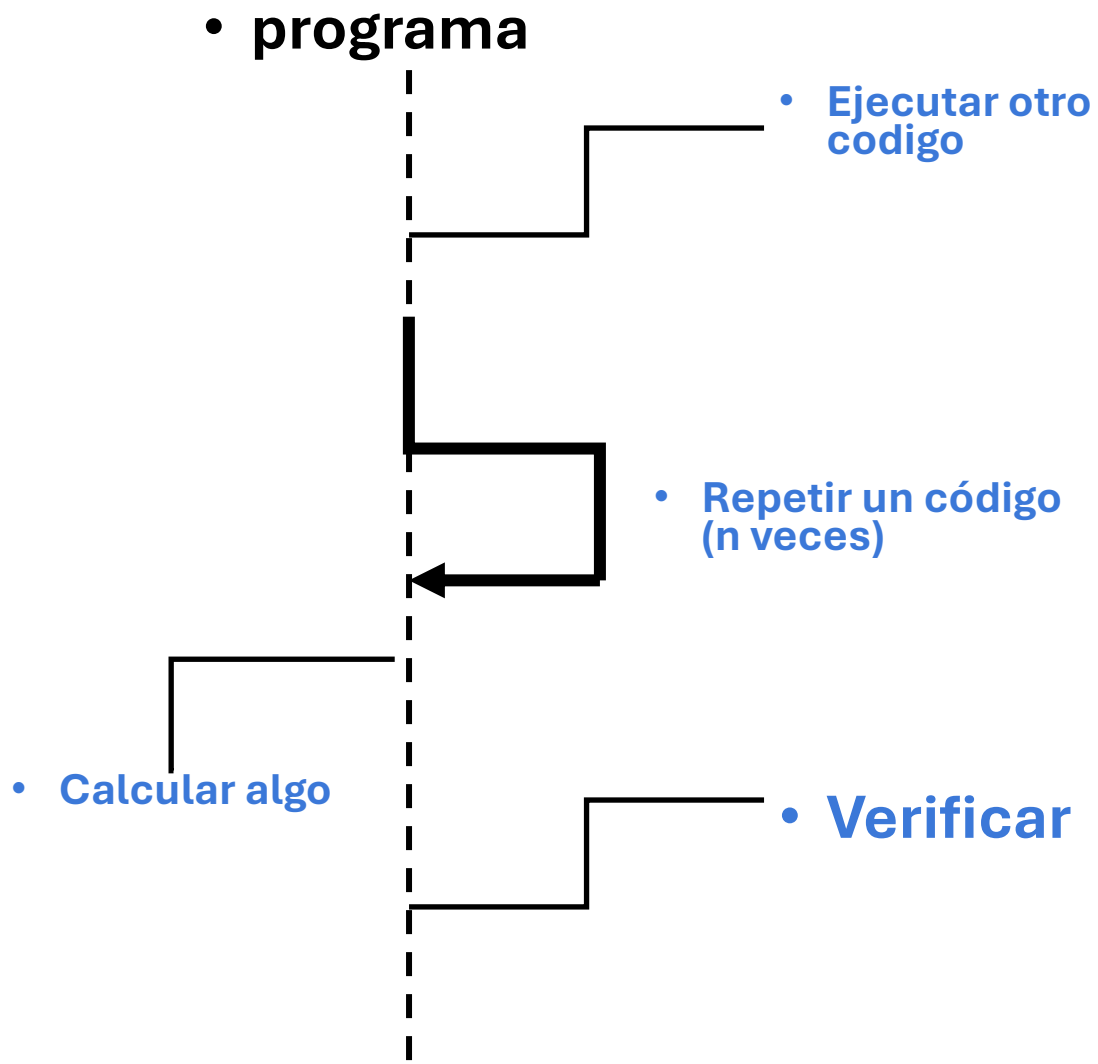
Loop

- En los programas de computadora, una secuencia de instrucciones que se ejecuta hasta que una condición se vuelve falsa se llama **bucle (loop)**.

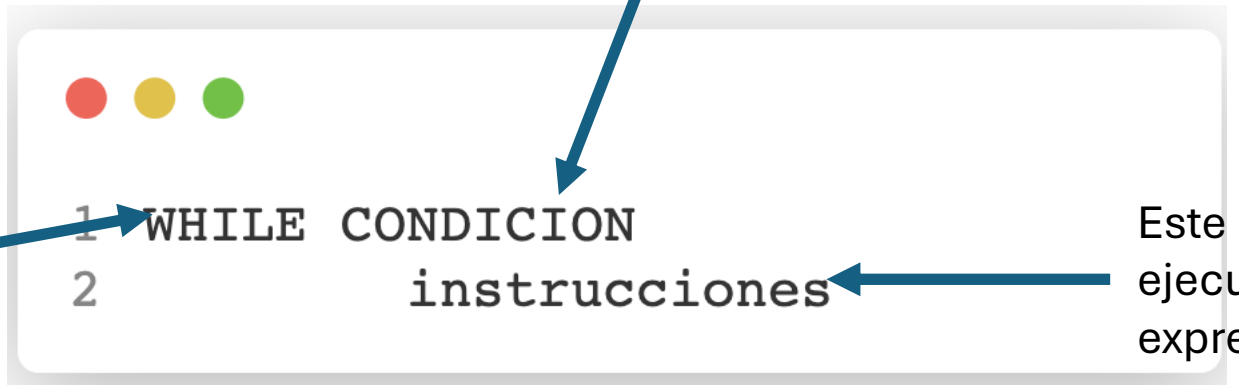


Loops

- While loop
- For loop



Loop: while



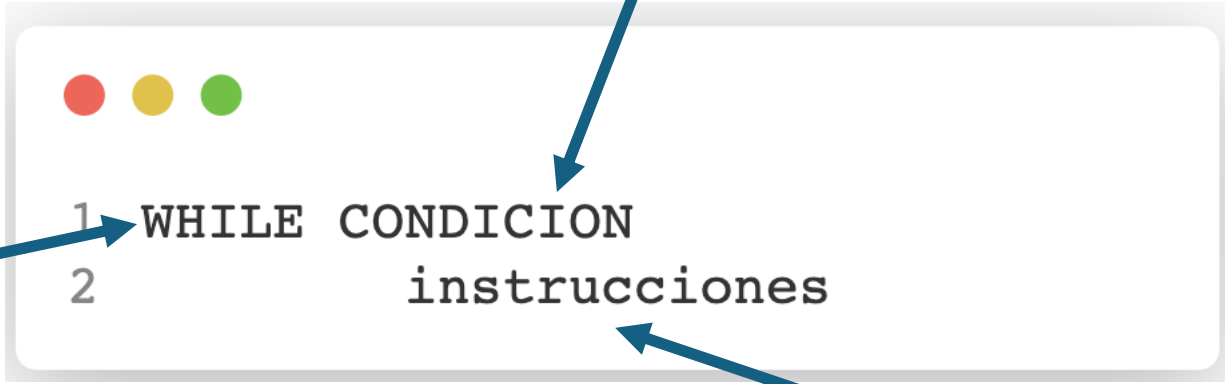
Una expresión que puede ser verdadera o false

Palabra Clave

Este bloque de código se ejecutara únicamente si la expresión es verdadera

Loop: while

Una expresión que puede ser verdadera o false



Palabra Clave

Este bloque de código se ejecutara si la expresión es verdadera. Despues de ejecutar estas instrucciones se volvera a verificar la condición. Una vez la condición sea falsa no volverá a ejecutar las instrucciones

Loop: Ventajas

- no tiene que escribir el mismo código una y otra vez, lo que garantiza que el código sea compacto y legible.
- Por ejemplo, suponga que desea generar números enteros de cero a diez. El siguiente es un método para lograr esto (sin loops):

Loop: Implementacion

PseudoCodigo

```
OUTPUT 0
OUTPUT 1
OUTPUT 2
OUTPUT 3
OUTPUT 4
OUTPUT 5
OUTPUT 6
OUTPUT 7
OUTPUT 8
OUTPUT 9
OUTPUT 10
```

C++

```cpp
int main() {
  cout << 0 << endl;
  cout << 1 << endl;
  cout << 2 << endl;
  cout << 3 << endl;
  cout << 4 << endl;
  cout << 5 << endl;
  cout << 6 << endl;
  cout << 7 << endl;
  cout << 8 << endl;
  cout << 9 << endl;
  cout << 10 << endl;

  return 0;
}
```

Loop: Implementacion

PseudoCodigo

```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

Loop: Implementacion

PseudoCodigo

```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

- En la primera línea, establecemos i en 1. i contiene el número que mostramos.
- Luego, verificamos si i es menor o igual a 10.
- Si lo es, mostramos el valor de i
- Lo incrementamos en 1.
- Repetimos estos pasos hasta que i sea mayor que 10.

Loop: Implementacion

PseudoCodigo

```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

- En la primera línea, establecemos i en 1. i contiene el número que mostramos.
- Luego, verificamos si i es menor o igual a 10.
- Si lo es, mostramos el valor de i
- Lo incrementamos en 1.
- Repetimos estos pasos hasta que i sea mayor que 10.

Loop: Implementacion

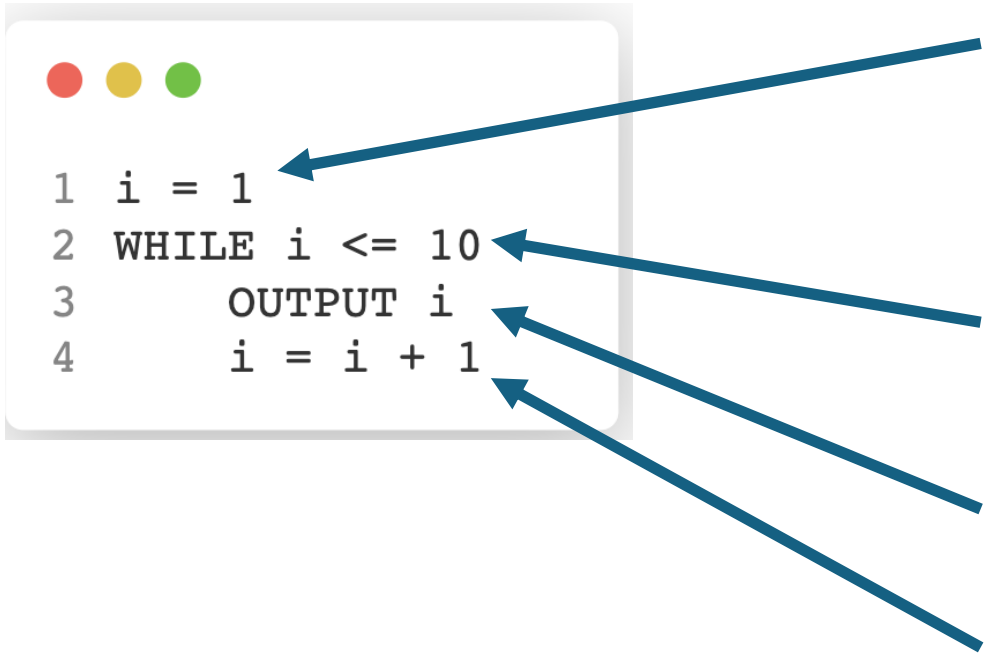
PseudoCodigo

```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

- En la primera línea, establecemos i en 1. i contiene el número que mostramos.
- Luego, verificamos si i es menor o igual a 10.
- Si lo es, mostramos el valor de i
- Lo incrementamos en 1.
- Repetimos estos pasos hasta que i sea mayor que 10.

Loop: Implementacion

PseudoCodigo



```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

- En la primera línea, establecemos i en 1. i contiene el número que mostramos.
- Luego, verificamos si i es menor o igual a 10.
- Si lo es, mostramos el valor de i
- Lo incrementamos en 1.
- Repetimos estos pasos hasta que i sea mayor que 10.

Loop: Implementacion

PseudoCodigo

```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

```
1 while (haya helado en la heladera) {
2     ir y volver a la heladera por helado;
3 }
```

Loop: Implementacion

PseudoCodigo

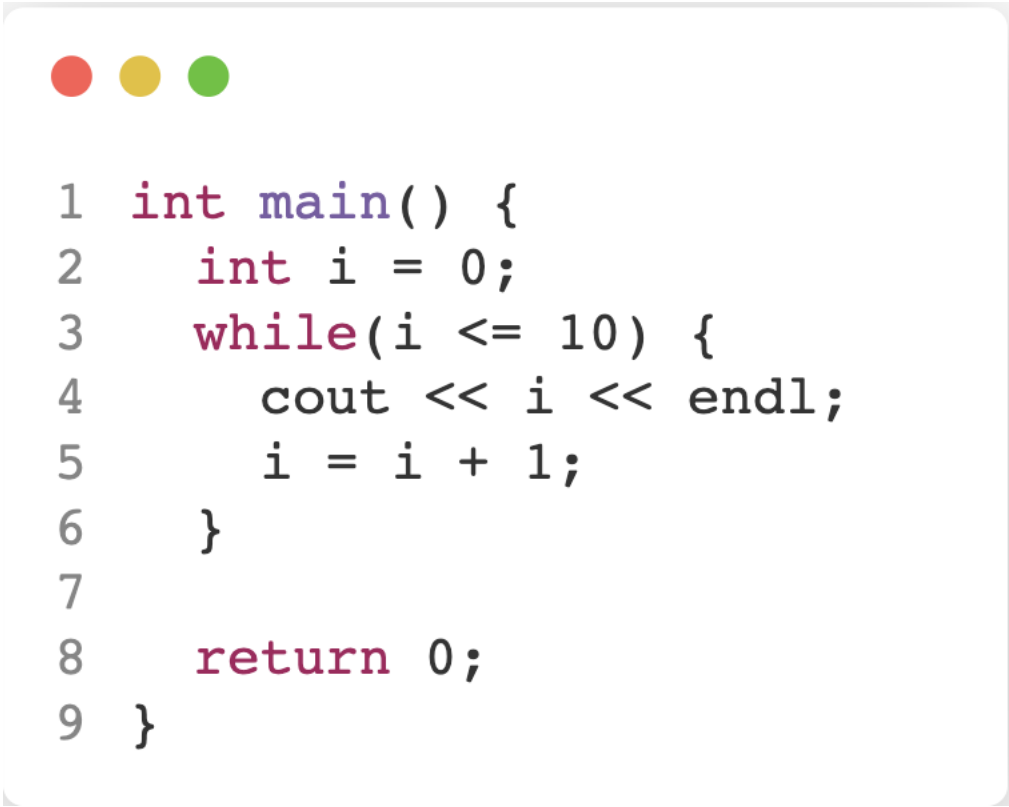
```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

C++

```
1 int main() {
2   int i = 0;
3   while(i <= 10) {
4     cout << i << endl;
5     i = i + 1;
6   }
7
8   return 0;
9 }
```

Loop: Implementacion

C++

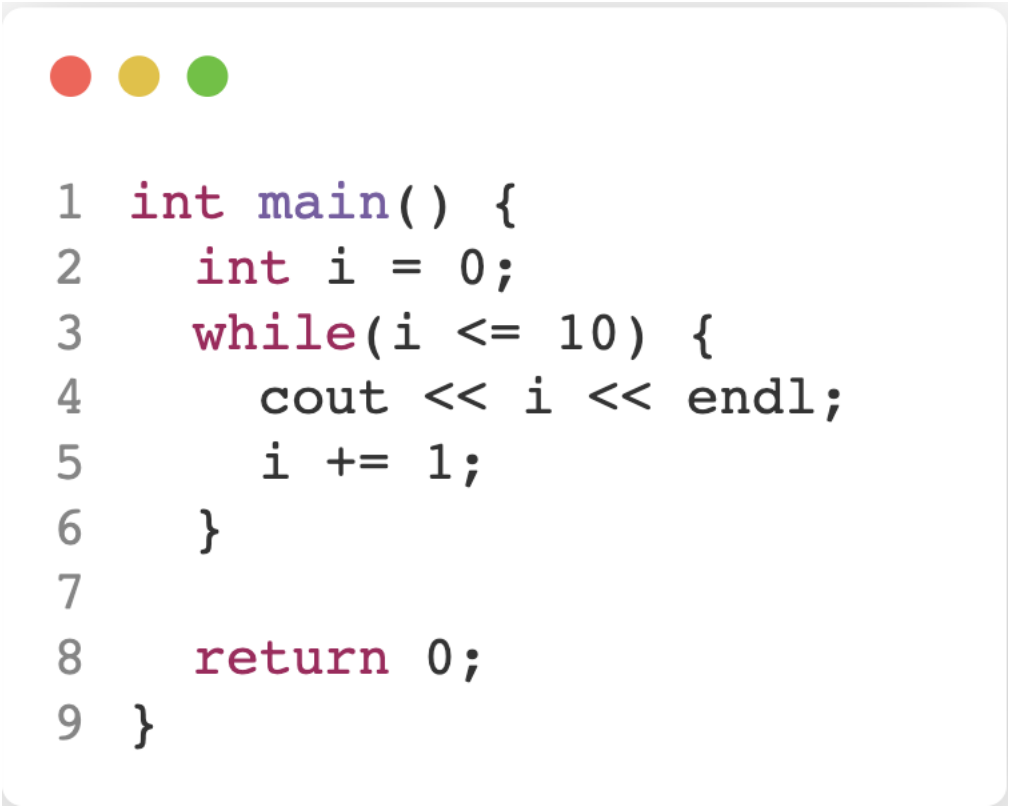


```cpp
int main() {
  int i = 0;
  while(i <= 10) {
    cout << i << endl;
    i = i + 1;
  }

  return 0;
}
```

Loop: Implementacion

C++

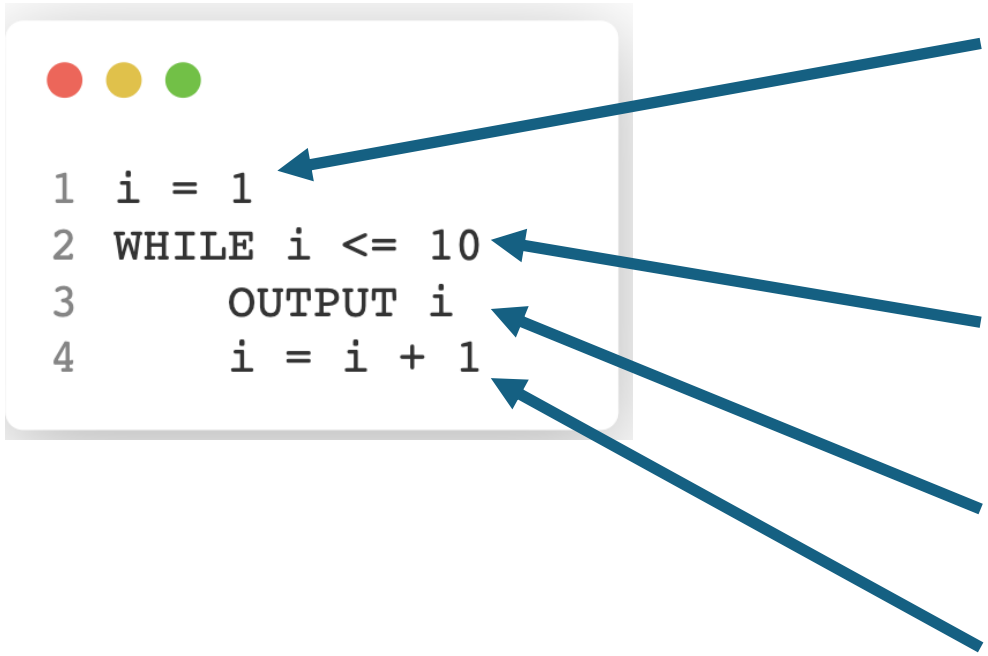


```cpp
int main() {
  int i = 0;
  while(i <= 10) {
    cout << i << endl;
    i += 1;
  }

  return 0;
}
```

Loop: Implementacion

PseudoCodigo



```
1 i = 1
2 WHILE i <= 10
3     OUTPUT i
4     i = i + 1
```

- En la primera línea, establecemos i en 1. i contiene el número que mostramos.
- Luego, verificamos si i es menor o igual a 10.
- Si lo es, mostramos el valor de i
- Lo incrementamos en 1.
- Repetimos estos pasos hasta que i sea mayor que 10.

Loop: For

```
1 cantidad_de_pasos = 0;
2 while (cantidad_de_pasos != 3) {
3     ir a la heladera;
4     cantidad_de_pasos = cantidad_de_pasos + 1;
5 }
```

Loop: For

```
1 cantidad_de_pasos = 0;
2 while (cantidad_de_pasos != 3) {
3     ir a la heladera;
4     cantidad_de_pasos = cantidad_de_pasos + 1;
5 }
```

```
1 for (<accion inicial>; <condicion>; <paso del ciclo>) {
2     <accion, la cual es necesaria repetir>
3 }
```

Loop: For

```
cantidad_de_pasos = 0;
while (cantidad_de_pasos != 3) {
    ir a la heladera;
    cantidad_de_pasos = cantidad_de_pasos + 1;
}
```

```
for (cantidad_de_pasos = 0; cantidad_de_pasos != 3; cantidad_de_pasos += 1) {
    ir a la heladera;
}
```

Loop: For

```
1 for (cantidad_de_pasos = 0; cantidad_de_pasos != 3; cantidad_de_pasos += 1) {
2     ir a la heladera;
3 }
```

- El orden de funcionamiento de este ciclo será el siguiente:

1. Se realizarán <acciones iniciales>: la cantidad de pasos será 0.
2. Se comprobará <condición de ejecución>: 0 no es igual a 3? La respuesta es positiva, la condición se cumple.
3. Vamos a la heladera.
4. Hacemos un <paso de ciclo>: el cantidad de pasos += 1. En otras palabras, ahora el número de acercamientos es 1.
5. Vuelva al paso 2 y repita los pasos 2, 3 y 4 hasta que la <condición de ejecución> no sea verdadera.

Loop: concisión

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

  return 0;
}
```

- A menudo, en la escritura de pasos de ciclo se utilizan versiones cortas para **i += 1**. Hay dos opciones de este tipo:
  - **++i** - incremento de **prefijo**.
  - **i++** - incremento de **postfijo**.

Loop: concisión

- A menudo, en la escritura de pasos de ciclo se utilizan versiones cortas para **i += 1**. Hay dos opciones de este tipo:
  - **++i** - incremento de **prefijo**.
  - **i++** - incremento de **postfijo**.

```
1 int main() {
2
3   for (int i = 0; i != 3; i++) {
4       cout << "Check the fridge"s << endl;
5   }
6
7   return 0;
8 }
```

Loop: concisión diferencias

- **El primer caso es posfijo.**
  - El programa creará una copia de la variable i. La copia será 0.
  - El programa aumentará i en 1.
  - El programa mostrará una copia que es igual al valor original de i.
  - La copia ya no es necesaria, puede deshacerse de ella.

```
1  int main() {
2
3    // postfijo
4    int i = 0;
5    cout << i++ << endl;
6
7    // prefijo
8    i = 0;
9    cout << ++i << endl;
10
11   return 0;
12 }
```

Loop: concisión diferencias

- **El segundo caso el prefijo:**
  - El programa aumentará i en 1.
  - El programa imprimirá i en la pantalla.

```
1  int main() {
2
3    // postfijo
4    int i = 0;
5    cout << i++ << endl;
6
7    // prefijo
8    i = 0;
9    cout << ++i << endl;
10
11   return 0;
12 }
```

Loop: Control

- **Para controlar el ciclo (loop) se utiliza dos palabras clave**
  - continue
  - break