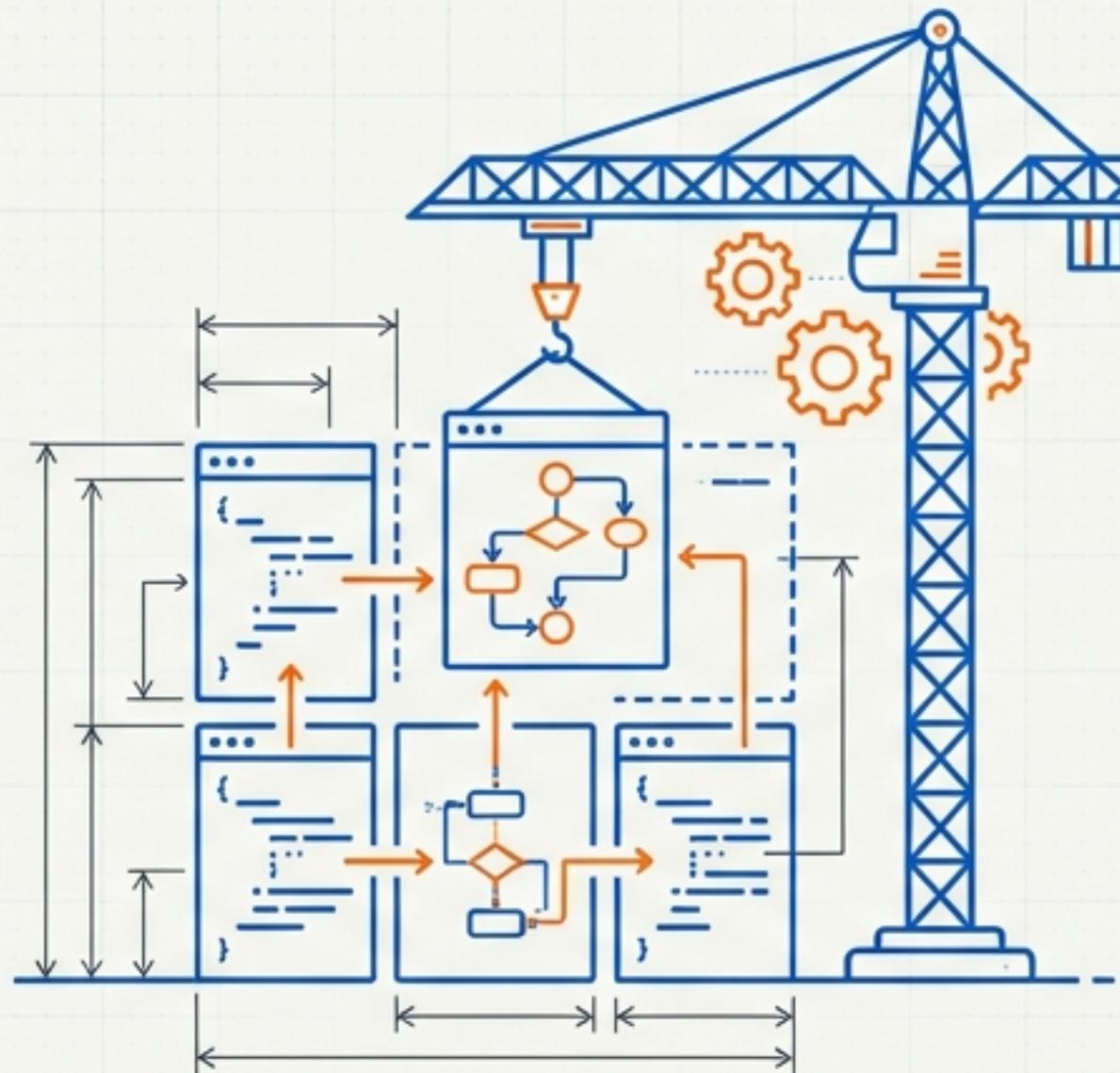


Project Info
MÓDULO: PROGRAMACIÓN
TEMA: 03



Domina el Flujo: Fundamentos de la Programación Estructurada

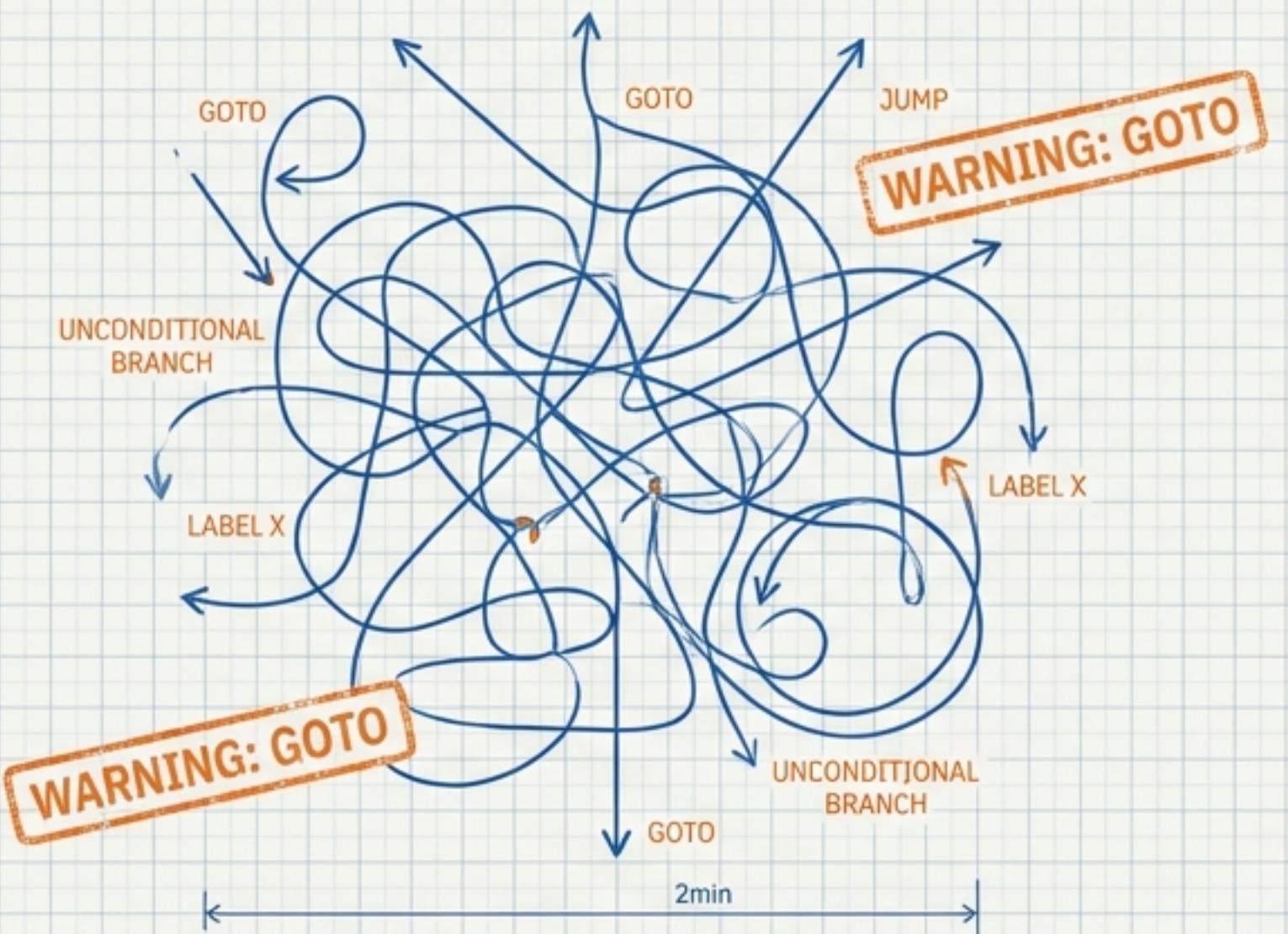
Construyendo algoritmos robustos con Lógica y Control

INTRODUCCIÓN AL TEMA

El desarrollo de software no es solo escribir líneas de código; es diseñar estructuras lógicas. Este módulo transforma la teoría en herramientas prácticas para controlar el flujo de información.

Del Caos al Orden: El Teorema de Estructura

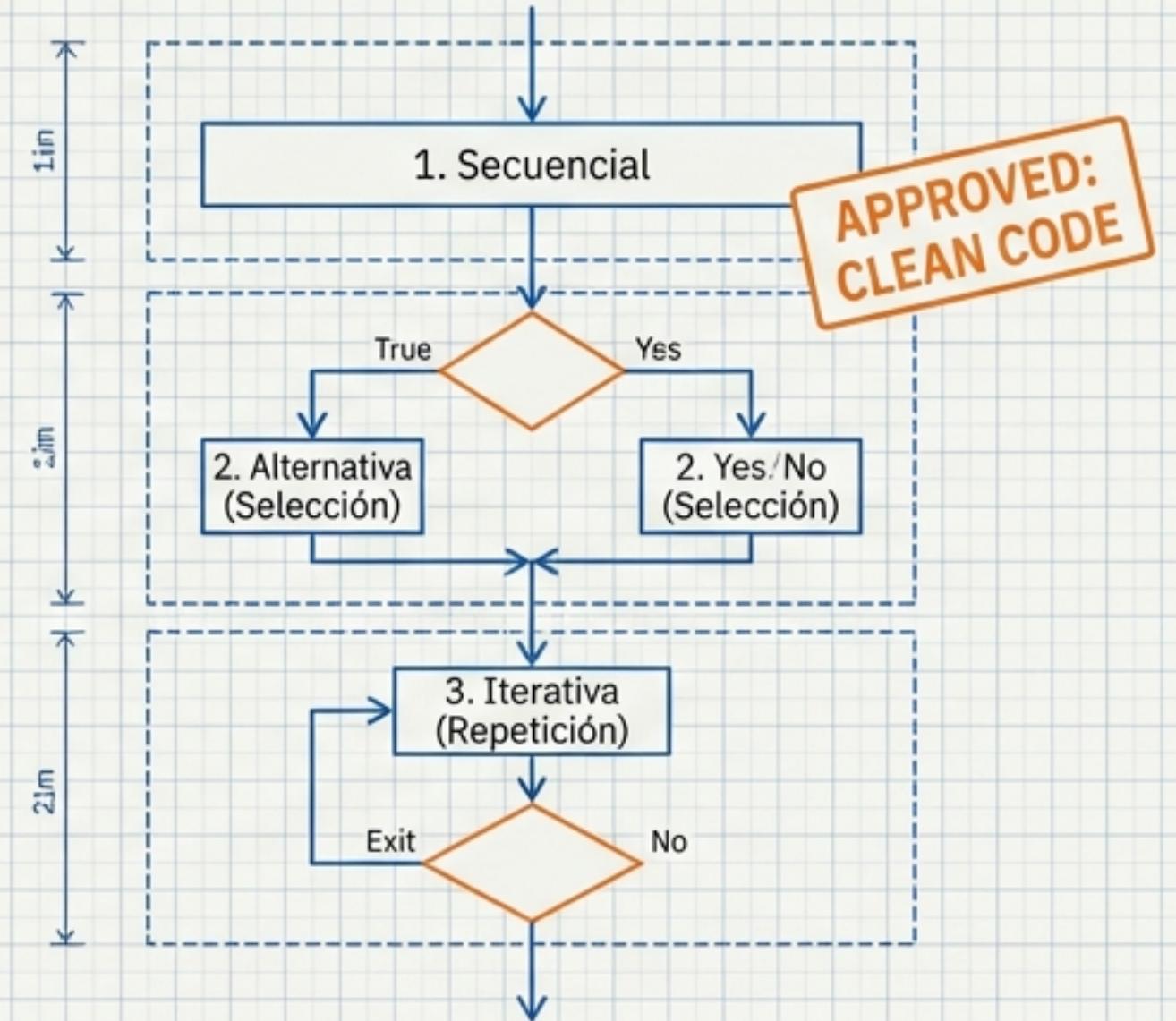
Spaghetti Code (< 1960)



Antes de los años 60, la programación era un caos de instrucciones sin orden. Un programa estructurado se expresa únicamente mediante la combinación de tres estructuras básicas:

“Un programa bien estructurado es legible, fácil de depurar y modificable por partes sin romper el todo.”

Programación Estructurada (Dijkstra)

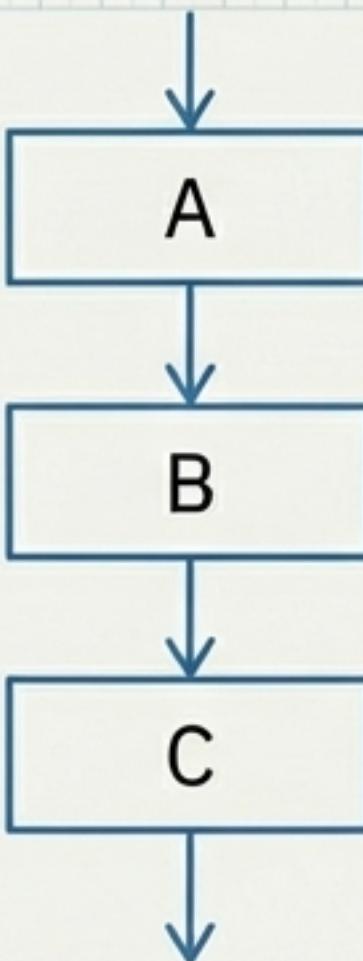


1. Secuencial
2. Alternativa (Selección)
3. Iterativa (Repetición)

PROJECT:	STRUCTURED PROGRAMMING FUNDAMENTALS	
DATE:	OCT 26, 2023	SEALE: -----
ENGINEER:	DJKSTRA	DATE: 2023 SHEET NO: 1 of 1

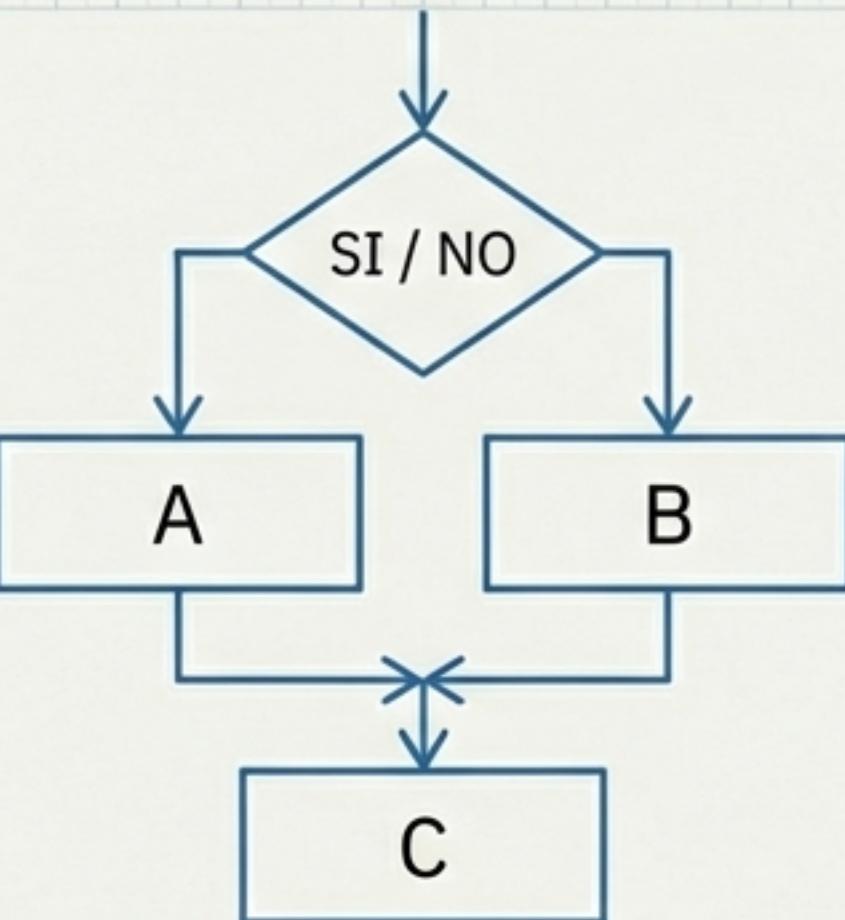
Los Tres Pilares del Control de Flujo

SECUENCIA



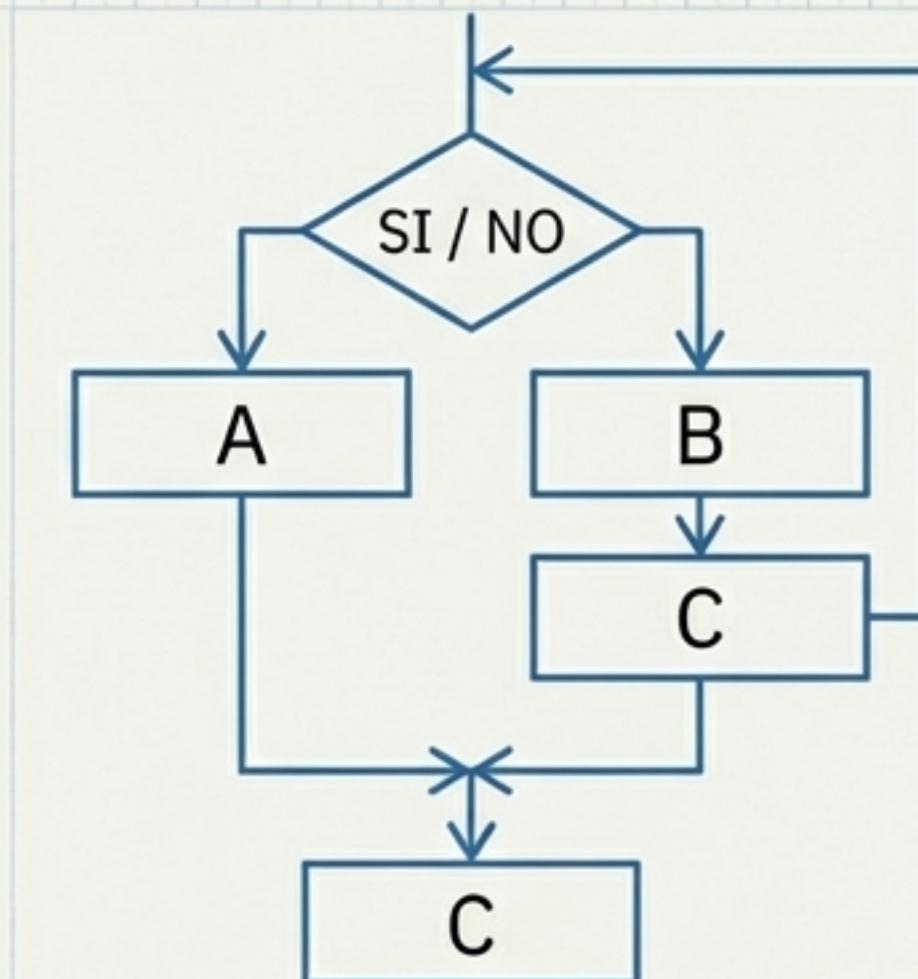
Ejecución lineal. Una instrucción sigue a la otra en orden descendente.

SELECCIÓN



Toma de decisiones. El flujo se divide en función de si una condición es verdadera o falsa (Sí/No).

ITERACIÓN



Repetición. Un bloque de instrucciones se ejecuta múltiples veces mientras se cumpla una condición.

La Bifurcación: Selección Simple (If)

Rompiendo el orden secuencial

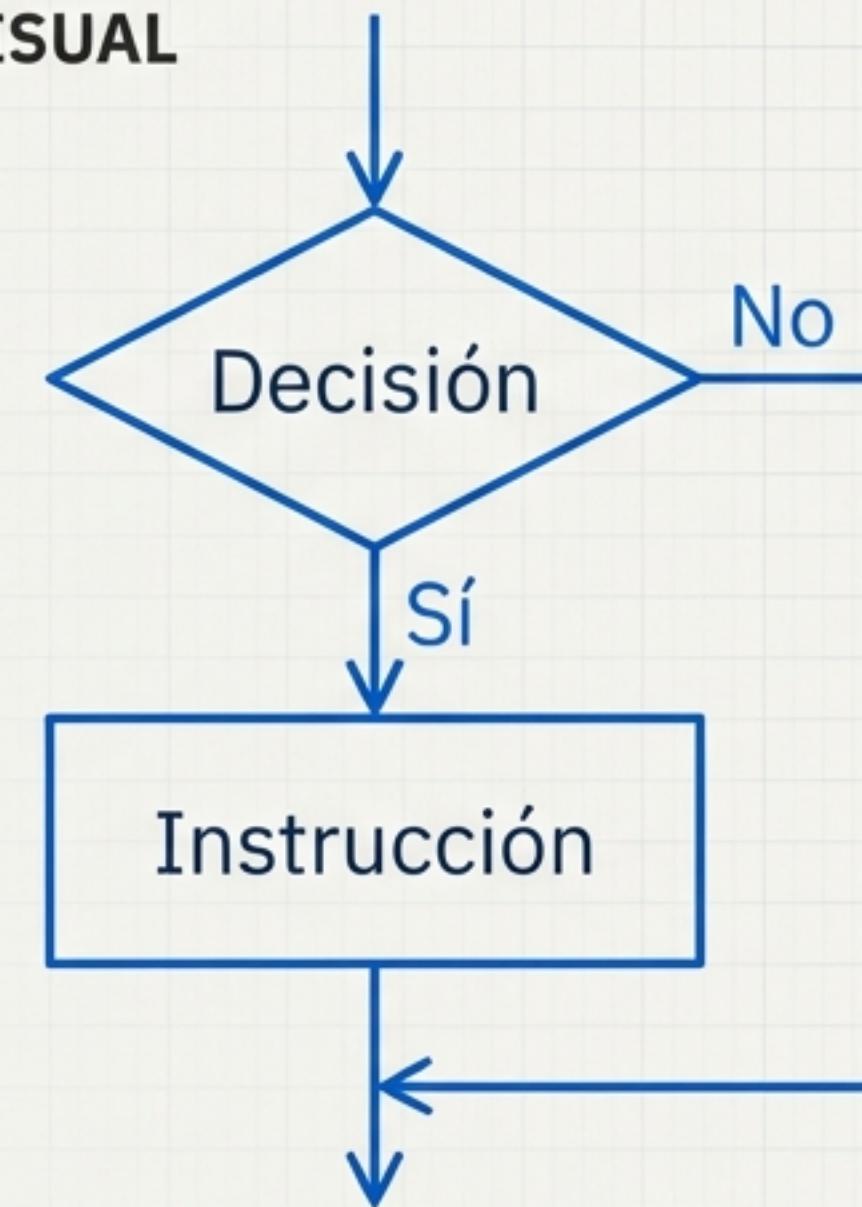
CONCEPTO/CÓDIGO

Permiten que ciertos fragmentos de código se ejecuten solo si una expresión lógica es **Verdadera**.

```
if (condición) {  
    // Sentencias si se cumple  
}
```

Si la condición no se cumple, el programa salta el bloque y continúa.

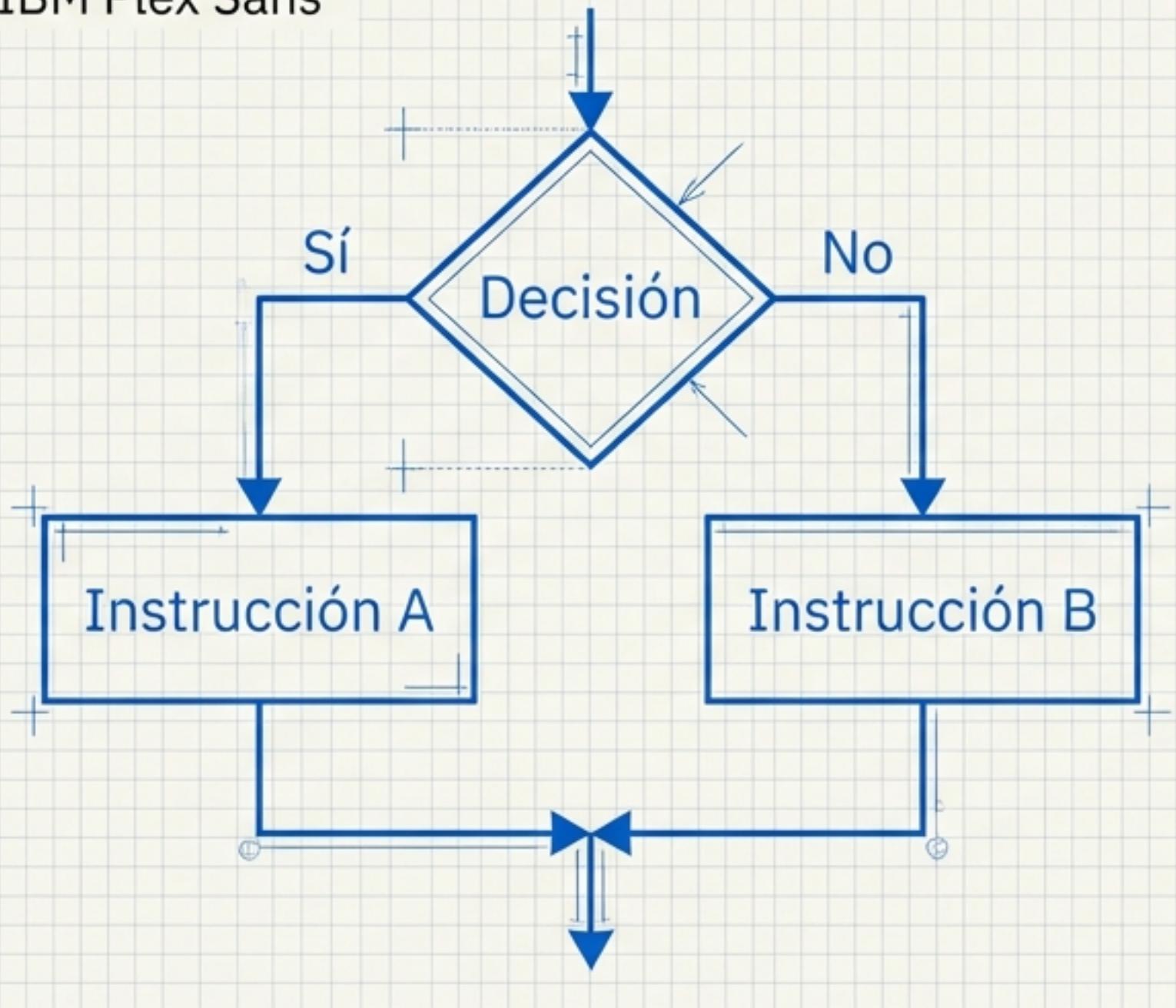
LÓGICA VISUAL



Caminos Divergentes: Selección Doble y Anidada

LÓGICA VISUAL

IBM Plex Sans



CONCEPTO/CÓDIGO

La estructura **if-else** maneja dos flujos: éxito y fracaso.

Podemos meter una decisión dentro de otra (**Anidación**) para tratar condiciones múltiples.

Caso Práctico: ¿Cine o Parque?

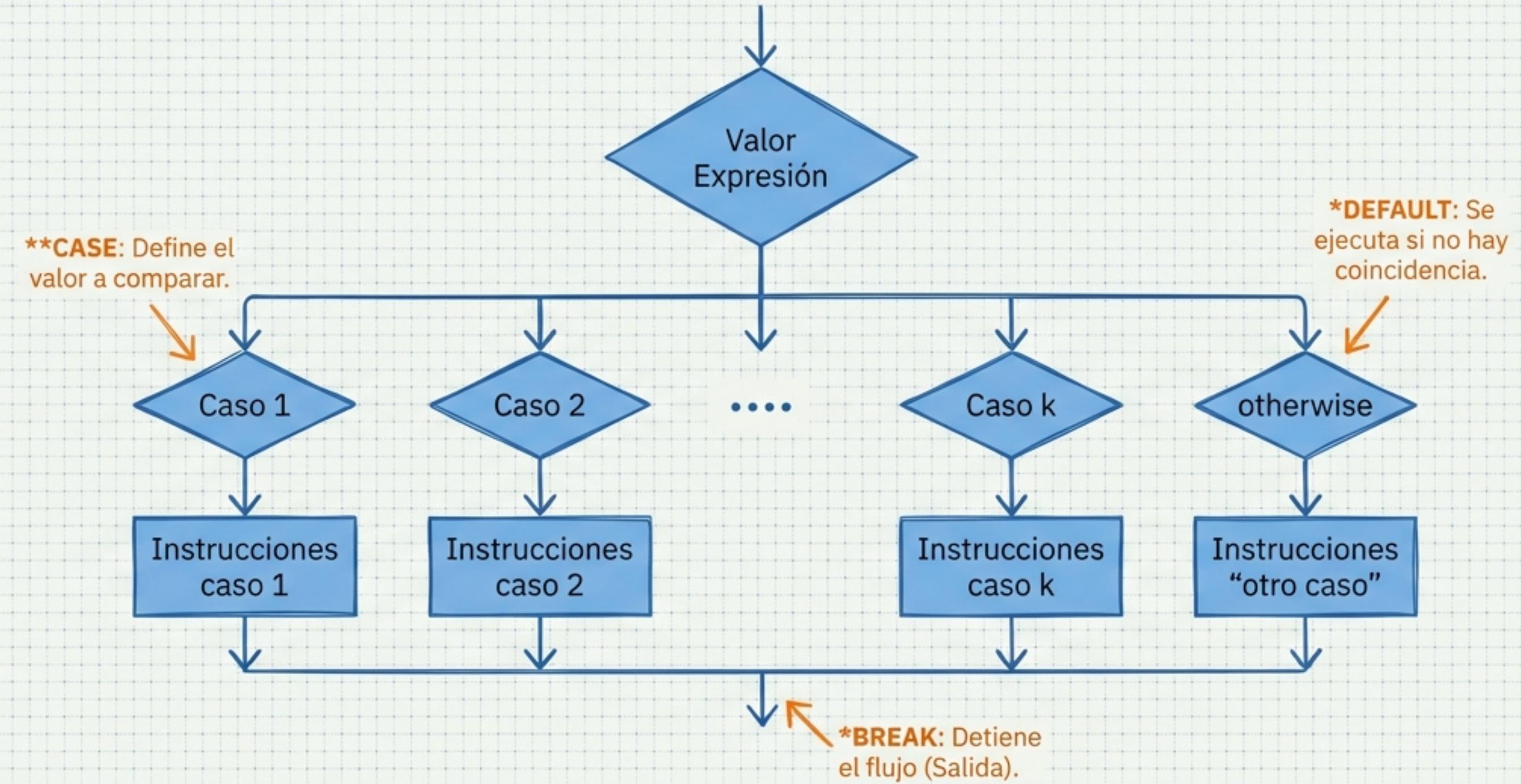
```
Si (Cine es apto para niños) {  
    Entrar al cine  
    Comprar palomitas  
}  
Sino {  
    Ir al parque  
}
```

Organización y Limpieza: Selección Múltiple (Switch)

Estructurando la evaluación de múltiples casos

Problema: Anidar demasiados "if" crea una escalera ilegible.

Solución: "Switch" organiza la evaluación de una variable contra valores específicos.



Estrategia Lógica: ¿Anidar o Operar?

Caso de Estudio: Validación de Acceso (Usuario + Contraseña)

Opción A: Anidación (Pilar)

```
if (usuario.equals("valorUsuario")) {  
    if (contrasena.equals("valorPass")) {  
        System.out.println("Acceso Permitido");  
    }  
}
```

Control granular, pero mayor indentación.

Opción B: Operador Lógico (José)

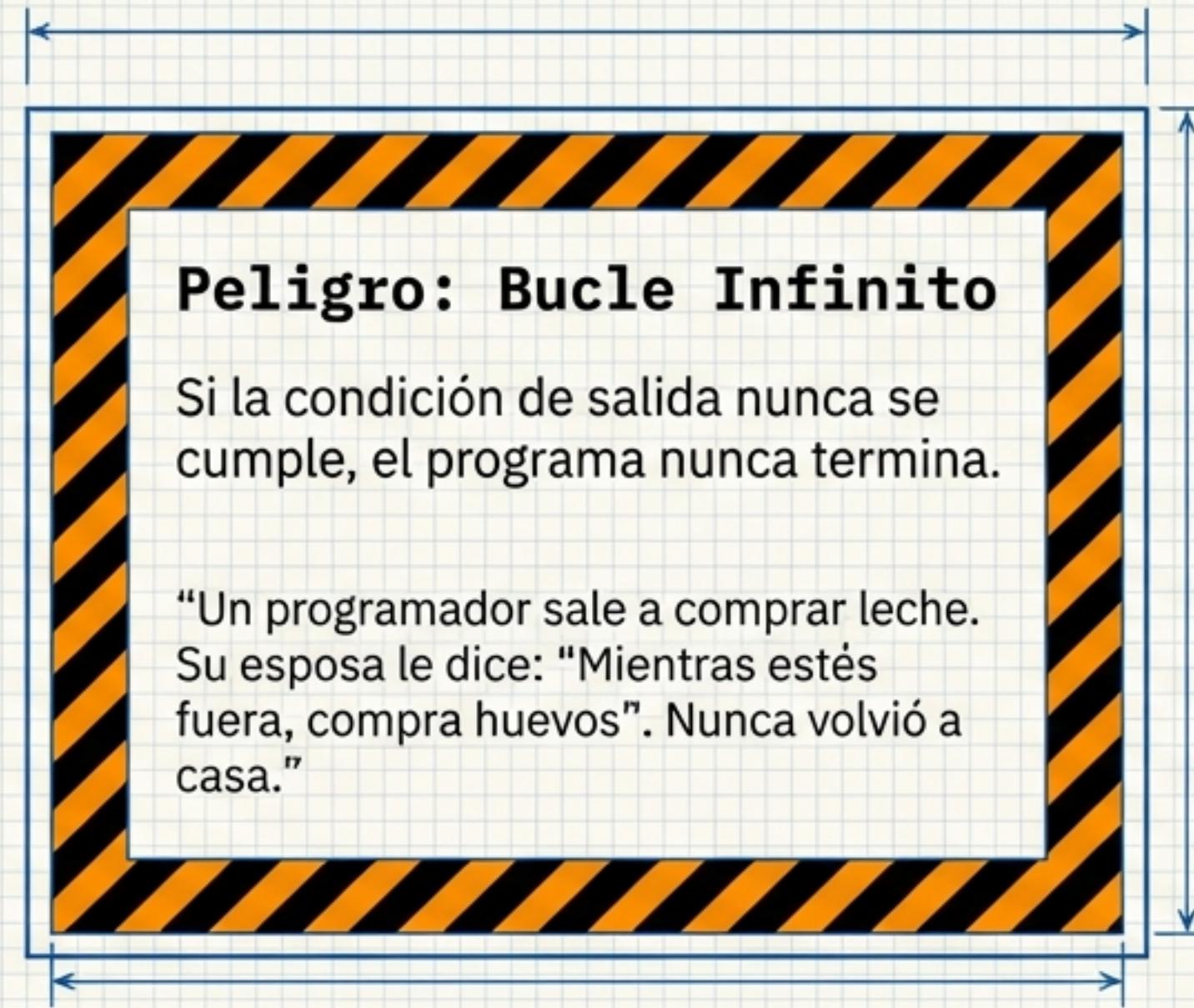
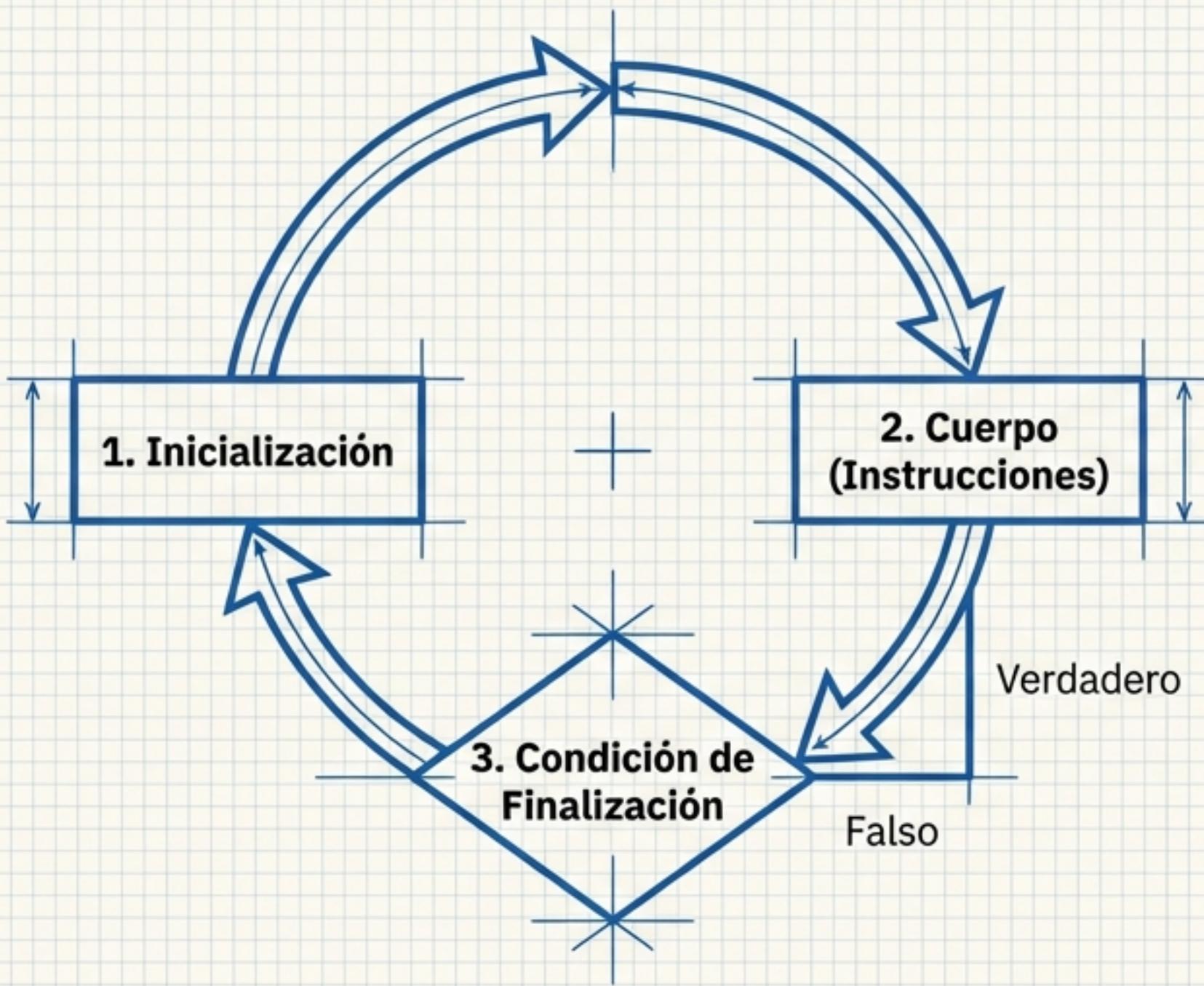
```
if (usuario.equals("valorUsuario") &&  
    contrasena.equals("valorPass")) {  
    System.out.println("Acceso Permitido");  
}
```

Código limpio (**Clean Code**) y compacto con **AND (&&)**.

Veredicto: La legibilidad es prioritaria. Agrupar condiciones suele ser más elegante.

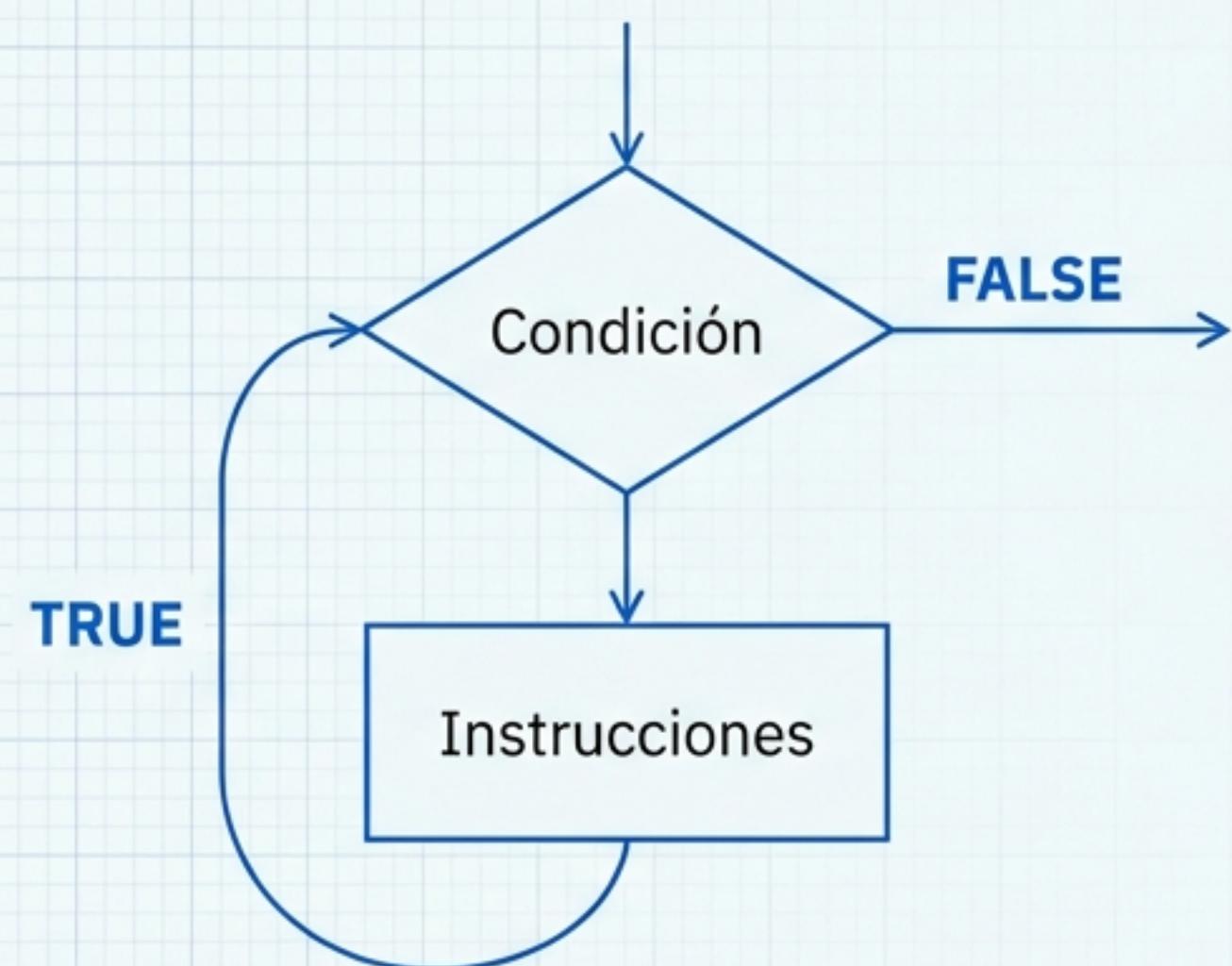
El Ciclo: Estructuras Iterativas

Las estructuras iterativas (bucles) repiten un bloque de código mientras una condición sea cierta.



Precaución Primero: El Bucle While

Evaluación Inicial



Si la condición es falsa al inicio, el código NUNCA se ejecuta.

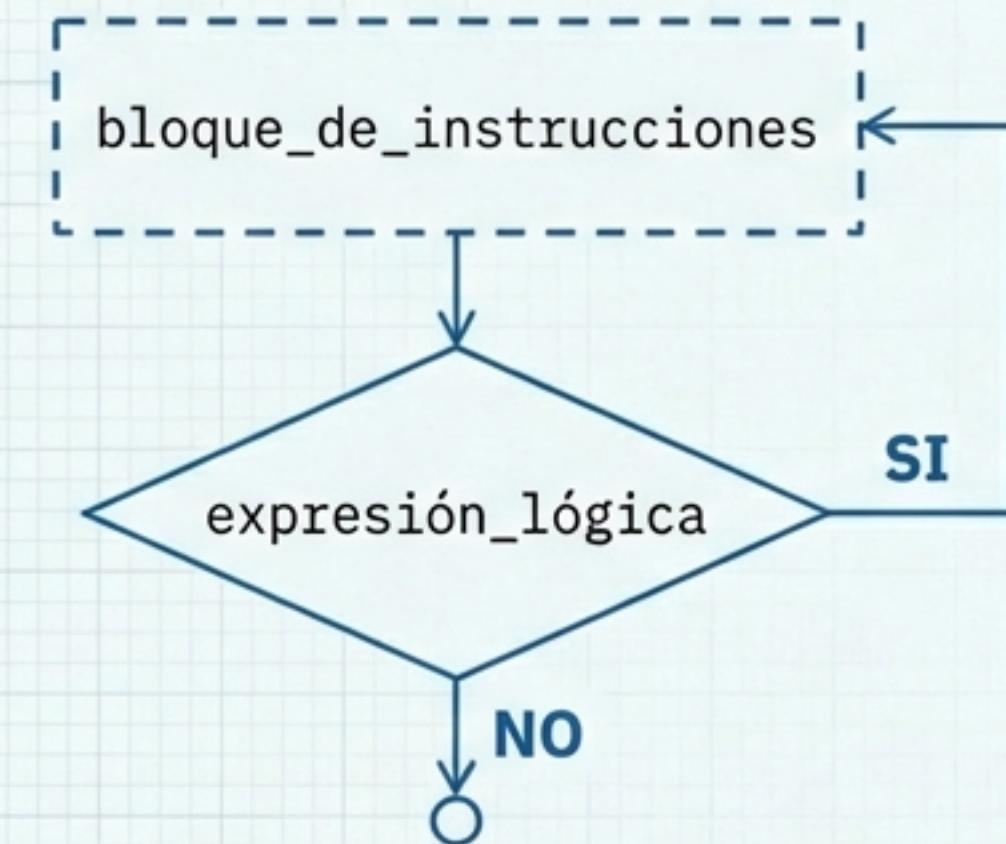
Escenario: Sumar números positivos.

```
while (numero >= 0) {  
    suma = suma + numero;  
    // Pedir siguiente número  
}
```

Acción Primero: El Bucle Do-While

Evaluación Final

Blueprint Blue: FLUJO LÓGICO



Garantiza que el código se ejecute
AL MENOS UNA VEZ.

Blueprint Blue: CONTEXTO DE USO

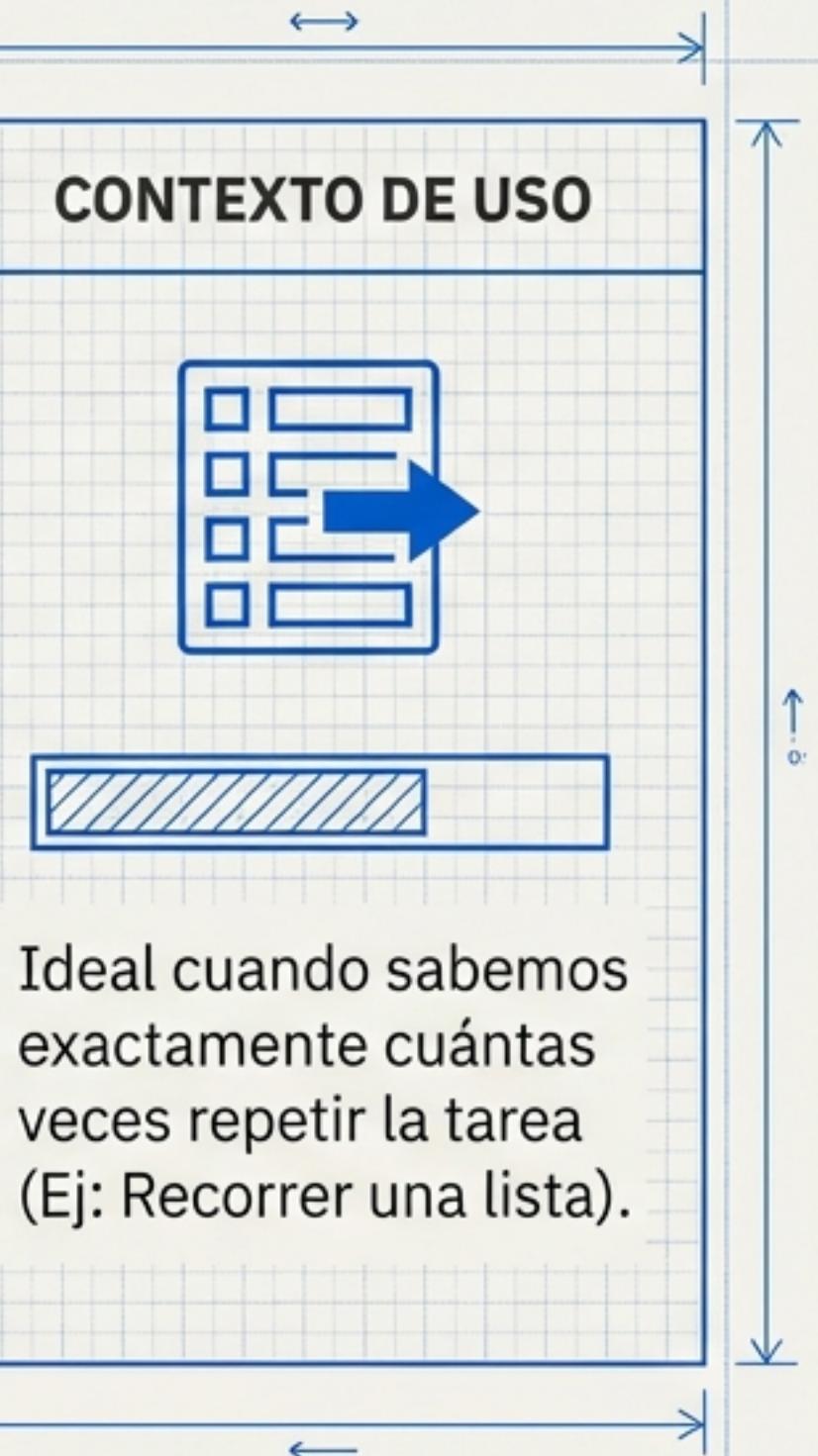
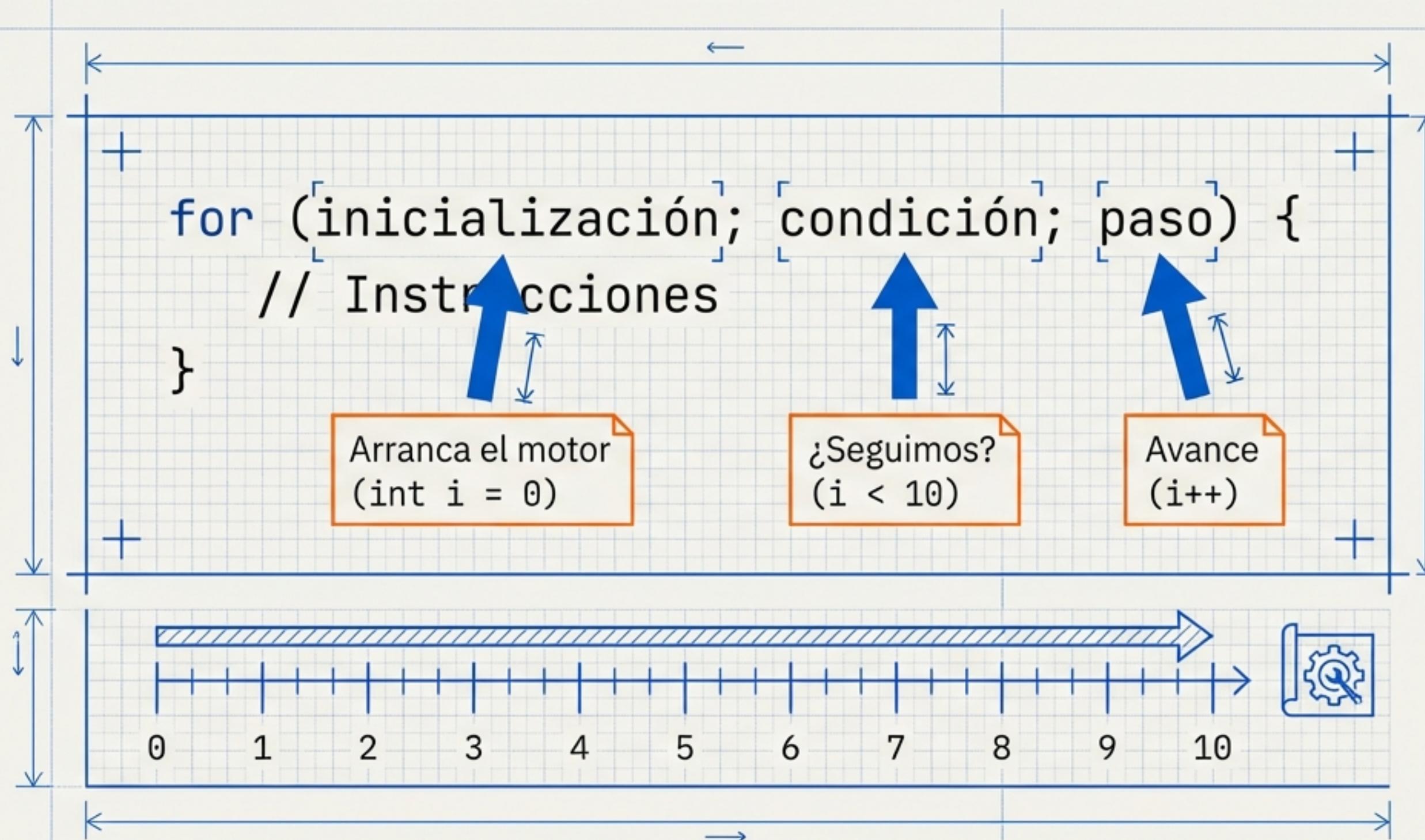


Ideal para menús o
validación de entrada.

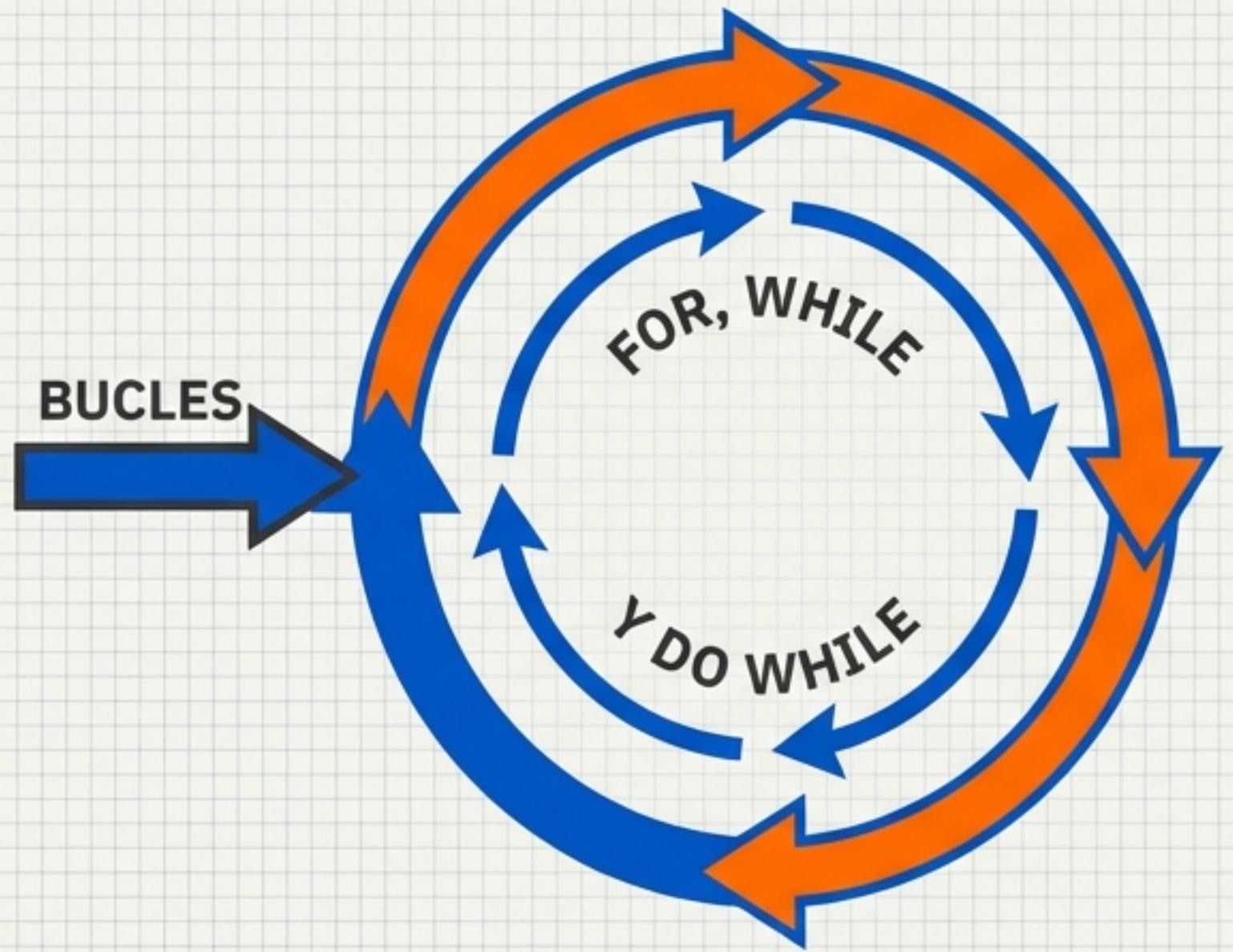
Blueprint Blue
EJEMPLO: Introduce un número
hasta que sea el correcto.

Precisión Controlada: El Bucle For

Iteración Determinada



Eligiendo la Herramienta Correcta



1	Iteraciones Conocidas	→ FOR (Ej: Sumar 100 números)
2	Iteraciones Desconocidas	→ WHILE (Ej: Esperar input “salir”)
3	Ejecución Obligatoria (min. 1)	→ DO-WHILE (Ej: Mostrar Menú)

Todos son equivalentes, pero el contexto dicta la elegancia.

Salidas de Emergencia y Saltos



Salida inmediata.
Abandona el bucle.



Siguiente vuelta.
Salta instrucciones restantes.



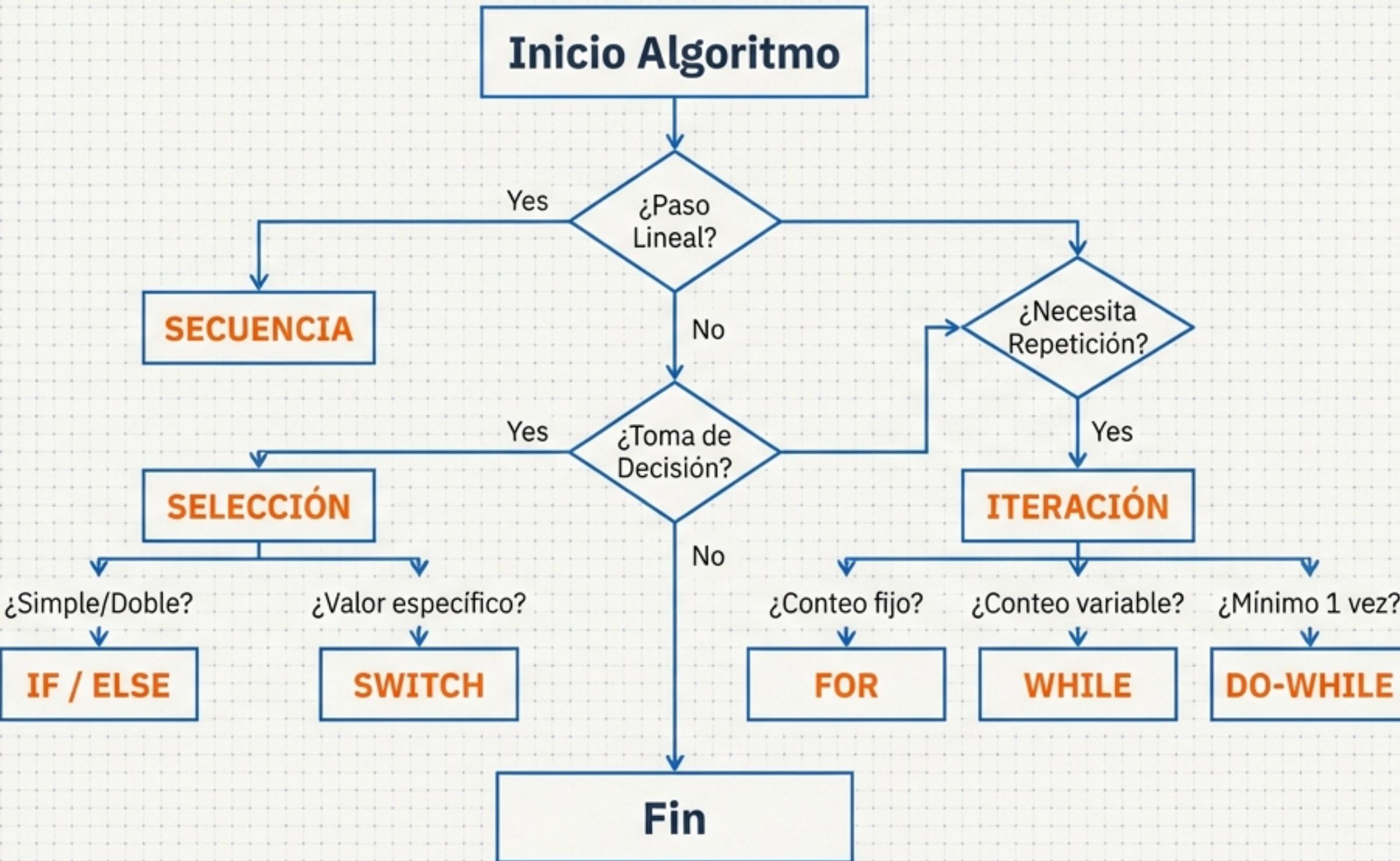
Fin de función.
Devuelve control.

```
// Ejemplo A: continue
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        continue; —————↑
    }
    System.out.println(i);
}
```

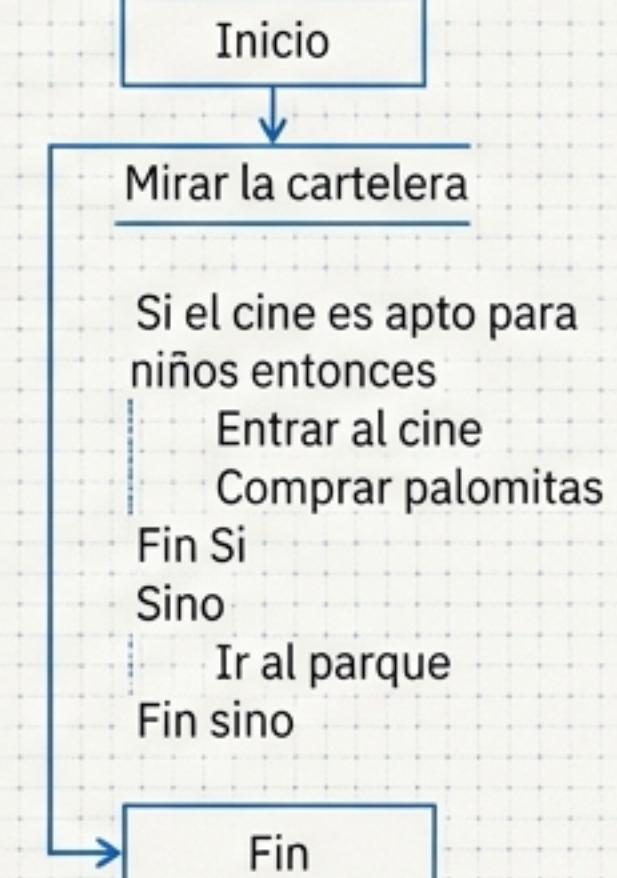
```
// Ejemplo B: break
for (int i = 0; i < 10; i++) {
    if (i == 4) {
        break; —————↑
    }
    System.out.println(i); ←
}
```

GOTO: ¡EVITAR! Crea ‘Código Espagueti’ y dificulta el mantenimiento.

Mapa de Navegación del Algoritmo



Resumen Visual de Estructuras de Control.



El Arte del Código Estructurado

La programación estructurada no es solo sobre hacer que el código funcione; es sobre hacerlo mantenable, lógico y escalable.

- ¿Tiene tu bloque una sola entrada y salida?
- ¿Es legible sin saltos caóticos?
- ¿Has elegido el bucle adecuado?



Domina estas estructuras y podrás construir cualquier algoritmo.