



FULL STACK

**Comenzamos en unos  
minutos**

---

ACADEMY  
by NUMEN

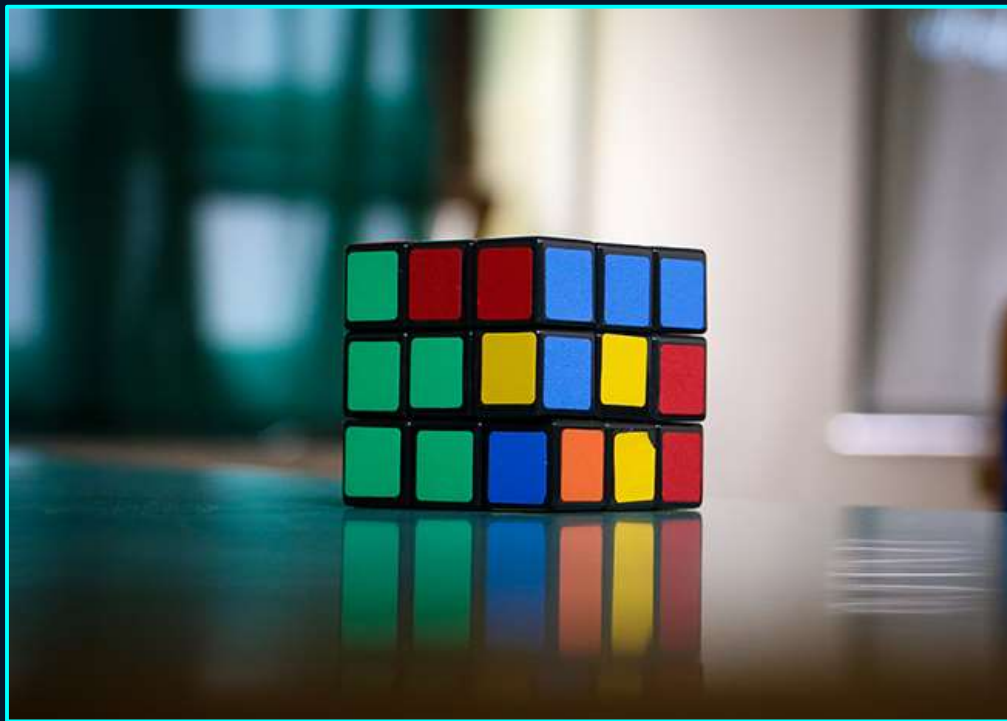
# Introducción a la Lógica de Programación - Primera parte

# ¿Que es la lógica de programación?

# Logica de Programación

La **lógica de programación** es todo conjunto de reglas y conceptos que necesitamos aplicar para crear códigos que serán interpretados y ejecutados por un computador.

Uno de los más importantes es el concepto de **algoritmo**, el cual es, ni más ni menos que un conjunto de instrucciones a ejecutar.



# Razonamiento Lógico

El razonamiento es la facultad que le permite al ser humano resolver problemas, evaluar opciones y elegir la más adecuada por medio pensamiento lógico.

Este tipo de pensamiento está compuesto por 2 características:

💡 Es deductivo

☒ Es analítico



# Razonamiento Deductivo

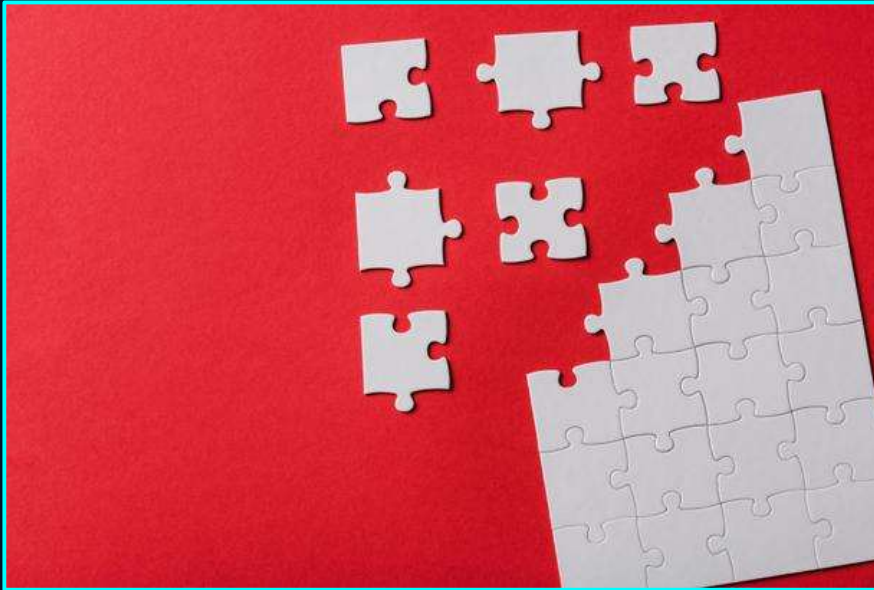
El **razonamiento deductivo** es aquel que parte de ideas generales para llegar a conclusiones particulares. Las conclusiones de los razonamientos deductivos no aportan información nueva, sino que exclusivamente confirman la premisa.





# Pensamiento analítico

El **pensamiento analítico** se define como la capacidad de identificar, separar y organizar sistemáticamente las partes que componen un problema o situación, de modo de facilitar su entendimiento.





# Diseño de algoritmos

# ¿Qué es un algoritmo?

Un **algoritmo** consiste en una serie de instrucciones detalladas finitas y escritas en un lenguaje cotidiano sobre cómo resolver un problema o ejecutar una acción por medio del razonamiento lógico.

Su uso amplía el panorama sobre los elementos que son necesarios para resolver un problema y permite evaluar de forma permanente si el razonamiento utilizado para encontrar una solución es el adecuado o hay que cambiar de enfoque.

## Ejemplos de algoritmos

Algoritmo para preparar una sopa instantánea en el horno de microondas

1. Inicio
2. Destapar la sopa
3. Agregar una taza pequeña con agua a la sopa
4. Introducir al horno de microondas
5. Programar el horno de microondas por 3 minutos
6. Sacar del horno
7. Fin



# Tipos de algoritmo



## Cualitativo

Se utilizan para listar los pasos a seguir en actividades diarias, como una receta de cocina o los pasos para lavarse los dientes.



## Cuantitativo

Este tipo de algoritmo implica cálculos numéricos, por ejemplo calcular descuentos a un producto o el costo de envío de MercadoLibre.

# ¿Cómo redactar un algoritmo?

Para redactar un algoritmo es necesario tener en cuenta 3 aspectos fundamentales:

- □ Todo algoritmo necesita una **entrada** (elementos).
- □ Todo algoritmo necesita **acciones** a ejecutar.
- □ Todo algoritmo necesita un **resultado** esperado sobre el cual basarse.

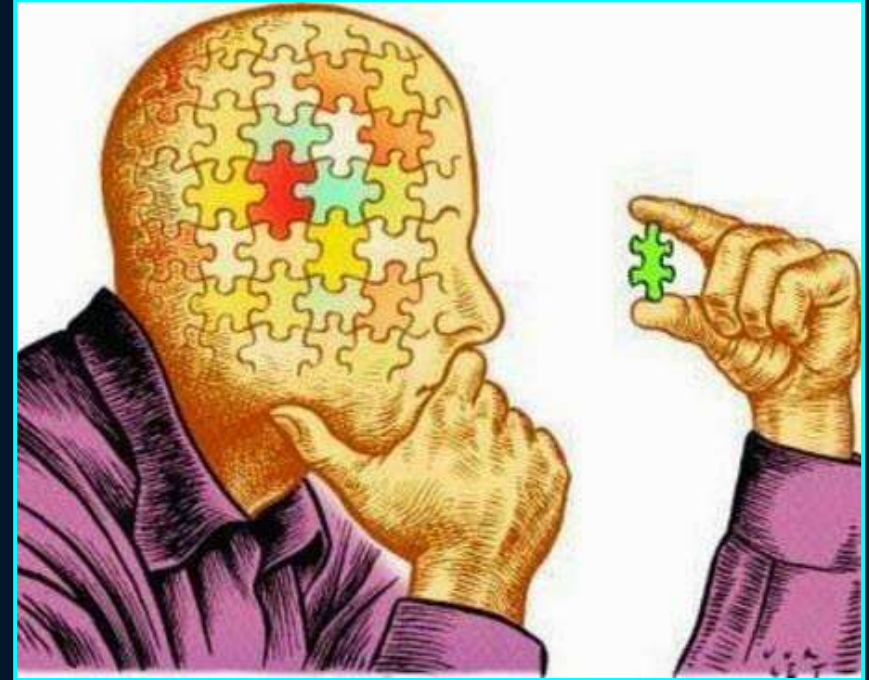


# Abstracción y Variables

# Abstracción

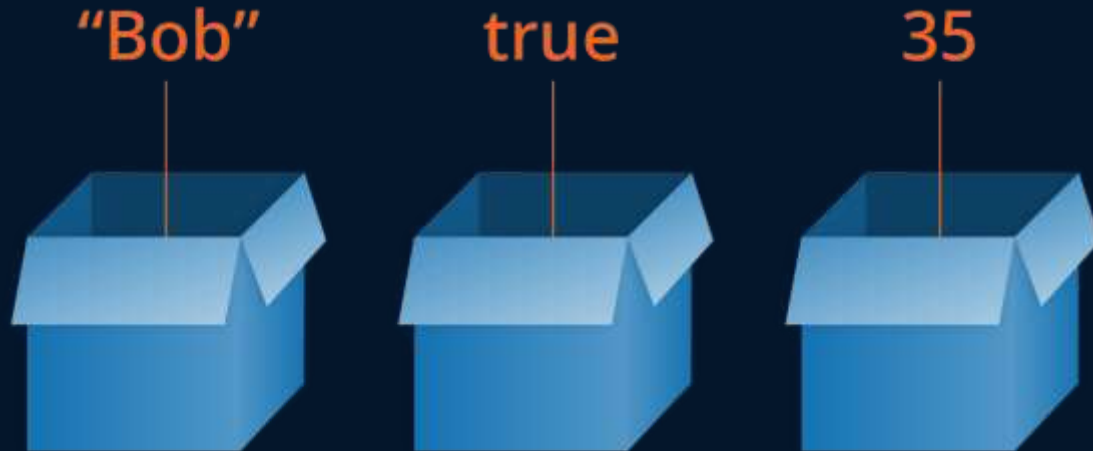
El pensamiento lógico facilita la abstracción del problema, es decir, la identificación de la información relevante de un problema o situación a resolver.

Esta abstracción es lo que realizamos cuando por ejemplo jugamos Sudoku, Ajedrez o elegimos qué información consumir de una base de datos para mostrar al cliente solo lo que este necesita en lugar de toda la información disponible.



# Variables

Cuando seleccionamos información relevante que abstraer, en programación, necesitamos guardar esa información para poder utilizarla con posterioridad. Para ello disponemos de las **variables**. Estas son, a modo explicativo, como cajas a las cuales podemos rotular con el nombre que queramos y en las que podemos guardar esos datos que quisiéramos utilizar más adelante.



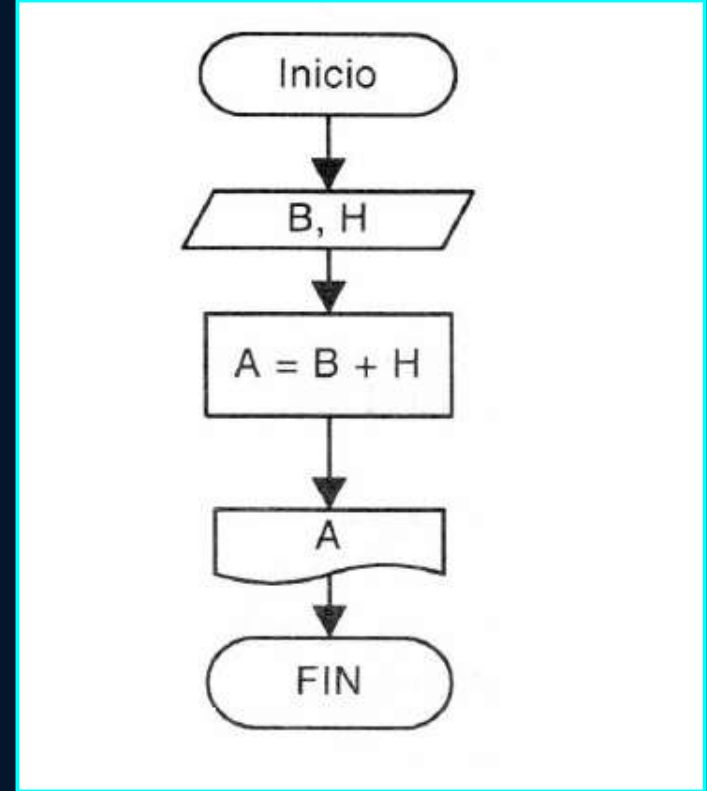


# Diagrama de Flujo de Datos



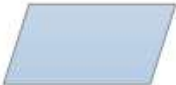
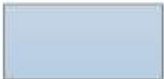

# ¿Qué es un diagrama de flujo?

Hasta ahora vimos como sería un algoritmo escrito como una simple lista. Es hora de ir un paso más allá.

Un **diagrama de flujo** es la representación gráfica de un algoritmo a través de símbolos que se relacionan entre sí e indican cómo deben ejecutarse las instrucciones para obtener un resultado.



# Simbología básica del diagrama de flujo

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Linea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

# Operadores

# Operadores aritméticos

Son los que te permiten realizar operaciones matemáticas, como suma, resta, multiplicación, etc y se utilizan en algoritmos cuantitativos para encontrar soluciones a un problema.

Operadores aritméticos			
Operador	Nombre	Ejemplo	Resultado
+	Suma	=10+5	15
-	Resta	=10-5	5
-	Negación	=-10	-10
*	Multiplicación	=10*5	50
/	División	=10/5	2
%	Porcentaje	=10%	0.1
^	Exponenciación	=10^5	100000

# Operadores de comparación

Los operadores de comparación comparan dos expresiones y devuelven un Boolean valor que representa la relación de sus valores. Hay operadores para comparar valores numéricos, operadores para comparar cadenas y operadores para comparar objetos.

Operador	Nombre	Ejemplo	Resultado
=	igual	=4=5	FALSO
>	mayor que	=8>6	VERDADERO
<	menor que	=3<2	FALSO
>=	mayor o igual que	5>=7	FALSO
<=	menor o igual que	2<=1	VERDADERO
<>	diferente	3<>3	FALSO

# Operadores lógicos

Se utilizan para evaluar 2 o más expresiones que utilizan operadores relacionales para determinar si la expresión en general es verdadera o falsa.

Operación	Operador	Ejemplo	Comparación	Resultado/Binario
AND	&&	9 > 8 && 11 > 10	¿9 es mayor que 8 <b>Y</b> 11 es mayor que 10?	Verdadero (1)
		9 > 7 && 10 < 9	¿9 es mayor que 7 <b>Y</b> 10 es menor que 9?	Falso (0)
OR		5 > 4    3 > 3	¿5 es mayor que 4 <b>O</b> 3 es mayor que 3?	Verdadero (1)
		5 > 5    3 == 4	¿5 es mayor que 5 <b>O</b> 3 es igual a 4?	Falso (0)
NOT	!	!(5>4)	¿5 <b>NO</b> es mayor que 4?	Falso (0)
		!(5<4)	¿5 <b>NO</b> es menor que 4?	Verdadero (1)



# Estructura condicional

# ¿Qué es la estructura condicional?

Existen problemas complejos para los que el flujo de datos lineal es insuficiente ya que para solucionarlos se debe elegir qué bloque de datos se debe ejecutar.

La finalidad de utilizar una estructura condicional es tomar una decisión con base en el valor booleano de una expresión, es decir, determinar si la condición es verdadera o falsa.

Estas estructuras se clasifican en tipos:

- Simple
- Compuesta
- Múltiple



# Estructura condicional simple

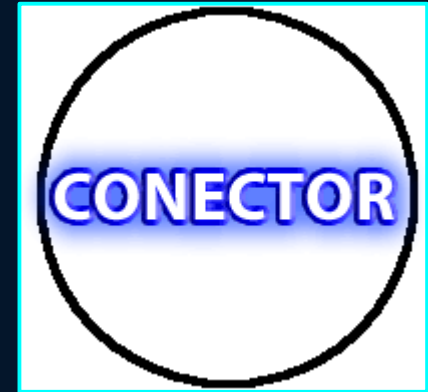
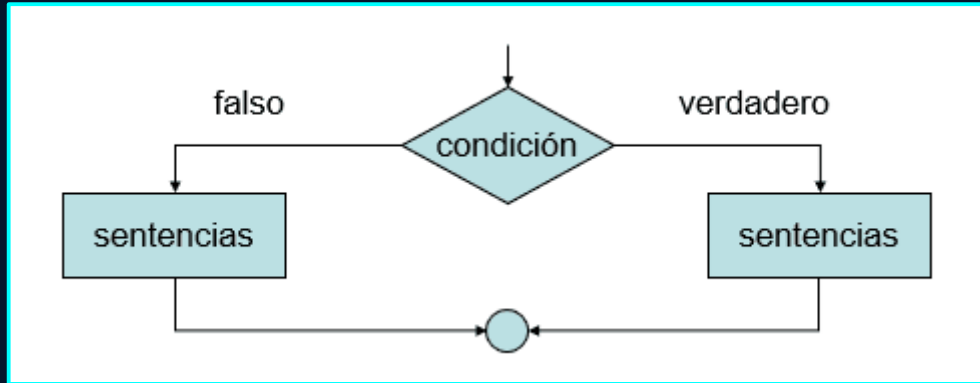
La estructura ejecuta un bloque de instrucciones cuando la condición es verdadera.

En caso contrario ignora ese bloque y continúa con la ejecución del resto de las instrucciones.



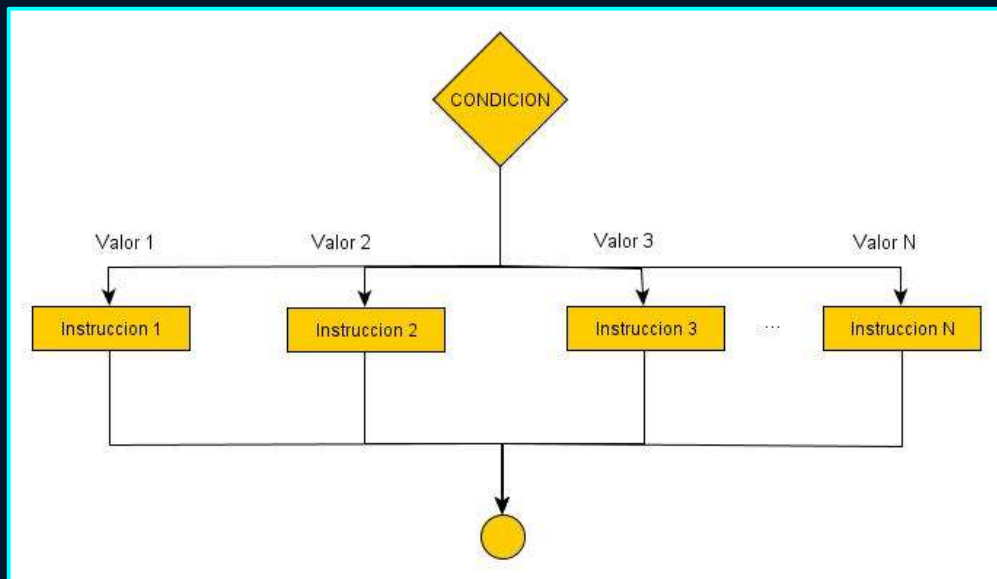
# Estructura condicional compuesta

Evalua una condición. Si esta es verdadera ejecuta el bloque de instrucciones más cercano, en caso contrario realiza una acción alternativa alternativa.



# Estructura condicional multiple

Evalúa una condición y dependiendo de su valor booleano, elige el bloque de instrucciones a ejecutar de entre varias opciones.



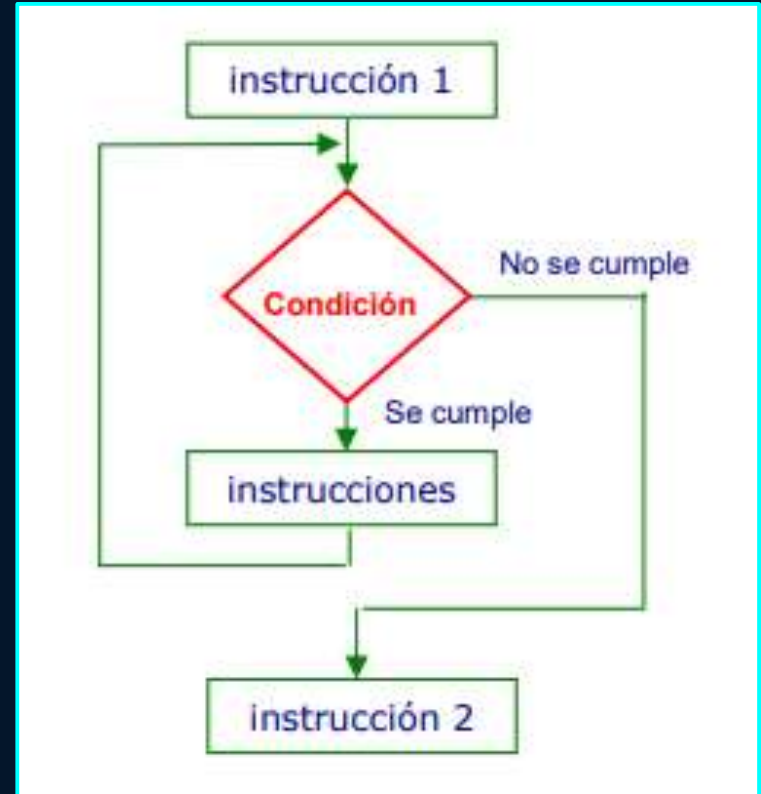
# Estructura iterativa

# ¿Qué es la estructura iterativa?

Las estructuras iterativas permiten ejecutar un conjunto de instrucciones las veces que sea necesario mientras se cumpla una condición, es decir, realiza repeticiones o bucles.

Cuando un ciclo se completa se comprueba nuevamente la condición y si es falsa el bucle se interrumpe.

Hay 3 tipos de estructuras iterativas, veamos cuales son:

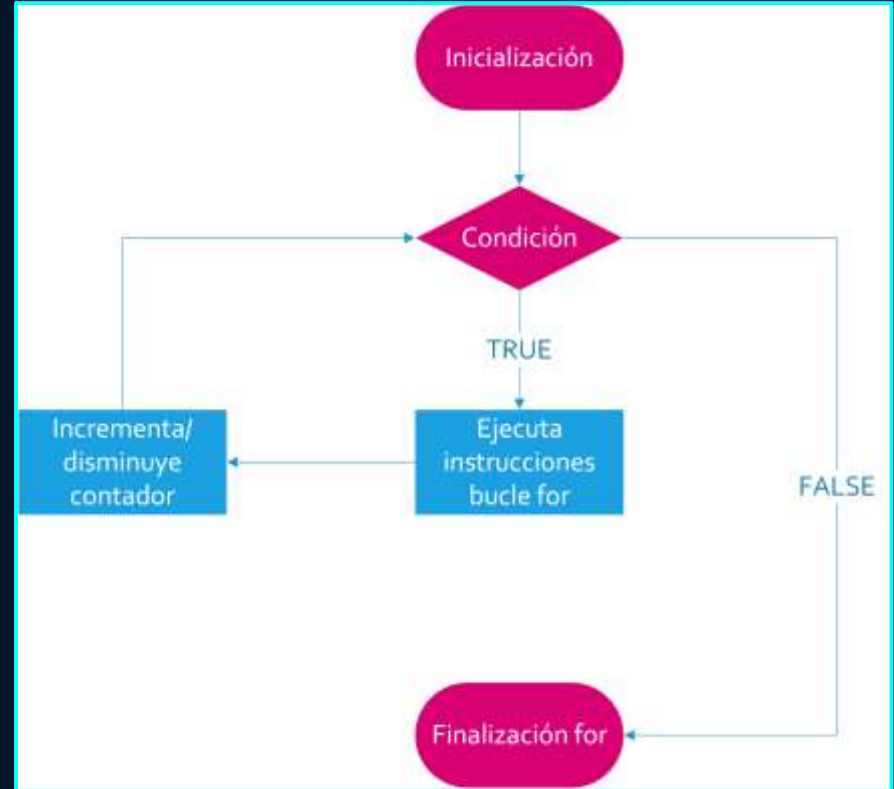




# Estructura Para (for)

En esta estructura se establece el número de veces que una serie de instrucciones debe repetirse, como parte de la solución a un problema en un algoritmo.

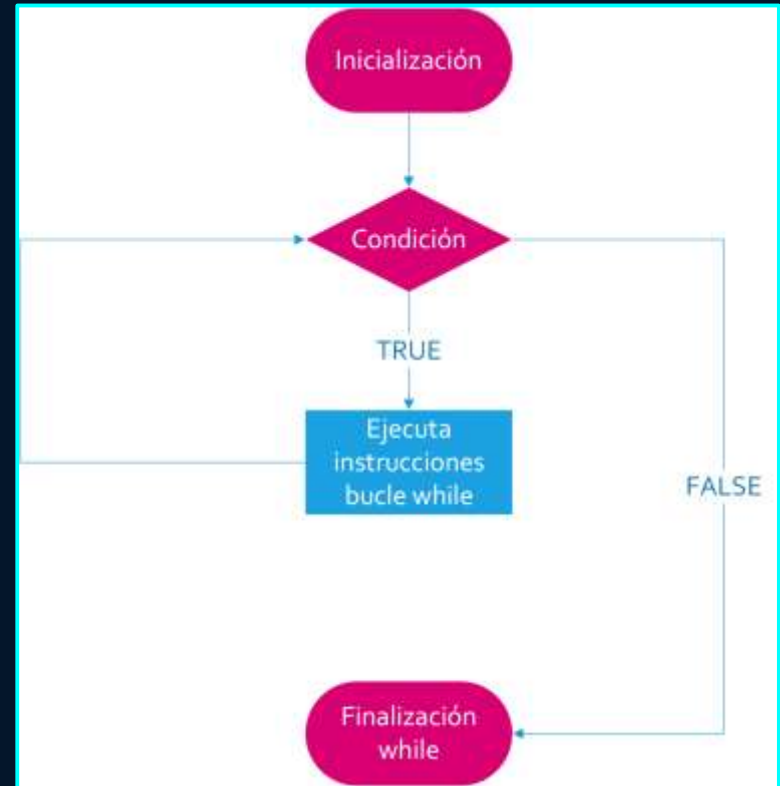
Se compone de una **expresión inicial**, que indica con qué valor numérico inicia el ciclo, una **condición**, que es la expresión relacional o lógica por evaluar que determina cuándo se detendrá el ciclo, y el **incremento** que indica el valor numérico que se le sumará a la expresión inicial tras completar el ciclo.



# Estructura Mientras (while)

Se utiliza para ejecutar un bloque de instrucciones en un ciclo sin necesidad de establecer el número de veces que lo hará.

Se compone por una expresión lógica, relacional o aritmética que es evaluada en una condición.

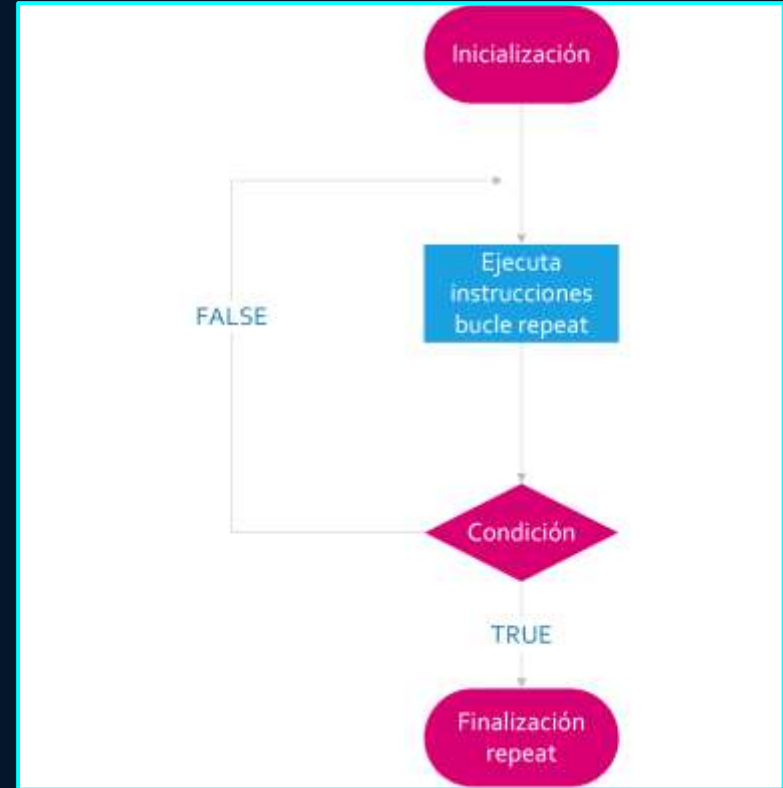


# Estructura Hacer mientras (do while)

Esta estructura se usa cuando el problema a resolver requiere que se ejecute por lo menos una vez el ciclo.

Se compone por una condicional cuya expresión se evalúa después de ejecutar el bloque de instrucciones.

El ciclo finaliza cuando la condición se vuelve falsa.



ACADEMY  
by NUMEN