

# PROGRAMACIÓN I

Presencial Turno noche

# Horarios de clases

 Diagramación

**Martes** de 18hs a 22hs  
 Aula 14

 Codificación

**Viernes** de 18hs a 22hs  
 Laboratorio 4

# Docentes

## Diagramación

**Maximiliano Sar Fernandez**

[msarfernandez@docentes.frgp.utn.edu.ar](mailto:msarfernandez@docentes.frgp.utn.edu.ar)

**José Alberto Rodriguez**

[jarodriguez@docentes.frgp.utn.edu.ar](mailto:jarodriguez@docentes.frgp.utn.edu.ar)

**Alejandro Gomez Nieto**

[alejandro.gomez@docentes.frgp.utn.edu.ar](mailto:alejandro.gomez@docentes.frgp.utn.edu.ar)

# Docentes

## Codificación

**José Alberto Rodriguez**

[jarodriguez@docentes.frgp.utn.edu.ar](mailto:jarodriguez@docentes.frgp.utn.edu.ar)

**Alejandro Gomez Nieto**

[alejandro.gomez@docentes.frgp.utn.edu.ar](mailto:alejandro.gomez@docentes.frgp.utn.edu.ar)

# Pautas de aprobación

## Diagramación

**Parcial 1** Se aprueba con 6 o más.

**Parcial 2** Se aprueba con 6 o más.

En caso de no haber aprobado el Parcial 1, el mismo podrá ser considerado recuperado si en el Parcial 2 se obtiene una nota de 8 o más. Caso contrario, ambos parciales cuantan con una etapa de recuperación.

## Códificación

**Trabajo Práctico Integrador** Al final del cuatrimestre se deberá presentar una aplicación de Consola que deberán desarrollar de manera grupal. Tendrá un tiempo asignado para su desarrollo, resolver dudas y trabajar en los avances. El Trabajo Práctico Integrador se califica mediante una defensa individual.

En caso de que la defensa del Trabajo Práctico Integrador no resulte satisfactoria, habrá una instancia de recuperación en el que deberán regresar con los cambios propuestos.

# Contenido de la materia

01

▶ Etapa 1

Tipos de datos

02

Variables y constantes

03

Operadores

04

Estructura secuencial

05

Estructura de decisión

- Simple
- Múltiple

06

Estructura de repetición

- Ciclo exacto
- Ciclo inexacto
- Ciclos combinados

07

▶ Etapa 2

Funciones

- Tipos de parámetros

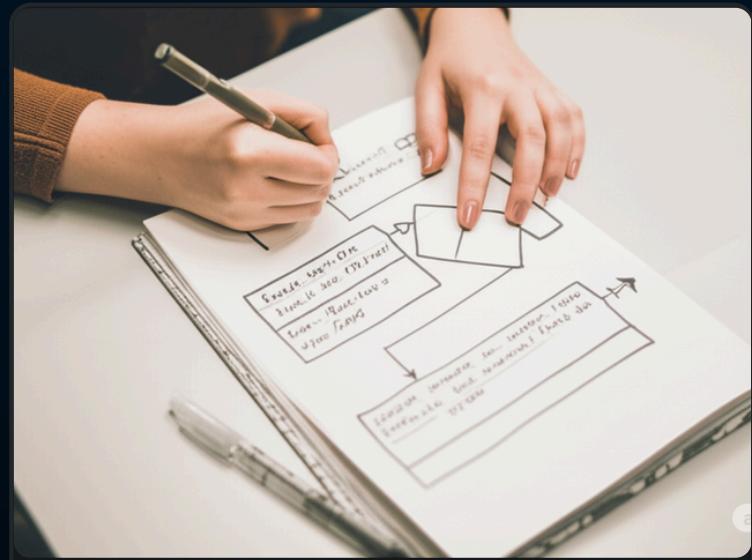
08

Vectores

09

Proyecto de software

# Herramientas de diagramación



Papel y lapiz



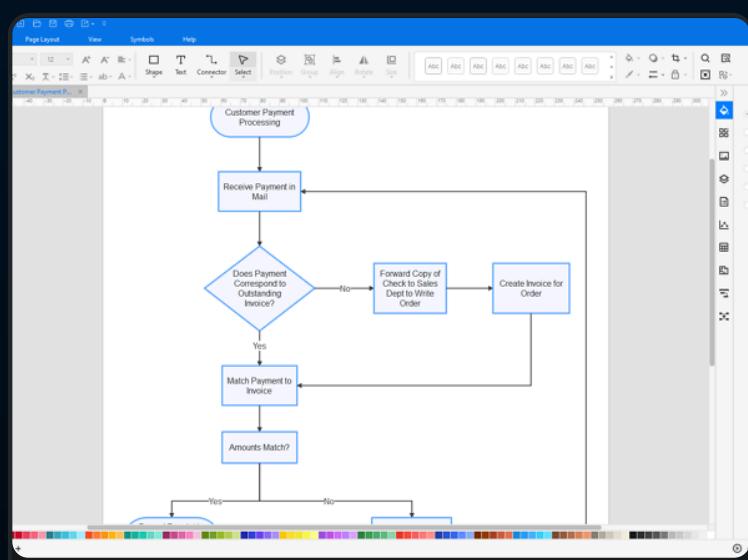
Muy recomendado para los ejercicios diarios



Tableta



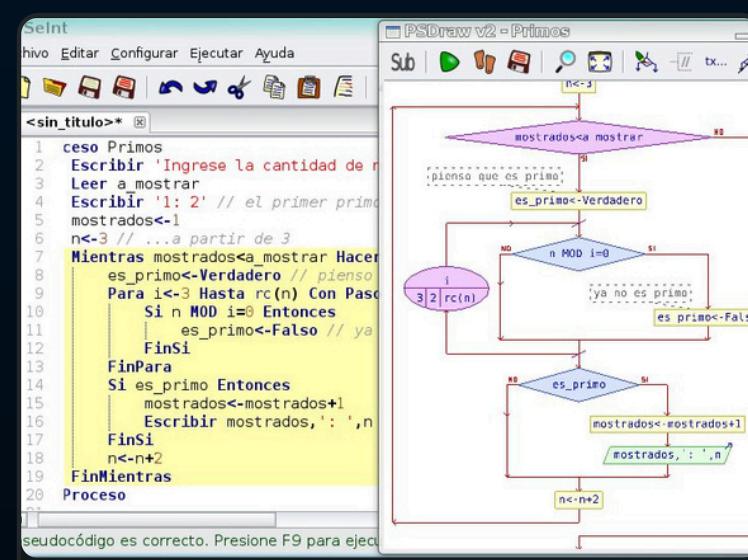
Recomendado para compartir los ejercicios



Aplicaciones web



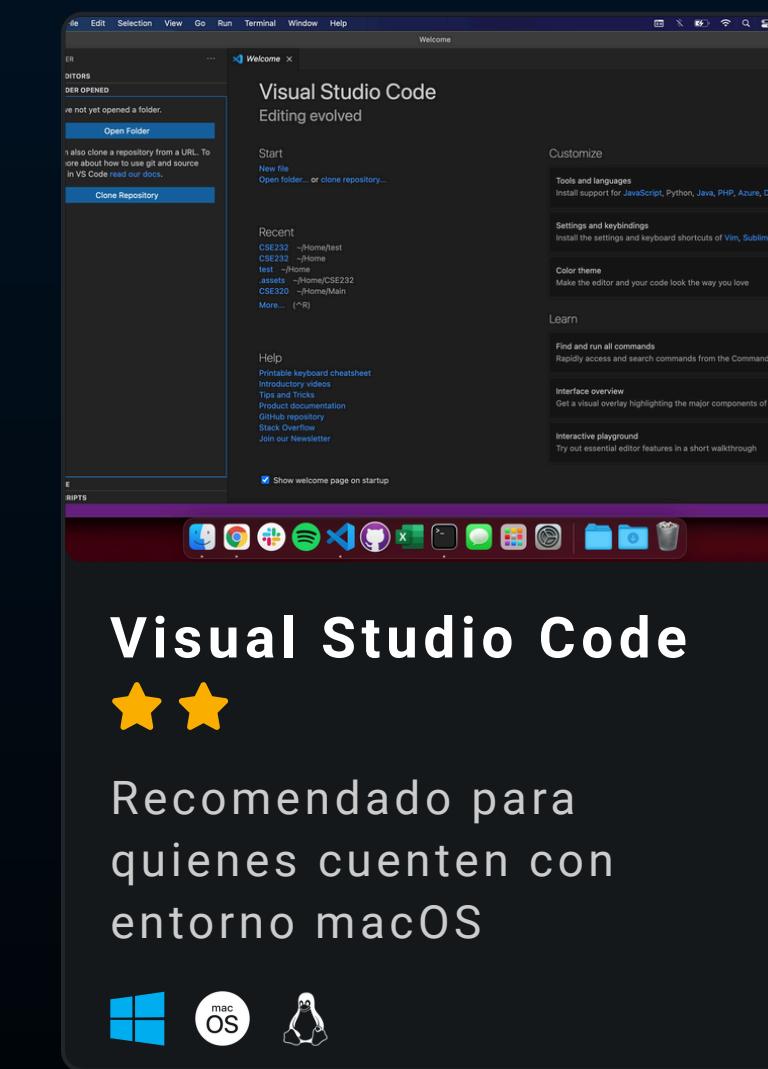
Recomendado para entregas proljas



Software

Evitar las herramientas de diagramación

# Herramientas de codificación

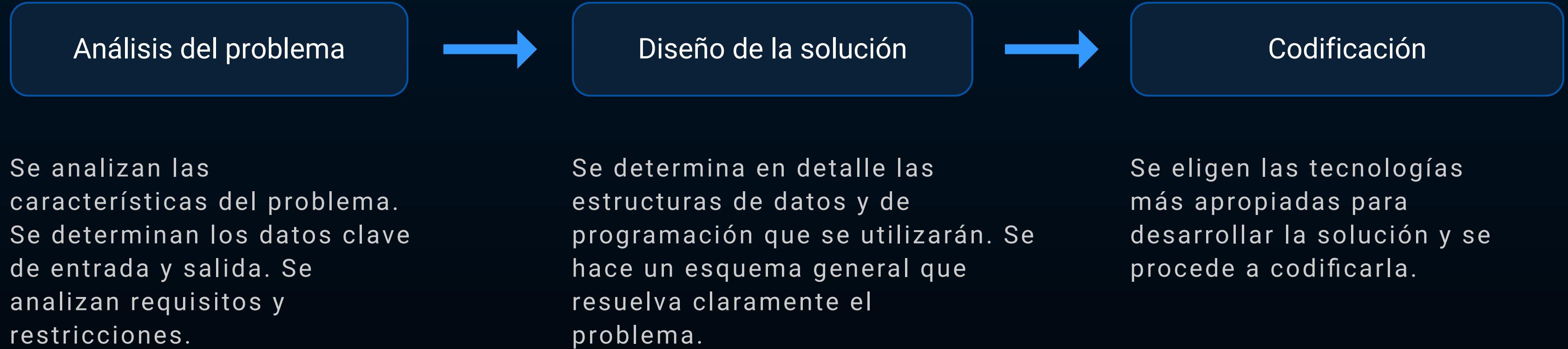


En caso de no contar con una computadora, también se pueden utilizar aplicaciones webs como [onlinegdb.com](http://onlinegdb.com)

# ¿Qué es programar?

¿Es lo mismo programar que codificar?

# Proceso de desarrollo de un programa



# Proceso de resolución del problema

DATOS DE ENTRADA

PROCESO

INFORMACIÓN DE SALIDA

Determinar cuántos y cuáles son los datos de entrada de nuestro programa. Ponerles un nombre y determinar su tipo de datos.

El algoritmo ideado debe poder transformar los datos de entrada en la información de salida.  
Se apreciará que el algoritmo sea eficiente en su resolución. Resolviendo el problema de la mejor manera y con el menor costo de recursos posibles.

La información de salida debe ser clara, prolífica e informar estrictamente lo necesario.

# Diseño de un algoritmo

## Estructura de datos

Variables simples o complejas, vectores, matrices, etc.



## Estructura de programación

Decisiones, estructuras de repetición, funciones, etc.



## Esquema del algoritmo

Secuencia de instrucciones precisas y finitas que resuelven un problema en particular. El conjunto de instrucciones en un lenguaje determinado se llama código fuente.

# Del código al programa

El proceso de transformar un código fuente en un programa que contiene las instrucciones que sean comprensibles por la computadora se llama compilación.



Codificación

# Estructura de control

01. Secuencial

# Codificación

Cada elemento que utilicemos en un lenguaje de programación debe estar sujeto a una estricta sintaxis. Los elementos que el lenguaje admite son:

- 01** Variables y Constantes
- 02** Operadores
- 03** Expresiones
- 04** Palabras reservadas

# Variables

Azucar



Sal



Harina



# Variables

Azucar



Sal

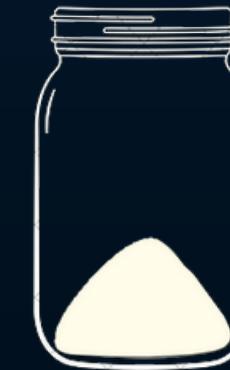


Harina



# Variables

Azucar



Sal

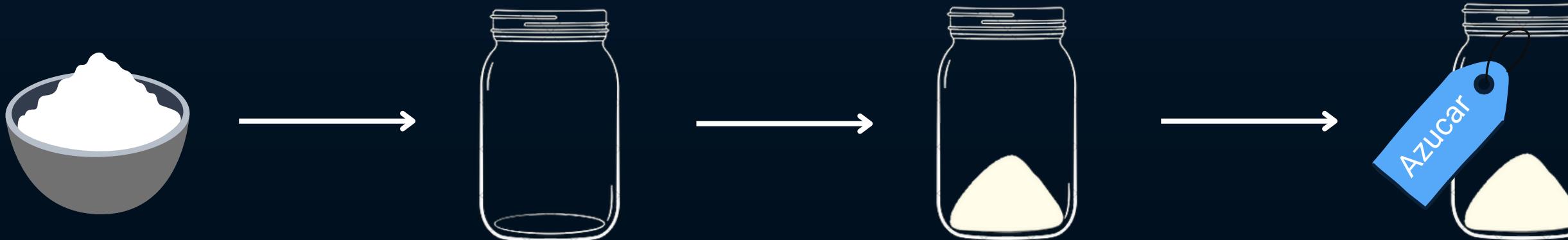


Harina

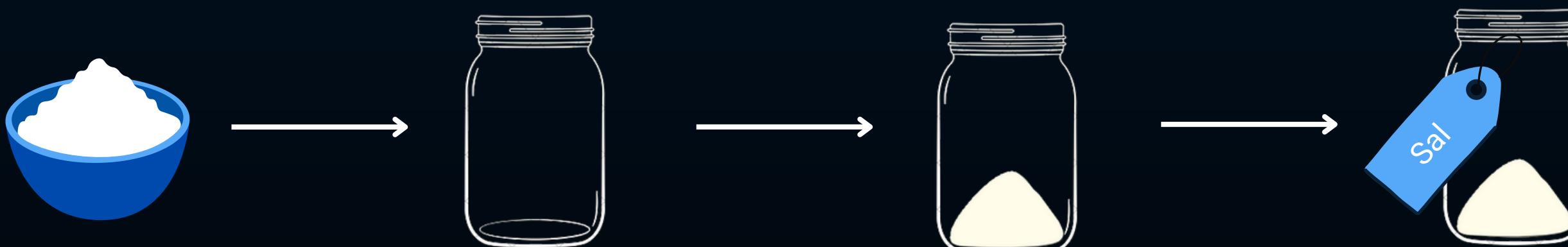


# Variables

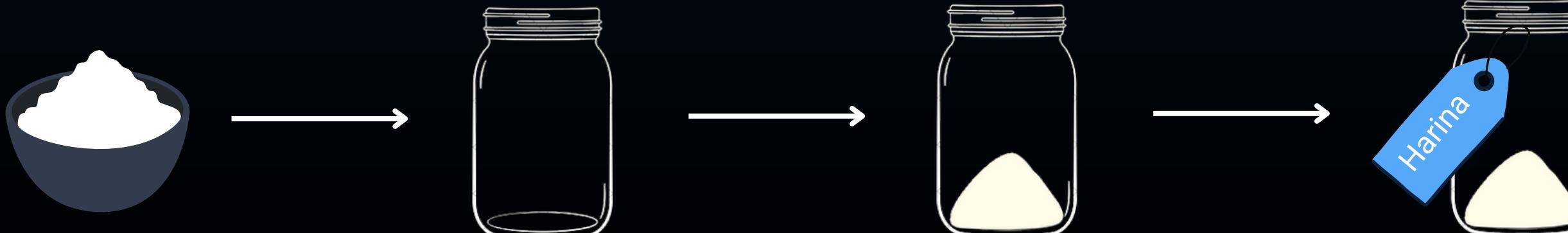
Azucar



Sal



Harina



# Variables

Representación simbólica de espacio de memoria. Es donde se almacenan los datos en procesamiento.

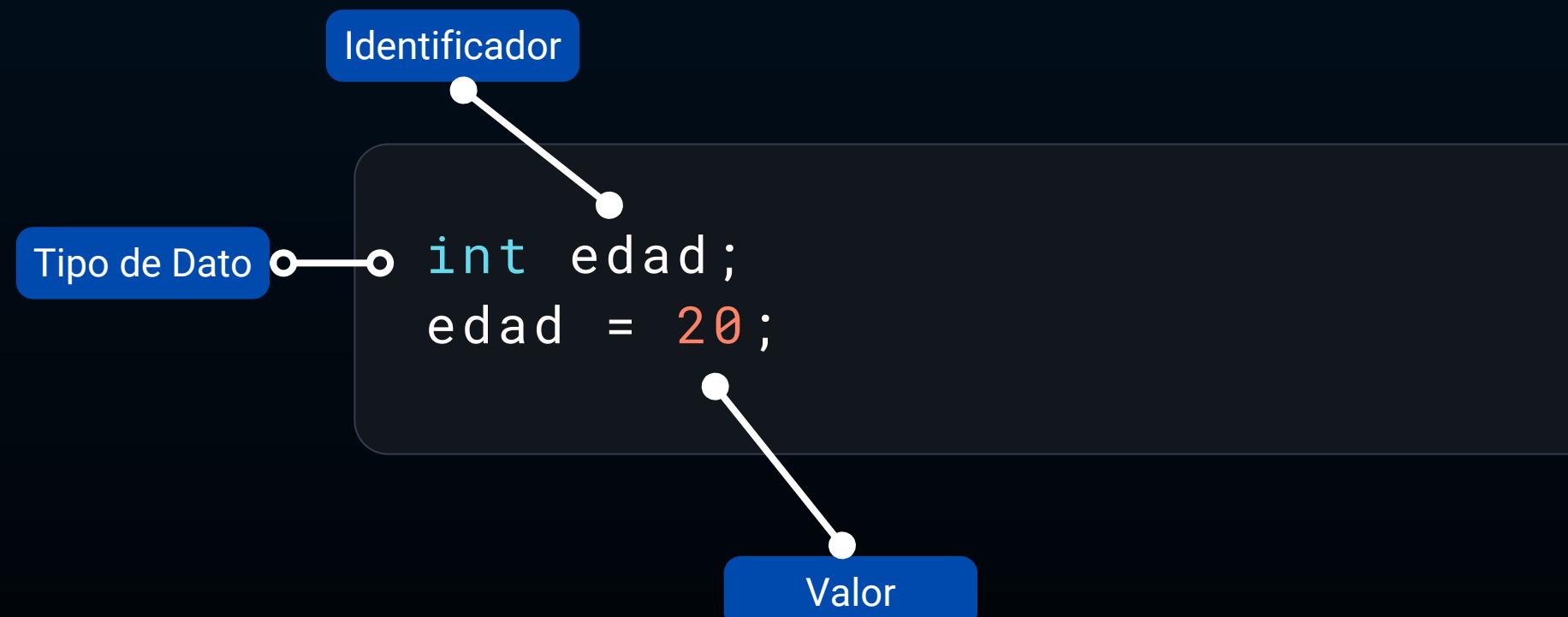
Una variable se identifica con un **tipo de dato** y un **identificador de nombre** (los elige el programador), y permite escribir un **valor** en la memoria o leer un dato de la memoria. Se puede modificar su valor las veces que sea necesario.

```
int edad;  
edad = 20;
```

# Variables

Representación simbólica de espacio de memoria. Es donde se almacenan los datos en procesamiento.

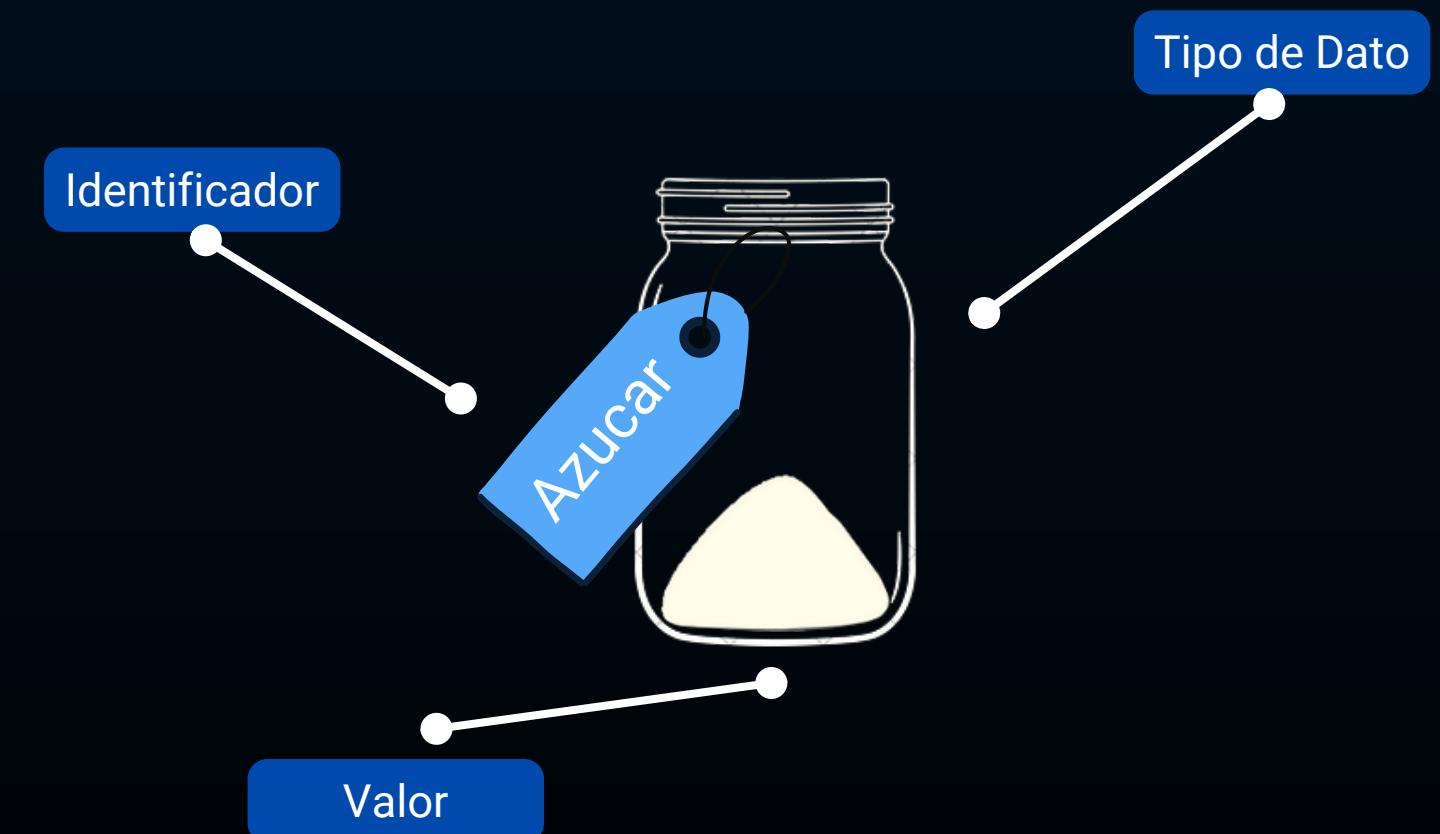
Una variable se identifica con un **tipo de dato** y un **identificador** de nombre (los elige el programador), y permite escribir un **valor** en la memoria o leer un dato de la memoria. Se puede modificar su valor las veces que sea necesario.



# Variables

Representación simbólica de espacio de memoria. Es donde se almacenan los datos en procesamiento.

Una variable se identifica con un **tipo de dato** y un **identificador** de nombre (los elige el programador), y permite escribir un **valor** en la memoria o leer un dato de la memoria. Se puede modificar su valor las veces que sea necesario.



# Variables

## Ejemplo

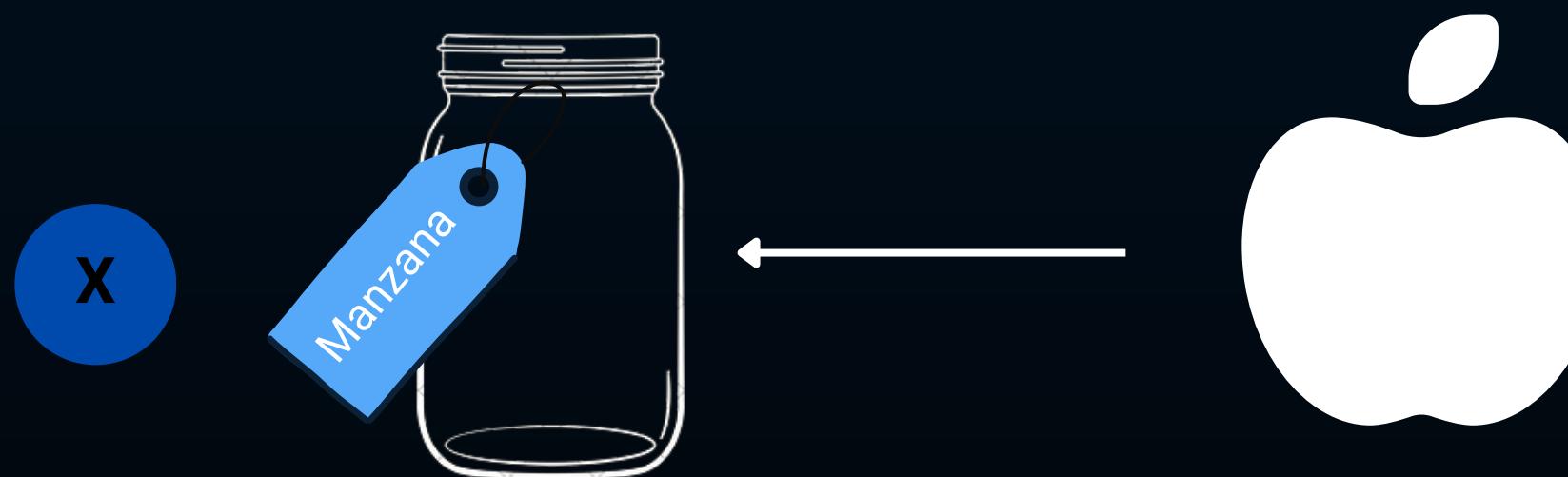
```
int edad;  
edad = 20;
```

Representación simbólica de la memoria.  
Al espacio de memoria identificado por  
edad se le asignó el valor 20.

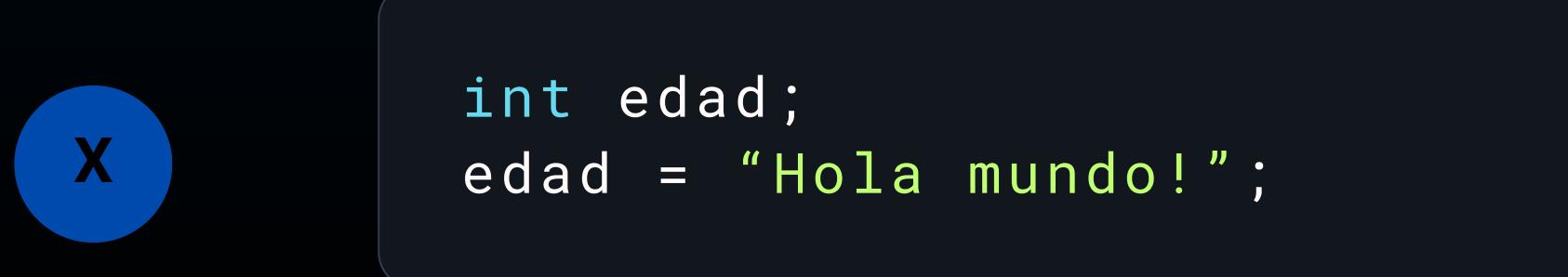


# Variables

El **tipo de dato** define la característica y el tamaño del valor que le vamos a almacenar. Los datos o valores tienen diferentes características y tamaños, por lo que tenemos que indicarle a la variable qué tipo de valor vamos a almacenar al momento de declararla.



- No podemos guardar en un contenedor pequeño algo que supere su capacidad



- No podemos asignar a una variable numérica e tipo `int` un valor de tipo `string`.

# Variables

En C++ hay varios tipos de datos. Se encuentran los llamados **Primitivos** que son propio del lenguaje y que el procesador entiende, y también están los tipos de datos **propios** que veremos más adelante.

```
int numerosEnteros = 10; //Numeros enteros, ya sea positivo o negativos
float numeroConDecimal = 10.5; //Numeros con decimales
char caracteres = 'a'; //Un solo caracter
bool valoresDeVerdad = true; //Puede ser true o false
string texto = "Hola mundo!"; //Cadena de textos
```

# Constantes

Una constante se identifica con la palabra reservada **const**, un **tipo de dato**, un **identificador** (lo elige el programador), y un **valor** que no puede ser modificado durante el transcurso del programa en ejecución.

```
const int EDAD = 20;  
const char LETRA = 'A';  
const float PRECIO = 10.5;
```



Como muchos lenguajes, C y C++ son case-sensitive. Esto significa que hace diferencias entre mayúsculas y minúsculas.

# Operadores

Conjunto de **símbolos** y **palabras reservadas** que nos permiten hacer operaciones con expresiones.

Existen diferentes categorías de **operadores**:



Aritméticos



Relacionales



Asignación



Lógicos



De dirección e indirección



De acceso a miembros



Punteros



Ternarios

# Operadores aritméticos

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la sparación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División
%	Resto

$((2+3)*5)+10$
$2+3*5+10$
$5/2$
$5.0/2$
$5\%2$

# Operadores aritméticos

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la sparación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División
%	Resto

$$((2+3)*5)+10 \longrightarrow 35$$

$$2+3*5+10 \longrightarrow 27$$

$$5/2 \longrightarrow 2$$

$$5.0/2 \longrightarrow 2.5$$

$$5\%2 \longrightarrow 1$$

# Operadores relacionales

Son necesarios para **decisiones** y **ciclos**. Nos permiten establecer proposiciones lógicas. El resultado de una proposición lógica puede ser verdadero o falso.

Si me pregunto  $5 > b$ , el resultado va a depender en función del valor de la variable  $b$ .

- Si  $b$  es igual a 6, el resultado es **falso**.
- Si  $b$  es igual a 1, el resultado es **verdadero**.

Operador	Significado
$>$	Mayor que
$<$	Menor que
$\geq$	Mayor o igual que
$\leq$	Menor o igual que
$=$	Igual que
$\neq$	Distinto que

# Operadores de asignación

Nos permite asignar la expresión que se encuentra a la derecha del operador en la variable que se encuentra a la izquierda.



```
edad = 20;  
a = b;  
nombre = "Pedro";  
precio = cant * pu;
```



```
20 = edad;  
20 = 30;
```

sin sentido

```
edad = edad;
```

# Expresiones e Instrucciones

Una **expresión** es un **conjunto** de variables, constantes, números y operadores ordenados de acuerdo a las reglas sintácticas establecidas en el lenguaje de programación.

Tienen como objetivo la construcción de **instrucciones** para la resolución del problema (o de parte del problema) planteado.

```
10  
50 + 10  
aux - 20  
'B'
```

Una **instrucción** es una unidad completa de código que realiza una acción. Puede contener una o más expresiones.

```
int a = 5;  
B = 50 + 100;  
  
if(10 > 3)  
{ cout << "Es menor"; }
```

# Palabras reservadas

Palabras que el lenguaje utiliza para identificar tipos de datos, estructuras de programación, etc. Tienen un significado especial para el lenguaje y no pueden ser utilizados como **identificadores de nombre**.

Tipos de datos	bool, int, float, char, short, void, long, double
Elementos de programación	if, else, switch, default, break, for, while, do, return, auto, struct, class, static, virtual
Operadores	new, delete, sizeof

Ejercicio

# Ejercicio en diagrama y código

# Ejercicio

Hacer un programa que permita **ingresar** dos números enteros por teclado.  
Luego **calcular e informar** la suma de ellos.



# Ejercicio

Hacer un programa que permita **ingresar** dos números enteros por teclado.  
Luego **calcular e informar** la suma de ellos.

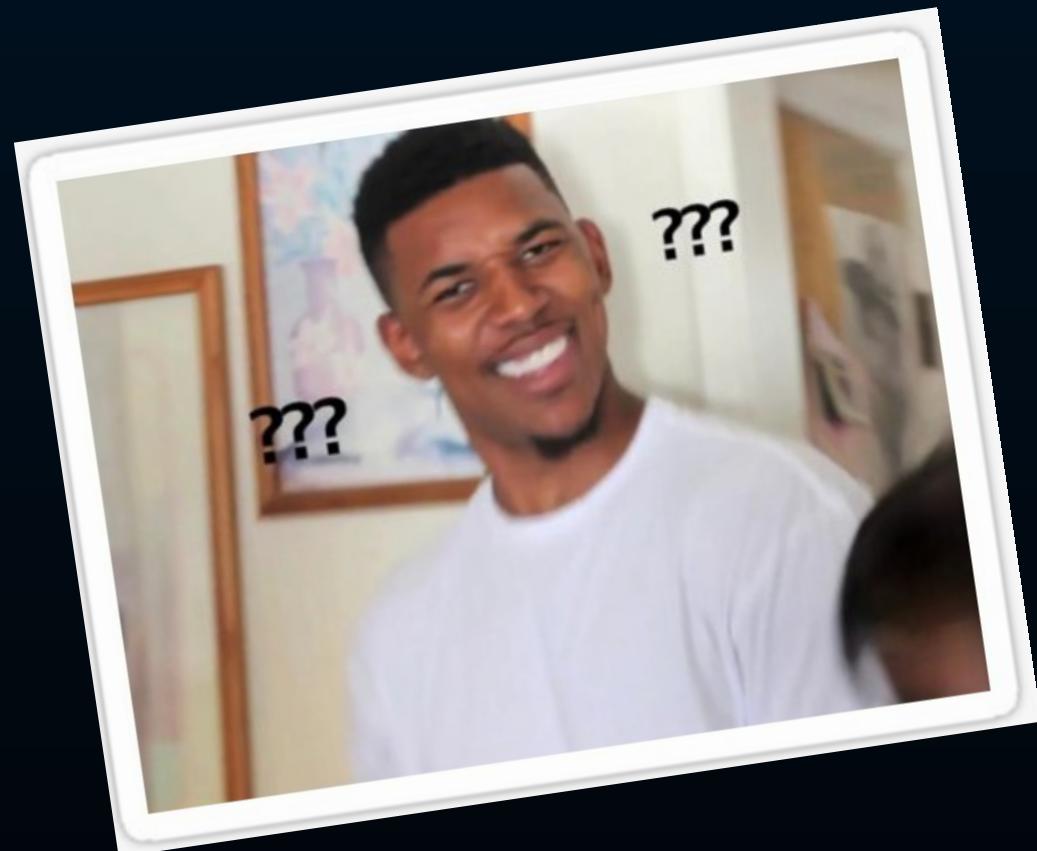
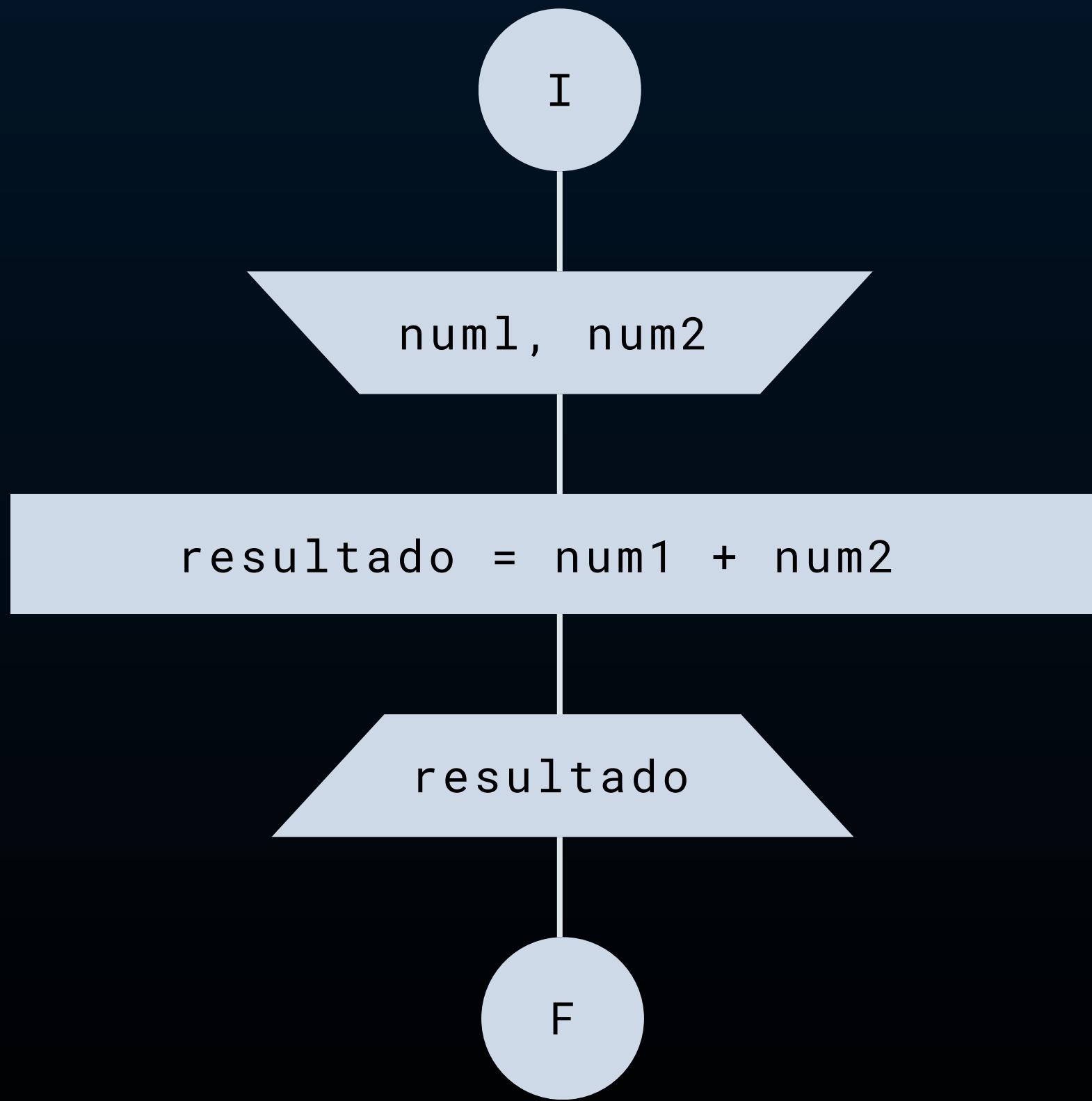


Ejercicio

# Resolució de problema

Diagrama

# Resolución



# Resolución: instrucciones

O1	DECLARAR LAS VARIABLES NO ES OBLIGATORIO	int num1, num2, resultado
O2	INGRESAR EL PRIMER NÚMERO POR TECLADO	num1
O3	INGRESAR EL SEGUNDO NÚMERO POR TECLADO	num2
O4	REALIZAR LA SUMA	resultado = num1 + num2
O5	MOSTRAR POR PANTALLA EL RESULTADO	resultado

Ejercicio

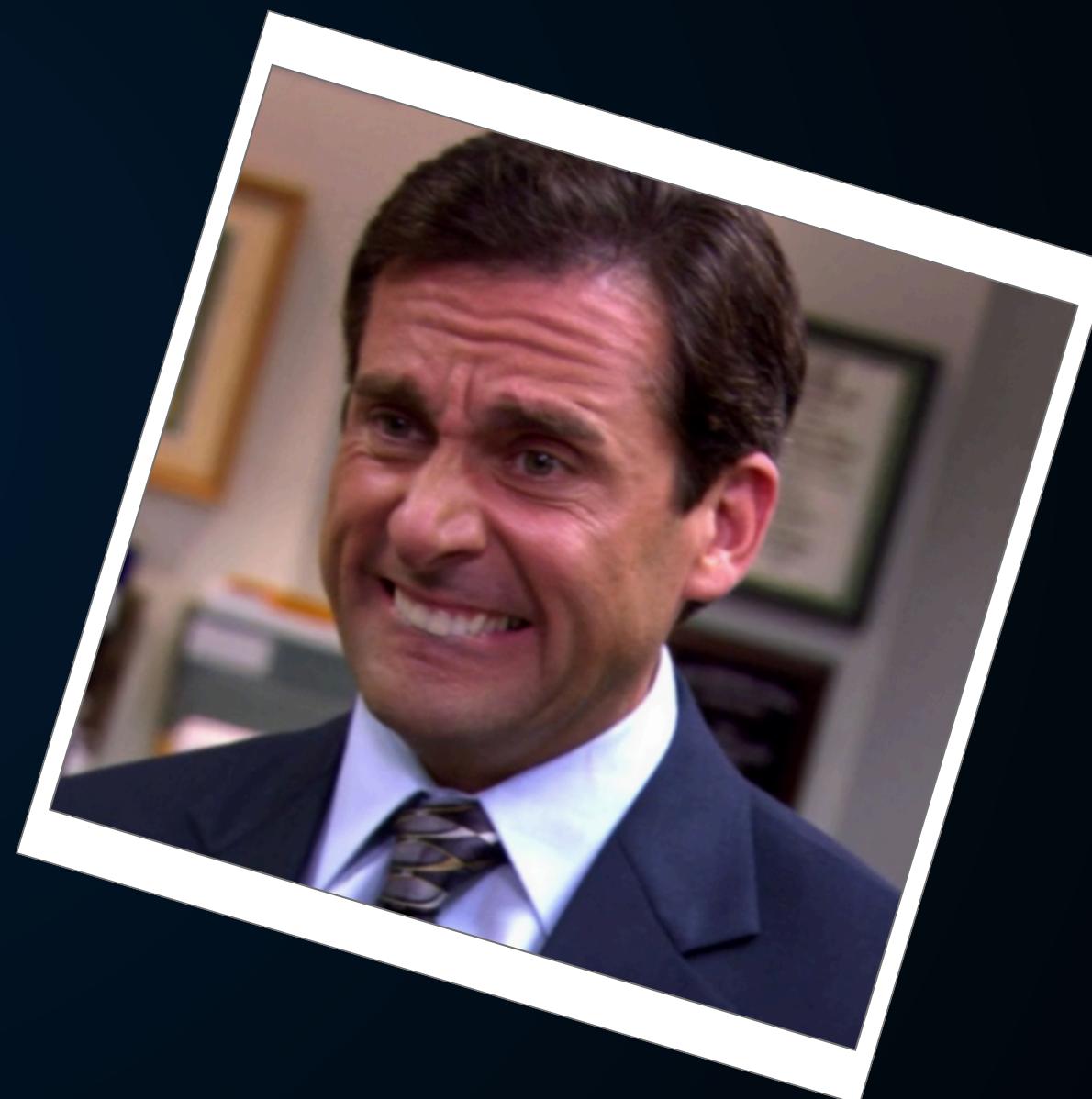
# Resolució de problema

Código

# Resolución: código fuente

```
#include <iostream>
using namespace std;

int main(){
    int num1, num2, resultado;
    cin >> num1;
    cin >> num2;
    resultado = num1 + num2;
    cout << resultado;
    return 0;
}
```



# Resolución: instrucciones

O1	DECLARAR LAS VARIABLES	int num1, num2, resultado;
O2	INGRESAR EL PRIMER NÚMERO POR TECLADO	cin >> num1;
O3	INGRESAR EL SEGUNDO NÚMERO POR TECLADO	cin >> num2;
O4	REALIZAR LA SUMA	resultado = num1 + num2;
O5	MOSTRAR POR PANTALLA EL RESULTADO	cout << resultado;