

Escuela de Código

Tallerista: Ing Misael

# PENSAMIENTO COMPUTACIONAL



GOBIERNO DE LA  
CIUDAD DE MÉXICO

AGENCIA DIGITAL DE  
INNOVACIÓN PÚBLICA

PiLARES

## TEMARIO PENSAMIENTO COMPUTACIONAL

- I. Algoritmo
  - a. Diseño de algoritmo (actividad)
- II. Operadores
  - a. Operadores aritméticos (cálculos numéricos)
  - b. Operadores relacionales
  - c. Operadores lógicos (resolución de problemas)
- III. Conceptos básicos de programación
- IV. Tipos de Datos
  - a. Datos primitivos
  - b. Numéricos
  - c. Símbolos y cadenas
  - d. Lógicos o booleanos (falso, verdadero)
- V. Variables
- VI. Diagramas
- VII. Estructuras
- VIII. Introducción al diseño del pseudocódigo

Escuela de Código

Tallerista: Ing Misael

# EL ARTE DE PROGRAMAR

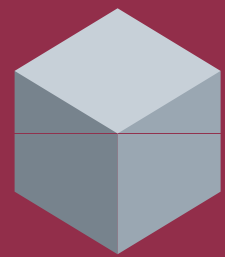


GOBIERNO DE LA  
CIUDAD DE MÉXICO

AGENCIA DIGITAL DE  
INNOVACIÓN PÚBLICA

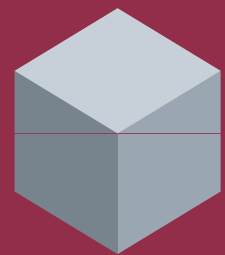
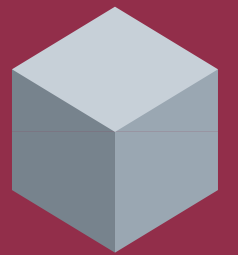
PiLARES

# I. ALGORITMO



Concepto de algoritmo

Actividad. Diseño de algoritmos

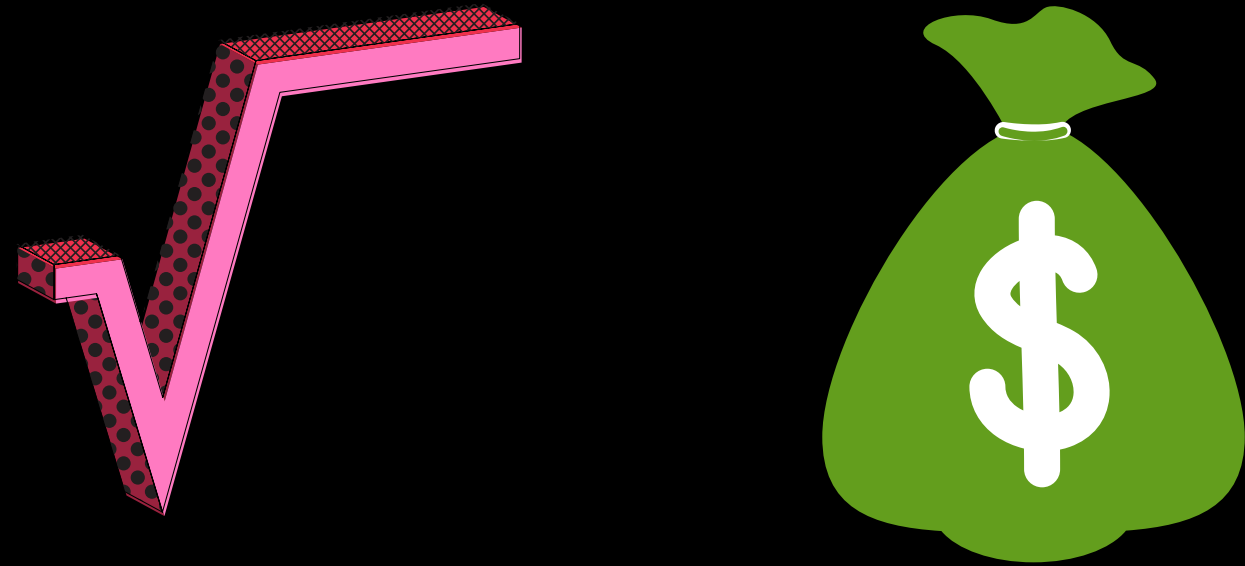


Ejercicio. Generacion de instrucciones



Serie de instrucciones detalladas finitas y escritas en un lenguaje cotidiano sobre como resolver un problema o ejecutar una acción por medio del razonamiento lógico





## CUALITATIVOS

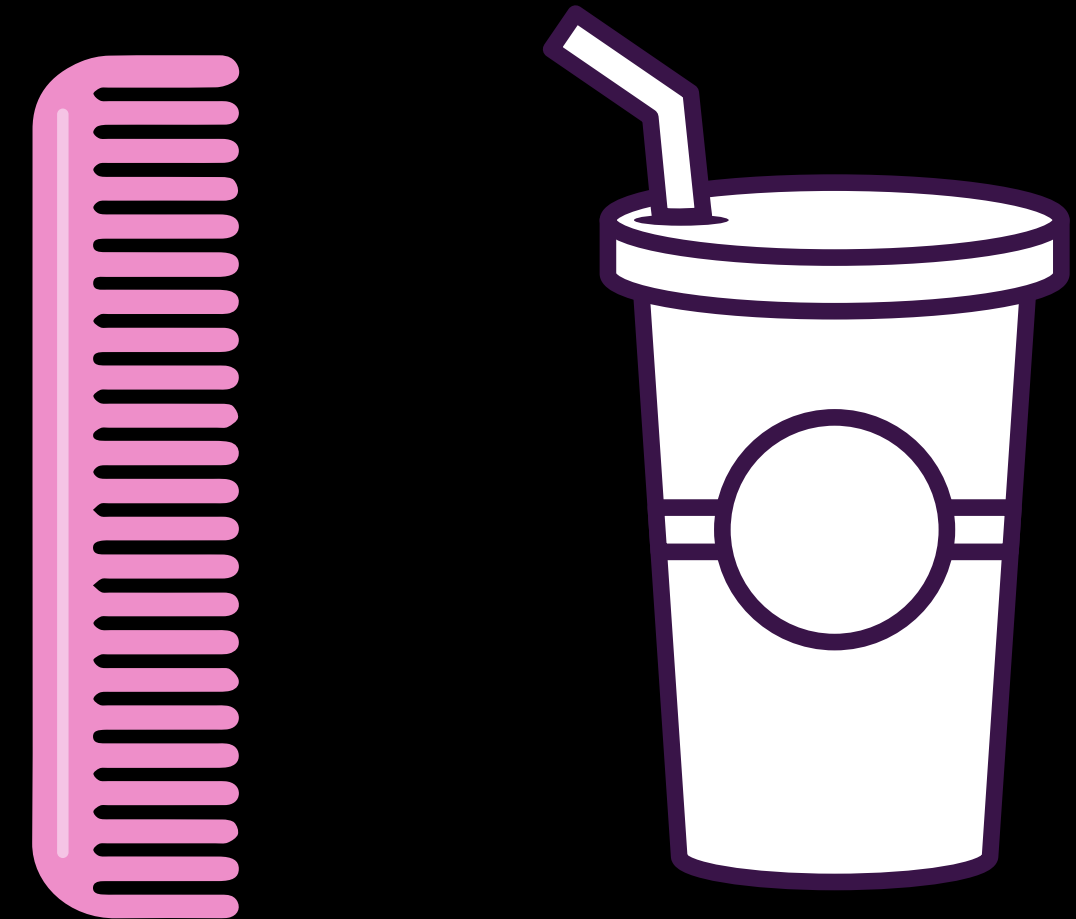
### Actividades diarias

(Preparar Bebidas, Peinarse)

## CUANTITATIVOS

### Cálculos Numéricos

(Calcular salarios o raíces cuadradas)



# Algoritmo

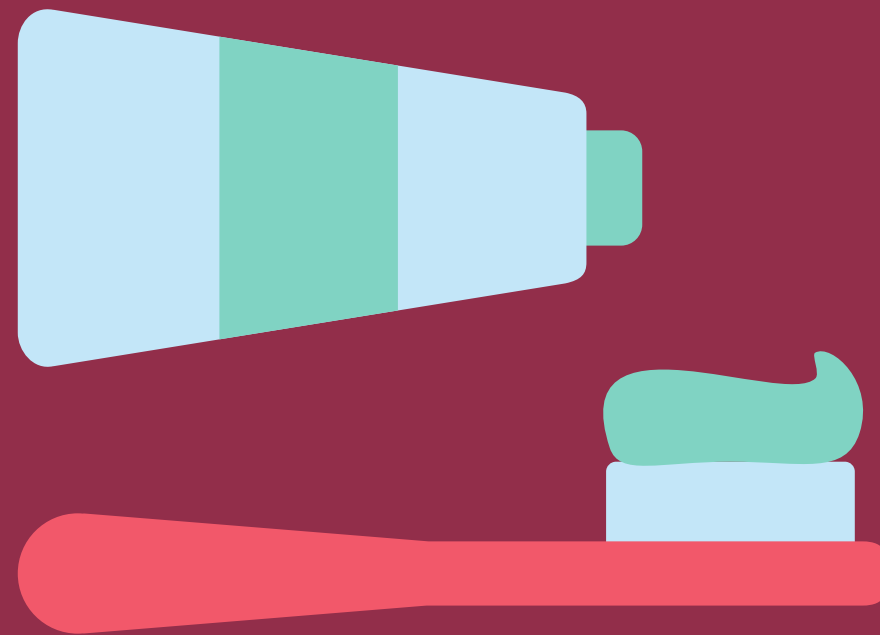
Escuela de Código

Tallerista: Ing Misael

# REDACTAR UN ALGORITMO

## ***ENTRADAS:***

LOS ELEMENTOS QUE SON NECESARIOS PARA REALIZAR UNA ACCIÓN.





## REDACTAR UN ALGORITMO

### **ACCIONES:**

LAS ACCIONES QUE SE LLEVÁRAN A CABO ESCRITAS DE MANERA SENSILLA Y ORDENADA (1...,2...,4...ETC.)

- 1.- TOMAR LA PASTA DENTAL,
- 2.- ABRIR EL PRODUCTO,
- 3.- COLOCAR LA TAPA DE LA PASTA EN ALGÚN LUGAR,
- 4.- TOMAR EL CEPILLO DE DIENTES,
- 5.- EXPRIMIR CIERTA CANTIDAD DE PASTA EN EL CEPILLO,
- 6.- DEJAR LA PASTA SOBRE ALGÚN LUGAR,
- 7.- ABRIR LA BOCA,
- 8.- COLOCAR EL CEPILLO EN LAS ENCÍAS,
- 9.- CEPILLAR LOS DIENTES, CEPILLAR LA LENGUA SUAVEMENTE,
- 10.- TOMAR EL VASO CON AGUA Y ENJUAGARSE LA BOCA,
- 11.- REPETIR EL PROCEDIMIENTO HASTA QUE LA BOCA ESTÉ LIBRE DE RESIDUOS.



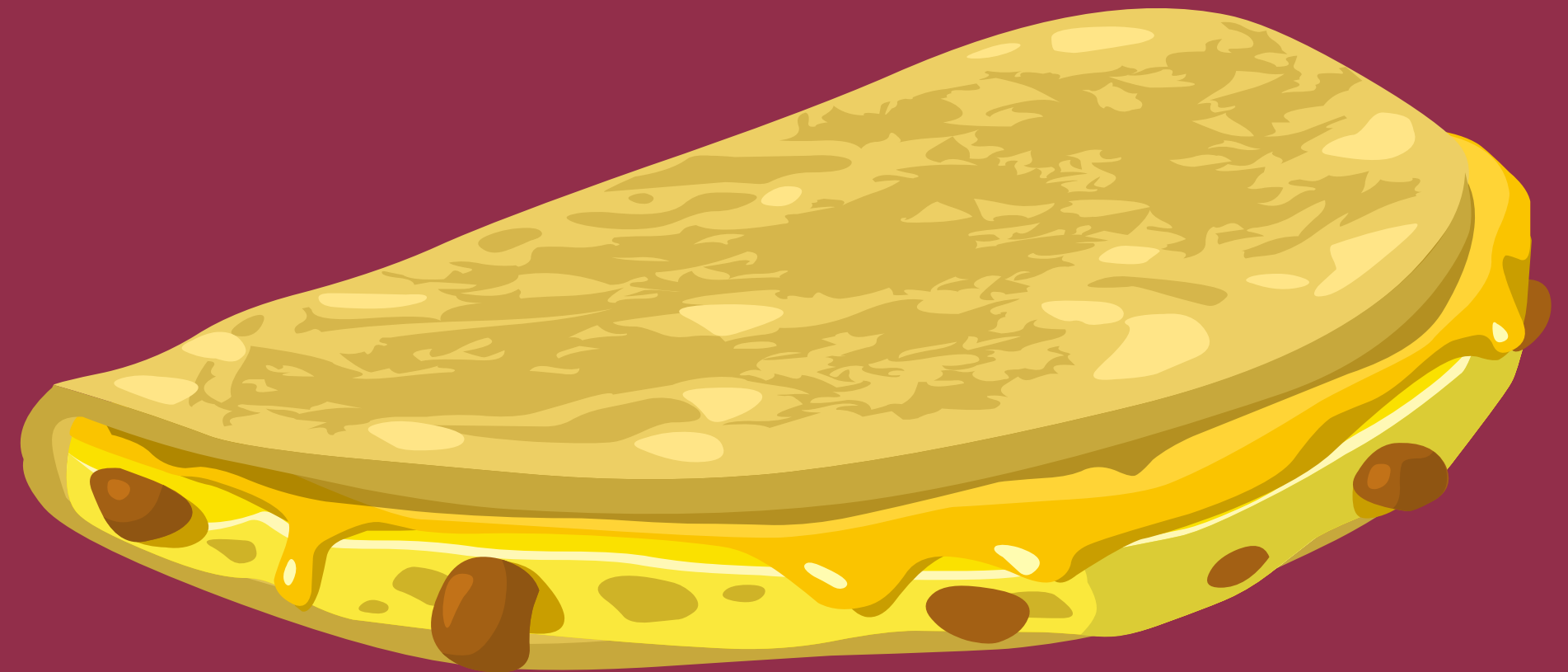
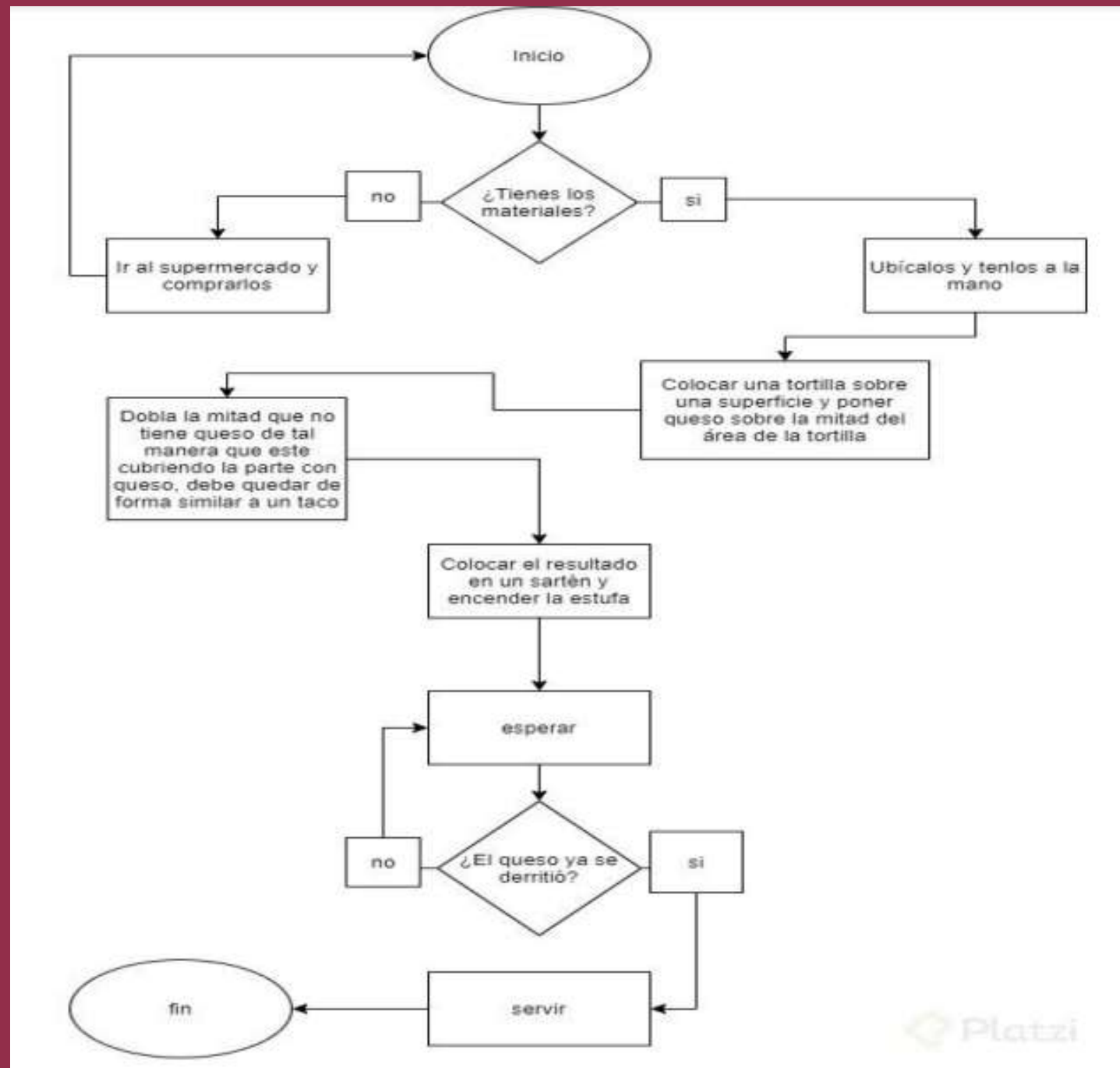
# REDACTAR UN ALGORITMO

## ***RESULTADOS:***

DETERMINAR QUÉ RESULTADOS SE PRETENDEN LLEGAR



## ALGORITMO CUALITATIVO

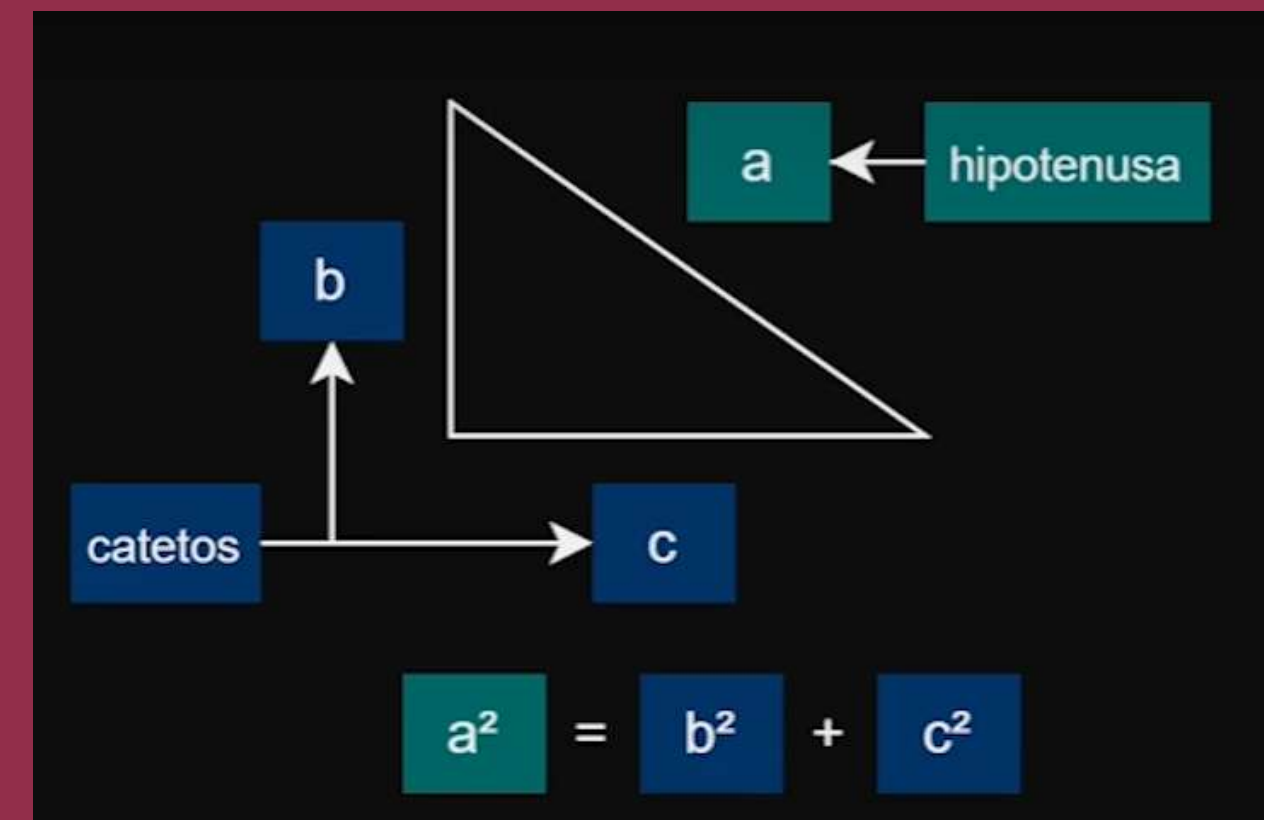


## ALGORITMO CUANTITATIVO

### PASOS

1. Identifica el valor del primer cateto  
3
2. Identifica el valor del segundo cateto  
4
3. Eleva el primer cateto al cuadrado  
 $3^2 = 9$
4. Eleva el segundo cateto al cuadrado  
 $4^2 = 16$
5. Suma el valor de cada uno de los catetos al cuadrado  
 $9 + 16 = 25$
6. Saca la raíz cuadrada de la suma de los cuadrados de los catetos  
 $\sqrt{25} = 5$
7. Da el valor de la hipotenusa  
5

Calcular el valor de la hipotenusa de un triángulo rectángulo cuyo valor de un cateto es 3 y el otro es 4



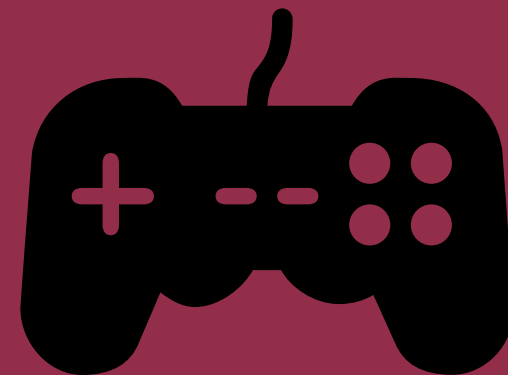


Escuela de Código

Tallerista: Ing Misael

# RETOS

1.- DISEÑA UN ALGORITMO SOBRE ALGUNA ACTIVIDAD DE TU VIDA  
COTIDIANA COMO PREPARAR UNA RECETA DE COCINA



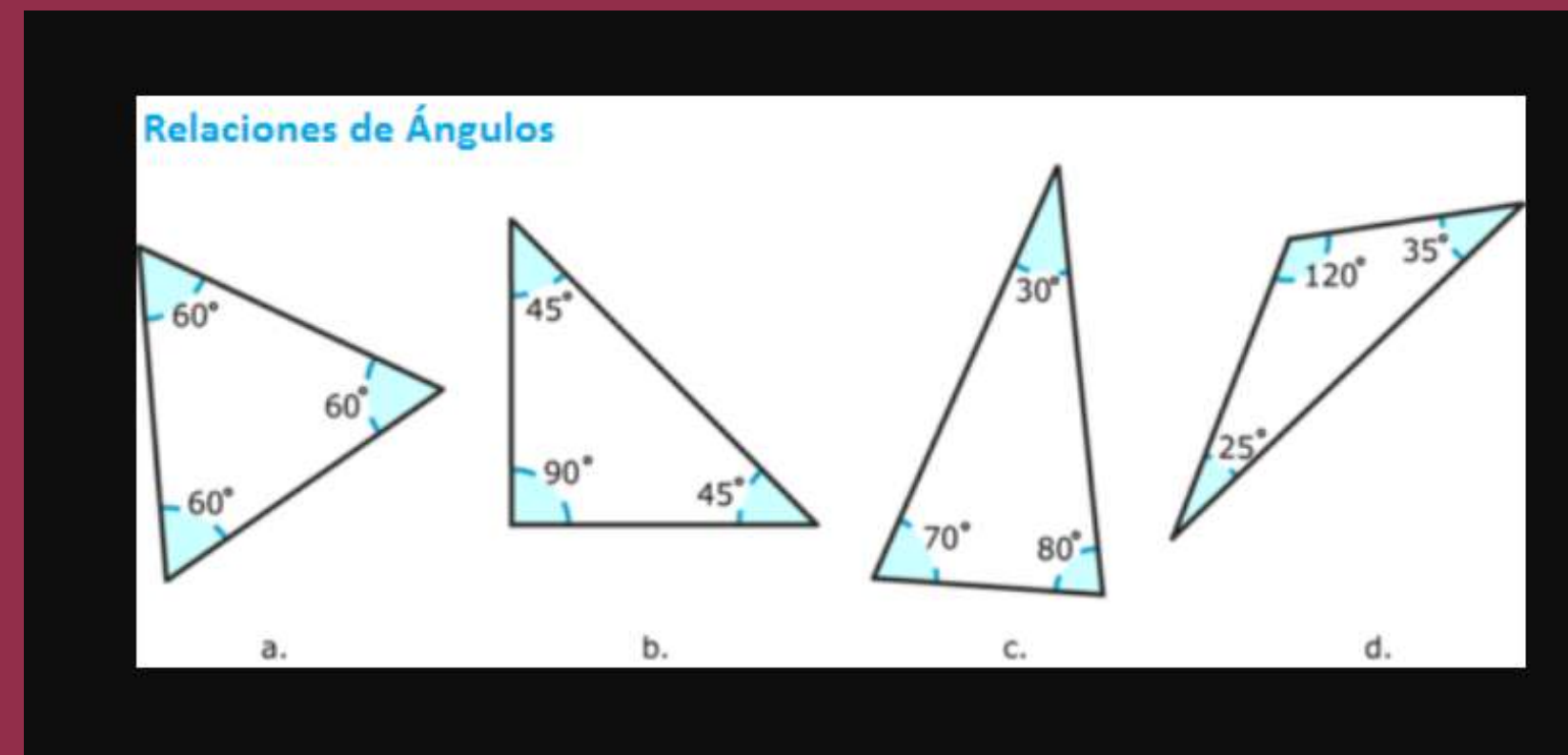
GOBIERNO DE LA  
CIUDAD DE MÉXICO

AGENCIA DIGITAL DE  
INNOVACIÓN PÚBLICA

PILARES

# RETOS

2. DISEÑA UN ALGORITMO PARA HALLAR EL ÁNGULO INTERIOR DE UN TRIANGULO DADO LOS ÁNGULOS DE LOS OTROS DOS ÁNGULOS INTERIORES DEL MISMO TRIANGULO



Escuela de Código

Tallerista: Ing Misael

# RETOS

3. DESARROLLA UN ALGORITMO QUE CALCULE LA EDAD DE UNA  
PERSONA CON BASE EN LA OBTENCIÓN DE SU FECHA DE  
NACIMIENTO



GOBIERNO DE LA  
CIUDAD DE MÉXICO

AGENCIA DIGITAL DE  
INNOVACIÓN PÚBLICA

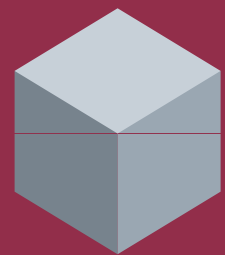
PILARES



Escuela de Código

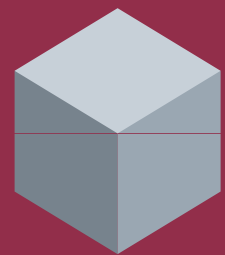
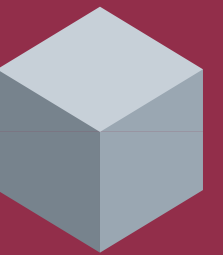
Tallerista: Ing Misael

# I. OPERADORES (parte 1)



Operadores aritmeticos

Actividad.Cálculos numéricos



Operadores Relacionales

# Operadores Aritméticos

Permiten realizar operaciones matemáticas

Jerarquía	Operador		Tipo	Entrada	Salida
0	()		Paréntesis	-	-
1	^	**	Potencias	3 ** 2	9
2	*		Multiplicación	4 * 2	8
	/		División	10 / 5	2
3	%	MOD	Residuo o Modulo	5 / 2	1
4	+		Suma	5 + 4	9
	-		Resta	10 - 5	5

Escuela de Código

Tallerista: Ing Misael

2  
2 | 5  
1



GOBIERNO DE LA  
CIUDAD DE MÉXICO

AGENCIA DIGITAL DE  
INNOVACIÓN PÚBLICA

PiLARES

# EJEMPLO

$$X = (3^2 + 10/2) + (3*9 \text{ MOD } 4 - 1)$$

$$X = (3^2 + 10/2) + (3*9 \text{ MOD } 4 - 1)$$

$$X = (3^2 + 10/2) + (3*9 \text{ MOD } 4 - 1)$$

$$X = (9 + 5) + (27 \text{ MOD } 4 - 1)$$

$$X = (14) + (3 - 1)$$

$$X = 14 + 2$$

$$X = 16$$

$$X = (18/9 \text{ MOD } 2 + 16) - (5*4 - 3^3)$$

$$X = (5 + 2*4) - 25 \text{ MOD } 5$$

$$X = (100 / 5^2 + 1) + 11 * 3$$

$$X = ((8-6) ^ {2*3 \text{ mod } 3}) ^ 3$$

# RETOS (Respuestas)

$$W = (18/9 \text{ MOD } 2 + 16) - (5*4 - 3^3) = 23$$

$$X = (5 + 2*4) - 25 \text{ MOD } 5 = 13$$

$$Y = (100 / 5^2 + 1) + 11 * 3 = 38$$

$$Z = ((8-6) ^ {2*3 \text{ mod } 3}) ^ 3 = 0$$



# Operadores Relacionales

Permiten comparar dos o mas valores determinando si el resultado final es verdadero o falso

Operador		Tipo	Entrada	Salida
<		Menor que	5 < 10	V
>		Mayor que	15 > 1	V
<=		Menor o igual que	4 <= 4	V
>=		Mayor o igual que	6 >= 5	V
=	==	Igualdad	5 == (4+2)	F
<>	!=	Diferente a	6 != 7	V

# EJEMPLO

Variable	Operador	Valor a comparar	Mensaje de Salida
Promedio del Alumno	<	6	Repite el año
Promedio del Alumno	>=	9	Recibe reconocimiento

# RETOS

Una empresa de logistica ofrece a sus trabajadores un bono de puntualidad. Si el empleado siempre llega puntual durante el mes, se le notifica que ha sido acreedor al bono de puntualidad. Si el empleado tiene 2 retardos se le suspende un día y si tiene 3 o más reatardos, se le da de baja al final del mes. Describe tal situación usando operadores relacionales

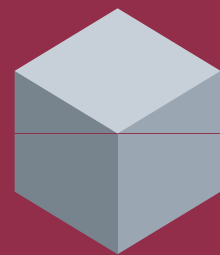
# RETOS

Analiza el problema y responde la opción que consideres que lo resuelve:

"C" es mayor que "D". "E" es menor que "F". "G" es menor que "E" y "D" es mayor que "F". ¿Cuál es el menor de todos?

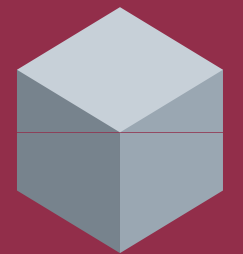
- a)G
- b)C
- c)E
- d)D
- e)F

# II. OPERADORES (parte 2)



Operadores Lógicos

Actividad. Resolución de problemas lógicos





# Operadores Lógicos

**Evalúan dos o más expresiones con operadores relacionales para determinar si en general es verdadero o falsa**

Operador	Traducción	Nombre	Descripción
&&	Y	Conjunción AND	Si todas las condiciones son verdaderas, todo es verdadero
	O	Disyunción OR	Si al menos una condición es verdadera, todo es verdadero
!	~	NO	Negación NOT
			Cambia el valor final de la expresión por su contrario, es decir, si todo es verdadero, lo hace falso y viceversa.



# EJEMPLO

$$X = ( 45 < 120 \text{ OR } 12 < 120 )$$
$$X = ( 45 < 120 \text{ OR } 12 < 120 )$$
$$X = ( V \text{ OR } V )$$
$$X = V$$

# RETOS

```
( 45<120 OR 12<120 )  
( 6!=6 ) && ( 12>22 )  
~ ( 128<145 && 12>9 )  
"Daniela" <> "DANIELA"  
10*20 <> 210
```

# RETOS(Respuestas)

$(45 < 120 \text{ OR } 12 < 120) = V$

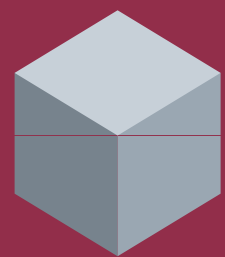
$(6 \neq 6) \&\& (12 > 22) = F$

$\sim (128 < 145 \&\& 12 > 9) = F$

$"Daniela" <> "DANIELA" = T$

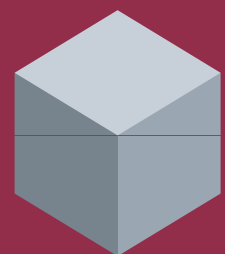
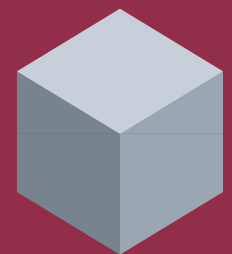
$10 * 20 <> 210 = T$

# III. Conceptos basicos de programación



Presentación teorica de conceptos base

Programa



Lenguaje de programación

# Escuela de Código



Es un idioma artificial **creado para indicarle a la Computadora lo que debe hacer. Tiene ciertas reglas de escritura (sintaxis) en las que utiliza simbolos y palabras clave, ademas de una semántica (Interpretación interna).**



Tallerista: Ing Misael

SINTAXIS

main()

SEMANTICA

“main()” indica el punto de entrada, inicio, de un programa

# PROGRAMA

ES UN BLOQUE DE INSTRUCCIONES  
(CÓDIGO FUENTE) ESCRITAS EN CIERTO  
LENGUAJE DE  
PROGRAMACIÓN CUYO PROPOSITO ES  
RESOLVER UN PROBLEMA

```
#include <stdio.h>
#include <math.h>

int main(){

    float cat1, cat2, hipo=0;

    printf("Digite el Cateto 1: ");
    scanf("%f", &cat1);

    printf("Digite el Cateto 2: ");
    scanf("%f", &cat2);

    hipo = sqrt( pow(cat1, 2) + pow(cat2, 2) );

    printf("La hipotenusa es: %f", hipo);

    return 0;
}
```



# EJECUTAR UN PROGRAMA

*"La computadora realiza una traducción de sus componentes al lenguaje máquina, es decir, convierte las instrucciones en cadenas de ceros y unos. "*



# INTERPRETE

## ¿Qué es un intérprete?

Un intérprete es un **programa informático** que procesa el código fuente de un proyecto de software durante su tiempo de ejecución, es decir, mientras el software se está ejecutando, y actúa como una **interfaz** entre ese proyecto y el procesador. Un intérprete siempre procesa el código línea por línea, de modo que lee, analiza y prepara cada secuencia de forma consecutiva para el procesador. Este principio también se aplica a las secuencias recurrentes, que se ejecutan de nuevo cada vez que vuelven a aparecer en el código. Para procesar el código fuente del software, el intérprete recurre a sus propias bibliotecas internas: en cuanto una **línea de código fuente se ha traducido a los correspondientes comandos legibles por máquina**, esta se envía directamente al procesador.

# COMPILADOR

## ¿Qué es un compilador?

Un compilador es un **programa informático** que traduce todo el código fuente de un proyecto de software a código máquina antes de ejecutarlo. Solo entonces el procesador ejecuta el software, obteniendo todas las instrucciones en código máquina antes de comenzar. De esta manera, el procesador cuenta con todos los componentes necesarios para ejecutar el software, procesar las entradas y generar los resultados. No obstante, en muchos casos, durante el proceso de compilación tiene lugar un paso intermedio fundamental: antes de generar la traducción final en código máquina, la mayoría de los compiladores suelen convertir el código fuente en un **código intermedio** (también llamado código objeto) que, a menudo, es compatible con diversas plataformas y que, además, también puede ser utilizado por un intérprete.

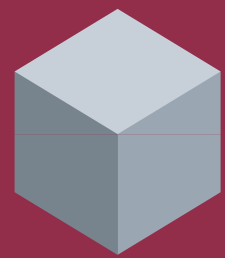
# COMPILADOR

	Intérprete	Compilador
Momento en que se traduce el código fuente	Durante el tiempo de ejecución del software	Antes de ejecutar el software
Procedimiento de traducción	Línea por línea	Siempre todo el código
Presentación de errores de código	Después de cada línea	En conjunto, después de toda la compilación
Velocidad de traducción	Alta	Baja
Eficiencia de traducción	Baja	Alta
Coste de desarrollo	Bajo	Alto
Lenguajes típicos	PHP, Perl, Python, Ruby, BASIC	C, C++, Pascal

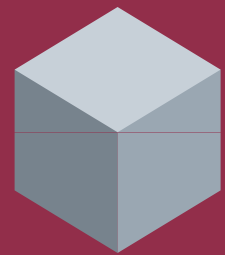
	Ventaja	Inconveniente
Intérprete	Proceso de desarrollo sencillo (sobre todo en términos de depuración)	Proceso de traducción poco eficiente y velocidad de ejecución lenta
Compilador	Proporciona al procesador el código máquina completo y listo para ejecutar	Cualquier modificación del código (resolución de errores, desarrollo del software, etc.) requiere volverlo a traducir



# IV. TIPOS DE DATOS

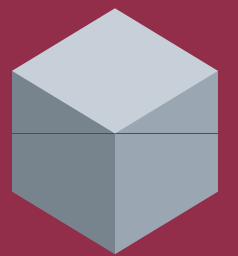


Datos primitivos

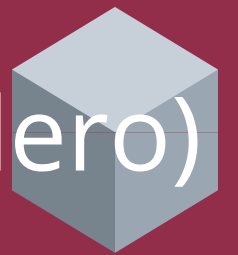


Símbolos y cadenas

Numéricos



Logicos o booleanos (Falso, Verdadero)



**"ROJO"**

¿Representa "Alto" o "Sangre de los Héroes"?

Es la representación de la  
la realidad en su estado más  
simple, por si mismo no tiene  
capacidad de comunicar algo



# DATOS PRIMITIVOS

**No pueden descomponerse en datos más simples**

## NUMÉRICOS

Enteros (2,4) y Flotantes (1.4 , 2.5)

Edad de alguien, sueldo ,km recorridos por una moto

## CÁRACTER

Cualquier simbolo de en una computadora , (A % .\* | )

Marcar F para femenino y M para masculino

## LÓGICOS O BOLEANOS

Valores que resultan de comparar  
(Verdadero / Falso )

"La edad de Elvira es mayor que la edad de Pedro"  
=  
Falso

# DATOS COMPUESTOS

Se forman de datos simples

## STRINGS

Cadenas conjuntos de caracteres





# DATOS LOGICOS O BOLEANOS

Estan formados por dos valores que son falso y verdadero

Por ejemplo, si **a** = 5 y **b** = 8, obtendríamos:

<b>a == b</b> : false	Pregunta si <b>a</b> es idéntico a <b>b</b> : la salida es falso
<b>a != b</b> : true	Pregunta si <b>a</b> es diferente a <b>b</b> : la salida es verdadero
<b>a &lt; b</b> : true	Pregunta si <b>a</b> es menor a <b>b</b> : la salida es verdadero
<b>a &gt; b</b> : false	Pregunta si <b>a</b> es mayor a <b>b</b> : la salida es falso
<b>a &lt;= b</b> : true	Pregunta si <b>a</b> es menor o igual a <b>b</b> : la salida es verdadero
<b>a &gt;= b</b> : false	Pregunta si <b>a</b> es mayor o igual a <b>b</b> : la salida es falso

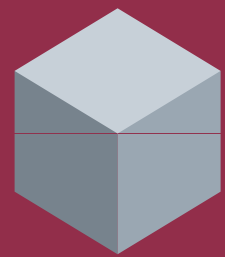


# POR LO TANTO

La siguiente tabla resume los tipos de datos:

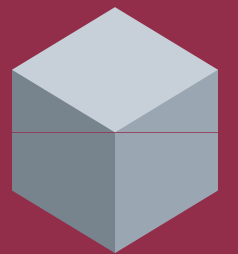
Tipo de dato	Uso	Ejemplo
Números enteros (int)	Úsalo para almacenar valores numéricos	int edad=18
Números decimales (float)	Utilízalo para almacenar valores numéricos que tengan decimales	float gravedad=9,81
Carácter (char)	Úsalo para almacenar una letra o su equivalente en código ASCII	char respuesta='S'; char respuesta='83';
Cadena de caracteres (string)	Utilízalo para almacenar mensajes.	string Nombre= "Mi nombre es Daniela"; string Alumno= "Soy el alumno 392 en la Escuela de Códigos";

# V. VARIABLES



Ejemplo de variables

Actividad : Identificacion de variables



# ACTIVIDAD

- 1.- Cierra los ojos
- 2.- Piensa en el Número 5
- 3.- Sin dejar de pensar en el 5, piensa en el 2
- 4.- Suma 1 al primer número en el que pensaste
- 5.- Suma el resultado al segundo número en el que pensaste
- 6.- ¿El resultado es 8?

**Este proceso que tu hiciste con la mente y memoria, la computadora lo realiza con variable.**

# ACTIVIDAD



```
public class Variables {  
    public static void main(String[] args) {  
  
        int num1 = 5;  
        int num2 = 2;  
        int resultado = num1 + 1;  
        int resultado2 = resultado + num2;  
        System.out.println(resultado2);  
  
    }  
}
```

# REGLAS PARA FORMAR UN IDENTIFICADOR

Representan los datos de un programa

- Comenzar con una letra  
(algunos lenguajes aceptan \_ y \$)
- Pueden contener letras, números y caracteres
- No deben ser palabras reservadas (aquellas que usan el lenguaje para funcionar)

```
public class Pruebas {  
    public static void main(String[] args) {  
  
        String listaReproduccion = "Mi mix";  
        String ape_paterno = "Hernandez";  
        int num1 = 20;  
        boolean _esMujer = true;  
  
    }  
}
```



# REGLAS PARA FORMAR UN IDENTIFICADOR

## CONSTANTES

No cambian su valor durante la ejecución del programa

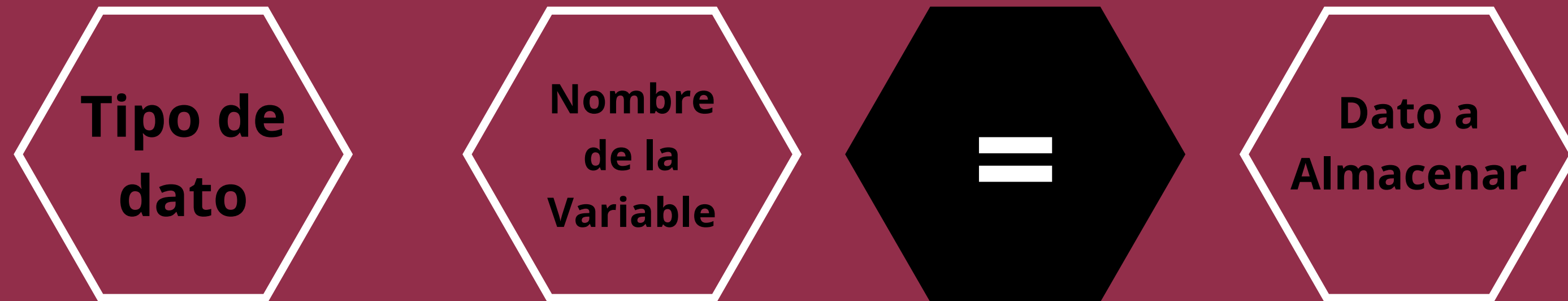
## VARIABLES

Se almacena temporalmente y puede cambiar durante la ejecución del programa

```
const FECHA_DE_NACIMIENTO = "30-05-1992"  
const NOMBRES = "Misael"  
const PI = 3.1416  
  
let edad = 30  
let estaDisponible = true  
let num1 = 20  
let resultado = num1 + 2
```

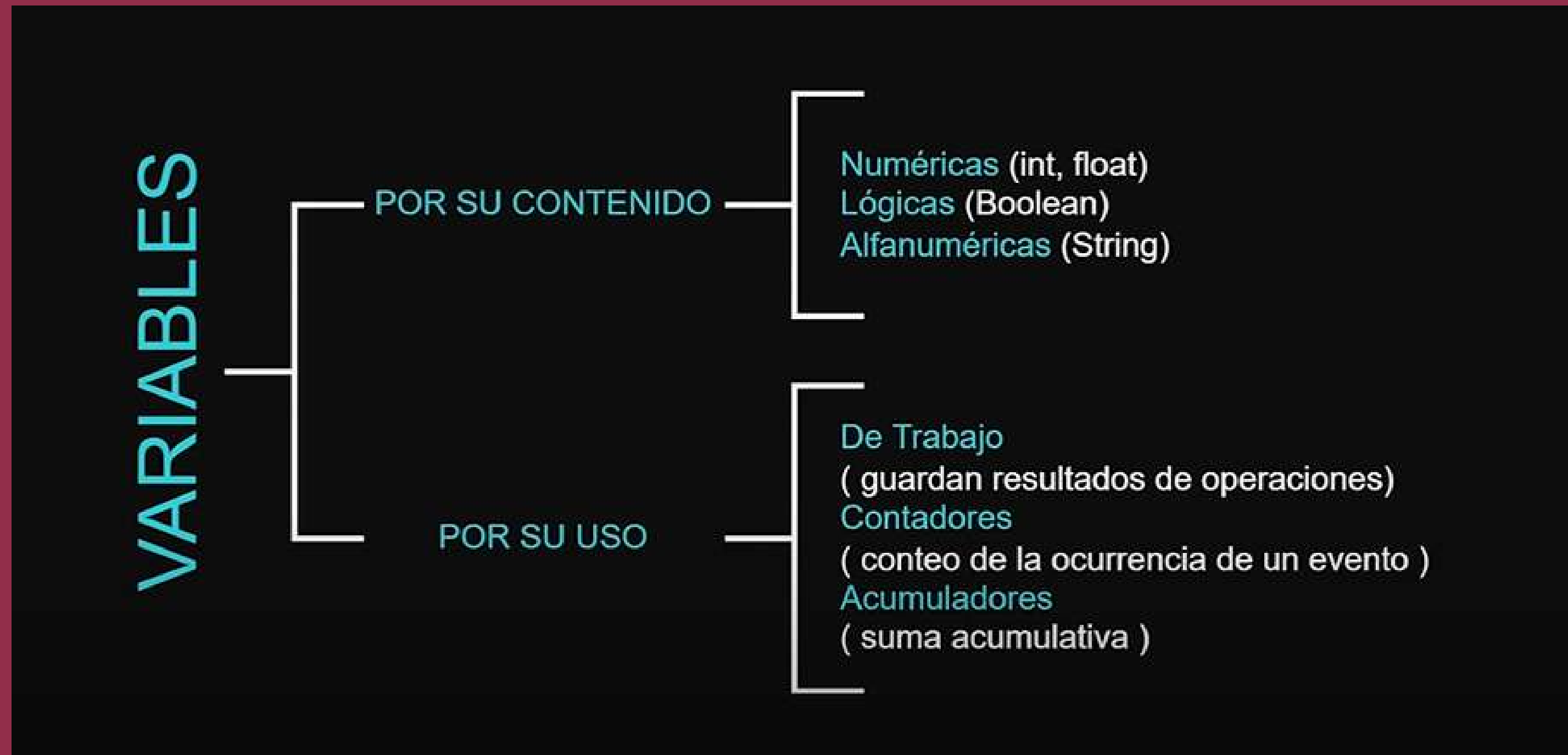
# REGLAS PARA FORMAR UN IDENTIFICADOR

No todos los lenguajes de programación son tipados (incluyen tipos de datos)



```
String cancion = "Save the World"  
int precio = 240
```

# CLASIFICACIÓN DE VARIABLES



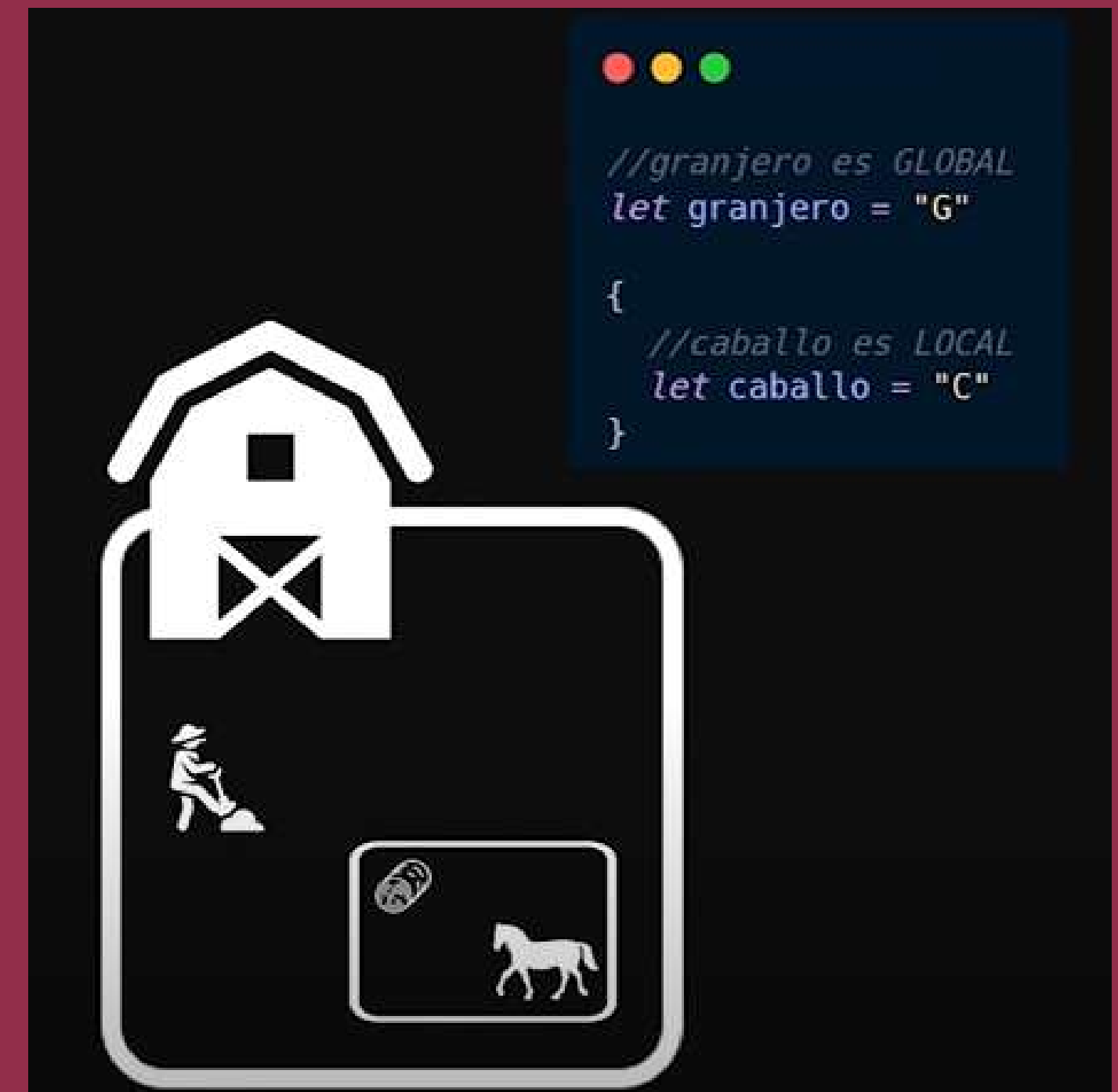
# CLASIFICACIÓN POR ALCANCE

## GLOBALES

Se declaran al comienzo de un programa y existen en todo el programa

## LOCALES

Se declaran dentro de un bloque de instrucciones y solo existen en ese bloque



¿Qué alcance tiene la casa?

¿Qué alcance tiene el heno?

¿Qué alcance tiene la pala?

¿Qué tipo de dato debe tener una variable para representar la calificación promedio de un curso?

¿Que tipo de dato debe tener una variable para representar el número de personas en un hogar?

¿Que tipo de dato debe tener una variable para contener el nombre de pila de una persona?

¿Qué tipo de dato debe tener una variable para registrar si esta lloviendo o no?

¿Que tipo de dato debe tener una variable para representar la cantidad de dinero que tienes?



# RETOS (respuestas)

¿Qué tipo de dato debe tener una variable para representar la calificación promedio de un curso

**Numérica (Flotante)**

¿Que tipo de dato debe tener una variable para representar el número de personas en un hogar?

**Numérica (Entero)**

¿Que tipo de dato debe tener una variable para contener el nombre de pila de una persona?

**Alfanumérica (String)**

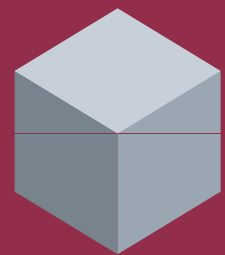
¿Qué tipo de dato debe tener una variable para registrar si esta lloviendo o no?

**Lógica (Booleana)**

¿Que tipo de dato debe tener una variable para representar la cantidad de dinero que tienes?

**Númerica (Flotante)**

# VI. DIAGRAMAS



Estructura de diagramas

Actividad: Ejercicios representación  
de algoritmos mediante diagrama



# ¿PARA QUE HACER UN DIAGRAMA?

*Analizar un problema y construir algoritmos para solucionarlo en ocasiones es abrumador y necesitamos pasarlos a la realidad para visualizarlos, encontrar errores, omisiones y en algunos casos, pedir ayuda a otras personas, la herramienta que nos ayudará son los DIAGRAMAS*

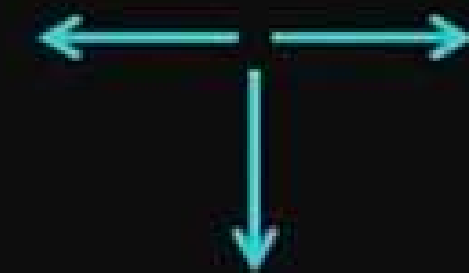
# DIAGRAMAS DE FLUJO

## CARACTERISTICAS

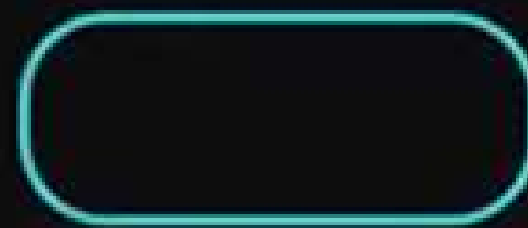
*Herramienta para representar gráficamente un algoritmo a través de símbolos que corresponden a las diferentes estructuras de control*

- Flujo de arriba hacia abajo y de izquierda a derecha
- Existe un único inicio
- Existe uno o más puntos de fin
- Solo usa líneas horizontales y verticales
- Evita el cruce de líneas
- No hay líneas de flujo sin conectar
- El lenguaje es conciso y claro

# DIAGRAMAS DE FLUJO



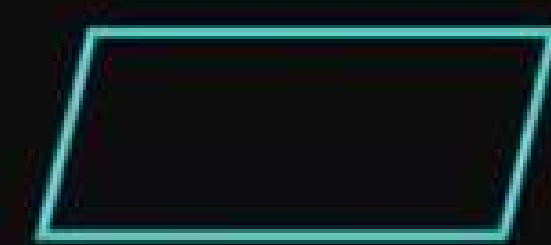
Flujo de datos



Inicio / Fin



Proceso  
Asignaciones  
Operaciones



Entrada /  
Salida



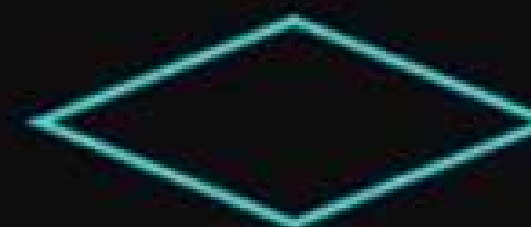
Conector

Continuidad  
en la otra  
página

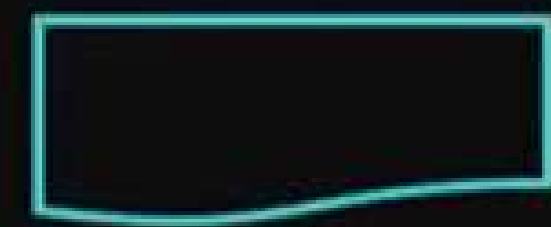
Continuidad  
en la misma  
página



Decisión  
Múltiple

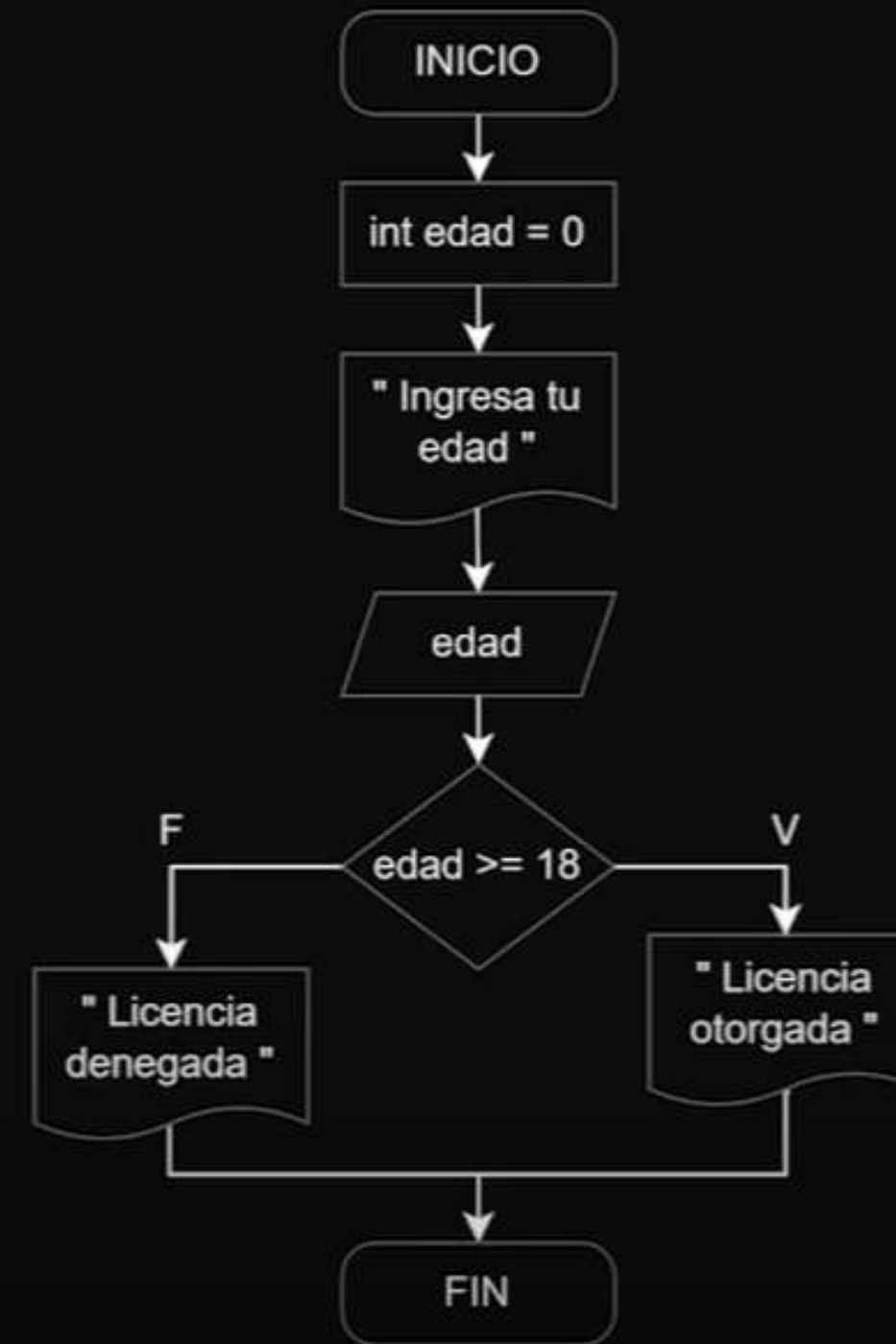


Decisión



Impresión

# DIAGRAMAS DE FLUJO PARA OTORGAR LICENCIAS A MAYORES DE EDAD





# RETOS

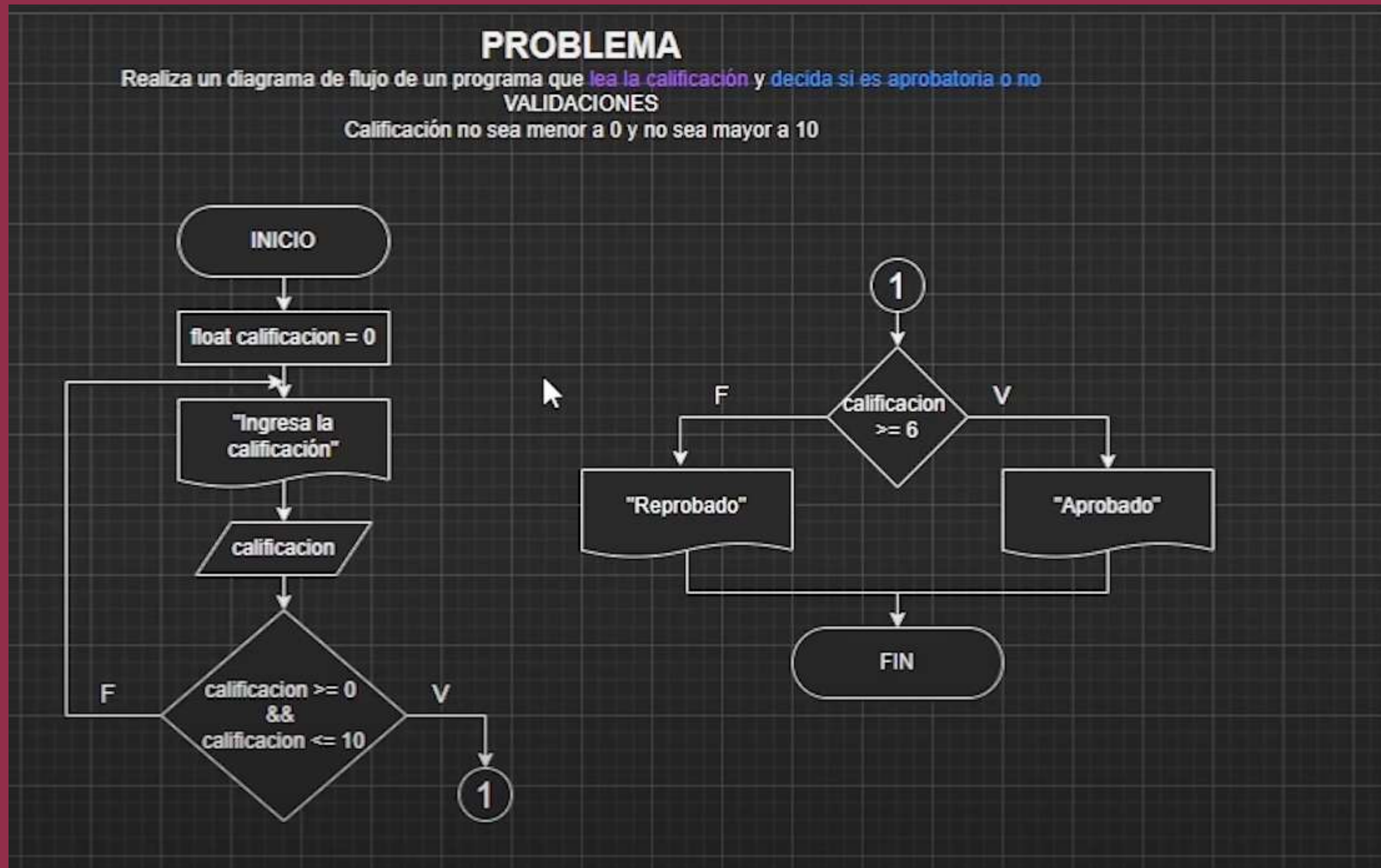
**Realiza un diagrama de flujo de un programa que lea la calificación y decida si es aprobatoria o no**

(Puedes utilizar herramientas como draw.io)

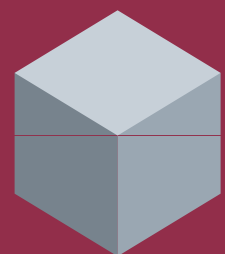
# RETOS

## Agregar validaciones al ejercicio anterior

Por ejemplo, valida que la calificación no sea menor a 0)



# VII. ESTRUCTURAS



Ejercicios: Diagramas de flujo, estructuras condicionales e iterativas

# ESTRUCTURAS DE CONTROL

"Controlan el flujo del programa.

Sin ellos el programa se ejecutaría línea a línea de principio a fin sin la posibilidad de tomar decisiones"

## CONDICIONALES

Permiten tomar decisiones

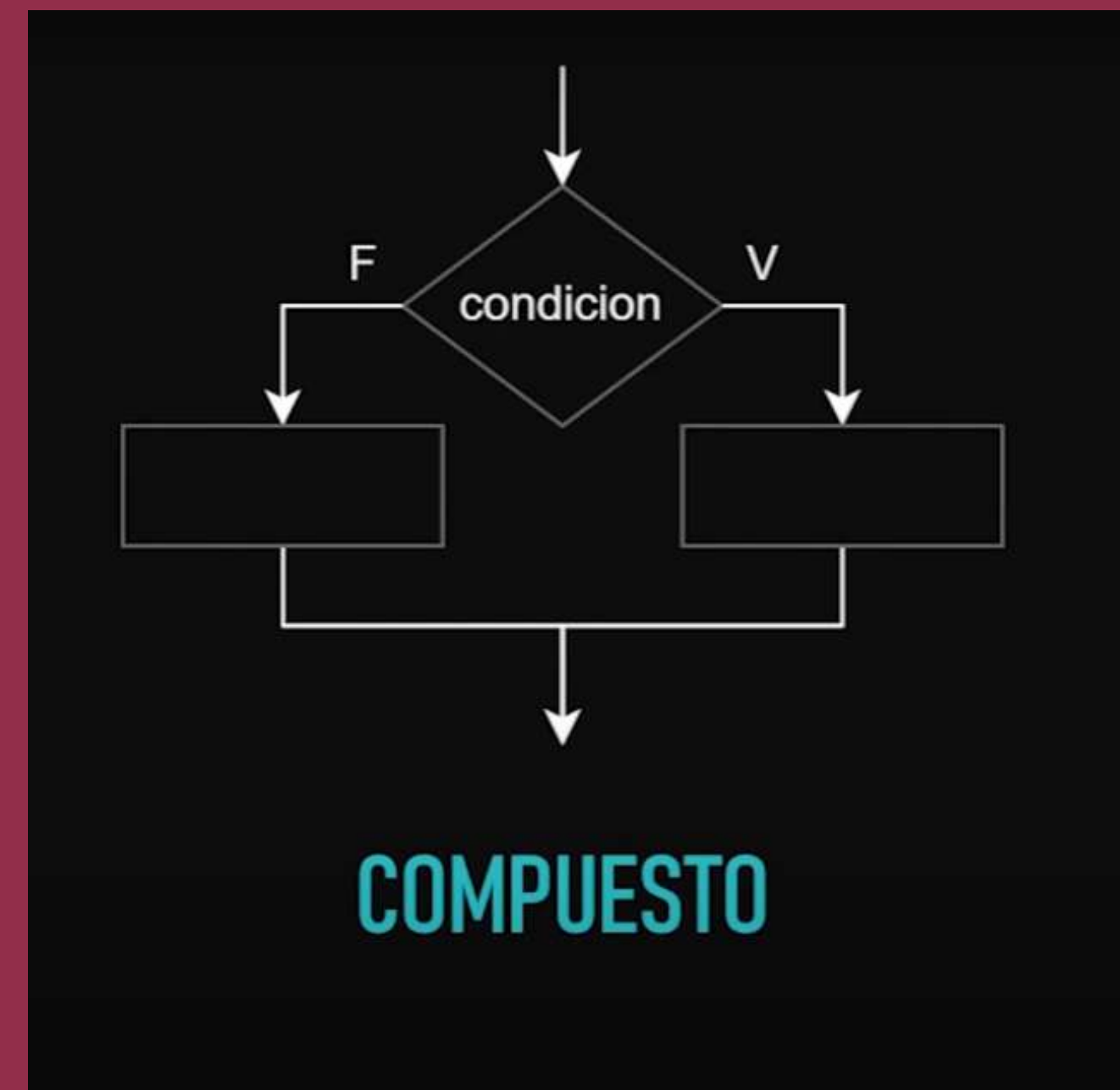
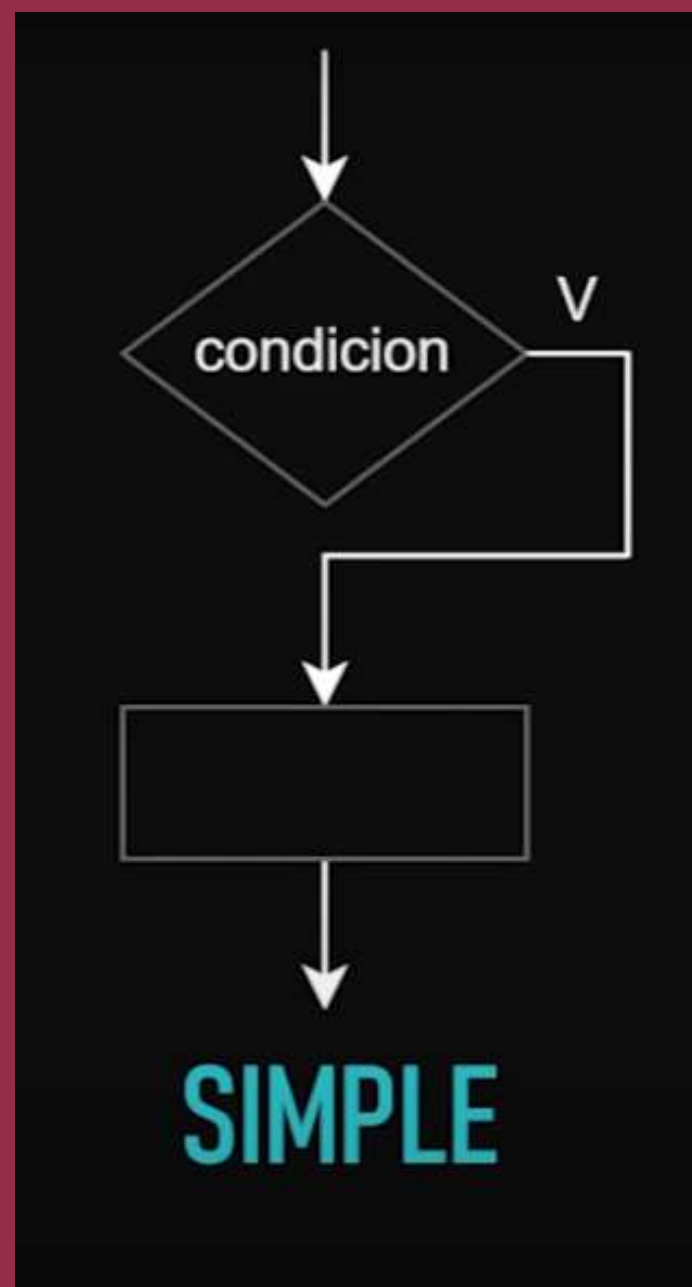
## ITERATIVAS

Repiten código un número determinado de veces



# CONDICIONAL SI (if)

Evalúan si una o más condiciones se cumplen o no





# CONDICIONAL SI (if)



# CONDICIONAL SI (if)

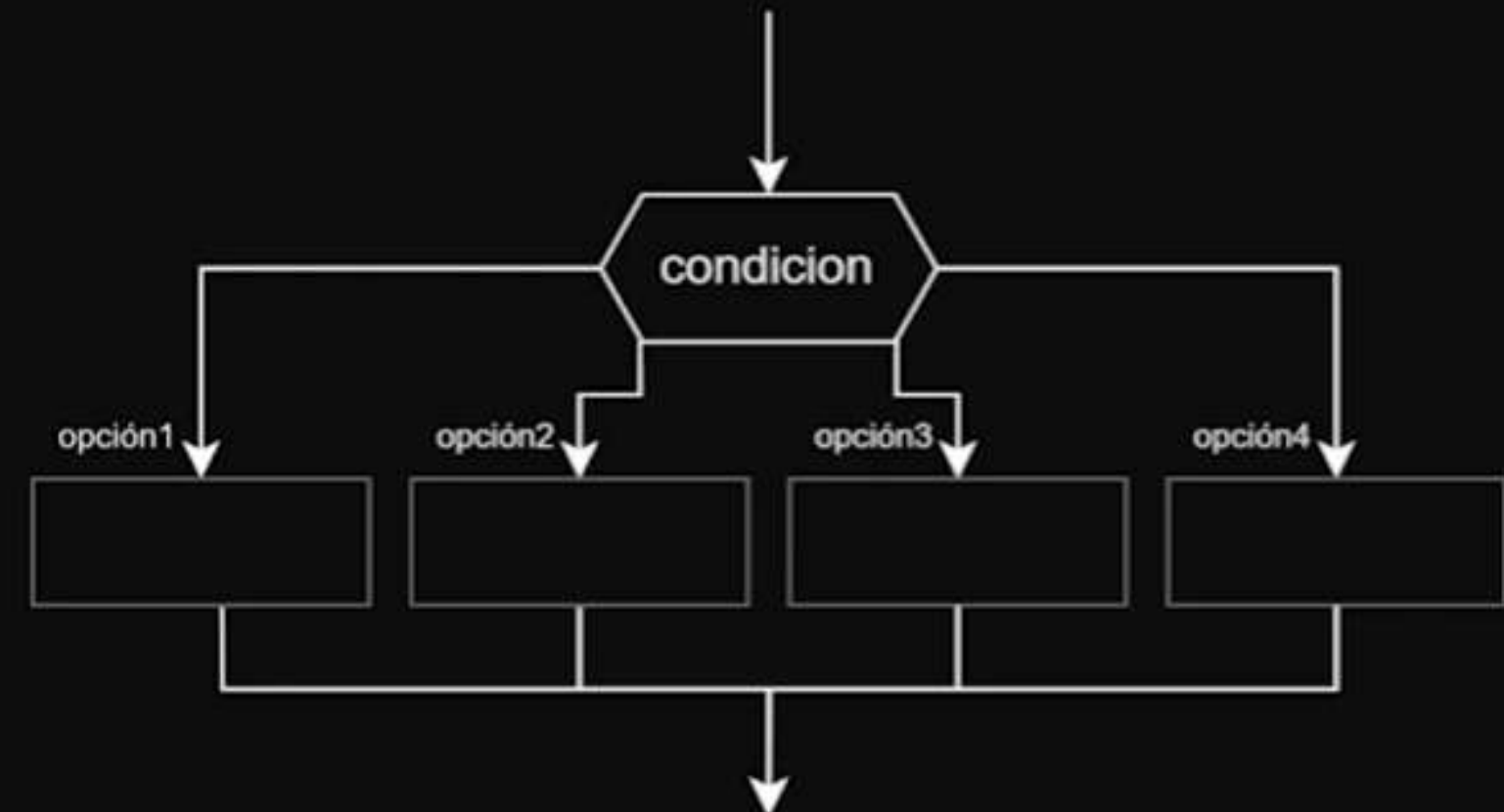
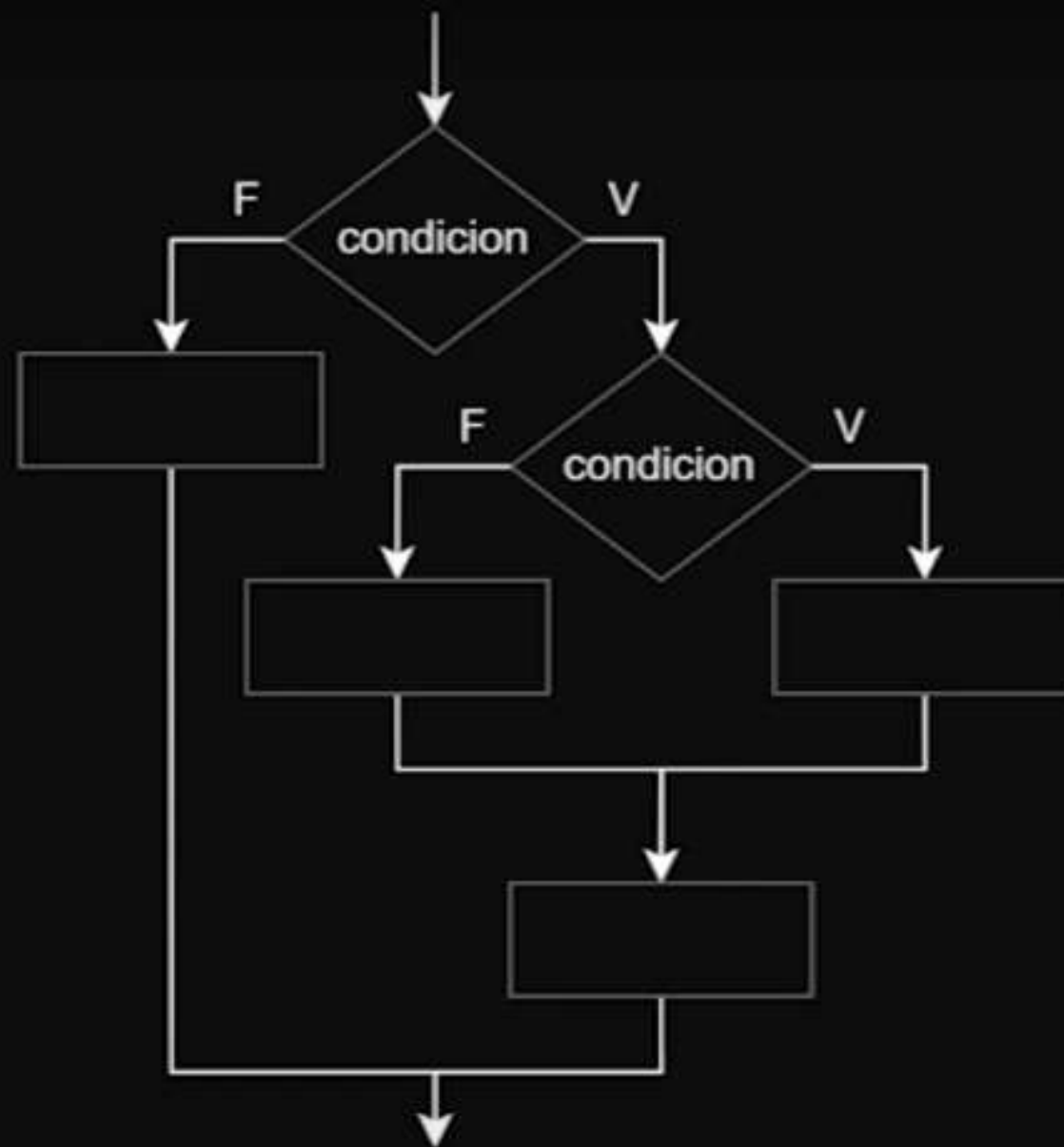
Si tengo crédito, llama,  
Sino, suena grabación  
COMPUESTO



# CONDICIONAL SI (if)

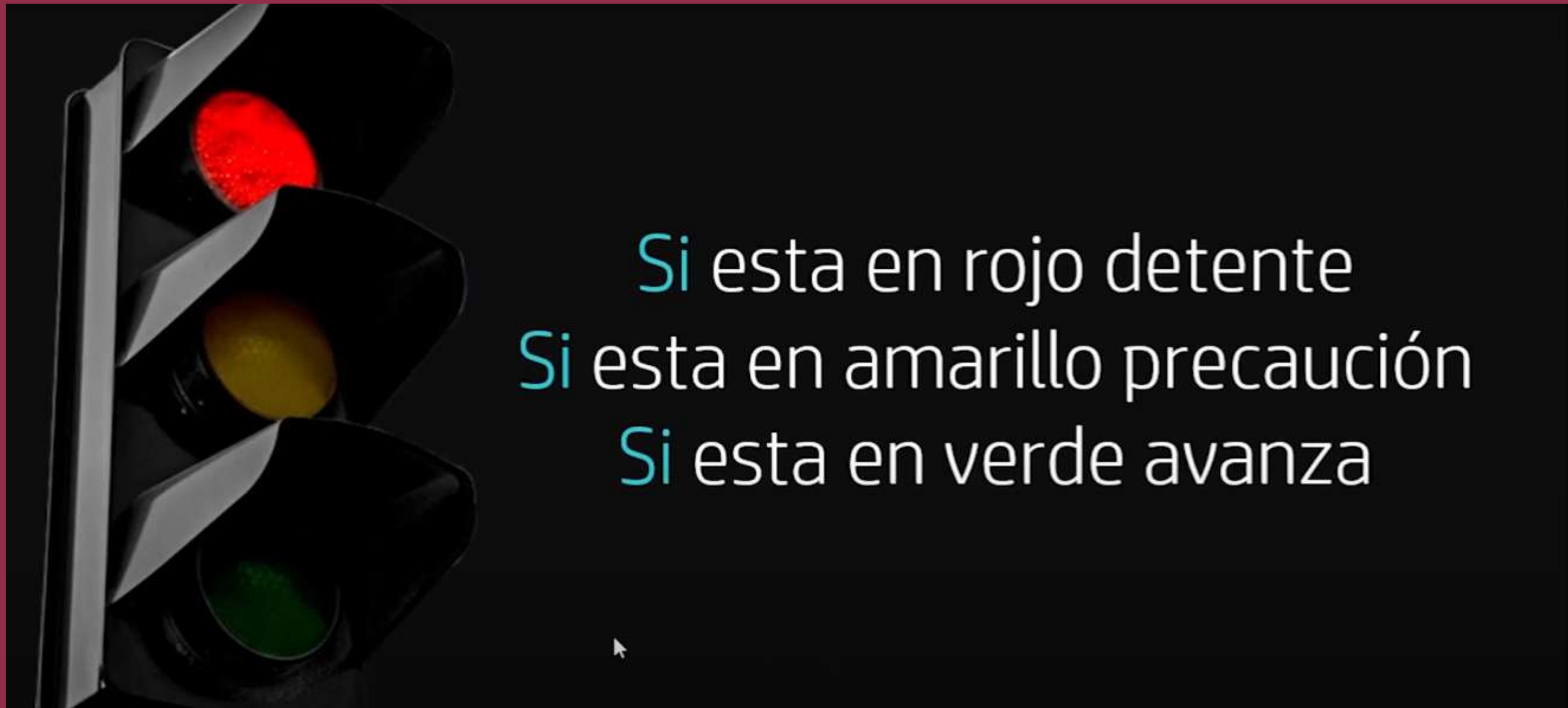
## MULTIPLE

Evalúa si una o más condiciones se cumplen o no



# CONDICIONAL SI (if)

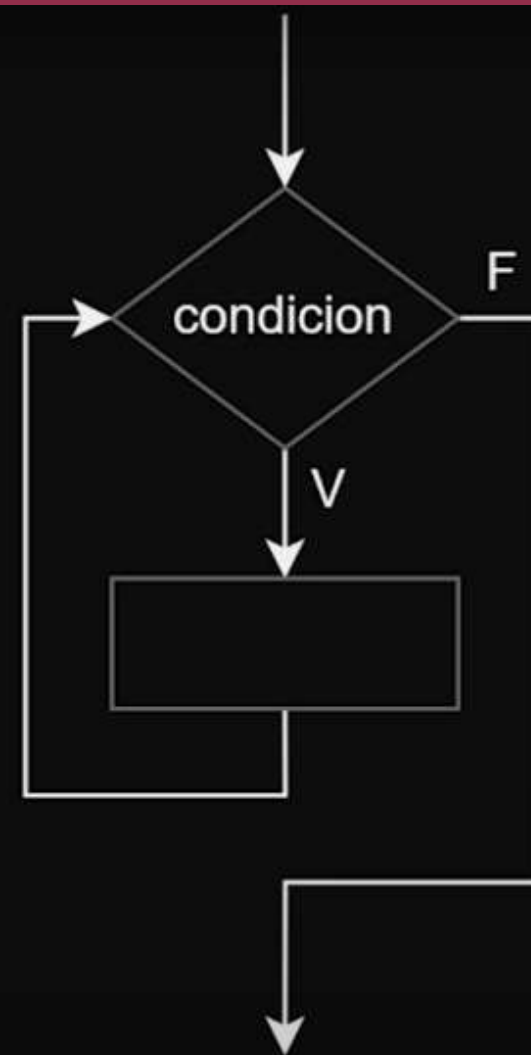
## MULTIPLE





# MIENTRAS (While)

Mientras una condición sea verdadera continua el ciclo, si desde el inicio la condición es falsa, nunca entra al ciclo



# MIENTRAS (While)

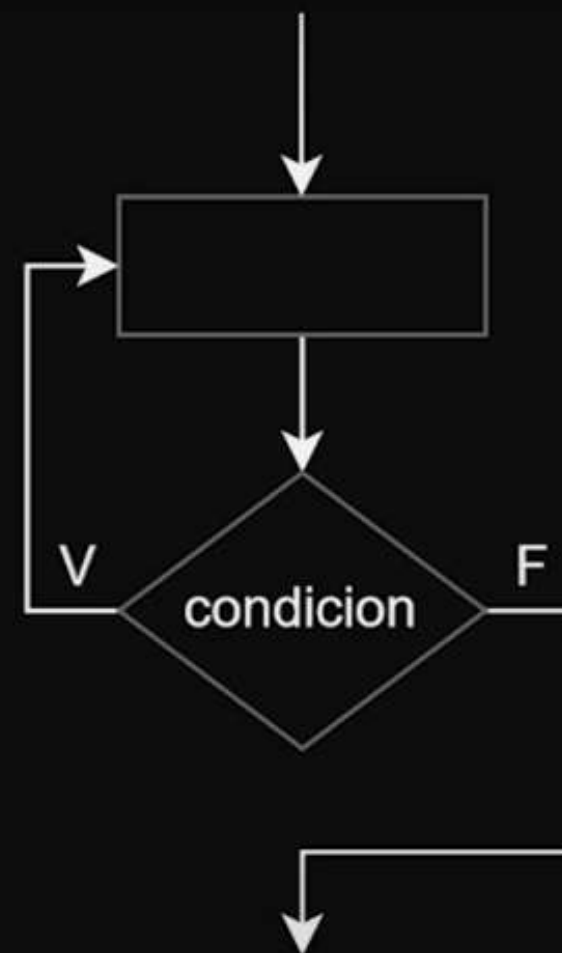
MIENTRAS haya hojas,  
imprime






# HACER-MIENTRAS (do-While)

Funciona igual que while pero primero hace y luego evalúa, por lo tanto, aun que desde el inicio la condición sea falsa, al menos pasará una vez por el ciclo.



# HACER-MIENTRAS (do-While)



Pregunta que  
deseas, mientras  
sigas queriendo  
algo

# PARA (for)

Se usa cuando se conoce o se puede conocer el número de ciclos (iteraciones)

## PARTES

### VALOR INICIAL

Valor con el que inicia el bloque

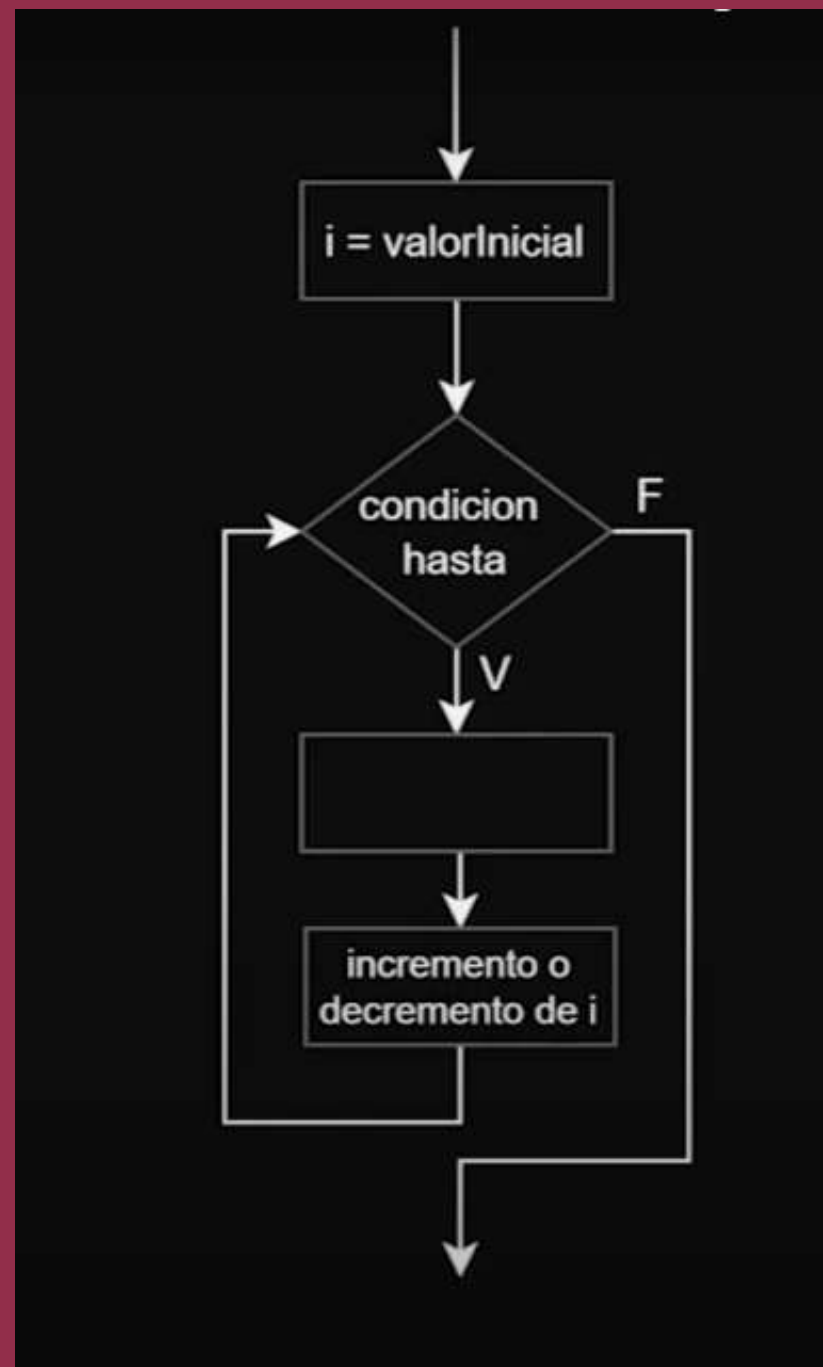
La variable que guarda este valor se suele nombrar "i"  
, "j", "k"

### CONDICIÓN

Determina cuando se detiene el bucle

### INCREMENTO (O DECREMENTO)

Indica el valor que se le sumará ( o restará) tras  
completar un ciclo



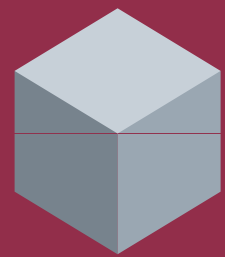


# PARA (for)

Desde 0 Hasta monto  
ganado, tira una  
moneda

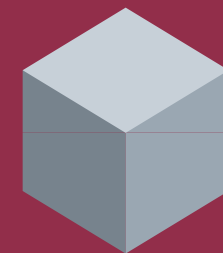


# VIII. INTRODUCCIÓN AL DISEÑO DEL PSEUDOCODIGO



Presentación de la estructura del pseudocodigo

Ejercicio transformación de un algoritmo



# PSEUDOCÓDIGO

Supuesto-Código

"Es un lenguaje  
que narra  
los pasos de un  
algoritmo"

## CARACTERÍSTICAS

- Lenguaje común y conciso
- Existe un único inicio
- Existe uno o más puntos de fin
- Alinear los bloques de código de acuerdo al nivel de la instrucción (sangría)



# PSEUDOCÓDIGO

Inicio / Fin

Inicio

Instrucciones

Fin

Decisión

Si ( condición ) entonces {

Instrucciones

} sino {

Instrucciones

}

# PSEUDOCÓDIGO

Mientras  
⬆

```
Mientras ( condición ) hacer {  
    Instrucciones  
}
```

Hacer - Mientras

```
Hacer {  
    Instrucciones  
} mientras ( condición )
```

Para

```
Para contador = valorInicial mientras condición hacer {  
    Instrucciones  
    Incrementar/Decrementar contador  
}
```

# MANOS A LA OBRA

## PROBLEMA

Calcular el monto total de la compra de libros en una librería, el número de libros comprados es variable, se lee el precio de cada libro, se suma y finalmente se despliega el total de la compra considerando que si el monto excede de \$1000.00 se otorga un descuento del 20%

Entradas

Salidas

Inicio

```
nLibrosComprados = 0
imprime "¿Cuandos libros quieres comprar?"
leer nLibrosComprados
totalCompra = 0
Para i = 1 mientras i <= nLibrosComprados hacer {
    precioLibro = 0
    imprime "Cual es el precio del libro: " + i
    leer precioLibro
    totalCompra = totalCompra + precioLibro
    i = i + 1
}
Si ( totalCompra > 1000 ) {
    descuento = (20 * totalCompra / 100)
    totalCompra = totalCompra - descuento
}
imprime "El total de la compra es: $" + totalCompra
```

Fin



# PRUEBA DE ESCRITORIO

nLibros Comprados	i	Precio Libro	Total Compra	descuento
0	1	0	0	
3	2	5	5	
	3	0	8	
	4	3	10	
		0		
		2		

```

Inicio
nLibrosComprados = 0
imprime "¿Cuandos libros quieres comprar?"
leer nLibrosComprados
totalCompra = 0
Para i = 1 mientras i <= nLibrosComprados hacer {
    precioLibro = 0
    imprime "Cual es el precio del libro: " + i
    leer precioLibro
    totalCompra = totalCompra + precioLibro
    i = i + 1
}
Si ( totalCompra > 1000 ) {
    descuento = (20 * totalCompra / 100 )
    totalCompra = totalCompra - descuento
}
imprime "El total de la compra es: $" + totalCompra
Fin
  
```

# RETOS

Realiza el pseudocódigo de un programa que lea una calificación y decida si es aprobatoria o no.

**(Puedes utilizar herramientas como Word o PSeInt)**