Teoría

Pseudocódigo y Diagrama de Flujo

Resolución de Problemas y Algoritmos
Fundamentos de la Programación

Ingeniería en Computación (TU - TFA)
Ingeniería en Minas (TU)
Profesorado en Ciencias de la Computación (TU - TFA)

CC BY NC SA

¿CÓMO RESOLVERLOS? ETAPAS EN LA RESOLUCIÓN DE PROBLEMAS

ETAPA 1: ANÁLISIS DEL PROBLEMA

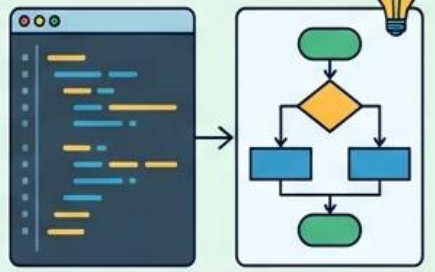
- Entender el enunciado y comprender el problema

- Representación e interpretación de datos (entrada/proceso/salida)

ETAPA 2: DISEÑO Y DESARROLLO DEL ALGORITMO

- Descomposición de un problema en tareas

- Algoritmo que da solución a un problema: pseudocódigo y diagrama de flujo



PASO A PASO HACIA LA SOLUCIÓN DIGITAL

Etapa 2: Diseño y desarrollo del algoritmo para resolver el problema.

1. Descomposición de un problema en tareas (**Versión 1**).
2. Algoritmo que da solución a un problema: **pseudocódigo y diagrama de flujo.**

Versión 1

T1: Definir el año de nacimiento y edad como entero.

T2: Ingresar el año de nacimiento de una persona.

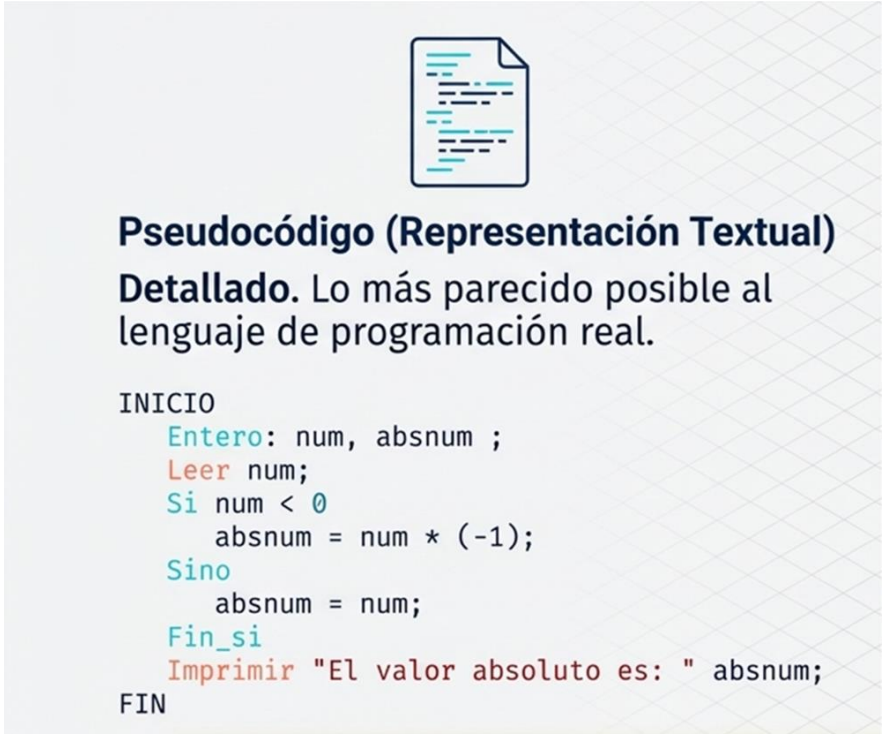
T3: Calcular la edad con el año de nacimiento ingresado.

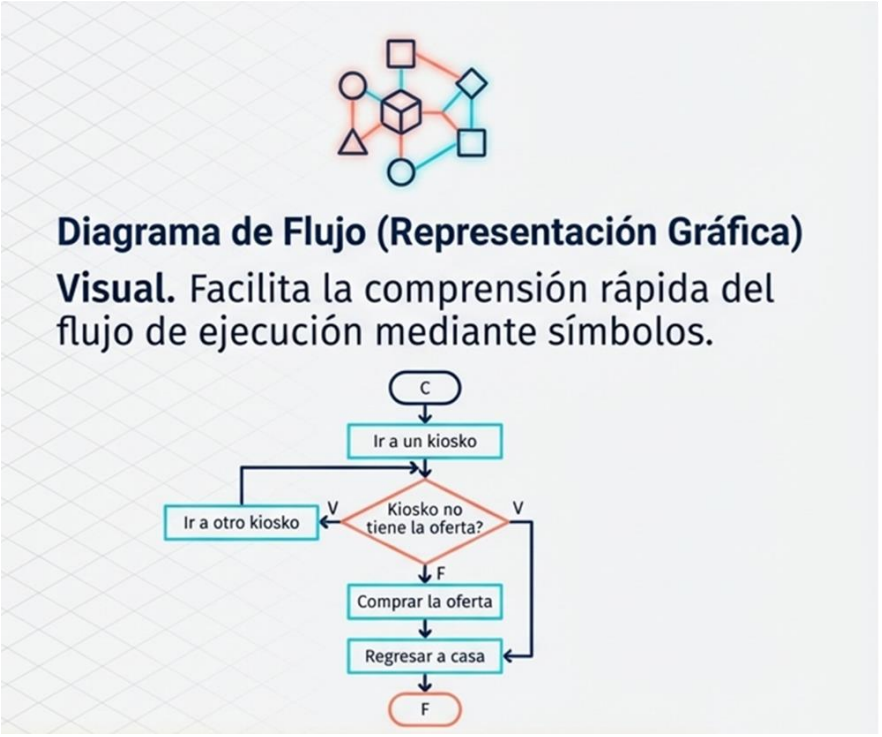
T4: Informar la edad calculada en T3.

```
Proceso CalcularEdad
    Definir edad, FNac como Entero
    Escribir "Ingrese anio de nacimiento "
    Leer FNac
    edad<-2026-FNac
    Escribir "La edad es: ", edad
FinProceso
```

PSEUDOCÓDIGO es usado para la **representación textual** de algoritmos, de la forma más detallada posible y a su vez lo más parecida posible al lenguaje de programación que posteriormente su utilizará para la codificación del mismo.

DIAGRAMA DE FLUJO es utilizado para la **representación gráfica** de algoritmos, facilita la visualización del flujo de ejecución del mismo.





Ambos representan exactamente la misma lógica, solo cambia la **perspectiva.**

NotebookLM

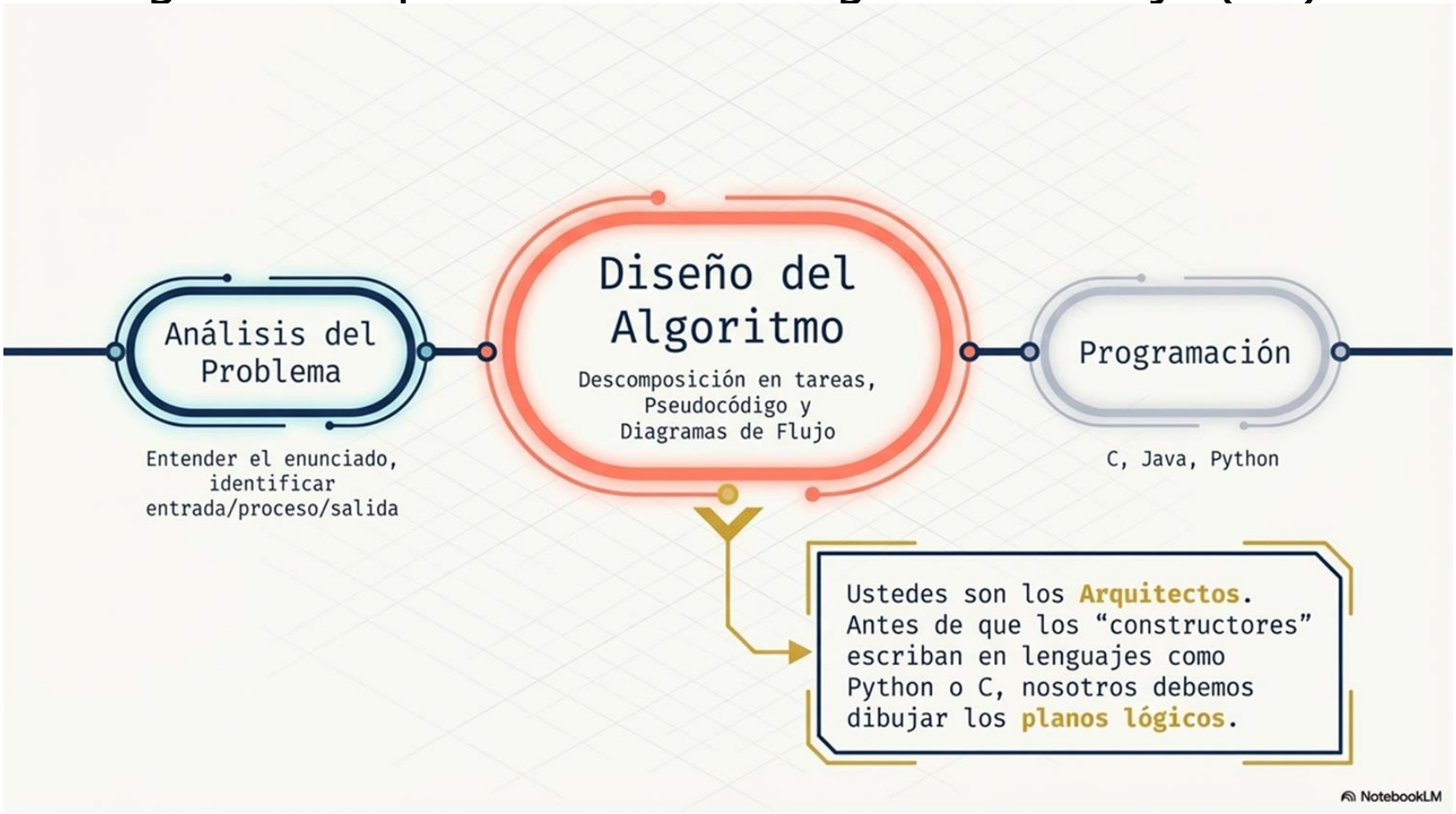
Pseudocódigo y Diagrama de Flujo

¿Qué temas veremos en esta Teoría?

- ✓ **Representación de algoritmos: Pseudocódigo.**
  - Representación de datos de entrada y salida.
  - Expresiones (aritméticas/lógicas/relacionales).
  - Asignación.
  - Entrada de datos y salida.
  - Estructuras de control: secuencial, condicional y de repetición.
- ✓ **Representación de algoritmos: Diagrama de Flujo.**
  - Simbología.
  - Ejemplo completo.
- ✓ **Pseudocódigo con PSeInt.**

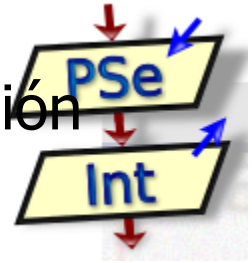
Diseño del algoritmo

1. Algoritmo escrito en pseudocódigo (PSeInt).
2. Algoritmo representado en Diagrama de Flujo (DF).



Diferencia entre
Lenguaje de diseño y Lenguaje de programación

Lenguaje de diseño, su objetivo es ser comprensible para las **personas** que van a interpretar los algoritmos escritos con él.



Los algoritmos escritos en **pseudocódigo** utilizan una combinación de lenguaje natural y elementos similares al lenguaje de programación.

Mientras que el propósito de un **lenguaje de programación** es ser comprensible por la **computadora** que va a ejecutar el **programa**.

DISEÑO VS. PROGRAMACIÓN: Entendiendo la la Diferencia

LENGUAJE DE DISEÑO

Explicar Soluciones (Para Personas):
Una forma de escribir pensada para que las personas entiendan fácilmente, combinando palabras del lenguaje natural con algunas reglas simples.


```
ALGORITMO
Proceso_Minero:
INICIO
Captar_Datos_Roca
Analizar_Dureza
SI Dureza > X
    -> Excavadora_Grande
SINO -> Excavadora_Media
FIN
```

PSEUDOCÓDIGO
(Algoritmo Organizado)

Pseudocódigo

LENGUAJE DE DISEÑO
(Flujo Conceptual)

CONCEPTUAL Y FLEXIBLE: Explica la lógica sin preocuparse por detalles técnicos estrictos.
ORIENTADO A HUMANOS: Facilita la comunicación de ideas y la comprensión del problema.
INDEPENDIENTE DE LENGUAJE: No depende de un lenguaje de programación especifico.

LENGUAJES DE PROGRAMACIÓN

Instrucciones Formales (Para Computadoras):
Un lenguaje formal con reglas estrictas para que la computadora entienda y ejecute instrucciones precisas.

```
#include <iostream>
using namespace std;
int main() {
  float dureza;

  // .. (Code in C++)
  if (dureza > 7.5) {
    cout << "Usar Excavadora Grande";
  } else {
    cout << "Usar Excavadora Media";
  }
  return 0;
}
```

LENGUAJES DE PROGRAMACIÓN

EJECUTABLE POR MÁQUINA: Las instrucciones se ejecutan directamente en la computadora.
SINTAXIS ESTRICTA: Requiere precisión total y cumplimiento de reglas técnicas complejas.
IMPLEMENTACIÓN TÉCNICA: Define cómo se realiza la solución a nivel de hardware/software.

DEL DISEÑO A LA PROGRAMACIÓN

TRADUCCIÓN

- ✓ **Representación de algoritmos: Pseudocódigo.**
  - Representación de datos de entrada y salida.
  - Expresiones (aritméticas/lógicas/relacionales).
  - Asignación.
  - Entrada de datos y salida.
  - Estructuras de control: secuencial, condicional y de repetición.

Representación de datos de entrada y salida.

Los principales **tipos de datos primitivos o simples** son:

- Numéricos: Enteros y Reales.
- Caracteres.
- Lógicos o Booleanos.

Entero	Real	Lógico	Caracter
			
Números completos sin decimales.	Números con punto decimal.	Valores booleanos de verdad.	Un solo símbolo de texto entre comillas simples.
128 -5	3.14 -0.5	VERDADERO FALSO	'A' '?'

Tipos de datos primitivos

ENTERO: consiste de un conjunto finito de los números enteros positivos y negativos. Ejemplos: 3, -3, 345, -56.

REAL: consiste de un conjunto finito de valores de los números reales. Números con **punto decimal**, positivos y negativos. Ejemplos: 2.3, -4.6.

LÓGICO: también llamado tipo **booleano**, es el conjunto de los valores de verdad: VERDADERO y FALSO.

CARACTER: es el conjunto finito y "ordenado" de caracteres que el procesador puede reconocer. Ejemplos: 'A', '3', '+'.

las letras mayúsculas del abecedario.
las letras minúsculas del abecedario.
los caracteres numéricos del 0...9.
el caracter de espacio blanco, caracteres especiales tales como: *, +, -, _, /, (,), $, ^, %, $, < ,>, ", .

¿Dónde se guardan los datos de entrada y salida?

Una **variable** es un espacio, en la memoria de la computadora, reservado para guardar un dato.

Se llama "variable" porque el dato que guardamos adentro puede cambiar a lo largo del tiempo, pero el espacio siempre es el mismo.

Para que esta variable funcione bien, necesita dos cosas:

- un **Nombre**: Para poder encontrar el dato rápidamente sin buscar en otros lados.
- un **Tipo**: Para saber qué "clase" de dato entra ahí (por ejemplo, si el espacio es para guardar un número o para guardar una palabra).

Toda **variable** debe definirse indicando el **nombre** y el **tipo** de valores que la misma puede tomar.

Definir <nombre de variable> [,<nombre de variable>]* **como** <tipo>

Ejemplos:

Definir NUMERO **como** Entero

Definir numero **como** Real

Definir Peso, Suma **como** Real

```
//Identificador de variable

No puede tener espacios;
No puede empezar por un numero;
No puede ser una palabra
reservada;
Distingue mayusculas
```

Expresiones (aritméticas/lógicas/relacionales).

Una expresión describe un cálculo a efectuar cuyo resultado es un valor único.

Una expresión consta de **operadores** y **operandos**.

Expresión	Resultado	Ejemplos
Expresión **aritmética**	Tipo numérico.	3 *-6 el resultado es -18 2.5 - 1 el resultado es 1.5
Expresión **relacional**	Tipo lógico	5 < 9 el resultado es VERDADERO -8 = 8 el resultado es FALSO
Expresión **lógica**	Tipo lógico	VERDADERO Y FALSO el resultado es FALSO NO VERDADERO el resultado es FALSO

Expresiones Aritméticas

Cálculos que generan un resultado numérico

Se utilizan para realizar operaciones matemáticas tradicionales utilizando constantes, variables numéricas u otras expresiones encerradas en paréntesis.

`2.5 - 1 = 1.5` → **Resultado Numérico Decimal**

Operadores Aritméticos

Operadores	Significado
+	Suma
-	Resta
*	Producto
/	División
^	Potencia
% ó MOD	Resto de la división

Orden de Precedencia Aritmética

Definir precio como Real

Definir descuento como Real

Definir edad como Entero

2.5 - 1

precio * (5 + 3.6)

edad + 35

precio - descuento

5 MOD 2

¿Qué valores tienen precio, descuento y edad?
¿Cómo almaceno valores esas variables?

Expresiones Relacionales

Comparaciones entre valores

5 < > 9

Estas expresiones comparan dos operandos (que pueden ser números o caracteres) para establecer una relación de igualdad o jerarquía.

5 < 9 resulta en VERDADERO

-8 = 8 resulta en FALSO

Resultado siempre es de Tipo Lógico (VERDADERO o FALSO)

Operadores Relacionales

Operadores	Significado
=	Igual
<	Menor
<=	Menor o igual
>	Mayor
>=	Mayor o igual
<>	Distinto

Definir NUM como Real

Definir VAL como Caracter

‘A’ < ‘M’

NUM >= 3.6

VAL <> ‘b’

1.3 >= 0.75

¿Qué valores tienen NUM y VAL?
¿Cómo almaceno valores en NUM y VAL?

Expresiones Lógicas

Conectores de lógica proposicional

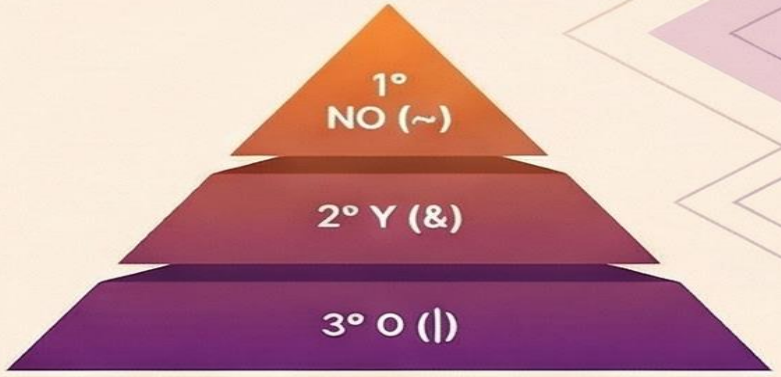
Evalúan operandos que ya son de tipo lógico (como variables booleanas o resultados de expresiones relacionales) mediante conectores.

VERDADERO **Y** FALSO = FALSO

La operación 'Y' solo es verdadera si ambos operandos son verdaderos.

Operadores Lógicos

Operadores	Técnico
& ó Y	Conjunción
| ó O	Disjunción
~ ó NO	Negación



Orden de Precedencia Lógica

Definir VAL como Logico

Definir NUM como Entero

(VAL <>FALSO) **Y** (NUM < 45)

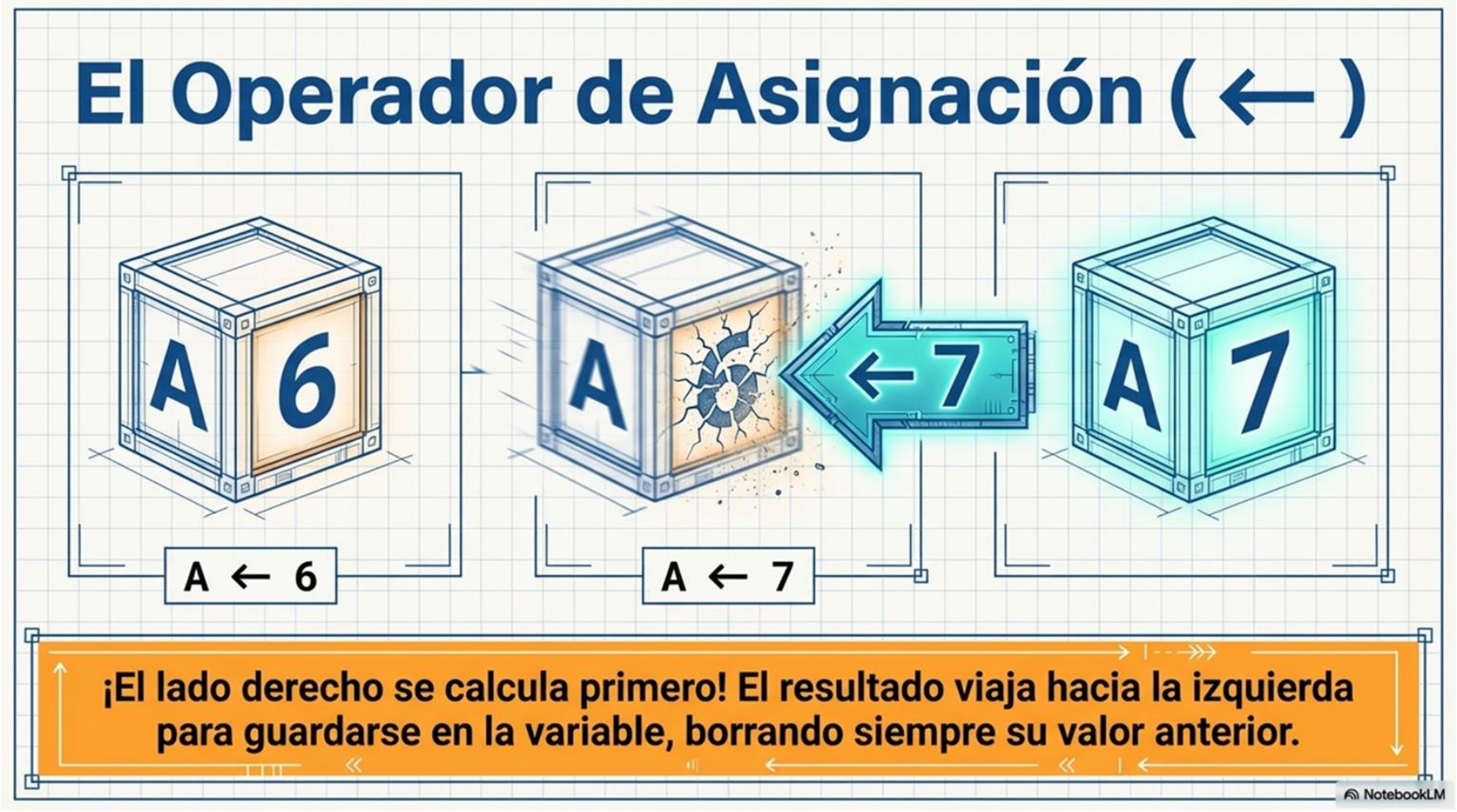
NO (5 > 3.6)

('A' < 'M') **O** ('s' > 'e')

¿Qué valores tienen NUM y VAL?
¿Cómo almaceno valores en NUM y VAL?

Asignación de valores.

Para asignar (dar) valor a una variable, se utiliza el **operador de asignación**.



Asignación de valores.

Var ← E

Var es el nombre de la variable a la que el procesador va a asignar (dar) el valor de E.

←, identifica al operador de asignación.

E representa el valor a asignar y puede ser una constante, otra variable, o el resultado de la evaluación de una expresión.

Num ← **6**
Inicial ← **'M'**

NUM ← Edad

Edad ← 6 +1
Edad ← NUM * 3

Ejemplos:

Acciones	Edad	NE
Edad ← 6	6	
NE ← 6 +1		7

Acciones	Med	NM
Med ← 2.14	2.14	
NM ← Med + 0.5		2.64

Acciones	Let	ver	VER
Let ← ‘M’	‘M’		
ver ← Let > ‘B’		verdadero	
VER ← Let = ‘m’			falso

Entrada y Salida de datos.

LEER: es la acción primitiva que permite la **entrada** de uno o más valores a objetos del ambiente a través de un dispositivo.

Una lectura es una asignación, toma valores del medio externo (por ejemplo, del teclado) y lo asigna a las variables del ambiente.

Leer <nombre de la variable>

Por ejemplo:

Leer Edad

Entrada y Salida de datos.

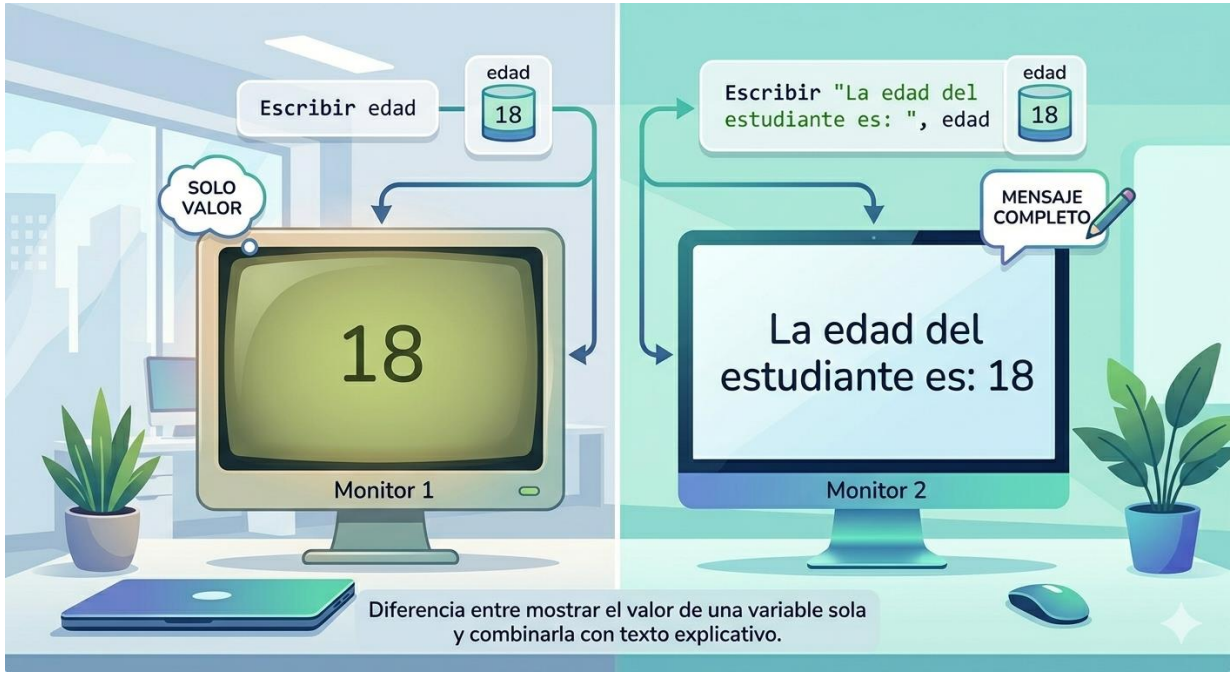
ESCRIBIR: es la acción primitiva que permite la **salida** de valores a través de un dispositivo (por ejemplo, monitor). Esta acción toma uno o más valores de datos y los comunica al medio externo.

Escribir "Texto", <nombre de la variable>

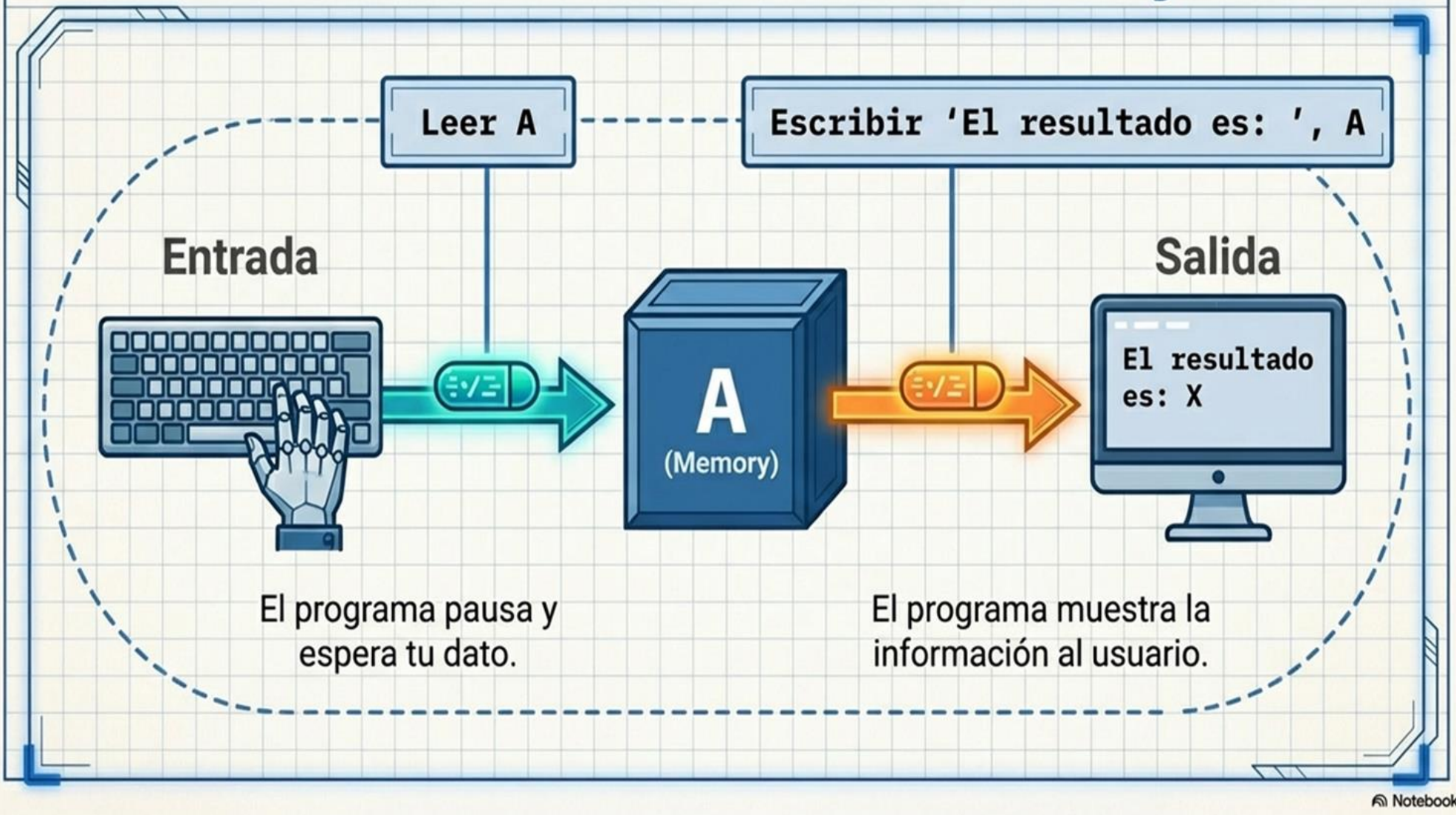
Por ejemplo:

Escribir edad

Escribir "La edad es: ", edad



Hablando con el Usuario: Entrada y Salida



NotebookLM

Acciones	edad	Pantalla
Definir edad como Entero	indefinido	
Escribir “Ingrese la edad”		Ingrese la edad
Leer edad	18	
Escribir “Tiene “, edad, “ anios”		Tiene 18 anios

¿Qué vimos hasta ahora?.

- Definir variables (nombre y tipo de dato).
- Dar valores a las variables:
  1. ←
  2. **Leer**

```
Leer NUM, VAR
```

- Operar con variables.
- Mostrar en la pantalla

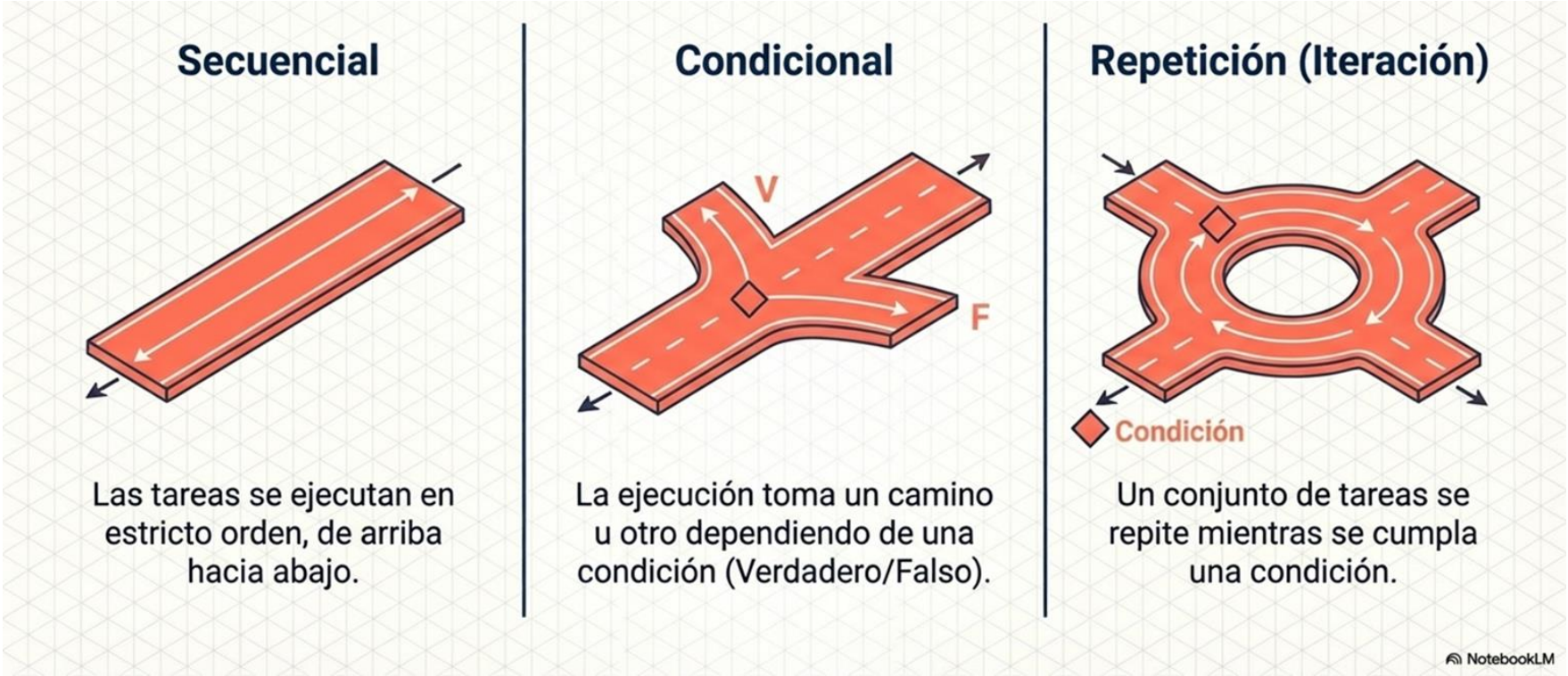
```
Escribir “Ingrese edad”
Escribir SUMA
Escribir “La suma es “, SUMA
```

Estructuras de control.

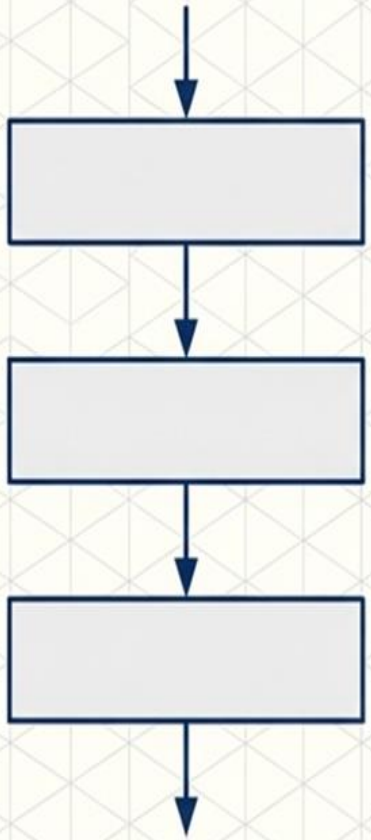
En la programación estructurada, existen las **estructuras de control de flujo**.

Esto hace referencia al orden en el que se ejecutarán las instrucciones de un programa, desde su comienzo hasta que finaliza.

Vamos a utilizar **3 estructuras de control**:



1. Estructura Secuencial



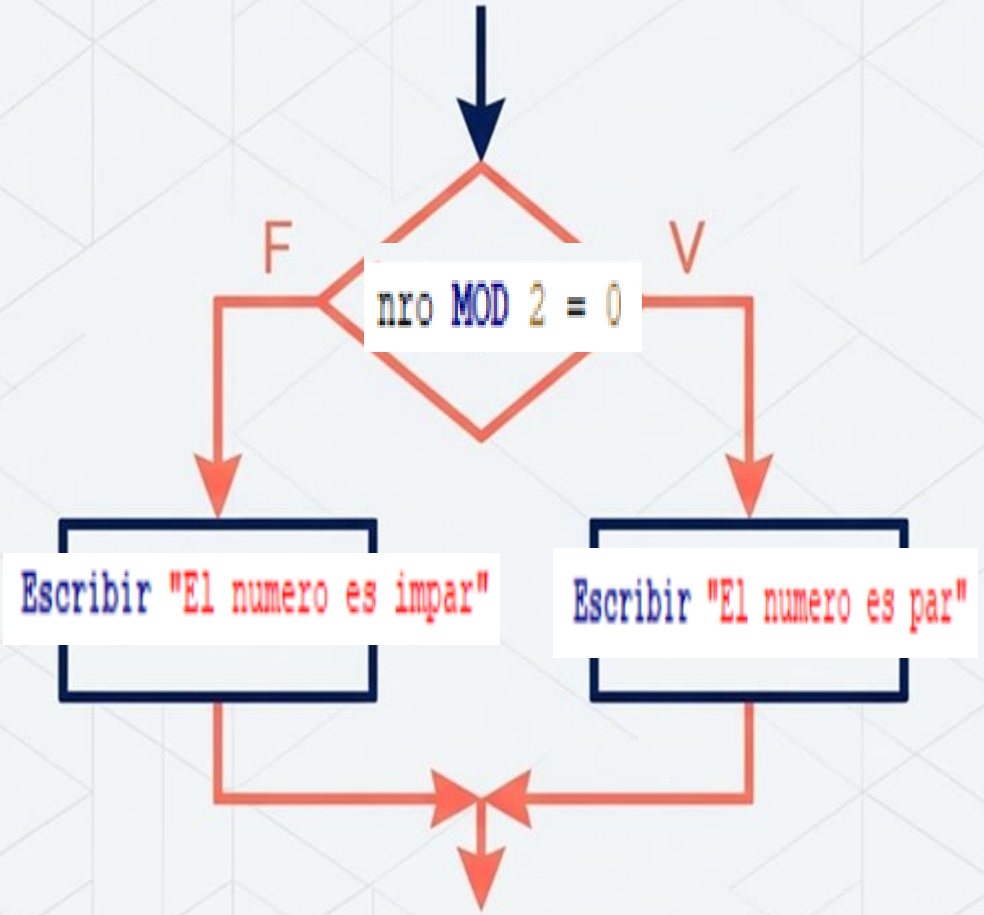
```
Definir edad, FNac como Entero
Escribir "Ingrese anio de nacimiento "
Leer FNac
edad<-2026-FNac
Escribir "La edad es: ", edad
```

¿Cuántos rectángulos debería dibujar?

Paso a paso. Una instrucción no comienza hasta que la anterior haya terminado.

NotebookLM

2. Estructura Condicional (Si / Sino)

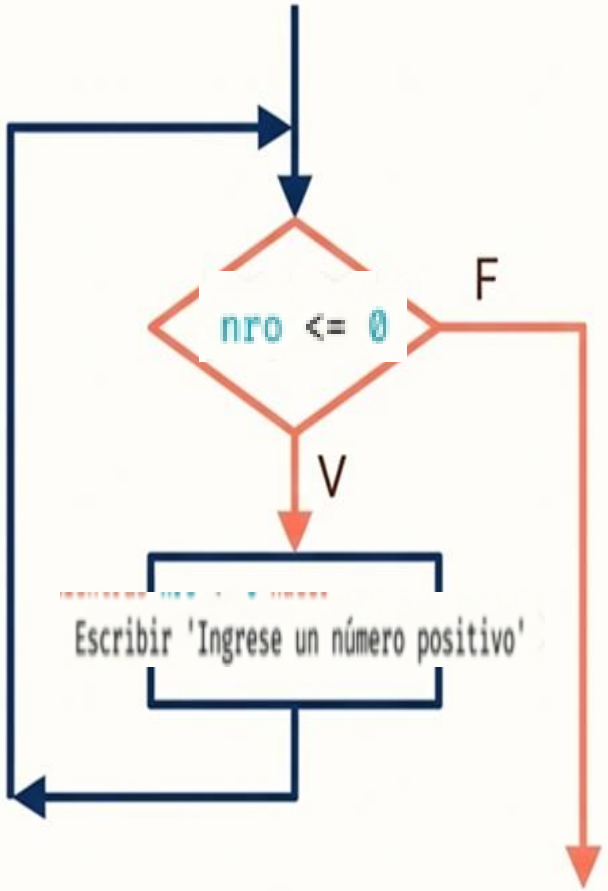



```
si nro MOD 2 = 0 Entonces
    Escribir "El numero es par"
sino
    Escribir "El numero es impar"
FinSi
```

El algoritmo 'evalúa' la situación en el rombo y elige su propio camino.

NotebookLM

3. Estructura de Repetición (Mientras)




```
Mientras nro <= 0 hacer
  Escribir 'Ingrese un número positivo'
  Leer nro
FinMientras
```

¿Cómo representaría la sentencia

```
Leer nro
```

Automatización real: la tarea se ejecuta repetitivamente hasta que la condición deja de cumplirse.

NotebookLM

```
1  Proceso ParImpar
2      Definir nro Como Entero
3      Escribir "Ingrese un número cualquiera"
4      Leer nro
5  FinProceso
```

Secuencial: las tareas (acciones) se deben realizar en el orden en que se escriben, es decir, primero una, luego otra, desde la primera hasta la última (o de arriba hacia abajo).

```
5      si nro MOD 2 = 0 Entonces
6          Escribir "El numero es par"
7      sino
8          Escribir "El numero es impar"
9      FinSi
```

Selección (Condicional): las tareas (acciones) se realizarán dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.

```
5      Mientras nro <= 0 hacer
6          Escribir "Ingrese un número cualquiera"
7          Leer nro
8      FinMientras
```

```
11     Mientras cont <= 4 Hacer
12         Escribir "Ingrese un número cualquiera"
13         Leer nro
14         si nro MOD 2 = 0 Entonces
15             Escribir "El numero es par"
16         sino
17             Escribir "El numero es impar"
18         FinSi
19         con<-cont+1
20     FinMientras
```

Iteración (Repetición): una tarea o conjunto de tareas (acciones) se deben realizar en forma repetitiva una **cantidad de veces específica** o dependiendo de una **condición** que se debe cumplir.

Condicional (selección)

Si <condición> **Entonces**
    <alternativa verdadera1>
Finsi

```
si nro MOD 2 = 0 Entonces
    Escribir "El numero es par"
FinSi
```

Si <condición> **Entonces**
    <alternativa verdadera>
Sino
    <alternativa falsa>
Finsi

```
si nro MOD 2 = 0 Entonces
    Escribir "El numero es par"
sino
    Escribir "El numero es impar"
FinSi
```

Repetición (Iteración)

Iteración simple

Mientras <condicion> **Hacer**

<secuencia de acciones>

FinMientras

```
cont<-1
Mientras cont <= 4 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
    si nro MOD 2 = 0 Entonces
        Escribir "El numero es par"
    FinSi
    cont<-cont+1
FinMientras
```

Iteración condicional

Mientras <condicion> **Hacer**

<secuencia de acciones>

FinMientras

```
Mientras nro <= 0 Hacer
    Escribir "Ingrese un número cualquiera"
    Leer nro
FinMientras
```

Etapa 2: Diseño y desarrollo del algoritmo para resolver el problema.

1. Descomposición de un problema en tareas (Versión 1)
2. Algoritmo que da solución a un problema: **pseudocódigo y diagrama de flujo.**

Versión 1

T1: Definir el año de nacimiento y edad como entero.

T2: Ingresar el año de nacimiento de una persona controlando que sea un número positivo.

T3: Calcular la edad con el año de nacimiento ingresado.

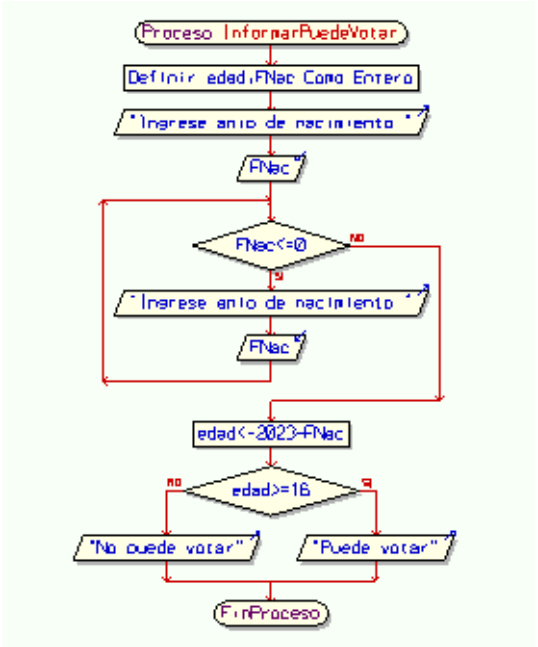
T4: Informar si puede votar o no teniendo en cuenta la edad calculada en T3.

```
Proceso InformarPuedeVotar
    Definir edad,FNac Como Entero
    Escribir 'Ingrese anio de nacimiento '
    Leer FNac
    Mientras FNac<=0 Hacer
        Escribir 'Ingrese anio de nacimiento '
        Leer FNac
    FinMientras
    edad<-2024-FNac
    Si edad>=16 Entonces
        Escribir 'Puede votar'
    Sino
        Escribir 'No puede votar'
    FinSi
FinProceso
```

Representación TEXTUAL Pseudocódigo

```
1   Proceso InformarPuedeVotar
2       Definir edad,FNac Como Entero
3       Escribir 'Ingrese anio de nacimiento '
4       Leer FNac
5       Mientras FNac<=0 Hacer
6           Escribir 'Ingrese anio de nacimiento '
7           Leer FNac
8       FinMientras
9       edad<-2024-FNac
10      Si edad>=16 Entonces
11          Escribir 'Puede votar'
12      Sino
13          Escribir 'No puede votar'
14      FinSi
15  FinProceso
```

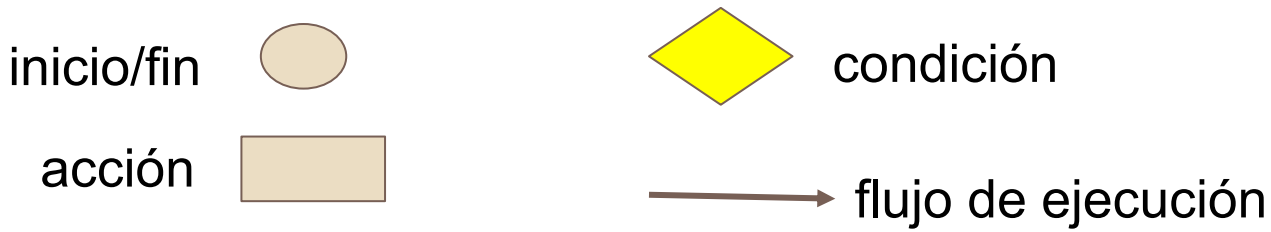
Representación GRÁFICA Diagrama de Flujo (DF)



Representación de algoritmos: Diagrama de Flujo.

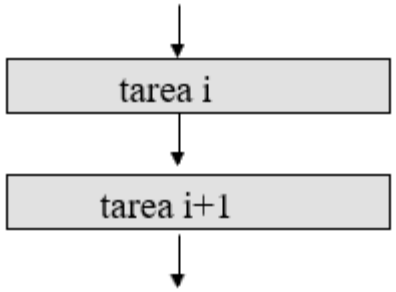
Un **Diagrama de Flujo de Datos** permite bosquejar **gráficamente**, el orden de las tareas involucradas en un algoritmo que da solución a un problema.

En esta materia se usan los siguientes símbolos para representar un DF:

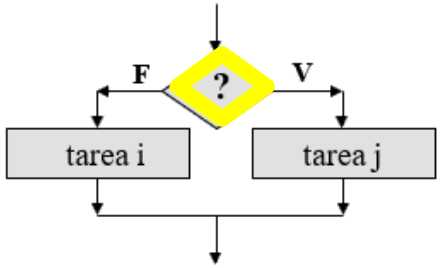


Algunas de las pautas para crearlos son:

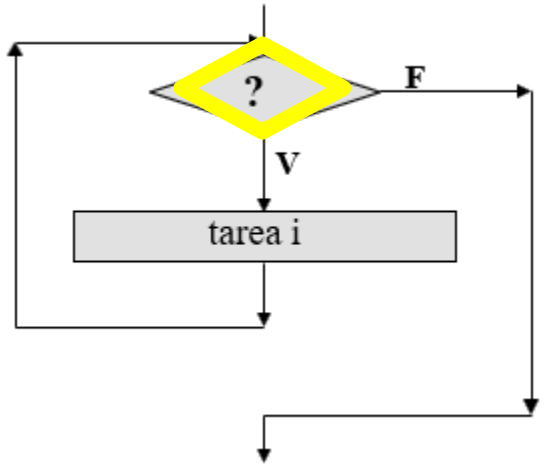
- Debe ser diseñado de forma que sea leído de arriba hacia abajo.
- Todo **DF** debe tener un **comienzo** y al menos un **final**.
- Para determinar el flujo de ejecución se deben usar **flechas** (NO líneas).



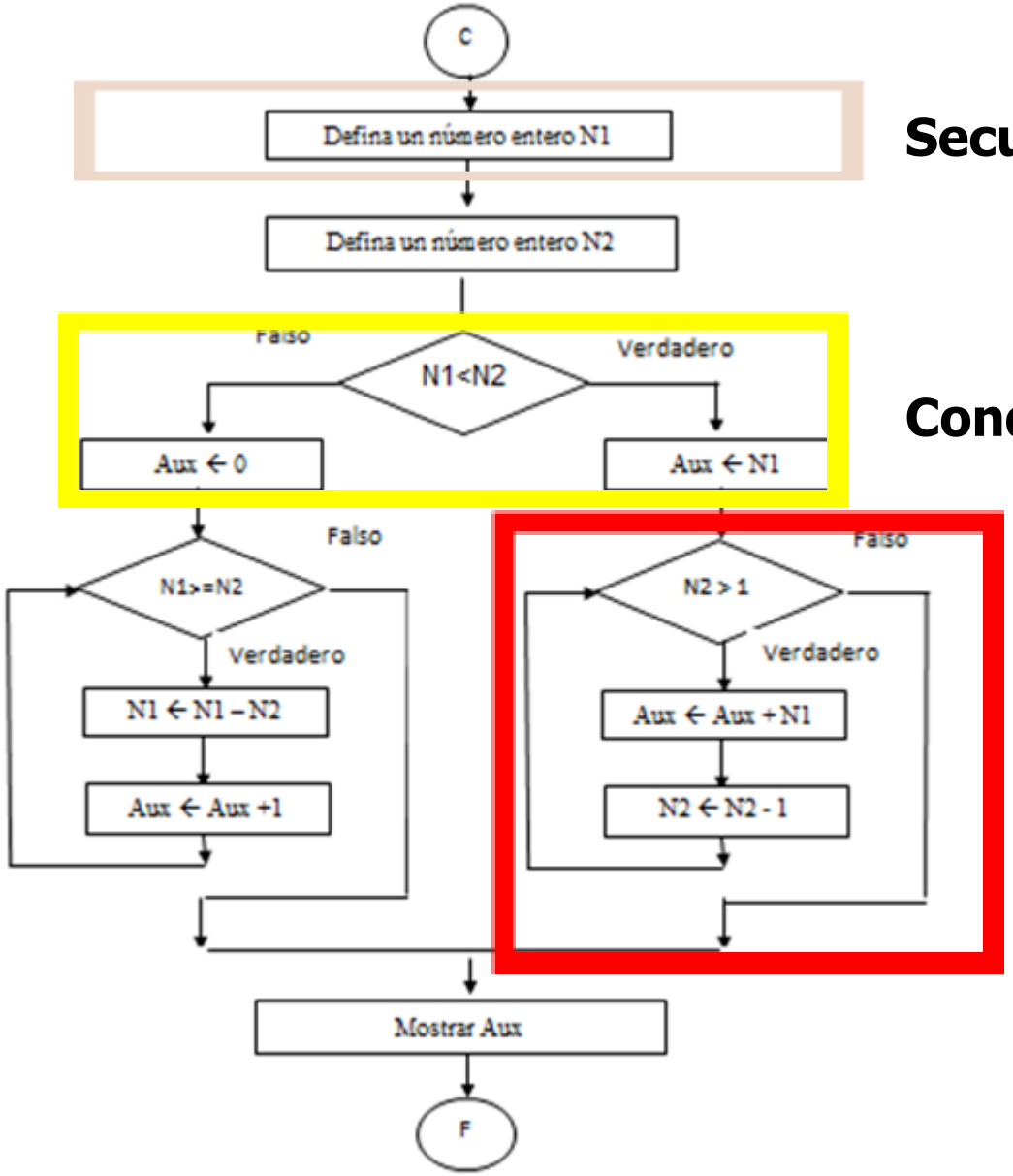
Secuencial: las tareas (acciones) se deben realizar en el orden en que se escriben, es decir, primero una, luego otra, desde la primera hasta la última (o de arriba hacia abajo).



Condicional (selección): las tareas (acciones) se realizarán dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.



Repetición (Iteración): una tarea o conjunto de tareas (acciones) se deben realizar en forma repetitiva dependiendo de cierta situación, estado previo o **condición** que se debe cumplir.



Secuencial

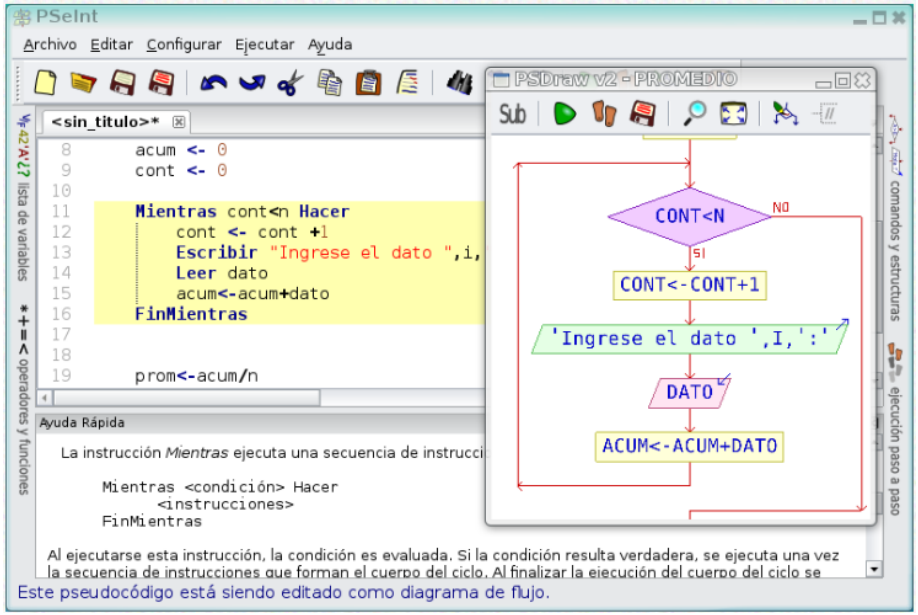
Condicional

Iteración

¿Qué es PSeInt?

https://pseint.sourceforge.net/

Es un editor e intérprete de algoritmos escritos en pseudocódigo que, además, genera su respectivo DF.



¿Por qué usar PSeInt?

Software Libre

Gratuito y sin restricciones.

En Español

Lógica en tu idioma nativo.

Práctico

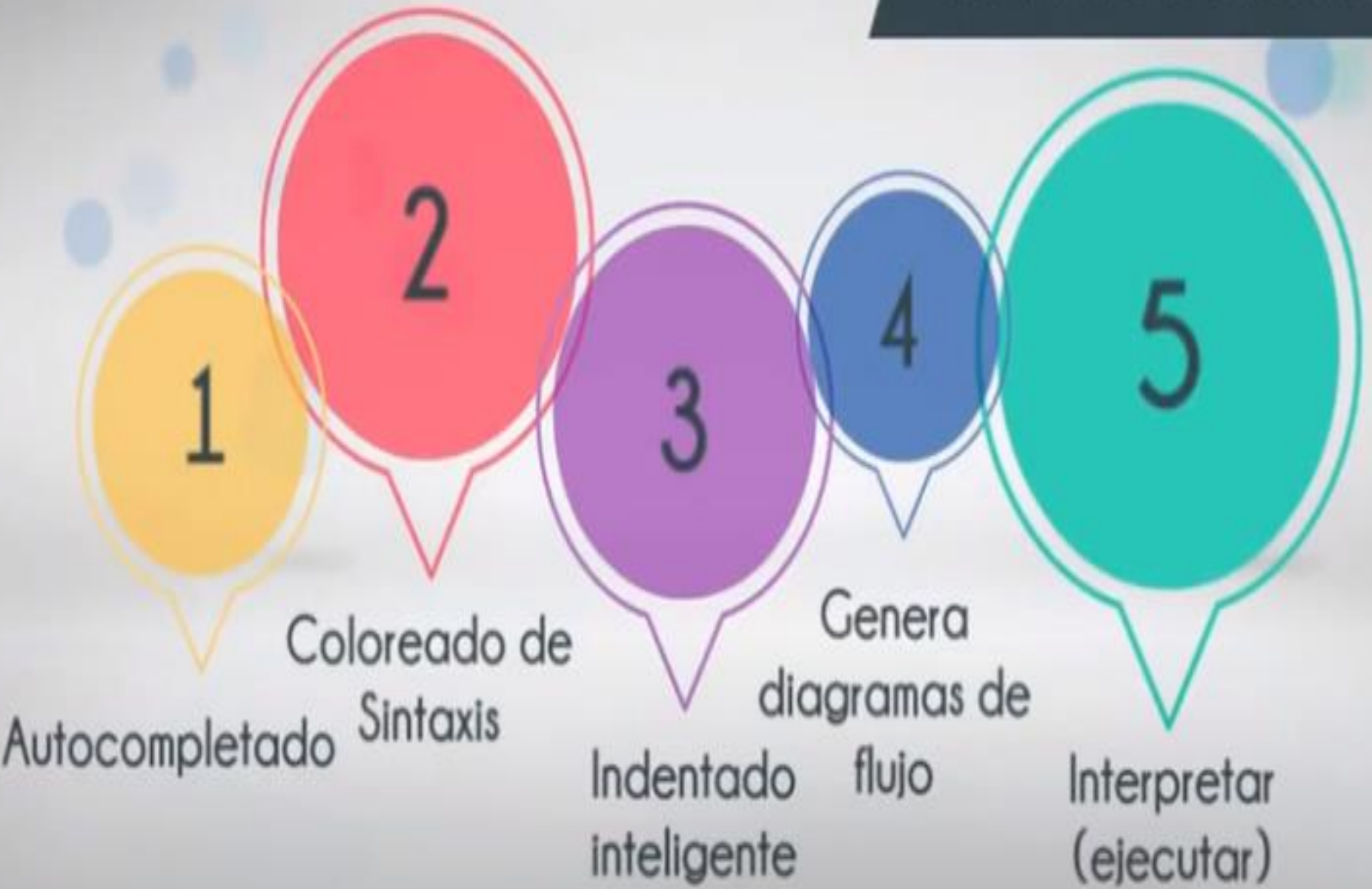
Ejecuta y prueba tu lógica instantáneamente.

Trampolín

Exportable a lenguajes reales como C++.

NotebookLM

CARACTERÍSTICAS



Ejemplos de algoritmos en PSeInt


```
// Calcula el promedio de una lista de N datos

Proceso Promedio

    Definir i,N como Entero
    Definir acum,dato,prom como Reales
    Escribir "Ingrese la cantidad de datos:"
    Leer n

    acum<-0

    Para i<-1 Hasta n Hacer
        Escribir "Ingrese el dato ",i,":"
        Leer dato
        acum<-acum+dato
    FinPara

    prom<-acum/n

    Escribir "El promedio es: ",prom

FinProceso
```

```
// este es el ejemplo más simple de esta ayuda,
// toma dos numeros, los suma y muestra el resultado

Proceso Suma

    Definir A,B,C como Reales

    // para cargar un dato, se le muestra un mensaje al usuario
    // con la instrucción Escribir, y luego se lee el dato en
    // una variable (A para el primero, B para el segundo) con
    // la instrucción Leer

    Escribir "Ingrese el primer numero:"
    Leer A

    Escribir "Ingrese el segundo numero:"
    Leer B


    // ahora se calcula la suma y se guarda el resultado en la
    // variable C mediante la asignación (<-)

    C <- A+B


    // finalmente, se muestra el resultado, precedido de un
    // mensaje para avisar al usuario, todo en una sola
    // instrucción Escribir

    Escribir "El resultado es: ",C

FinProceso
```

Ejemplo completo

Enunciado: dado el año de nacimiento de una persona, informarle si puede o no votar.

Versión 1

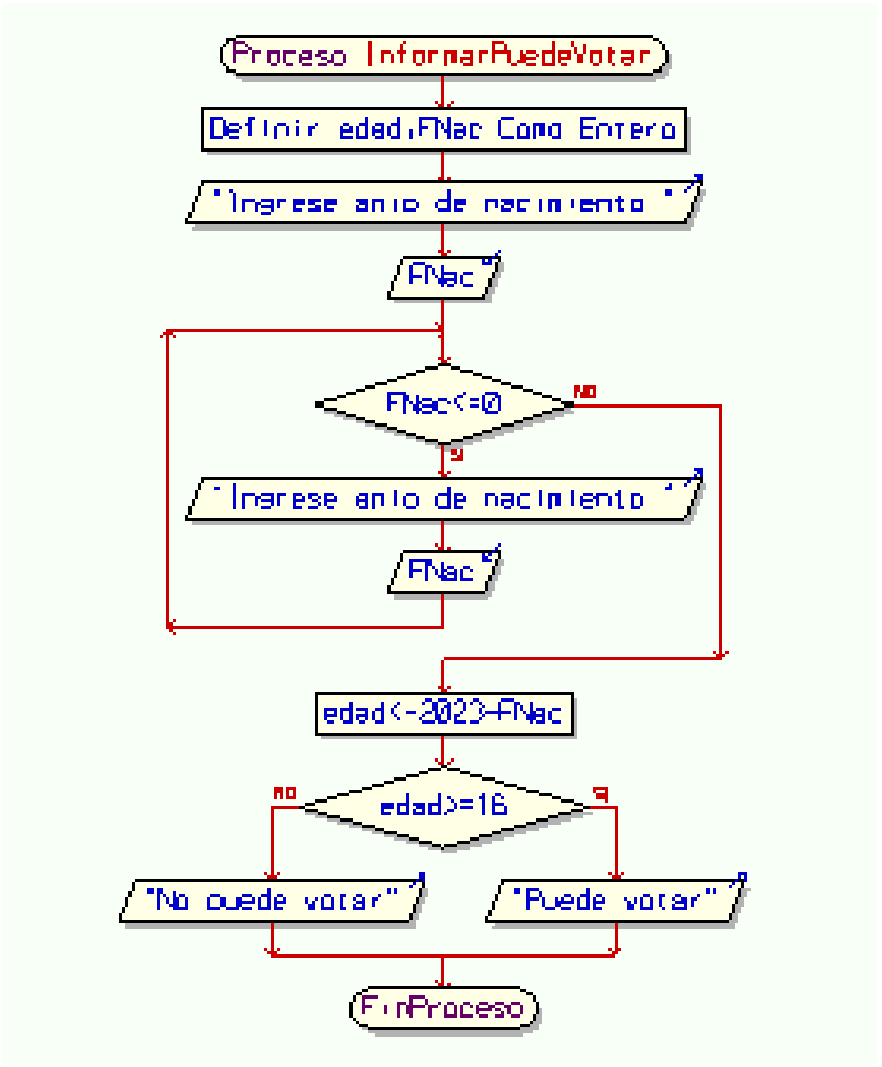
T1: Definir el año de nacimiento y edad como entero.

T2: Ingresar el año de nacimiento de una persona controlando que sea un número positivo.

T3: Calcular la edad con el año de nacimiento ingresado.

T4: Informar si puede votar o no teniendo en cuenta la edad calculada en T3.

```
Proceso InformarPuedeVotar
    Definir edad,FNac Como Entero
    Escribir 'Ingrese anio de nacimiento '
    Leer FNac
    Mientras FNac<=0 Hacer
        Escribir 'Ingrese anio de nacimiento '
        Leer FNac
    FinMientras
    edad<-2023-FNac
    Si edad>=16 Entonces
        Escribir 'Puede votar'
    Sino
        Escribir 'No puede votar'
    FinSi
FinProceso
```



Teoría
Pseudocódigo y Diagrama de Flujo

- ✓ **Representación de algoritmos: Pseudocódigo.**
  - Representación de datos de entrada y salida.
  - Expresiones (aritméticas/lógicas/relacionales).
  - Asignación.
  - Entrada de datos y salida.
  - Estructuras de control: secuencial, condicional y de repetición.
- ✓ **Representación de algoritmos: Diagrama de Flujo.**
  - Simbología.
  - Ejemplo completo.
- ✓ **Pseudocódigo con PSeInt.**

Teoría

PSeInt y DF

¡Ya podemos comenzar con el Práctico!