

# Informática I

## Unidad 2: Solución de Problemas

Cristian A. Jimenez C.

Departamento de Ingeniería Electrónica y Telecomunicaciones  
Facultad de Ingenierías  
Universidad de Antioquía

[cjimenez.castano@udea.edu.co](mailto:cjimenez.castano@udea.edu.co)

Oficina 20-405

## 1 Solución de Problemas

- Método de Polya

## 2 Algoritmos

- Diagramas de Flujo
  - Condicionales
  - Ciclos/Iteraciones
  - Contadores y Acumuladores
- Pruebas de Escritorios

## 3 Bibliografía

## 1 Solución de Problemas

- Método de Polya

## 2 Algoritmos

- Diagramas de Flujo
  - Condicionales
  - Ciclos/Iteraciones
  - Contadores y Acumuladores
- Pruebas de Escritorios

## 3 Bibliografía

# Solución de Problemas

A partir del planteamiento de un problema, oportunidad, o dificultad, es conveniente usar una **metodología** para la resolución del mismo, que tendrá como objetivo final la **fabricación** del **algoritmo** que dará la **solución**.



George Pólya, matemático húngaro, autor del libro *How to solve it* [1]. En él, Pólya propone cuatro (4) pasos generales para la solución de un problema.



## 1. Entienda el Problema

Parece obvio pero es con frecuencia un gran obstáculo.

- 👉 ¿Entiende todas las palabras de la formulación del problema?
- 👉 ¿Qué le están pidiendo que encuentre?
- 👉 ¿Puede usted reformular el problema en sus propias palabras?
- 👉 ¿Hay suficiente información para encontrar la solución?

## 2. Diseña un Plan

Escoja una estrategia para resolver el problema.

**divida el problema en problemas más simples, elimina posibilidades, aproveche simetrías, suponga y verifique, etc.**

## 3. Implemente el plan

Más fácil que el paso 2, sólo requiere mucho cuidado en los detalles y paciencia.

## 4. Revise

Haga una pausa, revise y reflexione sobre el trabajo realizado.

## 1 Solución de Problemas

- Método de Polya

## 2 Algoritmos

### • Diagramas de Flujo

- Condicionales
- Ciclos/Iteraciones
- Contadores y Acumuladores

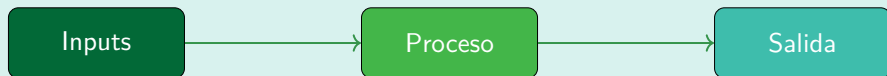
### • Pruebas de Escritorios

## 3 Bibliografía

## Definición

Recuerde que un algoritmo es un conjunto de instrucciones ordenadas, el cual es preciso, computable y finita, que paso a paso llevan a la solución de un problema.

Todo algoritmo tiene:





## ¿Cómo ingresar a la Universidad de Antioquia?

- 1 Comprar formulario de inscripción.
- 2 Elegir carrera.
- 3 Presentar examen.
- 4 Si no pasa, volver al paso 1.
- 5 Pagar matrícula.
- 6 Elegir materias.

## ¿Cómo dibujar una parabola en el plano cartesiano $(-10,10)$ ?

- 1 Asignar a  $x$  el valor de  $-10$ .
- 2 Asignar a  $y$  el valor de  $x^2$ .
- 3 Dibujar un punto en la coordenada  $(x, y)$ .
- 4 Sumar 1 a  $x$ .
- 5 Si  $x$  es menor o igual a  $10$ , vaya al paso 2.

- **¿Cuál es el objetivo buscando?**  
Calcular el área de un triángulo.

- **¿Cuál es el objetivo buscando?**  
Calcular el área de un triángulo.
- **¿Cuáles son los datos de entrada?**  
Base y altura.

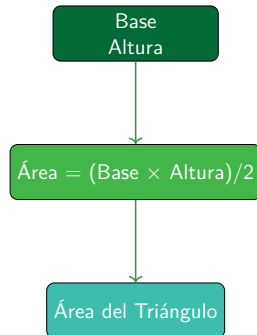
- **¿Cuál es el objetivo buscando?**  
Calcular el área de un triángulo.
- **¿Cuáles son los datos de entrada?**  
Base y altura.
- **¿Cuáles son los datos de salida?**  
Área de un triángulo.

- **¿Cuál es el objetivo buscando?**  
Calcular el área de un triángulo.
- **¿Cuáles son los datos de entrada?**  
Base y altura.
- **¿Cuáles son los datos de salida?**  
Área de un triángulo.
- **¿Qué cálculos/procesos deben de llevarse a cabo?**

$$A_{\Delta} = \frac{Base \times Altura}{2}$$

- **¿Cuál es el objetivo buscando?**  
Calcular el área de un triángulo.
- **¿Cuáles son los datos de entrada?**  
Base y altura.
- **¿Cuáles son los datos de salida?**  
Área de un triángulo.
- **¿Qué cálculos/procesos deben de llevarse a cabo?**

$$A_{\Delta} = \frac{Base \times Altura}{2}$$



# Data en un algoritmo

## Variable

Espacio de memoria al que se le da un nombre y que almacena un valor (un dato) que puede ser modificado por instrucciones del algoritmo.



# Data en un algoritmo

## Variable

Espacio de memoria al que se le da un nombre y que almacena un valor (un dato) que puede ser modificado por instrucciones del algoritmo.

## Tipos de variables

- Variables de entrada y salida.
- Variables auxiliares.
- **Constante:** un dato que no cambia.

# Data en un algoritmo

## Variable

Espacio de memoria al que se le da un nombre y que almacena un valor (un dato) que puede ser modificado por instrucciones del algoritmo.

## Tipos de variables

- Variables de entrada y salida.
- Variables auxiliares.
- **Constante:** un dato que no cambia.

Una variable puede representar un número decimal, un número entero, un arreglo de números o de caracteres, etc.

# Ejemplo de Algoritmo I

Se requiere diseñar un algoritmo que **calcule el número de meses que hay entre los años A y B.**

**Datos de entrada:** los años especificados (A y B).

**Datos de salida:** número total de meses transcurridos.

## Definición de variables:

- A: primer año
- B: segundo año
- years: años transcurridos
- months: meses transcurridos

# Ejemplo de Algoritmo II

## Algoritmo

- 1 Capturar valores de  $A$  y  $B$
- 2 Asignar a  $years$  la operación  $B - A$
- 3 Asignar a  $months$  la operación  $years \times 12$
- 4 Mostrar el valor de  $months$

# Diagramas de Flujo

## Definición

Representación **gráfica** de un **algoritmo** o proceso.

También conocido como **flujograma** o **diagrama de actividades**.

## Ejemplo

Se requiere diseñar un algoritmo que calcule el número de meses que hay entre dos años  $A$  y  $B$ .

## Solución Pseudo-código

- 1 Capturar valores de  $A$  y  $B$ .
- 2 Asignar a `years` la operación de  $B - A$ .
- 3 Asignar a `months` la operación  $\text{years} * 12$ .
- 4 Muestre el valor de `months`.

# Diagramas de Flujo

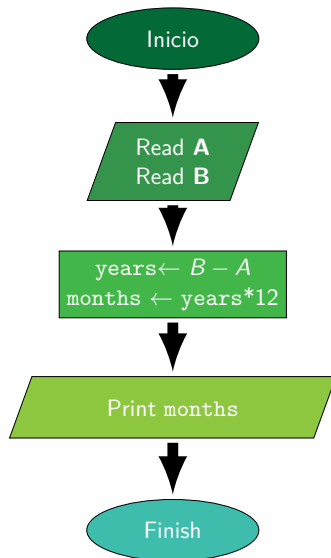
## Definición

Representación **gráfica** de un **algoritmo** o proceso.


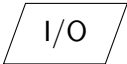
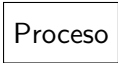
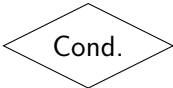

También conocido como **flujograma** o **diagrama de actividades**.

## Ejemplo

Se requiere diseñar un algoritmo que calcule el número de meses que hay entre dos años A y B.



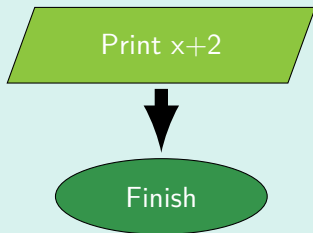
# Componentes de un Diagrama de Flujo.

Símbolo	Descripción
	Representa el punto de inicio o final del proceso.
	Representa operaciones de entrada/salida (leer o imprimir datos).
	Representa un proceso o cálculo que se realiza en el diagrama.
	Representa una decisión o condición que afecta el flujo (por ejemplo, "sí/no").
	Indican la secuencia entre dos secciones del algoritmo.

# Sintaxis Versus Semántica

## Sintaxis

Conjunto de reglas que determinan los símbolos y las combinaciones de estos, que son válidos en un lenguaje. Hace uso de **operaciones primitivas**.



## Semántica

Es el **significado** de las expresiones permitidas por la sintaxis de los algoritmos y programas.

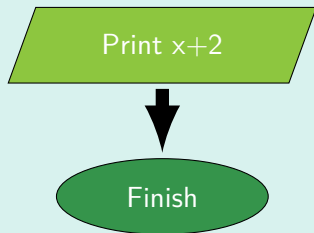
*Determina si un número es primo.*



# Sintaxis Versus Semántica

## Sintaxis

Conjunto de reglas que determinan los símbolos y las combinaciones de estos, que son válidos en un lenguaje. Hace uso de **operaciones primitivas**.



## Semántica

Es el **significado** de las expresiones permitidas por la sintaxis de los algoritmos y programas.

*Determina si un número es primo.*

**Suele requerirse un algoritmo, a parte, para obtener el resultado correspondiente.**

# Operaciones Primitivas

- Son las operaciones permitidas en un algoritmo para los cuales **no** es necesario hacer su propio algoritmo.
- En **lenguajes de programación**, son las instrucciones que el traductor entiende, que es capaz de traducir.

# Operaciones Primitivas

- Son las operaciones permitidas en un algoritmo para los cuales **no** es necesario hacer su propio algoritmo.
- En **lenguajes de programación**, son las instrucciones que el traductor entiende, que es capaz de traducir.

Operadores aritméticos			
Operador	Nombre	Ejemplo	Resultado
+	Suma	1+3	4
-	Resta	7-10	-3
*	Multiplicación	3*5	15
/	División	8/5	1.6
//	División entera	8//5	1
%	Modulo (residuo)	15 %7	1
**	Potencia	2**3	8

# Operaciones Primitivas

- Son las operaciones permitidas en un algoritmo para los cuales **no** es necesario hacer su propio algoritmo.
- En **lenguajes de programación**, son las instrucciones que el traductor entiende, que es capaz de traducir.

Operadores relacionales			
Operador	Nombre	Ejemplo	Resultado
>	Mayor	1>3	False
>=	Mayor o igual	2>=1	True
<	Menor	3<3	False
<=	Menor o igual	3<=3	True
!=	Diferente	13!=4	True
==	Igual	0==1	False

# Jerarquía de Operaciones

## Orden de Evaluación

Las operaciones se evalúan siguiendo esta jerarquía:

- ➊ **Paréntesis:** Se evalúan primero.
- ➋ **Exponentes:** Potenciación y raíces.
- ➌ **Multiplicación y División:** Se evalúan de izquierda a derecha.
- ➍ **Suma y Resta:** Se realizan al final.
- ➎ **Operadores Relacionales y Lógicos:** Se evalúan después de las operaciones aritméticas.

# Jerarquía de Operaciones

## Orden de Evaluación

Las operaciones se evalúan siguiendo esta jerarquía:

- ➊ **Paréntesis:** Se evalúan primero.
- ➋ **Exponentes:** Potenciación y raíces.
- ➌ **Multiplicación y División:** Se evalúan de izquierda a derecha.
- ➍ **Suma y Resta:** Se realizan al final.
- ➎ **Operadores Relacionales y Lógicos:** Se evalúan después de las operaciones aritméticas.

## Ejemplo

Realizar la operación:

$$\underbrace{3 + 4 \times 2^2 - \frac{6 - 2}{2 - 5}}_{\text{Forma Matemática}} \Leftrightarrow \underbrace{3 + 4 * (2 ** 2) - (6 - 2) / (2 - 5)}_{\text{En Lenguaje de Programación.}}$$

# Operador Asignación.

- Una expresión por si sola no indica que hacer con el resultado

$$((x * 3) - 5 * x - 2) / (a \% d) * k$$

- El operador de asignación permite asociar el resultado de una expresión a una variable.

$$z \leftarrow ((x * 3) - 5 * x - 2) / (a \% d) * k$$

## ¡Importante!

El operador de asignación es el único que **cambia de estado** de una variable.

# Ejemplo: Calculo de la distancia entre dos puntos I

## Enunciado

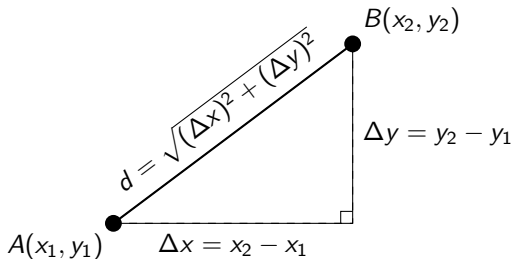
Manuel está estudiando geometría y necesita una manera rápida y confiable de verificar sus cálculos al determinar la distancia entre dos puntos en el plano cartesiano. Para facilitar este proceso, ha decidido desarrollar un pequeño programa que realice automáticamente este cálculo y le ayude a confirmar la precisión de sus resultados.

## A considerar

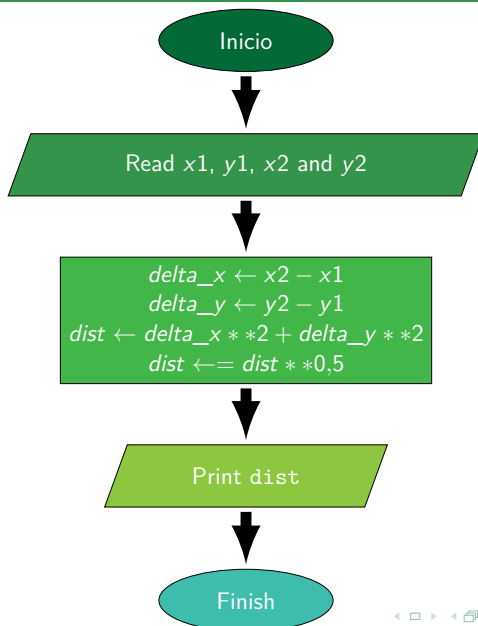
- **Datos de entrada:** Coordenadas de los dos puntos  $(x_1, y_1)$  y  $(x_2, y_2)$
- **Datos de Salida:** Distancia.



## Ejemplo: Calculo de la distancia entre dos puntos II



## Ejemplo: Calculo de la distancia entre dos puntos III

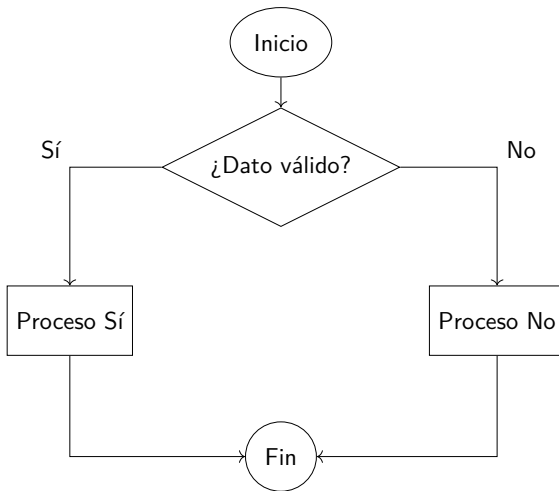


Los condicionales en un diagrama de flujo se utilizan para representar decisiones, es decir, puntos en los que el flujo puede bifurcarse en función de si se cumple o no una determinada condición. Normalmente se representan mediante un **rombo**.

## Ejemplos de uso:

- Verificar si un número es mayor que cero.
- Comprobar si un usuario ingresó datos válidos.
- Decidir entre dos caminos (Sí/No, Verdadero/Falso).

## Ejemplo gráfico:



## Ejemplo, cambio de moneda

Juan desea crear un programa para realizar conversiones de moneda. El usuario deberá ingresar un monto en una divisa origen y seleccionar la divisa destino. El programa verificará mediante condiciones si el monto es mayor que cero y si la divisa destino es válida (por ejemplo, USD, EUR o GBP). Si ambas condiciones se cumplen, se procederá a calcular el cambio usando la tasa de conversión correspondiente; de lo contrario, se mostrará un mensaje de error indicando que los datos ingresados no son correctos.

# Ejercicio

Juan necesita crear un programa que verifique si un año ingresado es bisiesto. Para ello, el programa debe pedir al usuario que introduzca un año. Luego, mediante condiciones, se evaluará si:

- El año es divisible por 4.
- Si el año es divisible por 100, también debe ser divisible por 400 para ser considerado bisiesto.

Si se cumplen las condiciones anteriores, el programa indicará que el año es bisiesto; de lo contrario, mostrará un mensaje indicando que el año no es bisiesto.

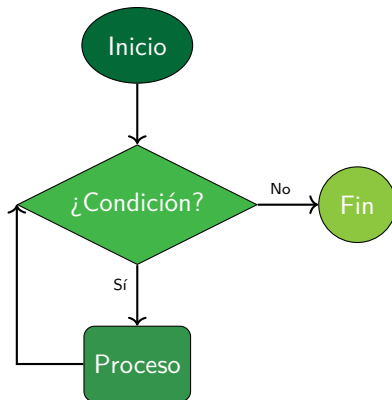
## Pasos

- 1 Realiza un pseudo-código, con sus palabras en lista de forma secuencial los pasos a seguir, desde el ingreso de las variables de entrada hasta la salida salidas correspondientes.
- 2 Lleva esto a un diagrama de flujo con los simbolos correspondientes.

## Descripción

Las iteraciones (o ciclos) permiten repetir un conjunto de operaciones hasta que se cumpla una condición. En un diagrama de flujo, se representan mediante un nodo de decisión (rombo) y flechas que retroalimentan el proceso. Esto es fundamental, por ejemplo, para recorrer listas o repetir cálculos hasta alcanzar un resultado deseado.

## Ejemplo gráfico:





# Ejemplo

## Contador primeros $N$ enteros.

Desarrolla un algoritmo y su correspondiente diagrama de flujo que permita calcular la suma de los primeros  $N$  números enteros positivos, donde  $N$  es un valor ingresado por el usuario. El programa debe realizar las siguientes tareas: Mostrar el resultado obtenido.

El diagrama de flujo debe reflejar claramente cada uno de estos pasos, incluyendo el proceso de validación y la iteración.

# Variables: Contadores y Acumuladores

## Variable Contador

Sirve para contar cuántas veces se da una condición o evento.

- Cuántas veces se repite un ciclo.
- Cuántas veces una variable no supera un límite.

$$\text{count} \leftarrow \text{count} + 1$$

## Variable Acumulador

Sirve para acumular los resultados de una operación que se repite.

- Acumular la suma iterativa de varios elementos.
- Acumular las multiplicaciones para calcular un factorial.

$$\text{sum} \leftarrow \text{sum} + \text{resul}$$

# Ejemplo

## Promedio números impares.

Haga un algoritmo que reciba números repetidamente y que en cada iteración muestre el promedio de los números impares ingresados hasta el momento. El algoritmo debe finalizar cuando el número ingresado sea el cero.

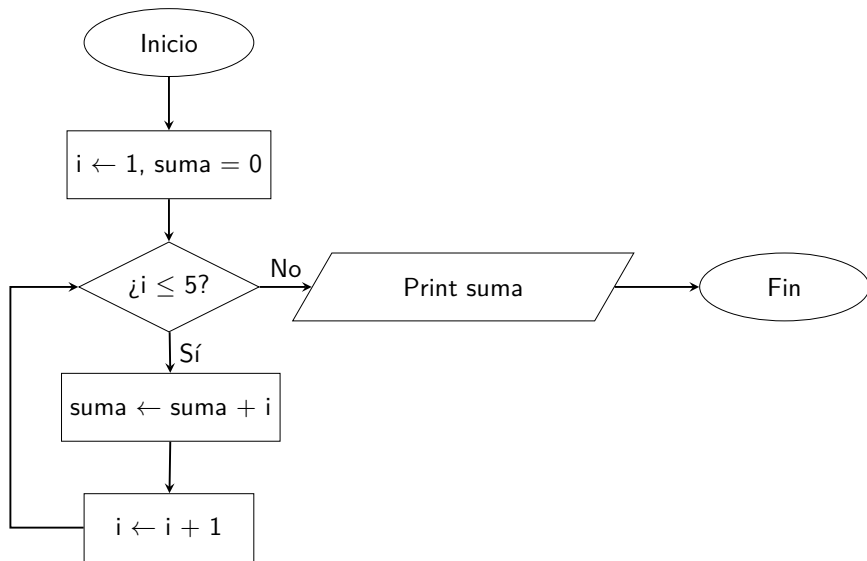
## Definición:

La prueba de escritorio es una técnica manual para simular la ejecución de un algoritmo paso a paso, con el objetivo de verificar su correcto funcionamiento y detectar errores lógicos antes de su implementación.

## Pasos para realizar una prueba de escritorio:

- 1 Escribir el algoritmo de forma clara.
- 2 Asignar valores iniciales a las variables.
- 3 Simular cada iteración y actualizar las variables.
- 4 Verificar que el resultado final sea el esperado.

# Prueba de Escritorio II



**Ejemplo:** Sumar los primeros 5 números naturales.

Iteración	i	i+1	suma	suma + i
Inicial	1	–	0	–
1	2	2	1	$0 + 1 = 1$
2	3	3	3	$1 + 2 = 3$
3	4	4	6	$3 + 3 = 6$
4	5	5	10	$6 + 4 = 10$
5	6	6	15	$10 + 5 = 15$

El resultado final de la suma es 15.

## Número de Dígitos de un Número

Desarrolla un algoritmo que permita determinar el número de dígitos que tiene un número entero positivo ingresado por el usuario. Deberás realizar lo siguiente:

- 1 Diagrama de Flujo: Elabora el diagrama de flujo que represente cada paso del algoritmo. Recuerda incluir los procesos de entrada, la iteración o condiciones necesarias para contar cada dígito y la salida del resultado.
- 2 Prueba de Escritorio: Realiza la prueba de escritorio del algoritmo, simulando la ejecución con un ejemplo específico (por ejemplo, con el número 12345) y verifica que el resultado obtenido sea correcto.

El programa debe:

- Pedir al usuario un número entero positivo.
- Contar, mediante un ciclo, la cantidad de dígitos del número.
- Mostrar el resultado al usuario.

## 1 Solución de Problemas

- Método de Polya

## 2 Algoritmos

- Diagramas de Flujo
  - Condicionales
  - Ciclos/Iteraciones
  - Contadores y Acumuladores
- Pruebas de Escritorios

## 3 Bibliografía





George Polya.

How to solve it: A new aspect of mathematical method.

In *How to solve it*. Princeton university press, 2014.