



# ¿QUÉ ES UN ALGORITMO?

“Un algoritmo es una secuencia lógica de pasos para solucionar un problema.”

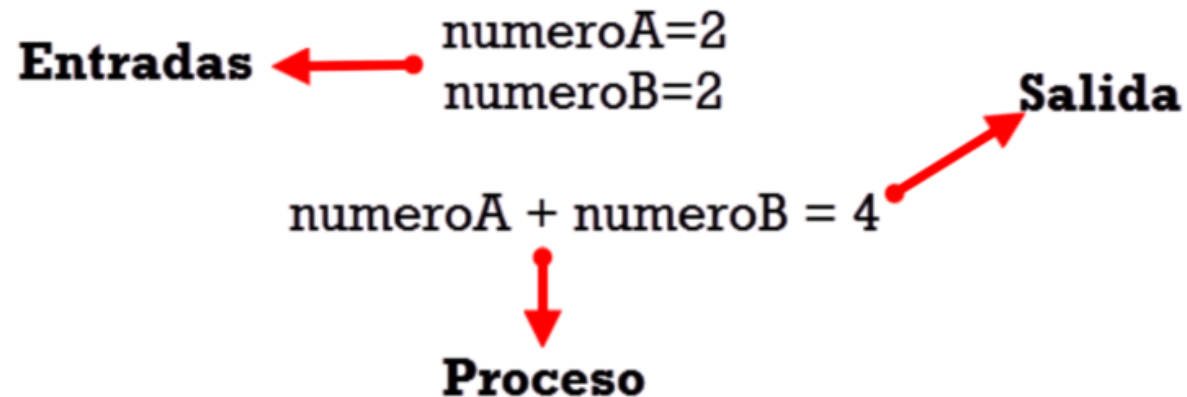
Todos los algoritmos deben cumplir las siguientes reglas:

- **Deben ser precisos:** Tener un paso a paso lógico y puntual.
- **Deben ser definidos:** El algoritmo debe comportarse de la misma manera siempre.
- **Debe ser finito:** El algoritmo debe tener un número finito de pasos, debe terminar en algún momento.

# ¿QUÉ ES UN ALGORITMO?

Un algoritmo se compone de 3 partes: Entrada, proceso y salida

- **Entrada:** Corresponde a los datos que el algoritmo recibe.
- **Proceso:** Equivalen a las acciones que se realizan sobre los datos de entrada.
- **Salida:** El resultado de las acciones sobre los datos de entrada.



# ALGORITMOS DE LA VIDA DIARIA

Estos tipos de algoritmos son aquellos que nos ayudan a resolver problemas cotidianos, por lo regular los realizamos casi sin darnos cuenta que seguimos una metodología para resolverlos.

Ej: Algoritmo para cepillarnos.

“Es importante recalcar que se puede dar solución a un problema de muchas formas”

- 1. INICIO
- 2. ir al baño
- 3. tomar el cepillo de dientes
- 4. tomar la crema de dientes
- 5. poner crema de dientes en el cepillo
- 6. cepillarse los dientes durante el tiempo deseado
- 7. escupir
- 8. enjuagarse la boca.
- 9. guardar el cepillo y la crema
- 10. FIN

# ALGORITMOS DE LA VIDA DIARIA

Ej: Algoritmo para cambiar la llanta de un carro.

- 1 Inicio.
- 2 Traer gato.
- 3 Levantar el coche con el gato.
- 4 Aflojar tornillos de las llantas.
- 5 Sacar los tornillos de las llantas.
- 6 Quitar la llanta.
- 7 Poner la llanta de repuesto.
- 8 Poner los tornillos.
- 9 Apretar los tornillos.
- 10 Bajar el gato.
- 11 Fin

# ALGORITMO COMPUTACIONAL

Los algoritmos computacionales permiten definir los procesos para dar solución a problemáticas mediante operaciones lógicas en un computador.

Estos a diferencia de los anteriores debes ser desarrollados siguiendo una metodología definida para la solución de problemas. (enfoque de la solución, sintaxis...)



# LENGUAJES ALGORÍTMICOS.

Ya se había mencionado que un lenguaje es la capacidad que tenemos para expresarnos, o comunicarnos entre nosotros, los lenguajes algorítmicos cumplen la misma función.

Definen la forma en la que nos comunicaremos por medio de algoritmos con el objetivo de que sea la máquina la que nos entienda.





# LENGUAJES ALGORÍTMICOS.

Los algoritmos pueden ser resueltos mediante la aplicación de diferentes técnicas, en este caso podemos resaltar 4 lenguajes que nos permitirán describir los pasos de un algoritmo de forma más detallada y estructurada.

- Lenguaje Natural
- Lenguaje de Diagrama de Flujo
- Lenguaje Seudocódigo
- Lenguaje de programación

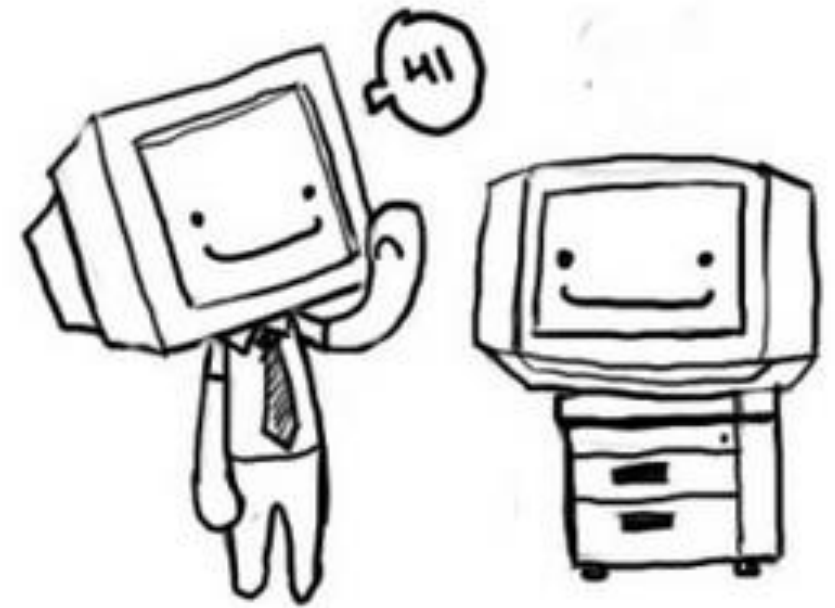


# LENGUAJE NATURAL

Este lenguaje nos permite describir la secuencia lógica de pasos de una manera mas natural o informal, se usa un vocabulario cotidiano al describir los pasos de forma simple sin tecnicismos.

Este lenguaje lo utilizamos en los ejemplos anteriores al describir los algoritmos para cepillarnos y cambiar la llanta de un carro.

Ej. Haga un **Algoritmo** para preparar limonada.



# LENGUAJE NATURAL

Ej. Algoritmo para preparar limonada.

- 1. Alistar los limones, el azúcar, el agua, una cuchara, un vaso y la jarra para preparar la limonada.
- 2. Llenar la jarra con agua
- 3. Cortar los limones por la mitad
- 4. Exprimir los limones en la jarra con agua
- 5. Adicionar azúcar al gusto
- 6. Usar la cuchara para revolver la limonada
- 7. Servir en el vaso

# SEUDOCÓDIGO.

El pseudocódigo cumple la misma función pero orientados a definir la solución de un problema de una manera más precisa y buscando definiciones formales, generalmente usados para la elaboración de fórmulas o problemas a resolver mediante algoritmos computacionales.

# SEUDOCÓDIGO.

El pseudocódigo debe cumplir con las siguientes características:

- Ser precisos y definidos.
- Evitar varias interpretaciones (ambigüedad)
- Usar términos formales pero familiares al sentido común
- Eliminar instrucciones innecesarias.

Cada lenguaje de programación define la manera en la que termina una sentencia o fin de línea, el caso mas popular es finalizar la sentencia con “;” (aunque otros lenguajes no lo requieren)

En nuestro caso todos los algoritmos en pseudocódigo los finalizaremos con ;

# SEUDOCÓDIGO.

## Reglas básicas:

- Se debe delimitar con las palabras INICIO y FINAL, estas determinan cuando inicia y termina el algoritmo.
- Toda la lógica debe estar encerrada entre INICIO y FINAL.
- Las variables que se utilicen en el algoritmo deben estar previamente declaradas de lo contrario no se podrán utilizar.
- Se debe indicar el tipo de dato al que pertenece cada variable.

INICIO

Declaración de variables;  
Expresiones y operaciones;

FINAL

# ¿QUÉ ES UNA VARIABLE?

Una variable es un contenedor que puede almacenar información y puede cambiar en el tiempo pues su contenido puede variar.

básicamente se puede definir como un nombre que identifica una dirección de memoria.

Por lo regular las variables se componen de un identificador y un tipo de datos que lo acompaña.

`<tipoDato> identificador`

Ej.:

- numerico edad
- texto nombre
- int num1
- var salario

# ¿CÓMO NOMBRAR UNA VARIABLE?

Los identificadores representan la forma correcta de definir nombres de variables, para esto como estándar se debe tener en cuenta lo siguiente:

- El primer carácter debe ser un caracter alfabético (a...z, A...Z) o \$, \_
- Después del primer caracter pueden ir caracteres alfanuméricos (a...z, A...Z) o \$, \_, (0...9)
- Los identificadores no pueden ser palabras reservadas del lenguaje
- Se recomienda aplicar la regla camelCase
- Los identificadores no pueden tener espacios, signos de puntuación, tildes ni otro caracter diferente a los mencionados.



# ¿QUÉ ES UN TIPO DE DATO?

Como se mencionó, las variables corresponden a contenedores de memoria donde se almacenarán valores.

Esos valores deben estar asociados a un tipo de dato específico, en general se tienen los siguientes tipos de datos:

- **Numéricos** (enteros, decimales)
- **Textos** (un carácter o cadena de caracteres)
- **Lógicos** (verdadero o falso)

Ej:      entero edad  
          texto nombre

# OPERADORES ARITMÉTICOS.

Estos operadores corresponden a los usados para labores académicas cotidianas tales como procesos de sumas, restas, división, multiplicación, modulo, incremento, decremento, su aplicación se puede evidenciar en la siguiente tabla.

Operador	Uso	Descripción
+	Operador1 + operador2	<b>Suma</b> (Operador1 mas Operador2)
-	Operador1 - operador2	<b>Resta</b> (Operador1 menos Operador2)
*	Operador1 * operador2	<b>Multiplicación</b> (Operador1 por Operador2)
/	Operador1 / operador2	<b>División</b> (Operador1 dividido Operador2)
%	Operador1 % operador2	<b>Módulo-residuo</b> (Operador1 Modulo Operador2)

# PRECEDENCIA DE OPERADORES.

La precedencia de operadores indica la forma correcta de resolver una operación matemática en caso de que no se use el operador de agrupación (paréntesis “()”), para eso se utilizan los operadores aritméticos vistos anteriormente.

Se resuelven primero los operadores de mayor precedencia, en caso de tener igual precedencia se resuelven de izquierda a derecha

OPERADOR	DESCRIPCION
()	Operador de agrupación
* / %	Operadores aritméticos (producto, división, módulo)
+ -	Operadores aritméticos (suma, resta)

# PRECEDENCIA DE OPERADORES.

Ej: Resolver  $10/5+6-2+3*8*1-12/2+9-7*4$

$$\begin{array}{r} 10/5 + 6 - 2 + 3*8*1 - 12/2 + 9 - 7*4 \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ 2 + 6 - 2 + 24 - 6 + 9 - 28 \\ 8 - 2 + 24 - 6 + 9 - 28 \\ 6 + 24 - 6 + 9 - 28 \\ 30 - 6 + 9 - 28 \\ 24 + 9 - 28 \\ 33 - 28 \\ 5 \end{array}$$

# SEUDOCÓDIGO.

Ej: pseudocódigo del algoritmo para hacer limonada.

INICIO

materiales limones, azucar, agua, cuchara, jarra;

llenar jarra con agua;

cortar limones;

exprimir limones en la jarra;

adicionar azucar;

revolver con cuchara;

servir limonada;

FINAL

# SEUDOCÓDIGO.

Ej: pseudocódigo del algoritmo para hacer limonada.

Algoritmo pseudocódigo	Algoritmo lenguaje natural
INICIO materiales limones, azucar, agua, cuchara, jarra; llenar jara con agua; cortar limones; exprimir limones en la jarra; adicionar azucar; revolver con cuchara; servir limonada; FINAL	1. Alistar los limones, el azúcar, el agua, una cuchara, un vaso y la jarra para preparar la limonada. 2. Llenar la jarra con agua 3. Cortar los limones por la mitad 4. Exprimir los limones en la jarra con agua 5. Adicionar azúcar al gusto 6. Usar la cuchara para revolver la limonada 7. Servir en el vaso

Nótese que este algoritmo cambia el estilo de lenguaje natural a un lenguaje un poco más formal, tratando de reducir las frases explicativas (alistar los limones y el azucar) y reemplazándolo por algo más preciso (materiales limones, azucar)

# SEUDOCÓDIGO.

Ej: pseudocódigo del algoritmo para sumar 2 números.

INICIO

    numerico a,b,result;

    result=a + b;

    imprimir result;

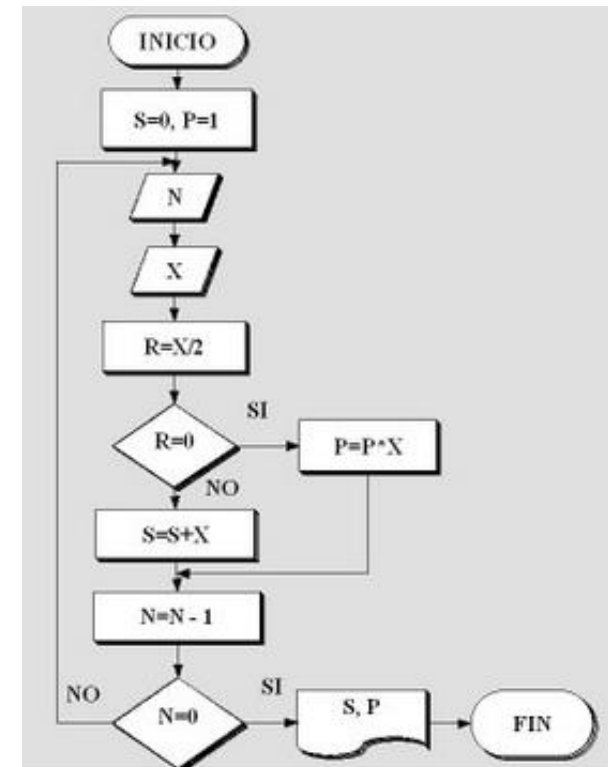
FINAL



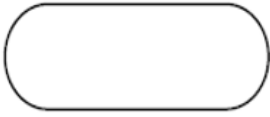

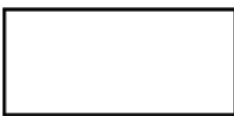

# DIAGRAMA DE FLUJO.

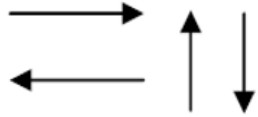
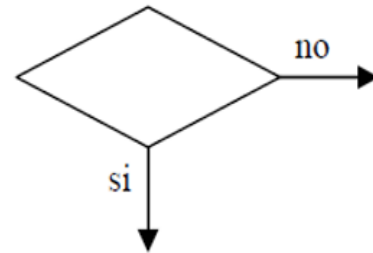
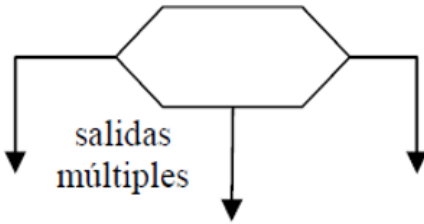

Representan los algoritmos por medio de símbolos que facilitan el entendimiento de la solución o proceso planteado.

Pueden existir muchas representaciones o símbolos.



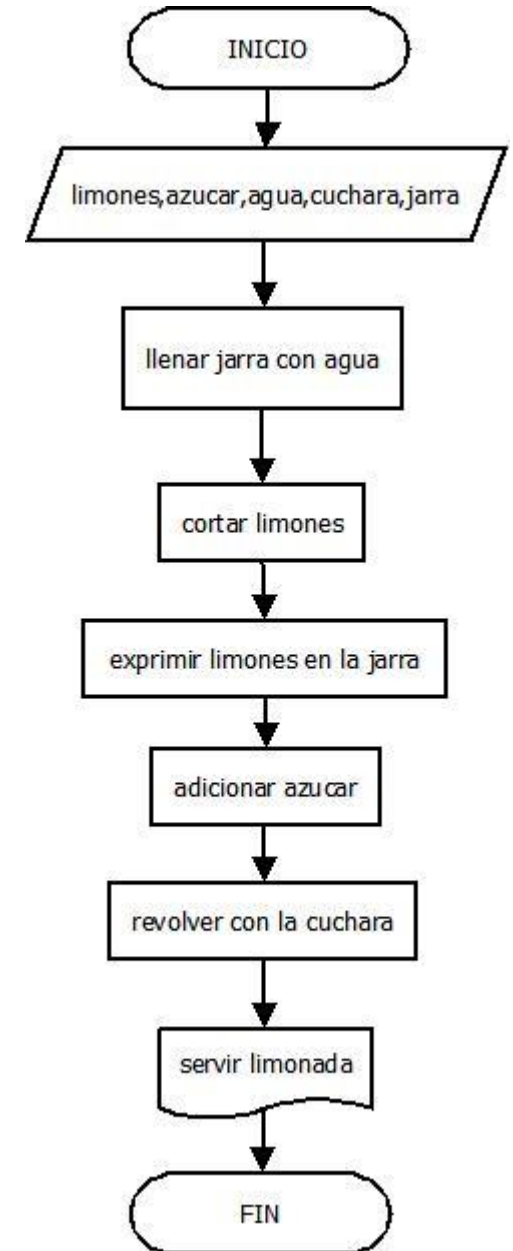
# DIAGRAMA DE FLUJO.

SIMBOLO	DESCRIPCION
	Representa el inicio y el final de un diagrama de flujo
	Símbolo usado para representar cualquier tipo de entrada o lectura de datos y también puede ser usado para la asignación de valores.
	Utilizado para representar cualquier operación o proceso lógico, por lo regular usado para asignar valores, realizar operaciones matemáticas.
	Este símbolo se usa para representar las entradas o salidas del sistema, permite imprimir el resultado de un proceso

SIMBOLO	DESCRIPCION
	Las flechas representan el flujo del sistema, la dirección indica cual es el sentido al momento de ejecutar o seguir el diagrama.
	Representa una decisión u operación de comparación entre datos, en este símbolo define una condición y dependiendo del resultado se determina cuál de los caminos debe tomar el sistema.
	Representa decisiones múltiples o bifurcación, dependiendo del resultado de una comparación que depende de un dato de entrada el sistema tomará uno de los caminos propuestos.
	Este símbolo permite conectar o enlazar dos partes de un diagrama (en caso de que sea muy grande y no lo podamos ver completamente) se usa mediante un conector de entrada y otro de salida.

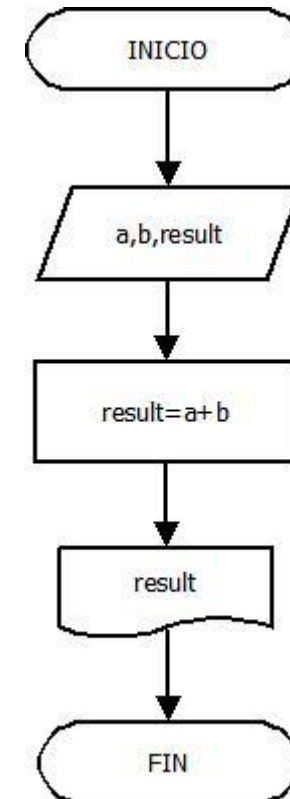
# DIAGRAMA DE FLUJO.

Ej. Realizar mediante diagrama de flujo el algoritmo de hacer limonada



# DIAGRAMA DE FLUJO.

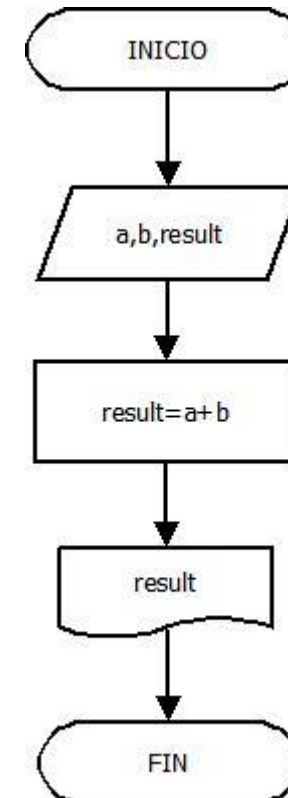
Ej. Realizar mediante diagrama de flujo un algoritmo que sume 2 números e imprima el resultado.



# DIAGRAMA DE FLUJO.

Ej. Realizar mediante diagrama de flujo un algoritmo que sume 2 números e imprima el resultado.

INICIO  
  numerico a,b,result;  
  result=a + b;  
  imprimir result;  
FINAL



# PRUEBA DE ESCRITORIO.

Cuando se realiza un algoritmo, lo ideal es verificar el proceso para asegurarnos que cumple con lo esperado, para esto se aplica una técnica llamada “Prueba de Escritorio”.

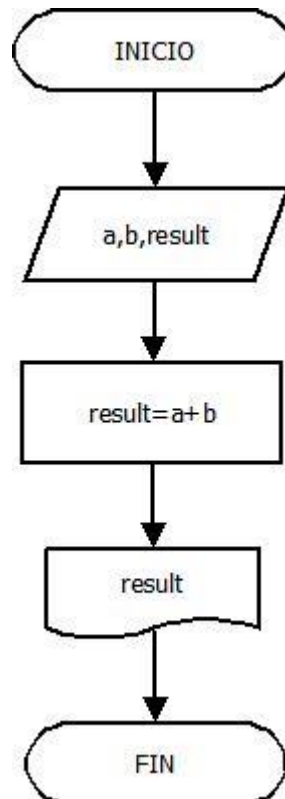
La prueba de escritorio consiste en un seguimiento paso a paso de la ejecución del algoritmo, a medida que el algoritmo avanza se hacen uso de variables e instrucciones que dan vida a nuestra solución, la prueba de escritorio valida este proceso para determinar cuáles son los valores paso a paso y verificar así el funcionamiento deseado.



# PRUEBA DE ESCRITORIO.

Ej: prueba de escritorio del algoritmo suma de 2 números:

INICIO  
  numerico a,b,result;  
  result=a + b;  
  imprimir result;  
FINAL



a	b	result
2	2	4
1	4	5



# PRUEBA DE ESCRITORIO.

Ej: realice la prueba de escritorio del siguiente algoritmo:

INICIO

ENTERO a, b, c;

a = 2;

b = a - 1;

c = (a \* b) - 2;

b = a + ( 3 + b );

a = b - 2 - c;

FIN

# PRUEBA DE ESCRITORIO.

Ej: realice la prueba de escritorio del siguiente algoritmo:

INICIO

ENTERO a, b, c;

a = 2;

b = a - 1;

c = (a \* b) - 2;

b = a + (3 + b);

a = b - 2 - c;

FIN

a	b	c	Descripción
2			Primera expresión, donde se asigna a la variable "a" el valor de 2
	2-1=1		El valor del resultado es almacenado en la variable b
		(2*1)-2=0	En esta expresión, se usan los últimos valores de a y b para darle valor a c
	2+(3+1)=6		Se realiza la operación con los valores de las variables a y b, reasignando los datos en b
6-2-0=4			Finaliza reasignando a la variable "a" el valor obtenido.