



## Algoritmos

Es el momento de realizar el algoritmo que resolverá el problema!  
En esta unidad, vamos a explicarles **qué es un algoritmo, sus características y componentes.**

### Definición de algoritmo

Aunque existen varias definiciones de algoritmos en la bibliografía y también en internet, nosotros te presentamos una definición que consideramos que abarca las principales características.

Entonces...

? ¿Qué es un algoritmo?

Es un conjunto de indicaciones **finitas y ordenadas** de forma **lógica** que permite la resolución de un problema dado.

Se atribuye el término al matemático y astrónomo persa Musa al-Juarismi, quien vivió entre los años 780 y 850.

Con seguridad, habrás leído e interpretado algoritmos sin que supieras que se trataba de ellos, por ejemplo:

- Un manual de instrucciones para colgar una televisión en la pared.
- Una receta de cocina para preparar un postre.
- Las directivas de un jefe a su empleado.
- Las indicaciones de un GPS para llegar a destino.
- Los pasos para calcular el cociente entre dos números enteros.

La importancia de los algoritmos en la informática es trascendental, dado que la programación tiene como objetivo la implementación de ellos en una computadora, para que sea ésta quien los ejecute y resuelva determinado problema, sin embargo, éstos trascienden la disciplina informática, pudiendo encontrarlos en la matemática o la lógica.

### Componentes de un algoritmo

Como explicamos antes, una de las características que tiene que cumplir un algoritmo es que se ejecute a lo largo de un tiempo finito. Es por eso que durante la ejecución podemos encontrarnos con distintos **componentes**, los cuales presentamos a continuación:



- **Entrada:** Representa la información que necesitan los pasos del algoritmo para ejecutarse.
- **Proceso:** Es el conjunto de acciones elementales, organizadas en el tiempo.
- **Salida:** Es el resultado que se obtiene ejecutando los pasos del algoritmo con los datos de entrada.



### Características de un algoritmo.

Para que un algoritmo pueda ser considerado como tal, debe cumplir con las siguientes claves:

#### *Es Finito:*

Todo algoritmo debe tener número de pasos que permita llegar a un final.

#### *Es Preciso:*

Indicando el orden de realización de cada paso.

#### *Debe estar Definido:*

Obteniéndose el mismo resultado si se repite el proceso con los mismos datos de entrada.







### Formas de representación

Podemos expresar los algoritmos en un lenguaje natural cuya intuición y comprensión resultan convenientes, pero con la desventaja de que son imprecisos.

La forma precisa y sin ambigüedades es representarlos mediante dos maneras formales: como **diagrama de flujo** o como **pseudocódigo**.

Un **diagrama de flujo** es una representación gráfica de un algoritmo. Seguramente te resulta familiar, pues es usado también para describir procesos industriales o de negocio. Son bastante convenientes al principio, dado que son bastante fáciles de entender, gracias a que presentan las instrucciones de una manera gráfica.

La clave para entenderlos es comprender que existen diferentes componentes, cada uno con una forma distinta. La unión de ellos permite formalizar una solución:

Forma	Descripción
	Delimitador. Representa el comienzo o la finalización de la secuencia de instrucciones.
	Entrada. Representa la adquisición de datos por medio de un periférico, como el teclado.
	Proceso. Representa una operación o una tarea.
	Salida. Representa la visualización de los resultados por medio de un periférico, como la pantalla.
	Condición. Representa un interrogante cuya evaluación debe dar únicamente VERDADERO o FALSO.
	Línea de flujo. Representa la dirección de la secuencia de instrucciones.

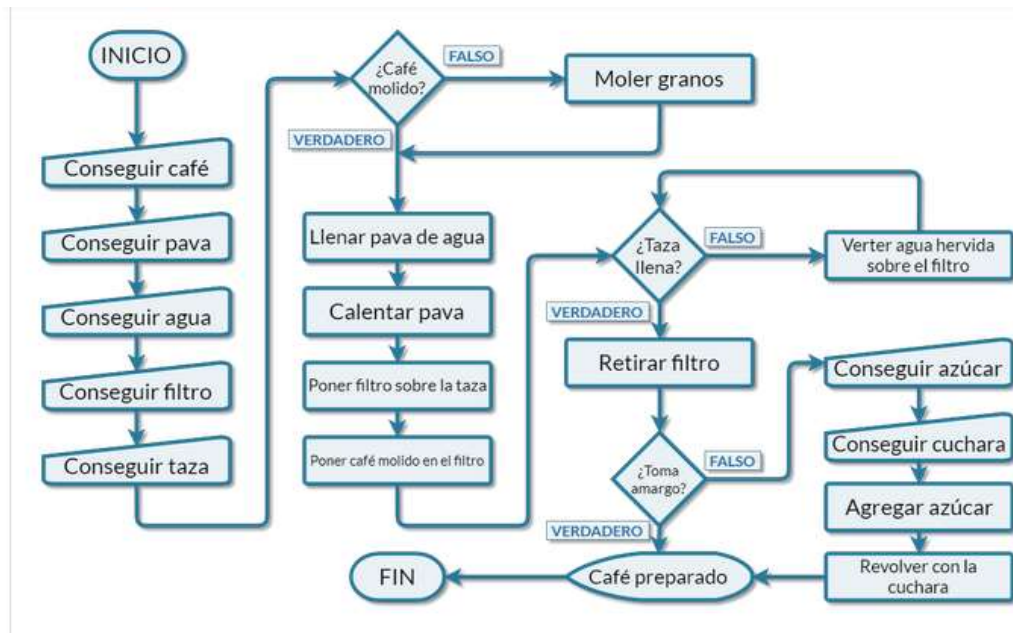
La desventaja de los diagramas de flujo es que una computadora no puede interpretarlos directamente. Son simples representaciones gráficas para los seres humanos.

La manera más cercana a la máquina de representar un algoritmo, pero todavía interpretable fácilmente por una persona, es mediante pseudocódigo.

Un **pseudocódigo** es una descripción de las instrucciones de manera tal de ser muy similar al formato que se obtiene al utilizar un lenguaje de programación. El punto es que el pseudocódigo no tiene un estándar de reglas sintácticas a seguir, sino que es constituido por convención por uno o más programadores para tener una solución abstracta del problema, algo así como una base para luego transcribir ese algoritmo en un lenguaje de programación real.

### Ejemplos

El siguiente, es un diagrama de flujo que muestra cómo preparar café:



A continuación se detalla un posible pseudocódigo correspondiente al problema de preparar café, cuyo diagrama de flujo ya se detalló en la ilustración anterior:

1	INICIO
2	CONSEGUIR café
3	CONSEGUIR pava
4	CONSEGUIR agua
5	CONSEGUIR filtro
6	CONSEGUIR taza
7	SI el café no está molido MOLER
8	café
9	LLENAR pava CON agua CALENTAR
10	pava
11	PONER filtro EN taza PONER café
12	EN filtro
13	MIENTRAS la taza no esté llena VERTER agua
14	EN filtro
15	RETIRAR filtro
16	SI no toma amargo
17	CONSEGUIR azúcar
18	CONSEGUIR cuchara
19	AGREGAR azúcar REVOLVER CON
20	cuchara SERVIR café
21	FIN
22	



## Tipo de errores

Los errores que se pueden cometer se dividen en dos tipos:

- Errores en tiempo de compilación y,
- Errores en tiempo de ejecución.

Los **errores en tiempo de compilación** son aquellos en los que, cuando el compilador de código detecta que algo no está bien, acusa un mensaje donde describe tal error, haciendo que el programa no pueda si quiera ejecutarse. PSeInt no es compilado, sino interpretado, por lo que la definición anterior no es del todo cierta para este tipo de lenguaje, por ello es que en este caso usaremos para el término **errores de sintaxis**.

Para resumir, los errores de sintaxis, como, por ejemplo, que a la instrucción:

Escribir "Hola Mundo!";

le quitemos el punto y coma, harán que no podamos siquiera ejecutar el programa.

Los **errores en tiempo de ejecución** son aquellos que se producen cuando el programa ya ha sido ejecutado sin errores sintácticos. En determinado momento, el programa detectará un error y no podrá continuar, ocasionando que finalice de forma abrupta.

Hay errores que no impiden que el programa se ejecute, pero provocan que los resultados quizá sean inesperados. Son **errores lógicos**, y son los más difíciles de detectar, dado que se requiere volver a analizar y probar el código en busca de la falla.

A lo largo de los temas que se irán desarrollando, veremos los errores más típicos que se pueden cometer. Recuerden que del error es de donde más y mejor se aprende.

## Comentar códigos

Una muy buena práctica a la hora de programar es **comentar el código**. Esto significa añadir notas que ayuden a entender alguna instrucción compleja o para describir tareas pendientes. No son tenidos en cuenta por el intérprete, solo sirven para el programador.

Hay dos tipos de comentarios: de **línea** y de **bloque**. PSeInt solo cuenta con el primero. Un comentario de línea se inserta con una doble barra, sin espacios, de esta manera:

//. Veremos que el texto a continuación de allí se pone en gris, indicando que no será tenido en cuenta por el intérprete. Desde un // en adelante, se trata de un comentario.

A continuación, se muestra un ejemplo con algunos comentarios sencillos:



## Código comentado

### 1 Algoritmo comentado

- 2 Definir numero1 Como Entero; //Declaración de Variable
- 3 Definir numero2 Como entero; //Declaración de Variable
- 4 Escribir "SUMADOR";
- 5 Escribir "Ingrese el primer número entero:";
- 6 Leer numero1; //Capturo el primer número
- 7 Escribir "Ingrese el segundo número entero:";
- 8 Leer numero2; //Capturo el segundo número
- 9 //Puedo poner un comentario que ocupe una línea completa
- 10 Escribir "El resultado es " ,(numero1 + numero2);
- 11 FinAlgoritmo