

# Módulo 1 - PROGRAMACIÓN BÁSICA EN JAVA

## Algoritmos ¿Que es un algoritmo?

Un algoritmo es una secuencia lógica y finita de pasos que permite solucionar un problema o cumplir con un objetivo.

Los algoritmos deben ser precisos e indicar el orden lógico de realización de cada uno de los pasos, debe ser definido y esto quiere decir que si se ejecuta un algoritmo varias veces se debe obtener siempre el mismo resultado, también debe ser finito o sea debe iniciar con una acción y terminar con un resultado o solución de un problema.

Cuando se elabora un algoritmo se debe tener en cuenta lo siguiente:

- Tener claro cuál es el problema que va a solucionar.
- Establecer un objetivo que permita medir la solución del problema.
- Elaborar un algoritmo que solucione el problema.
- Realizar pruebas al algoritmo para verificar los resultados.

**Ejemplo:** El objetivo es ir de la casa al colegio.

Inicio

1. Salir de la casa
2. Si está lejos del colegio entonces tomé un medio de transporte que lo deje cerca del mismo.
3. Si no está lejos del colegio entonces dirigirse caminando hacia él mismo
4. Llegar a la puerta del colegio

Fin



---

## Partes de un algoritmo

Todo algoritmo debe constar de las siguientes partes:

- **Input o entrada:**

El ingreso de los datos que el algoritmo necesita para operar.

- **Proceso:**

Se trata de la operación lógica formal que el algoritmo emprenderá con lo recibido del input.

- **Output o salida:**

Los resultados obtenidos del proceso sobre el input, una vez terminada la ejecución del algoritmo.

## ¿Para qué sirve un algoritmo?

Dicho muy llanamente, un algoritmo sirve para resolver paso a paso un problema. Se trata de una serie de instrucciones ordenadas y secuenciadas para guiar un proceso determinado.

En las Ciencias de la computación, no obstante, los algoritmos constituyen el esqueleto de los procesos que luego se codifican y programan para que sean realizados por el computador.

## Tipos de algoritmos

Existen cuatro tipos de algoritmos en informática:

- **Algoritmos computacionales.**

Un algoritmo cuya resolución depende del cálculo, y que puede ser desarrollado por una calculadora o computadora sin dificultades.

- **Algoritmos no computacionales.**

Aquellos que no requieren de los procesos de un computador para resolverse, o cuyos pasos son exclusivos para la resolución por parte de un ser humano.

- **Algoritmos cualitativos.**

Se trata de un algoritmo en cuya resolución no intervienen cálculos numéricos, sino secuencias lógicas y/o formales.



---

- **Algoritmos cuantitativos.**

Todo lo contrario, es un algoritmo que depende de cálculos matemáticos para dar con su resolución.

**Los algoritmos presentan las siguientes características:**

- **Secuenciales.** Los algoritmos operan en secuencia, debe procesarse uno a la vez.
- **Precisos.** Los algoritmos han de ser precisos en su abordaje del tema, es decir, no pueden ser ambiguos o subjetivos.
- **Ordenados.** Los algoritmos se deben establecer en la secuencia precisa y exacta para que su lectura tenga sentido y se resuelva el problema.
- **Finitos.** Toda secuencia de algoritmos ha de tener un fin determinado, no puede prolongarse hasta el infinito.
- **Concretos.** Todo algoritmo debe ofrecer un resultado en base a las funciones que cumple.
- **Definidos.** Un mismo algoritmo ante los mismos elementos de entrada (input) debe dar siempre los mismos resultados.

## Ejemplos de algoritmos

Un par de ejemplos posibles de algoritmo son:

Algoritmo **no computacional** para elegir unos zapatos de fiesta.

1. INICIO
2. Entrar a la tienda y buscar la sección de zapatos de caballero.
3. Tomar un par de zapatos.
4. ¿Son zapatos de fiesta?

SI: (ir al paso 5) – NO: (volver al paso 3)

5. ¿Hay de la talla adecuada?

SI: (ir al paso 6) – NO: (volver al paso 3)

6. ¿El precio es pagable?

SI: (ir al paso 7) – NO: (volver al paso 3)

7. Comprar el par de zapatos elegido.
8. FIN

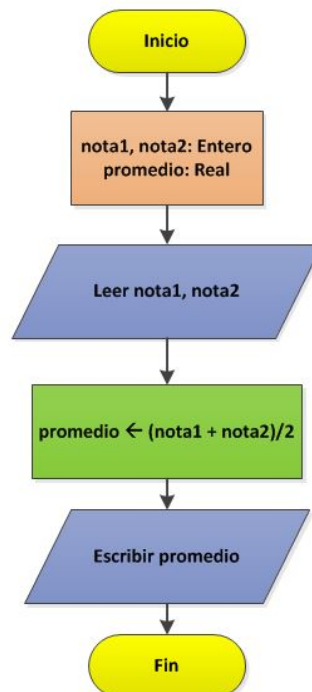
Algoritmo **computacional** para calcular el área de un triángulo rectángulo.

1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura ( $b \times h$ )
4. Dividir entre 2 el resultado  $(b \times h) / 2$
5. FIN

## Diagrama de flujos

Un diagrama de flujo es una representación esquemática de los distintos pasos de un programa. Constituyen otra forma de representar algoritmos distinta al pseudocódigo, pero que nos sirve de forma complementaria en el proceso de creación de la estructura del programa antes de ponernos delante del ordenador.

**Ejemplo:**



## Que representa cada símbolo:

SÍMBOLO	LO QUE REPRESENTA
Inicio Fin	Indica el inicio y el fin del algoritmo
	Entrada de datos
	Procesos
	Salida de datos
⇒ ↑↑	Flechas conectoras
	Decisiones
	Repeticiones o iteraciones

Las entradas son datos o insumos que necesita el algoritmo para que se pueda elaborar.

Los procesos son las acciones que permiten transformar las entradas (insumos o datos) en otros datos u otros insumos que permitirán dar solución al problema.

Las salidas hacen referencia a los resultados que debe dar al final el algoritmo.

Las decisiones se usan para tomar decisiones lógicas y de acuerdo a estas ejecutar o no conjuntos de instrucciones.

Las iteraciones permiten repetir un conjunto de instrucciones dentro de un algoritmo

Para elaborar un diagrama de flujo se deben tener en cuenta las siguientes reglas:

- Los diagramas se deben realizar de arriba hacia abajo y de izquierda a derecha.
- El algoritmo debe arrancar con el símbolo de inicio y terminar con símbolo de fin.
- La dirección de flujo se debe representar por medio de flechas.

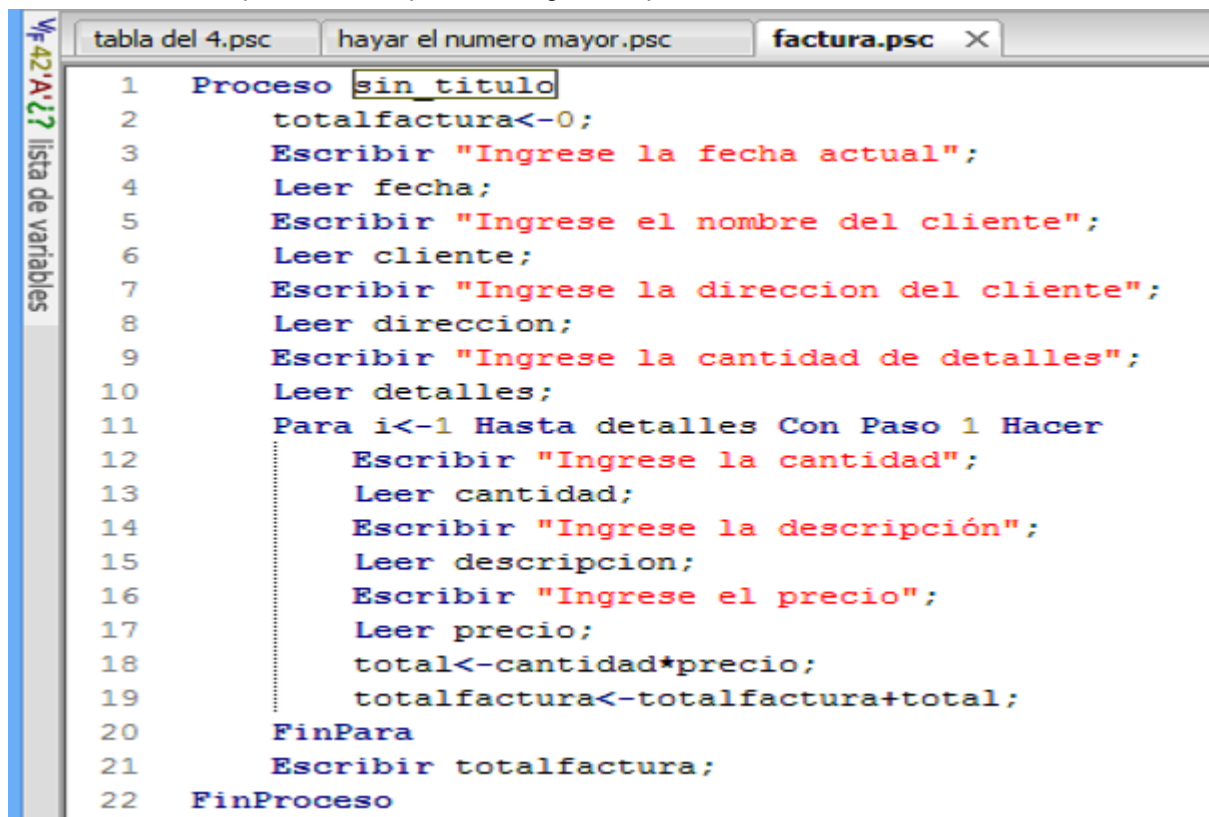
- Todas las líneas de flujo deben llegar a un símbolo o a otra línea.

## Pseudocódigo

Es la técnica que permite expresar la solución de un problema mediante un algoritmo escrito en palabras normales de un idioma (por ejemplo español), utilizando palabras imperativas. Es normal encontrar palabras como :inicie ,lea ,sume ,divida,calcule ,finalice. No hay un léxico obligado para el pseudocódigo , pero con el uso frecuente se han establecido algunos estándares.

## Características

Las características que definen el pseudocódigo se exponen a continuación:



```
1  Proceso sin titulo
2      totalfactura<-0;
3      Escribir "Ingrese la fecha actual";
4      Leer fecha;
5      Escribir "Ingrese el nombre del cliente";
6      Leer cliente;
7      Escribir "Ingrese la direccion del cliente";
8      Leer direccion;
9      Escribir "Ingrese la cantidad de detalles";
10     Leer detalles;
11     Para i<-1 Hasta detalles Con Paso 1 Hacer
12         Escribir "Ingrese la cantidad";
13         Leer cantidad;
14         Escribir "Ingrese la descripción";
15         Leer descripcion;
16         Escribir "Ingrese el precio";
17         Leer precio;
18         total<-cantidad*precio;
19         totalfactura<-totalfactura+total;
20     FinPara
21     Escribir totalfactura;
22 FinProceso
```

## No sigue un formato específico

Debido a que el pseudocódigo está orientado a la comprensión humana y no es interpretado por el ordenador de forma directa, éste puede escribirse en cualquier tipo de formato que pueda ser entendido por otras personas. No obstante, existen convenciones académicas que sugieren seguir cierta metodología de escritura, pero al no cumplir con fines informáticos de manera

directa, no se consideran formatos propiamente.

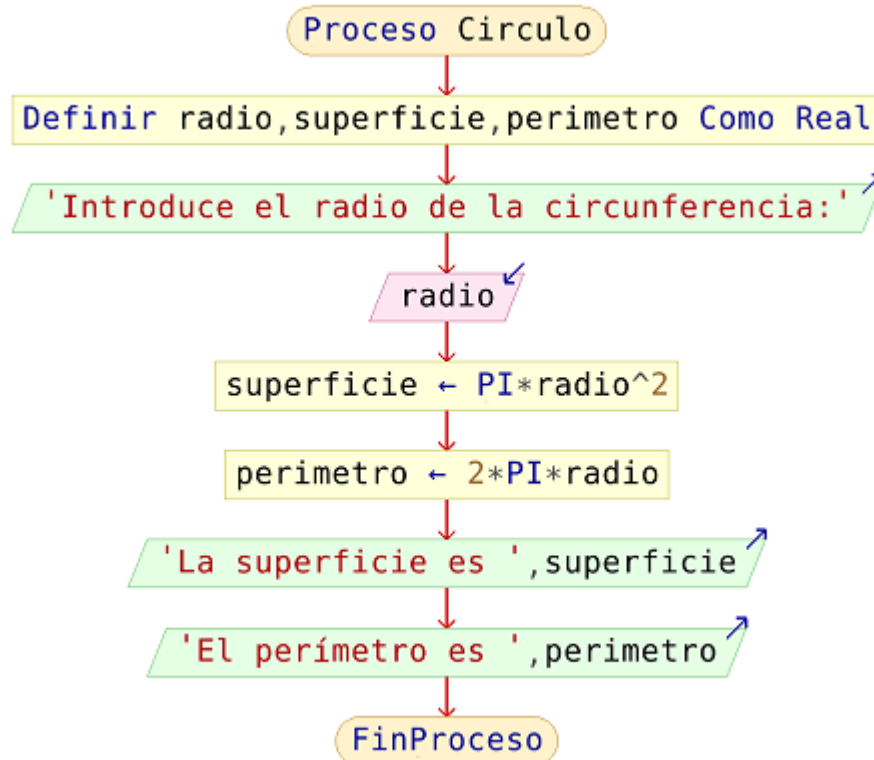
## Recurso para diseño de algoritmos

El pseudocódigo ofrece la posibilidad de escribir algoritmos lógicos que indiquen una serie de instrucciones secuenciales que llevan a la resolución de una tarea. Para que los fines de un algoritmo sean efectivos, el pseudocódigo permite escribir dicho algoritmo por medio de un esquema.

## Preámbulo a programación

Aunque el pseudocódigo no sea un lenguaje de programación, ofrece una herramienta útil para los programadores. Así, antes de desarrollar un software, es posible escribir un pseudocódigo que represente el esquema preliminar del software que se desea crear. Esto facilita de manera sustancial la posterior programación en cualquier lenguaje de programación, ya que por medio del algoritmo, es posible encontrar posibles errores y elementos que pueden mejorarse antes de finalizar el

## Estructura





---

## Declaraciones

Las declaraciones son las distintas instrucciones que deben ser leídas por el ordenador para llevar a cabo la resolución de una tarea. En otras palabras, las declaraciones son directrices que indican los pasos a seguir para resolver un problema. En el pseudocódigo, la forma en la que se escriben y ejecutan las directrices debe seguir normas de flujo concreta. Así, las direcciones se escriben de arriba hacia abajo. Igualmente, las distintas operaciones matemáticas que se pueden resolver por medio del pseudocódigo, deben estar involucradas de forma integral a las declaraciones.

## Keywords o palabras claves

Las keywords o palabras claves son aquellas palabras con un significado semántico que un ordenador puede reconocer. Así, las palabras clave pueden representar parámetros o comandos de significado concreto.

Todos los lenguajes de programación tienen palabras claves. En el caso del pseudocódigo, estas permiten ingresar declaraciones de entrada o salida de procesos. Así, algunas palabras claves comunes pueden ser Sumar, Multiplicar, Restar, Incrementar, Imprimir, Establecer, Ingresar, Mostrar...

## Condicionales

Durante el desarrollo de pseudocódigo, muchas veces es necesario evaluar el resultado de una operación concreta con el fin de tomar un camino a seguir en función de dicho resultado. Para estos casos, existen expresiones instruccionales que permiten hacer esta evaluación. A continuación se exponen algunas:

**En caso:** Esta instruccional se emplea cuando se desea comparar una única variable con varias condiciones. En estos casos, suele emplearse cuando las condiciones son caracteres o números.

**Si no – Si:** Es utilizada para ejecutar declaraciones concretas en función de una condición previamente determinada. Puede aplicarse también cuando existen más de una condición y diversas variables. Así, por ejemplo, un “Si” con una sección “Si no”, hace posible resolver una serie de tareas en caso de no cumplirse la condición “Si”.

## Iteraciones

Las iteraciones son las instrucciones que permiten crear un ciclo de instrucciones idénticas hasta obtener un objetivo concreto por medio de los resultados obtenidos en cada ciclo. Algunos comandos de iteraciones son los siguientes:

**Mientras:** Es utilizado para repetir un ciclo de instrucciones “bloque de código” de forma continua siempre y cuando una condición previamente definida siga cumpliendo con una condicional.

**Para:** Se emplea para asignar valores y ejecutar las tareas de iteración para cada uno.

## Funciones

Es la forma de referirse a la serie de bloques de tareas que pueden desglosarse del algoritmo



principal. Las funciones suelen tener propósitos particulares que buscan ejecutar Declaraciones. Igualmente, las funciones permiten reutilizar un código para ejecutar instrucciones un número indeterminado de veces sin necesidad de extender el diagrama principal.

## Ventajas y desventajas

El pseudocódigo trae consigo grandes ventajas para los programadores y la informática en general, sin embargo, padece de ciertas desventajas que es necesario considerar. A continuación se describen cuáles son:

### Pseudocódigo:

```
INICIO
  Num1, Num2, Suma : ENTERO
  ESCRIBA "Diga dos números: "
  LEA Num1, Num2
  Suma ← Num1 + Num2
  ESCRIBA "La Suma es:", Suma
FIN
```

### Diagrama de flujo:



## Ventajas

El pseudocódigo es fácil de entender, por lo que no es necesario ser un experto en programación para leer y entender cómo funciona un algoritmo escrito en pseudocódigo.

Hace más fácil desarrollar instrucciones para resolver problemas. Debido a que su escritura resulta sencilla y amigable, el programador puede enfocarse en el método por el cual un programa llevará a cabo una tarea.

Ayuda a optimizar el tiempo de desarrollo, ya que un algoritmo en pseudocódigo funciona como un esquema lógico preliminar que al momento de desarrollar en un lenguaje de programación, simplifica y guía el proceso.

Al no seguir una estructura, el pseudocódigo puede compartirse y ser comprendido por otros programadores, de hecho, puede escribirse en cualquier idioma o traducirse, lo que representa una gran virtud.

## Desventajas

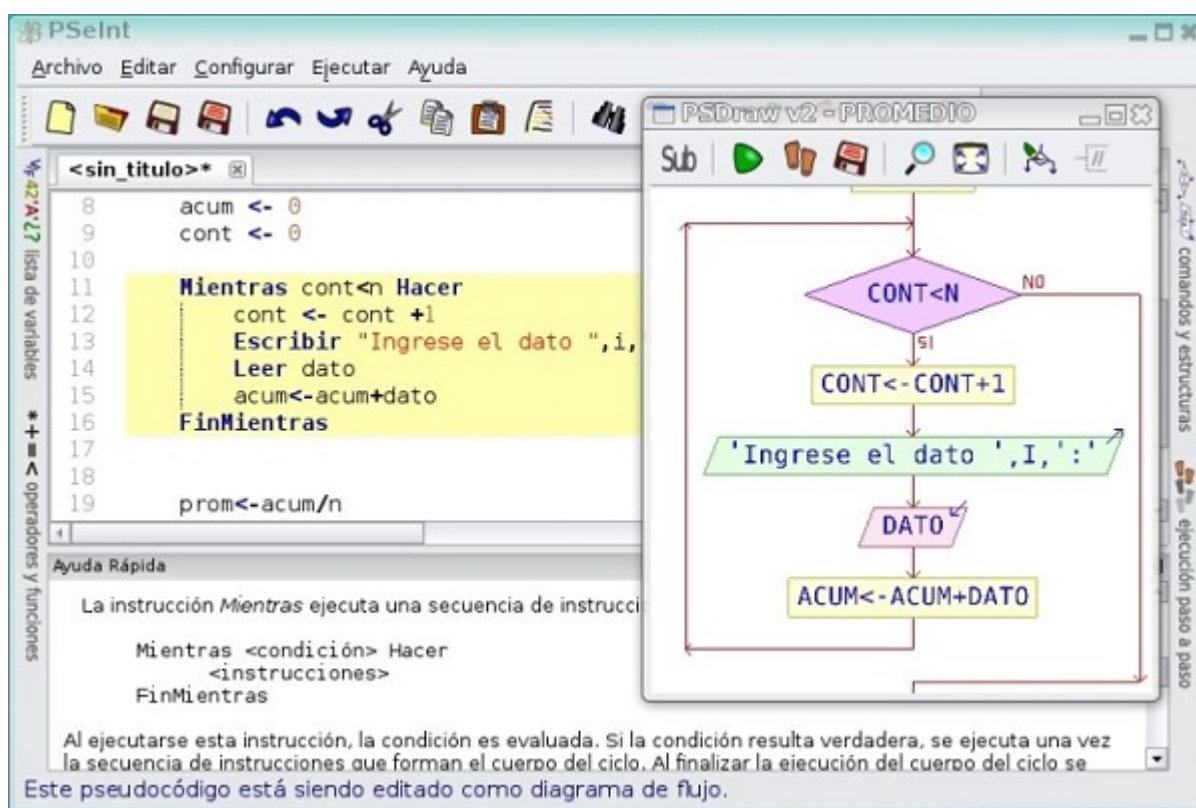
Al no ser un lenguaje de programación, no existen reglas universales para escribir pseudocódigo. Esto hace que puedan existir confusiones de interpretación entre los programadores.

Debido a que un ordenador no puede leer pseudocódigo de forma directa, no es posible representar de forma visual el funcionamiento del software.

Debido a la falta de estándares, la forma en la que se escribe el pseudocódigo puede ser subjetiva.

## Ejemplos

Los ejemplos de pseudocódigo se representan de forma práctica. A continuación exponemos algunos ejemplos:



## Conseguir la media aritmética de tres elementos

En este algoritmo, se escribirá un diagrama en pseudocódigo que permitirá obtener de manera directa la media aritmética de tres elementos. Para ello, se solicitará al usuario que ingrese tres valores numéricos, posteriormente, se indica la ecuación que se utilizará para llevar a cabo el cálculo y finalmente, se emplea una Declaración de Mostrar para que se enseñe el valor resultante de la operación. Una forma de desarrollar este algoritmo sería el siguiente:



---

**Inicio**

**Mostrar** "Ingresar valor 1": Pedir A

**Mostrar** "Ingresar valor 2": Pedir B

**Mostrar** "Ingresar valor 3": Pedir C

$$M = (A + B + C) / 3$$

**Mostrar** "La media aritmética de los tres valores ingresados es", M

**Fin**

## Determinar volumen de un cilindro

En este ejemplo, se desea desarrollar un algoritmo en pseudocódigo que permita obtener el volumen de un cilindro. Para ello, se solicita al usuario que ingrese valores conocidos como la altura y el diámetro del cilindro. Posteriormente, es necesario indicar al algoritmo cuál es la ecuación a utilizar y se indicarán los valores ingresados que corresponden a cada variables. A continuación se muestra una manera de desarrollar este diagrama de flujo:

**Inicio**

**Mostrar** "Ingresar altura del cilindro en metros": Pedir H

**Mostrar** "Ingresar diámetro del cilindro en metros": Pedir d

$$R = d / 2; \text{ Pi} = 3,14$$

$$V = \text{Pi} * (R^2) * H$$

**Mostrar** "El cilindro tiene un volumen de", V, "metros ^3"



---

**Fin**

## **Obtener el precio de un producto con descuento**

En este ejemplo, se utilizará la escritura en pseudocódigo para desarrollar un algoritmo que permite ingresar el precio de un producto y en consecuencia, permita determinar su valor real con un descuento determinado de manera automática. Para resolver este ejemplo, debe solicitarse al usuario que ingrese el precio (valor en número) del producto al que desea conocer su precio con descuento. A continuación, se indica en el diagrama cuál es la fórmula a seguir para obtener el nuevo precio y finalmente, se determina una Declaración que le indica al algoritmo que arroje el valor resultante del cálculo.

**Inicio**

**Mostrar** "Ingresar el precio del producto en \$": Pedir P

**Mostrar** "Ingresar la oferta de descuento para el producto en %": Pedir d

$m = d * 0,1$

$N = P * m$

**Mostrar** "El precio del producto con el descuento indicado es de", N, "\$"

**Fin**