

	Unitécnica Manizales Bootcamp Programación		
Programa	Bootcamp Programación	Grupo	Semana
Tema	Algoritmos		
Instructor	Julián Giraldo R.	Fecha	2025-08-04

Algoritmos

Toda actividad humana, conciente o inconcientemente, sigue una secuencia lógica, ordenada por nuestra estructura mental, que nos define, cual debe ser la manera de realizar cualquier actividad.

Ya sea elaborar un delicioso platillo, vestarnos en la mañana, dirigirnos a nuestro sitio de trabajo o realizar una rigurosa investigación científica, seguimos secuencias lógicas y ordenadas bien definidas.

Se puede decir que esas secuencias corresponden a los algoritmos con los cuales desarrollamos cualquier actividad. El concepto de algoritmo nace por el matemático árabe del siglo IX, Mohammed al-Khowârizmî, que definió un conjunto de reglas básicas para la realización de cálculos con números y ecuaciones.

Definición

Un algoritmo se puede definir como el conjunto de pasos ordenados, lógicos y finitos necesarios para llegar a la solución de un problema.

Un algoritmo debe tener las siguientes características para considerarse como tal y poder ser la herramienta para resolver un problema:

- Debe ser preciso: debe indicar como se realizan los diferentes pasos, en un orden exacto.
- Debe ser definido: si se sigue varias veces el mismo algoritmo, con los mismos datos, el resultado debe ser siempre el mismo.
- Debe ser finito: el número de pasos debe ser limitado. Debe tener un primer paso y siempre un último paso.
- Debe ser secuencial: debe tener un orden lógico en sus diferentes pasos para saber como se va del primer paso al último.
- Debe ser completo: debe solucionar el problema planteado.

Los algoritmos se pueden usar para resolver cualquier tipo de problema, no solo aquellos que implican una solución matemática, por ejemplo, el algoritmo para preparar un delicioso sánduche de jamón y queso podría ser:

1. tomar un pan tajado.
2. untarlo con mayonesa
3. poner sobre el pan una loncha de jamón

4. esparcir en esta mostaza
5. agregar rodajas de tomate verde.
6. agregar encima una hoja de lechuga fresca.
7. agregar una loncha de queso.
8. untar otra tajada de pan con mayonesa
9. poner el pan encima de la loncha de queso

En general este algoritmo cumple con las características, pues tiene un número definido de pasos (9), tiene un primer paso y un último paso, la secuencialidad permite llevarnos del primero al último paso sin equivocarnos, cumple su objetivo y es completo, ya que cada vez que lo sigamos, obtendremos como resultado un delicioso sánduche.

Otros ejemplos de algoritmos pueden ser:

- Hacer un pedido en una fábrica:
 1. Llenar y enviar el pedido
 2. Leer el pedido.
 3. Examinar el historial del cliente.
 4. Si el cliente es solvente, aceptar pedido; en caso contrario, rechazar pedido.
 5. Despachar los elementos del pedido
 6. Generar y enviar la factura de cobro
- Establecer si un número es primo o no:
 1. Establecer N como el número a averiguar
 2. Poner X igual a 2 ($X = 2$, X variable que representa a los divisores del número que se busca N).
 3. Dividir N entre X (N/X).
 4. Si el resultado de N/X es entero, entonces N no es un número primo y saltar hasta el punto 8; en caso contrario, continuar el proceso.
 5. Suma 1 a X ($X \leftarrow X + 1$).
 6. Si X es igual a N, entonces N es un número primo; en caso contrario, retroceder al paso 3.
 7. Terminar

Escritura de algoritmos

El sistema para describir (“escribir”) un algoritmo consiste en realizar una descripción paso a paso con un lenguaje natural del citado algoritmo. Se debe recordar que un algoritmo es un método o conjunto de reglas para solucionar un problema. En cálculos elementales estas reglas tienen las siguientes propiedades:

- deben estar seguidas de alguna secuencia definida de pasos hasta que se obtenga un resultado coherente,
- sólo puede ejecutarse una operación a la vez.

Un algoritmo normalmente se desarrolla de manera secuencial; consideremos el algoritmo que responde a la pregunta ¿Qué hacer para ir al cine?

La respuesta es muy sencilla y puede ser descrita en forma de algoritmo general de modo similar a:

- Consultar la cartelera
- Reservar el tiquete
- Trasladarse al cine
- comprar la entrada
- Ingresar a la sala
- ver la película
- Salir de la sala

En este algoritmo se presentan siete acciones básicas, que se deben ejecutar en un orden específico. En términos del computador, cada acción se codificará en una o varias sentencias que ejecutan una tarea particular.

El algoritmo descrito es muy sencillo; sin embargo, como ya se ha indicado en párrafos anteriores, el algoritmo general se descompondrá en pasos más simples en un procedimiento denominado refinamiento sucesivo, ya que cada acción puede descomponerse a su vez en otras acciones simples. Así, por ejemplo, un primer refinamiento del algoritmo ir al cine se puede describir de la forma siguiente:

1. inicio
2. Consultar la cartelera de cines en el periódico, por teléfono o por Internet.
3. si no proyectan una película de mi agrado, entonces
 - 3.1. decidir otra actividad
 - 3.2. saltar al paso 7sino
 - 3.3. Si la consulta se hace por teléfono o por Internet, reservar el tiquete.fin_si
4. ir al cine
5. si hay cola entonces
 - 5.1. ponerse en ella
 - 5.2. mientras haya personas delante haga
 - 5.2.1. avanzar en la colafin_mientrasfin_si
6. si no se hizo reservación y hay localidades entonces
 - 6.1. comprar una entradasino
 - 6.1.1. si se hizo reservación entonces
 - 6.1.1.1. comprar la entradafin_si
- 6.2. pasar a la sala
- 6.3. localizar la(s) butaca(s)
- 6.4. mientras proyectan la película haga

```

        6.4.1.      ver la película
        fin_mientras
    6.5.      abandonar el cine
    si_no
    6.6.      refunfuñar y ser mas previsivo la próxima vez
    fin_si

7. volver a casa
8. fin

```

En el algoritmo anterior existen diferentes aspectos a considerar. En primer lugar, ciertas palabras reservadas se han escrito deliberadamente en **negrita** (mientras, sino; etc.). Estas palabras describen las estructuras de control fundamentales y procesos de toma de decisión en el algoritmo. Estas incluyen los conceptos importantes de selección (expresadas por si-entonces-si_no if-then-else) y de repetición (expresadas con mientras-haga o a veces repetir-hasta y mientras - haga, en inglés, repeat-until y while-do) que se encuentran en casi todos los algoritmos, especialmente los de proceso de datos. La capacidad de decisión permite seleccionar alternativas de acciones a seguir o bien la repetición una y otra vez de operaciones básicas.

Representación gráfica de los algoritmos

Para representar un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo del lenguaje de programación elegido. Ello permitirá que un algoritmo pueda ser codificado indistintamente en cualquier lenguaje. Para conseguir este objetivo se precisa que el algoritmo sea representado gráfica o numéricamente, de modo que las sucesivas acciones no dependan de la sintaxis de ningún lenguaje de programación, sino que la descripción pueda servir fácilmente para su transformación en un programa, es decir, su codificación.

Los métodos usuales para representar un algoritmo son:

1. *Diagrama de flujo.*
2. *Diagrama N-S (Nassi-Schneiderman).*
3. *Lenguaje de especificación de algoritmos: pseudocódigo.*
4. *Lenguaje español, inglés...*
5. *Fórmulas*

Los métodos 4 y 5 no suelen ser fáciles de transformar en programas. Una descripción en español narrativo no es satisfactoria, ya que es demasiado prolija y generalmente ambigua. Una fórmula, sin embargo, es buen sistema de representación. Por ejemplo, las fórmulas para la solución de una ecuación cuadrática (de segundo grado) es un medio sucinto de expresar el procedimiento algorítmico que se debe ejecutar para obtener las raíces de dicha ecuación.

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

\ significa lo siguiente:

- 4ac) I 2a

1. *Eleve al cuadrado b.*
2. *Toma a; multiplicar por c; multiplicar por 4.*
3. *Restar el resultado obtenido de 2 del resultado de 1, etc.*

Sin embargo, no es frecuente que un algoritmo pueda ser expresado por medio de una simple fórmula.

Diagramas de flujo

Un diagrama de flujo (flowchart) es una de las técnicas de representación de algoritmos más antigua y a la vez más utilizada, aunque su empleo ha disminuido considerablemente, sobre todo desde la aparición de lenguajes de programación estructurados. Un diagrama de flujo es un diagrama que utiliza los símbolos (cajas) estándar y que tiene los pasos de algoritmo escritos en esas cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe ejecutar.

Los símbolos estándar normalizados por ANSI (abreviatura de American National Standard Institute) son muy variados. los símbolos más utilizados representan:

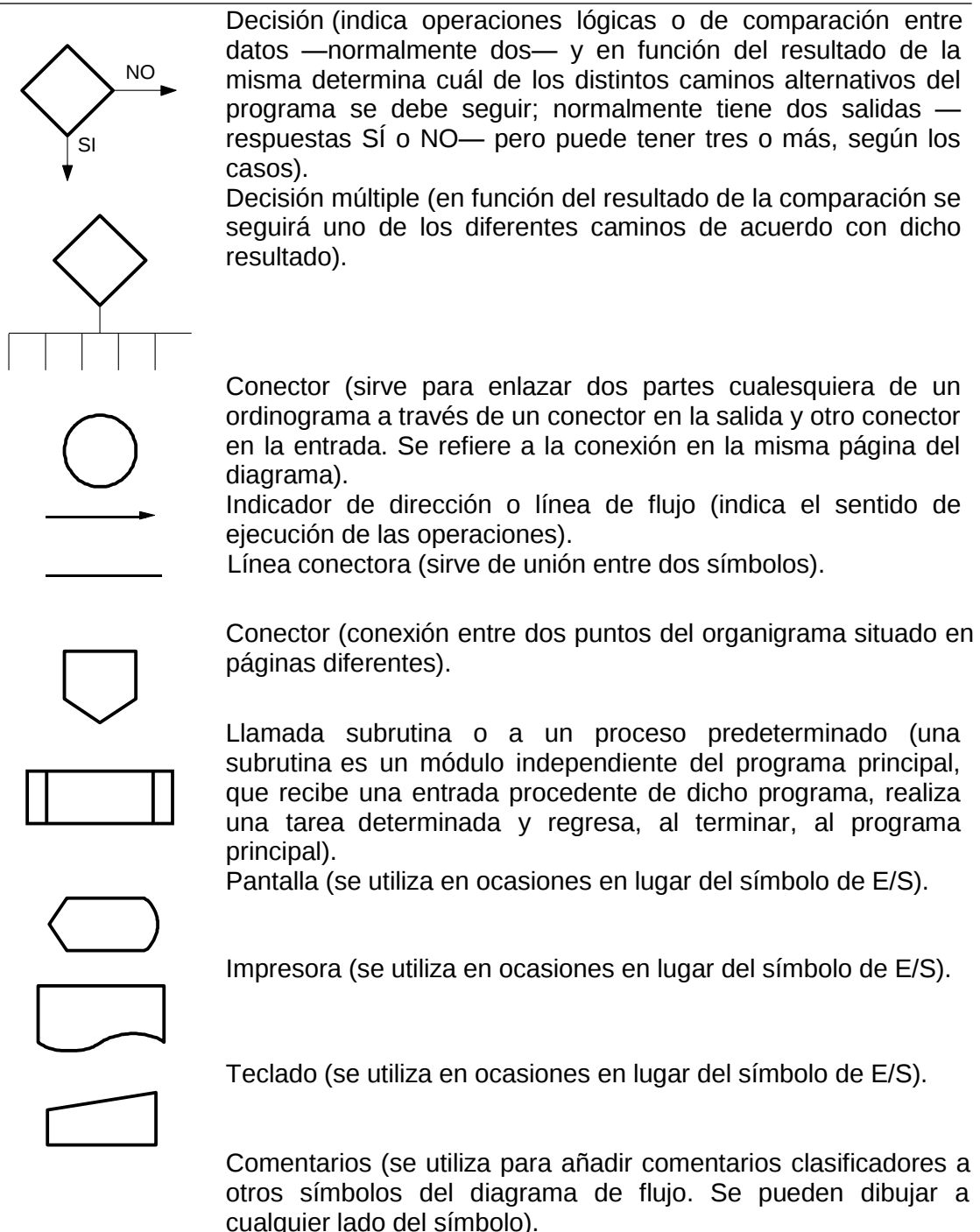
Símbolos de diagramas de flujo

Cada símbolo visto anteriormente indica el tipo de operación a ejecutar y el diagrama de flujo ilustra gráficamente la secuencia en la que se ejecutan las operaciones.

Las líneas de flujo ($\rightarrow$) representan el flujo secuencial de la lógica del programa. Un rectángulo () significa algún tipo de proceso en el computador, es decir, acciones a realizar (sumar dos números, calcular la raíz cuadrada de un número, etc.).

El paralelogramo () es un símbolo de entrada/salida que representa cualquier tipo de entrada.

Símbolos principales	Función
	Terminal (representa el comienzo, «inicio», y el final, «fin» de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa.
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos, «entrada», o registro de la información procesada en un periférico, «salida».
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.).



La siguiente figura representa es un diagrama de flujo básico. Este diagrama muestra la resolución de un programa que deduce el salario neto de un trabajador a partir de la

lectura del nombre, horas trabajadas, precio de la hora y sabiendo que los impuestos aplicados son el 25 por 100 sobre el salario bruto.

Ejemplos

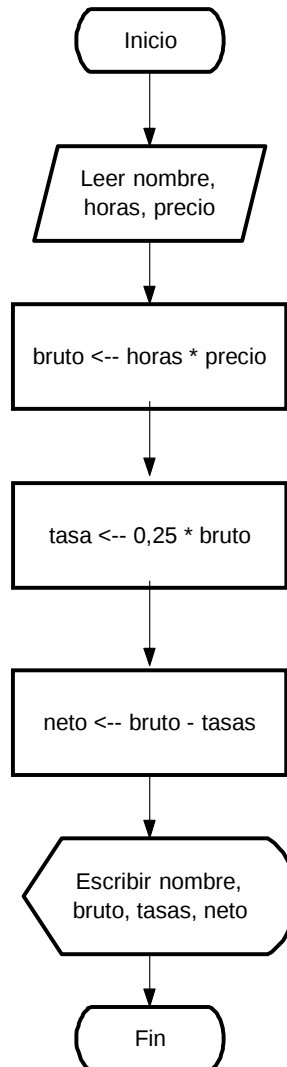


Figura 1.1 Diagrama de flujo

Diagramas de Nassi-Schneiderman (N-S)

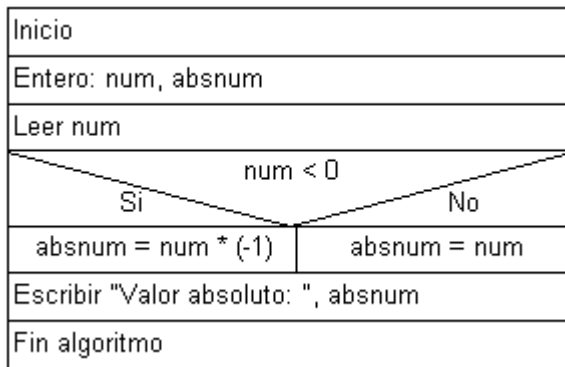
El diagrama N-S de Nassi-Schneiderman ☐ también conocido como de Chapin ☐ es como un diagrama de flujo en el que se omiten las flechas de unión y las cajas son contiguas. Las acciones sucesivas se escriben en cajas sucesivas y, como en los diagramas de flujo, se pueden escribir diferentes acciones en una caja.

Un algoritmo se representa con un rectángulo en el que cada banda es una acción a realizar:

Leer Nombre, horas, precio
Calcular $\text{Salario} \square \text{horas} * \text{precio}$
Calcular $\text{Impuestos} \square 0.25 * \text{salario}$
Calcular $\text{Neto} \square \text{salario} - \text{impuestos}$
Escribir Nombre, salario, impuestos, neto

Nombre del algoritmo
<acción 1>
<acción 2>
<acción 3>
...
Fin

Otra forma de implementar algoritmos como diagrama de Nassi-Schneiderman se presenta a continuación, en el cual se muestra el procedimiento para hallar el valor absoluto:



Pre y

poscondiciones

Ejemplos

Pseudocódigo

El pseudocódigo es un lenguaje de especificación (descripción) de algoritmos. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. Los lenguajes APL Pascal y Ada se utilizan a veces como lenguajes de especificación de algoritmos.

El pseudocódigo nació como un lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada. Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora. La ventaja del pseudocódigo es que en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica, una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes estructurados como Pascal, C, FORTRAN 77/90, C++, Java, C#, etc.

El pseudocódigo original utiliza para representar las acciones sucesivas palabras reservadas en inglés —similares a sus homónimas en los lenguajes de programación—, tales como start, end, stop, if-then-else, while-end, repeat-until, etc. La escritura de pseudocódigo exige normalmente la indentación (sangría en el margen izquierdo) de diferentes líneas.

La representación en pseudocódigo del diagrama de flujo de la Figura 1.1 es la siguiente:

```

Start
    //cálculo de impuesto y salarios read
    nombre, horas, precio_hora salario_bruto □
    horas * precio_hora tasas □ 0,25 *
    salario_bruto salario_neto □ salario_bruto –
    tasas
    write nombre, salario_bruto, tasas, salario_neto
end

```

Otro ejemplo aclaratorio en el uso de pseudocódigo, podría ser un sencillo algoritmo del arranque matinal de un automóvil.

```

Inicio
    //arranque matinal de un automóvil Introducir
    la llave en el switch Sacar el Choke
    Dar Start
    Acelerar
    Escuchar el ruido del motor
    Acelerar nuevamente
    Esperar unos momentos para que se caliente el motor Quitar el
    choke

```

Fin

El pseudocódigo también se puede usar con palabras en español como:

Inicio	leer	escribir
Fin	si entonces sino	mientras
Haga	repita	para
Retornar	hasta	seleccione