¿Qué es un algoritmo?

POR [GABY NIEVA](https://dcodingames.com/author/dcg_wp_admin/) · PUBLICADA 30 OCTUBRE, 2016 · ACTUALIZADO 18 NOVIEMBRE, 2018

Un algoritmo es  como un “plan de ataque” para resolver un problema. En éste artículo te presento información sobre los orígenes y notaciones de los algoritmos.



Origen de la palabra algoritmo.

La palabra algoritmo proviene del nombre del matemático Abu Jafar Muhamed ibn Musä al Khwàrizm quien nació en 780 A.C. en Bagdad y murió en 850 A.C. Fue tal el trabajo de este matemático, que se bautizó al método estructurado para encontrar soluciones en honor a su nombre.



*Figura 1. Un sello postal de la URSS  en honor a Abu Jafar Muhammed ibn Musä al Kwàrizm [*[*Wikipedia*](https://es.wikipedia.org/wiki/Al-Juarismi)*]*

La humanidad ha venido utilizando y diseñando algoritmos muchísimo antes de que la computadora existiera. Sin embargo, no fue hasta 1928 cuando David Hilbert y Alan Turing comenzaron a utilizarlos para describir un programa.

Definición.

Una definición más formal de lo que es un algoritmo, es la siguiente:

*“Un algoritmo es una secuencia de pasos a seguir, no ambiguos, finitos y determinísticos, que llevan a la solución de un problema”*

Así de simple. Pero esto tan simple es sumamente poderoso porque es la columna vertebral de la computación moderna, por así decirlo.

Características.

Las características del algoritmo, como lo mencioné arriba, definen al algoritmo en sí, por lo que es importante entenderlas para así entender la definición de algoritmo.

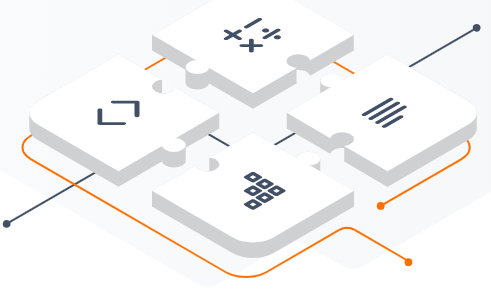
Primero que nada, debe quedar claro que un algoritmo es una secuencia de pasos. Eso implica que es una lista de actividades que alguien debe llevar a cabo para resolver un problema. Pero esta secuencia de pasos, esta lista de actividades tiene características significativas que son:

* La secuencia es finita: tan sencillo de entender como el hecho de que esta secuencia tiene inicio y fin. No importa que tan larga sea la lista de pasos, inicia en un momento, y debe terminar.
* Los pasos no son ambiguos: La definición de [ambigüedad](https://www.google.com.mx/search?q=ambig%C3%BCedad&ie=utf-8&oe=utf-8&client=firefox-b&gfe_rd=cr&ei=C_wTWJnaIq7I8gf0gaCYCg)es *“Comportamiento, hecho, palabra o expresión que puede entenderse o interpretarse de diversas maneras.”*. Por lo tanto, si en un algoritmo los pasos son no ambiguos, quiere decir que los pasos deben entenderse sin dar lugar a diversas interpretaciones. En pocas palabras, la secuencia de pasos debe ser clara, específica, entendible.  Creo que esta es la cualidad que más nos cuesta cumplir cuando comenzamos a diseñar algoritmos, sobre todo porque la forma natural en la que nos comunicamos, tiende a ser ambigua y dependemos mucho del contexto para que lo que decimos no sea mal interpretado.
* Los pasos son determinísticos: Un [*modelo determinista*](https://es.wikipedia.org/wiki/Determinismo_cient%C3%ADfico) es “*un modelo matemático donde las mismas entradas o condiciones iniciales producirán invariablemente las mismas salidas o resultados, no contemplándose la existencia de azar, o incertidumbre en el proceso modelada mediante dicho modelo”*. Quizá esta característica te resulte difícil de entender, (y más después de la definición técnica que acabo de involucrar) pero realmente es más simple: si corriste un programa con ciertas entradas, lo lógico es esperar que la siguiente ejecución del mismo programa con las mismas entradas generen el mismo resultado. En este sentido entonces creo que es válido afirmar que una característica de los algoritmos es su  estabilidad, por lo que  representan una solución universal para un problema, aunque no la única solución al problema.

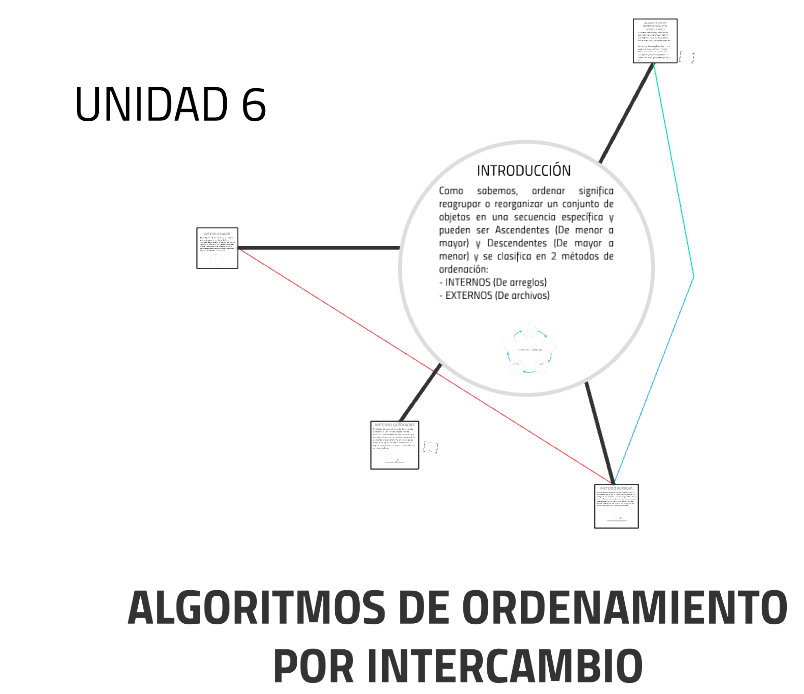
Creo que ahora podemos entender que entonces este “plan de ataque” que diseñamos para solucionar un problema debe:

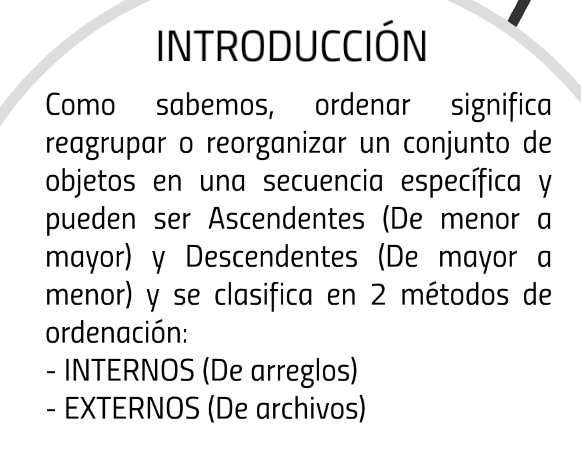
* Tener inicio y fin; el fin debe estar claramente identificado, es decir, sabremos a donde llegar, por lo tanto, sabremos cuando hayamos llegado.
* Tiene actividades claras y concisas, escritas de forma tal que no puedan dar lugar a malas interpretaciones.
* Y es estable, si lo probamos una y otra vez con un conjunto de entradas debería generar el mismo conjunto de salidas.

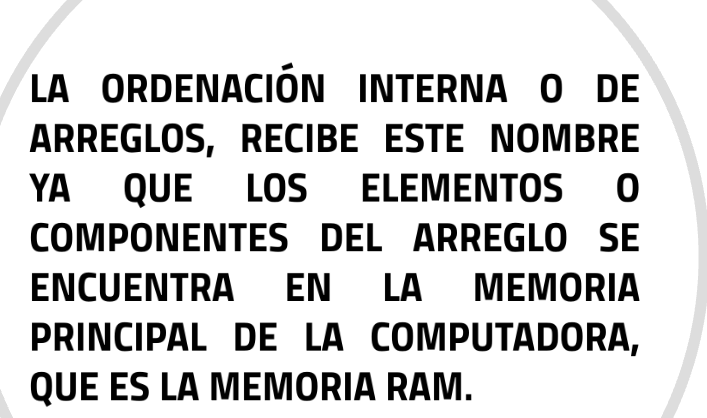
|  |  |  |
| --- | --- | --- |
| **Perfil**  ¿Quiere colocar su propia imagen en el círculo? Es fácil. Seleccione la imagen y haga clic derecho. Seleccione "Rellenar" en el menú contextual. Seleccione Imagen... de la lista. Vaya a su equipo para obtener la imagen adecuada. Haga clic en Aceptar para insertar la imagen seleccionada.  Después de insertar la imagen, selecciónela de nuevo. Vaya al menú Formato de herramientas de imagen. Haga clic en la flecha hacia abajo "Recortar" y seleccione "Rellenar" de la lista. Esto ajustará automáticamente la imagen para recortarla. Puede hacer clic y arrastrar la imagen para colocarla correctamente.  **Contacto**  TELÉFONO:  678-555-0103  SITIO WEB:  El sitio web va aquí  CORREO ELECTRÓNICO:  alguien@ejemplo.com  **Aficiones**  Afición n.º 1  Afición n.º 2  Afición n.º 3  Afición n.º 4 |  | **EDUCACIÓN**      **https://www.youtube.com/watch?v=DqXFtylWI10**  🔥Ejercita tu lógica de programación - Desafíos de razonamiento y código para mejorar tus algoritmos  **https://www.youtube.com/watch?v=wv1VFXgZbV0**  🔺0.1 Ejercicios resueltos de razonamiento lógico - Curso de programación desde cero (con Python)    **[Nombre de la compañía] [Puesto del trabajo]**  [Fecha de inicio] - [Fecha de finalización]  [Describa sus responsabilidades y logros recalcando el impacto y los resultados conseguidos. Puede poner ejemplos, pero sea breve.]  **[Nombre de la compañía] [Puesto del trabajo]**  [Fecha de inicio] - [Fecha de finalización]  [Describa sus responsabilidades y logros recalcando el impacto y los resultados conseguidos. Puede poner ejemplos, pero sea breve.]  APTITUDES |

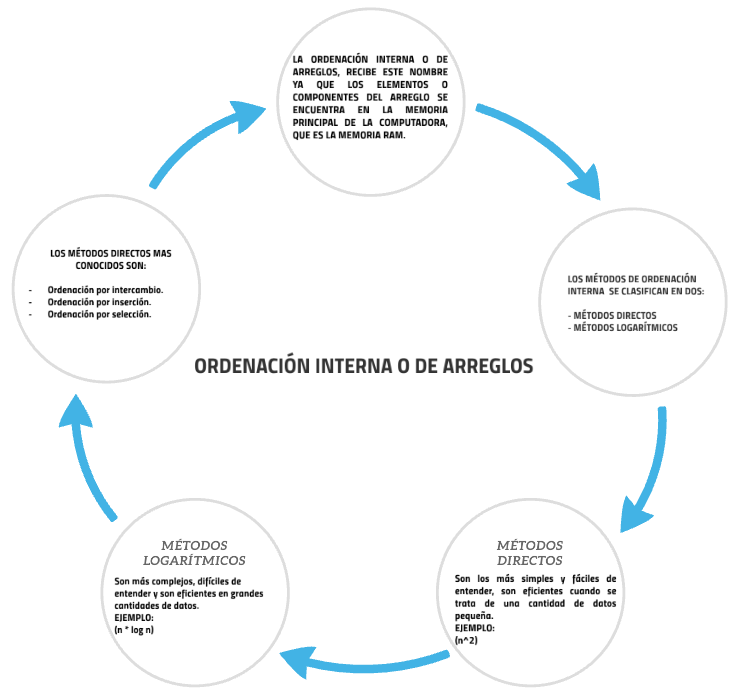












Notaciones

Una [notación](https://www.google.com.mx/search?q=notaci%C3%B3n&ie=utf-8&oe=utf-8&client=firefox-b&gfe_rd=cr&ei=7wIUWLXNN6zI8gekpKTQAg)es un conjunto de signos que se utilizan para representar ciertos conceptos, como en las matemáticas o en la música.

Para escribir un algoritmo existen diversas notaciones, y usar la apropiada definirá el uso que le daremos al algoritmo.  Las notaciones (o formas de escribir) un algoritmo son las siguientes:

Descripción narrada.

Un algoritmo se puede diseñar usando una descripción narrada, ya sea oral o escrita. Ésta no es la mejor notación para resolver un problema usando una computadora.

Los seres humanos hemos creado algoritmos en la notación de descripción narrada desde hace siglos. El ejemplo más clásico de un algoritmo descrito con esta notación es una receta de cocina, desde las recetas que intercambian de forma oral e informal dos comadres que se encuentran en el mercado hasta las más elaboradas que se pueden encontrar en los libros especializados.



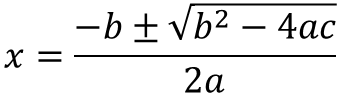
*Figura 2. Características de la técnica “Descripción narrada”*

En la figura 2 te muestro las características más relevantes de este tipo de notación. Destaco las siguientes:

* Se utiliza el lenguaje natural: lo cual es bueno para nosotros los humanos, porque es la forma en la que nos comunicamos, lo que hace de esta notación bastante fácil de transmitir y entender, además de que no necesita de muchas reglas para crear un algoritmo.
* Su principal ventaja es su principal desventaja: el lenguaje natural da pie a ambigüedades y mal interpretaciones, y depende del contexto para ser entendido claramente. Regresemos al ejemplo de la receta de cocina, una simple indicación como “corta el queso en rebanadas” puede generar preguntas como: ¿qué tan gruesas deben ser las rebanadas?, preguntas que para contestarlas quizá dependamos del contexto o de la experiencia del que ejecuta la receta.

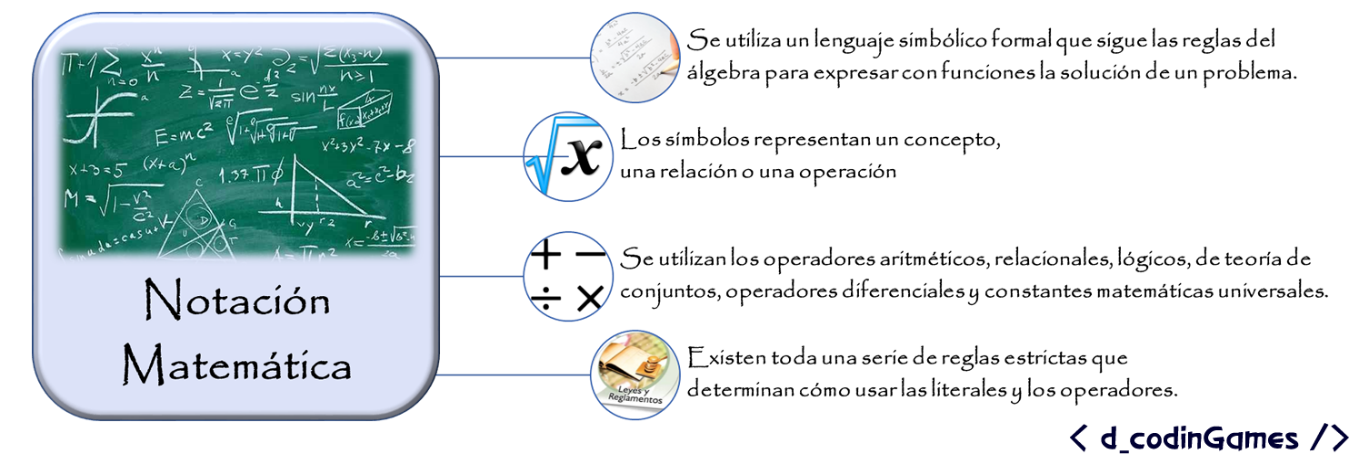
Notación matemática.

Otra forma de escribir algoritmos es usando la clásica y siempre socorrida notación matemática. Fórmulas como:



Son algoritmos que ya nos indican la forma de calcular algo, es decir, los pasos que debemos realizar para resolver un problema.

En la figura 3 muestro las características más relevantes de este tipo de notación. Aunque tiene muchas ventajas en comparación con la descripción narrada, desafortunadamente (y aunque los matemáticos lo nieguen) no todos los problemas se pueden resolver con fórmulas matemáticas, de ahí que tengamos que buscar otras formas de representar a los algoritmos.

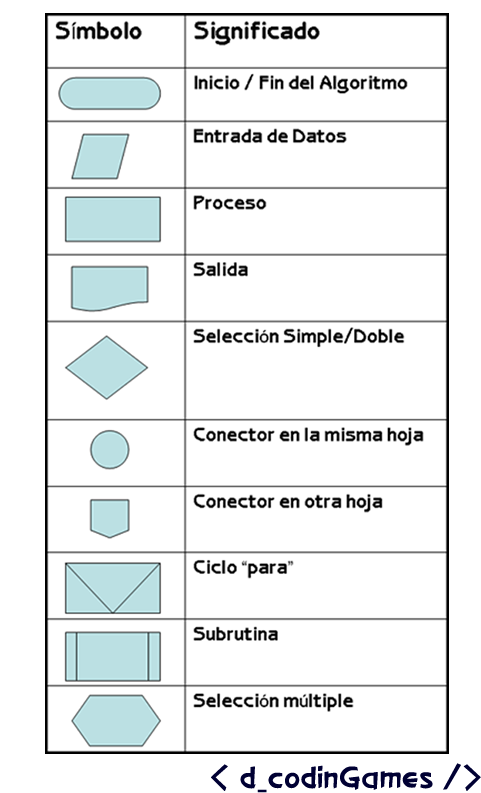


*Figura 3. Características de la notación matemática*

Diagrama de flujo.

En esta notación se utilizan símbolos gráficos que representan cada paso del algoritmo. Es la notación más usada para escribir algoritmos. Aunque ya existían desde los años 20’s de siglo pasado, se cree que [Herman Goldstine](https://ztfnews.wordpress.com/2013/09/13/centenario-del-nacimiento-de-herman-goldstine/) y [Jon Von Neumann](https://www.biografiasyvidas.com/biografia/n/neumann.htm) fueron los primeros en utilizarlos para planificar un programa de computadora. Su popularidad tuvo su pico más alto en la década de los 70’s, pero actualmente ya no se usan con tanta frecuencia debido a que existen otras notaciones y también debido a la aparición de los lenguajes de programación de tercera generación. También se utilizan en el ambiente industrial, pues son fáciles de leer y son excelentes auxiliares para describir procesos.

En la figura 4 muestro los símbolos más comunes que se utilizan en esta notación.



*Figura 4. Simbología utilizada en los diagramas de flujo*

Las ventajas de los diagramas de flujo son:

* Al ser muy visuales, favorecen la comprensión de los procesos y/o actividades, así como de la secuencia que se seguirá.
* Es útil para enseñar a programar. Se ha comprobado que el cerebro asocia de forma más rápida lo que dibuja que lo que ve o escucha.
* Ayuda a organizar los pensamientos y el proceso de creación de un algoritmo.

Su principal desventaja es cuando el problema es muy grande y dibujar un diagrama de flujo puede volverse tedioso. También he visto que a los estudiantes se les dificulta traducir a código las estructuras anidadas.

Algunas observaciones que considero necesarias para escribir buenos diagramas de flujo que nos ayuden a crear programas eficientes son:

* Es importante conservar los diagramas de flujo lo más sencillo posible, y apoyarse en el significado de la simbología. No se debe escribir demasiado texto dentro de los símbolos, es mejor utilizarlas variables y las expresiones aritméticas, relacionales o lógicas según sea el caso.
* Para un buen diseño del algoritmo, y ayudar a la codificación posterior, es necesario conservar la “verticalidad” del diagrama.
* Es muy importante utilizar los símbolos adecuados para cada operación a realizar. Por ejemplo, en el símbolo condicional, no deben incluirse asignaciones; ni en el símbolo de proceso se pueden incluir operadores relacionales ni lógicos.

Pseudocódigo.

El prefijo *pseudo* significa falso, según la [RAE](http://dle.rae.es/?id=XkBx392), y también es admisible escribirlo: seudo.

El pseudocódigo es un lenguaje informal para la descripción de algoritmos. Es una notación que es muy flexible y depende del estilo del programador. Toma características de un lenguaje de programación y de los procesos matemáticos para detallar el problema.

[Wikipedia](https://es.wikipedia.org/wiki/Pseudoc%C3%B3digo)también lo define como: una descripción de alto nivel compacta e informal del principio operativo de un programa informático u otro algoritmo.

Lo mejor de esta notación es que utiliza las convenciones estructurales de un lenguaje de programación, pero es independiente de un lenguaje específico. Además, está diseñado para que los seres humanos lo leamos y podamos entender. En un pseudocódigo:

* No se declaran variables
* Las funciones específicas de los lenguajes de programación se pueden sustituir ya sea por una oración en lenguaje natural, o una “función” que definamos para tal propósito. Por ejemplo, en Java para imprimir el valor de una variable se escribe:

System.out.println(“El valor de x es: “ + x);

Lo que en pseudocódigo puede quedar indicado como alguna de las siguientes formas:

Salida(x)

Imprimir(x)

Imprime el valor de x

* Se debe delimitar el inicio y el fin del algoritmo
* Se utilizan las estructuras de control de flujo básicas de cualquier lenguaje de programación moderno.
* Utilizar las reglas comunes a la mayoría de los lenguajes de programación para la creación de identificadores de variables y/o constantes; y ser coherentes y constantes con estos nombres.
* El programador define su estilo, pero es importante que sea fiel y constante con este estilo a lo largo de la escritura del algoritmo. Por ejemplo, alguien podría usar las palabras *Inicio* y *Fin* para abrir y cerrar bloques de código que pertenecen a las estructuras de control de flujo o usar las { } que se utilizan en lenguajes como C y Java, o incluso las palabras *BEGIN* y *END* que se usan en el lenguaje Pascal. Pero debe ser consistente con su estilo.
* Es conveniente y deseable el uso de la sangría. No sólo forma un buen hábito, sino que también hace al algoritmo legible. ([Aquí](https://dcodingames.com/identacion-y-llaves/) puedes leer un poco más sobre el uso de la sangría)

Algunas de las ventajas que yo le encuentro al pseudocódigo por sobre el uso de diagramas de flujo son:

* Ocupan menos espacio en una hoja de papel
* Es más fácil escribir estructuras de control anidadas
* El proceso de codificación es mucho más sencillo y “natural”
* Si se utiliza correctamente la identación, son más fáciles de leer que los diagramas de flujo.
* Aquél que sabe leer pseudocódigo, sabe leer código.

¿Desventajas? Yo no le encuentro, pero el programador debe ser constante con su estilo, y utilizar la identación adecuadamente.

Recuerda que es tan algoritmo el diagrama de flujo como el pseudocódigo, simplemente están escritos en forma diferente.  Y al final lo importante es que escribas el algoritmo antes de empezar a codificar, no sólo cuando estés en la fase de aprendiz, sino también como programador experimentado, te dará una vista previa de las acciones necesarias para resolver el problema y sobre todo ayudará a entender el programa o sistema generado para depuración de errores y/o futuras modificaciones.

El algoritmo es sumamente importante porque es **el primer elemento de abstracción** que nos lleva a la creación del programa que esperamos resuelva el problema. Todo programador necesita saber escribirlos, así como también leerlos. Contribuyen con el proceso de creación de la solución, ayudan al entendimiento de la lógica de programación y sobre todo, son una herramienta esencial de la documentación del programa.

No los tomes a la ligera. Ya sea que uses diagramas de flujo, que escribas una sencilla descripción narrada o que utilices un pseudocódigo (que personalmente considero es lo ideal), todos los programadores necesitamos escribirlos. Aún aquellas personas que ya tienen un nivel avanzado en programación necesitan un algoritmo. No es una pérdida de tiempo como algunos dicen. Al contrario, se pierde más tiempo si se empieza a programar algo que no se ha comprendido bien.

Recuerda que el algoritmo no sólo te ayudará a diseñar la solución, sino también a entender el problema.

Y tú, ¿qué tipo de notación prefieres? ¿Se te da eso de escribir algoritmos o pasas directo a inspirarte en la máquina? Opina escribiendo algún comentario, y ya sabes, si la información te resulta útil no olvides compartirla.

¡Hasta la próxima!