

LECTURA MÓDULO INTRODUCCIÓN A LA LÓGICA DE LA PROGRAMACIÓN

CONTENIDOS:

- Algoritmos: qué son, cómo se desarrollan y cómo se representan
- Pensamiento lógico: desarrollo del pensamiento lógico.
- Paradigmas de programación: programación estructurada, programación orientada a objetos, programación funcional, entre otros.

ALGORITMOS: QUÉ SON, CÓMO SE DESARROLLAN Y CÓMO SE REPRESENTAN

Un algoritmo se define como un “Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema”.

En muchas ocasiones, al oír hablar de algoritmos pensamos que se encuentran limitados al área de las matemáticas y la programación, pero esto no es correcto. En nuestra vida cotidiana vivimos rodeados de ellos; por ejemplo, al seguir una receta para preparar un delicioso pastel, o al realizar cada uno de los pasos para el armado de un nuevo estante para nuestra casa siguiendo el manual.

Teniendo en cuenta que los algoritmos no se encuentran solo relacionados a la programación y las matemáticas, y que convivimos con ellos a diario, nos centraremos en el algoritmo para programación.

En esta área es muy importante tener claridad sobre el algoritmo necesario para alcanzar la solución a un problema, porque este es el paso previo y fundamental antes de comenzar a escribir código. Es decir, primero nos encargamos de encontrar la solución al problema a través de un paso a paso ordenado y lógico (al alcanzar la solución, finaliza el algoritmo), y luego, a través de la escritura del código (independiente del lenguaje de programación utilizado) le indicamos a la máquina qué acciones debe llevar a cabo.

Un algoritmo se compone de 3 partes fundamentales: un input, un proceso, y un output. El input (entrada) es la información que damos al inicio del proceso, y con la cual va a trabajar el algoritmo para alcanzar la solución esperada. El proceso es el conjunto de pasos (finitos) que se llevarán a cabo a partir de la información de entrada para alcanzar la solución. Por su parte, el output (salida) es el resultado final, es decir, el momento en que se alcanza la solución al problema planteado y, por lo tanto, la finalización del algoritmo.

Es necesario mencionar que los algoritmos tienen características comunes: son precisos, ordenados, finitos, concretos, y definidos.

De esta forma podemos resumir todo lo anterior en que un algoritmo es un conjunto de pasos ordenados lógicamente para alcanzar la solución a un problema planteado, y además es necesario tener en cuenta que siempre tendrá un inicio y un final.

¿CÓMO SE REPRESENTA UN ALGORITMO?

Luego de conocer la definición de algoritmo, y comprender que este es fundamental para el proceso de programación de software, veamos sus formas de representación:

- **Lenguaje Natural:** instrucciones dadas en el lenguaje que utilizamos (español para este caso), con una secuencia lógica que soluciona un problema. Por ejemplo, una receta de cocina.

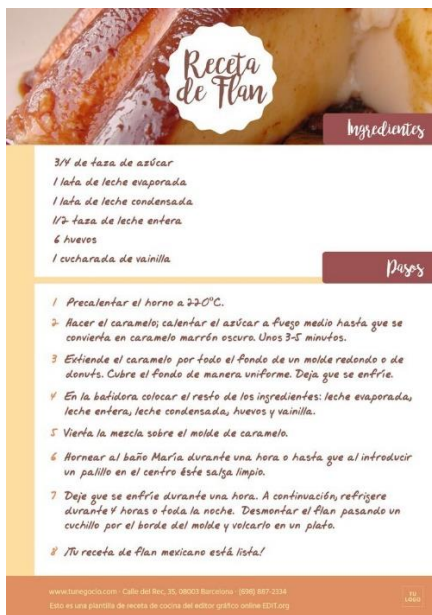


Ilustración 1: Algoritmo en lenguaje natural.

- **Diagrama de flujo:** forma visual o esquemática de representar la secuencia de pasos de un algoritmo. Permite que sea entendible para personas ajenas a la programación, y cuenta con símbolos bien definidos.

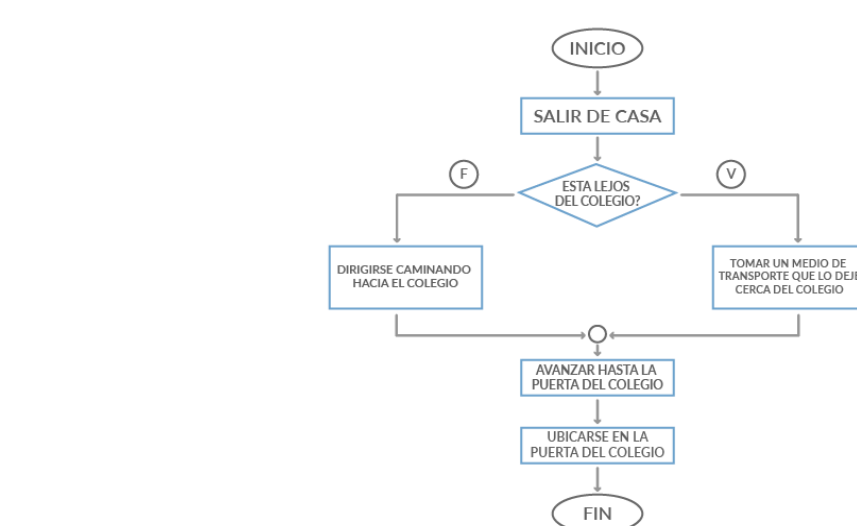


Ilustración 2: Algoritmo en diagrama de flujo.

- **Pseudocódigo:** nos permite detallar los pasos de nuestro algoritmo, siendo un punto intermedio entre el lenguaje natural y el lenguaje de programación.

La imagen muestra una interfaz de usuario con un editor de texto a la izquierda y una ventana de ejecución a la derecha.

Editor de texto (Pseudocódigo):

```

1 Algoritmo Tabla_Multiplicar
2   Escribir 'Ingresa el número a multiplicar'
3   Leer n
4   Escribir '-----'
5   Para x<-1 Hasta 10 Hacer
6     Escribir '|' ' ' , n , ' * ' , x , ' = ' , n*x
7   FinPara
8   Escribir '-----'
9   FinAlgoritmo
  
```

Ventana de ejecución (PSeint):

```

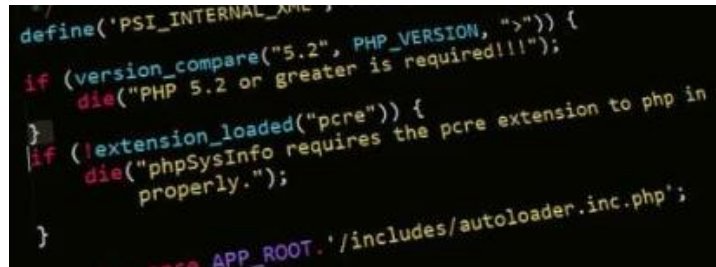
*** Ejecución Iniciada. ***
Ingresa el número a multiplicar
> 9

| 9 * 1 = 9
| 9 * 2 = 18
| 9 * 3 = 27
| 9 * 4 = 36
| 9 * 5 = 45
| 9 * 6 = 54
| 9 * 7 = 63
| 9 * 8 = 72
| 9 * 9 = 81
| 9 * 10 = 90

*** Ejecución Finalizada. ***
  
```

Ilustración 3: Algoritmo en pseudocódigo.

- **Lenguaje de programación:** escritura del algoritmo en un lenguaje comprensible por la máquina para dar instrucciones de las acciones que debe llevar a cabo.



```
define('PSI_INTERNAL_XML', '1.0.0');  
if (version_compare("5.2", PHP_VERSION, ">")) {  
    die("PHP 5.2 or greater is required!!!");  
}  
if (!extension_loaded("pcre")) {  
    die("phpSysInfo requires the pcre extension to php in o  
        properly.");  
}  
define('APP_ROOT', './includes/autoloader.inc.php');
```

Ilustración 4: Algoritmo en lenguaje de programación.

PENSAMIENTO LÓGICO: DESARROLLO DEL PENSAMIENTO LÓGICO

El pensamiento lógico es la capacidad de razonar y tomar decisiones de manera estructurada y coherente. Se basa en la observación y análisis de información para llegar a conclusiones precisas y fundamentadas. Este permite identificar patrones, solucionar problemas, y tomar decisiones informadas.

Para desarrollar el pensamiento lógico es importante fomentar la curiosidad y la exploración, así como la capacidad de hacer preguntas y cuestionar las cosas. También se pueden utilizar herramientas como juegos de ingenio, rompecabezas, acertijos, juegos de mesa, y programas de capacitación en línea. Además, la práctica de la resolución de problemas y la toma de decisiones basadas en datos concretos son fundamentales para el desarrollo de éste. La lectura y el estudio de disciplinas como la matemática, la lógica, y la filosofía también pueden ayudar a desarrollar esta habilidad.

Una forma clásica de trabajar el pensamiento lógico es con la Torre de Hanoi. Este puede parecer un simple juego de niños, pero permite desarrollar el pensamiento lógico a través de la generación de un algoritmo para alcanzar la solución al problema.

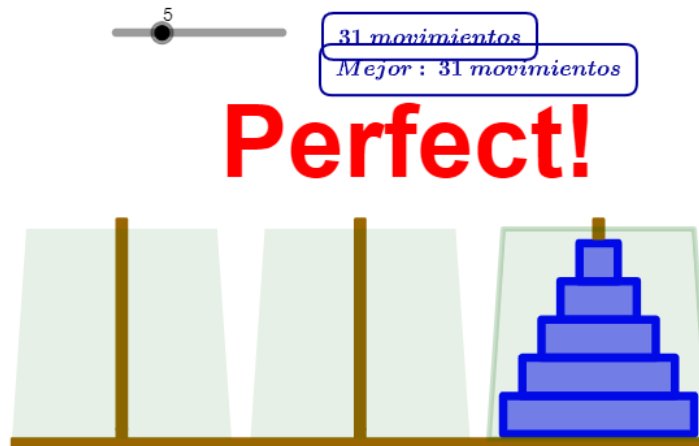


Ilustración 5: Ejemplo Torre de Hanoi.

Este juego consiste en tres varillas verticales y un número determinado de discos de diferentes tamaños, los cuales deben ser movidos de una varilla a otra, procurando dejar siempre el de mayor tamaño abajo, y el más pequeño arriba. Si es de interés, pueden intentar realizarlo en la página de [GeoGebra](#).

PARADIGMAS DE PROGRAMACIÓN

Son enfoques o modelos de pensamiento que guían la forma en que se desarrolla el software. Cada paradigma establece reglas y principios para organizar y estructurar el código. Hay varios paradigmas, como la programación imperativa, la orientada a objetos, la funcional y lógica, entre otros. Cada uno tiene su propia forma de abordar la resolución de problemas y la manipulación de datos. Estos ayudan a los desarrolladores a crear softwares eficientes y escalables, permitiendo diferentes formas de razonamiento y diseño, adaptándose a las necesidades específicas de cada proyecto, y facilitando la reutilización de código.

A continuación, se detallan cinco tipos de paradigmas:

PARADIGMA DE PROGRAMACIÓN IMPERATIVO

Es un enfoque que se centra en la descripción detallada de cómo realizar una serie de acciones para alcanzar un objetivo. Se basa en instrucciones explícitas y secuenciales, donde se especifica paso a paso cómo se debe ejecutar el programa.

En este paradigma, el código se divide en bloques de instrucciones que modifican el estado del programa a medida que se ejecutan. Se utilizan variables para almacenar y manipular datos, y se emplean estructuras de control, como bucles y condicionales, para controlar el flujo de ejecución.

El paradigma imperativo se aplica en una amplia gama de lenguajes de programación, como C, Java y Python. Permite una programación detallada y controlada, lo que facilita el seguimiento y la depuración del código. Sin embargo, también puede llevar a la escritura de programas extensos y propensos a errores si no se planifica y organiza adecuadamente.

PARADIGMA DE PROGRAMACIÓN FUNCIONAL

Es un enfoque que se basa en la evaluación de funciones matemáticas para resolver problemas. En este se considera que un programa es la composición de múltiples funciones, evitando el uso y cambios de estados.

Aquí las funciones se tratan como ciudadanos de primera clase, lo que significa que se pueden pasar como argumentos, devolver como resultados, y asignar a variables. Además, las funciones son puras, lo que implica que su resultado solo depende de sus argumentos y no producen efectos secundarios.

La aplicación del paradigma funcional se realiza utilizando lenguajes de programación como Haskell, Lisp y Erlang. Se enfoca en la inmutabilidad de los datos y la recursión (proceso en el cual la función se llama a sí misma) en lugar de bucles. Este enfoque promueve la legibilidad del código, la modularidad, y la reutilización de funciones. Al evitar los cambios de estado y los efectos secundarios, también facilita la concurrencia (ejecución de tareas paralelas) y el manejo de errores.



PARADIGMA DE PROGRAMACIÓN DECLARATIVO

Se centra en describir qué se quiere lograr, en lugar de como se debe hacer. En este enfoque, el programador se concentra en establecer una serie de condiciones o restricciones que el programa debe cumplir, y luego se delega al sistema el proceso de encontrar la solución.

En este paradigma se utilizan lenguajes como Prolog y SQL. Tiene ventajas como la abstracción de los detalles de implementación, y la posibilidad de expresar soluciones más concisas y elegantes. Por otro lado, puede no ser adecuado para todos los escenarios, especialmente cuando se requiere un control preciso del flujo de ejecución, o cuando se trabaja con problemas donde la eficiencia es fundamental.

PARADIGMA DE PROGRAMACIÓN REACTIVO

Este se enfoca en la programación de sistemas que reaccionan y responden de manera automática a los cambios en los datos o eventos del entorno. Se basa en la propagación de cambios, y en la capacidad de los componentes para ser notificados y responder de manera eficiente.

Aquí los sistemas se modelan como flujos de eventos o datos. Los componentes reactivos se conectan entre sí mediante suscripciones, y propagan los cambios a medida que ocurren. Esto permite que los sistemas sean altamente responsivos y adaptables a medida que evoluciona su entorno.


Algunos lenguajes que trabajan con enfoque reactivo son Java, JavaScript y TypeScript.

PARADIGMA DE PROGRAMACIÓN ORIENTADA A OBJETOS

Es un enfoque que organiza el software en torno a objetos, los cuales son entidades que combinan datos y comportamientos relacionados. En POO, se modelan los objetos del mundo real como clases, que actúan como plantillas para crear instancias de objetos.

En este paradigma, los objetos interactúan entre sí a través del intercambio de mensajes, donde cada uno tiene su propio estado interno y método para manipularlo y realizar acciones específicas. Se enfatiza la encapsulación, el ocultamiento de información, y la reutilización de código.

La aplicación de la programación orientada a objetos se realiza mediante lenguajes como Java, C++ y Python. Permite crear sistemas modulares y escalables, ya que los objetos se pueden diseñar, crear



y mantener de manera independiente. Además, la herencia y el polimorfismo son características claves que permiten extender y especializar las clases existentes, fomentando la flexibilidad y la modularidad del código.

Es ampliamente utilizado en el desarrollo de aplicaciones complejas, como sistemas de gestión, videojuegos y aplicaciones empresariales. Proporciona un enfoque intuitivo para modelar el mundo real, y para estructurar el código, lo que facilita el mantenimiento, la colaboración y la reutilización de software.

Puede ser que leer sobre los paradigmas de programación suene complejo en un inicio, y te preguntes ¿de qué me sirve saber esto? En estos momentos puede parecer no tener importancia, pero en un futuro, cuando estés sumergido en el mundo de la programación, notarás el valor de tener conocimiento sobre éstos.