
Generalmente, la primera razón que mueve a una persona hacia el aprendizaje de la programación es utilizar el ordenador como herramienta para resolver problemas concretos.

¿Qué virtudes debería tener nuestra solución?

- **Corrección y eficacia:** si resuelve el problema adecuadamente.
- **Eficiencia:** si lo hace en un tiempo mínimo y con un uso óptimo de los recursos del sistema.

Para conseguirlo, cuando afrontemos la construcción de la solución tendremos que tener en cuenta los siguientes conceptos:

1. **Abstracción:** se trata de realizar un análisis del problema para descomponerlo en problemas más pequeños y de menor complejidad, describiendo cada uno de ellos de manera precisa. ***Divide y vencerás,*** ésta suele ser considerada una filosofía general para resolver problemas y de aquí que su nombre no sólo forme parte del vocabulario informático, sino que también se utiliza en muchos otros ámbitos.
2. **Encapsulación:** consiste en ocultar la información para poder implementarla de diferentes maneras sin que esto influya en el resto de elementos.
3. **Modularidad:** estructuraremos cada parte en módulos independientes, cada uno de ellos tendrá su función correspondiente.

Después de analizar en detalle el problema a solucionar, hemos de diseñar y desarrollar el algoritmo adecuado. Pero, **¿qué es un algoritmo?**

Algoritmo: secuencia ordenada de pasos, descrita sin ambigüedades, que conducen a la solución de un problema dado.
La diferencia fundamental entre algoritmo y **programa** es que, en el segundo, los pasos que permiten resolver el problema, deben escribirse en un determinado **lenguaje de programación** para que puedan ser ejecutados en el ordenador y así obtener la solución.

Los lenguajes de programación son sólo un medio para expresar el algoritmo y el ordenador un procesador para ejecutarlo. El diseño de los algoritmos será una tarea que necesitará de la creatividad y conocimientos de las técnicas de programación. Estilos distintos, de distintos programadores a la hora de obtener la solución del problema, darán lugar a algoritmos diferentes, igualmente válidos.

En esencia, todo problema se puede describir por medio de un algoritmo y las características fundamentales que éstos deben cumplir son:

- Debe ser **preciso** e indicar el orden de realización paso a paso.
- Debe estar **definido**, si se ejecuta dos o más veces, debe obtener el mismo resultado cada vez.
- Debe ser **finito**, debe tener un número finito de pasos.

Pero cuando los problemas son complejos, es necesario descomponer éstos en sub-problemas más simples y, a su vez, en otros más pequeños. Estas estrategias reciben el nombre de **diseño descendente** o **diseño modular** (**top-down design**). Este sistema se basa en el lema **divide y vencerás**.

Para representar gráficamente los algoritmos que vamos a diseñar, tenemos a nuestra disposición diferentes herramientas que ayudarán a describir su comportamiento de una forma precisa y genérica, para luego poder codificarlos con el lenguaje que nos interese. Entre otras tenemos:

- **Diagramas de flujo:** Esta técnica utiliza símbolos gráficos para la representación del algoritmo. Suele utilizarse en las fases de análisis.
- **Pseudocódigo:** Esta técnica se basa en el uso de palabras clave en lenguaje natural, constantes, variables, otros objetos, instrucciones y estructuras de programación que expresan de forma escrita la solución del problema. Es la técnica más utilizada actualmente.
- **Tablas de decisión:** En una tabla son representadas las posibles condiciones del problema con sus respectivas acciones. Suele ser una técnica de apoyo al pseudocódigo cuando existen situaciones condicionales complejas.

Debes conocer

A continuación te ofrecemos dos enlaces muy interesantes:

- En el primer enlace puedes ver los elementos gráficos fundamentales que se utilizan para la generación de diagramas de flujo.

Significado de los símbolos de diagrama de flujo de procesos – Flujograma

El primer símbolo que se muestra es la flecha, un símbolo de conexión utilizado para indicar una interconexión entre otros dos símbolos, y la dirección del flujo.

A continuación, vamos a mostrar varios diseños de formas con el respectivo significado de estos símbolos en el diagrama de flujo del proceso.

Nombre: Terminación

Significado del símbolo en diagramas de flujo: **Indica el comienzo o el final de un flujo en el diagrama de procesos.**

Nombre: Proceso

Significado del símbolo en diagramas de flujo: **Indica un determinado proceso y sus funciones y actividades.**

Nombre: Decisión

Significado del símbolo en diagramas de flujo: **Esto demuestra que se debe tomar una decisión y que el flujo del proceso va a seguir cierta dirección según esta decisión.**

Nombre: Retardo

Significado del símbolo en diagramas de flujo: **Significa que pasará un tiempo antes de que el flujo del proceso continúe.**

Nombre: Datos

El segundo recurso, consiste en una aplicación que permite construir diagramas de flujo para la resolución de tareas mediante algoritmos. El programa se llama DFD, y se puede obtener de https://goo.gl/8M9g5c

Se trata de una aplicación portable. Tras su descarga, simplemente lanzamos el ejecutable

El tercer recurso permite la construcción de un diagrama de flujo con otra herramienta gráfica y su transformación a pseudocódigo. Se trata del programa PSeInt:

http://pseint.sourceforge.net/

3. Paradigmas de la programación.

¿Cuántas formas existen de hacer las cosas? Supongo que estarás pensando: varias o incluso, muchas. Pero cuando se establece un patrón para la creación de aplicaciones nos estamos acercando al significado de la palabra paradigma.

Paradigma de programación: es un modelo básico para el diseño y la implementación de programas. Este modelo determinará como será el proceso de diseño y la estructura final del programa.

El paradigma representa un enfoque particular o filosofía para la construcción de software. Cada uno tendrá sus ventajas e inconvenientes, será más o menos apropiado, pero no es correcto decir que exista uno mejor que los demás.

Como habrás podido apreciar, existen múltiples paradigmas, incluso puede haber lenguajes de programación que no se clasifiquen únicamente dentro de uno de ellos. Un lenguaje como **Smalltalk** es un lenguaje basado en el paradigma orientado a objetos. El lenguaje de programación **Scheme**, en cambio, soporta sólo programación funcional. **Phyton**, soporta múltiples paradigmas.

Para saber más

Te proponemos el siguiente enlace en el que encontrarás información adicional sobre los diferentes paradigmas de programación.

Paradigmas de programación y lenguajes

¿Cuál es el objetivo que se busca con la aplicación de los diferentes enfoques? Fundamentalmente, reducir la dificultad para el mantenimiento de las aplicaciones, mejorar el rendimiento del programador y, en general, mejorar la productividad y calidad de los programas.