LENGUAJES DE PROGRAMACIÓN

La informática, derivada de "información" y "automática", es la ciencia que estudia el tratamiento automático de la información, especialmente a través de computadoras. Utiliza el código binario (secuencia de 0 y 1) para representar datos. Los dispositivos que utilizan información en forma de bits se consideran digitales, mientras que los que no lo hacen se llaman analógicos.

El software es la parte intangible de un sistema informático, dividido en aplicaciones (realizan tareas específicas) y sistemas operativos (proporcionan una plataforma para el uso de la máquina). Programar implica especificar instrucciones y datos para que un dispositivo los ejecute, mientras que desarrollar software implica analizar, diseñar, probar y mantener el software en general.

Los lenguajes de programación se clasifican según el nivel de abstracción y el tipo de ejecución. Los lenguajes de alto nivel, más cercanos al lenguaje natural, son preferidos por los desarrolladores. Pueden ser interpretados (traducción instrucción por instrucción) o compilados (traducción previa a un archivo ejecutable). Existen también lenguajes intermedios o bytecode, que combinan características de ambos.

Java es un ejemplo de lenguaje intermedio, donde el código fuente se compila a bytecode, que luego se ejecuta en la Máquina Virtual de Java (JVM).

Los lenguajes también se clasifican según su paradigma, como imperativo (basado en instrucciones) o declarativo (fijando objetivos sin especificar el camino). Dentro de los imperativos, hay estructurados y orientados a objetos.

La elección del lenguaje de programación depende del contexto, como el sector, la existencia de bibliotecas, el tipo de dispositivo y el sistema operativo.

ENTORNOS DE DESARROLLO

Un Entorno de Desarrollo Integrado (IDE) es una aplicación informática diseñada para ayudar al programador en la tarea de diseñar y codificar software, integrando múltiples herramientas para facilitar la codificación y desarrollo.

Los elementos básicos de un IDE incluyen un editor de texto para escribir y modificar el código fuente, un compilador o intérprete para traducir y ejecutar el código, y un depurador para encontrar errores lógicos en el código.

Los IDEs modernos ofrecen características como resaltado de sintaxis, autocompletado de código, macros y facilitan la visualización del código. Aunque son útiles para la programación, generan dependencia y consumen más recursos.

Los motivos para utilizar un IDE incluyen funciones avanzadas como autocompletado de código, creación automática de estructuras y la integración con herramientas de control de versiones. Además, son altamente configurables según las preferencias del desarrollador.

Al elegir un IDE, se deben considerar factores como el sistema operativo, el lenguaje de programación, el framework utilizado, las herramientas disponibles y la licencia del IDE.

El depurador es una herramienta fundamental que permite visualizar el flujo de ejecución de un programa, detectar errores lógicos y observar el valor de las variables en diferentes puntos del código. Los depuradores tienen funciones básicas como breakpoints, resumen de ejecución y pasos a través del código.

Se muestra un ejemplo de uso de un depurador en Java, donde se configura un breakpoint, se inicia la ejecución en modo depuración, se observan las variables y se utilizan funciones básicas del depurador.

En resumen, los IDEs son herramientas esenciales para simplificar y agilizar el desarrollo de software, proporcionando funciones avanzadas y facilitando la tarea del programador.

QUE ES LA INGENIERA DEL PROGRAMARIO?

La ingeniería del software agrupa un conjunto de herramientas y métodos utilizados para desarrollar aplicaciones o sistemas informáticos que resuelven problemas o satisfacen necesidades específicas

CICLO DE VIDA DEL PROGRAMA

Análisis: Se identifican y especifican las necesidades del cliente.

Diseño: Se plantea una solución para la aplicación y se crea un modelo.

Codificación o Implementación: Se desarrolla el programa atendiendo a todos sus componentes.

Pruebas: Se realizan revisiones para detectar errores, incluyendo pruebas de usabilidad.

Documentación: Se documentan las decisiones técnicas tomadas durante el desarrollo.

Implantación o Despliegue: Se publica la solución final en la plataforma destinada o se entrega al cliente.

Mantenimiento: Se actualiza la aplicación y se corrigen errores.

Tipos de Ciclos de Vida:

Clásico o Cascada: Estructura lineal sin retroalimentación, válido para proyectos con requisitos claros.

Iterativo: Estructura iterativa donde se repite en bucle el sistema lineal, permitiendo ajustes continuos.

Espiral: Basado en el modelo iterativo, añade gestión de riesgos para determinar la viabilidad del desarrollo.