

trabajo sobre: lenguajes de PROGRAMACION.

AIU



sergio pedroso monterrey

Lenguajes de programación:

Un lenguaje de programación es un conjunto de palabras, símbolos y reglas que permiten construir programas para computadoras.

Estudiar las características de diferentes lenguajes de programación permite comprender las ventajas y desventajas de estos lenguajes para así poder seleccionar el lenguaje más conveniente.

No todos los lenguajes de programación son iguales clasificación:

Los lenguajes de programación se utilizan para hacer funcionar las páginas web, las apps, el software y todo tipo de dispositivos que requieren programación informática y conocimientos de códigos.

De acuerdo a su finalidad la clasificación de lenguajes de programación son las siguientes:

1. Lenguaje de máquina
2. Lenguaje de programación de bajo nivel
3. Lenguaje de programación de alto nivel

En grandes rasgos, el lenguaje de máquina es el que usa cualquier máquina y se basa en un código binario; Los lenguajes de bajo nivel, pueden variar según el ordenador o máquina que se utilice; y, por último, el lenguaje de alto nivel es un lenguaje que utiliza comandos y palabras (normalmente en inglés), las cuales es fácil de entender para un programador.

Sobre esta última clasificación de lenguaje de programación, los de alto nivel, son sobre los que hablamos, cuando hablamos de programación de web y apps.

Lenguajes de bajo nivel:

Un lenguaje de programación de características de bajo nivel o de primera generación es aquel en el que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan. El uso de la palabra ajo es una denominación no implica que el lenguaje sea menos potente que un lenguaje de alto nivel, sino que se refiere a la reducida abstracción entre el lenguaje y el hardware.

Lenguajes de alto nivel:

Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con la que las máquinas lo ejecutan. Estos lenguajes permiten una máxima flexibilidad al programador a la hora de abstraerse o de ser literal. Permite un camino bidireccional entre el lenguaje de máquina y una expresión casi oral entre la escritura del programa y su posterior compilación. Por lo general suelen estar orientados a objetos, a eventos o a funciones, pudiendo estos combinarse. Asimismo, pueden ser compilados o interpretados. algunos ejemplos son: Java, PHP, Python, JavaScript, C#.

Lenguajes compilados vs interpretados:

En el lenguaje compilado, la máquina de destino traduce directamente el programa. En un lenguaje interpretado, el código fuente no es directamente traducido por la máquina de destino. En cambio, un distinto programa conocido como interprete lee el código y lo ejecuta.

Los lenguajes compilados son convertidos directamente a código de máquina que el procesador puede ejecutar. Como resultado, suelen ser más rápidos y más eficientes al ejecutarse en comparación con los lenguajes interpretados. También le dan al desarrollador más control sobre aspectos del hardware, como la gestión de memoria y uso del CPU.

Los lenguajes compilados necesitan un paso de compilación (build)- primero necesitan compilarse manualmente. Necesitas ´´recompilar´´ el programa cada vez que necesites hacer un cambio. En nuestro ejemplo des hummus, la traducción completa está escrita antes de que la tengas. Si el autor original decide que quiere usar un distinto aceite de oliva, toda la receta necesita ser traducida de nuevo y reenviada a ti.

Algunos ejemplos de lenguajes compilados puros son: C, C++, ERLANG, HASHELL, RUST y GO.

Lenguajes interpretados

Estos lenguajes ejecutan línea por línea el programa y a la vez ejecutan cada comando.

Aquí, si el autor decide que quiere usar un distinto aceite de oliva, podrá borrar el anterior y agregar el nuevo. Tu amigo el traductor puede decirte ese cambio a medida que sucede.

Los lenguajes interpretados alguna vez fueron significativamente mas lentos que los lenguajes compilados. Pero, con el desarrollo de la compilación justo a tiempo, esa diferencia se está reduciendo.

Ejemplos comunes de lenguajes interpretados son PHP, RUBY, PYTHON Y JavaScript

Clasificación de lenguajes de acuerdo a su generación

Los lenguajes de programación han sido clasificados también en diferentes generaciones que históricamente han el poder de estilo de programación.

Existen 5 generaciones de lenguajes de programación:

1-Los lenguajes de primera generación -1GL, son lenguajes de bajo nivel conocidos como lenguaje de máquina. Se utilizan para programar las computadoras a un nivel muy bajo de abstracción.

2-los lenguajes de segunda generación -2GL, también son lenguajes ensambladores de bajo nivel. Se usan en kernels y unidades de hardware, además de ser usados comúnmente para edición de video, gráficos y videojuegos.

3-lenguajes de tercera generación 3GL, son lenguajes de alto nivel, como C, C++, C#, Java, BASIC, JavaScript y Visual Basic.

4-los lenguajes de cuarta generación 4GL, son lenguajes que consisten en declaraciones similares a las declaraciones hechas en un lenguaje humano. Los lenguajes de cuarta generación se usan normalmente en la programación de bases de datos y los ejemplos de scripts incluyen Unix Shell, Perl, PHP, Python, Ruby y SQL.

5-Los lenguajes de programación de quinta generación 5GL, son lenguajes de programación que contienen herramientas visuales para ayudar a desarrollar un programa. Ejemplos de lenguajes de 5ta generación incluyen Mercury 0PS5 Y Prolog.

En resumen, los lenguajes de programaciones 1GL y 2GL son también de bajo nivel y los de las generaciones 3GL, 4GL Y 5GL son de alto nivel.

Aspectos importantes de la programación moderna:

La programación visual permite construir parte o todo el programa mediante símbolos gráficos, es usada frecuentemente para crear programas con una interfaz moderna, programas como VISUAL BASIC, JAVA Y C# permiten crear la interfaz gráfica mediante programación visual.

Los lenguajes interpretados, tales como los scriptings lenguajes, son muy utilizados hoy en día en la creación de sistemas de información para la web.

Paradigma de programación

La programación engloba diferentes enfoques y técnicas guían el desarrollo de software.

Estos enfoques, conocidos como paradigmas de programación, ofrecen distintas filosofías y metodologías para abordar problemas computacionales. Desde la clásica programación imperativa hasta los modernos paradigmas orientados a objetos funcionales, cada uno tiene su propias fortalezas, debilidades y aplicaciones específicas.

En este post, nos adentraremos en el mundo de los paradigmas de programación. Exploraremos como cada uno de ellos aborda la resolución de problemas, que los hace únicos y como influyen en la forma en que los programadores piensan y desarrollan software.

Clasificación de acuerdo al paradigma de programación

Los paradigmas de programación son enfoques alternativos para construir programas. Tradicionalmente, los paradigmas más importantes son los siguientes:

Imperativo a procedimental

Orientado a objetos

Funcional

Declarativo

Paradigma imperativo

El paradigma imperativo o de procedimiento es, probablemente uno de los paradigmas más conocidos en el mundo de la programación. Como su nombre lo indica, este es un método que permite desarrollar programas a través de procedimientos. Mediante una serie de instrucciones, se explica paso por paso cómo funciona el código para que el proceso sea lo más claro posible.

Programación imperativa

La programación imperativa se centra en describir paso a paso como realizar una tarea. En este enfoque, un programa se compone de una serie de instrucciones que modifican el estado del programa a lo largo del tiempo. Es como darle a la computadora una lista detallada de tareas que realizar, una tras otra.

Programación orientada a objetos

La programación orientada a objetos es un modelo de programación en la que el diseño se software se organiza alrededor de datos u objetos, en vez de usar funciones y lógica. Se enfoca en los objetos que lo programadores necesitan manipular, en lugar de centrarse en la lógica necesaria para esa manipulación. un objeto se puede definir como un campo de datos con atributos y comportamientos únicos.

Por tanto, la principal característica de este tipo de programación es que soporta objetos, que tiene un tipo de clase asociado. Esas clases pueden heredar atributos de la clase superior o superclase. Por esa razón, este enfoque de programación se utiliza en programas grandes y complejos que se deben de actualizar con cierta regularidad.

Programación funcional

La programación funcional (PF) es un paradigma de programación al igual que la programación orientada a objetos (POO) la PF se basa en calculo lambda y concretamente en composición de funciones puras para modelar las soluciones de software. En cambio, la POO está muy ligada a la programación imperativa y mutable (lista de instrucciones que se van ejecutando) que tiene mucha más relación con el modelo mental de Turing que hemos comentado.

El desarrollo del software va de crear soluciones a problemas pequeños y después componerlos para solucionar un problema mayor. Es por eso que un modelo basado en funciones y en composición de las mismas como única herramienta para crear programas, nos brinda una forma muy elocuente de crear software.

Planteemos por ejemplo el problema de querer incrementar un numero por 1. Podemos enfocar el problema creando una función que resuelva directamente el problema:

Const inc =x => x +1;

Programación lógica

La programación lógica es un tipo de paradigma de programación dentro del paradigma de programación declarativa. El resto de los sub paradigmas de programación entro de la programación declarativa son: programación funcional, programación con restricciones, programas DSL (de dominio específico) e híbridos. La programación funcional se basa en el concepto de función (que no es más que una evaluación de los predicados), de corte más matemático. La programación lógica gira entorno al concepto de predicado, o relación entre elementos.

Programación orientada a objetos

La programación orientada a objetos (POO u OOP) para las siglas en ingles es un paradigma de programación que define los programas en términos de ´´Clases de objetos ´´

La programación orientada a objetos se basa en el concepto de crear un modelo del problema de destino en sus programas .la programación orientada a objetos disminuye los errores y promociona la utilización de códigos. Python es un lenguaje orientado a objetos. Los objetos definidos en Python tienen las características siguientes:

Java

Java sirve para crear aplicaciones y procesos en una gran diversidad de dispositivos. Se basa en una programación orientada a objetos, permite ejecutar un mismo programa en múltiples sistemas operativos y hace posible ejecutar el código en sistemas remotos de manera segura

Su ámbito de aplicaciones es tan amplio que java se utiliza tanto en móviles como en electrodomésticos. Muchos programadores también utilizan este lenguaje para crear pequeñas aplicaciones que se insertan en el código HTML de una pagina para que pueda ser ejecutado desde un navegador

Java es un sistema de programación versátil y con multitud de aplicaciones.

Orientada a objetos

Dispone de un gran número de clases (API)

No existen lenguajes libres a diferencia de C++

Dispone de tipos primitivos como es C++

No posee templetes característica importante de C++

Java incrementa mecanismos de seguridad que limitan el acceso a recursos de la máquina donde se ejecuta la aplicación sobre todo en los applets.

Aplicaciones que se cargan desde un servidos y se ejecutan en el cliente

Java es interpretado

Sus prestaciones no son excesivamente altas, conforme cargan as aplicaciones java genera el código máquina correspondiente.

La ejecución se realiza a velocidad de código compilado

Multitarea

Java permite construir aplicaciones multitareas (múltiples hilos de ejecución)

Dos versiones de java:

(JSE: Java estándar edition)

(J2EE: Java Enterprise edition)

Java 2

Orientado a objetos: herencia polimorfismo

Robusto

Lenguaje fuertemente tipado (detecta errores en tiempo de compilación)

Arquitectura natural

JDBC API

Java RMI

JAVA IDL

Jaba beans

Programación concurrente

La concurrencia, es el contexto de la programación, es la capacidad de descomponer un programa en partes que puedan ejecutarse independientemente unas de otras. En esta línea la programación concurrente es una técnica en la que dos o más procesos o hilos de un software se inician, se ejecutan se forma intercalada mediante el cambio de contexto y se completan en un periodo de tiempo superpuesto gestionando el acceso a recursos compartidos como, por ejemplo, en un solo núcleo de CPU.

Esto no significa que se ejecuten varios procesos en el mismo instante de tiempo, aunque por los resultados puede parecerlo.

Tarea 1 proceso3 -tarea 2 proceso 1-tarea 1 proceso2-tarea 1 proceso1 núcleo CPU

Las distintas tareas de cada proceso o de cada thread (evento o circunstancia) que tiene que ejecutar el procesador las va ejecutando secuencialmente una detrás de otra y de manera intercalada, pero en instantes de tiempo tan pequeños que parece que las tareas se ejecutan simultáneamente.

Generalmente se confunde la programación concurrente con programación en paralelo. En la programación en paralelo las tareas se ejecutan en distintos procesadores al mismo tiempo.

Los programas concurrentes son difíciles de escribir porque la gestión de procesos independientes requiere la coordinación de recursos compartidos.

Con el conocimiento que e adquirido con este tema de estudio ´´lenguaje de programación´´ puedo entender de forma correcta el funcionamiento de los sistemas que tanto usamos a diario teléfonos, computadoras y todo tipo de dispositivos actuales que en su mayoría están regidos por alguno de los sistemas informáticos que tenemos en la actualidad.

Agradecimiento

Una vez más quiero expresar mi agradecimiento a esta gran universidad AIU que me está brindando una magnifica formación.