Lenguajes, Paradigmas y Estándares de Programación

Introducción.

Los lenguajes de programación desempeñan una parte crucial en la programación, haciendo un software y eficiente y sostenible. Aquí dejo una breve descripción de la importancia de cada uno:

1. Lenguajes de programación

    - Los lenguajes de programación son cruciales a la hora de que el programador se comunique con el ordenador para mandarle unas tareas o comandos para indicarle como quiere que realice dichas tareas.
    - La elección correcta del lenguaje depende del trabajo que tengamos que hacer ya que esto afectara a la eficiencia, legibilidad del código, mantenibilidad y escalabilidad del software.
    - Cada lenguaje esta diseñado para cumplir diferente función, por eso es importante seleccionar el más adecuado para la tarea.

2. Paradigmas de programación

    - Los paradigmas ayudan a enfocar o estilizar la guía de la estructura y la organización del código. Algunos ejemplos de esto es la programación imperativa, orientada a objetos, funcional y lógica.
    - La selección del paradigma afecta a la forma de la que se solucionan los problemas, el modularidad del código y la facilidad de mantenimiento.
    - Comprender estos paradigmas es importante para un programador ya que le ayudara a ser mas flexible a la hora de abordar nuevos problemas y tener una solución más rápida y efectiva.

3. Estándares de programación

    - Los estándares son pautas aceptadas y normalizadas por la comunidad de programadores para garantizar una calidad de código y una consistencia
    - Los estándares ayudan a la colaboración entre desarrolladores, mejoran la legibilidad y ayuda a mantener una coherencia en el código.

Tipos de lenguajes de programación

Los lenguajes de bajo nivel (lenguajes ensambladores*)*, permiten al programador escribir instrucciones de un programa usando abreviaturas del inglés como: ADD, DIV, SUB, etc. Un programa escrito en un lenguaje ensamblador no es comprensible para la computadora, ya que, no está compuesto por ceros y unos. Para traducir las instrucciones de un programa escrito en un lenguaje ensamblador a instrucciones de un lenguaje máquina hay que utilizar un programa llamado ensamblador, como se muestra continuación:


```
[0x00000000]> pd
            0x00000000    90            nop
            0x00000001    90            nop
            0x00000002    6800009c00    push 0x9c0000 ; 0x009c0000
            0x00000007    e8c7ace37b    call 0x7be3acd3
               0x7be3acd3(unk)
            0x0000000c    bb04009c00    mov ebx, 0x9c0004
            0x00000011    8903          mov [ebx], eax
            0x00000013    e81903f47b    call 0x7bf40331
               0x7bf40331()
            0x00000018    bb08009c00    mov ebx, 0x9c0008
            0x0000001d    8903          mov [ebx], eax
            0x0000001f    bb00009c00    mov ebx, 0x9c0000
            0x00000024    c60300        mov byte [ebx], 0x0
        .-> 0x00000027    68e8030000    push 0x3e8 ; 0x000003e8
        |   0x0000002c    e81124e37b    call 0x7be32442
        |      0x7be32442(unk)
        `=< 0x00000031    ebf4          jmp 0x100000027
            0x00000033    90            nop
            0x00000034    ff            invalid
            0x00000035    ff            invalid
            0x00000036    ff            invalid
            0x00000037    ff            invalid
```

Algunos de los usos mas comunes del lenguaje de ensamblador es la programación de sistemas operativos, desarrollo de controladores de

dispositivos, programación de tiempo real, optimización de código crítico y programación de interrupciones y excepciones.

Lenguaje de medio nivel es un lenguaje de programación que se encuentran entre los lenguajes de alto nivel y los lenguajes de bajo nivel.

Suelen incluirse en los lenguajes de alto nivel, pero permiten ciertos manejos de bajo nivel. Son buenos a la hora de la creación de sistemas operativos, ya que permiten un manejo más, pero sin perder eficiencia que tienen los lenguajes de bajo nivel.

ejemplo C: Hola Mundo!

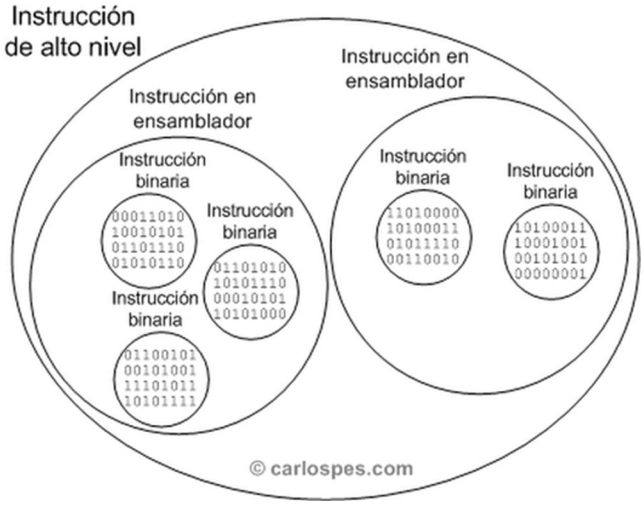
```
#include <stdio.h>

int main()
{
  printf("Hola Mundo!\n");
  return 0;
}
```

Uno de sus mayores usos es la implementación de algoritmos como listas ligadas, tablas hash y algoritmos de búsqueda y ordenamiento que para otros lenguajes de programación.

El lenguaje de alto nivel es mas parecido a un lenguaje humano, haciéndolo más sencillo y alejándose de lo binario y lo complejo haciéndolo un lenguaje más simple que el resto.

Su principal objetivo es usar un mismo lenguaje para todas las computadoras usando el mismo lenguaje implementando el uso de un traductor o un compilador.

El uso de palabras más parecidas a lo humano hace que el desarrollador se centre más en el código y menos en el lenguaje haciendo más efectivo el código y a la hora de colaborar es más sencillo ya q es más fácil de comprender.




```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print '    %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for n, child in enumerate(ast[1:]):
         children.append(dotwrite(child))
      print '    %s -> {' % nodename,
      for name in children:
         print '%s' % name,
```

Los usos más comunes del lenguaje de alto nivel son para desarrollo de aplicaciones software, programación orientada a objetos, desarrollo web, IA, machine learning, automatización de tareas y muchas más.

Paradigmas de programación.

Un paradigma de programación es un enfoque o estilo básico de desarrollo de software. Define cómo se estructuran y organizan las tareas para construir un programa. Es una forma de abordar y resolver problemas a través de la programación. Cada paradigma tiene su propia forma de presentar la información y las acciones que se pueden realizar con ella. A continuación hare un resumen básico de los paradigmas de programación comunes:

1.Programación obligatoria (o procedimental):

Se basa en dar instrucciones al ordenador sobre cómo realizar una tarea paso a paso. Se centra en describir como hacer las cosas. Algunos ejemplos son: C, Pascal, Fortran. Unas características únicas de este paradigma es la modificación de estado, las estructuras de control, los procedimientos/Funciones y el enfoque en el paso a paso.

2.Programación Orientada a Objetos (POO):

Organiza el software en una colección de objetos que contienen tanto datos como comportamiento. Introduce conceptos como herencia, encapsulación y polimorfismo. Algunos ejemplos son: Java, C++, Python, Ruby. Unas características únicas de este paradigma es la organización del código en clase y objetos, la encapsulación, herencia, polimorfismo y un modelado del mundo real mediante abstracción de entidades y sus interacciones.

3.Programación funcional:

Se basa en la evaluación de funciones matemáticas y enfatiza la inmutabilidad (evitación de un estado cambiante). Los programas se tratan como evaluaciones de actividades combinadas. Algunos ejemplos son: Haskell, Lisp, Erlang, Scala. Unas características únicas de este paradigma es el centrado de funciones matemáticas y evitar el cambio de estado, el movimiento de funciones y la inmutabilidad de los datos evitando así cambios en ellos.

4.Programación lógica:

Los programas son vistos como un conjunto de declaraciones y reglas lógicas. Se utiliza principalmente para inferencia y consulta de datos. Algunos ejemplos son: Prólogo. Unas características únicas de este paradigma son sus reglas lógicas y relaciones y su interfaz lógica para reducir resultados.

5.Programación Declarativa:

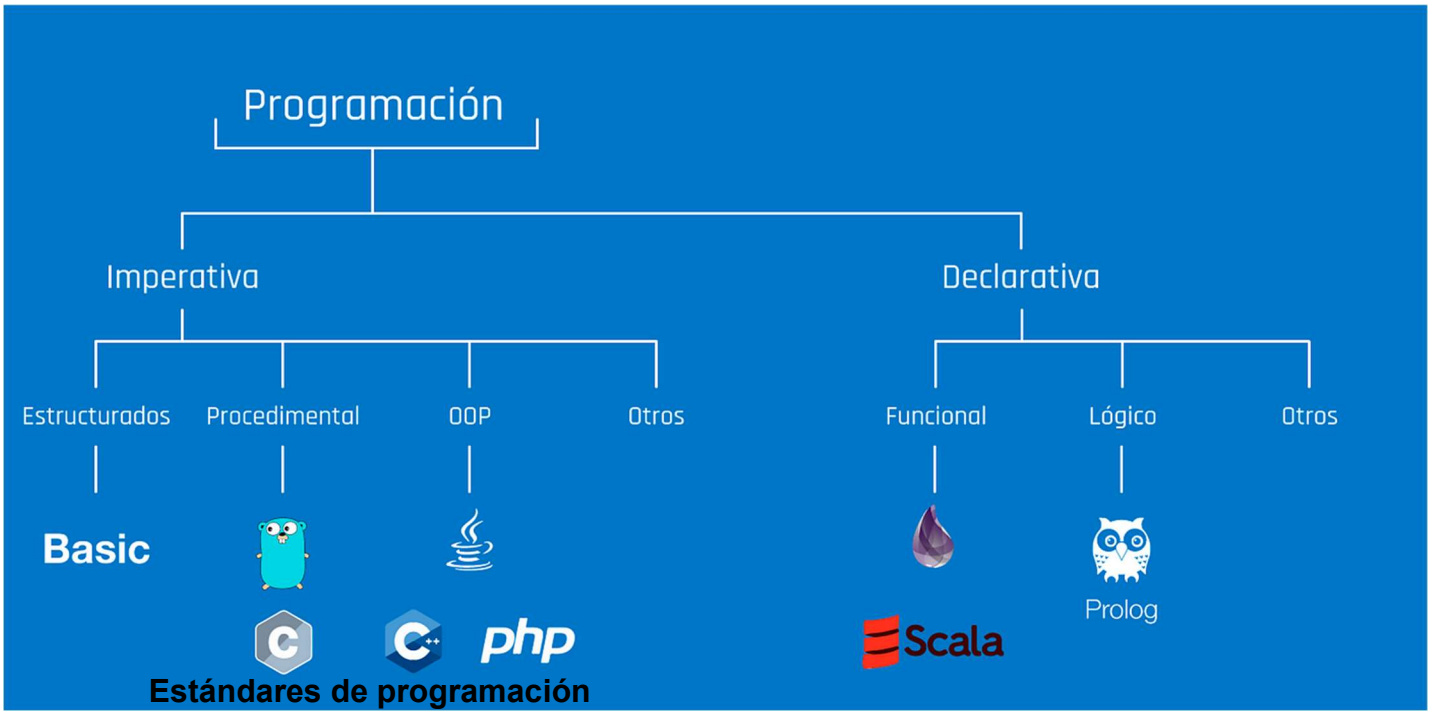
En lugar de describir "cómo" lograr algo (como en la programación imperativa), describe "qué" desea lograr. SQL (utilizado para consultas de bases de datos) es un ejemplo de lenguaje declarativo. Unas características únicas de este paradigma se centra en conseguir los resultados no en como obtenerlos y menos énfasis en el control de flujo explicito.

6.Programación impulsada por eventos (o basada en eventos):

Se centra en responder a eventos como entradas del usuario o señales del sistema. Amplio en el desarrollo de interfaces gráficas de usuario y aplicaciones web. Algunos ejemplos son: JavaScript. Unas características únicas de este paradigma es el uso de eventos como puntos de control, un manejo asincrónico, implementación de listeners, desacoplamiento, una interfaz de usuario interactiva y una escucha continua.

7.Programación paralela y concurrente:

Diseñado para la ejecución simultánea de tareas. Se centra en la creación de software que pueda realizar múltiples funciones simultáneamente. Algunos ejemplos son: Ir, Erlang. Unas características únicas de este paradigma esta diseñado para sistemas concurrentes, y distributivos, usa procesos ligeros para manejar la concurrencia y esta enfocado a la comunicación entre entidades y concurrentes.



Estándares de programación

Los estándares de programación son pautas usadas en la comunidad de desarrollo software que se acuerda seguir para escribir un código consistente y mantener la calidad y la legibilidad del software. Estos estándares ayudan a mejorar la legibilidad, la mantenibilidad y la colaboración entre programadores. Hay muchos estándares normalizados, pero estos son algunos de los principales y los más usados:

1. Estilo de codificación:

   Define la apariencia del código incluyendo el espaciado, el uso de mayúsculas y minúsculas, y la alineación. Algunos ejemplos en diferentes lenguajes de programación son PEP 8 en Python y Google java style para Java.

2. Nombrado de variables y Funciones:

   Define valores predeterminados para nombras diferentes variables haciendo más fácil la interpretación del código y facilitar la cooperación entre desarrolladores. Algunos nombres más comunes son camelCase o camel_case para nombres de variables y PascalCase para nombres de clase en diferentes lenguajes.

3. Comentarios y documentación:

   Se establecen reglas sobre la documentación del código incluyendo el formato de los comentarios y la frecuencia. Algunos ejemplos son Javadoc y docstrings para java y Python.

4. Manejo de errores y excepciones:

   Se establecen acciones y practicas comunes a la hora de manejar errores y excepciones de manera coherente, incluye como y cuando usar declaraciones como try-catch o bloques try-except.

5. Manejo de versiones:

   Define como se tienen q gestionar las versiones del software utiliza sistemas de control como Git y establece practicas para escribir mensajes claros y descriptivos.
6. Seguridad:

   Establece unas normas a la hora de escribir código seguro y resistente a ataques, Hace el uso de un manejo seguro para los datos sensibles, la

validación de entrada y la prevención a ataques teniendo en cuenta las vulnerabilidades conocidas.

7. Uso de bibliotecas y frameworks:

   Se usan unas normas para controlar el uso de bibliotecas externas y frameworks dentro de un proyecto. Estas normas abarcan acciones como importar, configurar y extender funcionalidades de bibliotecas externas.

8. Estructura de directorios y organización de proyectos.

   Establece una estructura predeterminada para organizar un proyecto y sea más fácil la organización de archivos entre colaboradores. Esta norma incluye cosas como donde deben ubicarse los archivos fuente, las pruebas, la documentación, ect.

9. Formato de archivos y codificación de caracteres:

   Define un formato para los archivos de un proyecto y la codificación de caracteres (UTF-8) para hacer de su uso en el proyecto.

10. Pruebas y cobertura de código:

    Establece unas acciones para escribir pruebas unitarias de integración y funcionales, estos incluyen estándares para medir y mejorar la cobertura del código.

11. Estándares de diseño:

    Define unas reglas y practicas para el diseño del código, como el (SRP) principio de responsabilidad única y el principio de inversión de dependencias (DIP).

Depende del lenguaje usado estos estándares pueden ser diferentes y sus usos variar, si eres un programador puedes no adherirte a estos estándares que no va a pasar nada, pero a la hora de programar en un proyecto en conjunto estos estándares van a ayudar a entender el código entre desarrolladores facilitando el trabajo y haciendo un código mas efectivo y más fácil de mantener. Pero en cambio si decides no seguir con estos estándares trabajando en conjunto, tus compañeros de proyecto pueden tener problemas a la hora de entender tu código, invirtiendo tiempo en comprenderlo y podría generar mas problemas a la hora de manejar código.

Conclusión

Hasta lo más básico puede afectar a cómo funciona tu código o incluso al mantenimiento de este en el tiempo, saber los lenguajes de programación a los que puedes optar es muy importante ya que dependiendo del trabajo que se quiere realizar existen algunos lenguajes que pueden realizar lo que solicitas de una manera mas efectiva que otros, así que la elección del lenguaje q se va a usar para el proyecto es importante para el optimo funcionamiento. El uso de paradigmas es importante a la hora de trabajar en el código ya que se siguen unas pautas las cuales ayudan a lidiar con el problema con efectividad y solucionar el problema de la mejor forma y lo más rápido posible. Los estándares de programación son muy importantes a la hora de trabajar en conjunto sobre un código, ya que gracias a estos es más fácil entender el código de otros compañeros y si facilitar el trabajo sobre este y su correcto mantenimiento.

Referencias:

Chat gpt

https://speckle-porpoise-32d.notion.site/Introducci-n-a-la-programaci-n-906cf521f30f41d9893cd2098b41cd87

https://conceptobasicodecomputacion.weebly.com/lenguaje-de-alto-medio-y-bajo-nivel.html