

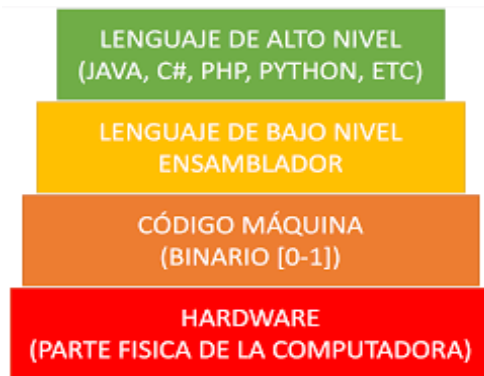
Informe de Lenguajes, Paradigmas y Estándares de la programación

Los lenguajes de programación son un conjunto estructurado de instrucciones que son utilizadas para que una computadora realice diversas tareas, desde un algoritmo complejo, hasta una simple calculadora. Los desarrolladores utilizan estos lenguajes de programación para escribir el código de sus programas, que o bien, después son convertidos a lenguaje máquina, o a código interpretable por la máquina. Los lenguajes pueden variar en sintaxis y en nivel de abstracción, pueden ser de alto nivel, más cercanos al humano o de bajo nivel, más cercanos al lenguaje máquina.

Dentro de los lenguajes de programación encontramos los paradigmas, que son algo así como la filosofía que existe en cuanto a conceptos y prácticas para el desarrollo de un programa. Sirven para organizar y estructurar el código. Existen distintos tipos de paradigmas, como el imperativo, el funcional o el orientado a objetos.

Por último, tenemos los estándares que son aquellos acuerdos que sirven para uniformizar la forma en que se organiza y se escribe el código. El objetivo es que este sea legible y mantenible no solo para el autor, sino para cualquier desarrollador, por lo que son muy importantes en entornos colaborativos.

Existen diferentes tipos lenguajes de programación, están los lenguajes de alto nivel de abstracción que son aquellos más cercanos al lenguaje humano y los de bajo nivel, que son de menor abstracción y más similares al lenguaje máquina. Los lenguajes de programación son el puente entre el lenguaje humano y el lenguaje máquina.



Como observamos en la imagen tenemos el código maquina en lo más bajo de la pirámide, siendo este el lenguaje de más bajo nivel, un ejemplo es el lenguaje binario que usan los ordenadores.

El lenguaje binario es el sistema de codificación que utilizan ordenadores y otros dispositivos electrónicos para almacenar, procesar y transmitir datos. Este lenguaje se constituye enteramente de “0” y “1” que se conocen como bits, estos bits representan dos estados, encendido que es igual a 1 y apagado que es igual a 0.

El lenguaje binario lo usan los ordenadores o dispositivos electrónicos para, almacenar datos, dar instrucciones al hardware de la máquina, transmitir datos, entre otros usos.


Justo despues nos encontramos el lenguaje ensamblador, que es también de muy bajo nivel, pero es entendido por humanos y muy cercano al lenguaje de la máquina. A pesar de que la mayoría de la programación moderna se realiza en lenguajes de alto nivel, el ensamblador todavía se utiliza en el desarrollo de software crítico en tiempo y recursos, en la escritura de rutinas de bajo nivel (como controladores de dispositivos), y en la educación para enseñar cómo funciona una computadora a nivel de hardware.

Luego estarían los lenguajes de bajo nivel, un ejemplo de ellos es C, cuyo uso más común es para el desarrollo de sistemas operativos, controlador de dispositivos, programación de redes, creación de compiladores, videojuegos, etc.

Por ultimo los lenguajes de medio y alto nivel, como son Java, C++, JavaScript y Python, entre otros


muchos. Los lenguajes de medio y alto nivel son los más populares en la actualidad, podemos comentar algunos de los usos específicos que tiene cada uno, como Java que se utiliza mucho para desarrollo de apps empresariales y en Android, JavaScript para el desarrollo front-end y full stack o Python para la ciencia de datos y análisis, y el desarrollo rápido de aplicaciones. Comparten muchos usos entre todos estos lenguajes, como en el desarrollo web, aunque en diferentes capacidades, JavaScript es esencial en el front-end, y Java, C++ y Python son mas usados en el back-end, todos son comunes en la automatización de tareas, así como en el desarrollo de software, aunque el tipo y la complejidad de las aplicaciones puede variar, y por supuesto todos son utilizados para propósitos educativos.

python

 Copy code

```
print("Hola Mundo")
```


c

 Copy code

```
#include <stdio.h>

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

asm

 Copy code

```
section .data
msg db 'Hola Mundo', 0

section .text
global _start

_start:
    mov eax, 4          ; la llamada al sistema para write (sys_write)
    mov ebx, 1          ; el descriptor de archivo 1 es stdout
    mov ecx, msg        ; el mensaje a escribir
    mov edx, 10         ; escribir los primeros 10 bytes de msg
    int 0x80            ; llamar al kernel

    mov eax, 1          ; la llamada al sistema para exit (sys_exit)
    xor ebx, ebx        ; terminar con el estado 0
    int 0x80            ; llamar al kernel
```

En estas imágenes se muestran las diferencias entre un lenguaje de alto nivel como es Python (1ª imagen) y uno de muy bajo nivel como es el lenguaje ensamblador (3ª imagen) para realizar una simple tarea como imprimir en consola “Hola Mundo”.

Como ya mencionamos en la introducción los paradigmas es algo así como la filosofía que sirve para organizar y estructurar los códigos. Ofrecen una perspectiva única para la solución de problemas, y en base a este, a las preferencias del desarrollador y el contexto en el que trabaja, se hará la elección del paradigma y del lenguaje de programación. Algunos de los principales paradigmas son Imperativo, Declarativo, Orientado a objetos (OO), Funcional y Lógico.

- Imperativo.

Este paradigma se centra en describir como se realiza una tarea mediante distintas secuencias de instrucciones para la máquina. Este paradigma es uno de los más antiguos que existen. Utiliza las declaraciones de variables, los bucles o los condicionales para controlar el flujo que se ejecuta en el código. Los lenguajes más representativos del paradigma imperativo son C, Fortran, Pascal y BASIC. Es de uso común en aplicaciones de sistemas, juegos y programas que requieren de un control del hardware.

- Declarativo.

Este paradigma no se enfoca en el control de flujo del código como en el imperativo, sino que se centra en que tareas se deben realizar sin especificar cómo deben de hacerse, es decir, se describe el resultado deseado, no como alcanzarlo, permitiendo que el compilador o interprete sea el que determine la forma óptima de ejecutarlo. Ayuda a desarrollar un código conciso y fácil de entender. Los lenguajes más representativos del paradigma declarativo son SQL, HTML y Prolog, es muy utilizado en bases de datos,

desarrollo web y en áreas donde el resultado es más importante como se alcanzó.

- Orientado a Objetos (OO).

Para empezar a describir este paradigma debemos definir primero los “objetos”, son entidades que contienen datos en forma de campos, conocidos como atributos, y código en forma de procedimientos, conocidos como métodos. Esto da lugar a que los programas se organicen en “objetos” que representan una combinación de datos y comportamientos. Facilita mucho la reutilización de código y es útil para modelar sistemas complejos del mundo real. Algunos de los lenguajes más representativos del paradigma orientado a objetos son Java, Python, Ruby, entre otros. Es utilizado en software empresarial, sistemas de gestión, aplicaciones de escritorio y desarrollo de juegos.

- Funcional.

El paradigma funcional se centra en las funciones puras, llamadas así ya que son funciones que siempre producen el mismo resultado con los mismos argumentos sin efectos secundarios, como la modificación de variables globales. Los lenguajes más representativos del paradigma funcional son Scala y Clojure, y es común su uso en aplicaciones concurrentes, sistemas distribuidos y como parte del manejo de eventos en interfaces de usuario.

- Lógico.

El paradigma lógico se basa en la lógica formal, el programa se escribe como un conjunto de sentencias en forma lógica. Es declarativo y permite expresar hechos y reglas dentro de un programa sin especificar un algoritmo. Los lenguajes más representativos de este paradigma son Prolog y Datalog, y es usado en sistemas de inteligencia artificial, sistemas expertos y en el procesamiento del lenguaje natural.

Tras hablar sobre los paradigmas de la programación, los usos más comunes y los lenguajes más

representativos de cada paradigma, nos queda hablar sobre los estándares de la programación.

Los estándares de la programación no son más que un conjunto de reglas que sirven para la construcción consistente y ordenada del código. Pueden ser por ejemplo estándares de la programación un acuerdo en la nomenclatura de ciertas variables, estilos de codificación, técnicas de documentación, lo que se conoce con el término “buena práctica”, son ampliamente aceptados y seguidos por los desarrolladores de software. Es importante seguir los estándares a la hora de empezar a programar, ya que puede tener un gran impacto en la calidad, mantenibilidad y escalabilidad del código.

El seguimiento de los estándares en el código ayuda a mejorar la legibilidad de este, lo cual es un asunto importante cuando se trabaja en grandes proyectos con múltiples desarrolladores, aportan una gran consistencia en el código, facilitando así la detección de errores y que el código sea predecible y fácil de seguir. Se puede conseguir una mayor eficiencia del código en términos de recursos utilizados y velocidad, gracias a esto el código también tendrá una mantenibilidad mejorada, ya que un código bien estructurado y que sigue estándares es fácil de mantener y actualizar. El seguimiento de estándares añade portabilidad al programa, ya que permite su uso en diferentes plataformas y sistemas operativos. Como mencionamos al principio, los estándares de la programación aportan una gran facilidad para la colaboración entre desarrolladores, es por eso que son establecidos por organizaciones, comunidades de código abierto o incluso dentro de una misma empresa. Por último, representan una mejora de calidad en el código, previniendo errores comunes, gracias a muchas de las reglas y directrices, mejorando a su vez la seguridad del código al evitar prácticas que son conocidas por tener vulnerabilidades de seguridad.

Algunos ejemplos incluyen la Guía de Estilo de Python, conocida como PEP8, las Directrices de Estilo de Código de Google, y los estándares de Codificación del Framework Symfony para PHP.

- PEP 8 – Style Guide for Python Code:

Algunas de las características principales son, la utilización de 4 espacios por nivel de indentación, limita cada línea a un máximo de 79 caracteres, reglas sobre cómo y dónde aplicar espacios en blanco para mejorar la legibilidad. Sobre las importaciones, deben estar en líneas separadas y en un orden específico: bibliotecas estándar, terceros relacionados y luego importaciones locales. La nomenclatura “CamelCase” para nombres y clases, no utilizando espacios y empezando la siguiente palabra en mayúsculas, y “snake_case”, para funciones y variables, utilizando espacios sin el guion bajo.

- Google’s Style Guides:

Google aporta guías de estilo para una gran variedad de lenguajes, como C++, Python, Javascript, AngularJS, entre algunos otros. Se enfoca en la claridad y consistencia del código dentro de cada lenguaje. Hace mención a prácticas determinadas como el uso de promesas en JavaScript y la preferencia por funciones sin estado en C++. Además, incluye consejos sobre la revisión de código, lo que ayuda a mantener la calidad y la cohesión del código en un ambiente donde trabaja más de un desarrollador.

- Symfony Coding Standards:

Este estándar está basado en PSR, en consonancia con las recomendaciones de PHP-FIG, especialmente PSR-1 (Estándares Básicos de Codificación) y PSR-2 (Guía de Estilo de Codificación). Define también la nomenclatura tanto para interfaces, nombres, clases, métodos, variables, y más. Una de sus prácticas de codificación es la inyección de dependencias en lugar de la herencia de

clases. Y por último, proporciona una estructura estándar para la realización de testing.

El seguimiento de estos estándares tiene como objetivo hacer que el código sea fácilmente entendible y que mantenga una alta calidad. La adopción de estos estándares también facilita la colaboración entre diferentes equipos, ya que, al ser un código escrito en base a un conjunto de reglas comunes, será mucho más predecible y fácil de fusionar y revisar.

Como conclusión, cabe destacar la importancia de entender los distintos lenguajes de programación, paradigmas y estándares para el desarrollo del software porque cada uno de ellos ofrecen herramientas únicas que permiten abordar una gran cantidad de problemas de la manera más eficiente posible. Conocer los diferentes lenguajes de programación, sus puntos fuertes y débiles, para saber en cada momento a cuál recurrir según el contexto. De igual forma con los paradigmas y los estándares, que nos ofrecen marcos de referencia que nos ayudan a estructurar el código, y en el caso de los estándares, que actúan como un idioma común, permitiendo una colaboración efectiva entre desarrolladores, permitiendo también que el código sea legible, mantenible y seguro a lo largo del tiempo. En resumen, el entendimiento y la aplicación efectiva de los diversos lenguajes, paradigmas y estándares de la programación son esenciales para el desarrollo de software, facilitando no solo esto, sino también la colaboración y la innovación continua, que son de vital importancia para un campo que evoluciona tan deprisa como es la tecnología de la información.

Referencias:

- ChatGPT 4. url: <https://chat.openai.com>
- Wikipedia – List of programming languages by type. url: https://en.wikipedia.org/wiki/List_of_programming_languages_by_type
- CodeDocs url: <https://codedocs.org>
- The Coderpedia url: <https://www.thecoderpedia.com>

- LambdaTest – Coding Standards and Guidelines url:
<https://www.lambdatest.com/learning-hub/coding-standards>