Introducción a la programación de computadores

La programación tiene sus inicios en la década de 1930, cuando se desarrollaron los primeros dispositivos electrónicos. En ese momento, los programas se escribían directamente en lenguaje de máquina, el cual es el idioma que las máquinas usan para ejecutar instrucciones. Esto hacía que los programas fueran difíciles de escribir y modificar, y se necesitaba un conocimiento profundo de la estructura de las máquinas.

En la década de 1950, se desarrollaron los primeros lenguajes de programación de alto nivel, como FORTRAN y LISP. Estos lenguajes permitieron a los programadores escribir código en un lenguaje más cercano al lenguaje natural, lo que hizo que fuera más fácil escribir y modificar programas.

En la década de 1960, surgieron nuevos lenguajes de programación, como BASIC y C, que se convirtieron en populares para la enseñanza de la programación en las universidades. En la década de 1980, surgieron lenguajes de programación orientados a objetos, como C++ y Smalltalk, que permitieron a los programadores crear programas más complejos y reutilizables.

En la actualidad, hay cientos de lenguajes de programación disponibles, cada uno con sus propias características y usos específicos. La programación es una habilidad esencial en la era digital actual, y se utiliza en una amplia variedad de campos, desde la ciencia de la computación y el desarrollo de aplicaciones hasta la finanzas y la ingeniería.

Para aprender los conceptos básicos de la programación, recomendamos seguir los siguientes pasos:

- *Elegir un lenguaje de programación:* hay muchos lenguajes disponibles, cada uno con sus propias características y usos específicos. Algunos lenguajes populares incluyen C, Python, Java, C++ y JavaScript. Es importante elegir un lenguaje de programación que se ajuste a tus necesidades y objetivos.
- *Aprender los conceptos básicos*: una vez que hayas elegido un lenguaje, es importante tener una comprensión sólida de conceptos básicos como variables, operadores, estructuras de control de flujo y funciones. También es útil entender cómo se almacenan y manipulan los datos en una computadora.
- *Practicar:* la práctica es esencial para dominar cualquier habilidad, incluyendo la programación. Es importante dedicar tiempo a escribir código y solucionar problemas utilizando el lenguaje de programación que has elegido.
- *Aprender a depurar:* es inevitable que encuentres errores o "bugs" en tu código mientras lo estés escribiendo. Es importante aprender a depurar tu código, es decir, a encontrar y solucionar estos errores.
- *Continuar aprendiendo:* una vez que hayas dominado los conceptos básicos de la programación, puedes comenzar a explorar temas más avanzados y especializados. Esto incluye la programación orientada a objetos, la creación de aplicaciones web y la integración de bases de datos.

Lenguajes de alto nivel y de bajo nivel

Los lenguajes de programación se pueden clasificar en dos categorías principales: lenguajes de alto nivel y lenguajes de bajo nivel.

Los lenguajes de alto nivel son lenguajes de programación que se parecen más al lenguaje natural y son más fáciles de leer y escribir. Se utilizan para escribir programas que realizan tareas complejas y abstractas. Algunos ejemplos de lenguajes de alto nivel incluyen Python, Java, C++ y JavaScript.

Por otro lado, los lenguajes de bajo nivel son lenguajes de programación que se parecen más a la forma en que funciona una computadora. Se utilizan para escribir programas que realizan tareas más básicas y específicas a nivel de hardware. Algunos ejemplos de lenguajes de bajo nivel incluyen lenguaje de máquina y ensamblador.

Es importante tener en cuenta que todos los lenguajes son "traducidos" en lenguaje de máquina por la computadora para que puedan ser ejecutados. Sin embargo, en los lenguajes de alto nivel existe una

capa de abstracción adicional, lo que significa que son más fáciles de leer y escribir para los seres humanos, pero también pueden ser más lentos al ejecutarse en comparación con los de bajo nivel.

Un ejemplo de lenguaje de alto nivel es Python. Python es un lenguaje de programación interpretado y orientado a objetos que se ha vuelto muy popular en los últimos años debido a su sintaxis sencilla y su amplia gama de aplicaciones.

A continuación, te proporcionamos un ejemplo de código Python que define una función llamada sumar que toma dos argumentos a y b, y devuelve la suma de ambos. Luego, se llama a la función pasándole los valores 4 y 8, y se almacena el resultado en una variable llamada resultado. Finalmente, se imprime el resultado en la pantalla utilizando la función print:

```
def sumar(a, b):
    return a + b

resultado = sumar(4, 8)
print(resultado)
```

Como puedes ver, la sintaxis de Python es muy legible y se asemeja al lenguaje natural. Esto hace que sea fácil de aprender y utilizar para los programadores principiantes y experimentados. Además, Python tiene una gran cantidad de librerías y herramientas disponibles para realizar tareas específicas, lo que lo hace ideal para una amplia gama de proyectos de programación.

Assembler Code:

```
suma(int, int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     DWORD PTR [rbp-8], esi
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        pop     rbp
        ret
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     edx, DWORD PTR [rbp-8]
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, edx
        mov     edi, eax
        call    suma(int, int)
        mov     DWORD PTR [rbp-12], eax
        mov     eax, 0
        leave
        ret
```

Cada línea del código representa una instrucción que se ejecutará por la computadora. Las instrucciones son escritas en un lenguaje específico que es entendido por la computadora, y cada instrucción corresponde a una operación básica que puede ser realizada por la misma.

Es importante tener en cuenta que este código es solo un ejemplo y puede variar dependiendo del procesador y la plataforma que se esté utilizando. Además, es posible que sea necesario utilizar otras instrucciones o utilizar el código de manera diferente para lograr el mismo resultado en otras plataformas.

¿Puede C ser considerado un lenguaje de nivel medio?

Sí, el lenguaje de programación C se considera un lenguaje de nivel medio o intermedio. Esto significa que se encuentra entre los lenguajes de bajo nivel, como el lenguaje ensamblador, y los lenguajes de alto nivel, como Python o Java.

Como ya se mencionó, los lenguajes de bajo nivel están más cerca de la máquina y son más eficientes en términos de rendimiento, pero suelen ser más difíciles de usar y requieren un conocimiento más profundo de la arquitectura del hardware y el sistema operativo. Por otro lado, los lenguajes de alto nivel proporcionan una mayor abstracción y son más fáciles de usar, pero pueden tener un rendimiento inferior.

C es un lenguaje de programación muy versátil y ha sido utilizado durante muchos años para crear aplicaciones de sistema y aplicaciones de alto rendimiento en general. Aunque es más fácil de usar que los lenguajes de bajo nivel, todavía requiere un conocimiento más profundo de la arquitectura del hardware y del sistema operativo en comparación con los lenguajes de alto nivel.

Compiladores, intérpretes

Un compilador es un programa informático que convierte el código fuente de un lenguaje de programación en código ejecutable para una computadora. El código fuente es el código escrito por el programador en un lenguaje de programación determinado, mientras que el código ejecutable es el código que puede ser ejecutado directamente por la computadora.

El proceso de compilación consiste en leer el código fuente y generar un archivo ejecutable que contenga el código máquina necesario para ejecutar el programa. El compilador verifica la sintaxis y otros errores en el código fuente y muestra un mensaje de error si encuentra algún problema. Una vez que el código ha sido compilado correctamente, el archivo ejecutable puede ser ejecutado en la computadora.

Un intérprete, por otro lado, es un programa que ejecuta directamente el código fuente sin necesidad de compilarlo previamente. El intérprete lee el código fuente línea por línea y lo ejecuta en tiempo real. Si encuentra un error, muestra un mensaje de error y detiene la ejecución del programa. Los lenguajes de programación interpretados suelen ser más fáciles de usar que los lenguajes compilados, pero tienen un rendimiento inferior.

Aquí hay algunos ejemplos de lenguajes de programación compilados:

```
let lenguajes = ["C", "C++", "C#", "FORTRAN", "Pascal", "Swift"]
```

Aquí hay algunos ejemplos de lenguajes de programación interpretados:

```
lenguajes = ['Python','Ruby','PHP','JavaScript','Bash','Perl']
```

Es importante tener en cuenta que algunos lenguajes de programación pueden ser tanto compilados como interpretados, dependiendo de la implementación que se use. Por ejemplo, Java puede ser compilado a bytecode, que luego puede ser ejecutado por un intérprete de Java.

Lenguajes de programación, generalidades

Podemos clasificar los lenguajes de programación según su paradigma:

- Lenguajes imperativos: se basan en la ejecución de órdenes y en la modificación del estado de la máquina. Ejemplos: C, Pascal.
- Lenguajes declarativos: se basan en la declaración de hechos y en la solución de problemas a partir de ellos. Ejemplos: Prolog, SQL.
- Lenguajes orientados a objetos: se basan en el encapsulamiento de datos y comportamientos en unidades llamadas objetos. Ejemplos: Java, C++.
- Lenguajes funcionales: se basan en la aplicación de funciones matemáticas para resolver problemas. Ejemplos: Haskell, Lisp.
- Lenguajes de script: son utilizados para automatizar tareas y son interpretados en lugar de compilados. Ejemplos: Bash, Python, Ruby.
- Lenguajes de programación visual: utilizan diagramas y bloques de construcción en lugar de código escrito para crear programas. Ejemplos: Scratch, Blockly.

También podemos clasificarlos según su uso:

- Lenguajes de sistemas: son utilizados para el desarrollo de sistemas operativos, controladores de dispositivos y otras aplicaciones de bajo nivel. Ejemplos: C, C++.
- Lenguajes de aplicación: son utilizados para el desarrollo de aplicaciones de negocio, juegos, herramientas de desarrollo y otros tipos de software de alto nivel. Ejemplos: Java, Python.
- Lenguajes web: son utilizados para el desarrollo de sitios y aplicaciones web. Ejemplos: Ruby, JavaScript, PHP.

Es importante mencionar que estas clasificaciones no son mutuamente excluyentes y que muchos lenguajes de programación pueden encontrarse en varias categorías a la vez. Por ejemplo, Python es un lenguaje de alto nivel, orientado a objetos y de script.

Desafío 1: Suma de Dos Números (tema1-1des1)

Objetivo: Crear un programa que sume dos números predefinidos y muestre el resultado.

Descripción:

Define dos variables con números específicos asignados por ti.

Suma esos dos números y almacena el resultado en una tercera variable.

Muestra el resultado de la suma.

Desafío 2: Calcular el Área de un Rectángulo con Presentación (tema1-1des2)

Objetivo: Escribir un programa que, utilizando valores predefinidos, calcule el área de un rectángulo y presente los resultados de manera amigable utilizando texto.

Descripción:

Define dos variables numéricas que representen el ancho y el largo de un rectángulo.

Define variables de tipo string que contengan texto explicativo sobre lo que hace el programa.

Calcula el área del rectángulo (ancho x largo).

Utiliza tanto variables numéricas como de texto para presentar el resultado de una manera que sea fácil de entender para alguien que no está viendo el código.