



Taller Conceptos

Juan Camilo Bernal Gordillo
Estructura de datos g1



Introducción a las computadoras y lenguajes de programación

💻 Computadoras y su funcionamiento:

Un computador es una máquina digital programable, de funcionamiento electrónico, capaz de procesar grandes cantidades de datos a altas velocidades..Entre las principales funciones de este tipo de aparatos están:almacenar y recuperar información,facilitar el trabajo especializado,permitir las telecomunicaciones y entretenir al usuario.



Introducción



- Evolución de los lenguajes de programación

La evolución de los lenguajes de programación comenzó con lenguajes de bajo nivel como el código máquina y ensamblador. A lo largo de la historia, hitos como el ENIAC y la arquitectura de von Neumann marcaron el avance de la computación. La aparición de lenguajes de alto nivel como Fortran, LISP y C facilitó la programación, mientras que la programación orientada a objetos, con lenguajes como Smalltalk, introdujo nuevos paradigmas. Hoy, lenguajes como Python, JavaScript y Java dominan el desarrollo web y de software, impulsando la innovación y adaptándose a nuevas tecnologías.

- Importancia de la programación en la actualidad

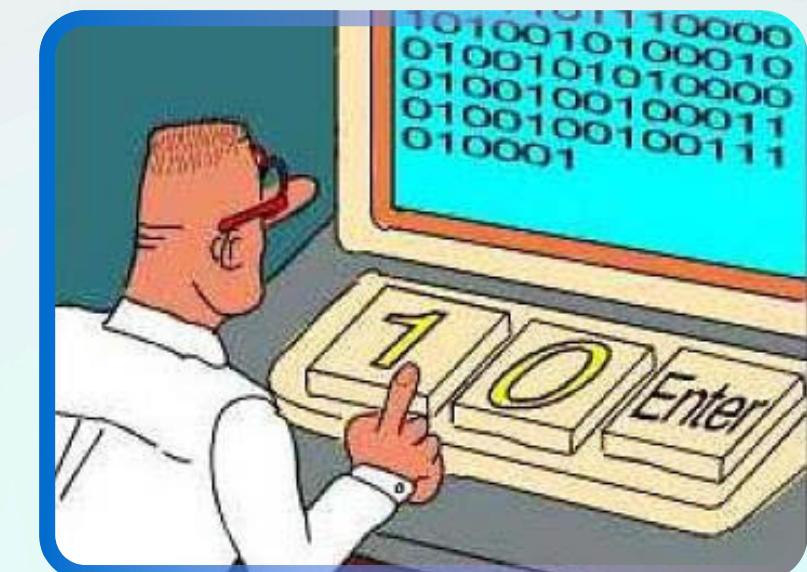
La programación es crucial en el mundo moderno, impulsando la innovación tecnológica, mejorando la educación, optimizando la atención médica y promoviendo la sostenibilidad ambiental. Facilita la automatización de procesos, mejora la eficiencia, empodera a las personas para resolver problemas complejos y abre oportunidades laborales en diversos sectores, debido a la creciente demanda de habilidades tecnológicas.



Tipos de lenguajes de programación

⚙️ Lenguaje de máquina:

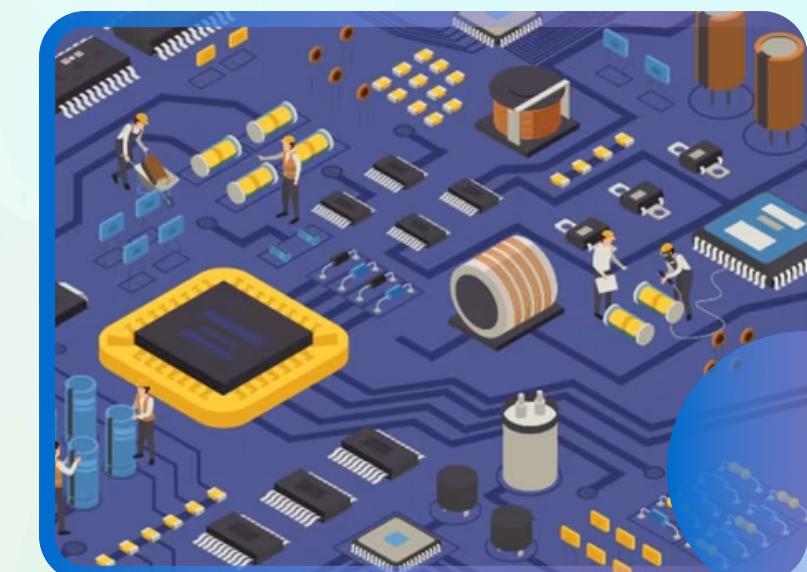
El lenguaje máquina es el conjunto de instrucciones binarias (ceros y unos) que el procesador interpreta y ejecuta directamente. Es el nivel más bajo de programación, donde las instrucciones se ejecutan sin necesidad de traducción adicional.

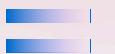


🔧 Lenguaje ensamblador:

El lenguaje ensamblador es un lenguaje de bajo nivel que se traduce a código máquina mediante un ensamblador. Cada instrucción en ensamblador corresponde a una operación específica del procesador, y permite un control detallado sobre el sistema. Aunque su curva de aprendizaje es alta, es valioso para trabajos que requieren interacción directa con el hardware.

```
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 ; 0x0000003e8
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
```





- **Lenguajes de alto nivel:**

Los lenguajes de programación de alto nivel son diseñados para simplificar la programación, ocultando detalles complejos del hardware. Estos lenguajes se traducen a código máquina mediante un compilador o intérprete.. Los principales lenguajes de programación de alto nivel incluyen:

- Java
- C++
- Python
- JavaScript
- Ruby
- PHP
- C#
- Swift
- Go
- Kotlin



Herramientas necesarias para programar



Editor de texto vs IDEE

Editor de texto

Un editor de texto es una herramienta simple que permite escribir y editar código. Es ligero, rápido y generalmente se enfoca solo en el texto, sin funcionalidades adicionales. Ejemplos populares incluyen Notepad++, Sublime Text y Visual Studio Code (en su forma básica).

Ventajas del editor de texto:

- Ligereza: No requiere muchos recursos del sistema.
- Velocidad: Inicia rápidamente y permite escribir código sin interrupciones.
- Simplicidad: Ideal para proyectos pequeños o rápidos sin necesidad de muchas herramientas adicionales.
- Personalización: Generalmente es muy configurable y soporta una variedad de complementos o extensiones.

IDE (Entorno de Desarrollo Integrado)

Un IDE es una herramienta más compleja que combina un editor de texto con funcionalidades adicionales como depuración, autocompletado, control de versiones, compilación automática, entre otros. Ejemplos comunes de IDE son IntelliJ IDEA, Eclipse, y Visual Studio.

Ventajas del IDE:

- Características completas: Incluye herramientas como depuradores, integración de bases de datos, control de versiones y más.
- Autocompletado y resaltado de sintaxis: Mejora la productividad al escribir código.
- Facilidad de gestión de proyectos: Ideal para proyectos más grandes y complejos, proporcionando un entorno organizado.
- Integración de pruebas: Permite realizar pruebas unitarias y otras validaciones directamente desde el IDE.



Compiladores e intérpretes

Compiladores e intérpretes son herramientas fundamentales en la programación que transforman el código fuente en instrucciones que una computadora puede ejecutar, pero lo hacen de manera diferente.

- **Compilador:** Traduce todo el código fuente de un programa a código máquina de una sola vez, creando un archivo ejecutable. Esto permite que el programa se ejecute rápidamente, pero requiere un paso previo de compilación. Ejemplos de lenguajes que usan compiladores incluyen C, C++ y Java (que se compila a bytecode, luego ejecutado por la JVM).
- **Intérprete:** Traduce el código fuente línea por línea, ejecutando las instrucciones de manera inmediata. Esto hace que el desarrollo sea más flexible, pero generalmente más lento, ya que la traducción y ejecución ocurren simultáneamente. Ejemplos de lenguajes interpretados son Python, JavaScript y Ruby.





Fundamentos de programación

🚀 Conceptos básicos :

- Variables: Son espacios de almacenamiento en memoria que se utilizan para guardar valores que pueden cambiar durante la ejecución del programa. Cada variable tiene un nombre y un tipo de dato que define qué tipo de información puede almacenar. Ejemplos de variables pueden ser: edad, nombre, temperatura.
- Constantes: Son similares a las variables, pero su valor no puede modificarse una vez asignado. Se utilizan para almacenar valores fijos que no cambian durante el ciclo de vida del programa. Ejemplos de constantes pueden ser: PI = 3.1416, TASA_IMPUUESTO = 0.16.
- Tipos de datos: Los tipos de datos definen el tipo de información que puede almacenar una variable o constante. Los tipos de datos más comunes son:
- Números: Representan valores numéricos y se utilizan para realizar operaciones matemáticas y lógicas.
- Enteros (int): Números sin decimales, como 1, -5, 42.
- Flotantes (float): Números con decimales, como 3.14, -0.001, 100.5.
- Cadenas (string): Representan secuencias de caracteres y se usan para almacenar texto.
- Ejemplos: "Hola", "123", "Programación".
- Fechas (date): Representan valores relacionados con el tiempo, como días, meses y años.
- Ejemplos: 2025-03-07, "2023-12-31". Dependiendo del lenguaje de programación, el formato y tipo de dato pueden variar (como datetime en Python o DATE en SQL).



QUE ES

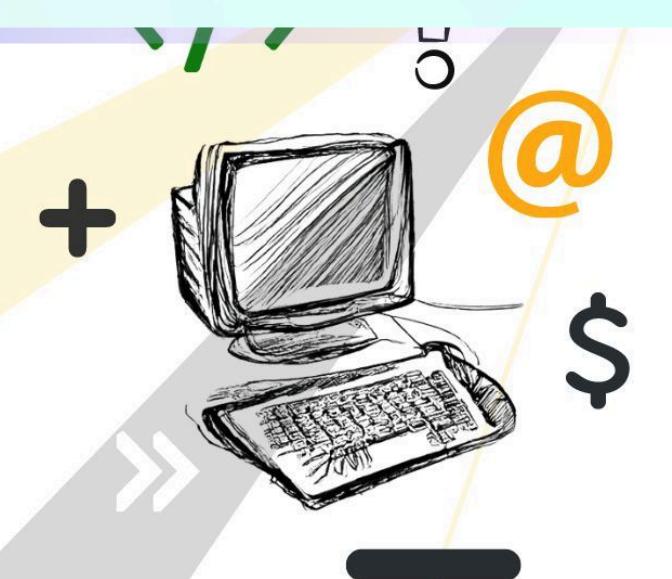


OPERADOR

+ Operadores y expresiones

- Operadores Aritméticos: Se utilizan para realizar operaciones matemáticas sobre números. Los principales operadores aritméticos son: + (suma), - (resta), * (multiplicación), / (división), % (módulo) y // (división entera).
- Operadores Lógicos: Se utilizan para realizar operaciones lógicas que devuelven un valor booleano (True o False). Los operadores lógicos más comunes son: and (y), or (o) y not (no).
- Operadores Relacionales: Se utilizan para comparar dos valores y obtener un resultado booleano. Los operadores relationales incluyen: == (igual a), != (diferente de), > (mayor que), < (menor que), >= (mayor o igual que) y <= (menor o igual que).

QUÉ SON LOS OPERADORES EN PROGRAMACIÓN





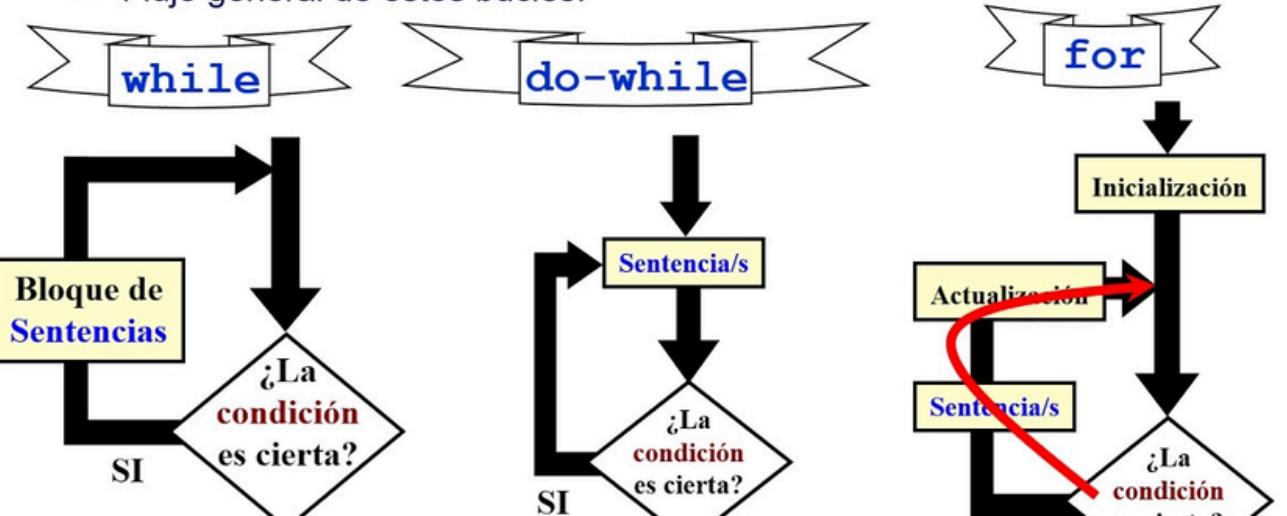
Fundamentos de programación

Control de flujo:

- Condicionales: Permiten tomar decisiones en el código basadas en una condición booleana.
- if: Ejecuta un bloque de código si una condición es verdadera.
- switch: Evalúa una expresión y ejecuta el bloque de código correspondiente al valor de la expresión, utilizado generalmente en lenguajes como C, C++ o Java.
- Bucles: Permiten repetir un bloque de código mientras se cumpla una condición.
- for: Se utiliza para iterar sobre un rango o colección de elementos, realizando una acción repetida durante un número determinado de veces.
- while: Ejecuta un bloque de código mientras una condición sea verdadera. El número de repeticiones depende de la evaluación de la condición en cada iteración.

- Un **proceso repetitivo, iterativo o bucle** permite que un conjunto de sentencias se ejecute varias veces.
- En **C** hay tres tipos de bucles: **while**, **do-while** y **for**.
 - Todos requieren una **condición** que determina si el bucle continua o no:
 - Si la **condición** es cierta el bucle continua, si es falsa se detiene.

- Flujo general de estos bucles:





REFERENCIAS



- . Nutshellapp. (s.f.). La evolución de los lenguajes de programación: una visión histórica. Recuperado el 7 de marzo de 2025, de <https://www.nutshellapp.com/publicsummaries/la-evolucion-de-los-lenguajes-de-programacion-una-vision-historica>
- Tiffin University. (2023, 22 de febrero). Importancia de la programación. Recuperado el 7 de marzo de 2025, de <https://global.tiffin.edu/blog/importancia-de-la-programacion>
- KeepCoding. (2022, 9 de noviembre). Lenguaje máquina: Guía para principiantes. Recuperado el 7 de marzo de 2025, de <https://keepcoding.io/blog/lenguaje-maquina-guia-para-principiantes/>
- EducaOpen. (s.f.). Lenguaje ensamblador: Qué es y para qué sirve. Recuperado el 7 de marzo de 2025, de <https://www.educaopen.com/digital-lab/blog/software/lenguaje-ensamblador>
- NinjaOne. (s.f.). Lenguaje de programación de alto nivel. Recuperado el 7 de marzo de 2025, de <https://www.ninjaone.com/es/it-hub/it-service-management/lenguaje-de-programacion-de-alto-nivel/>
- Blancarte, Ó. (2017, 26 de octubre). IDE vs editor de texto. Oscar Blancarte Blog. Recuperado el 7 de marzo de 2025, de <https://www.oscarblancarteblog.com/2017/10/26/ide-vs-editor-de-texto/>
- IONOS. (s.f.). Compilador e intérprete: Diferencias y ejemplos. Recuperado el 7 de marzo de 2025, de <https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/compilador-e-interprete/>