





Tema 24

Lenguajes de programación

Tipos, características y paradigmas

1.1 Introducción

 Un lenguaje de programación es un sistema **formal** para expresar algoritmos.

- Es el puente entre la lógica del programador y la máquina
- Ha evolucionado con nuevas necesidades:
 -  Eficiencia
 -  Seguridad
 -  IA
 -  Desarrollo web

1.2 Elementos fundamentales




 Sintaxis: cómo se escribe

 Semántica: qué significa

Estructuras de control:

- Secuencia
- Condicionales: `if`, `switch`
- Bucles: `for`, `while`

Tipos de datos

-  Primitivos: `int` , `char` , `bool`
-  Estructurados: `arrays` , `struct`
-  Abstractos: `listas` , `pilas` , `árboles` , `diccionarios`









Tipado

Tipo	Característica	Ejemplos
Estático	Se conoce en compilación	C , Java
Dinámico	Se evalúa en tiempo de ejecución	Python , JS
Fuerte	No permite conversiones implícitas inseguras	Python
Débil	Permite conversiones automáticas	JavaScript

Cualidades deseables




-  Legibilidad
-  Seguridad
-  Eficiencia
-  Portabilidad

1.3 Paradigmas de programación





-  **Imperativo** → Cómo hacerlo (C, Python)
-  **Declarativo** → Qué lograr (SQL)
-  **Funcional** → Funciones puras (Haskell)
-  **Orientado a objetos** → Objetos con estado (Java, Python)
-  **Lógico** → Reglas y hechos (Prolog)
-  **Reactivo** → Basado en eventos (JavaScript + Vue)
-  **Tiempo real** → Respuesta garantizada (C embebido)
-  **Cuántico** → Qubits y superposición (Q#)

1.4 Clasificación de lenguajes

Por nivel de abstracción:

-  Bajo nivel: ensamblador
-  Medio nivel: `C`, `Rust`
-  Alto nivel: `Python`, `Java`

Por ejecución:

-  Compilados → C , C++
-  Interpretados → Python , JS
-  Híbridos → Java , C#
-  Transpilados → TypeScript → JavaScript

Por generación:


- 🏛️ Clásicos: Pascal , COBOL
- 🆕 Modernos: Kotlin , Go , Rust
- 🚀 Emergentes: Q# , Cirq , Mojo

1.5 Lenguajes y usos comunes

Lenguaje	Usos principales
C/C++	Sistemas, controladores, embebidos
Java	Apps empresariales, Android
Python	IA, datos, automatización
JavaScript	Web frontend y backend
SQL	Bases de datos relacionales
Q#	Programación cuántica

1.6 Herramientas de desarrollo

 **Compiladores:** `gcc` , `javac`

 **Intérpretes:** `python` , `node`

 **IDEs:**

- Visual Studio Code
- IntelliJ
- Replit (en la nube)

1.7 Tendencias actuales

☁️ **Cloud-native:** desarrollo para la nube (Go , Python)

🧠 **IA / ML:** uso de librerías como TensorFlow (Python)

🎨 **No-code / low-code:** Appgyver, Glide

🧬 **Cuántica:** Q# , Cirq

🤝 **Colaboración remota:**

- GitHub Codespaces
- Google Colab
- Live Share (VS Code)

1.8 Conclusión

✓ No existe un único lenguaje ideal: depende del propósito.

- 🧠 Múltiples paradigmas conviven
- 🛠️ La herramienta depende del problema
- 🌍 La tendencia va hacia la accesibilidad y colaboración


1.9 ¿Cómo elegir un lenguaje?

Criterios clave:

- Tipo de proyecto (web, IA, móviles...)
- Rendimiento necesario
- Curva de aprendizaje
- Portabilidad
- Productividad y ecosistema
- Comunidad y soporte
- Seguridad y mantenibilidad

Cierre

 Los lenguajes son herramientas para **pensar y construir soluciones**.

 Saber cómo y cuándo elegir uno marca la diferencia entre un desarrollador principiante y uno profesional.

 ¡Explora, compara y combina lo mejor de cada mundo!