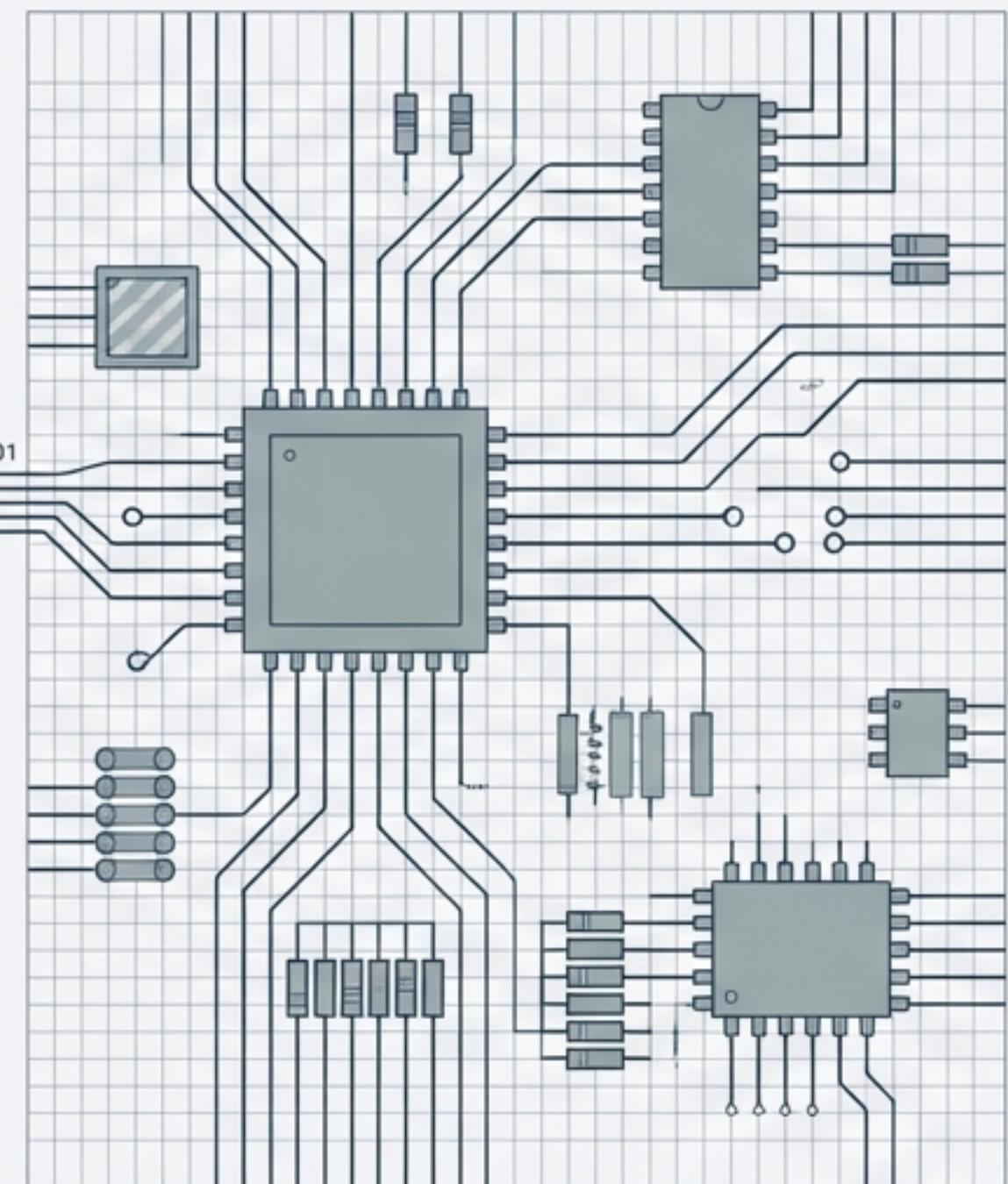
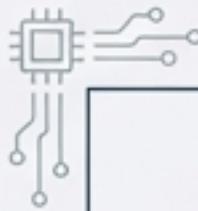


Del Código al Silicio: Del Código al Silicio: Entendiendo el Software

Fundamentos, Lenguajes y Paradigmas de Desarrollo



El Problema de la Comunicación



LENGUAJE NATURAL

Ambiguo, Abstracto.



LENGUAJE MÁQUINA

Corrientes Eléctricas, Binario, Rígido.

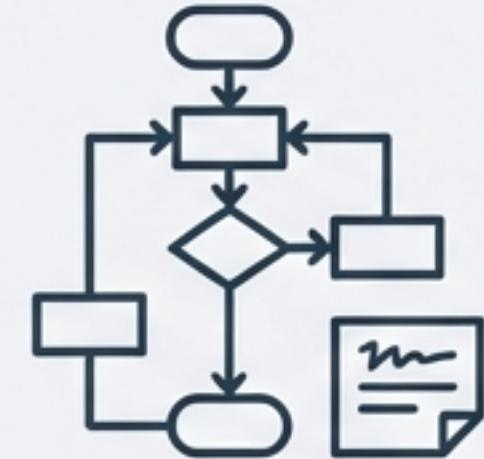
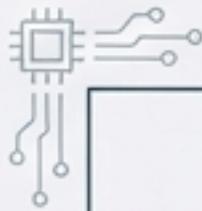


Aunque el hardware y el software han evolucionado en paralelo hacia una mayor complejidad, el principio básico de ejecución se mantiene intacto: transformar ideas en algo que la CPU pueda procesar.

INSIGHT CLAVE: Los programas actuales se escriben en lenguajes de alto nivel para facilitar el desarrollo, pero el hardware sigue requiriendo un modelo de ejecución de bajo nivel.

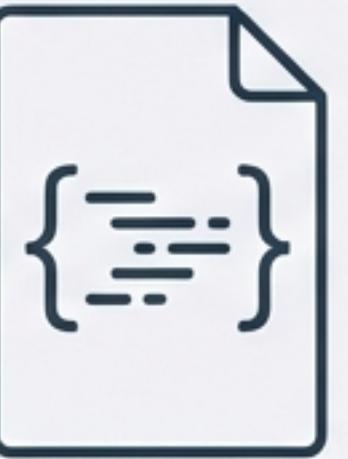


La Receta Detallada: Definiciones Clave



1. Algoritmo

La lógica pura. Los pasos necesarios para resolver un problema (ej. ordenar una lista). No es necesariamente informático.



2. Programa Informático

La “receta” sin ambigüedades. Una secuencia de instrucciones lógicas específicas para resolver un problema determinado.



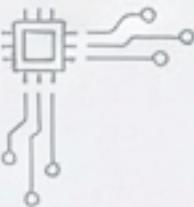
3. Software

El conjunto. Una colección de programas que se combinan para un fin (ej. Sistema Operativo, BIOS).

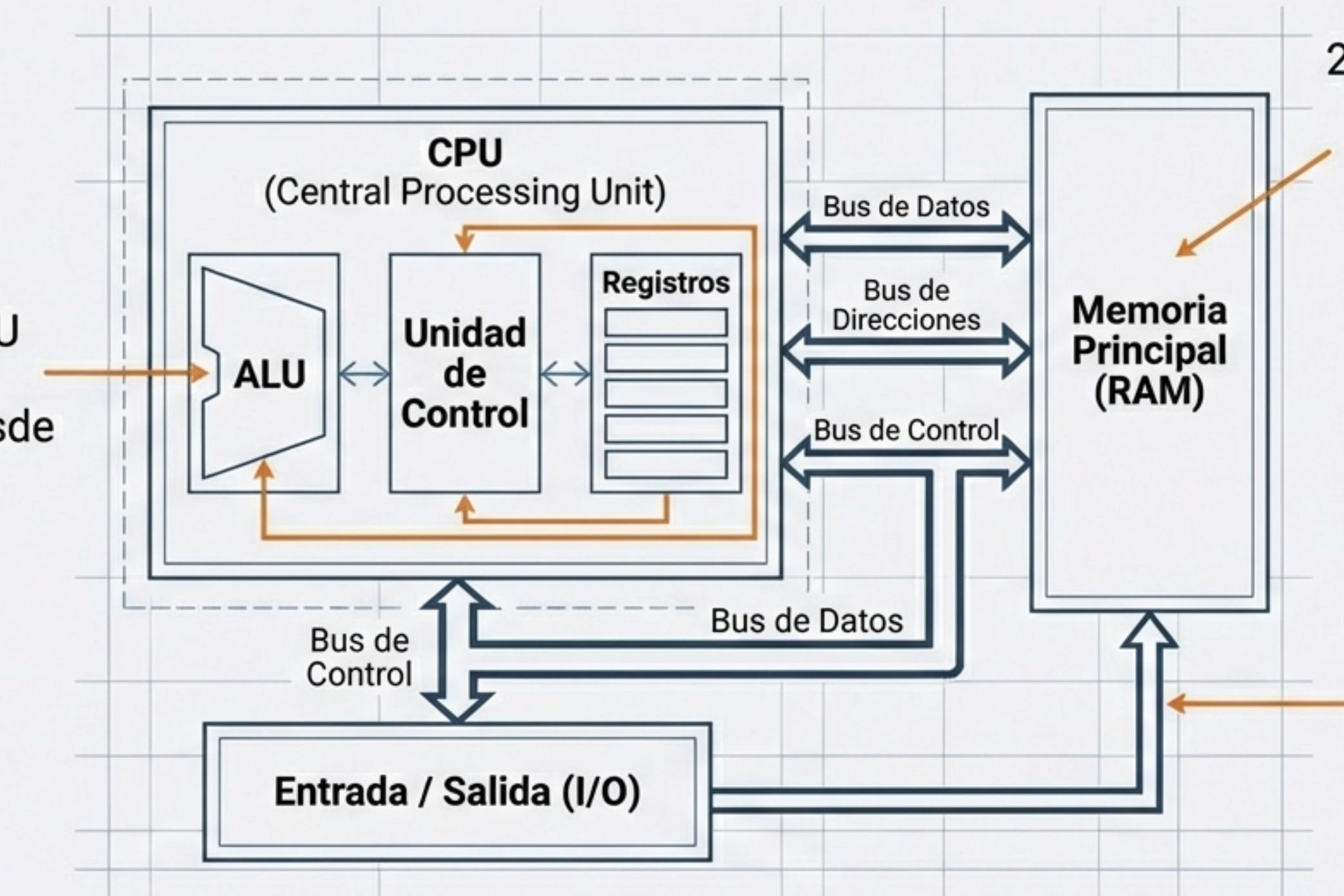
Nota: Para que un programa funcione, necesita dos cosas: instrucciones y datos. Un programa informático no puede ser ambiguo.



El Destino: Arquitectura Von Neumann



3. **Ejecución:** La CPU lee y ejecuta las instrucciones desde la RAM.



1. **Carga:** El Sistema Operativo mueve el programa del almacenamiento a la RAM.

2. **Almacenamiento:** Datos e instrucciones residen temporalmente en la RAM.

El software no flota en el aire; debe ser cargado físicamente en la memoria para ser ejecutado.



La Torre de Abstracción

La capacidad de obviar los detalles complejos del hardware.



Java, Kotlin, C++.

Cercano al lenguaje natural.
Oculta la CPU.

C, Fortran.

Equilibrio entre abstracción y
control.

Lenguaje Ensamblador.

Dependencia total de la máquina.
Sin abstracción.

Caso Práctico: El Portal de Comunicación



El Reto

El CTO solicita una plataforma web + Apps (Android/iOS).

Decisión Estratégica:

X **¿Bajo Nivel?** No. Demasiado dependiente del hardware y difícil de portar.

✓ **¿Alto Nivel?** Sí. Necesitamos ocultar la complejidad de las diferentes plataformas.

La Solución Stack

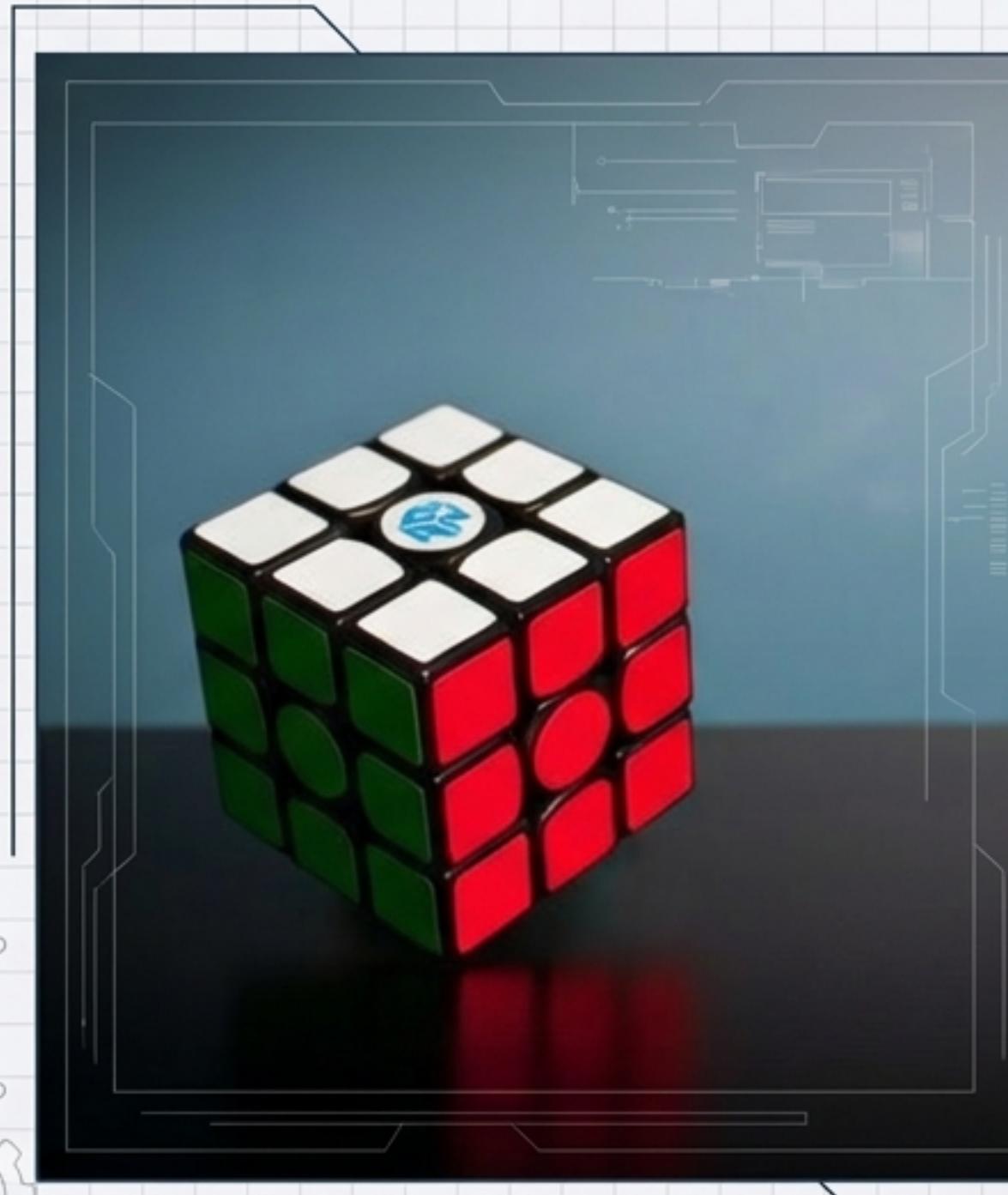


Frontend: HTML, CSS, JavaScript

Backend: PHP o Java

Alternativa: CMS
(Drupal/Wordpress) + API Rest

Estilos de Mandato: Paradigmas de Programación



Imperativo (El Cómo)

Especifica el flujo de control paso a paso.

Incluye Programación Orientada a Objetos (POO).

Es el enfoque tradicional.

Declarativo (El Qué)

Especifica el resultado deseado, no los pasos.

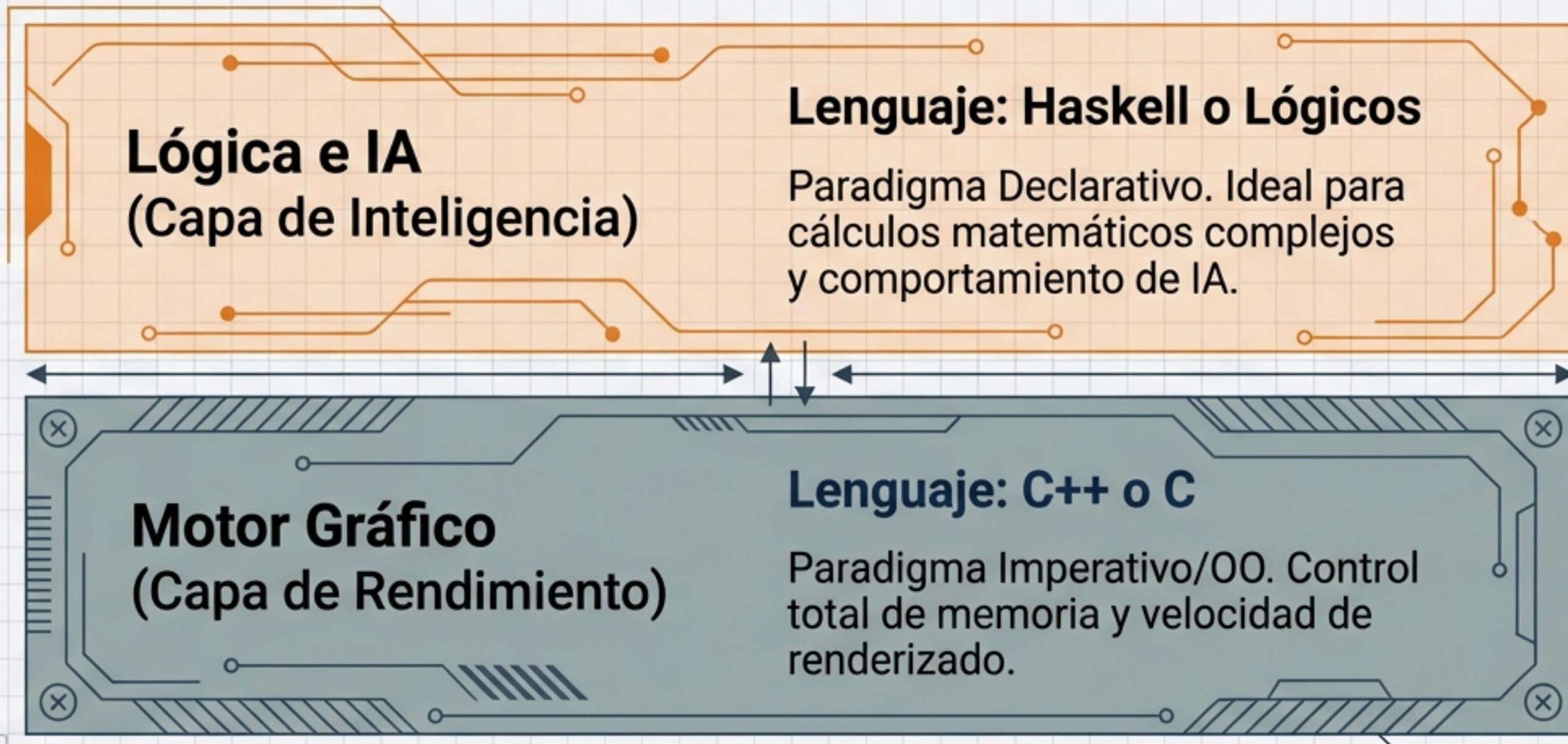
Se basa en relaciones lógicas y matemáticas.

Incluye Funcional.

Convergencia Moderna: Lenguajes como Kotlin o Swift son híbridos; imperativos en estructura pero integran funciones 'Lambda' del paradigma funcional.

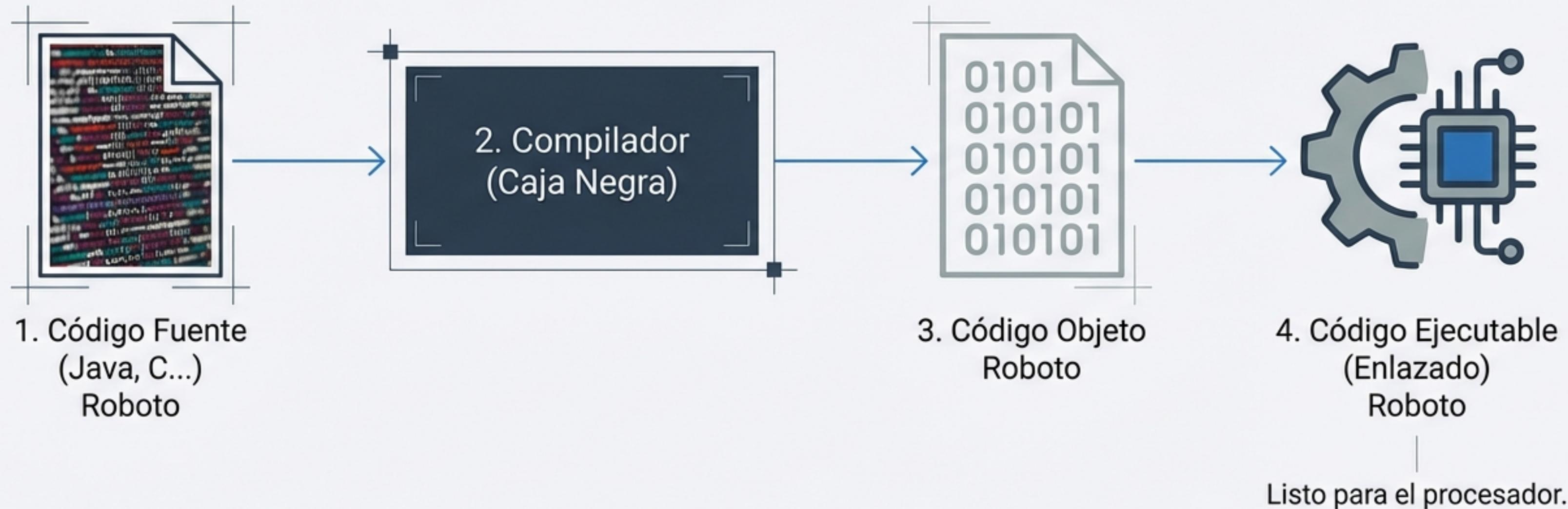
Caso Práctico: El Motor de Videojuegos

Solución Híbrida para Alto Rendimiento



No existe un “mejor paradigma”. Se elige la herramienta según el módulo del problema.

La Traducción: El Proceso de Compilación



Fase de Análisis: Entendiendo la Intención



Análisis Léxico

¿Son estas palabras válidas? (Comprobación de vocabulario)

Análisis Sintáctico

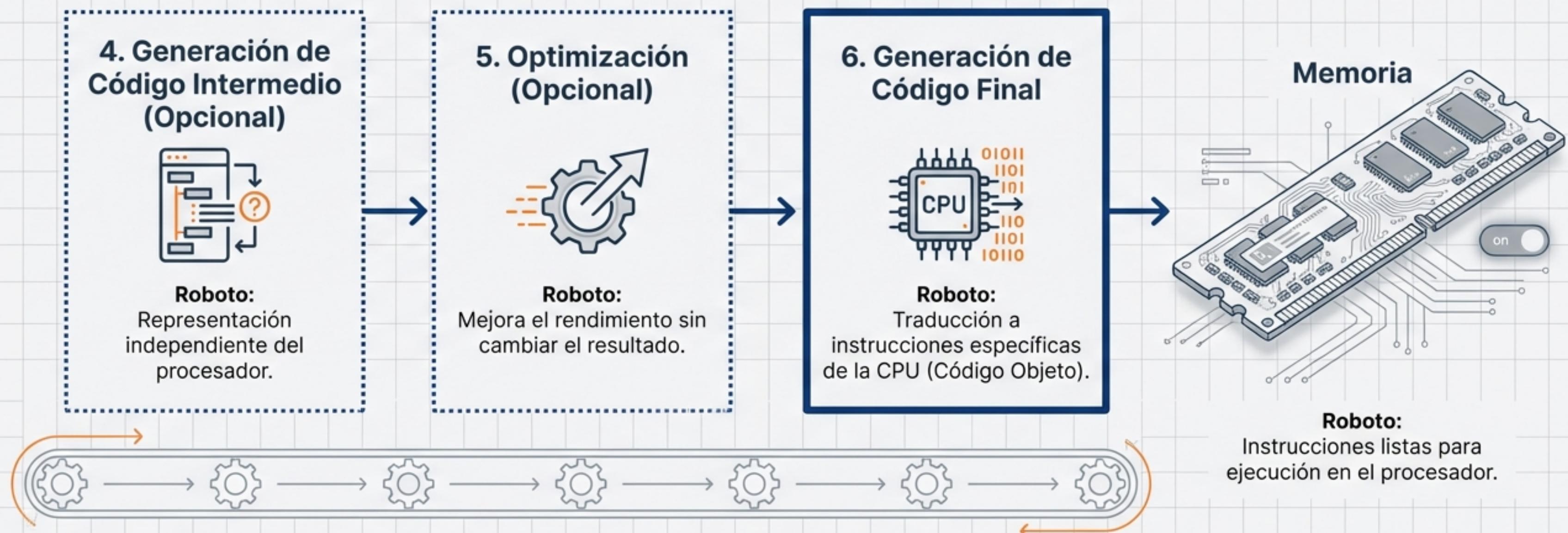
¿Está el orden correcto?
(Verificación de gramática y secuencia)

Análisis Semántico

¿Tiene sentido la frase?
(Lógica y significado válido)

Comprensión del Código Fuente

Fase de Síntesis: Generando Instrucciones



Resumen: La Ingeniería de la Elección

Niveles de Abstracción



Elegir cuánto nos acercamos al metal o al humano.
(Alto vs. Bajo).

Paradigmas



Herramientas de pensamiento.
Imperativo para control,
Declarativo para lógica.

El Puente (Compilación)



Análisis y Síntesis. Garantiza que la idea abstracta se ejecute en la arquitectura Von Neumann.

« La clave no es saber un solo lenguaje, sino saber cuál se ajusta al problema. »

Bibliografía y Referencias

Villalba, C., Moraleda, A. & Iez, M. (2011). Lenguajes de programación. Madrid: UNED.

Aho, A., Sethi, R., Ullman, J., Suárez, P. & López, P. (1998). Compiladores: principios, técnicas y herramientas. México: Addison-Wesley.

Alexander, A. (2017). Functional programming, simplified. Boulder, Colo: Alvin Alexander.

Contenido basado en “Entornos de desarrollo. Tema 1” (MEDAC).