



INSTITUTO  
TECNOLOGICO  
DE PACHUCA

SISTEMAS COMPUTACIONALES

# Programacion Web

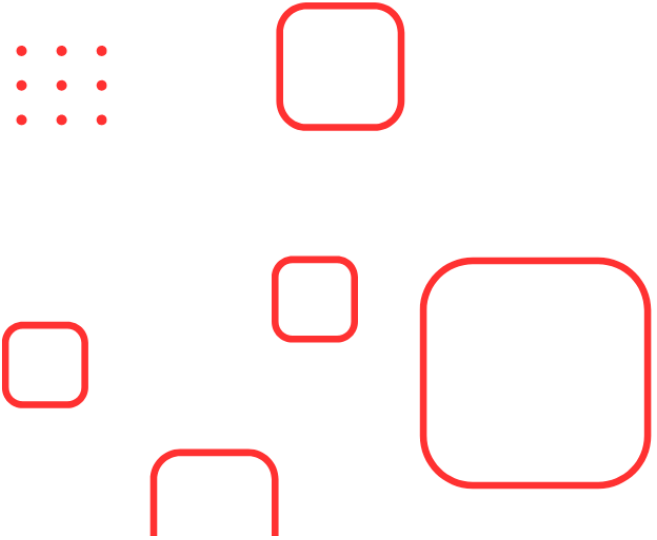

U1\_A1: resumen de lenguajes de programación compilados e interpretados

Nombre del profesor: FLORES BACA EFREN

GARCÍA SANDOVAL JORGE RAFAEL

Grupo B

30/01/2025



## Introducción

---

En los momentos actuales la programación es un pilar fundamental en la informática, y uno de los aspectos más importantes a considerar al seleccionar un lenguaje es cómo se traduce y ejecuta el código. Tradicionalmente, los lenguajes de programación se han dividido en dos categorías principales: **lenguajes compilados** y **lenguajes interpretados**. Esta clasificación afecta el rendimiento, la portabilidad y la flexibilidad del lenguaje en distintos entornos. Sin embargo, en la actualidad, muchos lenguajes adoptan un enfoque híbrido, combinando características de ambos métodos de ejecución.

## Compilación e Interpretación: Definición y Diferencias

---

Los lenguajes compilados requieren un programa llamado compilador, que traduce el código fuente completo a código máquina antes de su ejecución. Este proceso implica varias etapas: análisis léxico, análisis sintáctico, optimización y generación del código final. Una vez compilado, el programa resultante puede ejecutarse directamente en el sistema operativo sin necesidad del código fuente ni del compilador. Esto permite una ejecución más rápida y eficiente, pero también implica que cualquier modificación en el código requiere una recompilación completa.

Por otro lado, los lenguajes interpretados utilizan un intérprete, que lee y ejecuta el código línea por línea en tiempo real sin necesidad de generar un ejecutable previo. Esto facilita la depuración y la portabilidad, ya que el mismo código puede ejecutarse en distintos sistemas sin necesidad de recompilación. Sin embargo, este enfoque suele ser más lento, ya que la traducción ocurre en el momento de la ejecución.

Aunque esta clasificación ha sido ampliamente utilizada, diversos autores, como Scott (2009) en *Programming Language Pragmatics* y Sebesta (2019) en *Concepts of Programming Languages*, enfatizan que la distinción no es estricta, ya que muchos lenguajes pueden tener compiladores e intérpretes según la implementación utilizada.

## Lenguajes Compilados: Características y Ejemplos

---

Los lenguajes compilados se destacan por su eficiencia y optimización. Según Aho et al. (2006) en *Compilers: Principles, Techniques, and Tools*, el proceso de compilación incluye varias fases clave, como el análisis del código fuente, la generación de un árbol de sintaxis, la optimización y la conversión final a código máquina.

Entre las principales ventajas de los lenguajes compilados se encuentran su alto rendimiento, ya que el código se traduce completamente antes de ejecutarse, permitiendo una ejecución más rápida y eficiente. Además, los compiladores pueden aplicar técnicas avanzadas de optimización que mejoran la utilización de los recursos del sistema. No obstante, una desventaja significativa es el tiempo requerido para la compilación, especialmente en proyectos grandes. También, cualquier cambio en el código fuente requiere recompilación antes de ejecutarse nuevamente.

Algunos de los lenguajes compilados más representativos incluyen **C, C++, Rust, Go y Swift**. C y C++ son lenguajes tradicionales utilizados en el desarrollo de sistemas operativos, software de alto rendimiento y aplicaciones embebidas. Rust ha ganado

popularidad por su énfasis en la seguridad de memoria sin sacrificar rendimiento, mientras que Go se destaca en entornos de computación en la nube. Swift, desarrollado por Apple, es ampliamente utilizado en el ecosistema de macOS e iOS.

### Lenguajes Interpretados: Características y Ejemplos

---

En contraste con los lenguajes compilados, los lenguajes interpretados ofrecen una mayor flexibilidad y facilidad de uso. Según Abelson et al. (1996) en *Structure and Interpretation of Computer Programs*, la interpretación permite una ejecución inmediata del código sin necesidad de un proceso previo de compilación, lo que facilita el desarrollo y la depuración.

Entre sus ventajas se encuentran la portabilidad, ya que el mismo código puede ejecutarse en múltiples plataformas sin modificaciones, y la capacidad de realizar cambios en el código sin necesidad de recompilarlo. Sin embargo, su principal desventaja es el menor rendimiento en comparación con los lenguajes compilados, ya que la traducción a código máquina se realiza durante la ejecución.

Ejemplos de lenguajes interpretados incluyen **Python, JavaScript, Ruby, PHP y Perl**. Python es ampliamente utilizado en inteligencia artificial, análisis de datos y desarrollo web, mientras que JavaScript es el lenguaje estándar para el desarrollo web del lado del cliente. Ruby es conocido por su sintaxis elegante y facilidad de uso en el desarrollo de aplicaciones web con Ruby on Rails, PHP es común en el desarrollo de sitios web dinámicos, y Perl es utilizado en administración de sistemas y procesamiento de texto.

### Lenguajes Híbridos: Un Enfoque Intermedio

---

Con el avance de la tecnología, muchos lenguajes han adoptado un enfoque híbrido, combinando compilación e interpretación para optimizar tanto el rendimiento como la flexibilidad. En estos casos, el código fuente se compila primero a un bytecode intermedio, que luego es ejecutado por una máquina virtual. Este enfoque permite un equilibrio entre la eficiencia de los lenguajes compilados y la portabilidad de los lenguajes interpretados.

Un ejemplo claro es Java, cuyo código fuente se compila en bytecode y se ejecuta en la Java Virtual Machine (JVM), permitiendo su ejecución en cualquier sistema que tenga una JVM instalada. De manera similar, C# usa el Common Language Runtime (CLR) de .NET para lograr independencia de la plataforma. Python también sigue este modelo, generando archivos .pyc que contienen bytecode ejecutable por el intérprete. JavaScript, aunque tradicionalmente interpretado, ha mejorado su rendimiento mediante compilación Just-In-Time (JIT) en motores como V8 de Google Chrome y SpiderMonkey de Firefox.

### Factores a Considerar al Elegir un Lenguaje

---

La elección de un lenguaje de programación depende de varios factores. Si se busca máximo rendimiento, los lenguajes compilados como C, C++ o Rust son ideales. Para

desarrollo rápido y flexible, lenguajes interpretados como Python y JavaScript ofrecen ventajas significativas. Si la portabilidad es una prioridad, lenguajes híbridos como Java y C# proporcionan compatibilidad multiplataforma sin perder eficiencia.

Además, es importante considerar el contexto de uso. Lenguajes como C y Rust son preferidos en sistemas embebidos y programación de bajo nivel, mientras que Python y JavaScript dominan en desarrollo web y aplicaciones de alto nivel. Go y Swift, por su parte, han sido diseñados para entornos modernos como el desarrollo en la nube y aplicaciones móviles, respectivamente.

## Conclusión

---

La dicotomía entre lenguajes compilados e interpretados ha evolucionado con el tiempo. Mientras que los lenguajes compilados siguen siendo la opción preferida en aplicaciones de alto rendimiento, los lenguajes interpretados han crecido en popularidad debido a su facilidad de uso y portabilidad. Por otro lado, los enfoques híbridos han permitido combinar lo mejor de ambos mundos, optimizando la ejecución sin perder flexibilidad.

Según Scott (2009), la distinción entre compilado e interpretado ya no es tan rígida como en las primeras décadas de la informática, y hoy en día, la elección de un lenguaje depende más de las herramientas y el ecosistema que de su mecanismo de ejecución. En última instancia, comprender estas diferencias y sus implicaciones permite tomar decisiones informadas para elegir el lenguaje adecuado según el contexto y los objetivos del desarrollo de software.

## Bibliografía

---

*Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2006). Compilers: Principles, Techniques, and Tools (2nd ed.). Pearson.*

*Scott, M. L. (2009). Programming Language Pragmatics (3rd ed.). Morgan Kaufmann.*

*Sebesta, R. W. (2019). Concepts of Programming Languages (12th ed.). Pearson.*