

Nombre: <Zully Fernanda Ortiz Avendaño>

Correo: <zullyortiz.09@gmail.com>

Grupo: T2

Compiladores e interpretes

Los compiladores e intérpretes son herramientas utilizadas en la ejecución de programas escritos

en lenguajes de programación. Aunque ambos cumplen la función de traducir el código fuente en

lenguaje humano a la máquina, lo hacen de manera diferente y tienen distintas implicaciones en el

proceso de ejecución. Aquí te explico cada uno y te proporciono ejemplos:

Compiladores:

- **Funcionamiento:** Un compilador traduce el código fuente completo en un lenguaje de

programación en código máquina u otro lenguaje de nivel inferior antes de su ejecución.

- **Proceso:** El código fuente se pasa por el compilador en su totalidad, y el resultado es un archivo

ejecutable que contiene el código máquina.

- **Ejecución:** El programa compilado puede ejecutarse en múltiples ocasiones sin necesidad de

recompilar, a menos que se realicen cambios en el código fuente.

- **Ventajas:** El código compilado suele ser más rápido en su ejecución ya que se traduce

completamente antes de ser ejecutado.

- **Ejemplos:** GCC (GNU Compiler Collection) para C/C++, Visual C++ Compiler, Swift Compiler, etc.

Interpretes:

- **Funcionamiento:** Un intérprete traduce y ejecuta el código fuente línea por línea, instrucción por

instrucción, en tiempo real.

- **Proceso:** El intérprete lee cada línea del código fuente, la traduce y la ejecuta de inmediato. No

genera un archivo ejecutable separado.

- **Ejecución:** Cada vez que se ejecuta el programa, el intérprete realiza la traducción y ejecución en

tiempo real.

- **Ventajas:** La depuración y la modificación del código son más fáciles, ya que los errores se

detectan a medida que el programa se ejecuta.

- **Ejemplos:** Python (Python Interpreter), JavaScript (Browser's JavaScript Engine), Ruby (Ruby

Interpreter), etc.

Lenguajes de tipado fuerte y tipado débil

Lenguajes tipados (Strongly Typed):

- En un lenguaje tipado, se requiere que las variables sean declaradas con un tipo de dato

específico antes de que se les pueda asignar un valor. Esto significa que el tipo de dato de una variable se verifica en tiempo de compilación o en tiempo de ejecución, según el lenguaje.

- Las operaciones y funciones en lenguajes tipados se aplican a variables del mismo tipo, lo

que ayuda a prevenir errores de tipo durante la ejecución.

- Ejemplos de lenguajes tipados incluyen C, C++, Java, C#, Python (a partir de Python 3.5 con

anotaciones de tipo) y TypeScript

Lenguajes no tipados (Dynamically Typed):

- En un lenguaje no tipado, las variables no están vinculadas a un tipo de dato específico

durante la declaración. Puedes asignar diferentes tipos de datos a la misma variable a lo largo de la ejecución del programa.

- La verificación de tipos se realiza en tiempo de ejecución, lo que significa que los errores

de tipo pueden no aparecer hasta que el programa se esté ejecutando.

- Ejemplos de lenguajes no tipados incluyen JavaScript y PHP