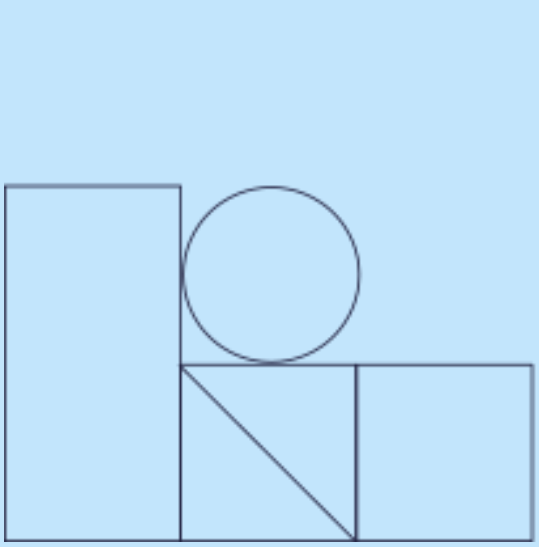
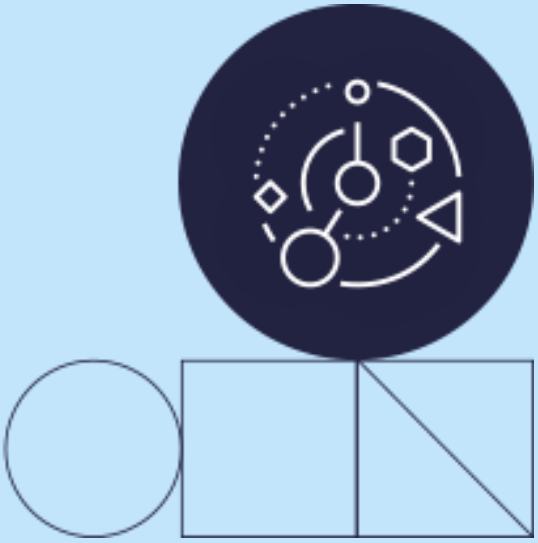
IBM SkillsBuild

Conceptos básicos de programación

Lenguaje compilado *vs.* Lenguaje interpretado





Índice

Introducción

El idioma siempre ha sido el medio de comunicación más eficaz. La buena comunicación es siempre un puente de la confusión a la claridad. No solo los humanos, todos y cada uno de los elementos de esta naturaleza también tienen su propio lenguaje para compartir sus emociones y sentimientos con los demás. A veces, incluso una suave brisa nos habla de cientos de cosas.

Hoy en esta era 21, la era de la tecnología, las computadoras tienen lenguajes para comunicarse con nosotros los humanos... el lenguaje de programación.

El lenguaje de programación se puede clasificar en dos tipos:

- Lenguaje compilado
- Lenguaje interpretado

¿Qué es el lenguaje compilado?

Los lenguajes de programación que la máquina de destino puede leer sin la ayuda de otro programa. Generalmente, los humanos no pueden entender el código de este idioma.

Compilador

Es un programa que toma como entrada un texto escrito en un lenguaje fuente, y como salida proporciona otro texto en un lenguaje objeto. Su función es traducir lenguaje fuente a lenguaje objeto.

Un compilador no funciona de manera aislada.

Necesita de otros programas para conseguir su objetivo: obtener un programa ejecutable a partir de un programa fuente en un lenguaje de alto nivel.
Algunos de esos programas son el preprocesador, el linker, el depurador y el ensamblador.
El preprocesador se ocupa de incluir ficheros, expandir macros, eliminar comentarios, y otras tareas similares.

El linker se encarga de construir el fichero ejecutable añadiendo al fichero objeto generado por el compilador las cabeceras necesarias y las funciones de librería utilizadas por el programa fuente.
El depurador permite, si el compilador ha generado adecuadamente el programa objeto, seguir paso a paso la ejecución de un programa.
Muchos compiladores, en vez de generar código objeto, generan un programa en lenguaje ensamblador que debe después convertirse en un ejecutable mediante un programa ensamblador.

Ventajas

- Ejecución rápida.
- El código compilado no puede ser "abierto" por otras personas.
- No es necesario transmitir el código fuente.
- El código compilado se compacta en un solo archivo.

Desventajas

- El código compilado suele ocupar bastante en disco, ya que incorpora en el propio código algunas librerías del sistema.
- Depurar un programa implica volver a compilar tras los cambios.

¿Qué es el lenguaje interpretado?

Son lenguajes de programación que la máquina de destino no puede leer por sí mismos. Las máquinas siempre necesitan otro programa para la conversión de instrucciones a su forma legible.

Básicamente, las palabras clave de cualquier lenguaje de programación deben convertirse a lenguaje de máquina, para que el código pueda ejecutarse y darnos la salida a través del sistema.

La metodología del lenguaje de máquina siempre está más allá de la inteligencia humana.

Es necesaria pues la mediación de un intérprete que, como vimos en anteriores lecciones, son programas traductores que transforman programas fuente escritos en lenguaje de alto nivel en programas objeto escritos en lenguaje máquina. En estos programas intérpretes la traducción se realiza de forma que después de transformar una instrucción del programa fuente en una o varias instrucciones en lenguaje máquina. No esperan a traducir la siguiente instrucción, sino que inmediatamente la ejecutan.

Ventajas

- Depuración rápida del código, ya que no es necesario volver a compilar tras un cambio.
- Mantenimiento sencillo y rápido del código fuente.

Desventajas

- La ejecución se ralentiza, al ser necesaria la interpretación línea a línea cada vez que se ejecuta el programa.
- El usuario tiene acceso al código y puede copiarlo o modificarlo.

Son ejemplos de lenguajes de código interpretado: JavaScript, Python, BASIC, PHP, etc.

Software de traducción

El software de traducción convierte las palabras clave de los lenguajes de programación en lenguaje de máquina y lo ejecuta para obtener el resultado.

Normalmente, el software del traductor funciona de dos formas:

- Como compilador
- Como intérprete

Como compilador, lo que hace el software traductor es lo siguiente:

Considera todas las instrucciones del Programa y convierte el código fuente en un código de máquina equivalente.

Por ejemplo: C, C ++, CLEO, C #... use un compilador para que estos se incluyan en la categoría de lenguajes compilados.

En el papel de un intérprete, el software del traductor toma una instrucción a la vez y la ejecuta en tiempo real. En realidad, esta es una forma interactiva de ejecutar un programa.

Por ejemplo: JavaScript, Python, BASIC, etc., use un intérprete para que se incluyan en la categoría de lenguajes interpretados.

De forma esquemática las diferencias entre lenguajes compilados e interpretados podría quedar de la siguiente forma:

Lenguaje compilado *vs.* lenguaje interpretado

Idiomas compilados	**Idiomas interpretados**
El compilador trabaja en el programa completo a la vez. Toma todo el programa como entrada.	Un intérprete lleva los programas línea a línea. Toma una declaración a la vez como entrada.
Genera el código intermedio conocido como código máquina o código objeto.	Un intérprete no genera un código intermedio conocido como código máquina.
Los lenguajes de compilación son más eficientes pero difíciles de depurar.	Los lenguajes interpretados son menos eficientes pero fáciles de depurar. Esta especialidad lo convierte en una opción ideal para nuevos estudiantes.
El compilador no permite que un programa se ejecute hasta que esté completamente libre de errores.	El intérprete ejecuta el programa desde la primera línea y detiene la ejecución solo si encuentra un error.
Compile una vez y ejecútelo en cualquier momento. El programa compilado no necesita compilarse cada vez.	Cada vez que ejecutan el programa, se interpreta línea por línea.
Los errores se informan después de que se comprueba todo el programa, los errores sintácticos y de otro tipo.	El error se informa cuando se encuentra el primer error. El resto del programa permanece sin marcar hasta que se resuelve el error.
El programa compilado ocupa más memoria porque todo el código objeto tiene que residir en la memoria.	El intérprete no genera código de memoria intermedio. Por lo tanto, los programas de intérprete son eficientes en la memoria.
Por ejemplo: C, C ++, CLEO, C #.	Por ejemplo: JavaScript, Python, BASIC, etc.

Conclusión

En el resultado del proceso de interpretación o compilación radica la diferencia entre lenguaje interpretado y compilado. Siempre un intérprete produce un resultado de un programa; mientras tanto, un compilador produce un programa escrito en lenguaje ensamblador.

Luego le toca al ensamblador de arquitectura convertir el programa resultante en código binario. Para cada computadora individual, dependiendo de su arquitectura, el lenguaje ensamblador varía. En consecuencia, solo en computadoras que tienen la misma arquitectura que la computadora en la que fueron compilados, los programas compilados pueden ejecutarse.

En los lenguajes compilados no tenemos acceso al código directamente por estar en un archivo y podemos detectar errores en tiempo de compilación y tiempo de ejecución, mientras que en los interpretados sí tenemos acceso al código porque se traduce en tiempo real y podemos detectar errores en tiempos de ejecución.