**Tarea 4**

**Lenguajes de programación**

1. ¿Qué diferencia existe entre los lenguajes declarativos y los imperativos? Nombra al menos 2 de cada tipo.

En el lenguaje declarativo se describe el resultado final deseado en lugar de mostrar todos los pasos del trabajo, es decir, no se determina el “cómo”, sino el “qué”.

Si al escribir un programa describimos qué hay que hacer, estaremos programando en forma declarativa, describimos el problema que queremos solucionar, pero no las instrucciones necesarias para resolverlo.

En el lenguaje imperativo es, al contrario, indicamos los pasos que debemos seguir y llevamos a cabo una serie de procesos para dar lugar al resultado deseado.

Como ejemplo de los lenguajes declarativos, tenemos a SQL, Haskell y Prolog, y como ejemplo de los lenguajes imperativos, tenemos C++, JavaScrip y C.

2. ¿Explica qué es compilar? ¿Explica qué es interpretar?

Compilar es el proceso de traducción del código fuente a código de máquina u otro formato ejecutable que una computadora o dispositivo pueda ejecutar.

El resultado de la compilación es un programa ejecutable.

Interpretar se refiere al proceso de ejecución directa del código fuente escrito en un lenguaje de programación de alto nivel sin la necesidad de compilarlo previamente en código máquina.

En lugar de generar un archivo binario o de máquina, como en la compilación, un intérprete lee, analiza y ejecuta el código fuente directamente en tiempo real.

3. Ventajas de los lenguajes compilados.

* Rendimiento: El código compilado tiende a ser más rápido en tiempo de ejecución que el código interpretado, ya que se traduce directamente en código de máquina optimizado para la plataforma de destino.
* Optimización: Los compiladores pueden realizar optimizaciones a nivel de máquina, como la eliminación de código no utilizado, la reorganización de instrucciones y la gestión eficiente de la memoria, lo que puede mejorar el rendimiento.
* Seguridad: El código compilado generalmente es más seguro, ya que no se puede ver el código fuente original ni modificarlo una vez que se ha compilado.
* Requisitos de instalación más bajos: Suelen requerir menos dependencias y bibliotecas externas.
* Uso eficiente de recursos: Puede utilizar de manera más eficiente los recursos de hardware.
* Detección temprana de errores: Los errores de compilación son detectados antes de que el código se ejecute.
* Compatibilidad: El código compilado es menos vulnerable a cambios en el entorno de tiempo de ejecución, lo que puede garantizar la compatibilidad a largo plazo de las aplicaciones.

4. Ventajas de los lenguajes interpretados.

* Facilidad de desarrollo: Los lenguajes interpretados a menudo tienen una sintaxis más simple y son más fáciles de aprender y usar, lo que acelera el proceso de desarrollo.
* Portabilidad: Los programas escritos en lenguajes interpretados son más portátiles, ya que el código fuente puede ejecutarse en múltiples plataformas y sistemas operativos sin necesidad de recompilar.
* Flexibilidad: Permiten una mayor flexibilidad en tiempo de ejecución, ya que pueden adaptarse a las condiciones cambiantes.
* Desarrollo más rápido: Permiten un desarrollo más rápido y ciclos de desarrollo más cortos.
* Scripts y automatización: Los lenguajes interpretados son ideales para la creación de scripts y tareas de automatización.
* Mayor legibilidad del código fuente: Los lenguajes interpretados suelen tener una sintaxis más clara y legible.

5. Nombra 2 lenguajes compilados y otros 2 interpretados.

Lenguajes compilados: C y C++.

Lenguajes interpretados: Python y JavaScript.

6. ¿Puede considerarse código objeto el bytecode generado en Java tras la compilación? Explica la respuesta.

No, el bytecode generado en Java tras la compilación no se considera código objeto.

El código objeto generalmente se refiere al código binario de máquina generado por un compilador para un lenguaje de programación de nivel más bajo y generalmente no es legible por humanos, en cambio, el bytecode de Java es legible por humanos, puedes examinarlo y descompilarlo, pero no es un código de máquina directo y no se ejecuta en la CPU.

Entonces, mientras que el bytecode de Java es el resultado de la compilación de código fuente, no se clasifica como código objeto, ya que está destinado a ser ejecutado por una máquina virtual y no se compila directamente en código de máquina.

7. Pon un ejemplo de lenguaje de los siguientes tipos:

* Bajo nivel: Lenguaje ensamblador.
* Nivel medio: C.
* Alto nivel: Java.

8. ¿Qué paradigma de programación siguen los siguientes lenguajes?

* C: Paradigma orientado a objetos.
* C++: Multiparadigma, programación estructurada y orientada a objetos, de procedimiento.
* SQL: Multiparadigma.
* Java: Paradigma de la orientación a objetos.
* Javascript: Multiparadigma, programación funcional, basada en prototipos, imperativo, interpretado.
* Lisp: Multiparadigma de propósitos generales.
* Prolog: Paradigma lógico.

Puedes consultar el siguiente enlace: <https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n>

9. Explica qué criterios pueden seguirse a la hora de elegir un lenguaje de programación para el desarrollo software.

* Requisitos del proyecto.

Algunos lenguajes son más adecuados para aplicaciones web, mientras que otros son mejores para aplicaciones móviles, análisis de datos, etc.

Por esto debemos tener en cuenta los requisitos de nuestro proyecto a la hora de elegir el lenguaje.

* Experiencia previa.

La familiaridad con el lenguaje de programación es un factor importante, si tienes experiencia previa en un lenguaje en particular, es mejor utilizar el mismo para acelerar el desarrollo.

* Rendimiento.

Es importante considerar el rendimiento del lenguaje y su capacidad para optimizar el código.

* Seguridad.

Algunos lenguajes están diseñados con características de seguridad incorporadas.

Esto puede ser importante en aplicaciones donde la seguridad es fundamental.

* Mantenimiento y su facilidad.

La legibilidad del código y la facilidad de mantenimiento son factores importantes.

Un lenguaje con una sintaxis limpia puede hacer que el mantenimiento del software sea menos costoso y propenso a errores.

* Costo.

Algunos lenguajes y entornos de desarrollo pueden implicar costos de licencia.

* Disponibilidad.

Puedes mirar si el lenguaje tiene una amplia gama de bibliotecas y frameworks disponibles que puedan acelerar el desarrollo y simplificar tareas.

* Plataforma de destino.

Debemos asegurarnos de que el lenguaje sea compatible con las plataformas de destino.