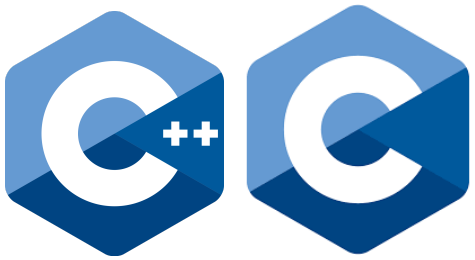
Tarea 4
Lenguajes de programación

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

Los lenguajes de programación declarativos tienen como principal objetivo lo que debe hacerse, sin tener en cuenta cómo hacerlo. En lenguaje declarativo, se debe definir el resultado deseado o las relaciones entre los datos, y el sistema se encarga de determinar la mejor manera de lograrlo. Algunos ejemplos son SQL y HTML.

Los lenguajes de programación imperativos se centran en cómo se debe hacer una tarea específica. En los lenguajes imperativos, se debe especificar detalladamente los pasos que el sistema debe seguir para alcanzar el resultado deseado. Algunos ejemplos son C y C++.



Diferencias clave entre lenguajes imperativos y declarativos

Legibilidad

- Imperativa: su comprensión tiende a ser más compleja, requieren de una curva de aprendizaje para implementarlos o interpretarlos.
- Declarativa: su comprensión tiende a ser más simple, relega en su nombre la funcionalidad que persigue.

Reusabilidad

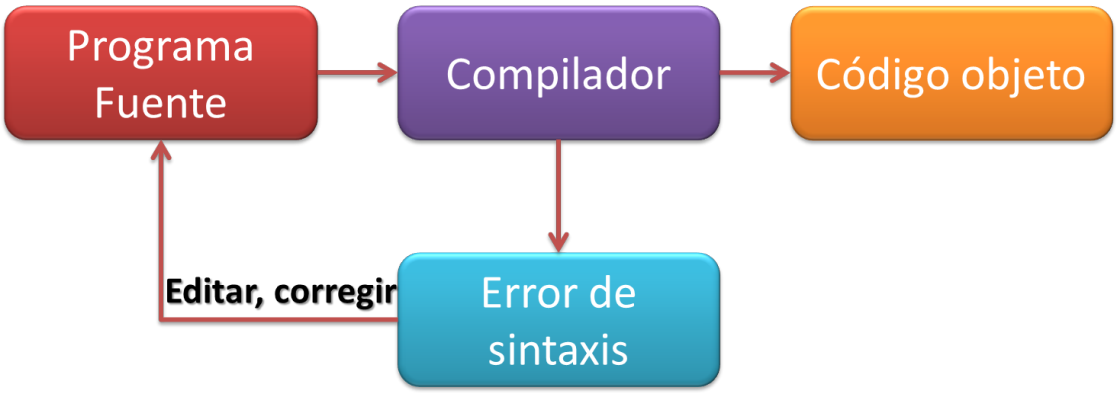
- Imperativo: Para reutilizar una misma función en diferentes partes del programa, la misma secuencia de instrucciones debe ser replicada o copiada.
- Declarativo: Los procesos y las operaciones se encapsulan. De modo que se define una operación compleja una vez y posteriormente se reutilizarla en múltiples partes del programa.

Abstracción

- Imperativo: suelen ser más detallados y cercanos a la máquina, permitiendo un control más preciso
- Declarativo: abstrae el desarrollo lógico a capas inferiores, siendo accedido a través de una función con un nombre descriptivo. No se tiene en cuenta cómo lo hace, lo importante es qué hace.

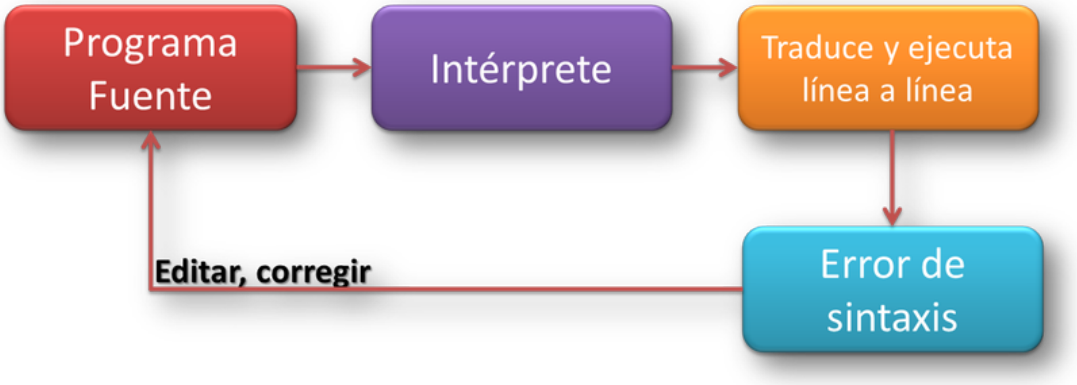
2. ¿Explica qué es compilar? ¿Explica qué es interpretar?

Compilar es el proceso de traducir por completo el código fuente que ha sido escrito por un programador de alto nivel a un lenguaje que la máquina sea capaz de comprender y ejecutar (lenguaje de máquina). Se lleva a cabo mediante la acción de un compilador.



Interpretar es el proceso mediante el cual un programa escrito en un lenguaje de programación es ejecutado línea por línea por un intérprete, a diferencia de la

compilación, donde el código se traduce completamente a un código ejecutable antes de ejecutarlo, la interpretación implica analizar y ejecutar el código fuente directamente, instrucción por instrucción, durante el tiempo de ejecución.



3. Ventajas de los lenguajes compilados.

Los lenguajes compilados son convertidos directamente a código máquina que el procesador puede ejecutar. Como resultado, suelen ser más rápidos y más eficientes al ejecutarse en comparación con los lenguajes interpretados.

Los compiladores pueden aplicar diversas técnicas de optimización al código fuente para mejorar su rendimiento.

Los errores en el código fuente se detectan durante la fase de compilación, lo que permite a los desarrolladores corregir los errores antes de que el programa se ejecute. Los programas compilados suelen utilizar menos recursos del sistema (como memoria y CPU) en comparación con los programas interpretados, ya que no hay un intérprete en ejecución que consuma recursos constantemente.

4. Ventajas de los lenguajes interpretados.

La principal ventaja de los lenguajes interpretados es que son independientes de las máquinas y de los sistemas operativos ya que no contienen instrucciones propias

Los lenguajes interpretados suelen tener una sintaxis más sencilla y permiten el desarrollo más rápido.

Los lenguajes interpretados suelen ser más flexibles y dinámicos en términos de tipos de datos y manipulación de datos. Las variables pueden cambiar de tipo durante la ejecución del programa, lo que proporciona una mayor flexibilidad en el manejo de datos.

5. Nombra 2 lenguajes compilados y otros 2 interpretados.

Como lenguajes interpretados destaca JavaScript y Python, como lenguajes compilados destaca C y C++.

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

El código fuente Java se compila y se obtiene un código binario intermedio denominado bytecode. Puede considerarse código objeto pero destinado a la máquina virtual de Java en lugar de código objeto nativo. Después este bytecode se interpreta para ejecutarlo.

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

- Bajo nivel. Código de máquina
- Nivel medio. C
- Alto nivel. Python

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

- C : Se basa en el paradigma de programación estructurada.
- C++ : Los paradigmas comúnmente asociados con C++ incluyen programación genérica y orientada a objetos, de procedimiento
- SQL : SQL es multiparadigma
- Java : Orientado a objetos
- Javascript : programación funcional
- Lisp : multiparadigma de propósitos generales
- Prolog : lógico

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

Antes de seleccionar un lenguaje de programación, es fundamental comprender tus requisitos y objetivos específicos. ¿Qué funcionalidades necesitas? ¿Qué plataformas deseas admitir? ¿Qué tipo de rendimiento esperas? Al responder a estas preguntas, podrás limitar tus opciones y centrarte en los lenguajes que mejor se adapten a tus necesidades. A continuación se exponen diferentes criterios a tener en cuenta para

elegir un determinado lenguaje.

- Campo de aplicación: El campo de aplicación se refiere a la naturaleza del problema que estás tratando de resolver con el software.
- Experiencia previa: La experiencia previa de los desarrolladores en un lenguaje particular es un criterio importante. Si un equipo de desarrollo ya tiene experiencia en un lenguaje específico, puede ser más eficiente y productivo utilizar ese lenguaje para nuevos proyectos.
- Herramientas de desarrollo: Las herramientas de desarrollo disponibles para un lenguaje pueden influir significativamente en la productividad y la calidad del código.
- Documentación disponible: La disponibilidad de documentación detallada, tutoriales y recursos educativos es esencial, especialmente para los desarrolladores principiantes o para proyectos complejos.
- Base de usuarios: A mayor cantidad de usuarios que lo vayan a poder usar mejor.
- Reusabilidad: Algunos lenguajes están diseñados con principios que fomentan la reutilización del código, como la modularidad y la orientación a objetos. Un lenguaje que permite la fácil reutilización de componentes y bibliotecas puede acelerar el desarrollo y mejorar la calidad del software.
- Portabilidad: capacidad del software para ejecutarse en diferentes plataformas y sistemas operativos sin modificaciones significativas.

- Imposición del cliente: En algunos casos, el cliente o la organización para la que se desarrolla el software puede imponer restricciones sobre el lenguaje a utilizar.