Luis Eduardo Lorduy.

Actividad De Aprendizaje (Desarrollo De Software I)

1. **Describa que es un arreglo en POO.**

En programación orientada a objetos (POO), un arreglo es una estructura de datos que almacena una colección de elementos del mismo tipo, dispuestos de manera contigua en la memoria. Los arreglos son utilizados para almacenar y manipular datos de forma eficiente, ya que permiten acceder a los elementos utilizando índices numéricos.

Los arreglos en la POO suelen tener algunas características comunes:

Tipo de elementos: Todos los elementos de un arreglo son del mismo tipo de datos. Por ejemplo, un arreglo puede contener enteros, cadenas de texto, objetos de una clase específica, etc.

Tamaño fijo: En muchos lenguajes de programación, el tamaño de un arreglo se define en el momento de su creación y no puede ser cambiado posteriormente. Sin embargo, algunos lenguajes ofrecen arreglos dinámicos que pueden cambiar de tamaño durante la ejecución del programa.

Acceso mediante índices: Los elementos dentro de un arreglo se identifican mediante un índice numérico que indica su posición dentro del arreglo. Los índices generalmente comienzan desde cero y van hasta el tamaño del arreglo menos uno.

Eficiencia en el acceso: Acceder a un elemento en un arreglo es una operación de tiempo constante, lo que significa que el tiempo necesario para acceder a cualquier elemento no depende del tamaño del arreglo, siempre y cuando se conozca el índice.

2. **Describa que es una matriz en POO.**

En la programación orientada a objetos (POO), una matriz es una estructura de datos que puede ser representada como un objeto. Una matriz es una colección de elementos, organizados en filas y columnas, donde cada elemento tiene una posición única identificada por un par de índices (fila, columna).

En la programación orientada a objetos (POO), una matriz es una estructura de datos que puede ser representada como un objeto. Una matriz es una colección de elementos, organizados en filas y columnas, donde cada elemento tiene una posición única identificada por un par de índices (fila, columna).

En el contexto de la POO, una matriz puede ser implementada como una clase. La clase matriz generalmente contendrá atributos para almacenar los elementos de la matriz y métodos para acceder y manipular estos elementos. Estos métodos pueden incluir operaciones básicas como la asignación de valores a elementos individuales, la obtención de elementos específicos por sus índices, la suma de matrices, la multiplicación de matrices, entre otras operaciones comunes.

3. Defina que es un objeto en POO.

En la programación orientada a objetos (POO), un objeto es una instancia concreta de una clase. Una clase es una plantilla o un "molde" que define la estructura y el comportamiento de los objetos que se pueden crear a partir de ella.

Un objeto tiene características (atributos o propiedades) y comportamientos (métodos o funciones). Los atributos representan el estado del objeto, mientras que los métodos representan las acciones que el objeto puede realizar.

Por ejemplo, si tienes una clase "Perro", los objetos de esta clase podrían tener atributos como "nombre", "edad" y "raza", y métodos como "ladrar" o "correr".

4. Que es una lista en POO.

En la programación orientada a objetos (POO), una lista es una estructura de datos que puede contener un conjunto de elementos ordenados. La implementación de una lista puede variar según el lenguaje de programación, pero en términos generales, una lista suele permitir agregar, eliminar y acceder a elementos de manera eficiente.

En muchos lenguajes de programación orientados a objetos, como Python o Java, las listas suelen ser clases predefinidas que proporcionan métodos para manipular los elementos que contienen. Estos métodos pueden incluir funciones para agregar elementos al final de la lista, insertar elementos en posiciones específicas, eliminar elementos, buscar elementos y acceder a elementos mediante índices.

5. Defina las diferencias entre programación secuencial (PS) y programación orientada a objetos (POO).

La programación secuencial (PS) y la programación orientada a objetos (POO) son dos enfoques distintos para desarrollar software, estas son sus principales diferencias:

Paradigma:

(PS): En la programación secuencial, el código se estructura en una secuencia lineal de instrucciones que se ejecutan una tras otra. El flujo de control sigue una ruta única, de arriba hacia abajo.

(POO): La POO se basa en el concepto de "objetos", que son entidades que encapsulan datos y funciones relacionadas. Los objetos interactúan entre sí a través de mensajes. La programación orientada a objetos se centra en modelar el mundo real mediante la creación de clases, objetos y relaciones entre ellos.

Abstracción:

(PS): En la programación secuencial, la abstracción puede ser limitada. Las funciones y procedimientos pueden usarse para dividir el código en partes más pequeñas, pero la relación entre los datos y las funciones puede no estar claramente definida.

(POO): La POO permite una abstracción más avanzada al modelar entidades del mundo real como objetos con atributos (datos) y métodos (funciones). Esto facilita la representación de conceptos complejos de manera más clara y organizada.

Reutilización de código:

(PS): En la programación secuencial, la reutilización de código puede ser más limitada. Las funciones y procedimientos pueden ser reutilizables, pero a menudo la modularidad y la reutilización pueden no ser tan eficientes como en la POO.

(POO): La POO fomenta la reutilización de código a través de la creación de clases y objetos. Los objetos pueden heredar comportamientos y características de otras clases, lo que promueve la reutilización y la modularidad del código.

Encapsulación:

(PS): En la programación secuencial, la encapsulación puede no ser tan prominente como en la POO. Los datos y las funciones pueden no estar siempre bien encapsulados, lo que puede llevar a problemas de mantenimiento y de seguridad.

(POO): La POO favorece la encapsulación al agrupar datos y funciones relacionadas en objetos. Los objetos pueden ocultar la implementación interna de sus datos y proporcionar una interfaz clara para interactuar con ellos, lo que mejora la modularidad y la seguridad del código.

6. Define cinco características de la programación estructural.

Secuencialidad: Los programas estructurados se componen de una secuencia de instrucciones que se ejecutan una tras otra. Esto implica que el flujo de control del programa sigue un orden determinado y predecible.

Estructuras de control: Se utilizan estructuras de control como secuencias, selecciones (if/else) y bucles (for, while) para controlar el flujo de ejecución del programa de manera clara y ordenada. Estas estructuras ayudan a evitar el uso excesivo de saltos incondicionales (como goto) que pueden hacer que el código sea difícil de entender y mantener.

Procedimientos y funciones: La programación estructurada fomenta la modularidad a través de la definición de procedimientos y funciones. Estos bloques de código encapsulan una tarea específica y pueden ser reutilizados en diferentes partes del programa, lo que facilita la comprensión, la depuración y el mantenimiento del código.

Descomposición jerárquica: Los programas estructurados se dividen en módulos o unidades lógicas más pequeñas, que a su vez se dividen en submódulos aún más pequeños. Este enfoque jerárquico facilita la comprensión del diseño del programa y permite abordar problemas complejos dividiéndolos en partes más manejables y fácilmente entendibles.

Ausencia de saltos incondicionales: La programación estructurada evita el uso de saltos incondicionales (como go to) en favor de estructuras de control más legibles y comprensibles. Esto ayuda a mejorar la legibilidad y mantenibilidad del código, ya que el flujo de control se determina de manera más explícita y comprensible mediante estructuras como bucles y selecciones.

7. Define cinco objetivos que tenga la reutilización de código.

La reutilización de código es una práctica fundamental en la programación que implica utilizar componentes existentes en diferentes partes de un sistema de software en lugar de volver a escribir el código desde cero. Algunos de los objetivos de la reutilización de código incluyen:

Eficiencia en el desarrollo: La reutilización de código permite a los desarrolladores aprovechar soluciones probadas y optimizadas en lugar de tener que desarrollar nuevas funcionalidades desde cero. Esto reduce el tiempo necesario para crear software y acelera el proceso de desarrollo.

Mantenibilidad: Al utilizar componentes de código reutilizable, se puede mantener un código más limpio y modular. Esto facilita la identificación y corrección de errores, así como la implementación de actualizaciones y mejoras en el software, ya que los cambios solo necesitan hacerse en un lugar en lugar de múltiples ubicaciones dispersas.

Consistencia: La reutilización de código promueve la consistencia en todo el sistema de software, ya que los mismos componentes se utilizan en diferentes partes de la aplicación. Esto ayuda a mantener un estilo de codificación coherente y reduce la posibilidad de errores causados por discrepancias en la implementación.

Facilita la colaboración: Al utilizar bibliotecas y componentes reutilizables, los equipos de desarrollo pueden colaborar más fácilmente, ya que no necesitan reimplementar funcionalidades comunes. Esto fomenta una mayor eficiencia y cohesión dentro del equipo.

Promueve las mejores prácticas: Al compartir y reutilizar código, los desarrolladores pueden beneficiarse de las mejores prácticas establecidas por otros miembros de la comunidad de desarrollo. Esto puede incluir la implementación de algoritmos eficientes, el diseño de interfaces intuitivas o la adopción de patrones de diseño probados. La reutilización de código fomenta el aprendizaje y la adopción de estas prácticas en diferentes contextos de desarrollo.

Algunas Referencias

https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/

https://developer.mozilla.org/es/docs/Glossary/Array#:~:text=Un%20arreglo%20(matriz)%20es%20una,valor%20(por%20cada%20variable).

https://keepcoding.io/blog/que-es-estructura-secuencial-programacion/