1. ¿Qué son los modificadores de acceso a atributos y métodos?

Los modificadores de acceso son palabras clave que determinan la visibilidad y el acceso a los atributos y métodos de una clase en C++. Controlan cómo y dónde se puede acceder a los miembros de la clase.

**Diferentes modificadores de acceso en C++:**

**public**: Los miembros son accesibles desde cualquier parte del programa.

**private:** Los miembros son accesibles solo desde dentro de la propia clase y no desde fuera.

**protected**: Los miembros son accesibles en la propia clase y en las clases derivadas, pero no desde fuera.

**Cómo se utilizan**: Se utilizan para encapsular el estado y el comportamiento de una clase, protegiendo los datos sensibles y asegurando que la interacción con la clase se realice de manera controlada.

2. ¿Qué es un constructor de clase?

Un constructor de clase es una función especial que se ejecuta automáticamente cada vez que se crea un objeto de esa clase.

**Propósito**: Su finalidad es inicializar los objetos, asignando valores a los atributos de la clase o realizando cualquier configuración que se requiera al crear el objeto.

**Cómo se define en C++:** Se define con el mismo nombre que la clase y no tiene tipo de retorno, por ejemplo:

class MiClase {

public:

MiClase() {

}

};

3. ¿Qué es un destructor de clase?

Un destructor de clase es una función especial que se llama automáticamente cuando un objeto de la clase se destruye.

**Propósito**: Su objetivo es liberar recursos, como memoria dinámica, y realizar limpiezas necesarias antes de que el objeto se elimine.

**Cómo se define en C++:** Se define con el mismo nombre que la clase precedido por un tilde (~), por ejemplo:

class MiClase {

public:

~MiClase() {

}

};

4. ¿Qué es la parte private y public de una clase en C++?

En C++, las partes `private` y `public` de una clase son secciones que determinan la visibilidad de los miembros de la clase.

**Diferencias:**

**private**: Los miembros no son accesibles desde fuera de la clase, pudiendo ser accedidos solo desde métodos de la misma clase.

**public:** Los miembros son accesibles desde cualquier parte del programa.

**Cómo se utilizan**: Se utilizan para prototipar qué datos y métodos pueden ser accesibles por otros objetos o funciones, protegiendo así la integridad del objeto.

5. ¿Qué es la sobrecarga de métodos?

La sobrecarga de métodos es una característica que permite a múltiples funciones tener el mismo nombre pero diferentes parámetros (tipos o cantidad).

**Propósito**: Facilitar la creación de funciones con el mismo propósito pero que operan en diferentes tipos de datos o en diferentes contextos.

**Cómo se implementa en C++:**

class Ejemplo {

public:

void mostrar(int n) {

}

void mostrar(double d) {

Mostrar un doble

}

};

6. ¿Qué es la colaboración de clases en C++?

La colaboración de clases implica que varias clases trabajan juntas para realizar una funcionalidad o tarea específica.

**Propósito**: Fomentar la reutilización de código y la separación de preocupaciones, permitiendo que cada clase gestione su propia lógica mientras contribuye a un comportamiento global.

**Cómo se implementa en C++:** Se implementa utilizando composiciones o agregaciones, donde una clase contiene instancias de otras clases y las utiliza para realizar operaciones. Por ejemplo:

class Motor {

public:

void arrancar() {}

};

class Coche {

private:

Motor motor;

public:

void conducir() {

motor.arrancar();

}

};

7. ¿Qué es la herencia?

La herencia es un mecanismo de programación orientada a objetos que permite crear una nueva clase a partir de una clase existente, reutilizando sus atributos y métodos.

**tipos de herencia:**

- Herencia simple

- Herencia múltiple

- Herencia jerárquica

- Herencia multinivel

**Ventajas:**

- Reutilización de código

- Menor duplicidad

- Facilita la creación de jerarquías de clases

**Desventajas:**

- Complejidad en el diseño

- Problemas con la herencia múltiple (ambigüedad)

**Cómo se implementa en C++:**

class Base {

public:

void mostrar() {}

};

class Derivada : public Base {

public:

void otraFuncion() {}

};

8. ¿Cuál es la diferencia entre la colaboración y la herencia en programación orientada a objetos?

La colaboración se refiere a utilizar instancias de otras clases como componentes, mientras que la herencia implica extender el comportamiento de una clase base en una clase derivada.

**Diferencias:**

- La colaboración utiliza la \_composición\_ e involucra instancias de varias clases trabajando juntas.

- La herencia crea una relación \_is-a\_ (es-un), donde la clase derivada es una versión especializada de la clase base.