1. Modificadores de acceso a atributos y métodos

Los modificadores de acceso en C++ controlan la visibilidad y el nivel de acceso de los atributos y métodos de una clase. Los diferentes modificadores disponibles son public, private y protected. Public permite que los miembros sean accesibles desde cualquier lugar. Private restringe el acceso solo a la propia clase, mientras que protected permite el acceso desde la clase y sus clases derivadas. Estos modificadores ayudan a proteger los datos y asegurar la correcta interacción entre los objetos.

2. Constructor de clase

Un constructor es un método especial utilizado para inicializar los objetos de una clase. Su propósito principal es asignar valores iniciales a los atributos o realizar cualquier configuración necesaria al crear un objeto. En C++, un constructor se define con el mismo nombre que la clase, sin especificar un tipo de retorno. Por ejemplo:

```
class MiClase {
public:
    MiClase() {
        // Código de inicialización
    }
};
```

3. Destructor de clase

Un destructor es un método especial que se utiliza para liberar los recursos cuando un objeto deja de existir. Su propósito es liberar memoria o cerrar archivos u otros recursos utilizados por el objeto antes de que sea destruido. En C++, un destructor se define utilizando el mismo nombre que la clase, precedido por un tilde (~). Ejemplo:

```
class MiClase {
public:
    ~MiClase() {
        // Código de limpieza
    }
};
```

4. Parte private y public de una clase en C++

La parte public de una clase incluye los atributos y métodos que pueden ser accedidos desde cualquier parte del programa, mientras que la parte private contiene miembros que solo pueden ser accedidos desde dentro de la clase. La principal diferencia entre ambas partes es el nivel de acceso, lo que permite controlar la visibilidad y modificar la encapsulación. Los modificadores de acceso como public y private se utilizan para definir qué partes de una clase son accesibles desde fuera y cuáles no.

5. Sobrecarga de métodos

La sobrecarga de métodos es la capacidad de definir múltiples funciones o métodos con el mismo nombre pero con diferentes parámetros. Su propósito es permitir el uso de un mismo nombre de función para realizar acciones similares con distintos tipos o números de argumentos. En C++, la sobrecarga se implementa definiendo métodos con el mismo nombre pero con diferentes firmas de parámetros. Ejemplo:

```
class MiClase {
public:
    void mostrar(int a) {
        // Código
    }
    void mostrar(double b) {
        // Código
    }
};
```

6. Colaboración de clases en C++

La colaboración de clases se refiere a la interacción entre diferentes clases para resolver un problema o completar una tarea. Su propósito es fomentar la reutilización de código y permitir que las clases trabajen juntas de manera eficiente. Se implementa mediante la creación de objetos de una clase dentro de otra o mediante la definición de relaciones entre clases, como la agregación o la composición.

7. Herencia

La herencia es un mecanismo que permite que una clase herede atributos y métodos de otra clase, llamada clase base. Existen diferentes tipos de herencia como simple, múltiple y jerárquica. La principal ventaja de la herencia es la reutilización de código, mientras que una desventaja puede ser la complejidad que añade a los programas grandes. En C++, se implementa utilizando la palabra clave : public para indicar que una clase deriva de otra:

```
class ClaseBase {
public:
    void metodo() {}
};

class ClaseDerivada : public ClaseBase {
    // Métodos y atributos adicionales
};
```

8. Diferencia entre colaboración y herencia

La colaboración implica que dos o más clases interactúan entre sí, generalmente creando instancias una de otra o utilizando sus métodos, mientras que la herencia se refiere a una relación jerárquica en la que una clase nueva se crea a partir de una existente, heredando sus características. La colaboración es horizontal (entre iguales), mientras que la herencia es vertical (entre una clase base y sus derivadas).