Objects and Classes in C++

In C++, **objects** and **classes** are key concepts in Object-Oriented Programming (OOP). They help organize code into small, reusable parts, making it easier to manage and understand.

What is a Class?

A **class** is like a blueprint used to create objects. It defines the properties (data) and actions (functions) that an object will have. For example, if you want to create a program to manage cars, you can make a Car class to represent their common features like brand, speed, and behavior.

Structure of a Class

A class contains three main parts:

1. **Access Specifiers**: These control which parts of a class can be accessed:
    - public: Can be accessed from anywhere.
    - private: Can only be accessed inside the class.
    - protected: Can be accessed inside the class and by derived classes.
2. **Data Members**: Variables that hold the data (e.g., brand, speed).
3. **Member Functions**: Functions that define the actions of the class (e.g., displayDetails).

Here's an example:

```
#include <iostream>

using namespace std;


class Car {

private:

   string brand;

   int speed;


public:

   // Constructor

   Car(string b, int s) : brand(b), speed(s) {}


   // Function to show details
```

```
    void displayDetails() {
        cout << "Brand: " << brand << ", Speed: " << speed << " km/h" << endl;
    }

    // Function to update speed
    void setSpeed(int s) {
        speed = s;
    }

    // Function to get speed
    int getSpeed() {
        return speed;
    }
};

int main() {
    Car car1("Toyota", 120); // Create an object
    car1.displayDetails();

    car1.setSpeed(150); // Update speed
    cout << "Updated Speed: " << car1.getSpeed() << " km/h" << endl;

    return 0;
}
```

What is an Object?

An **object** is a real-world instance of a class. For example, if Car is the blueprint, then car1 is an object of that class. Each object has its own copy of the data (like brand and speed) and can call the class functions.

How to Create an Object

You can create an object by declaring it with the class name:

Car car1("Ford", 100); // car1 is an object of class Car

You can also create multiple objects with the same class, each holding different data:

Car car2("Honda", 140); // car2 is a different object

Constructors and Destructors

Constructors

A **constructor** is a special function that initializes an object when it is created. It has the same name as the class and doesn't have a return type. For example:

Car(string b, int s) : brand(b), speed(s) {}

Destructors

A **destructor** is a special function that is called automatically when an object is destroyed. It is used to free up resources. It starts with a ~ symbol:

```
~Car() {
    // Cleanup code here
}
```

Key Features of Classes and Objects

1. **Encapsulation**: Groups data and functions into a single unit and hides private details from outside access.
2. **Abstraction**: Simplifies complex systems by exposing only what is necessary.
3. **Inheritance**: Allows one class to inherit properties and behaviors from another, promoting code reuse.
4. **Polymorphism**: Lets objects behave differently depending on their type, like overriding functions.

Advantages of Classes and Objects

1. **Modularity**: Code is divided into smaller, manageable parts.
2. **Reusability**: Classes can be reused in different programs.
3. **Data Security**: Private data is protected from unauthorized access.
4. **Scalability**: Adding new features is easier without affecting existing code.
