

C++ Syllabus Theory

Class in C++

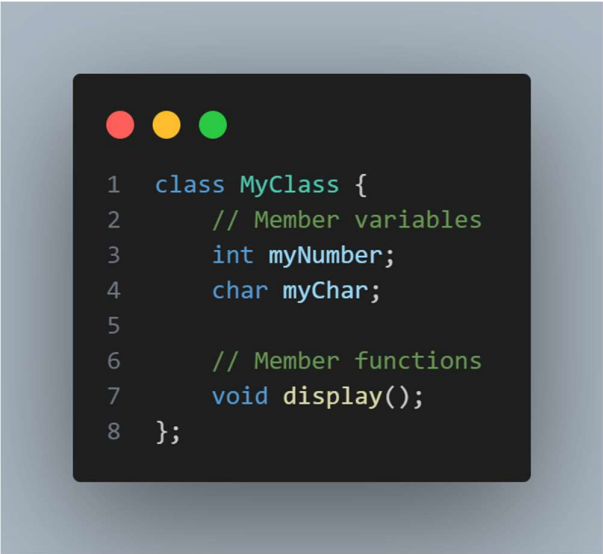
In C++, a class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables), and implementations of behavior (member functions or methods). A class is a user-defined type that serves as the basis for object-oriented programming in C++.

Key Concepts of a Class

1. **Class Declaration:** Defines the structure and behavior of a class.
2. **Member Variables (Attributes):** Data stored in an object.
3. **Member Functions (Methods):** Functions that operate on the data members of the class.
4. **Access Specifiers:** Control the access levels for members (public, private, and protected).
5. **Object:** An instance of a class.

Class Declaration

A class is declared using the class keyword followed by the class name and a body enclosed in curly braces {}. Here's a basic example:



```
1 class MyClass {
2     // Member variables
3     int myNumber;
4     char myChar;
5
6     // Member functions
7     void display();
8 };
```

Member Variables :

Member variables are variables defined inside a class. They hold the data specific to the object created from the class. In the above example, myNumber and myString are member variables.

Member Functions :

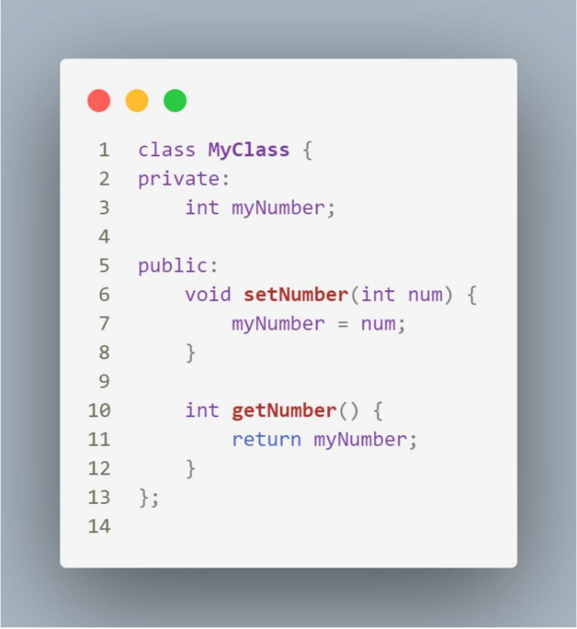
Member functions are functions defined inside a class. They operate on the member variables of the class. In the above example, display is a member function.

Access Specifiers :

Access specifiers define how the members (variables and functions) of the class can be accessed.

There are three access specifiers:

1. **public:** Members are accessible from outside the class.
2. **private:** Members are accessible only from within the class.
3. **protected:** Members are accessible within the class and by derived class(es).



```
1  class MyClass {
2  private:
3      int myNumber;
4
5  public:
6      void setNumber(int num) {
7          myNumber = num;
8      }
9
10     int getNumber() {
11         return myNumber;
12     }
13 };
14
```

Creating Objects :

An object is an instance of a class. Once a class is defined, you can create objects of that **class type**.

Here's how to create an object of MyClass:

Constructors and Destructors

Constructors are special member functions that are called when an object is instantiated. They are used to initialize objects.

Destructors are special member functions that are called when an object is destroyed. They are used for cleanup.

Example:

```
class MyClass {
private:
    int myNumber;

public:
    // Constructor
    MyClass(int num) : myNumber(num) {
        std::cout << "Constructor called, number: " << myNumber << std::endl;
    }

    // Destructor
    ~MyClass() {
        std::cout << "Destructor called, number: " << myNumber << std::endl;
    }

    void display() {
        std::cout << "Number: " << myNumber << std::endl;
    }
};
```

How to use ?

```
int main() {
    MyClass obj(42);
    obj.display();
    return 0;
}
```

Example: Full Class with Various Members

```
#include <iostream>
#include <string>
```

```
class Car {
private:
    std::string brand;
    int year;
    double mileage;

public:
    // Constructor
    Car(std::string br, int yr, double mi) : brand(br), year(yr), mileage(mi) {}

    // Member function to display car details
    void displayDetails() {
        std::cout << "Brand: " << brand << ", Year: " << year << ", Mileage: " << mileage << std::endl;
    }

    // Accessor for mileage
```

```

double getMileage() {
    return mileage;
}

// Mutator for mileage
void setMileage(double mi) {
    mileage = mi;
}

};

int main() {
    Car car1("Toyota", 2015, 55000);
    car1.displayDetails();

    car1.setMileage(60000);
    std::cout << "Updated Mileage: " << car1.getMileage() << std::endl;

    return 0;
}

```

- **Function Declaration:** Specifies the function's signature.
- **Function Definition:** Contains the actual implementation.
- **Function Call:** Invokes the function to perform its task.
- **Return Type:** The type of value the function returns.
- **Parameters:** Inputs to the function, passed by value or reference.
- **Overloading:** Defining multiple functions with the same name but different parameters.
- **Inline Functions:** Functions expanded inline to reduce overhead.
- **Recursion:** A function that calls itself.