What is Member Function in C++?

Member functions are functions and operators that are declared as a member of a class. It is declared inside the class in any of the visibility modes i.e. public and private, and it can access all the data members of the class.

Member functions, often referred to as methods, are functions that belong to a specific class in C++. They operate on the data members of the same class, promoting encapsulation - a vital characteristic of object-oriented programming. This ensures that data and the functions acting upon that data are housed together, offering higher security and data integrity.

Syntax of Member function

```
class MyClass {
private:
  int myVar;

public:
  void setMyVar(int a) {
    myVar = a;
  }
 int getMyVar() {
    return myVar;
  }
};
```

In the above example, **setMyVar** and **getMyVar** are member functions of MyClass.

Member Function inside the Class

A member function inside a class in C++ is a function declared and defined within the class definition. It has access to the class's private and protected members, allowing it to operate on the class's data directly. These functions are typically declared using the class's scope resolution operator (::).

```
#include <iostream>
using namespace std;
```

```
class MyClass {
public:
    // Member function inside the class
    void memberFunctionInside() {
        cout << "Inside memberFunctionInside()" << endl;
    }
};

int main() {
    MyClass obj;

    // Calling member function inside the class
    obj.memberFunctionInside();

    return 0;
}
```

Output:

```
Inside memberFunctionInside()
```

Member Function outside the Class

A member function outside the class in C++ is a function declared inside the class but defined outside of it. It is declared using the class's name followed by the scope resolution operator (::) and the function name. Member functions defined outside the class have access to the class's public members but not its private or protected members.

C++

```
#include <iostream>
using namespace std;

class MyClass {
public:
    // Declaration of member function outside the class
    void memberFunctionOutside();
};

// Definition of member function outside the class
void MyClass::memberFunctionOutside() {
    cout << "Inside memberFunctionOutside()" << endl;
}

int main() {
    MyClass obj;

    // Calling member function outside the class
    obj.memberFunctionOutside();

    return 0;
}
```

Output:

```
Inside memberFunctionOutside()
```

Types of Member Functions in C++

Member functions can be broadly categorized into two types:

1- Static Member Function

2- Non Static Member Function

Non-static Member Functions

These are the standard member functions which are invoked using object instances.

Static Member Functions

These functions belong to the class rather than individual objects. They cannot access non-static data members or call non-static member functions directly.

Syntax of Static Member Functions

```
class MyClass {
public:
  static void staticFunction() {
    // Can only access static members
  }

  void nonStaticFunction() {
    // Can access both static and non-static members
  }
};
```

Static Member in C++

Static members of a class are not associated with the objects of the class. Just like a static variable once declared is allocated with memory that can’t be changed every object points to the same memory.

Static Objects in C++

An object becomes static when a ***static*** keyword is used in its declaration. Static objects are initialized only once and live until the program terminates

Static objects are always destroyed at the end of the program whether their scope is local or global. This property allows them to retain their value between multiple function calls and we can even use a pointer to them to access them out of scope.

Const member functions in C++

Constant member functions are those functions that are denied permission to change the values of the data members of their class. To make a member function constant, the keyword const is appended to the function prototype and also to the function definition header.

Like member functions and member function arguments, the objects of a class can also be declared as const. An object declared as const cannot be modified and hence, can invoke only const member functions as these functions ensure not to modify the object. A const object can be created by prefixing the const keyword to the object declaration. Any attempt to change the data member of const objects results in a compile-time error.

Syntax

The const member function can be defined in three ways:

1. For function declaration within a class.

```
return_type function_name() const;
```

Example:

```
int get_data() const;
```

2. For function definition within the class declaration.

```
return_type function_name () const

{

    //function body

}
```

Example:

```
int get_data() const

{

            //function body

}
```

3. For function definition outside the class.

```
return_type class_name::function_name() const

{

       //function body

}
```

Example:

```
int Demo :: get_data() const

{

}
```

Important Points

- When a function is declared as const, it can be called on any type of object, const object as well as non-const objects.
- Whenever an object is declared as const, it needs to be initialized at the time of declaration. however, the object initialization while declaring is possible only with the help of constructors.
- A function becomes const when the const keyword is used in the function's declaration. The idea of const functions is not to allow them to modify the object on which they are called.

- It is recommended practice to make as many functions const as possible so that accidental changes to objects are avoided.