Objects and Classes

Sakshi Khare

K. J Somaiya Polytechnic, Department of Diploma in Computer Engg.

Abstract: ***In this lesson, we will understand what is a class and object. With that, we will also see some examples to create classes and objects. In C++, a class is a template for an object, whereas an object is an instance of a class.***

Keywords: Objects and classes, member functions, data members

1. Introduction

C++ is an object - oriented language that is used to model real - world entities into programs. All object - oriented programming languages achieve this task using classes and objects. Classes act as a blueprint to create objects with similar properties. The concept of classes and objects in C++ is the fundamental idea around which the object - oriented approach revolves around. It enhances the program's efficiency by reducing code redundancy and debugging time.

Now, you will understand the concept of the class and object in C++ with the help of a real - life example. Suppose you have a small library. In a library, all books have some common properties like book_name, author_name, and genre. Now imagine you want to create a catalog of all the books in your collection. Instead of creating separate classes for every book you own, you can create a Book class that serves as a template for all the books in your library.

What are Classes in C++?

A class is a template or a blueprint that binds the properties and functions of an entity. You can put all the entities or objects having similar attributes under a single roof, known as a class. Classes further implement the core concepts like encapsulation, data hiding, and abstraction. In C++, a class acts as a data type that can have multiple objects or instances of the class type.

Consider an example of a railway station having several trains. A train has some characteristics like train_no, destination, train_type, arrival_time, and departure_time. And its associated operations are arrival and departure. You can define a class can for a train as follows:

```
class train
{
 // characteristics
 int train_no;
 char destination;
 char train_type;
 int arrival_time;
 int departure_time;
 // functions
 int arrival (delayed_time)
 {
 arrival_time += delayed_time;
 return arr_time;
 }
 int departure (delayed_time)
 {
 departure_time += delayed_time;
 return departure_time;
 }
}
```

The above class declaration contains properties of the class, train_no, destination, train_type, arrival_time, and departure_time as the data members. You can define the operations, arrival, and departure as the member functions of the class.

Syntax to Declare a Class in C++:

```
class class_name
{
 // class definition
 access_specifier: // public, protected, or private
 data_member1; // data members
 data_member2;
 func1 () {} // member functions
 func2 () {}
};
```

What are Objects in C++?

Objects in C++ are analogous to real - world entities. There are objects everywhere around you, like trees, birds, chairs, tables, dogs, cars, and the list can go on. There are some properties and functions associated with these objects. Similarly, C++ also includes the concept of objects. When you define a class, it contains all the information about the objects of the class type. Once it defines the class, it can create similar objects sharing that information, with the class name being the type specifier.

Consider the example of a railway station discussed in the previous section. After defining the class train, you can create similar objects for this class. For example, train_A and train_B. You can create the objects for the class defined above in the following way:

```
train train_A, train_B;
```

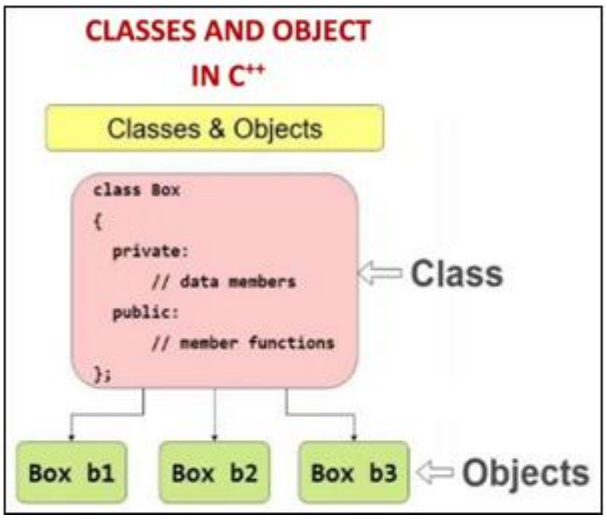
The syntax to create objects in C++:

```
class_nameobject_name;
```

The object object_name once created, can be used to access the data members and member functions of the class class_name using the dot operator in the following way:

```
obj. data_member = 10; // accessing data member
obj. func (); // accessing member function
```

Significance of Class and Object in C++
The concept of class and object in C++ makes it possible to incorporate real - life analogy to programming. It gives the data the highest priority using classes. The following features prove the significance of class and object in C++:



- Data hiding: A class prevents the access of the data from the outside world using access specifiers. It can set permissions to restrict the access of the data.
- Code Reusability: You can reduce code redundancy by using reusable code with the help of inheritance. Other classes can inherit similar functionalities and properties, which makes the code clean.
- Data binding: The data elements and their associated functionalities are bound under one hood, providing more security to the data.
- Flexibility: You can use a class in many forms using the concept of polymorphism. This makes a program flexible and increases its extensibility.

Member Functions in Classes
The member functions are like the conventional functions. It defines these methods inside a class and has direct access to all the data members of its class. When you define a member function, it only creates and shares one instance of that function by all the instances of that class. The following syntax can be used to declare a member function inside a class:

```
class class_name
{
access_specifier:
 return_typemember_function_name (data_typearg);
};
```

It specifies the access specifier before declaring a member function. It also specifies the return type and data type in the same way in which it declares a usual function.

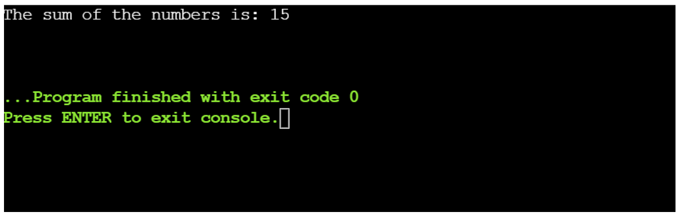
Method Definition Outside and Inside of a Class

The following two ways can define a method or member functions of a class:
1) Inside class definition
2) Outside class definition

The function body remains the same in both approaches to define a member function. The difference lies only in the function's header. Now, have a deeper understanding of these approaches.

Inside Class Definition
This approach of defining a member function is generally preferred for small functions. It defines a member function inside a class in the same familiar way as it defines a conventional function. It specifies the return type of the function, followed by the function name, and it provides arguments in the function header. Then it provides the function body to define the complete function. The member functions that are defined inside a class are automatically inline. The following example illustrates defining a member function inside a class.

```
#include <iostream>
using namespace std;
// define a class
class my_class
{
public:
 // inside class definition of the function
 void sum (int num1, int num2) // function header
 {
 cout<< "The sum of the numbers is: "; // function body
 cout<< (num1 + num2) << "\n\n";
 }
};
int main ()
{
 // create an object of the class
 my_classobj;
 // call the member function
 obj. sum (5, 10);
 return 0;
}
```



```
The sum of the numbers is: 15


...Program finished with exit code 0
Press ENTER to exit console.
```

In the above example, you define the function sum () inside the class my_class. The function is automatically an inline function. Whenever you call this function by an object of its class, it inserts the code of the function’s body there, which reduces the execution time of the program. Here, the statement obj. sum (5, 10) is replaced by the body of the function sum ().

References

Wikipedia, google, w3schools etc.
Links
[1] https: //www.simplilearn. com/tutorials/cpp - tutorial/class - and - object - in - cpp
[2] www.google. com,
[3] www.w#schools. com
[4] *Books: OOPS using C++*