# **C++ Object Oriented**

The prime purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages.

The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

There are a few principle concepts that form the foundation of object-oriented programming:

## Object

This is the basic unit of object oriented programming. That is both data and function that operate on data are bundled as a unit called as object.

## Class

When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

## Abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

## Encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

## Inheritance

One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

## Polymorphism

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.

## Overloading

The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new data type, it is said to be overloaded.

# **Basic Concepts of Object Oriented Programming using C++**

Object oriented programming – As the name suggests uses [objects](http://www.geeksforgeeks.org/basic-concepts-of-object-oriented-programming-using-c/#objects) in programming. Object oriented programming aims to implement real world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operates on them so that no other part of code can access this data except that function.  
Let us learn about different characteristics of an Object Oriented Programming language:  
**Object:** Objects are basic run-time entities in an object oriented system, objects are instances of a class these are defined user defined data types.  
ex:

|  |
| --- |
| class person  {      char name[20];      int id;  public:      void getdetails(){}  };    int main()  {     person p1; //p1 is a object  } |

Object take up space in memory and have an associated address like a record in pascal or structure or union in C.

When a program is executed the objects interact by sending messages to one another.

Each object contains data and code to manipulate the data. Objects can interact without having to know details of each others data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

**Class:** Class is a blueprint of data and functions or methods. Class does not take any space.

syntax for class:

|  |
| --- |
| class class\_name  {    private:       //data members and member functions declarations    public:       //data members and member functions declarations    protected:       //data members and member functions declarations  }; |

Class is a user defined data type like structures and unions in C.

By default class variables are private but in case of structure it is public. in above example person is a class.  
  
**Encapsulation and Data abstraction:** Wrapping up(combing) of data and functions into a single unit is known as encapsulation. The data is not accessible to the outside world and only those functions which are wrapping in the class can access it. This insulation of the data from direct access by the program is called data hiding or information hiding.

Data abstraction refers to, providing only needed information to the outside world and hiding implementation details. For example, consider a class Complex with public functions as getReal() and getImag(). We may implement the class as an array of size 2 or as two variables. The advantage of abstractions is, we can change implementation at any point, users of Complex class wont’t be affected as out method interface remains same. Had our implementation be public, we would not have been able to change it.

**Inheritance:** inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. Inheritance provides re usability. This means that we can add additional features to an existing class without modifying it.

**Polymorphism:** polymorphism means ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation.  
C++ supports operator overloading and function overloading.  
Operator overloading is the process of making an operator to exhibit different behaviors in different instances is known as operator overloading.  
Function overloading is using a single function name to perform different types of tasks.  
polymorphism is extensively used in implementing inheritance.

**Dynamic Binding:** In dynamic binding, the code to be executed in response to function call is decided at runtime. C++ has [virtual functions](http://www.geeksforgeeks.org/virtual-functions-and-runtime-polymorphism-in-c-set-1-introduction/) to support this.

**Message Passing:** Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

**C++ Classes and Objects**

The main purpose of C++ programming is to add object orientation to the C programming language and classes are the central feature of C++ that supports object-oriented programming and are often called user-defined types.

A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

**C++ Class Definitions**

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword **class** followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations. For example, we defined the Box data type using the keyword **class** as follows:

class Box {

public:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

The keyword **public** determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as **private** or **protected** which we will discuss in a sub-section.

**Define C++ Objects**

A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types. Following statements declare two objects of class Box:

Box Box1; // Declare Box1 of type Box

Box Box2; // Declare Box2 of type Box

Both of the objects Box1 and Box2 will have their own copy of data members.

**Accessing the Data Members**

The public data members of objects of a class can be accessed using the direct member access operator (.). Let us try the following example to make the things clear:

#include <iostream>

using namespace std;

class Box

{

public:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

int main( ) {

Box Box1; // Declare Box1 of type Box

Box Box2; // Declare Box2 of type Box

double volume = 0.0; // Store the volume of a box here

// box 1 specification

Box1.height = 5.0;

Box1.length = 6.0;

Box1.breadth = 7.0;

// box 2 specification

Box2.height = 10.0;

Box2.length = 12.0;

Box2.breadth = 13.0;

// volume of box 1

volume = Box1.height \* Box1.length \* Box1.breadth;

cout << "Volume of Box1 : " << volume <<endl;

// volume of box 2

volume = Box2.height \* Box2.length \* Box2.breadth;

cout << "Volume of Box2 : " << volume <<endl;

return 0;

}

When the above code is compiled and executed, it produces the following result:

Volume of Box1 : 210

Volume of Box2 : 1560

It is important to note that private and protected members can not be accessed directly using direct member access operator (.). We will learn how private and protected members can be accessed.

**Classes & Objects in Detail**

So far, you have got very basic idea about C++ Classes and Objects. There are further interesting concepts related to C++ Classes and Objects which we will discuss in various sub-sections listed below:

|  |  |
| --- | --- |
| **Concept** | **Description** |
| [Class member functions](https://www.tutorialspoint.com/cplusplus/cpp_class_member_functions.htm) | A member function of a class is a function that has its definition or its prototype within the class definition like any other variable. |
| [Class access modifiers](https://www.tutorialspoint.com/cplusplus/cpp_class_access_modifiers.htm) | A class member can be defined as public, private or protected. By default members would be assumed as private. |
| [Constructor & destructor](https://www.tutorialspoint.com/cplusplus/cpp_constructor_destructor.htm) | A class constructor is a special function in a class that is called when a new object of the class is created. A destructor is also a special function which is called when created object is deleted. |
| [C++ copy constructor](https://www.tutorialspoint.com/cplusplus/cpp_copy_constructor.htm) | The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. |
| [C++ friend functions](https://www.tutorialspoint.com/cplusplus/cpp_friend_functions.htm) | A **friend** function is permitted full access to private and protected members of a class. |
| [C++ inline functions](https://www.tutorialspoint.com/cplusplus/cpp_inline_functions.htm) | With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function. |
| [The this pointer in C++](https://www.tutorialspoint.com/cplusplus/cpp_this_pointer.htm) | Every object has a special pointer **this** which points to the object itself. |
| [Pointer to C++ classes](https://www.tutorialspoint.com/cplusplus/cpp_pointer_to_class.htm) | A pointer to a class is done exactly the same way a pointer to a structure is. In fact a class is really just a structure with functions in it. |
| [Static members of a class](https://www.tutorialspoint.com/cplusplus/cpp_static_members.htm) | Both data members and function members of a class can be declared as static. |

C++ is a multi-paradigm programming language. Meaning, it supports different programming styles.

One of the popular ways to solve a programming problem is by creating objects, known as object-oriented style of programming.

C++ supports object-oriented (OO) style of programming which allows you to divide complex problems into smaller sets by creating objects.

Object is simply a collection of data and functions that act on those data.

## C++ Class

Before you create an object in C++, you need to define a class.

A class is a blueprint for the object.

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As, many houses can be made from the same description, we can create many objects from a class.

### **How to define a class in C++?**

A class is defined in C++ using keyword class followed by the name of class.

The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

class className

{

// some data

// some functions

};

### Example: Class in C++

class Test

{

private:

int data1;

float data2;

public:

void function1()

{ data1 = 2; }

float function2()

{

data2 = 3.5;

return data2;

}

};

Here, we defined a class named Test.

This class has two data members: data1 and data2 and two member functions: function1() and function2().

### **Keywords: private and public**

You may have noticed two keywords: private and public in the above example.

The private keyword makes data and functions private. Private data and functions can be accessed only from inside the same class.

The public keyword makes data and functions public. Public data and functions can be accessed out of the class.

Here, data1 and data2 are private members where as function1() and function2() are public members.

If you try to access private data from outside of the class, compiler throws error. This feature in OOP is known as data hiding.

## C++ Objects

When class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, you need to create objects.

### Syntax to Define Object in C++

className objectVariableName;

You can create objects of Test class (defined in above example) as follows:

class Test

{

private:

int data1;

float data2;

public:

void function1()

{ data1 = 2; }

float function2()

{

data2 = 3.5;

return data2;

}

};

int main()

{

Test o1, o2;

}

Here, two objects o1 and o2 of Test class are created.

In the above class Test, data1 and data2 are data members and function1() and function2() are member functions.

### How to access data member and member function in C++?

You can access the data members and member functions by using a . (dot) operator. For example,

o2.function1();

This will call the function1() function inside the Test class for objects o2.

Similarly, the data member can be accessed as:

o1.data2 = 5.5;

It is important to note that, the private members can be accessed only from inside the class.

So, you can use o2.function1(); from any function or class in the above example. However, the code o1.data2 = 5.5; should always be inside the class Test.

### Example: Object and Class in C++ Programming

// Program to illustrate the working of objects and class in C++ Programming

#include <iostream>

using namespace std;

class Test

{

private:

int data1;

float data2;

public:

void insertIntegerData(int d)

{

data1 = d;

cout << "Number: " << data1;

}

float insertFloatData()

{

cout << "\nEnter data: ";

cin >> data2;

return data2;

}

};

int main()

{

Test o1, o2;

float secondDataOfObject2;

o1.insertIntegerData(12);

secondDataOfObject2 = o2.insertFloatData();

cout << "You entered " << secondDataOfObject2;

return 0;

}

**Output**

Number: 12

Enter data: 23.3

You entered 23.3

In this program, two data members data1 and data2 and two member functions insertIntegerData() and insertFloatData() are defined under Test class.

Two objects o1 and o2 of the same class are declared.

The insertIntegerData() function is called for the o1 object using:

o1.insertIntegerData(12);

This sets the value of data1 for object o1 to 12.

Then, the insertFloatData() function for object o2 is called and the return value from the function is stored in variable secondDataOfObject2 using:

secondDataOfObject2 = o2.insertFloatData();

In this program, data2 of o1 and data1 of o2 are not used and contains garbage value.

## Polymorphism in C++

The process of representing one Form in multiple forms is known as **Polymorphism**. Here one form represent original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.

Polymorphism is derived from 2 greek words: **poly** and morphs. The word "poly" means many and **morphs** means forms. So polymorphism means many forms.

## Real life example of Polymorphism in C++

Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person have different-different behaviors.

## Type of polymorphism

* Compile time polymorphism
* Run time polymorphism

## Compile time polymorphism

In C++ programming you can achieve compile time polymorphism in two way, which is given below;

* Method overloading
* Method overriding

## Method Overloading in C++

Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as **method overloading**. In below example method "sum()" is present in Addition class with same name but with different signature or arguments.

## Example of Method Overloading in C++

#include<iostream.h>

#include<conio.h>

class Addition

{

public:

void sum(int a, int b)

{

cout<<a+b;

}

void sum(int a, int b, int c)

{

cout<<a+b+c;

}

};

void main()

{

clrscr();

Addition obj;

obj.sum(10, 20);

cout<<endl;

obj.sum(10, 20, 30);

}

## Output

30

60

## Method Overriding in C++

Define any method in both base class and derived class with same name, same parameters or signature, this concept is known as **method overriding**. In below example same method "show()" is present in both base and derived class with same name and signature.

## Example of Method Overriding in C++

#include<iostream.h>

#include<conio.h>

class Base

{

public:

void show()

{

cout<<"Base class";

}

};

class Derived:public Base

{

public:

void show()

{

cout<<"Derived Class";

}

}

int mian()

{

Base b; //Base class object

Derived d; //Derived class object

b.show(); //Early Binding Ocuurs

d.show();

getch();

}

## Output

Base class

Derived Class

## Run Time Polymorphism

In C++ Run time polymorphism can be achieve by using [virtual function](http://www.sitesbay.com/cpp/cpp-virtual-function).

# Define object, classes and instances.

## Define object, classes and instances.

**Object :**   
Object is the basic run time entity in an Object Oriented (OO) System. From a programmer’s perspective, it is a storage region with associated semantics. From a designer’s perspective, it is an identifiable component in the problem domain. Object is essentially a variable of user defined data type class.  
  
**Class :**   
User defined data type which contains data and methods to manipulate that data; is known as class. It is the fundamental packaging unit of OO technology. An object is a variable of a Class. Each object is associated with the data of type class with which it is created. Thus we can also say that class is a collection of objects of similar types. It is a user defined data type and behaves like built-in data type of the language.  
  
**Instance :**   
Object can also be called as runtime instance of a class. It is used to access data and function members while a program is running. As stated earlier, class specification provides only a template. It does not create any memory space for the object. To create memory space at runtime, we need to create instance of the class, which is essentially the class object.  
  
**Consider the following example :**

Class Student   --- Class Definition  
{  
   int rollno;  
   int marks;  
   public:   
       void show(int roll)  
       {  
          cout<< rollno: << marks;  
       }  
};  
  
void main()  
{  
   Student s;    --Object of Class  
   s.show();  
}

- Here, Student is a user defined data type - Class  
- In main(), variable of this user defined data type Student is created, which is the object of that Class.  
- This object of class is also known as instance.

# What is an instance in c++?

In C++, an instance is an object, which has its properties (description of object) and its methods (what it can do - activities of object).  
Example: A car instance has red color (property) and it can start and run (methods)...

In C++, we usually define a class then create instance of this class, so instance looks like a presentation of a class.