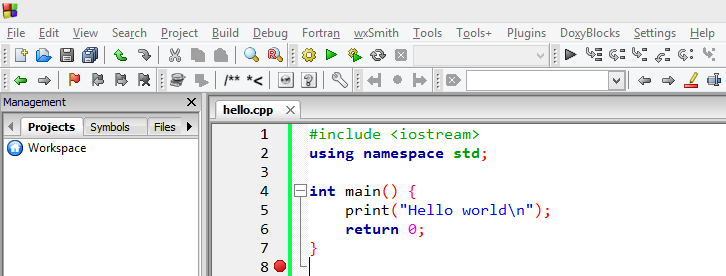
**What is C++ (programming language)?**

“C++ is a statically-typed, free-form, (usually) compiled, multi-paradigm, intermediate-level general-purpose middle-level programming language.”

In simple terms, C++ is a sophisticated, efficient and a general-purpose programming language based on C. It was developed by [Bjarne Stroustrup](http://www.stroustrup.com/" \o "Bjarne Stroustrup) in 1979.

Many of today’s operating systems, system drivers, browsers and games use C++ as their core language. This makes C++ one of the most popular languages today.

Since it is an enhanced/extended version of [C programming](https://www.programiz.com/c-programming) language, C and C++ are often denoted together as C/C++.



#include <iostream>

using namespace std;

int main()

{

cout<<"Hello World!";

return 0;

}

1. **What is using namespace std;”?**  
     
   The statement is intuitive in itself, you are “using” the “namespace” “std” in your file.  
   We use the namespace std to make it easier to reference operations included in that namespace.  
   If we hadn’t used the namespace, we’d have written **std::cout** instead of **cout**. This tells the compiler that every **cout** is actually **std::cout**.  
     
   **What’s a namespace?**  
     
   It’s a region where your code resides. It limits or expands the scope of your code to one or more files.  
     
   **Why do you use namespace?**  
     
   Like two persons can have the same name, variables and functions in C++ can have same names as well. The use of namespace is to avoid the confusion of which variables/functions you are referencing to.  
     
   **What is std?**  
     
   std is a standard namespace used in C++.
2. **Semicolon ”;”**  
     
   Ask any C++ programmer and they will tell you at least one horror story related to the semicolon ; .  
     
   The semicolon is a terminal. It terminates a statement. When missed or incorrectly used, it will cause a lot of issues.
3. **cout << “Hello World!”;**  
     
   This statement prints “Hello World!” onto the output screen.  
     
   The cout is an object of standard output stream. What this means is, it outputs/prints the data after *<<* , i.e. Hello World! into a stream (in this case, the output screen).  
     
   **What is a stream?**  
     
   Stream is basically a sequence of objects, usually bytes. It can describe files, input/output terminal, sockets, etc.  
   **What is <<?**  
     
   << is the insertion operator used to write formatted data into the stream.
4. **What is return 0;?**  
     
   This statement returns 0 ‘zero’.  
     
   This is called a return statement. It isn’t mandatory to return anything from the main() function but is rather a convention. If not return, the compiler returns a status automatically.

# C++ Classes and Objects

Object oriented programming is a way of solving complex problems by breaking them into smaller problems using objects. Before Object Oriented Programming (commonly referred as OOP), programs were written in procedural language, they were nothing but a long list of instructions. On the other hand, the OOP is all about creating objects that can interact with each other, this makes it easier to develop programs in OOP as we can understand the relationship between them.

## Object Oriented Programming(OOP)

In Object oriented programming we write programs using classes and objects utilising features of OOPs such as **abstraction**, **encapsulation**, **inheritance** and **polymorphism**

### Class and Objects

A class is like a blueprint of data member and functions and object is an instance of class. For example, lets say we have a class **Car** which has data members (variables) such as speed, weight, price and functions such as gearChange(), slowDown(), brake() etc. Now lets say I create a object of this class named FordFigo which uses these data members and functions and give them its own values. Similarly we can create as many objects as we want using the blueprint(class).

### Abstraction

[Abstraction](https://beginnersbook.com/2017/09/abstraction-in-c-with-example/) is a process of hiding irrelevant details from user. For example, When you send an sms you just type the message, select the contact and click send, the phone shows you that the message has been sent, what actually happens in background when you click send is hidden from you as it is not relevant to you.

### Encapsulation

[Encapsulation](https://beginnersbook.com/2017/09/cpp-encapsulation/) is a process of combining data and function into a single unit like capsule. This is to avoid the access of private data members from outside the class. To achieve encapsulation, we make all data members of class private and create public functions, using them we can get the values from these data members or set the value to these data members.

### Polymorphism

[Function overloading](https://beginnersbook.com/2017/08/cpp-function-overloading/) and Operator overloading are examples of polymorphism. Polymorphism is a feature using which an object behaves differently in different situation.  
In function overloading we can have more than one function with same name but different numbers, type or sequence of arguments.

### Inheritance

[Inheritance](https://beginnersbook.com/2017/08/cpp-inheritance/) is a feature using which an object of child class acquires the properties of parent class.

### Example:

class Test

{

private:

int data1;

float data2;

public:

void function1()

{ data1 = 2; }

float function2()

{

data2 = 3.5;

return data2;

}

};

int main()

{

Test o1, o2;

}

### 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();

### 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;

}**C++ Constructors**



 constructor is a special type of member function that initialises an [object](https://www.programiz.com/cpp-programming/object-class) automatically when it is created.

Compiler identifies a given member function is a constructor by its name and the return type.

Constructor has the same name as that of the class and it does not have any return type. Also, the constructor is always public.

// Cpp program to illustrate the

// concept of Constructors

#include <iostream>

using namespace std;

class construct

{

public:

    int a, b;

        // Default Constructor

    construct()

    {

        a = 10;

        b = 20;

    }

};

int main()

{

        // Default constructor called automatically

        // when the object is created

    construct c;

    cout << "a: "<< c.a << endl << "b: "<< c.b;

    return 1;

}

**Parameterized Constructors:**It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor’s body, use the parameters to initialize the object.

|  |
| --- |
| // CPP program to illustrate  // parameterized constructors  #include<iostream>  using namespace std;    class Point  {      private:          int x, y;      public:          // Parameterized Constructor          Point(int x1, int y1)          {              x = x1;              y = y1;          }            int getX()          {              return x;          }          int getY()          {              return y;          }      };    int main()  {      // Constructor called      Point p1(10, 15);        // Access values assigned by constructor      cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();        return 0;  } |