Week 1 : Introduction to Object Oriented Programming

Prepared by , Mahrukh Khan

Objective- Computer Programming Course

The course is designed to

- Provide the students with solid foundations in the basic concepts of Object Oriented Paradigm.
- Teach students how to develop application using OO Techniques and how debugging and code optimization impacts the performance of the programs.
- Understand advanced features of C++ ,Exception Handling,Stream I/O, Templates and Operator Overloading
- to design larger programs by structuring them into multiple classes, with a variety of relationships between those classes, such as association, composition, and inheritance.

Books

Textbooks:

1. “Problem Solving with C++”, 9th edition, Walter Savitch,2015
2. “C++ How to Program” ,Deitel & Deitel

Reference Books:

1. The C++ Programming Language by Bjarne Stroustrup
2. Object Oriented Software Engineering by Jacobson

Assessments/Grading

- Assignments
- Midterm Exams
- Quizzes
- Final Exam
- Project

Machine Language

- Comprised of 1s and 0s
- The “native” language of a computer
- Difficult to program – one misplaced 1 or 0 will cause the program to fail.
- Example of code:

```
1110100010101   111010101110
10111010110100  10100011110111
```

Assembly Language

- Assembly languages are a step towards easier programming.
- Assembly languages are comprised of a set of elemental commands which are tied to a specific processor.
- Assembly language code needs to be translated to machine language before the computer processes it.
- Example:

```
ADD  1001010, 1011010
```

High Level Languages

- High-level languages represent a giant leap towards easier programming.
- The syntax of HL languages is similar to English.
- Historically, we divide HL languages into two groups:

  –Sequential languages

  –Procedural languages

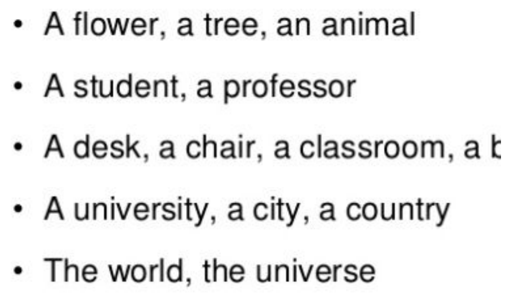
  –Object-Oriented languages (OOP)

Procedural Languages

- Early high-level languages are typically called procedural languages.
- Procedural languages are characterized by sequential sets of linear commands. The focus of such languages is on *procedures*.
- Examples include C, COBOL, Fortran, LISP, Perl, HTML, VBScript

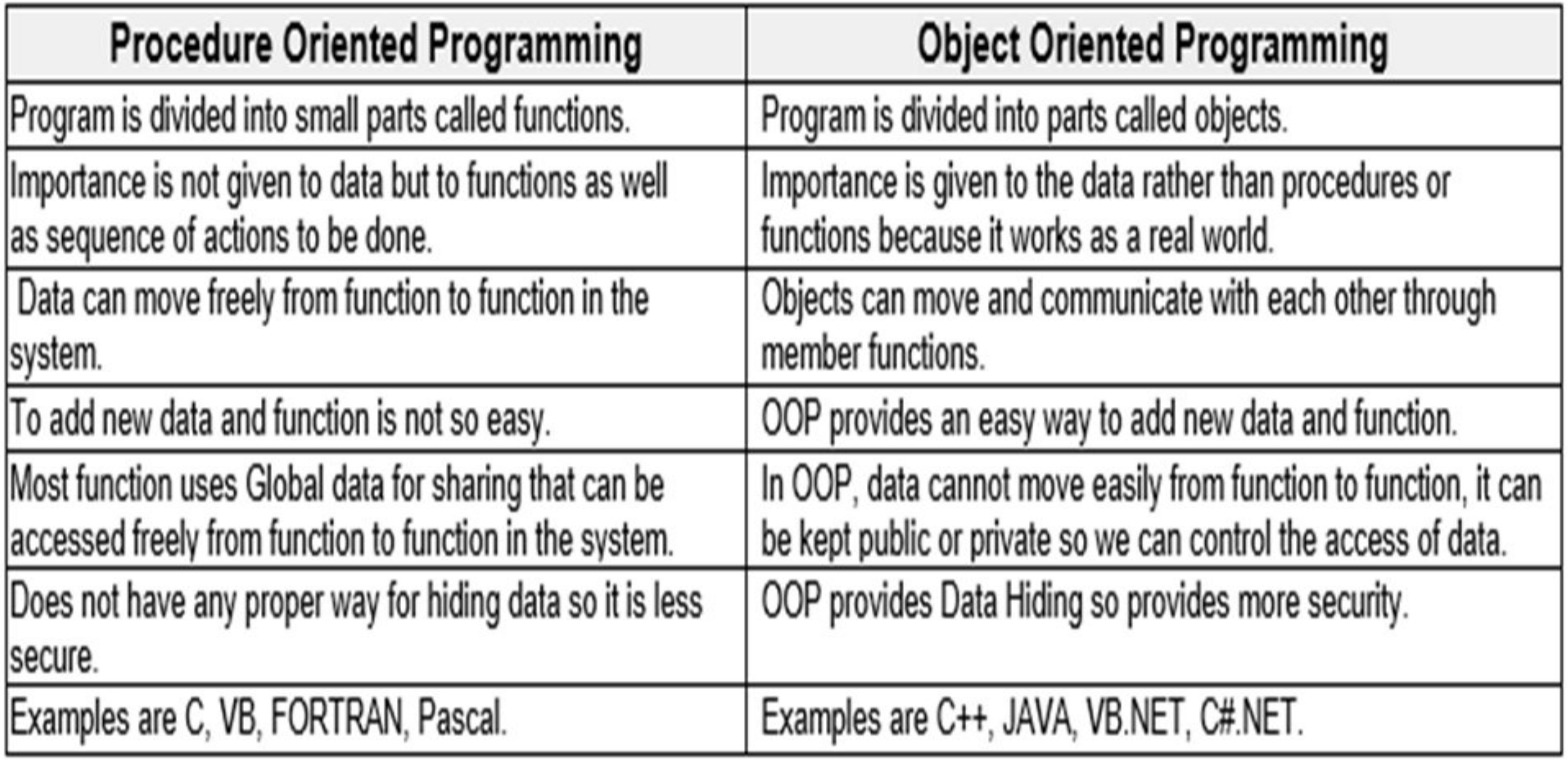
Object Oriented Languages

- Real world can be accurately described as a collection of objects that interact.
- The focus of OOP languages is not on structure, but on *modeling data*.
- Programmers code using “blueprints” of data models called *classes*.
- Examples of OOP languages include C++, Visual Basic.NET and Java.
- Almost anything in the world can be represented as a class.



- A flower, a tree, an animal
- A student, a professor
- A desk, a chair, a classroom, a b
- A university, a city, a country
- The world, the universe

Procedural Language v/s Object Oriented Language



Procedure Oriented Programming	Object Oriented Programming
Program is divided into small parts called functions.	Program is divided into parts called objects.
Importance is not given to data but to functions as well as sequence of actions to be done.	Importance is given to the data rather than procedures or functions because it works as a real world.
Data can move freely from function to function in the system.	Objects can move and communicate with each other through member functions.
To add new data and function is not so easy.	OOP provides an easy way to add new data and function.
Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data cannot move easily from function to function, it can be kept public or private so we can control the access of data.
Does not have any proper way for hiding data so it is less secure.	OOP provides Data Hiding so provides more security.
Examples are C, VB, FORTRAN, Pascal.	Examples are C++, JAVA, VB.NET, C#.NET.

Why OOP?

Before Object Oriented Programming programs were viewed as procedures that accepted data and produced an output. There was little emphasis given on the data that went into those programs.

PROBLEMS WITH PROCEDURAL LANGUAGES

- Functions have unrestricted access to global data
- Unrelated Functions and data

Example

Imagine a personal Address book with some data stored about your friends.

- Name
- Address
- Telephone no.

List three things you may do to this book.

Next identify someone else who may use an identical address book for some purpose other than storing a list of friends.

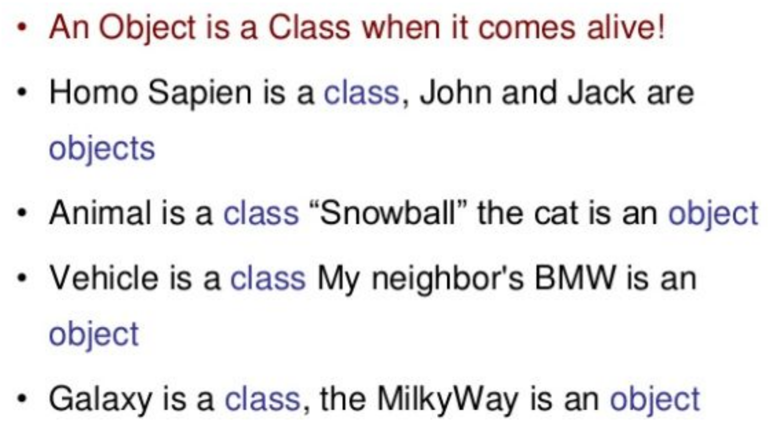
Example

- Find out details of a friend
- Add an address(friend details) in address book.
- Delete an address.

By creating a software component to hold the data and its operations, we can re-use this in another software system. (E.g. it could be a business manager to store and find details of customers or it could be used by a librarian to store and find users of the library.

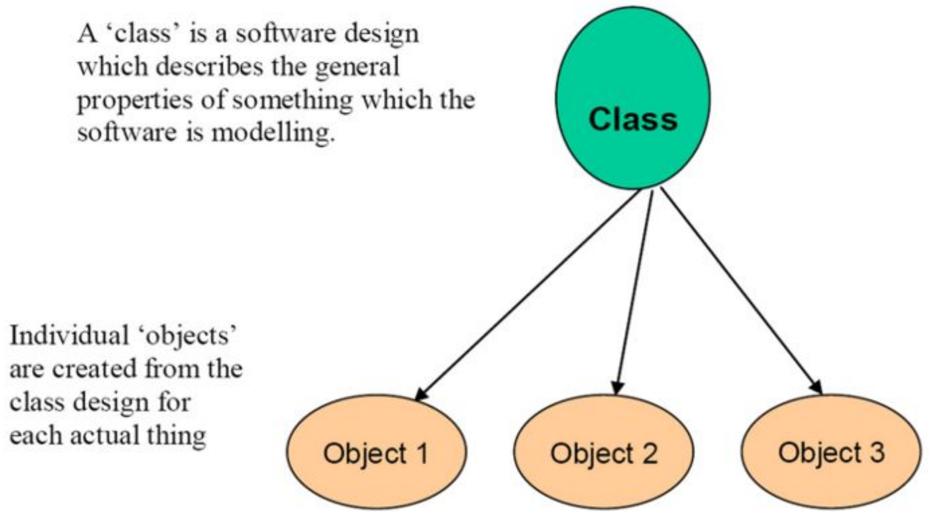
Class and Object

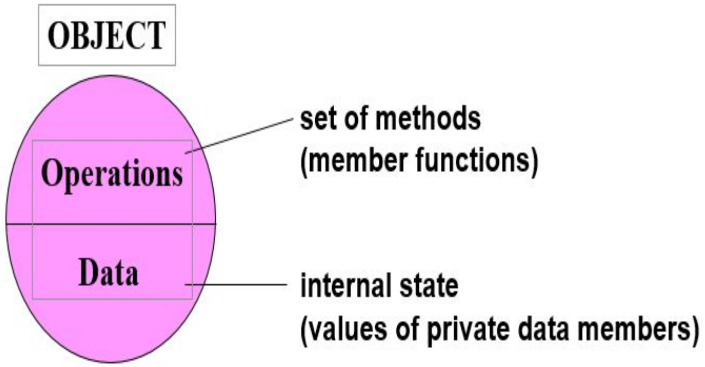
- A **class** is a data type that allows programmers to create objects. A class provides a definition for an object, describing an object’s attributes (data) and methods (operations).
- An **object** is an *instance* of a class. With one class, you can have as many objects as required.



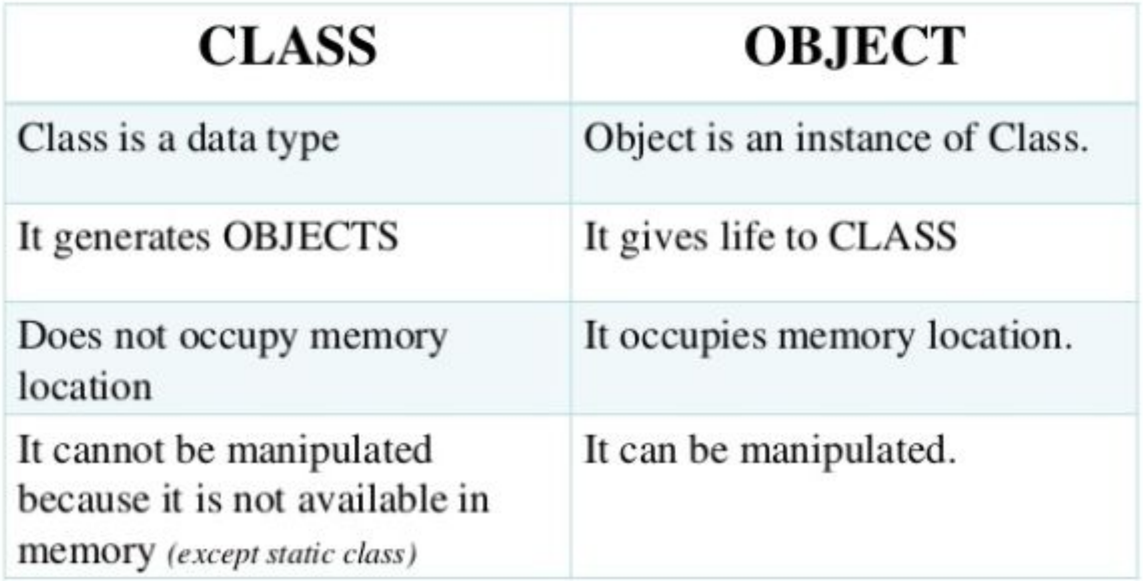
- An Object is a Class when it comes alive!
- Homo Sapien is a class, John and Jack are objects
- Animal is a class “Snowball” the cat is an object
- Vehicle is a class My neighbor's BMW is an object
- Galaxy is a class, the MilkyWay is an object

Class and Object


A 'class' is a software design which describes the general properties of something which the software is modelling.
Class
Individual 'objects' are created from the class design for each actual thing
Object 1
Object 2
Object 3


OBJECT
Operations
Data
set of methods (member functions)
internal state (values of private data members)

Class and Object

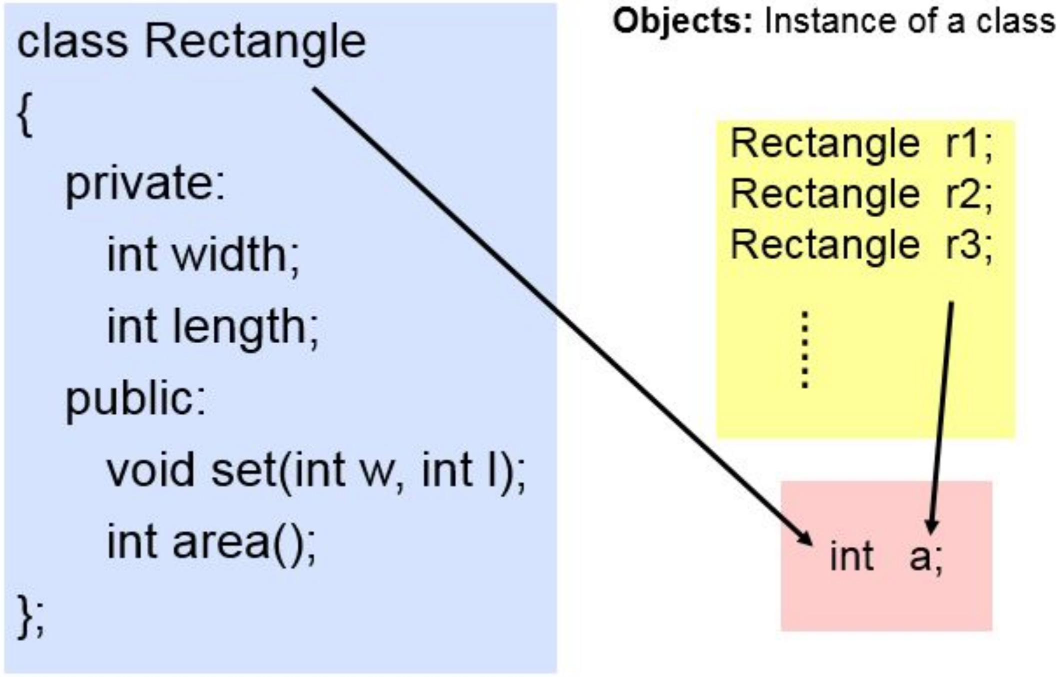


CLASS	OBJECT
Class is a data type	Object is an instance of Class.
It generates OBJECTS	It gives life to CLASS
Does not occupy memory location	It occupies memory location.
It cannot be manipulated because it is not available in memory *(except static class)*	It can be manipulated.

Class and Object

- Every object belongs to (is an instance of) a class
- An object may have fields, or variables
  - The class describes those fields
- An object may have methods
  - The class describes those methods
- A class is like a template, or cookie cutter
  - You use the class’s constructor to make objects

Example



```
class Rectangle
{
   private:
      int width;
      int length;
   public:
      void set(int w, int l);
      int area();
};
```

Objects: Instance of a class

```
Rectangle  r1;
Rectangle  r2;
Rectangle  r3;
   ⋮
```

```
int   a;
```

OOP Pillars

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

What is C++ ?

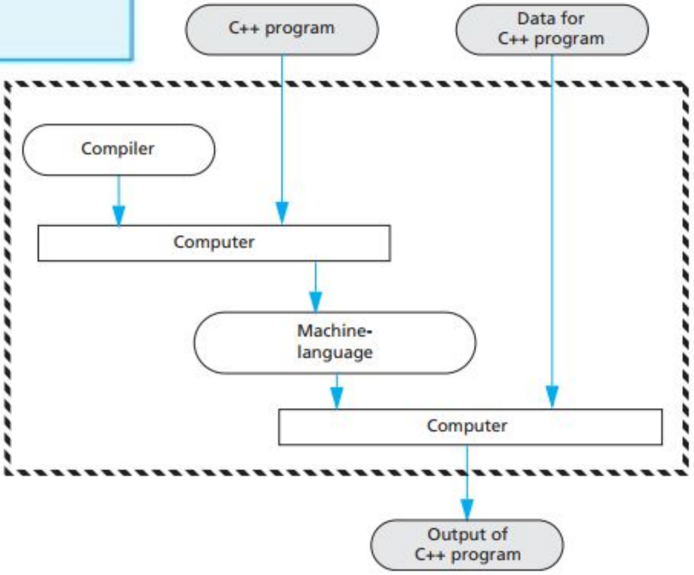
Like (low-level) assembly language, C language programs can directly manipulate the computer’s memory. Bjarne Stroustrup of AT&T Bell Laboratories developed C++ in the early 1980s.

Unlike C, C++ has facilities to do object-oriented programming, which is a very powerful programming techniques

Compiling and Running of a C++ program

Compiler

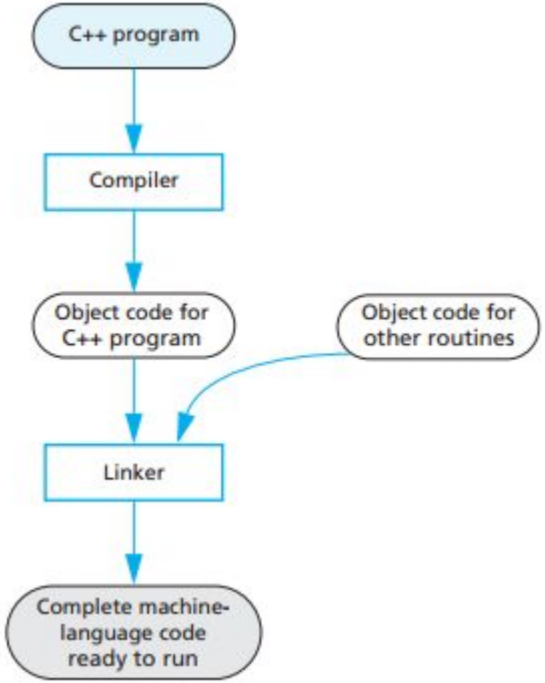
A **`compiler`** is a program that translates a high-level language program, such as a C++ program, into a machine-language program that the computer can directly understand and execute.



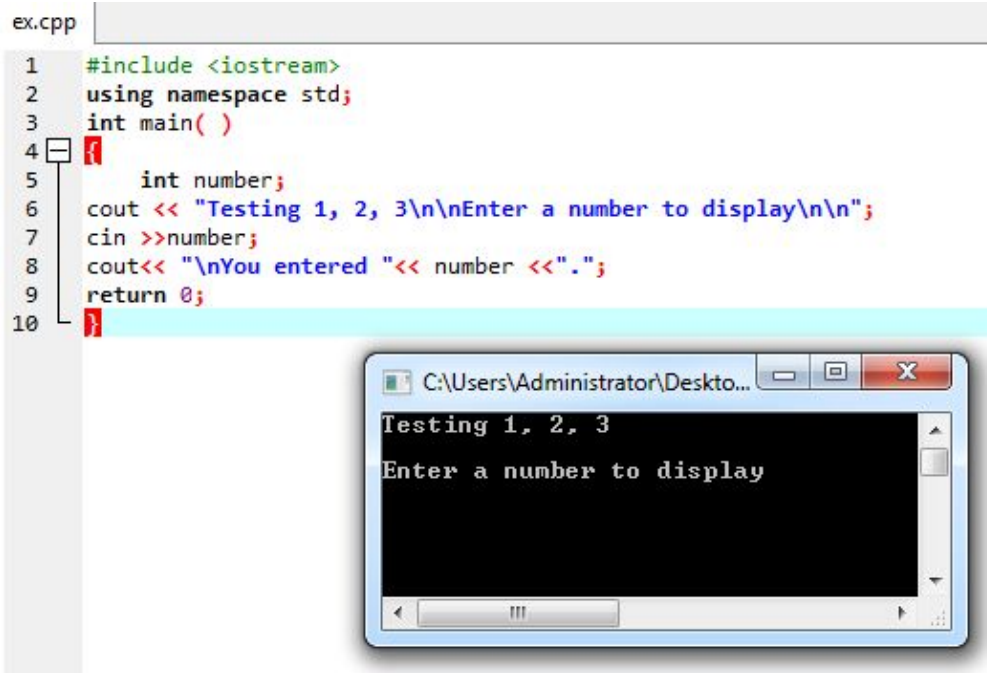
Compiling and Running of a C++ program

Linking

The object code for your C++ program must be combined with the object code for routines (such as input and output routines) that your program uses. This process of combining object code is called **`linking`** and is done by a program called a **`linker`**. For simple programs, linking may be done for you automatically.


C++ program
Compiler
Object code for C++ program
Object code for other routines
Linker
Complete machine-language code ready to run

Sample C++ Program



ex.cpp

```
#include <iostream>
using namespace std;
int main( )
{
    int number;
cout << "Testing 1, 2, 3\n\nEnter a number to display\n\n";
cin >>number;
cout<< "\nYou entered "<< number <<".";
return 0;
}
```

C:\Users\Administrator\Deskto...

```
Testing 1, 2, 3

Enter a number to display
```

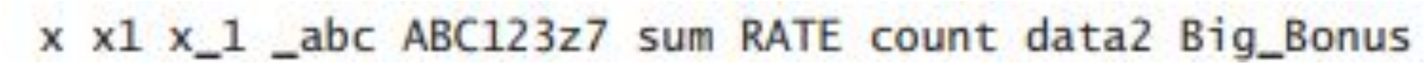
Include directive tells the compiler where to find information about certain items that are used in your program. In this case iostream is the name of a library that contains the definitions of the routines that handle input from the keyboard and output to the screen

It says that the names defined in iostream are to be interpreted in the “standard way”

The return statement says to end the program when it gets here.

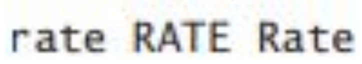
Variables

- Variable -> To hold a certain type of value in memory.
- Identifier -> The name of the variable.It must start with either a letter or   the underscore symbol.



```
x x1 x_1 _abc ABC123z7 sum RATE count data2 Big_Bonus
```

- Keywords -> Reserved words that have special meaning and cannot be used as identifiers.
- C++ is a case-sensitive language; that is, it distinguishes between uppercase and lowercase letters in the spelling of identifiers. Hence the following are three distinct identifiers and could be used to name three distinct variables.



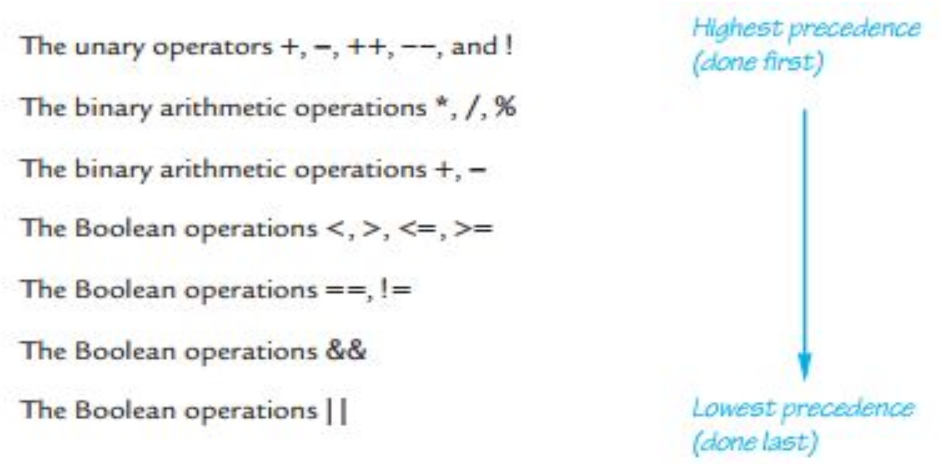
```
rate RATE Rate
```

Data Types

- The Type int and double
- The Type char
- The Type bool
- The Class String
    - C++ lacks a native data type to directly manipulate strings, there is a string class that may be used to process strings

Operators

- Arithmetic
- Assignment
- Relational/Comparison
- Logical



Highest precedence (done first)

The unary operators +, −, ++, −−, and !

The binary arithmetic operations *, /, %

The binary arithmetic operations +, −

The Boolean operations <, >, <=, >=

The Boolean operations ==, !=

The Boolean operations &&

The Boolean operations ||

Lowest precedence (done last)

Iterative Statements/ Loops

- while
- for
- do-while

Input/ Output

- C++ I/O occurs in streams, which are sequences of bytes. If bytes flow from a device like a keyboard, a disk drive, or a network connection etc. to main memory, this is called input operation and if bytes flow from main memory to a device like a display screen, a printer, a disk drive, or a network connection, etc., this is called output operation.
- cout is used together with the *insertion operator*, which is written as <<
- The cin is used in conjunction with the stream extraction operator, which is written as >>
- Multiple insertion operations (<<) may be chained in a single statement

```
cout << "This " << " is a " << "single C++ statement";
```

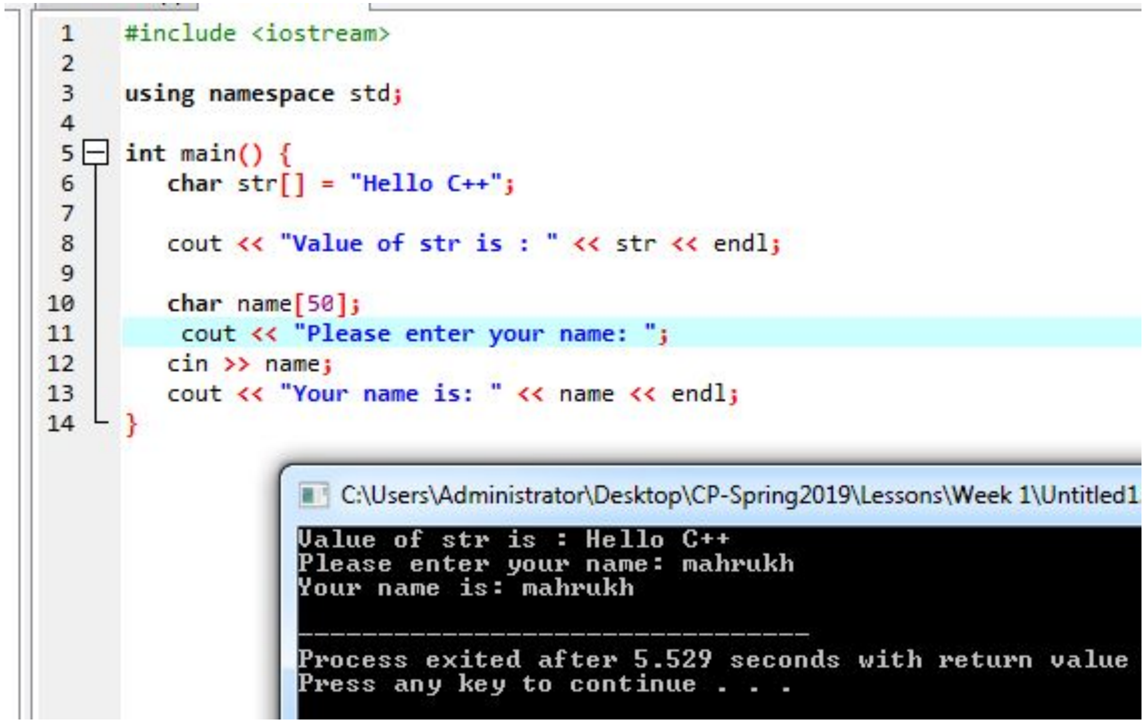
- endl manipulator can also be used to break lines.

```
cout << "First sentence.\n";
cout << "Second sentence.\nThird sentence.";
```

```
cout << "First sentence." << endl;
cout << "Second sentence." << endl;
```

- <iostream> defines the cin, cout objects, which correspond to the standard input stream, the standard output stream.

Example



```cpp
#include <iostream>

using namespace std;

int main() {
   char str[] = "Hello C++";

   cout << "Value of str is : " << str << endl;

   char name[50];
    cout << "Please enter your name: ";
   cin >> name;
   cout << "Your name is: " << name << endl;
}
```

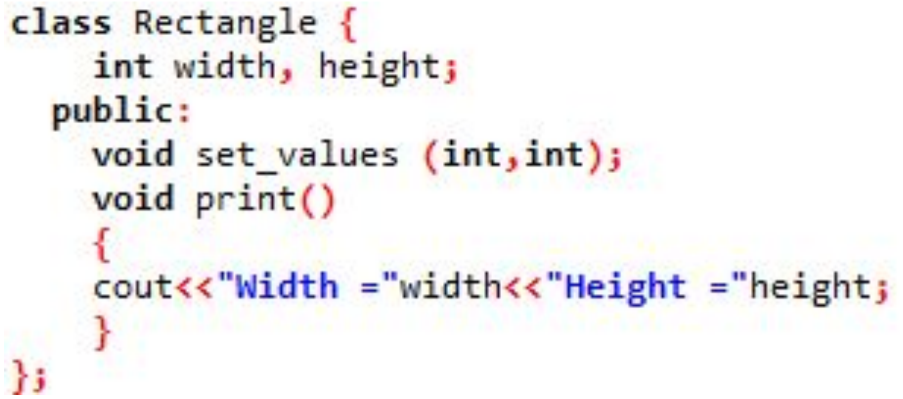
C:\Users\Administrator\Desktop\CP-Spring2019\Lessons\Week 1\Untitled1

```
Value of str is : Hello C++
Please enter your name: mahrukh
Your name is: mahrukh

--------------------------------
Process exited after 5.529 seconds with return value
Press any key to continue . . .
```

Defining class and class members

- class= member variable +member function

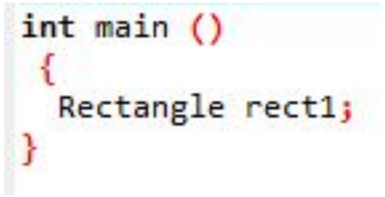


```
class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    void print()
    {
    cout<<"Width ="width<<"Height ="height;
    }
};
```

Object Instantiation

Syntax:

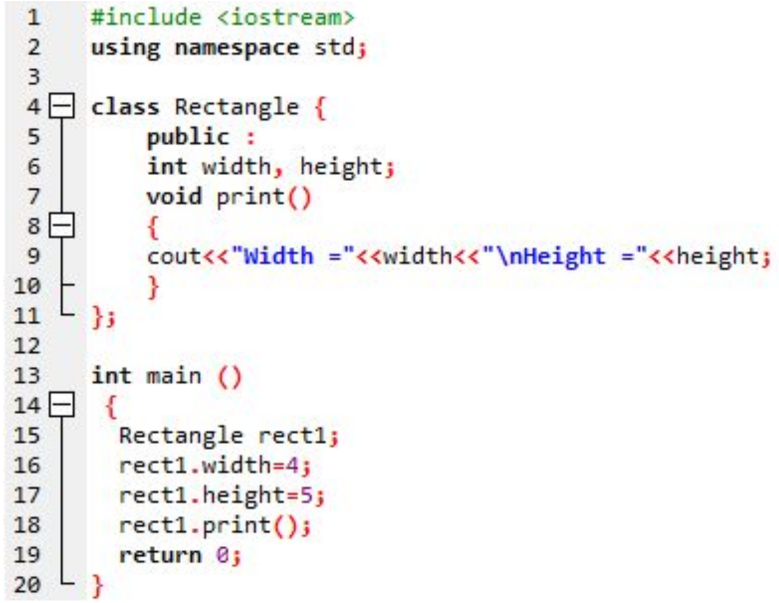
classname objectname ;



```
int main ()
 {
  Rectangle rect1;
}
```

Accessing member variable and functions

- Dot operator used with objects.



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

class Rectangle {
    public :
    int width, height;
    void print()
    {
    cout<<"Width ="<<width<<"\nHeight ="<<height;
    }
};

int main ()
 {
  Rectangle rect1;
  rect1.width=4;
  rect1.height=5;
  rect1.print();
  return 0;
}
```