Introduction to Classes and Objects

C++ is a middle level programming language developed by Bjarne Stroustrup in 1970 at Bell laboratories. C++ is a general purpose, case-sensitive, free-form programming language which supports procedural, object-oriented, and generic programming.

Now days, Object oriented programming playing significant role in design and implementation of software systems. It simplifies the development of large and complex software systems and helps in the production of software, which is modular, easily understandable, reusable, and adaptable to changes. Object oriented modelling is a new way of visualizing problems using models organized around the real world concepts. Objects are the result of programming methodology rather than a language.

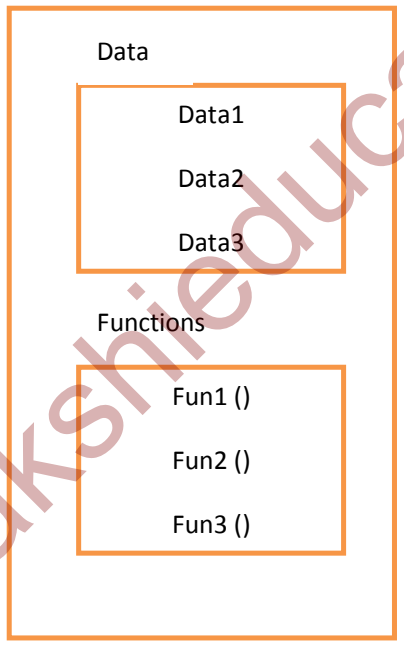


Figure1: Class grouping data and functions

Object oriented programming constructing a modelled out of data types called classes. Defining variables of class data type is known as class instantiation and such variables are called objects (It is an instance of a class). A class encloses both data and functions that operate on the data, into a single unit shown in figure1. The variables and functions enclosed in a class are called data members and member functions respectively. Member functions define the permissible operations on the data members of a class.

Placing data and functions together in a single unit is the central aim of object oriented programming. Classes are the basic language construct of c++ for creating user defined data

types. These are extension of structures. The only difference is that, all member of structures are public by default whereas, the members of classes are private by default. Class follows the principle data and member functions are private by default until we specifically declared as public.

Class

A class is a name or blue print form of an object. Thus, class can be declared as collection of data members along with member functions. This property combining of data and functions into a single unit is called encapsulation. This class doesn't define any data; it just defines class name, what an object of class consists of and which operations performed on such an object.

The syntax of class shown in below figure:

```
class Classname{

    // body of a class

    Access_Specifier:

      Member1;

      Member2;

     ……

};
```

Where class keyword indicates that the name which follows defined by user. The body of class enclosed with in curly braces followed by a semicolon- the end of class specification. The body of class contains declaration of variables and functions, collectively known as members. The variables declared inside class is called as data members and functions defined in class are called as member functions.

Access_specifier, is used to define the visibility of members. An access specifier is one of the following three keywords: private, public, protected. These access specifiers modify the access rights for the members that follow them.

private: Members of a class are accessible only from within other members of the same class.

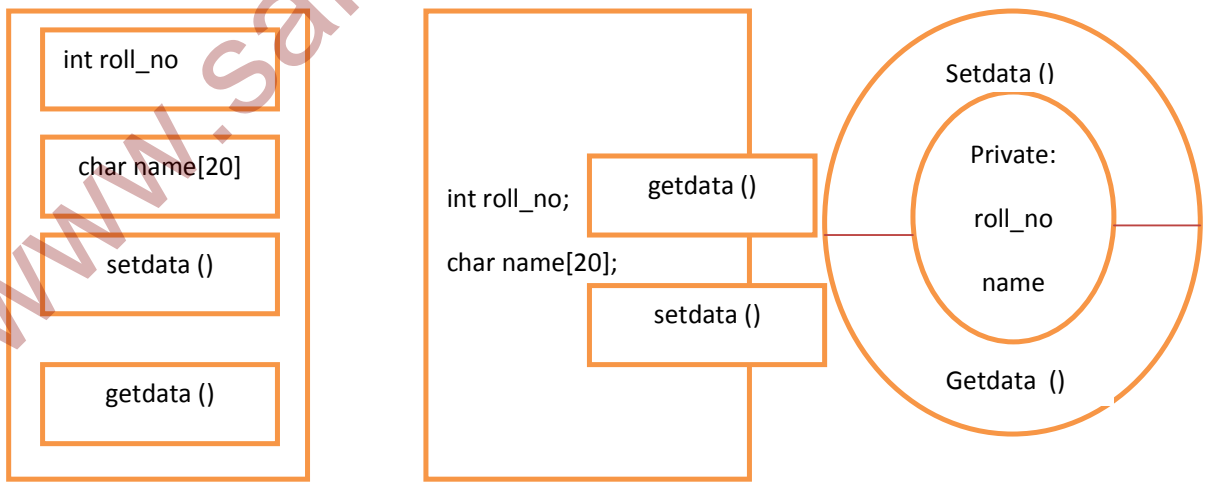
protected: Members of a class are accessible from within other members of the same class and also members of their derived classes.

public: Members are accessible anywhere in the class, where object visible.

The following example illustrates the specification of a class called student having roll_ no and name as its data members.

```
class Student{
    private:
        int roll_no;   //roll number
        char name[50];  //student name
        void setdata(int roll_no_in char* name_in){
            roll_no=roll_no_in;
            strcpy(name,name_in);
        }
         void getdata(){
            cout<<"Roll No"<<roll_no<<endl;
            cout<<"Name:"<<name<<endl;
        }
};
```

A class should be given some meaningful name, reflecting the information it holds. The class student contains two data members and two member functions. The data members are private by default. The member function setdata() can be used to assign values to the data members roll_no and name. The member function getdata() can be used for displaying the values of data members. Three different notations for representation of the student class is shown in figure 2.



(a) (b) (c)

Figure(2): Different representations of class student

Object

An object is an instance of a class. A class specify only structure of an object and it must be instantiated to use services provided by it. This process of creating objects of a class is called class instantiation. It is the definition of object that actually creates objects in the program by setting aside memory space for its storage. Hence, a class is like a blueprint of a house and it indicates how the data and functions are used when the class instantiated. The necessary resources are allocated only when a class is instantiated. The syntax for defining an object of class shown in below:

ClassName objectname;

An example of instantiation for creating objects for student class is shown below.

Student s1;

It creates the object s1 of the class student. More than one object can be created with a single statement as follows:

Student s1, s2, s3;

It creates multiple objects for the class student.

The definition of an object is similar to that of a variable of any primitive data type. Objects can also be created by placing their names immediately after the closing brace like in the creation of the structure variables.

```
class Student{
        ……..
        ……..
}s1, s2, s3;
```

In C++, convention of defining objects at the point of class specification is rarely followed. The user would like to define the objects as and when required, or at the point of their usage.

Accessing Class Members: Once the object of a class instantiated we can access its members. This is achieved by using the member access operator, dot (.). The syntax for accessing both its data and functions of class is shown below figure.

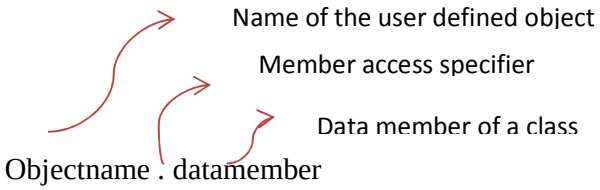


Figure (a): Syntax for accessing data member of a class

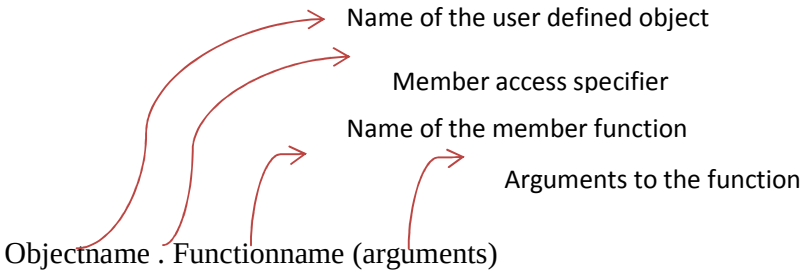


Figure (b): Syntax for accessing member function of a class

If a member to be accessed in a function, then a pair of parenthesis is to be added following function name. The following statements access member functions of the object s1, which is an instance of the student class:

s1.setdata(10, “Tarunkumar”);

s1.getdata();

The following program student.cpp demonstrates the declaration of class Student with the operations on its objects.

//**Student.cpp:** member functions defined inside the body of the student class

```
#include<iostream>

#include<cstring>

class Student{

    private:

            int roll_no;   //roll number

            char name[50];  //student name

    public:

           // initializing data members

            void setdata(int roll_no_in char* name_in){

                    roll_no=roll_no_in;

                    strcpy(name,name_in);

            }

            // display data members on the console screen

             void getdata(){

                    cout<<”Roll No:”<<roll_no<<endl;
```

```
            cout<<"Name:"<<name<<endl;
        }
};
int main(){
    Student s1;         // first object/variable of class student
    Student s2;         // second object/variable of class student
    s1.setdata(1, "Tarun kumar"); // object s1 calls member setdata()
    s2.setdata(10, "Pallavi");    // object s2 calls member setdata()
    cout<<"Student Details….."<<,endl;
    s1.outdata();          // object s1 calls member function outdata()
    s2.outdata();          // object s2 calls member function outdata()
}
```

Output:

Student Details…..

Roll No: 1

Name: Tarun Kumar

Roll No: 10

Name: Pallavi

Access Control Specifiers: Each user has different privileges to access object. A class differentiates these access privileges by portioning its contents and associating each one of them with any one of the following keywords:

- Private
- Public
- Protected

These keywords are called access – control specifiers. All the members that follow a key word belong to that type. If no keyword is specified, then the members are assumed to have private privilege.

Private: The private data members of a class have strict access control. Only the member functions of same class can access these members. The private members of a class are

inaccessible outside the class. If someone tries to access these private members, they will get compile time error. By default member functions and class variables are private.

```
class PrivateAccess{
    private:  // private access specifier
int x;      //Data member declaration
void display(){}  //member function declaration
};
```

Public: The public data members of a class are visible to outside the class, should be declared in public section. All the data members and member functions are accessible anywhere in the program without any restriction either by same class or external class.

```
class PublicAccess{
    public:  // public access specifier
        int x;      //Data member declaration
        void display(){}  //member function declaration
};
```

Protected: This is similar to private; it makes data members inaccessible outside the class. But they can be accessed by sub class of that same class (called inheritance).

```
class ProtectedAccess{
    protected:  // protected access specifier
        int x;      //Data member declaration
        void display(){}  //member function declaration
};
```

Class Scope: There should be a mechanism of binding the functions to the class to which they belong. This is done by using scope resolution operator (::). It acts as an identity-label to inform the compiler, the class to which the function belongs.

This operator can also be used to quantify the hidden names in the class, global scope name is hidden by explicit declaration same class or block, and still we can use them.

The below example illustrates the use of scope resolution operator.

// This Example program illustrates the accessing the global variable using scope resolution operator

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

char ch = 'x';     // global variable

int main() {
  char ch = 'y';   //local variable

  cout << "Local ch: " << ch<< "\n";
  cout << "Global ch: " << ::ch<< "\n";  //using scope resolution operator

  return 0;
}
```

// This Example program illustrates the use of scope resolution operator in class

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

class ClassScope {
public:
  void display();  //function declaration
};

// function definition outside the class

void ClassScope::display() {
  cout << "Function defined outside the class.\n";
}

int main() {
ClassScopex;
  c.display();

  return 0;
}
```