C++ & Object Oriented Programming Concepts

The procedural programming is the standard approach used in many traditional computer languages such as BASIC, C, FORTRAN and PASCAL. The procedural programming is a step by step program that guides the application through a sequence of instructions. These languages such as C, Pascal etc are having some problems in creating reusable software modules. In this the programs are made up functions and functions are often not reusable or it is very difficult to copy a function from one program to another program and reuse because the functions are likely to reference global variables, headers and refers to other functions. The structural programming languages are not suitable for high level abstraction for solving real world problems. In procedural programming the programs are basically list of instructions and data and these instructions works with data to give results. Instructions and data are the two separate entities that are combined together to produce required results.

Advantages of procedural programming languages

- Simple and ease of implementation of compilers and interpreters.
- The ability to re-use same code at different places in the program without copying it.
- Easy to keep track of program flow.

The limitations of procedural programming languages

- The data is global that exposed to the whole program, so no security for the data.
- Procedural languages are difficult to relate with real time problems.
- Procedural codes are difficult to maintain as the size of code increases.
- Difficult to create new data types and reduce extensibility.
- The procedural language does not have automatic memory management, so it makes the programmer to concern more on the managing the memory for program.

OOPS:

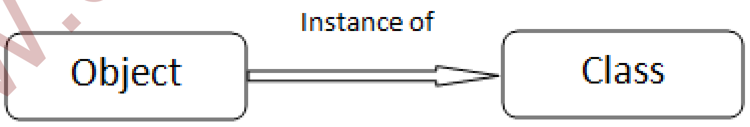
The Object oriented programming languages are designed to overcome the following problems.

- The traditional programming language thinks in terms of computer terminology rather than thinking in terms of problem solving. Whereas object oriented programming languages provides higher level of abstraction for solving real time problems.
- The basic unit of object oriented programming is a class, which encapsulates both static and dynamic behaviors within a box and specifies the public interface for using these boxes. The object oriented programming combines the data structures and instructions of a software entity inside the same box.

Object oriented programming provides the ability to design an 'object', and which associated with the concepts of objects and having data fields and related member functions. This allows programs to be written in more modular way, which makes them easier to understand and also provides higher degree of code reusability.

Object

An object can be considered as a 'thing' that can perform a set of related activities. The set of activities that the object performs defines the objects behavior. Objects are instances of classes and are used to interact amongst each other to create application. The instance means the object of class on which we are currently working. The C language with classes can be said as C++. In C++ everything revolves around the object of class, which their data members and methods (functions). For example consider a human body as class and multiple objects of this class with variables as hair, color and etc. and methods as speaking, walking etc.



Comparison of OOPS with procedural programming language:

The Procedural oriented languages are focus on procedures with functions as the basic unit, that is first figure out all the functions and then think about how to represent the data. Whereas the Object oriented languages focus on component that the user prospective with objects as the basic unit, here figure out all the objects by putting all the data and instructions that describe the user interaction with the data. In procedural programming the data and functions are stored in separate memory locations, where as in object oriented programming the data and methods (functions) are stored in same memory locations.

Advantages of OOPS

- OOPS supports and provides modular structure for developing applications.
- Object oriented programs are easier to understand, therefore easier to debug and maintain.
- It is easier to modify the programs and reusability which increases the speed of software development.
- Ease in software design as we think in the problem space rather than computer terminology.

Disadvantages of OOPS

- Object oriented programs are bigger in size because they have many lines of code, so they are slower than the procedure oriented programs.

The main features of object oriented programming are

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

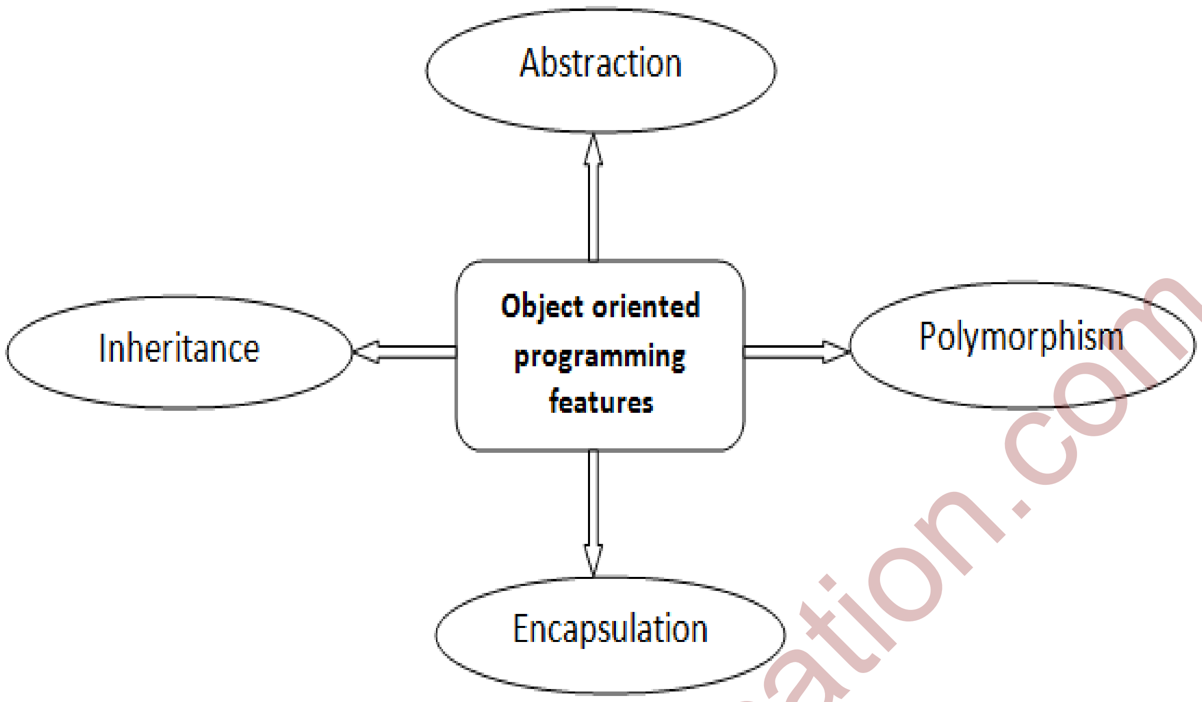


Figure1: Features of OOPS

Now let us have some discussion on the features of object oriented programming which will be used in C++ programming.

<u>C++ Class</u>

The classes are the most important feature of C++ programming. A class is the collection of related data members (Variables) and methods (functions) under a single name. A class is a user defined data type, which can be accessed and used by creating instance of that class. A C++ program can have any number of classes. The class helps to visualize the complex problem efficiently and effectively.

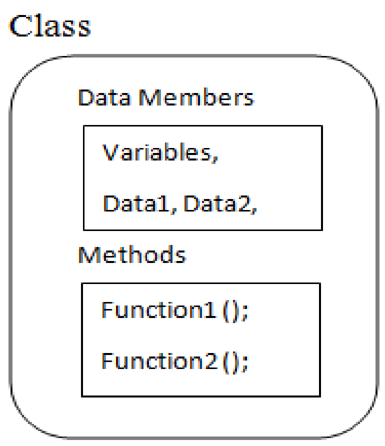


Figure2: Class

Defining a Class:

Class is defined using a keyword 'class' followed by identifier which is the name of class and the body of class starts with open curly brace and terminated with close curly brace with semicolon or with a list of objects declarations at the end. When a class is defined no memory is allocated.

Syntax:

```
class Class_Name
 {
        Access specifier:
        Data_members;
        Methods; // functions();
 };
```

Example:

```
        class Account
        {
        private:
                int acc_no;
                float balance;
         public:
                void acc_details();
                float deposit(float amt);
                float withdraw(float amt);
        }; or
} obj1, obj2;
```

As mentioned above the definition of class starts with keyword ‘class’ followed by name ‘account’ in the above example. The body of class starts with curly braces and terminated by semicolon at the end. There are two keywords used in the above example private and public. These are called access specifiers and we will learn these following sections. By default every member in the class is private.

C++ object

When a class is defined, only specification for the object is defined. Objects are instances of class, which holds the data members declared in the class and methods work on these class data members. Object has same relationship to the class as the variable has with the data type.

The objects can be defined in similar way as structure is defined. The objects get physical memory and contain data members which are declared in the class.

Syntax:

```
Class_Name Object_Name;
```

For the above example the object definition is

Account obj1; → obj1 is the object of Account class.

Example:

```
int main()
 {
        Account obj1;
        Account obj2;
 }
```

Access specifiers

Access specifiers are used for access control in classes. The access specifiers are used to set boundaries for availability of data members and member function (methods) of a class. By default the class members and methods are private. We can use one, two or all three access specifiers in the same class to set different boundaries for different class members.

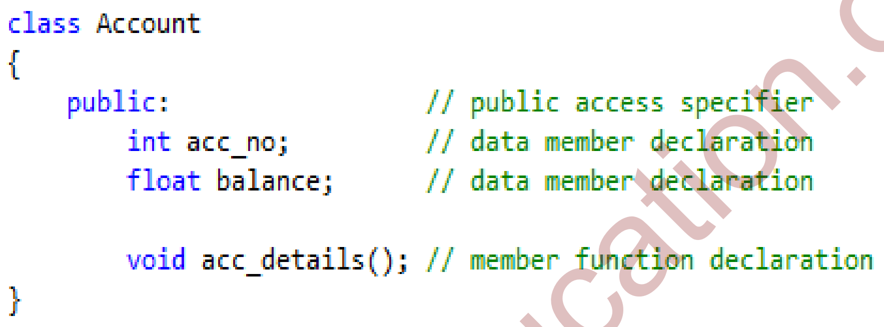
In C++ three types of access specifiers namely

1. Public
2. Private
3. Protected

Public Access specifier

Public keyword indicates all the class members declared under this section will be available to everyone. The data members and methods declared under public section can be accessed by other classes also, so there are chances that other classes can change the members. Therefore the important data members must not be declared in the public section.

Example:

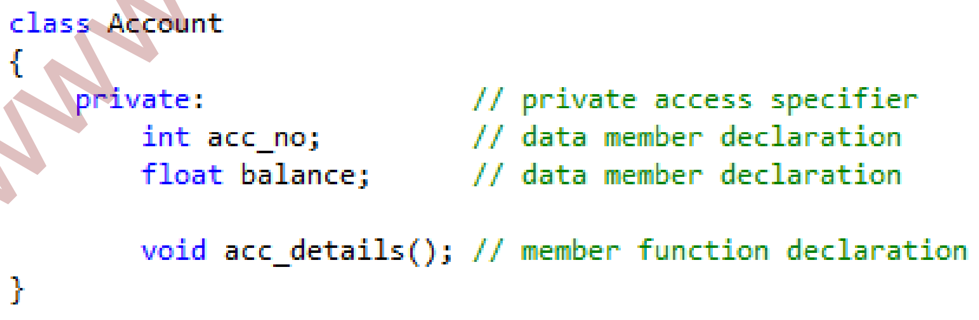


```
class Account
{
    public:                 // public access specifier
        int acc_no;         // data member declaration
        float balance;      // data member declaration

        void acc_details(); // member function declaration
}
```

Figure3: Public member declarations

Private Access Specifier

Private keyword indicates that the class members declared under private section are not available to everyone and no one can access these class members outside of that class. If someone tries to access these data members then the compiler will through a compile time error.

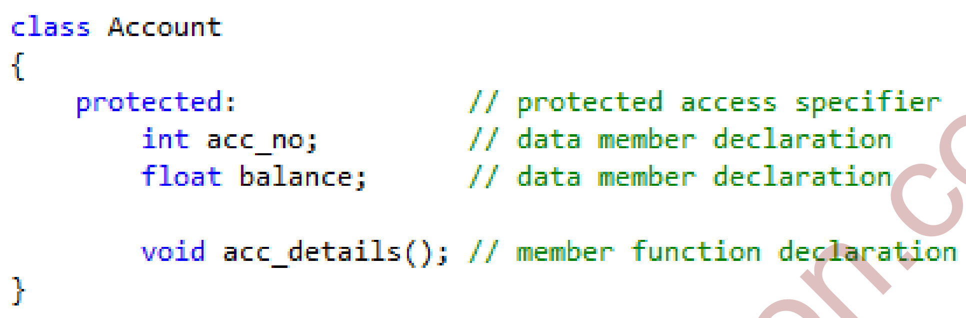


```
class Account
{
    private:                // private access specifier
        int acc_no;         // data member declaration
        float balance;      // data member declaration

        void acc_details(); // member function declaration
}
```

Figure4: Private member declarations

Protected Access Specifier

Protected keyword is similar to private access specifier that is class members are inaccessible outside the class, but they can be accessed by any subclass of that class.



```
class Account
{
    protected:              // protected access specifier
        int acc_no;         // data member declaration
        float balance;      // data member declaration

        void acc_details(); // member function declaration
}
```

Figure5: Protected member declarations

Data members and methods

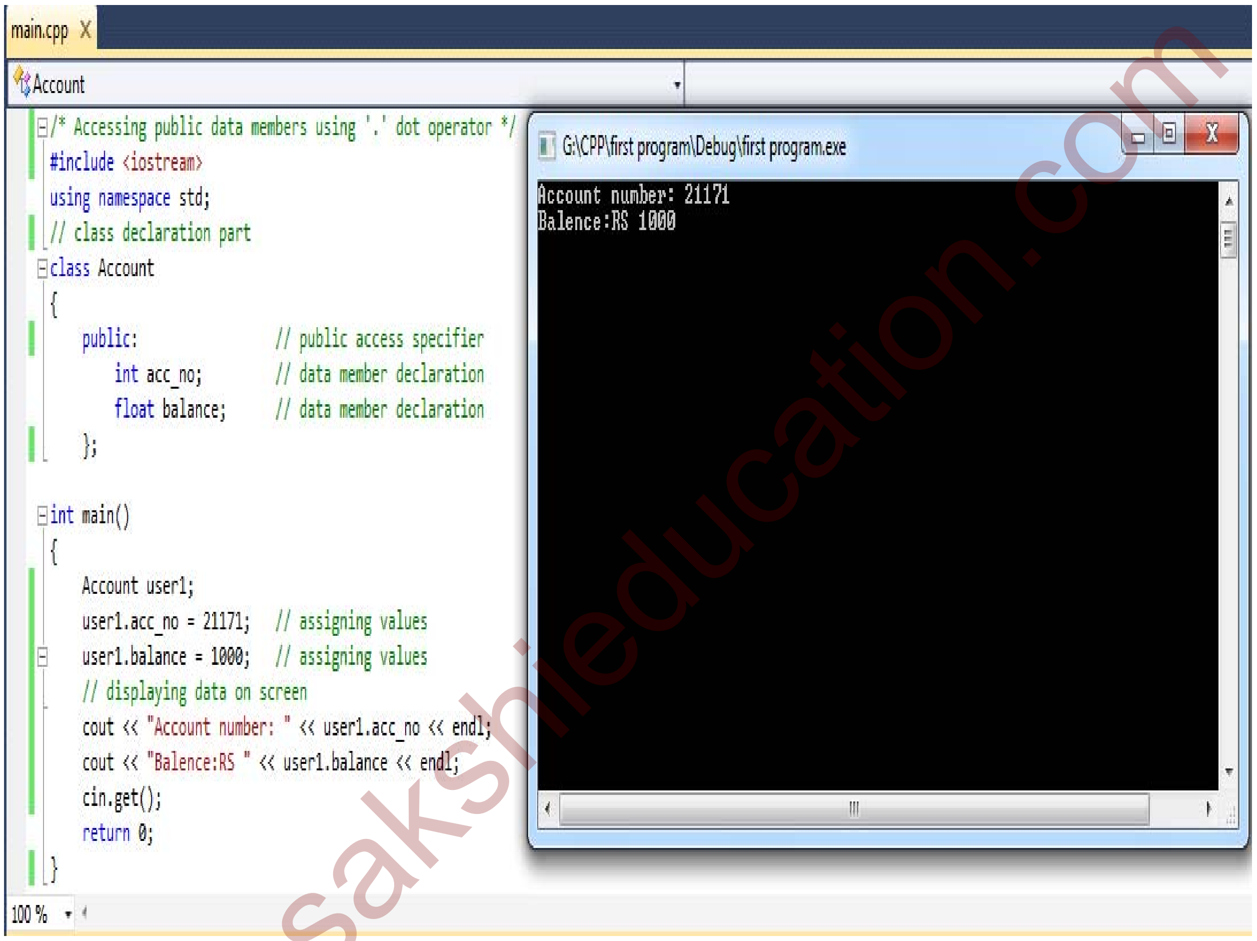
The data within the class is known as data members and the functions defined within the class are known as methods. The data members are declared with any of fundamental data types and also with array types, pointer types, reference types and with user defined data types also. By default every data member in a class is private.

Accessing data members and methods:

Accessing data members depends on the access control specifiers of the data members. If the access control is public then the data members and methods can be accessed in similar way the members of structure is accessed using dot (.) operator. If the access control is private and protected then we cannot access the data members directly, so we have to create public member functions or methods to access the private and protected data members. These member functions are also called set or get functions.

Accessing public data members:

The following example illustrates how to initialize and access the public data members using the '.' dot operator.



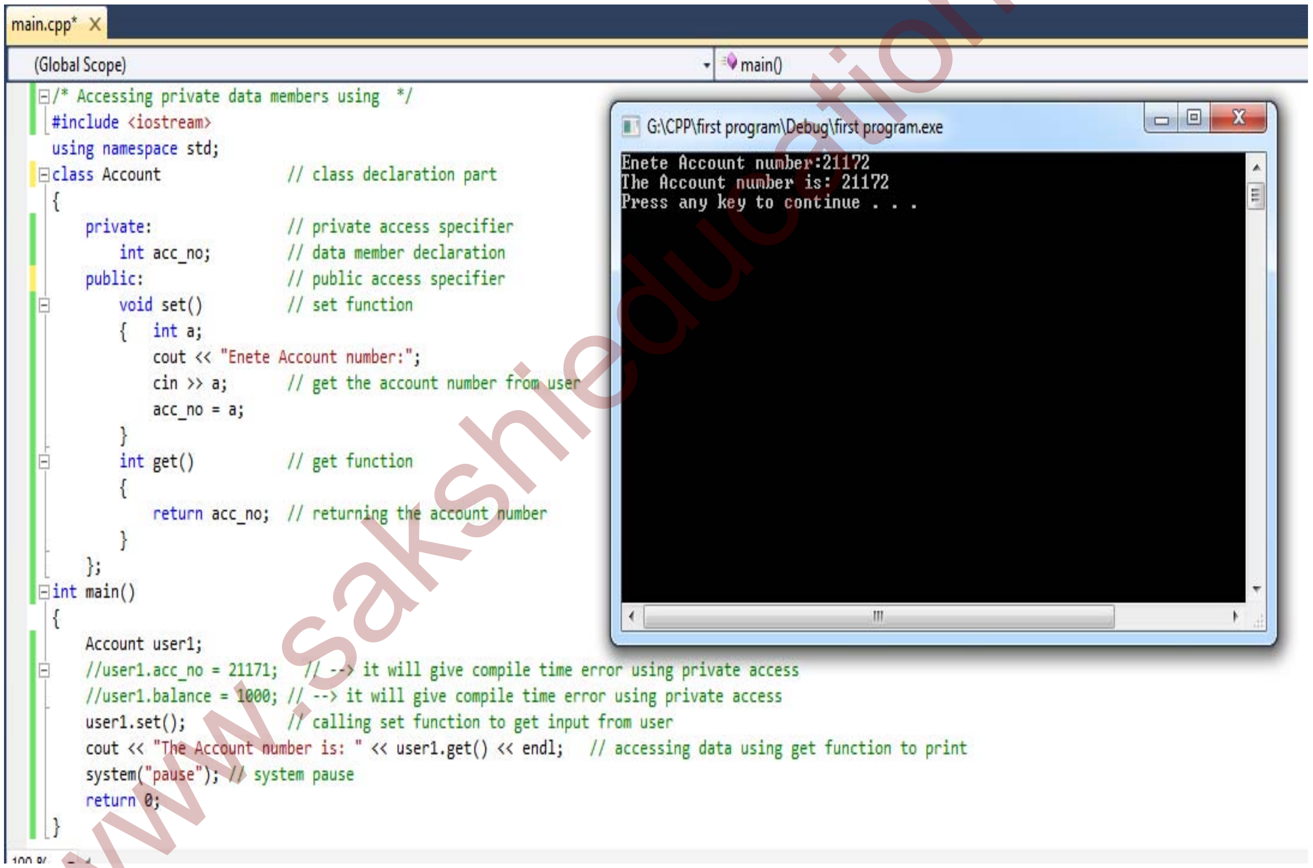
```
/* Accessing public data members using '.' dot operator */
#include <iostream>
using namespace std;
// class declaration part
class Account
{
    public:             // public access specifier
        int acc_no;     // data member declaration
        float balance;  // data member declaration
    };

int main()
{
    Account user1;
    user1.acc_no = 21171;   // assigning values
    user1.balance = 1000;   // assigning values
    // displaying data on screen
    cout << "Account number: " << user1.acc_no << endl;
    cout << "Balence:RS " << user1.balance << endl;
    cin.get();
    return 0;
}
```

```
Account number: 21171
Balence:RS 1000
```

Figure6: Accessing public data members

Accessing private and protected data members:

The following example illustrates how to initialize and access the private data members, to access the private data members we need to create setter and getter functions to set and get the values of the data members. The both the functions must be defined as public, the setter function will set the value passed as argument to private data members and getter function will return the value of the private data member to be used. The protected data members can be accessed as same as private data members, and inside the subclass the members can be accessed directly using '.' dot operator.



```
/* Accessing private data members using  */
#include <iostream>
using namespace std;
class Account                // class declaration part
{
    private:                 // private access specifier
        int acc_no;          // data member declaration
    public:                  // public access specifier
        void set()           // set function
        {   int a;
            cout << "Enete Account number:";
            cin >> a;        // get the account number from user
            acc_no = a;
        }
        int get()            // get function
        {
            return acc_no;   // returning the account number
        }
    };
int main()
{
    Account user1;
    //user1.acc_no = 21171;   // --> it will give compile time error using private access
    //user1.balance = 1000; // --> it will give compile time error using private access
    user1.set();             // calling set function to get input from user
    cout << "The Account number is: " << user1.get() << endl;   // accessing data using get function to print
    system("pause"); // system pause
    return 0;
}
```

```
Enete Account number:21172
The Account number is: 21172
Press any key to continue . . .
```

Figure7: Accessing private data members