**Classes AND Object**

A class is used to encapsulate the data member and member functions in a single unit. Contents of a class are known as **members** of the class; data inside the class is known as **data member** and function inside the class is known as **member function**. Member of class are **private** by default i.e they are not accessible outside the scope of the class).

An object has the similar relationship to a class that a variable has to it’s data type. An **object** is an **instance** of a class, in the same way ‘YOU’ and ‘ME’ are the instance of class ‘HUMAN’.

Class may contain data as well as functions

**EXAMPLE of class**

A class with no instance (no object) is known as **Abstract Class** and a class having objects is known as **concrete class**. Every class has it’s **identity, state and behavior**. Identity is the property of the object. Static and dynamic properties associated with the object signify the **state** of the object. The **operation/Function** associated with the object shows the behavior of the object.

A class has three visibility mode for it’s members

<table>
<thead>
<tr>
<th>Visibility</th>
<th>Symbol</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>✗</td>
<td>Object</td>
</tr>
<tr>
<td>Protected</td>
<td>✗</td>
<td>Object</td>
</tr>
<tr>
<td>Public</td>
<td>✓</td>
<td>Object</td>
</tr>
</tbody>
</table>

Private member – Not Accessible to object

Protected member – Not accessible to object

Public member- Accessible to the object
In C++, a class can be defined with the following example

```cpp
class Class_name {
    Data member;
    ......
    ......
    Member function;
protected:
    Data member;
    ......
    ......
    Member function;
public:
    Data member;
    ......
    ......
    Member function;
};

Class_name list-of-objects
```

member of the class can be kept any of the three visibility modes private, protected and public. Members in **public visibility mode** of the class area accessible to the object of the class whereas the member of **private visibility mode** are not accessible to the object of class (are only accessible inside the class to the member function of the class)

**NOTE:** Protected members will be discussed in the inheritance topic

**NOTE:** if a class defined outside function (main or any other) then it is called global class and it is defined inside the class then it is called local class.

**NOTE:** data member (private and public both) are directly accessible through it’s member function. There is no need to pass them in the member function.

following C++ program demonstrate the use of class and object

```cpp
#include<iostream.h>
class student {
    int roll;    //private data member
    char name[30];   // private data member
public:
    void input_data(); // public member function prototype
    void show_data()   // public member function with definition
    {        cout<<"\n Roll No  :";    cout<<"\n Name         :";        cin>>roll;
        cin>>name;
    }
};

void student :: input_data ()   // member function definition outside the class
{    cout<<"\n Enter roll no :";    cin>>roll;
    cout<<"\n Enter Name :";    cin>>name;
}
```
```c++
int main()
{
    student S1,S2; // declaration of the object S1,S2 of student
    S1.input_data(); // member function call for the object S1
    S2.input_data(); // member function call for object S2
    S1.show_data();
    S2.show_data();
}
```

In the above example, one can see that a class `student` is defined with two private data members `roll` and `name` and two member functions `input_data` and `show_data`. Member function can be defined inside the class and outside the class, as shown in the example, member function `show_data()` is defined inside the class whereas member function `input_data()` is defined outside the class using `scope resolution operator(::)` with the name of the class it belongs to. Both of these functions can be called from the objects of the class in the same way. Generally, single line functions are defined inside the class and multiple line functions are defined outside the class to have clarity in understanding the behavior of the class. **Function with control structure should be defined outside the class.**

The size of object (in bytes) depends upon the data members present in the class it belongs to. In the above example, object `S1` occupies 32 bytes.

**Nesting of member function**: A member function can call another member function without the use of object.

**Example**
```c++
/* program to demonstrate nesting of member function */
#include<iostream>
#include<conio.h>
using namespace std;
class area
{
    int b,h,ar;
    public:
    void input_data()
    {
        cout<<"\n Enter b:";cin>>b;
        cout<<"\n Enter h:";cin>>h;
    }
    void calculate()
    {
        input_data();
        ar =0.5*b*h;
    }
    void output_data()
    {
        calculate();
        cout<<"\n Area of triangle :"<<ar;
    }
};
int main()
{
    area A;
    A.output_data();
    getch();
    return 0;
}
```
**Member function with return type value**

```cpp
#include<iostream>
#include<conio.h>
using namespace std;

class area
{
    private:
    float b,h,ar;
    public:
    void input_data(void);
    float calculate(void);
};

void area :: input_data(void)
{
    cin>>b>>h;
    cout<<"\n Area of Triangle :"<<calculate();
}

float area :: calculate(void)
{
    ar = 0.5*b*h;
    return(ar);
}

int main()
{
    area A;
    A.input_data();
    getch();
    return 0;
}
```

**Member function with parameter**

```cpp
/* member function with parameter */
#include<iostream>
#include<conio.h>
using namespace std;

class area
{
    float b,h,ar;
    public:
    void input_data(int x,int y)
    {
        b=x;
        h=y;
    }
    void calculate();
    void output_data();
};

void area :: calculate()
{
    ar = 0.5*b*h;
}

void area :: output_data()
{
    cout<<"\n Area of Triangle :"<<ar;
}

int main()
{
    area A;
    A.input_data(10,20);
    A.calculate();
    A.output_data();
    getch();
    return 0;
}
```
Additional Definition

Polymorphism: The process of using an operator or a function in different ways for different set of inputs given is known as polymorphism. Function overloading is an example of polymorphism, where the function having same name with different set of parameters perform different operations.

Example
```cpp
void disp( ) // function1
{
    cout<"Hello"<<endl;
}
void disp (int N) // function2
{
    for(int i=1;i<=N;i++)
        cout<<i<<endl;
}
void disp(int N, int M)
{
    for(int i=1;i<=M;i++)
        cout<<N<<"x"<<i<<N*i<<endl;
}
int main( )
{
    int x=5,y=10;
    disp(x); // call for function2 - print number from 1 to 5
    disp(x,y); // call for function 3 - print table of 5 upto 10 multiple
    disp( ); // call function1 - print Hello
    return 0;
}
```

Encapsulation: It is a property of a class by which data member and member function can be wrapped in a single entity is called encapsulation. It is implemented in C++ by class.

Example
```cpp
class xyz
{
    int a;
    float b;
    public:
        void input_data(void);
};
```

Data Hiding: By this property of a class only the desired members of a class in visible(accessible) by the object of the same class and rest of it’s members are hidden from it’s object. The hidden members are defined inside the class as a private members and the members which is visible to it’s object is defined as public.

Example
```cpp
class xyz
{
    int a;
    float b;
    public :
        void input_data(void);
};
```

These members are hidden as they are defined as private

This function is visible by the object of this class as it is defined