Object Oriented Programming(OOP).

OOP terminology:

Class : A class is a way to bind data and its associated function together. It allows the data to be hidden.

class Crectangle
Data members length; breadth;

Member Functions
set_val();
area();
Perimeter ();
print_val();

Data members can be viewed as the **attributes** or the features of the class and the member functions are the actions to be performed on the data .

This concept of integrating data and functions together is termed as **Data encapsulation.**
In the above example length and the breadth are the attributes or the characteristics of a rectange i.e. using these parameters we can describe a rectangle. The 4 functions declared above are the actions to be performed on these data.
For e.g the function area acts on the attributes length and breadth and printouts out the are of the rectangle.

Base and height are the attributes of a triangle.

Class Specification comprises of the following two parts

1) Class declaration.
2) Class function definition.

Class declaration

class class-name

```
{
   private:
            variable declaration;
             function declaration;
   public:
            variable declaration;
            function declaration;

};
```

This semicolon is important and is used to terminate the class declaration

The data members and the member functions can be declared in the public section and the private section. These two are collectively referred to as members.

Private section: This section is used to conceal the data (data hiding) and the members declared in this section are blocked from external access, i.e the functions outside the class declaration will be prevented from accessing these members, with one exception i.e **friend function.** These members can only be accessed by the member functions.

Public section: This section is transparent to all the functions outside the class declaration.

*By default all the members of the class are private , i.e if the visibility tag is not specified explicitly then the members will be considered as private.

2)Defining member functions

There are two ways to do that

- **Outside the class declaration**: The prototype is written in the class declaration, then by using the scope resolution operator the function is defined outside the class declaration.
- **Inside the class declaration**(inline functions): These functions are defined inside the class declaration.

:: -- scope resolution operator, enables us to define a function outside the class declaration.

```
class CRectangle
   {
    int length,breadth;

  public:
    void set_values (int , int);

   //function inside the class definition (inline function)
   int area (void) {return (length*breadth);}

   int perimeter(void);
  };

void CRectangle::set_values (int a, int b)

{
  length = a;
  breadth = b;
}

int CRectangle::perimeter(void)

{
 return(2*(length + breadth));
}
```

Creation of Object:

Objects are the instances of the class and possess all the features of the class

Syntax classname objectname;

e.g. CRectangle rect1;
CRectangle rect1,rect2, rect3, rect4; //multiple objects

This declaration is similar to the following declaration
int a , b; // a and b are the variables of the type integer.

rect1, rect2, rect3 are the instances of the class CRectangle and possess all the characteristics of the class CRectangle including the access to its member functions.

rect1 , rect2 , rect3 are rectangles which share the same attributes length and breadth and can be acted upon by the same set of actions(functions).

Accessing class members:

Syntax

Object-name.member;

dot operator

For e.g.

```
rect1.area();
rect2.set_val();
```

Accessing the private members using the above procedure is considered invalid

e.g.

```
rect1.length = 10;
rect1.breadth = 6;
```

Invalid expressions

Private members can only be accessed by using the member functions.

e.g.

```
rect1.area();
rect1.perimeter();
```

To understand, class definition, object creation, accessing class members and defining the member function refer to the inclass example [33] on the class website .

.

Accessing the private functions

The functions which are defined in the private section cannot be accessed by any function outside the class. For accessing such functions we make use of the member functions. refer to example[34].

Arrays with in a class : The usage of arrays inside the class is identical to their usage inside a C++ program.

e.g.

```
class array
{
int a[10];

public:
    void set_val(void);
};
```

Function prototype

```
void array:: set_val(void)
{
   for(int i = 0;i<=10 ; i++)
      {
          a[i] = i;
       }

}

int main()
{
   array array1;// creating an object array1 of the class array

    array1.set_val();
return 0;

}
```

Array of Objects
e.g.

```
class employee
{
   char name[30];
   float age;
public:
    void getdata(void);
    void putdata(void);
};

int main()
{
    employee worker[75];//declares an array of objects of the class employee

    for(int i = 0;i<75;i++)
    worker[i].putdata();    // accessing member functions using the object.

return 0;
}
```

*the above program is incomplete , the functions putdata and getdata aren't defined

Passing Objects as Function Arguments :

This concept is similar to the concept of passing variables, arrays to the functions.

There are two kinds of function calls

- Call by reference.
- Call by value.

Call by value: A copy of the object is passed to the function, but any change made to the object inside the function is not reflected on the object used to call the function.
i.e in order to have the effect felt outside we need to specify an explicit return operation
e.g

```
return(rect1);  //returns an object
```

where rect1 is an object to of the class CRectangle.

To understand this concept better, refer to inclass example[35] .

Constructors and Destructors:

Constructors

In all the examples considered so far, the member function set_val() has been used to initialize the private members of the object.

```
CRectangle rect1;
rect1.set_val(10,6);  //assigns the values 10 and 6 to the length and the breadth of the
                      //rectangle.
```

The above two steps of creation of an object and the initialization of the data members can be clubbed together by using a special member function called the constructor which helps an object to initialize itself when it is created.
Whenever an object is created a constructor is invoked .Unless we intend to perform some operation like initializing the variables at the time of creation, we need not explicitly define a constructor.

There are two kinds of Constructors

- Default Constructors.
- Parameterized Constructors.

Characteristics of Constructors

- They bear the same name as the class , this is how the C++ compiler recognizes that the function defined is a constructor .
- They should be declared in the public section of the class declaration.
- They do not have return types and hence cannot return values.
- We can define a number of constructors by the same in the class definition ,this concept is called as **Constructor Overloading** . C++ compiler distinguishes between these constructors on the basis of their arguments ie default or parameterized constructor.

Destructors:

Destructor as the name implies is used to destroy objects that have been created by the constructor. It is used to release the memory which was allocated when the object was created. It is invoked automatically by the compiler upon exit from the program. It is written in the following manner

```
~CRectangle()
    {
     }
```

refer to the inclass example [37] .

Friend Functions:

Recall that private members cannot be accessed from outside the class i.e. non-member functions cannot access the private members.
There can arise a situation , which might require two classes to share a particular function . In such situations , C++ allows common functions to be made friendly with both classes , thereby allowing the function to have an access to the private data of the classes.

Features of the Friend Classes

- Friend Functions need not be a member of any of the classes.
- It is not in the scope of the class to which it has been declared .
- Since it is not in the scope of the class it is declared , it need not be called using the object of that class . It can be invoked like a normal function with out the object reference .
- Unlike the member functions it cannot access the member name directly and has to use the object name and the dot operator with each member name.

- It can be declared in the public or private section of a class without changing its meaning.
- It has objects as arguments.

To understand this concept better refer to the inclass example [35] on the class website.

References:

[1] C++ for Engineers and Scientists, Gary J. Bronson.
[2] C++ How To Program , Deitel and Deitel.
[3] Object Oriented Programming in Turbo C++ , Robert Lafore.
[4] Object-Oriented Programming , E Balagurusamy.