For example:

The class of a rectangle contains data members: (*length* and *width*). A new class *Square* will have similar data members with the special case (length = width). We also need a member function to calculate the area of a square. Instead of defining the class (*square*) from scratch, we can think of square as a special case of a rectangle. However, we can use the class of a rectangle by inheriting its behavior and redefining the area function to work for the class of (*square*).

3. **Polymorphism**: It is the quality that allows one name to be used for two or more related but technically different purpose. Such as redefining member functions to define a new behavior with different number and/or parameters. It simply means "one name, multiple forms".

Example1:

```
# include <iostream.h>
class item
{  int number;
   float cost;

    public:
     void getdata ( int a  , float  b)
         {  number = a  ;  cost = b  ;  }
     void putdata ( )
         { cout << number   << "\n"  <<  cost  ;  }
};
 main ( )
{  item x  ;   // x is an object
 //complete the program …
}
```

To access the above functions from main function use:

Object-name . function-name (actual-arguments);

To access a member of a class use:

Object-name . member

The dot (**.**) operator is also called (*class member access operator*).

Therefore, the function call statement will be

x **.** getdata (6 , 75.5) ;

While the following statement

x . number =100 ; // is illegal

Since ***number*** declared as private in the class therefore it could be accessed only through a member function and not by the **object** directly.
The objects communicate by sending and receiving messages. This is achieved through the member function, for example:

x.putdata( ) ; // Send message to the object x to display its data.

While the variables which are defined in public section of class could be accessed by the object directly.

Example 2:

```
# include < iostream.h >
class xyz
{  int   x ;        int   y  ;

 public:  int z;

 };

main( )
{  xyz   p ;

  p.x = 0 ;   // error  x  is private

  p.z = 10 ;  // ok,  z is pubic
 }
```

Function Definition:

1- Inside Class.
2- Outside Class.

Inside Class:

Example 3: Class of Rectangle

```
# include < iostream.h >
class Rectangle
{ public: int length , width ;
          int area ( )
            { return   length * width  ;  }
} ;
main ( )
{ Rectangle R ;
  R.length = 8 ;
  R.width =  5 ;
  cout << R.area( ) ; }
```

H.W. Write a program with a rectangle class whose its length and width are private (note: that you should write an additional function to enter length and width).

Outside Class:

The general form is:

Return type Class-name :: function-name(argument declaration)
{
Function body
}

Note: the operator (**::**) can be referred to as *Scope Resolution Operator*.

Example 4: Write a program to use a model of employee class (its data are: name, age, and department. Its functions are input( ) and display( ).
We can write the function definitions outside the class.

```
# include < itostream.h >
class employee
{ private:
    char name [30] ;
    int age ;
    char dept [10] ;
  public: void input ( ) ;
          void display ( ) ;
} ;
void employee :: input ( )
{ cin >> name ;
```

```
  cin >> age ;
  cin >> dept ; }

void employee :: display ( )
{  cout << name << endl << age  << endl << dept  ;  }


main ( )
{ employee E ;
  E . input ( ) ;
  E . display ( ) ;
 }
```

Note: if more than one object be created from the same class the process is called ***multiple objects***.

Example:

```
employee   A ,  B  ;
```

Example 5:

```
# include < iostream.h >
class employee
{ private:   char name [30] ;
             int age ;
             float salary ;
  public:

        void getdata ( )
         { cin >> name ;
           cin >> age ;
           cin >> salary ; }

        void putdata ( )
         { cout << name << endl ;
           cout << age << endl ;
           cout << salary << endl ; }
};

 void main ( )
{ employee doctor , nurse , worker ;
```

```
  doctor . getdata ( ) ;
  nurse . getdata ( ) ;
  worker . getdata ( ) ;

  doctor . putdata ( ) ;
  nurse . putdata ( ) ;
  worker . putdata ( ) ;     }
```

Example 6:

```
# include < iostream.h >
class set
{ private: int a , b ;
           void input ( )
           { cin >> a;
             cin >> b ; }
           void outp ( )
           { cout << a << endl ;
             cout << b << endl ;  }
 } ;
void main ( )
{  set   S  ;
  S . input ( ) ; // error because input is private function can not be accessed in main function

  S . outp ( ) ; // error because outp is private function can not be accessed in main function
}
```

* In order to avoid the above errors, the functions (*input* and *outp*) must be defined as public in class *set*. In such case, we can use these functions in main( ).

Memory Allocation of a Class

Once the member functions are defined as a part of a class, they are placed in the memory space.

Since all objects of the same class use the same member functions, no separate space is allocated for member functions when the objects are created. For each object, separate memory locations are allocated only for member data because member variables hold different data values for different objects.