CLASS ⇨ A class represents a group of data together with associated functions, describing an object. In C++, class makes a type that is used to create objects of this type. Classes are needed to make software representing real world objects that have data nd associated operations.

DECLARATION OF CLASS ⇨ The general form for defining a class is as given below:

```
Class class_name { private :
                       variable declarations ;
                       functions declarations ;
                   protected :
                       variable declarations ;
                       functions declarations ;
                   private :
                       variable declarations ;
                       functions declarations ;
                 } ;
```

Here, class_name is the tag-name of the class which acts as type – specifier by which we can create objects of that class type.

The declaration of variable and functions inside the class are called members of class. Members are of three types: private, public and protected. The private members can be accessed only form within the class and public members can be accessed both from within and outside the class. If no level for class is given, then by default, members are private.

Protected members are the members that can be used only by members functions and friends of the class in which it is declared. Protected members cannot be accessed from non – member functions.

Ex:
```
class Library {   int accession_no ;
                  char book_name[25] ;
                  char author_name[25] ;
                  int book_issued ;
                  void add_new_book(void) ;
                public :
                  search_book( ) ;
                  void print_book_info( ) ;
                } ;
```

CLASS METHOD DEFINITION ⇨ The codes for member functions related to class are called class method-definition. Member functions can be defined in two places:

① Outside the class definition ② Inside the class definition

The member function definition outside the class is written as:

```
Class_name : : function_name(parameter list)
{       …….
        ……. //function body.
        …….
}
```

The symbol : : is called scope resolution operator. It indicates that the scope of the function is restricted to the class_name.

Ex:
```
void Library : : read_info( )
        {  cout<< "Enter Accession no:"
           cin>> accession_no ;
           cout<< "\n Enter Book name :" ;
```

```
        gets(book_name) ;
        cout<< "\n Enter Author name:" ;
        gets(author_name) ;
    }
```

When the member function is defined inside a class, it is written as inline function within the class body. All the restrictions of inline function must be followed as well.

Ex:
```
class Salesboy { int salesboy_no ;
                 char salesboy_name ;
                 int product_no ;
                 float target, sales_made, commission ;
                 void calc_comm(void)
                  { float comm. = 0 ;
                    if(target <= slaes_made)
                       comm = slaes_made*0.50 ;
                       comm. = comm ;
                  }
                 public :
                       void print_data( )
                       { cout<< "Salesboy No :" << slaesboy_no<<endl ;
                         cout<< "Salesboy Name :" <<salesboy_naem <<endl ;
                         cout<< "Commission = " <<commission <<endl ;
                       }
               } ; //class definition ended.
```

Note:✍ We must note that the data members must appear before function members, under public, private or protected member type.

REFERENCING CLASS – MEMBERS ⇨ For referencing or using a member of class, we have to declare the class variable. In C++, a class variable is known as an object.

Ex:
```
class Suad {    int x, y ; //private by default.
             public :
                int z ;
                int Sub(int a, int b)
                 { int c = a – b ;
                   return c ;
                 }
                int Add(int a, int b)
                 { int c = a + b ;
                   return c ;
                 }
           } ; // class definition over.
void main( )
{  Suad  Ob1, Ob2 ; //Ob1 and Ob2 are class-variables i.e. objects of class Suad type.
   ……..
   ……..    }
```

The public data can be accessed through the objects of that class. The general format for calling a member function is:

Object_name • function_name(actual parameters) ;

Ex:
```
Ob1 • Add(4, 3) ;
Ob2 • Sub(47, 14) ;
```

Note that calling a member function through an object is also known as sending message to the object. Similarly, the general format for calling a member data is:

Object_name • public data_name ;

Ex:
```
Ob2 • z ;
```

Ob1 • z ;

Now let us, understand the error if any in the following statements:

Ob2 • z = 16 ; //valid	Ob2 • y = 3 ; //invalid
Ob1 • x = 5 ; //invalid	Sub(17, 2) ; //invalid
Ob1 • z = 7 ; //valid	Ob2 • Add(4, 3) ; //valid
	Ob1 • Sub(7, 2) ; //valid

GLOBAL AND LOCAL CLASS, GLOBAL AND LOCAL OBJECTS ⇨ A class is said to be global class if it is defined outside the bodies of all functions in a program. A class is said to be local class if it is defined inside a function body.

An object is said global object if it is declared outside all the function bodies. Global objects are globally available to all function in the program i.e. global objects can be used anywhere in the program. We must remember that a global object can be declared using a global class type only.

An object is said local object if it is declared within a function. Local objects are locally available to the function that declares it. We must remember that local objects cannot be used outside the function declaring it.

Let us understand all these concepts by following raw programs:

Ex-1:
```
#include<iostream.h>
class Pro                    //Global class named Pro
{   ……….
    ……….
 } ;
Pro Ob1 ;                    //Clobal object of type Pro
void main( )
{  Pro Ob2 ;                 //Local object o type Pro
   ……..
   ……..
}                            //End of main( ) function
int Fab(int a)
{  Pro Ob3 ;                 //Loal object of type Pro
   ……….
   ……….
}
```

Ex-2:
```
#include<iostream.h>
   ……….
   ……….
void main( )
{   class Pre                //Local class named Pre
        {  ……..
           ……..
    Pre Ob1 ;                //Local object of type Pre
    ……
    ……
 }
 int Tab(int a, int b)
 {  Pre Ob2 ;                //Invalid as Pre is a local class. Hence, Ob2
    ……                       //cannot be declared at all.
    ……
 }
```

Ex-3:
```
#include<iostream.h>
{  class Per                          //global class named Per
        {  public:
              int a ;
```

```
                    int Tom(int a, int b)  } ;
        Per Ob1 ;                          //global object
    int main( )
    {  Ob1 • a = 10 ;                      //valid as Ob1 is globally available
       Ob1 • Tom(5, 7) ;                   //valid
       ……
       ……
    }
    int Teem(int a)
    {  Ob1 • a = 20 ;                      //valid as Ob1 is globally available
       Ob1 • Tom(3, 7) ;                   //valid
       ……
       ……
    }
```

Ex-4:

```
#include<iostream.h>
class Fe                                   //global class named Fe
        {  public :
                    int a ;
                    void Bi(void) ;
        } ;
void main( )
{  class Fo                                //Local class named Fo
          {  public :
                    int I ;
                    void Bo(void) ;
           } ;
    Fe Ob1 ;                               //Local object Ob1 of global class type Fe
    Fo Ob2 ;                               //Local object of local class type Fo
    Ob1 • a = 5 ;                          //valid
    Ob2 • Bi( ) ;                          //valid
    Ob2 • I = 15 ;                         //valid
    Ob2 • Bo( ) ;                          //valid
    ……
    ……
}
void Bo(void)
{   Fe ob3 ;                               //Local o bject Ob3 of Fo class type
    Fo Ob4 ;        //Invalid because Fo is local class, hence not available to function Bo
    Ob3 • a = 25 ;                         //valid
    Ob3 • Bi( ) ;                          //valid
    Ob1 • a = 10 ;                         //invalid as Ob1 is local object.
    Ob2 • Bo( ) ;                          //invalid as Ob2 is local object.
}
```

Note:✍ A local object can be created from both class type, global as well as local.

SCOPE RULE OF CLASSES ⇨

Element	Scope	Description
Global class	Global scope	Global class is available to all the functions. Objects of this class type can be created anywhere in the program
Local class	Local scope	Local class is locally available to the function in which class is defined. Object of this type can be created only in the function in which the class is defined.
Global object	Global scope	Global object can be used anywhere in the program.

Element	Scope	Description
Local object	Local scope	Local object can be used only within the function where it is declared
Private members	Class scope	Private members can be used only within the class. These cannot be accessed directly with the help of objects.
Protected members	Class scope	Private members can be used only within the class. These cannot be accessed directly with the help of objects.
Public members	Global for global object, Local for local objects	The scope of public members depends upon the referencing object. If referencing object is local, the scope of public member is local. If the referencing object is global, the scope of public member is global.

TYPES OF MEMBER FUNCTIONS ⇨ There are three types of member functions – ① Accessor functions, ② Mutator functions and ③ Manager functions.

The member functions by which data members can be accessed is called accessor function. Accessor function cannot change the value of data member. Accessor functions are used to read values of private data members of the class.

The member functions by which the value of data member cab be changed is called mutator function.

The member functions by which constructor and destructor and are made, is called manager functions.

Ex:
```
class student{  private :
                    int rollno ;
                    char name[25] ;
                    float marks ; char grade ;
                public :
                    void read info( )
                    { ……….          //code for obtaining data members' values by usrer
                      ……….
                    }
                    void shoinfo( )
                    { ……….          //code for the display of data members
                      ……….
                    }
                    int getRollno( )              //accessor function
                      { return rollno ;  }
                    float getmarks( )             //accessor function
                      { return makrs ;  }
                    void calgrade( )              //mutator function
                      { if((marks >=75)  &&  (marks< 69=0))
                            grade = 'A' ;
                        else if((marks>=60) && (marks<50))
                            grade = 'B' ;
                        else grade ' 'C' ;
                      }
                } ;
```

Note-1:✍ Sometimes, the scope resolution operator (: :) is used to give full name (known as qualified name) to the variables, as is given in the following example :

```
class M {  int x ;
           int y ;
               public :
                   void read( ) const ; //const function that can't change any data in the function.
                   ……
                   ……
```

```
};
```

The qualified name or full name of variable x, y and function read( ) are:

M : : x , M : : y and M : : read( ) respectively.

Note-2:✍ If a global variable is declared as a data member, then such variable becomes hidden, as is given in the following example:

```
……
……
int a, x ;                  //a and x are global variables
class {  int x ;            //global x becomes hidden now
         ……
         ……
       } ;
```

Let us see the other example:

```
int x, y ; //x and y are global variables
class M{ public :
                int x ;          //global x is hidden
                void f(int i)
                { x = I ;        //it assigns I to M : : x, not to global x
                  y = I ;        //it assigns I to the global y
                  : : x = I ;    //it assings I to global x. here : : operator uncovers a
           } ;                      hidden file scope of global item.
```

MEMORY ALLOCATION OF MEMBER FUNCTIONS AND OBJECTS ⇨ The memory space is allocated for member functions when the class having these member function is defined. The memory space is allocated for objects data member when the objects are declared.

ARRAY OF OBJECTS ⇨ A array having class type element is known as array of objects. Declaration of array of objects is illustrated in the following program snippet :

```
Class A {  int x, y ;
           Int Add(int a, int b)
              { a = x; b = y ;
                return(a + b)
              }
            public :
              int l, m ;
              int Mul(int c, int d)
                 { c = l ; d = m ;
                   return(c*d) ;  }
         } ;
A Item[ 7 ] ;  //Array of 7 objects of class A type is declared.
```

OBJECTS AS FUNCTION ARGUMENTS ⇨ We know the objects are class variable. Hence, like any other variable, objects may also be passed as function argument. Objects as function argument are also called in two ways : (i) call by value method and (ii) call by reference method.

When an object is called/passed by value, the function creates its own copy of the object and acts upon its own copy., hence, any change done to the objects inside the function do not reflect back in the original object.

When an object is called/passed by reference its memory address is passed to the function. Hence, any change done to the object inside the function is reflected back in the original object also.

ψψψψ THE END OF THE CHAPTER ψψψψ