Chapter 2

CLASSES AND OBJECTS

Objectives:

- Defining classes & objects.
- Declaring & using static data member & static member function, friend function.
- Programs based on classes & objects.

What to learn

- 2.1 Structures in C++.
- 2.2 Class & Object: Introduction, specifying a class, access specifies, defining member functions, creating Objects, memory allocations for objects.
- 2.3 Array of Objects, Object as function arguments.
- 2.4 Static data members, static member function, friend Function

Concept Explanation: Class

- Class is an important feature of C++.
- It is an extension of the idea of the structure in ‘C’.
- Class is a new way of creating and implementing a user defined data type.
- Class is similar to the structure data type, but structure in ‘C’ is having some limitations.
- In C++ class and structure is similar in that, both can have data members as well member functions.
- The only difference between the structure in C++ and class in C++ is that, by default the members are private in class where as by default members are public in structure

Introduction

Specifying a class:

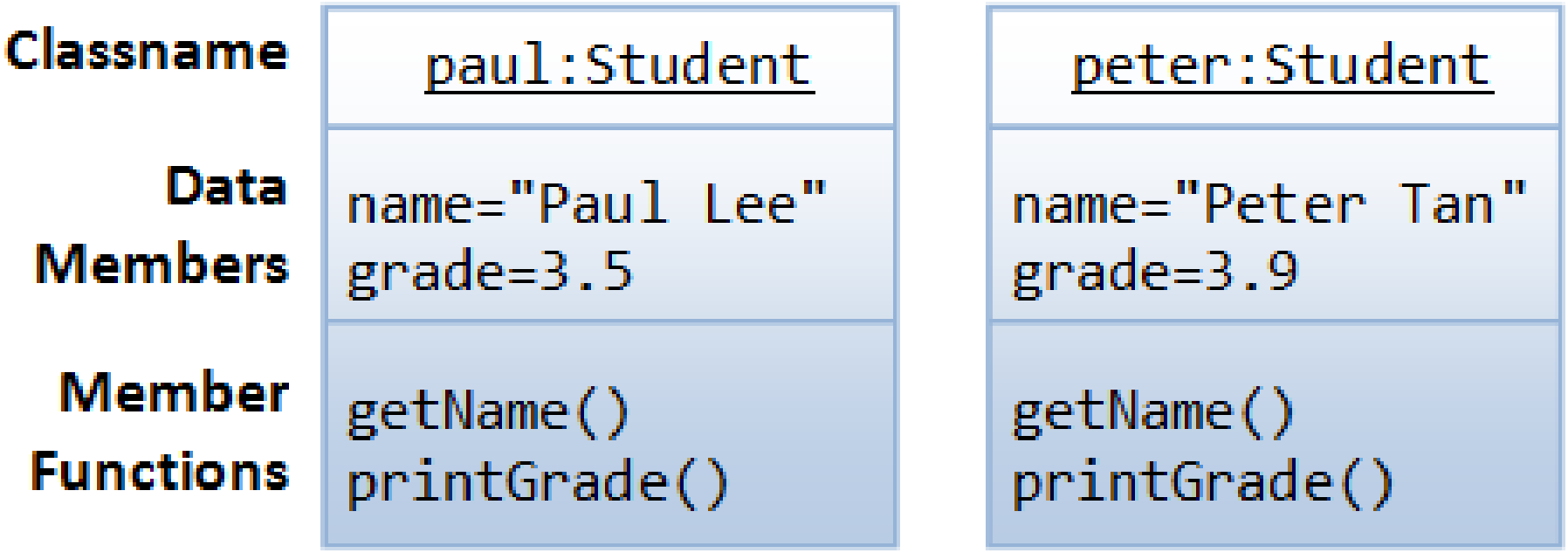
- The class is a way of binding data and functions associated with data together.
- The entire set of data and the functions can be made user defined data type with help of class. Thus Class is nothing but a user defined data type.
- Class act as a template, which is used to create objects. The class contains data and member functions that operate on the data.
- It allows the data to be hidden, if necessary from the external use.
- **Generally class specification has two parts**
- class declaration
- class function definitions.

class

- They are usually grouped under two sections, private and public.
- The keyword private and public are called visibility labels.
- The private members can be accessed only from within the class and the public members can be accessed from outside the class.
- The use of keyword private of optional. By default all the members of class are private.
- Only the member functions can access the private members of class.

class

- The class declaration describes the type and scope of its members.
- The class function definitions describe how the class functions are implemented.
- The class body contains the declarations of variables and functions. These variables and functions are collectively called members.
- The variables are called data members or member variables and the functions are called member functions.



Two instances of the **Student** class

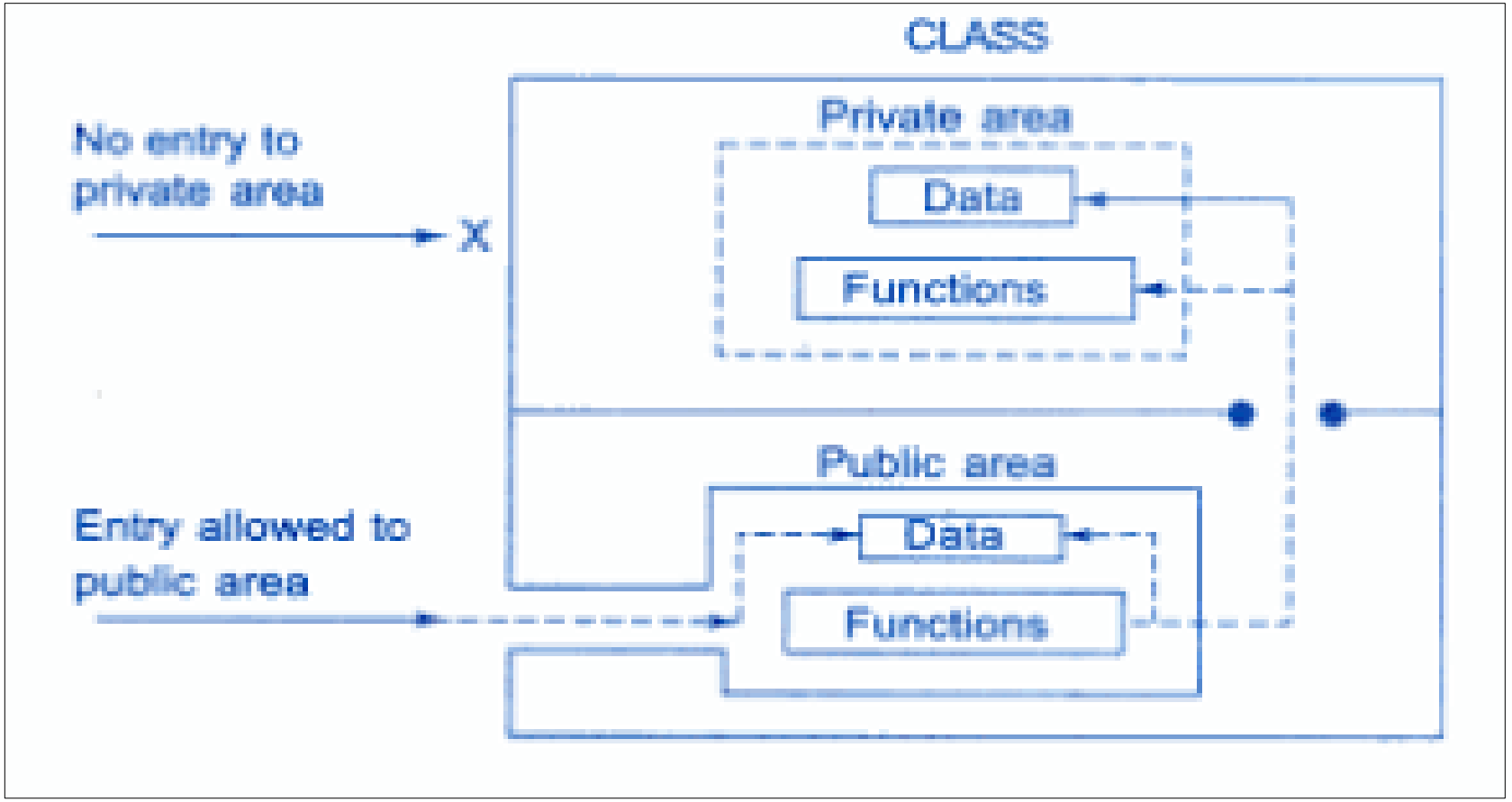
Syntax of class

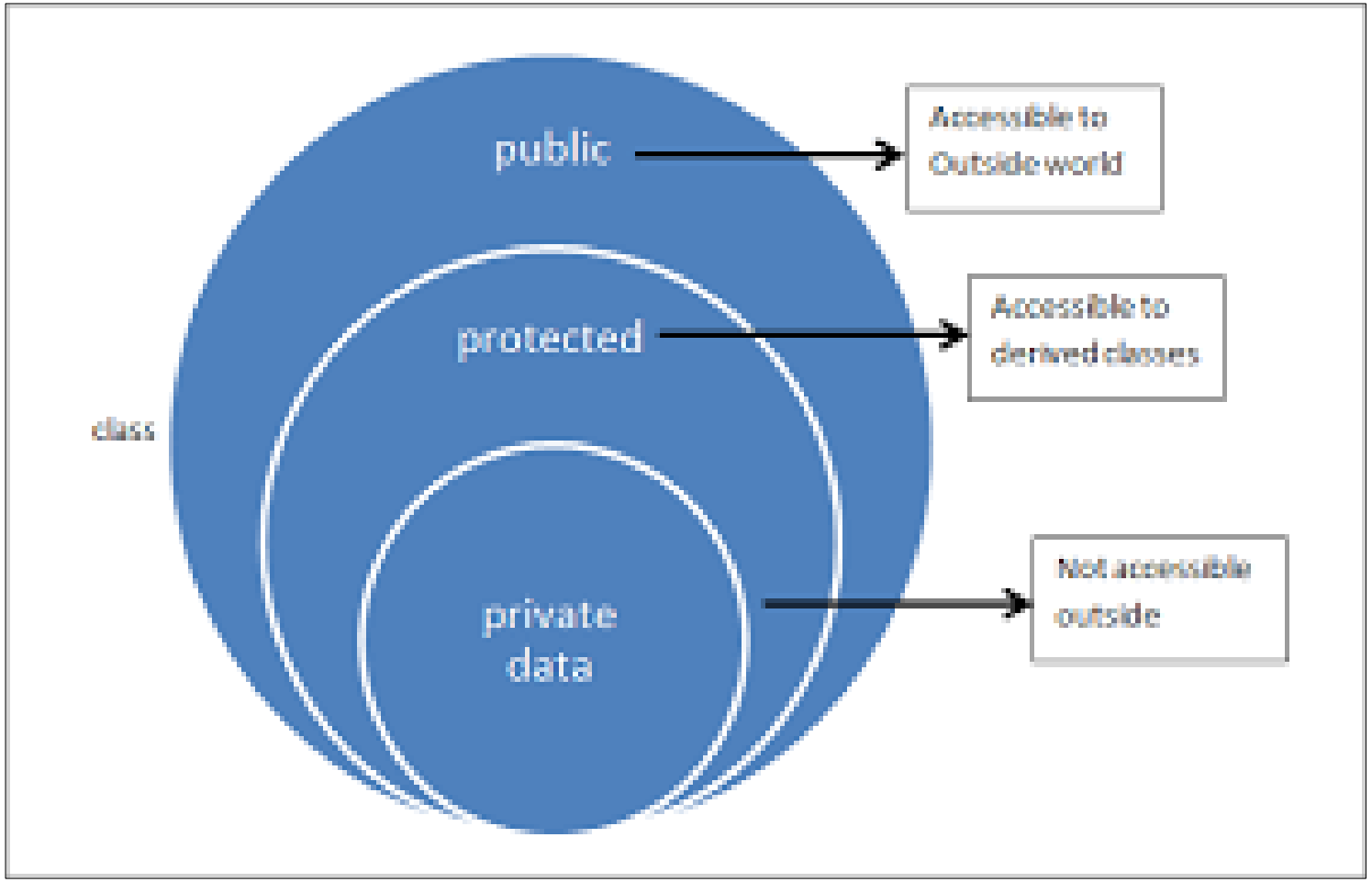
```
class class-name
{
 private :
            variable declarations;
            function declarations;
 public:
            variable declarations;
            function declarations;
};
```

Access specifiers

- In C++ Access specifies are also called as visibility mode. They are as follows: Public, Private, and Protected .

1. Public:- When access specifier is public i.e. base class is publicly inherited by derived class all public member of base class become public members of derived class and all protected members become protected in derived class.
2. Private: - All public and protected members of base class become private members of derived class.
3. Protected: - In this all public and protected members of base class become protected of derived class.


CLASS
Private area
Data
Functions
No entry to
private area
X
Public area
Data
Functions
Entry allowed to
public area


public
Accessible to Outside world
protected
Accessible to derived classes
class
private data
Not accessible outside

DATA HIDING

OOP emphasis on data rather than procedures. The primary focus is on Data.

In OOP the data and the functions that operate on the data are encapsulated in to a single unit known as object.

Thus object contains data and the member functions that operate on the data.

Only the member functions can access the private data of the class.

The private data is not accessible to the external functions.

This insulation of data from the external functions is called Data Hiding or Information Hiding.

Syntax:

```
class class-name
{
 private :
            variable declarations;
            function declarations;
 public:
            variable declarations;
            function declarations;
};
```

Example :

```
class student
{
private :
    int rollno;                    // data member
    char name[30];  // data member
public:
    void getdata (void);
// member function
    void showdata(void);
// member function
};
```

CREATING VARIABLES / OBJECTS OF CLASS:

After declaration of class, one can create any number of variables (objects) of that type by using class name.

Syntax: Class-name variable-name;

Example: Student s1; // memory for s1 is created.

The above statement will create a variable ‘s1’ of type student. These variables are known as objects. Here S1 is an object of class student.

One may declare more than one variable as shown.

Student s1, s2, s3;

Data member Declaration

- Data members should be declared in private access specifier of a class

Syntax:

- Data_type variable_name;

e.g.

```
 int a;
char b;
int roll_no;
float salary;
Int emp_no;
Char name[20];
```

Member Function Declaration

Member function declared in public access specifier

Syntax:

```
Return type  Function_name(arguments)
{
Statements;
}
```

e.g.

```
void getdata(void)
{
cout<<“\n Enter a number”;
}
```

ACCESSING CLASS MEMBERS

The private data of the class can be accessed only through the member functions of that class.

To call a member function use following syntax,

Syntax: Object-name. function-name (argument-list);

E.g. If s1 is an object of student class and showdata() is a member function of the class student , then to invoke a member function we write,

s1. showdata();

Access specifiers

- In C++ Access specifies are also called as visibility mode. They are as follows: Public, Private, and Protected .
- Public:- When access specifier is public. public data members can access outside the world.
- Private: - Default Mode is private. Private data members can not access outside the world.
- Protected: - In future at time of Inheritance. In this all public and protected members of base class become protected of derived class.

Creating Objects Of Class:

- After declaration of class, one can create any number of variables (objects) of that type by using class name.
- Syntax: Class-name variable-name:
- Example: Student s1; // memory for s1 is created.
- The above statement will create a variable ‘s1’ of type student. These variables are known as objects. Here S1 is an object of class student.
- One may declare more than one variable as shown.
- Student s1, s2, s3;

- The private data of the class can be accessed only through the member functions of that class.
- To call a member function use following syntax,
- Syntax: Object-name. function-name (argument-list);
- (.) Operator is called as member access operator which will call all data member and member functions from public access specifiers which declared in class
- E.g. If s1 is an object of student class and showdata() is a member function of the class student , then to invoke a member function we write,

  s1. showdata();

E.g. student s1,s2;

```
s1.getdata();
s1. showdata();
s2.getdata();
s2. showdata();
```

EXAMPLE:

```
Class student
{
private :
int rollno;
char name[30];
public:  int no-of-student;
};
main()
{
  student s; // create an object
  s. rollno = 10                    //Error , rollno is private member , cannot be accesses
     directly
  s. no_of_student =100;        // OK  no_of_student is public member , can be accessed
     directly
  return (0);
}
```

NOTE: A variable declared as public can be accessed directly but the private variables cannot be accessed by the object directly. Private member can be accessed using member functions.

Syntax of class

```
class class-name
{
 private :
            variable declarations;
            function declarations;
 public:
            variable declarations;
            function declarations;
};
void main()
{
 classname  objects;
objects.member function();
}
```

Simple C++ Program of class - Example

Vidyalankar Polytechnic

Write a program to declare a class 'Student' containing data member's name, roll_no, and percentage. Declare getdata() and showdata() function and display and accept this data for 1 object of the class.

```
#include<iostream>
using namespace std;
class Student     //class declaration
{
private:
char name[20];   //data member1
int roll_no;     //data member2
float percentage;//data member3
public:
void getdata()  //member function1
{
cout<<"\n Enter Student name : ";
cin>>name;
cout<<"\n Enter  student roll  number ";
cin>>roll_no;
cout<<"\n Enter  percentage : ";
cin>>percentage;
}
void showdata()  //member function2
{
cout<<"\n Student name is"<<name;
cout<<"\n Student roll no is "<<roll_no;
cout<<"\n  Student percentage is : "<<percentage;
}
};   //class termination
```

```
int  main()
{
    Student  s;   //creating objects
    s.getdata();  //calling member function1-getdata()
    s.showdata(); //calling member function1-getdata()
}
```

```
Enter Student name : ganesh

Enter  student roll  number 23

Enter  percentage : 89.65

Student name isganesh
Student roll no is 23
 Student percentage is : 89.65

...Program finished with exit code 0
Press ENTER to exit console.
```

Simple C++ Program of class - Example

Vidyalankar Polytechnic

Write a C++ program to find the area of circle using class circle which have following details:

a. Accept radius from the user b. Calculate the area c. Display the result

```
**********************************************************************
#include<iostream> //header File
using namespace std;
class circle //class decalration
{
float radius; //data member1
float area;   //data member1
public:
void calculate() //member function1
{
cout<<"\n Enter the value of Radius : ";
cin>>radius;
}
void display() //member function1
{
area = 3.14 * radius * radius;
cout<<"\n Area of Circle : "<<area;
}
};  //class termination
int main() // main function
{
circle cr;//creating objects
cr.calculate(); // calling member function as calculate()
cr.display();   // calling member function as display()
return 0;
}
```

```
Enter the value of Radius : 10.5

Area of Circle : 346.185

...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment

1) Write a program to declare a class ‘Person’ containing data member’s name, address, and mobile_number. Declare getdata() function to accept data of one person and display we help of showdata() function.
2) Write a program to declare a class ‘Person’ containing data member’s name, address, and mobile_number.Declare getdata() function to accept data of two person’s data and display with the help of showdata() function.
3) Write a C++ program to define a class employee having members Emp-id, Emp-name, basic salary and functions accept() and display(). Calculate DA=25% of basic salary, HRA=800, I-tax=15% of basic salary. Display the payslip using appropriate output format.
4) Write a program to declare a class ‘emp’ containing data member’s emp_id and salary. Accept and display this data for 1 object of the class.

Note:Run all programs on turboc++ or dev c++ or gdb complier complier with given syntax.

DEFINING MEMBER FUNCTIONS:

- The functions, which are declared inside the class, are called member functions of that class.
- Only member functions can access the private members of the class.
- Generally member functions are placed in a public section. But they can be placed in a private section also.
- Member function can be defined in two places.
- Outside the class definition
- Inside the class definition

Syntax of inside member function declaration in a class

- Syntax:

```
class class-name
{
public:
Return_type member function name(Parameters list or argument)
{
Statements;
}
};
```

- where class is keyword and class name is name of class want to declare
- Return type is return type of member function that can be void, int ,char, float
- Member function is the name of function that want to declare
- Parameter list or arguments want to give to your function.
- Argument syntax can be

(Data_type1 variable name1, Data_type2 variable name2,….., Data_typen variable namen)

Example of inside member function declaration in a class

- **Write a program to declare a class ‘Emp’ containing data member’s emp_id and salary. Accept and display this data for 1 object of the class.(inside member declaration)**

```cpp
#include<iostream>
using namespace std;
class Emp
{
private:
        int emp_id;
        float salary;

public:
        void accept()
        {
                cout<<"\n Enter Employee id : ";
                cin>>emp_id;
                cout<<"\n Enter  Salary : ";
                cin>>salary;
        }
void showdata()
        {
                cout<<"\n Employee  id is "<<emp_id;
                cout<<"\n Employee salary is  : "<<salary;
          }
};
```

```cpp
int  main()
{
        Emp e;
        e.accept();
        e.showdata();
  }
```

Output

```
Enter Employee id : 1

Enter  Salary : 45000

Employee  id is 1
Employee salary is  : 45000

...Program finished with exit code 0
Press ENTER to exit console.
```

Defining member function out-side the class definition

In this, the member functions that are declared inside the class have to be defined outside the class.

The member function incorporates a' membership identity label' in the header.

This label tells the compiler which class the function belongs to.

The general syntax of defining any member function outside the class is as follows:

```
Return- data-Type   class-name :: function-name (argument –declaration)
{
function Body
}
```

- Syntax:

Return- data-Type class-name **::** function-name (argument – declaration)
{
function Body
}

Syntax of Outside member function declaration in a class

- In this, the member functions that are declared inside the class must be defined outside the class.
- The member function incorporates a' membership identity label' in the header.
- This label tells the compiler which class the function belongs to.
- The general syntax of defining any member function outside the class is as follows

Syntax:

```
class class-name
{
public:
Return_type member function name(parameters list or argument);
};
Return_type  class_name :: member function name(parameters list or argument);
{
Statements;
}
```

The membership label 'class-name**::** ' tells the compiler that the function belongs to the spiffed class. The symbol "**::**" is called scope resolution operator.

Example of outside member function declaration

```
class student
{
 private :int roll-no;
                float marks;
public :void  getdata (void);                // member  function declaration
                void  showdata (void);          // member function declaration
};
void student :: getdata (void)                                  // member function definition outside the class
{
cout<< ”Enter  Roll No : “ ;
   cin >> roll-no;
   cout<< ”Enter  Marks  : “ ;
   cin >> marks;
}
void student :: showdata (void)         // member function definition outside the class
{  cout<<” Roll No =  “ <<  roll-no << endl;
   cout<<”  Marks  = “ <<  marks << endl;
}
```

Example of Outside member function declaration in a class

- **Write a program to declare a class 'Emp' containing data member's emp_id and salary. Accept and display this data for 1 object of the class.**

```
#include<iostream>
using namespace std;
class Emp
{
private:
        int emp_id;    //data member1
        float salary;   //data member2

public:
        void accept();     //member function1
         void showdata(); //member function2

};
void Emp::accept()
        {
                cout<<"\n Enter Employee id : ";
                cin>>emp_id;
                cout<<"\n Enter  Salary : ";
                cin>>salary;

        }
```

```
void Emp::showdata()
        {
                cout<<"\n Employee  id is "<<emp_id;
                cout<<"\n Employee salary is  : "<<salary;
         }
int  main()
{
        Emp e;
        e.accept();
        e.showdata();
        return 0;
 }
```

```
C:\Users\SB-PC\Desktop\emp.exe

Enter Employee id : 1

Enter  Salary : 34000

Employee  id is 1
Employee salary is  : 34000
--------------------------------
Process exited after 4.828 seconds with return value 0
Press any key to continue . . .
```

Example :

```
Class student
{
 private :int roll-no;
      float marks;
public :void  getdata (void);    // member  function declaration
      void  showdata (void); // member function declaration
};
void student :: getdata (void)  // member function definition outside the class
{
cout<< ”Enter  Roll No : “ ;
   cin >> roll-no;
   cout<< ”Enter  Marks  : “ ;
   cin >> marks;
}
void student :: showdata (void)        // member function definition outside the class
{
cout<<” Roll No =  “ <<  roll-no << endl;
   cout<<”  Marks  = “ <<  marks << endl;
}
```

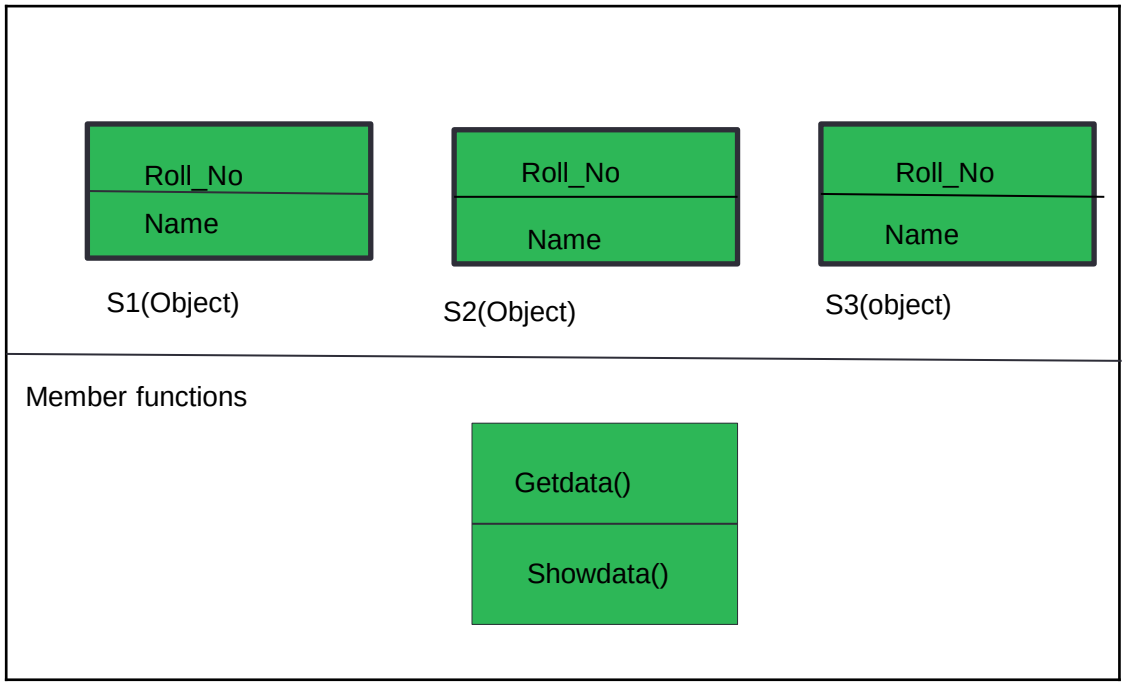
CHARACTERISTICS OF MEMBER FUNCTION:

Use	Access	Call
Several different classes can use the function name. The 'membership label – (class-name ::)' will resolve their scope.	•Member functions can access the private data of the class. ○A non member function cannot do so. However friend function can access the private data. This is an exceptional case.	A member function can call another member function directly without using the dot (.) operator.

Memory allocation for objects

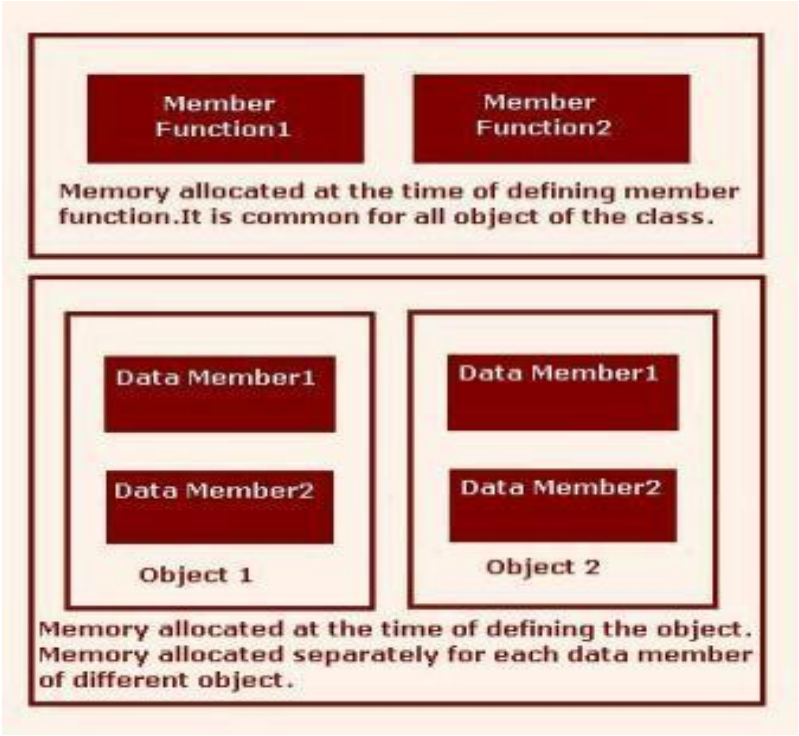
- Memory space for object is allocated when they are declared and not when the class is specified.
- But the member functions are created and placed in the memory space only once when they are defined as a part of the class specification.
- As all objects belongs to same class, use the same member functions, no separate memory is allocated for member functions when the objects are created i.e. when the objects are created the memory space is allocated only for data members (member variables)
- Space for data members is allocated separately for each object.
- The memory locations are also distinct for object of same class because the data members of different object will hold different values.
- Data member of the class can contain different value for the different object, memory allocation is performed separately for each data member for different object at the time of creating an object.
- Member function remains common for all object. Memory allocation is done only once for member function at the time of defining it.

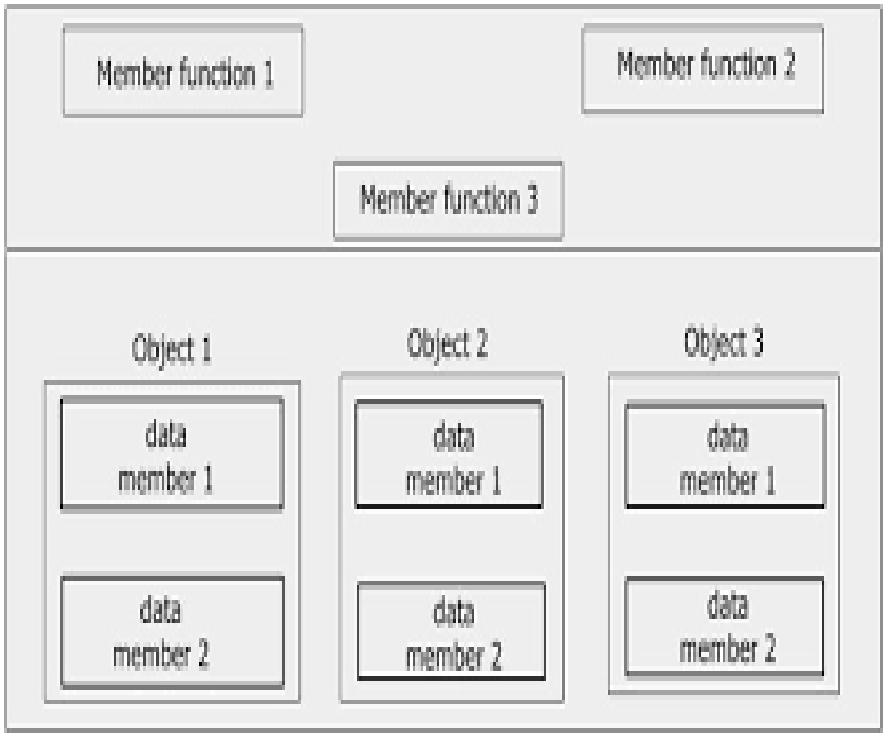
Example



- Consider class student having two data member and three member functions.
- Student s2,s2,s3; three objects for class student.
- Data members Roll_No and name required separate memory space as per obect.
- Objcet s1 requires separate meory as so on
- Member functiond same for all objects.getdta and showdata requires same meory space

Example:


Member Function1
Member Function2
Memory allocated at the time of defining member function.It is common for all object of the class.
Data Member1
Data Member2
Object 1
Data Member1
Data Member2
Object 2
Memory allocated at the time of defining the object.
Memory allocated separately for each data member of different object.


Member function 1
Member function 2
Member function 3
Object 1
Object 2
Object 3
data member 1
data member 2
data member 1
data member 2
data member 1
data member 2

- Functions are used to avoid repetitive task and also to save some memory space. However each time a function is called, it takes a lot of extra time in executing a series of instructions for task such as jumping to the function, saving registers, pushing arguments into the stacks and returning to the calling function.
- To eliminate these problems C++ provides a new feature called inline function.
- " An inline function is a function that is expanded in line when it is invoked." That is the compiler replaces each function call with the corresponding function code.
- The functions defined inside the class definition are treated as inline.
- One can define the function outside the class and still make it inline using inline key word.

- Advantages of Inline functions:
  - In line expansion makes a program run faster.
- Disadvantage of inline functions:
  - Takes more memory
- NOTE: The functions containing for loop or a function with a larger code are not made inline.
- If you make them inline compiler gives you a warning.

- Syntax : inline return- data-type function-name (Argument declaration)

```
{
  function Body
}
```

- Example:

```
inline double cube ( double a)
{
  return ( a * a * a);
}
```

```
Class student
{
    int rollno ;
public :   void getdata (void );
};
inline void student :: getdata (void )          // defined outside the class but still
inline
{
        cout << “ENTER ROLL NO : “;
          cin >> rollno;
}
```

1) **Write a program to declare a class 'Person' containing data member's name, address, and mobile_number. Declare getdata() function to accept data of one person and display with the help of showdata() function.**
2) **Write a C++ program to define a class employee having members Emp-id, Emp-name, basic salary and functions accept() and display(). Calculate DA=25% of basic salary, HRA=800, I-tax=15% of basic salary. Display the payslip using appropriate output format.**
3) **Write a program to declare a class 'Student' containing data member's name, age, percentage . Accept and display this data for 1 object of the class.**

Note: Run all programs on turbo c++ or dev c++ complier with given syntax.

ARRAYS WITHIN A CLASS:

- Arrays can be used as a member variable in a class.
- It can be declared as private or public member of class and can be used by the member functions like any other member variables.

E.g. class array

```
{       int rollno [10];
 public :
    void setvalue (void );
    void display (void );
};
```

One can perform any operation on array. In the above example setvalue() function sets the values of array rollno[ ] and display() function displays the values

Write a program to find out the sum of numbers stored in array.

```
# include <iostream.h>
class array
{
  private :  int a [10] ,n ;    // array as a data member
 public :void getdata( void );
void add (void );
};
void array  : : getdata (void )
{  cout << “Enter how many numbers ?“;
 cin >> n;
 cout << “Enter “ << n << “numbers : “ << end l;
 for ( int i =0 ; i <n ; i ++)
       cin >>a [ i ];
}
void array :: add ( void )
{   sum =0 ;
 for ( int i =0 ; i <n ; i ++)
 {
  sum = sum + a [ i ];
 }
cout << “SUM = “ << sum;
}
void main ()
{ array stud;
  stud.getdata ();
  stud.add();
}
```

PROGRAMS BASED ON CLASS AND OBJECTS:

Write a program to declare a class 'emp' containing data member's emp_id and salary. Accept and display this data for 2 objects of the class.

```
# include <iostream.h>
class emp
{
 private :  int emp_id;
  float salary;
public:  void getdata (void)       // Inline function
          { cout <<"ENTER EMP ID "<< endl;
   cin>>emp_id ;
   cout <<"ENTER SALARY"<<endl;
   cin >> salary;
           }
         void showdata (void)     // Inline function
         { cout <<"EMP ID =  "<< emp_id<<endl;
 cout <<"SALARY = "<< salary << endl;
        }
};  // END OF CLASS
int main()
{
emp e1,e2 ;  // creating 2 objects of class emp
e1.getdata();   // get data for object e1
e2.getdata();   // get data for object e2
e1.showdata(); // display data of object e1
e2.showdata();// display data of object e2
return (0);
}
```

write a program to define a class ‘rectangle’ having data member’s length and breadth. Accept this data for one object and display area and perimeter.

Write a program to declare a class product containing data members product_code , name and price . Accept and display this information for 2 objects.

Write a program to declare a class student containing data members roll_no and precentage. Accept this data for 2 objects and display the roll_no of the student having higher percentage.

Write a program to declare a class ‘account’ having data members principal, rate_of_interest and no_of_years. Accept this data for one object of a class and calculate Simple Interest using formulae (SI = (P*N*R) /100)

- The way we can create an array of built in data types, we can also create an array of objects of user defined data type It is possible to have array of objects of class.
- Consider the following example

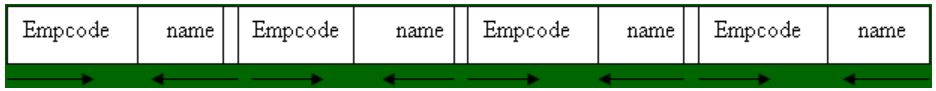
```
class employee
{
int empcode;
char name[40];
public :
 void accept(void) ;
 void display(void) ;
};
```

Contd…..

To declare an array of 4 employee objects we write.

Employee E[4]; // array of 4 employee

E[0] E[1] E [2] E [3]



To access the member function of individual objects.

E[0].accept(); // Invoking the accept() for E[0]

E[1].accept(); // Invoking the accept() for E[1]

Using for loop we can access all the members as follows

```
For ( int I=0 ; I<4 ; I++)
{
  E[ I ].accept();
}
```

Write a program to define a class 'account' having data members as account_no , name and balance. Accept and display this information for 10 objects of this class.

Vidyalankar Polytechnic

```
# include <iostream.h>
class account
{
 private :  int account_no;
                    char name [50];
                    float balance;
public :         void getdata ( void );// function
declaration
                    void showdata (void);      // function
declaration
};
void account :: getdata (void)// Outside function
definition
 { cout << "ENTER account no: " ;
 cin >> account_no;
 cout << "ENTER name : "
 cin>> name;
 cout << "ENTER balance : "
 cin >> balance;
}
void account :: showdata (void)  // outside function
 { cout << " Account no = " << account_no <<endl;
 cout << " Name = " << name <<endl;
 cout << " Balance = " << balance <<endl;
}
```

```
int main(void)
{
account  b [10 ] ;              // create array of 10 objects
  for ( int I =0 ; I < 10 ; I++ )
                     b [ I ] .getdata ();
   for ( I =0 ; I < 10 ; I++ )
                     b [ I ] .showdata ();
                    return 0;
}
```

Modify the above program to display only names of customers having balance >10000

Ans : Change the Showdata() function as shown below.

```
void account :: showdata (void)   // outside function
definition
{
  if ( balance > 10000)
 {
 cout << " Account no = " << account_no <<endl;
 cout << " Name = " << name <<endl;
 cout << " Balance = " << balance <<endl;
 }
}
```

Write a program to declare a class employee having data members as name and basic salary. Accept this data for 5 employees and calculate and display Gross Sarary. (Gross Salary = Basic + DA + HRA where DA = 44.5 % of basic HRA = 15% of basic)

```
# include <iostream.h>
Class employee
{
  private :
char name[30];
 float basic;
public:
void getdata (void)
         {
cout << “enter name “ ;
cin >> name;
cout << “enter basic salary “;
 cin>>basic;
         }
          void showdata(void)
         {
float GS , HRA ,DA ;
HRA = 0.445 * basic;
DA = 0.15 * basic;
GS = HRA + DA + basic;
 cout << “\n NAME = “ <<name ;
cout << “\n GROSS SALARY = “<< GS;

}
};
Int  main()
{
employee e [5];
for (int I=0 ; I<5 ; I++)
{
   e [ I ] .getdata();
}
for (I=0 ; I<5 ; I++)
{
   e [ I ]. showdata();
}
return 0;
}
```

Write a program to create a class worker having data members as name and skill ('plumber' , 'fitter' etc). Accept this details for 5 workers and display the names of workers having skill as plumber

Vidyalankar Polytechnic

```
# include <iostream.h>
# include <string.h>
class worker
{
 private :
char name[40];
     char skill[40];
public:
void accept ( void)
{
cout << "Enter name : " ;
 cin >> name;
 cout << "Enter skill"
 cin >> skill;
}
```

```
void display (void )
{
int i ;
 i = strcmp ( skill , "plumber");
 if (i == 0)
Cout << " \n NAME = " << name ;
}
};
void main ( void)
{
 worker w [5];
for ( int i=0 ; i< 5 ; i++)
      w[ i  ] . accept();
cout << "\n NAMES OF PLUMBERS ARE :
";
for ( int i=0 ; i< 5 ; i++)
    w[ i  ] . display();
}
```

Programming assignment

- **Write a program to declare a class student having data members as name and roll no. Accept this data for 5 students and display data for 5 students.**
- **Write a program to declare a class person having data members as name and age . Accept this data for 5 persons and display data for 5 persons.**
- **Write a program to create a class worker having data members as name and skill ('plumber' , 'fitter' etc). Accept this details for 5 workers and display the names of workers having skill as plumber**
- **Write a program to declare a class employee having data members as name and basic salary. Accept this data for 5 employees and calculate and display Gross Sarary.**

  (Gross Salary = Basic + DA + HRA where DA = 44.5 % of basic HRA = 15% of basic)

PASSING OBJECT AS ARGUMENT TO MEMBER FUNCTIONS:

- Object can be used as function arguments. The object can be passed to the function by two ways:
- Pass by values: A copy of entire object is passed to the function .
- Pass – by –reference : Only the address of the object is passed to the function
- As in first case, a copy of entire object is passed to the function. The function creates another copy of that object and therefore any changes made to the object inside the function do not affect the actual object. This method is also called Pass-by-value
- The second method is called Pass-by-reference. In this method, the address of the object is passed to the function. Any changes made to the object inside the functions are directly reflected to the actual object.

Advantage of Pass-By- Reference

- The pass-by-reference method is more efficient than pass-by value since it requires passing only the address of the object and not the entire copy. Thus it saves memory space.
- Any changes made in the functions are reflected to the actual object.

Pass by values

Adding complex numbers by passing two objects as arguments.

```
Class complex
{    int real,;
   float img;
public :
          void setdata ( int , float);
          void add ( complex , complex);
          void showdata(void);
};
void complex ::setdata ( int r, float I)
  {
          real = r;
          img =I;
  }
void  complex :: add ( complex c1 , complex c2) //takes two objects as argument c1 , c2
{
 real = c1.real + c2.real ;
 img = c1.img + c2.img;
}
void complex :: showdata (void)
{
 cout << real << "+" << "i" << img;
}
void main(void)
{
  complex c1,c2,c3;
  c1.setdata(6,-7);
  c2.setdata ( 3, 4);
  c3.add(c1,c2);   // c1 and c2 will be explicitly passed and result will be in c3.
  c3.showdata();
}
```

Adding complex numbers by passing only one object as argument.

```
Class complex
{    int real,;
   float img;
public :
          void setdata ( int , float);
          void add ( complex , complex);
          void showdata(void);
};
void complex ::setdata ( int r, float I)
  {
          real = r;
          img =I;
  }
 void  complex :: add ( complex c2)        // take only one object c2  as argument
 {
  real = real + c2.real ;
  img = img + c2.img;
 }
 Note that here we are passing only one object c2 as argument to the add function.
 void main(void)
 {
  complex c1,c2;
  c1.setdata (3,4);
  c2.setdata ( 5,7);
  c1.add (c2);                  // c2 will be explicitly passed ,result will be in c1.
  c1.showdata();
 }
```

Returning entire object from the function

```
class complex
{    int real,;
     float img;
public : void setdata ( int , float);
           complex add (complex);
           void showdata(void);
};
void complex :: setdata ( int r, float I)
   {
           real = r;
           img =I;
   }
complex  complex :: add ( complex c2)
{
 complex temp; // create a temporary object
 temp. real = real + c2.real ;
temp. img = img + c2.img;          // result is in temp object
 return temp;                  // return temp object to the main prog.
}
void complex :: showdata (void)
{
 cout << real << "+" << "i" << img;
}
void main(void)
{
  complex c1,c2,c3;
  c1.setdata(5,-6)
  c2.setdata ( 3,4);
  c3 = c1.add (c2);   // c2 is explicitly passed, result will be in c3.
  c3.showdata();
}
```

PROGRAMS BASED ON PASSING OBJECT AS FUNCTION ARGUMENT.

- **Write a program to add two complex numbers.**

- ❑ **Adding complex numbers by passing two objects as arguments.**
- ❑ **Adding complex numbers by passing only one object as argument.**
- ❑ **Returning entire object from the function**

- **Create a class time having data members as hour and minutes and W A P to add to Time objects.**
- **Declare a class distance having data members as feet and inches. Write a program to add two distances.**

2.4 STATIC DATA MEMBERS:

Characteristics of static data members

- It is initialized to zero when the first object of the class is created. No other initialization is permitted.
- Only one copy of that member is created for entire class and is shared by all the objects of that class. It doesn't matter with the number of objects created.
- It is visible only within the class, but its lifetime is the entire program.
- The static data members are associated with the class rather than with any object ,that is why they are sometimes called **class variables**.
- To access static data always use name of the class instead of object with scope resolution operator (**::**).
- Eg: item **::** count;
- Where item name of the class and the count is is the static data member.
- The type and the scope of each static variable must be declared outside the class definition
- **int item :: count**
- This type of variable is normally used to maintain values, which is common to entire class.

STATIC MEMBER FUNCTIONS

- Like member variables, member functions can also be declared as static.
- Characteristics of static member functions:
- A static function can have access to other static members declared in the same class.
- A static member function can be called using the class name and **::**
- Example : If display() is static function of class student then display() can be called as
- **Student :: display();**

FRIEND FUNCTION

- Private members of a class cannot be accessed from outside the class. However there could be a situation where more than one classes want to share a particular function.
- For example consider two classes ’manager’ and ‘scientist’. We can use a function incom_tax ( ) that will operate on the objects of both these classes and calculate the total income tax. Here the function income_tax () need to access private members of both the classes.
- This can be done with the help of friend function.
- “Friend function is a normal c++ function that can access private members of the class to whom it is declared as friend.”
- To make a normal c++ function “friendly” to the class , precede the declaration of the function with keyword ‘friend’ inside the class.

Example

```
Class fun
{
private:
    int a, b;
public:
    friend void xyz ( fun s);  // function xyz is a friend of class fun
};
```

- The function is defined somewhere in the program (outside the class) like a normal c++ function.
- The function can be declared as to any number of classes.
- The friend function is not a member of a class, but it has full access rights to the private members of the class.

CHARACTERISTICS OF FRIEND FUNCTION:

- It is not in the scope of the class to which it is declared as friend.
- Since it is not in the scope of the class it cannot be called using an object of the class.
- It can be invoked like a normal c++ function without the help of any object.
- Unlike member function, it cannot access the data members directly and has to use an object name and dot (.) operator.
- It can be declared either in the public or private sections of a class without affection its meaning.
- Usually it has object as argument.

PROGRAM BASED ON FRIEND FUNCTION

Vidyalankar Polytechnic

/*Program to find the mean value of a given number using friend function

```
#include<iostream.h> #include<conio.h>
class  sample
{
   int val1,val2;
  public:
   void get()
   {
     cout<<"Enter two values:";
     cin>>val1>>val2;
   }
   friend float mean(Sample ob);
};
```

```
float mean(sample ob)
{
  return float(ob.val1+ob.val2)/2;
}
void main()
{
   clrscr();
   sample  obj;
   obj.get();
   cout<<"\n Mean value is :
"<<mean(obj);
   getch();
}
```

Output:

Enter two values: 10, 20

Mean Value is: 15

Write a program where test1 and test2 are two classes. Write a friend function which will calculate average of test1 and test2. (pass by value)


```
# include <iostream.h>
class test2;   // prior declaration of class2
class test1
{
 private: float m1,m2;
public:         void setdata (float a , float b)
       {
                m1=a ;
                m2 =b;
        }
        friend void average ( test1 ,test2);  // function
average is a friend of test1
};
class test2
{
 private: float m3,m4;
public:         void setdata (float a , float b)
        {
                m3=a ;
                m4 =b;
        }
       friend void average ( test1 ,test2);      // function
average is a friend of test1
};
```

```
void average ( test1 t1 , test2 t2)
{
  float avg;
  avg =( t1.m1 + t1.m2 + t2.m3 + t2.m4)/4;
  cout << avg;
}
void main ()
{
 test t1;
 test t2;
 t1.setdata (34, 70);
 t2.setdata (40, 60);
 average ( t1 ,t2 );  // friend fun. is invoked like a
normal  c++ function without any object
}
```