Chapter 5

Class & Object

Structure

We know that one of the unique features of the C language is structure. They provide a method of packing together data of different types. A structure is a convenient tool for handling a group of logically related data items. It is user-defined data types with a template that serves to define its data properties. Once the structure type has been defined, we can create variables of that type using declarations that are similar to the built-in type declarations.

The general form of a structure declaration is

```
struct <structure name / tag name >
{
        structure member 1;
        structure member 2;
        . . . .
        . . . .
};
```

Limitation of C Structure

The standard C does not allow the struct data type to be treated like built-in types. For example consider the following structure.

```
struct complex
{
    float x;
    float y;
};
struct complex c1,c2,c3;
```

The complex numbers c1,c2 and c3 can easily be assigned values using dot operator. But cannot add two complex numbers or subtract one from the other.

Extension to Structure

C++ supports all the features of structure as defined in C. It attempts to bring the user-defined types as close as possible to the built-in types, and also provides a facility to hide the data which is one of the main precepts of OOP. Inheritance, a mechanism by which one type can inherit characteristic from other types, is also supported by C++.

In C++, a structure can have both variables and functions as members. It can also declare some of its members as 'private' so that they cannot be accessed directly by the external functions. In C++, the structure names are stand-alone and can be used like any other type names. That is, the keyword struct can be omitted in the declaration of structure variables.

There is very little syntactical difference between structure and classes in C++ and, therefore, they can be used interchangeably with minor modification. Since class is a specially introduced data type in C++, most of the C++ programmers tend to use the structure for holding only data, and classes to hold both the data and functions.

Note

The only difference between a structure and a class in C++ is that, by default, the members of a class are private while, by default, the members of a structure are public.

Unions

In C++, we can declare unions tag names. For example,

```
union
{
    int height;
    float weight;
}
```

is legal declaration in C++. The members can be directly accessed by the name.

```
height = 174;
weight = 53;
```

Class

A class is a way to bind the data & it associates function together. It allows the data to be hidden from external use. Generally a class specification has two parts.

1. Class Declaration
2. Class Function Definition

The class declaration describe members of the class. The definition of the function describes how the class function are implemented. The general form of class declaration is as follows.

```
Class < Class-Name >
{
  Private :
            Variable Declaration;
            Function Declaration;
  Public :
            Variable Declaration;
            Function Declaration;
};
```

The class declaration is similar to a structure declaration. The keyword specified that whatever declare a class. The body of class is enclosed within open & close brace bracket i.e. { and } and always terminated by semicolon.

The class body contain declaration of variable & function. This function & variables are collectively called member of the class. The members are generally grouped under section namely Public & Private. The keyword Private & Public are known as visibility labels or mode. This labels are always followed by colon.

The members that are declared as private can only be access from the function declared in a class (i.e. Public section). On the other hand public members can be access from outside the class. The use of keyword private is optional. By default all the members of class are private. The variables declared inside the class are known as data members & the functions are known as member function. Only the member function can access the private data members & private function.

Defining Member Function

The member function can be defined in two different places.

1) Inside the class
2) Outside the Class

1) Function Definition Inside the Class

We can define a member function inside a class rather than to declared it outside the class. A member function can be define inside the class similar to normal function. When member function are define inside the class they are considered as inline function. Therefore it is convention that when member functions are small in size it is always better to define the member function inside the class.

Write a Program to read Student Information using Class

```
#include<iostream.h>
#include<conio.h>

class stud
{
   private:
      int rno;
      float per;
   public:
      void getdata()
      {
        cout<<"Enter the roll no & percentage = ";
        cin>>rno>>per;
      }
      void putdata()
      {
        cout<<"Roll no = "<<rno<<endl
            <<"Percentage = "<<per<<endl;
      }
};
```

```
main()
{
clrscr();
stud s,p;
s.getdata();
s.putdata();
p.getdata();
p.putdata();
getch();
return(0);
}
```

2) Function Definition Outside the Class

The member function can also be define outside the class but there is a difference between normal function definition & member function definition. The member function has identity label in the header part of function definition. The label tells the complier that to which class the function belongs. The general form of defining a member function outside the class is

```
Return Type  Class-name : : Function-name(Argument list)
{
        Function body;
}
```

The class name is a membership label which tells the complier that which to class the function belongs i.e. the scope of the function is restricted to the class specified by the class member. The symbol : : is known as scope resolution operator.

Write a Program to read Student Information using Class

```
#include<iostream.h>
#include<conio.h>

class stud
{
   private:
      int rno;
      float per;
   public:
      void getdata();
      void putdata();
};

void stud :: getdata()
{
  cout<<"Enter the roll no & percentage = ";
  cin>>rno>>per;
}
```

```
void stud :: putdata()
{
  cout<<"Roll no = "<<rno<<endl
      <<"Percentage = "<<per<<endl;
}

main()
{
clrscr();
stud s,p;
s.getdata();
s.putdata();
p.getdata();
p.putdata();
getch();
return(0);
}
```

Write a Program to read Item no & cost using Class

```
#include<iostream.h>
#include<conio.h>

class item
{
   private:
      int no;
      float cost;
   public:
      void getdata()
      {
        cout<<"Enter the item no & cost = ";
        cin>>no>>cost;
      }
      void putdata()
      {
        cout<<"Item no = "<<no<<endl
            <<"Cost = "<<cost<<endl;
      }
};

main()
{
clrscr();
item s,p;
s.getdata();
s.putdata();
p.getdata();
p.putdata();
```

```
getch();
return(0);
}
```

Write a Program to read Item no & cost using Class

```
#include<iostream.h>
#include<conio.h>

class item
{
   private:
      int no;
      float cost;
   public:
      void getdata();
      void putdata();
};

void item :: getdata()
{
  cout<<"Enter the item no & cost = ";
  cin>>no>>cost;
}

void item :: putdata()
{
  cout<<"Item no = "<<no<<endl
      <<"Cost = "<<cost<<endl;
}

main()
{
clrscr();
item s,p;
s.getdata();
s.putdata();
p.getdata();
p.putdata();
getch();
return(0);
}
```

Creating Object

Once a class is declared we can create variables of that type by using the class name.

e.g.

stud p,q;

This statement will create a variables p & q of type stud. In C++ class variable are known as object. We can create any no of object in a single statement.

There is another way to declared object as

```
Class stud
{
  Private :

                ……
  Public :
                ……
}s1,s2,s3;
```

Accessing Member Function

The private data of the class can be access by the member function of that class. By calling a member function the general form is as follows

```
Object-name . Function-name(Argument-list);
e.g.
        stud p,q;
        p.getdata();
        q.putdata();
```

Write a Program to Calculate Factorial of given no using Class.

```
#include<iostream.h>
#include<conio.h>

class factorial
{
   private:
      int no,fa;
   public:
      void getdata()
      {
        cout<<"Enter the no = ";
        cin>>no;
      }
      void fact();
};

void factorial::fact()
{
fa=1;
for(int a=1;a<=no;a++)
{
   fa=fa*a;
}
cout<<"Factorial of given no = "<<fa;
```

```
}

main()
{
factorial f;
clrscr();
f.getdata();
f.fact();
getch();
return(0);
}
```

Write a Program to Find Reverse no of given no using Class.

```
#include<iostream.h>
#include<conio.h>

class reverse
{
   private:
      int n,sum;
   public:
      void getdata()
      {
        cout<<"Enter the no = ";
        cin>>n;
      }
      void rev();
};

void reverse::rev()
{
sum=0;
do
{
   sum=sum*10+n%10;
   n=n/10;
}while(n>0);
cout<<"Reverse no = "<<sum;
}

main()
{
reverse r;
clrscr();
r.getdata();
r.rev();
getch();
return(0);
}
```

Write a Program to Find sum of digit of given no using Class.

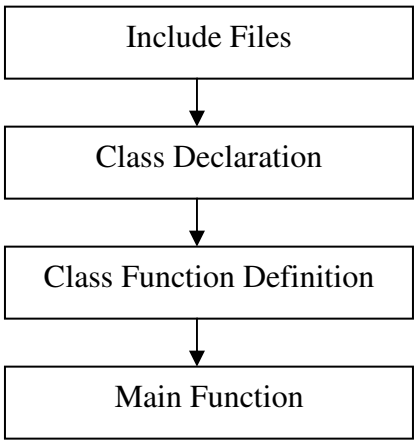
```
#include<iostream.h>
#include<conio.h>

class sumdigit
{
   private:
      int n,s;
   public:
      void getdata()
      {
        cout<<"Enter the no = ";
        cin>>n;
      }
      void sum();
};


void sumdigit::sum()
{
s=0;
do
{
   s=s+n%10;
   n=n/10;
}while(n>0);
cout<<"Sum of Digit = "<<s;
}

main()
{
sumdigit r;
clrscr();
r.getdata();
r.sum();
getch();
return(0);
}
```

Write a Program to Check given no is Palindrome or not using Class.

```
#include<iostream.h>
#include<conio.h>

class palindrom
{
   private:
      int n,sum;
   public:
```

```
      void getdata()
      {
        cout<<"Enter the no = ";
        cin>>n;
      }
      void pali();
};

void palindrom::pali()
{
   int m=n;
   sum=0;
   do
   {
      sum=sum*10+n%10;
      n=n/10;
   }while(n>0);
   cout<<"Reverse no = "<<sum<<endl;
   if(m==sum)
     cout<<"No is Palindrom";
   else
     cout<<"No is not Palindrom";
}

main()
{
palindrom r;
clrscr();
r.getdata();
r.pali();
getch();
return(0);
}
```

Structure of C++ Program



Nesting of Member Function

A member function of a class can be called only by an object of that class using a dot operator. A member function can be called by using its name inside another member function of the same class. This is known as nesting of member function.

Write a Program to Find large no in given two object values using nested member function.

```
#include<iostream.h>
#include<conio.h>

class large
{
   private:
      int a,b;
   public:
      void getdata()
      {
         cout<<"Enter two no = ";
         cin>>a>>b;
      }
      void putdata();
      int largest();
};

void large::putdata()
{
cout<<"Largest no = ";
int c;
c=largest();
cout<<c;
}

int large::largest()
{
if(a>b)
   return(a);
else
   return(b);
}

main()
{
clrscr();
large p;
p.getdata();
p.putdata();
getch();
return(0);
}
```

Write a Program to Read & Write Student Roll no & 3 sub marks & Calculate total marks & per. using nested member function.

```
#include<iostream.h>
#include<conio.h>

class stud
{
  private:
      int rno,m1,m2,m3,total;
      float per;
  public:
      void getdata()
      {
       cout<<"Enter roll no. and three sub. marks = ";
       cin>>rno>>m1>>m2>>m3;
      }
      void putdata();
      void sum();
};

void stud::putdata()
{
cout<<"Roll no = "<<rno<<endl
    <<"Marks 1 = "<<m1<<endl
    <<"Marks 2 = "<<m2<<endl
    <<"Marks 3 = "<<m3<<endl;
sum();
cout<<"Total = "<<total<<endl
    <<"Percentage = "<<per;
}
void stud::sum()
{
  total=m1+m2+m3;
  per=total/3;
}
main()
{
clrscr();
stud s;
s.getdata();
s.putdata();
getch();
return(0);
}
```

Making an Outside Function Inline

One of the objectives of OOP is to separate the details of implementation from the class definition. It is therefore good practice to define the member functions outside the class.

We can define a member function outside the class definition and still make it inline by just using the keyword Inline in the header line of function definition.

Write a Program to Read & Write Student Information using Inline Functions in Class.

```
#include<iostream.h>
#include<conio.h>

class stud
{
   private:
      int rno;
      float per;
   public:
      void getdata()
      {
         cout<<"Enter the roll no. and per = ";
         cin>>rno>>per;
      }
      void putdata();
};

inline void stud::putdata()
{
cout<<"Roll no = "<<rno<<endl
    <<"Percentage = "<<per;
}

main()
{
clrscr();
stud s;
s.getdata();
s.putdata();
getch();
return(0);
}
```

Write a Program to Calculate cube of given no using Inline Functions in Class.

```
#include<iostream.h>
#include<conio.h>

class cube
{
   private:
      int no;
   public:
      void getdata()
      {
```

```
         cout<<"Enter the no = ";
         cin>>no;
      }
      void calculate();
};

inline void cube::calculate()
{
   int a;
   a=no*no*no;
   cout<<"Cube = "<<a;
}

main()
{
clrscr();
cube c;
c.getdata();
c.calculate();
getch();
return(0);
}
```

Private Member Function

It is general convention to declare the data members as private and member function as public. But there are some situation in which some function should be hidden from outside calls. For this purpose the member function can be place in the private part of the class.

A private member function can only be called by another member function of the same class. The object of the class is also not able to call private member function of the class using dot operator.

e.g.

```
class abc
{
   private :
                int x;
                int y;
                void getxy();
   public :
                void putxy();
}a;
```

If we want to call the getxy() using the object 'a' then the statement a.getxy() is invalid. This is because the object of a class can not be access the private data of a class. In this case getxy() must be called function the putxy().

Write a Program to Read & Write Student Information using Private member Functions.

```
#include<iostream.h>
```

```
#include<conio.h>

class stud
{
  private:
      int rno;
      char name[30];
      float per;
      void getdata()
      {
       cout<<"Enter the roll no,name and per = ";
       cin>>rno>>name>>per;
      }
  public:
      void putdata()
      {
      getdata();
      cout<<"Roll no = "<<rno<<endl
          <<"Name = "<<name<<endl
          <<"Percentage = "<<per;
      }
};
main()
{
clrscr();
stud s;
s.putdata();
getch();
return(0);
}
```

Write a Program to Read 2 no using Private member Functions.

```
#include<iostream.h>
#include<conio.h>

class number
{
   private:
      int x,y;
      void get_xy()
      {
       cout<<"Enter the value of x and y = ";
       cin>>x>>y;
      }
   public:
      void put_xy()
      {
       get_xy();
       cout<<"Value of x = "<<x<<endl
           <<"Value of y = "<<y;
```

```
      }
};

main()
{
clrscr();
number n;
n.put_xy();
getch();
return(0);
}
```

Write a Program to Read & Write Book Information

```
#include<iostream.h>
#include<conio.h>

class book
{
    private:
      int bno;
      float bprice;
    public:
      void getdata(int a,float b)
      {
          bno=a;
          bprice=b;
      }
      void putdata()
      {
         cout<<"Book no = "<<bno<<endl
             <<"Book price = "<<bprice;
      }
};

main()
{
book p;
clrscr();
int x;
float y;
cout<<"Enter the book no. and price = ";
cin>>x>>y;
p.getdata(x,y);
p.putdata();
getch();
return(0);
}
```

Create a Class "Bank" with data members acno & bal and the member function balance(), deposit(), withdrawal(). Among this member function the function

balance() is private. Write a Program to Deposit & Withdrawal the Amount from the balanced amount while checking the balanced amount must be greater or equal withdrawal amount.

```
#include<iostream.h>
#include<conio.h>

class bank
{
   private:
      int acno;
      float bal;
      void balance();
   public:
      void deposite();
      void withd();
};

void bank::balance()
{
   cout<<"\nBalance = "<<bal;
}
void bank::deposite()
{
   int de;
   cout<<"\nEnter acc. no. and deposite amount = ";
   cin>>acno>>de;
   bal=bal+de;
   balance();
}
void bank::withd()
{
   int wt;
   cout<<"\nEnter acc no. and withd amt = ";
   cin>>acno>>wt;
   if(bal<wt)
     cout<<"\nCheck out your pass book";
   else
   {
     bal=bal-wt;
     balance();
   }
}

main()
{
clrscr();
bank b;
b.deposite();
b.withd();
b.deposite();
```

```
b.withd();
getch();
return(0);
}
```

Write a Program to Read & Write Book Information

```
#include<iostream.h>
#include<conio.h>

class book
{
   private:
      int bno;
      float bprice;
   public:
      void putdata()
      {
         cout<<"Book no = "<<bno<<endl
             <<"Price = "<<bprice;
      }
      void getdata(int,float);
};

void book::getdata(int a,float b)
{
    bno=a;
    bprice=b;
}

main()
{
book p;
clrscr();
int x; float y;
cout<<"Enter the book no. and price = ";
cin>>x>>y;
p.getdata(x,y);
p.putdata();
getch();
return(0);
}
```

Write a Program to Calculate Factorial of given no. using Class.

```
#include<iostream.h>
#include<conio.h>

class fact
{
   private:
      int n,f;
```

```
   public:
      void getno(int);
      int facto();
};

void fact::getno(int a)
{
   n=a;
}

int fact::facto()
{
f=1;
while(n>0)
{
   f=f*n;
   n=n-1;
}
return(f);
}


main()
{
fact x;
clrscr();
int p,q;
cout<<"Enter the no = ";
cin>>p;
x.getno(p);
q=x.facto();
cout<<"Factorial = "<<q;
getch();
return(0);
}
```

Write a Program to Calculate Reverse of given no. using Class.

```
#include<iostream.h>
#include<conio.h>

class reverse
{
   private:
      int n,r;
   public:
      void getno(int);
      int rev();
};

void reverse::getno(int a)
```

```
{
   n=a;
}
int reverse::rev()
{
r=0;
do
{
   r=r*10+n%10;
   n=n/10;
}while(n>0);
return(r);
}

main()
{
reverse x;
clrscr();
int p,q;
cout<<"Enter the no = ";
cin>>p;
x.getno(p);
q=x.rev();
cout<<"Reverse no = "<<q;
getch();
return(0);
}
```

Array of Object

We know that an array can be of any data type including structure. Similarly we can also have arrays of variables that are of the type class. Such variables are called arrays of object.

Write a Program to Read & Write 10 Employee Information.

```
#include<iostream.h>
#include<conio.h>

class emp
{
   private:
      int eno;
      float sal;
   public:
      void getdata()
      {
         cout<<"Enter the emp no. and salary = ";
         cin>>eno>>sal;
      }
      void putdata()
```

```
      {
         cout<<"Emp. no = "<<eno<<endl
             <<"Salary = "<<sal<<endl;
      }
};

main()
{
emp e[10];
clrscr();
cout<<"Enter 10 Employee Information\n";
for(int a=0;a<10;a++)
{
   e[a].getdata();
}
for(int b=0;b<10;b++)
{
   e[b].putdata();
}
getch();
return(0);
}
```

Write a Program to Read & Write 10 Student Information.

```
#include<iostream.h>
#include<conio.h>

class stud
{
   private:
      int rno;
      float per;
   public:
      void getdata()
      {
         cout<<"Enter the roll no & per = ";
         cin>>rno>>per;
      }
      void putdata()
      {
         cout<<"Roll no = "<<rno<<endl
             <<"Percentage = "<<per<<endl;
      }
};

main()
{
stud s[10];
clrscr();
cout<<"Enter 10 student Information"<<endl;
```

```
for(int a=0;a<10;a++)
{
   s[a].getdata();
}
cout<<"Student Information\n";
for(a=0;a<10;a++)
{
   s[a].putdata();
}
getch();
return(0);
}
```

Object as Argument

Like any other data type an object may be used as function argument. This can be done in two different ways.

1) A copy of the entire objects is pass to the function.
2) Only the address of a object is transfer to the function.

The first method is called pass by value since copy of object is pass to the function any changes made to the object inside the function do not affect the value of actual object.

The second method is called pass by reference when address of the object is pass. The called function directly works on the actual objects. This means that any changes made to object inside the function will reflect the value of actual object. The pass by reference method is more efficient than pass by value.

Write a Program to Addition of data Members of 2 Object.

```
#include<iostream.h>
#include<conio.h>

class add
{
   private:
      int n;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>n;
      }
      void putdata()
      {
         cout<<"Number = "<<n<<endl;
      }
      void sum(add,add);
};
```

```
void add::sum(add x,add y)
{
  int p;
  p=x.n+y.n;
  cout<<"Addition = "<<p;
}

main()
{
add a,b,c;
clrscr();
a.getdata();
b.getdata();
a.putdata();
b.putdata();
c.sum(a,b);
getch();
return(0);
}
```

Write a Program to Subtraction of data Members of 2 Object.

```
#include<iostream.h>
#include<conio.h>

class number
{
   private:
      int a;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>a;
      }
      void sub(number,number);
};

void number::sub(number x,number y)
{
  int c;
  c=x.a-y.a;
  cout<<"Substraction = "<<c;
}

main()
{
number p,q,r;
clrscr();
p.getdata();
q.getdata();
```

```
r.sub(p,q);
getch();
return(0);
}
```

Write a Program to Find Average of Object Values.

```
#include<iostream.h>
#include<conio.h>

class num
{
   private:
      int a;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>a;
      }
      void putdata()
      {
          cout<<"Number = "<<a<<endl;
      }
      void avg(num,num);
};

void num::avg(num x,num y)
{
  float c;
  c=(x.a+y.a)/2;
  cout<<"Average = "<<c;
}

main()
{
num p,q,r;
clrscr();
p.getdata();
q.getdata();
p.putdata();
q.putdata();
r.avg(p,q);
getch();
return(0);
}
```

Write a Program to Calculate Addition of Complex no.

```
#include<iostream.h>
#include<conio.h>
```

```
class complex
{
   private:
      int real,imag;
   public:
      void getdata()
      {
         cout<<"Enter real and imag no = ";
         cin>>real>>imag;
      }
      void putdata()
      {
         cout<<real<<"+"<<imag<<"i"<<endl;
      }
      void sum(complex,complex);
};

void complex::sum(complex x,complex y)
{
    int r,im;
    r=x.real+y.real;
    im=x.imag+y.imag;
    cout<<r<<"+"<<im<<"i";
}

main()
{
complex p,q,s;
clrscr();
p.getdata();
q.getdata();
p.putdata();
q.putdata();
cout<<"Addition = ";
s.sum(p,q);
getch();
return(0);
}
```

Write a Program to Interchange Object Value.

```
#include<iostream.h>
#include<conio.h>

class inter
{
   private:
      int no;
   public:
      void getdata()
```

```
      {
       cout<<"Enter the no = ";
       cin>>no;
       }
       void putdata()
       {
       cout<<"Number = "<<no<<endl;
       }
       void swap(inter &,inter &);
};

void inter::swap(inter &fa1,inter &fa2)
{
int p;
p=fa1.no;
fa1.no=fa2.no;
fa2.no=p;
}


main()
{
inter x,y,z;
clrscr();
x.getdata();
y.getdata();
cout<<"Interchange value"<<endl;
z.swap(x,y);
x.putdata();
y.putdata();
getch();
return(0);
}
```

Returning Object

A function declared in a class not only receives object as argument but also can return an object i.e. function may return a value of class type. When a function return an object in the declaration of that function we must specify the return type as the class name.

Write a Program to Calculate Addition of Complex no.

```
#include<iostream.h>
#include<conio.h>

class complex
{
   private:
      int real,imag;
   public:
      void getdata()
      {
```

```
          cout<<"Enter real no = ";
          cin>>real;
          cout<<"Enter imag no = ";
          cin>>imag;
      }
      void putdata()
      {
          cout<<"Complex no = "
            <<real<<"+"<<imag<<"i"<<endl;
      }
      complex sum(complex,complex);
};

complex complex::sum(complex a,complex b)
{
   complex c;
   c.real=a.real+b.real;
   c.imag=a.imag+b.imag;
   return(c);
}
main()
{
clrscr();
complex p,q,r,s;
p.getdata();
q.getdata();
p.putdata();
q.putdata();
cout<<"Addition ";
s=r.sum(p,q);
s.putdata();
getch();
return(0);
}
```

Write a Program to Find Multiplication of 2 Matrix.

```
#include<iostream.h>
#include<conio.h>

class matrix
{
   private:
      int m[3][3];
   public:
      void getdata();
      void putdata();
      matrix mul(matrix,matrix);
};

void matrix::getdata()
```

```
{
cout<<"Enter matrix element =\n";
for(int a=0;a<3;a++)
{
   for(int b=0;b<3;b++)
      cin>>m[a][b];
}
}

void matrix::putdata()
{
cout<<"Matrix is = "<<endl;
for(int a=0;a<3;a++)
{
   for(int b=0;b<3;b++)
      cout<<m[a][b]<<"\t";
   cout<<endl;
}
}

matrix matrix::mul(matrix x,matrix y)
{
matrix z;
for(int a=0;a<3;a++)
{
   for(int b=0;b<3;b++)
   {
      z.m[a][b]=0;
      for(int c=0;c<3;c++)
          z.m[a][b]=z.m[a][b]+x.m[a][c]*y.m[c][b];
   }
}
return(z);
}

main()
{
clrscr();
matrix p,q,r,s;
p.getdata();
q.getdata();
p.putdata();
q.putdata();
s=r.mul(p,q);
s.putdata();
getch();
return(0);
}
```

Write a Program to Find Average of Object Values.

```
#include<iostream.h>
#include<conio.h>

class num
{
   private:
      int a;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>a;
      }
      void putdata()
      {
          cout<<"Number = "<<a<<endl;
      }
      num avg(num,num);
};

num num::avg(num x,num y)
{
  num z;
  z.a=(x.a+y.a)/2;
}

main()
{
num p,q,r,s;
clrscr();
p.getdata();
q.getdata();
p.putdata();
q.putdata();
s=r.avg(p,q);
s.putdata();
getch();
return(0);
}
```

Friend Function

The private member of a class can not be access by the function which is not a member function of the class i.e. a non-member function can not have an access to the private data of a class. But there are some situation where we would like that two classes to share the same function.

e.g.

Suppose there are two classes Manager & Worker. The Manager class store the information about Manager where as the Worker class store the information about Worker in an

organization. A function tax() is used to calculate the income tax for Worker & Manager i.e. tax() function must be work on the object of Manager & Worker.

In such situation C++ allows the common function tax() to be made friendly with both the classes by allows the function to access the private data of the different classes.

To make an outside function friendly to a class we have to simply declared this function as a friend of the class as shown below.

```
class worker
{
        ---------
        ---------
    public :
        ---------
        ---------
        friend void tax();
}
```

The function declaration should be preceded by the keyword “friend”. The function is defined anywhere in the program like a normal C++ function. The function definition does not use either the keyword “friend” or the scope resolution operator. The function that are declared with keyword “friend” function. A function can be declared as a friend in any number of classes. A friend function has fully accessed right to the private data of the class even though it is not a member function of the class.

A friend function has certain special characteristics.

1) It is not in the scope of the class to which it has been declared as friend.
2) Since it is not in the scope of the class it can not be called using the object of that class.
3) It can be invoke like normal function.
4) In can be declared either in the public or the private part of the class.
5) Usually friend function has object as argument.

Write a Program to Find Mean value using Friend Function.

```
#include<iostream.h>
#include<conio.h>

class number
{
   private:
      int no;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>no;
      }
      friend int mean(number,number,number);
};
```

```
int mean(number a,number b,number c)
{
float p;
p=(a.no+b.no+c.no)/3;
return(p);
}

main()
{
int m;
clrscr();
number x,y,z;
x.getdata();
y.getdata();
z.getdata();
m=mean(x,y,z);
cout<<"Mean Value = "<<m;
getch();
return(0);
}
```

Write a Program to Using Friend Function perform Addition of 2 Matrix.

```
#include<iostream.h>
#include<conio.h>

class matrix
{
   private:
      int m[3][3];
   public:
      void getdata();
      void putdata();
      friend matrix add(matrix &,matrix &);
};

void matrix::getdata()
{
cout<<"Enter matrix element";
for(int a=0;a<3;a++)
{
   for(int b=0;b<3;b++)
      cin>>m[a][b];
}
}

void matrix::putdata()
{
cout<<"Matrix is"<<endl;
for(int a=0;a<3;a++)
{
```

```
   for(int b=0;b<3;b++)
       cout<<m[a][b]<<"\t";
   cout<<endl;
}
}

matrix add(matrix &x,matrix &y)
{
matrix z;
for(int a=0;a<3;a++)
{
   for(int b=0;b<3;b++)
      z.m[a][b]=x.m[a][b]+y.m[a][b];
}
return(z);
}

main()
{
matrix p,q,r;
clrscr();
p.getdata();
q.getdata();
r=add(p,q);
p.putdata();
q.putdata();
cout<<"Addition of ";
r.putdata();
getch();
return(0);
}
```

Write a Program to Create 2 Class & Interchange the value of Data Members.

```
#include<iostream.h>
#include<conio.h>

class no1;
class no2
{
   private:
      int n;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>n;
      }
      void putdata()
      {
         cout<<"Number = "<<n;
      }
```

```
      friend void inter(no1,no2);
};

class no1
{
      int n;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>n;
      }
      void putdata()
      {
         cout<<"Number = "<<n<<endl;
      }
      friend void inter(no1,no2);
};



void inter(no1 a,no2 b)
{
   int c;
   c=a.n;
   a.n=b.n;
   b.n=c;
   cout<<"\nInterchange value = "<<a.n<<" "<<b.n;
}

main()
{
clrscr();
no1 x;
no2 y;
x.getdata();
y.getdata();
x.putdata();
y.putdata();
inter(x,y);
getch();
return(0);
}
```

Write a Program to Create 2 Class & Interchange the value of Data Members.

```
#include<iostream.h>
#include<conio.h>

class no1;
class no2
```

```
{
   private:
      int n;
   public:
      void getdata()
      {
        cout<<"Enter the no = ";
        cin>>n;
      }
      void putdata()
      {
        cout<<"Number = "<<n<<endl;
      }
      friend void inter(no1 &,no2 &);
};

class no1
{
   private:
      int n;
   public:
      void getdata()
      {
         cout<<"Enter the no = ";
         cin>>n;
      }
      void putdata()
      {
         cout<<"Number = "<<n<<endl;
      }
      friend void inter(no1 &,no2 &);
};

void inter(no1 &a,no2 &b)
{
int c;
c=a.n;
a.n=b.n;
b.n=c;
}

main()
{
no1 x;
no2 y;
clrscr();
x.getdata();
y.getdata();
x.putdata();
y.putdata();
inter(x,y);
cout<<"Interchange Value\n";
```

```
x.putdata();
y.putdata();
getch();
return(0);
}
```

Constant Member Function

If a member function does not change any data member in class then we may declare it as a constant member function.

e.g.

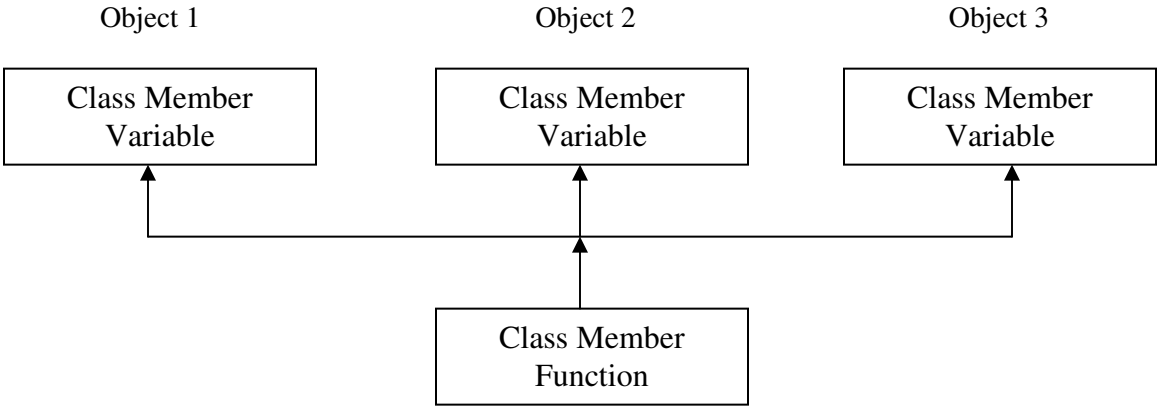
void putdata(); const

The keyword const is append to the function prototype (In both declaration & definition). The complier will generate an error massage if such function try to change the data values.

Memory Allocation for Object

When the objects are declared or created then memory space is reserved, by only declaring a class no memory space is reserved. The member function are created & place in the memory. The member functions are created when they are defined as a part of class specification.

Since all the object belonging to that class use the same member functions, no separate space is allocated for member function. Only the space for member variable get allocated separately for each object. This can be shown using a following diagram.



Static Data Member

A data member of a class can be qualified as static. A static member variable has certain special characteristics.

1) It is initialized to zero when the first object of its class is created. No other initialization is permitted.
2) Only one copy of that member is created for the entire and is shared by all the object of that class.

Static variable is normally used to maintain values common to the entire class. Note that the type & scope of each static member variable must be defined outside class definition. This is necessary because the static data members are stored separately rather than as a part of an object.

Write a Program to Demonstration of Static date Member.

```
#include<iostream.h>
#include<conio.h>

class emp
{
   private:
      int sal,total;
      static int incr;
   public:
      void getdata()
      {
         cout<<"Enter the salary = ";
         cin>>sal;
         cout<<endl;
      }
      void putdata()
      {
         cout<<"Salary = "<<sal<<endl
             <<"Increment = "<<incr<<endl
             <<"Total Salary = "<<total<<endl;
      }
      void cal()
      {
           total=sal+incr;
      }
};

int emp::incr=100;

main()
{
emp e1,e2,e3;
clrscr();
e1.getdata();
e1.cal();
e1.putdata();
e2.getdata();
e2.cal();
e2.putdata();
e3.getdata();
e3.cal();
e3.putdata();
getch();
return(0);
```

```
}
```

Static Member Function

Like static variable we can also have static member function. The static member function has the following properties.

1) A static function can have access to only other static members declared in the same class.
2) A static member function can be called using the class name as follows.

Class name : : Function name ();

Write a Program to Demonstration of Static Member Function.

```
#include<iostream.h>
#include<conio.h>
class test
{
   private:
      int code;
      static int count;
   public:
      void setcode()
      {
         code=++count;
      }
      void showcode()
      {
         cout<<"Object no = "<<code<<endl;
      }
      static void showcount()
      {
         cout<<"Count = "<<count<<endl;
      }
};

int test::count;

main()
{
test t1,t2;
clrscr();
t1.setcode();
test::showcount();
t2.setcode();
test::showcount();
test t3;
t3.setcode();
t3.showcount();
t1.showcode();
t2.showcode();
```

```
t3.showcode();
getch();
return(0);
}
```

Write a Program to Calculate Electric Bill
For First 100 Units Rate is 0.5 Rs.
For Next 200 Units Rate is 0.75 Rs.
Beyond 300 Units Rate is 1 Rs.

```
#include<iostream.h>
#include<conio.h>

class bill
{
      float unit,amt;
  public:
      void getdata()
      {
       cout<<"Enter the Unit = ";
       cin>>unit;
      }
      void putdata()
      {
       cout<<"Amount of Bill = "<<amt;
      }
      void cal();
};

void bill :: cal()
{
   if(unit<=100)
     amt=unit*0.5;
   else
     if(unit>100 && unit<=200)
     {
       unit=unit-100;
       amt=100*0.5+unit*0.75;
     }
     else
     {
       unit=unit-300;
       amt=100*0.5+200*0.75+unit*1;
     }
}

main()
{
bill b;
clrscr();
b.getdata();
```

```
b.cal();
b.putdata();
getch();
return(0);
}
```

Pointer to Member

It is possible to take the address of a member of a class and assign it to a pointer. The address of a member can be obtained by applying the operator & to a "fully qualified" class member name. A class member pointer can be declared using the operator : :* with the class name.

e.g.
```
class A
{
    private:
        int m;
    public:
        void show();
};
```

We can define a pointer to the member m as follows:

```
int A: :*ip = &A : : m;
```

The ip pointer created thus acts like a class member in that it must be invoked with a class object. In the statement above, the phrase A: :* means "pointer-to-member of A class". The phrase &A : : m means the "address of the m member of A class".

Remember, the following statement is not valid:

```
int *ip = &m;
```

This is because m is not simply an int type data. It has meaning only when it is associated with the class to which it belongs. The scope operator must be applied to both the pointer and the member.

The pointer ip can now be used to access the member m inside member functions (or friend functions). Let us assume that a is an object of A declared in a member function. We can access m using the pointer ip as follows:

```
cout<<a.*ip;
cout<<a.m;
```

Now, look at the following code:

```
ap = &a;                    // ap is pointer to object a
cout<<ap->*ip;
cout<<ap->m;
```

The dereferencing operator ->* is used to access a member when we use pointers to both the object and the member. The dereferencing operator .* is used when the object itself is used with the member pointer. Note that *ip is used like a member name.

We can also design pointers to member functions which, then, can be invoked using the dereferencing operators in the main as shown below.

(object-name.*pointer-to-member function)(10);
(pointer-to-object ->*pointer-to-member function)(10);

The precedence of ( ) is higher than that of .* and ->*, so the parentheses are necessary.

Write a Program to Demonstration of Pointer to Member.

```
#include<iostream.h>
#include<conio.h>

class M
{
  private:
      int x;
      int y;
  public:
      void setxy(int a,int b)
      {
       x=a;
       y=b;
      }
      friend int sum(M);
};

int sum(M m)
{
   int M ::* px=&M :: x;
   int M ::* py=&M :: y;
   M *pm = &m;
   int s=m.*px+pm->*py;
   return(s);
}

main()
{
M n;
void (M :: *pf)(int,int)=&M :: setxy;
(n.*pf)(10,20);
clrscr();
cout<<"Addition = "<<sum(n)<<endl;
M *op=&n;
(op->*pf)(30,40);
cout<<"Addition = "<<sum(n);
getch();
return(0);
}
```

Arrays within Class

The array can be used as member variables in a class. The following class definition is valid.

```
const int size = 10;

class no
{
        int a[size];
    public :
        void setval( );
        void display( );
};
```

The array variable a[ ] declared as a private member of the class no can be used in the member function like any other array variable.