Practical Manual

Sub:Objected Oriented Programming

By:Ms.Snehal Singh

Depatment of IT/CS

B N B Bandodkar college of Science,Thane

Topics

Sr.No	Aim	Pg no
1(a)	Design an employee class for reading and displaying the employee information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be private method.	3
1(b)	Design the class student containing getData() and displayData() as two of its methods which will be used for reading and displaying the student information respectively. Where getData() will be private method.	4
1(c)	Design the class Demo which will contain the following methods: readNo(), factorial() for calculating the factorial of a number, reverseNo() will reverse the given number, isPalindrome() will check the given number is palindrome, isArmstrong() which will calculate the given number is armStrong or not.Where readNo() will be private method.	5
1(d)	Write a program to demonstrate function definition outside class and accessing class members in function definition.	7
2(a)	Using friend functions. Write a friend function for adding the two complex numbers, using a single class	8
(b)	Write a friend function for adding the two different distances and display its sum, using two classes.	10
(c)	Write a friend function for adding the two matrix from two different classes and display its sum.	12
3(a)	Constructors and method overloading. Design a class Complex for adding the two complex numbers and also show the use of constructor.	14
(b)	Design a class Geometry containing the methods area() and volume() and also overload the area() function .	15
(c)	Design a class StaticDemo to show the implementation of static variable and static function.	16
4(a)	Operator Overloading Overload the operator unary(-) for demonstrating operator overloading.	18
(b)	Overload the operator + for adding the timings of two clocks, And also pass objects as an argument.	19
(c)	Overload the + for concatenating the two strings. For e.g “Py” + “thon” = Python	20
5(a)	Inheritance Design a class for single level inheritance using public and private type derivation	21
(b)	Design a class for multiple inheritance.	23
(c)	Implement the hierarchical inheritance.	25
6(a)	Virtual functions and abstract classes Implement the concept of method overriding.	27
(b)	Show the use of virtual function	28
(c)	Show the implementation of abstract class.	29
7(a)	String handling String operations for string length , string concatenation	30
(b)	Console formatting functions.	31
8(a)	Exception handling Show the implementation of exception handling	32
(b)	Show the implementation of exception handling for using the pointers.	33
9(a)	File handling Design a class FileDemo open a file in read mode and display the total number of words and lines in the file.	34
(b)	Design a class to handle multiple files and file operations	35
(c)	Design a editor for appending and editing the files	37
10(a)	Templates Show the implementation of template class library for swap function.	39
(b)	Design the template class library for sorting ascending to descending and viceversa	41

Practical No:1

(1) Implement the following:

(a) Design an employee class for reading and displaying the employee information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be private method.

Code:

```
#include<iostream.h>
#include<conio.h>
class employee
{
char name[20];
int age;
float basic_sal;
void getInfo()
{
cout<<endl<<"Enter name: "; cin>>name; cout<<endl<<"Enter
age: "; cin>>age;
cout<<endl<<"Enter basic salary: "; cin>>basic_sal;
}
public:
void displayInfo()
{
getInfo();
cout<<endl<<"\t Employee Information:\n";
cout<<"\t ----------";
cout<<endl<<" Name: "<<name; cout<<endl<<" Age: "<<age;
cout<<endl<<"Basic Salary:"<<basic_sal;
cout<<endl<<"Gross Salary:"<<basic_sal+(0.6*basic_sal)+ (0.4*basic_sal);
}
};

void main()

{
clrscr();
employee e;
e.displayInfo();
getch();
}
```

Output:

```
Enter name: Suman
Enter age: 35
Enter basic salary: 6000
Employee Information
----------
Name: Suman Age: 35
Basic Salary: 6000
Gross Salary: 12000
```

(b) Design the class student containing getData() and displayData() as two of its methods which will be used for reading and displaying the student information respectively. Where getData() will be private method.

Code:

```
#include<iostream.h>
#include<conio.h>
class student
{
char name[20];
int age;
float percentage;
void getData()
{
cout<<endl<<"Enter name: ";
cin>>name;
cout<<endl<<"Enter age: ";
cin>>age;
cout<<endl<<"Enter percentage: ";
cin>>percentage;}
public:
void displayData()
{
getData();
cout<<endl<<"\tSTUDENT INFORMATION\n";
cout<<"\t ----------";
cout<<endl<<"Name: "<<name;
cout<<endl<<"Age: "<<age;
cout<<endl<<"Percentage: "<<percentage;
}
};
void main()
{
clrscr();
student s;
s.displayData();
getch();
}
```

Output:

```
Enter name: Shikha
Enter age: 20
Enter percentage: 82
STUDENT INFORMATION
------------------
Name: Shikha Age: 20
Percentage: 82
```

(c) Design the class Demo which will contain the following methods: readNo() ,factorial() for calculating the factorial of a number, reverseNo() will reverse the given number, isPalindrome() will check the given number is palindrome, isArmstrong() which will calculate the given number is armStrong or not.Where readNo() will be private method.

Code:

```
#include<conio.h>
#include<iostream.h>
Class Demo
{
int n;
void readno()
{
cout<<”Enter no:”;
cin>>n;
}
public:
void factorial();
void reverse_no();
void palindrome();
};
void demo::factorial()
{
readno();
int fact=1;
for(int i=1;i<=n;i++)
{
fact=fact*i;
}
cout<<”Factorial is :”<<fact<<endl;
}
void demo::palindrome()
{
readno();
int n1,d,sum=0;
n1=n;
while(n1!=0)
{
d=n%10;
sum=d+(sum*10);
n1=n1/10;
}
if(sum==n)
{
cout<<”No is palindrome”;
 }
else
{
cout<<”No is not palindrome”;
}
void demo::reverse_no()
{
readno();
```

```
int n1,d,sum=0;
n1=n;
while(n1!=0)
{
d=n%10;
sum=d+(sum*10);
n1=n1/10;
cout<<"Reverse no is:"<<sum<<endl;
}
void main()
{
clrscr();
Demo d;
int a;
cout<<"Select the following: \n n1:factorial n2:palindrome\n n3: reverse_no \n ";
cin>>a;
switch(a)
{
case 1:
{
d.factorial;
break;
}
case 2:
{
d.palindrome;
break;
}
case 3:
{
d.reverse_no;
break;
}
default:
{
cout<<"Invalid choice";
}}
getch();
}
```

Output:

```
Select the following:
n1:factorial
n2:palindrome
n3:reverse_no    1
Enter no:5
Factorial is:120
```

(d) Write a program to demonstrate function definition outside class and accessing class members in function definition.

Code:

```
#include<iostream.h>
#include<conio.h>
class Student
{
        int  roll_no;
        char name [30];
        float percentage;
        public:
        void getdata ();
        void show ();
   };
   void Student:: getdata ()
   {
        cout <<"Enter Roll No: "; cin >> roll_no;
        cout << endl <<"Enter name: "; cin >> name;
        cout << endl <<"Enter Percentage: "; cin >> percentage;
   }
   void Student:: show ()
   {
        cout << endl <<"Roll No: " << roll_no;
        cout << endl <<"Name: " << name;
        cout << endl <<"Percentage: " <<percentage;
   }
   void main()
   {
    clrscr();
    Student studObj;
     studObj.getdata();
     studObj.show(); getch();
   }
   Output:
   Enter Roll No: 123
  Enter name: Sam
 Enter percentage: 62
 Roll No: 123
  Name: Sam
  Percentage: 62
```

(2) Implement the following

(a) Write a friend function for adding the two complex numbers, using a single class.

Code:

```
#include<iostream.h>
#include<conio.h>
class complex
{
 float n,m;
 public:
          void getData()
          {
           cout<<"\nEnter real number: "; cin>>n;
           cout<<"\nEnter imaginary number: "; cin>>m;
          }
          void showData()
          {
           cout<< n <<" + j"<< m ;
          }
  friend complex sum(complex, complex);
};
complex sum(complex c1, complex c2)
{
 complex c3;
  c3.n=c1.n+c2.n;
 c3.m=c1.m+c2.m;
 return c3;
}
void main()
{
 clrscr();
 complex obj1, obj2, obj3;
 cout<<"\nEnter Data for 1st Complex Number \n";
 cout<<" --------------------------";
 obj1.getData();
 cout<<"\nEnter Data for 2nd Complex Number \n";
 cout<<" --------------------------";
 obj2.getData();
 obj3=sum(obj1,obj2);
 cout<<"\nComplex Number1: ";
 obj1.showData();
 cout<<"\nComplex Number2: ";
 obj2.showData();
  cout<<"\nComplex Number3: ";
 obj3.showData();
 getch();
}
```

Output:

```
Enter Data for 1st Complex Number
---------------------------------
Enter real number: 2.2
Enter imaginary number: 4.5
Enter Data for 2nd Complex Number
```

```
--------------------------------
Enter real number: 6.2
Enter imaginary number: 7.2
Complex Number1:
2.2 + j4.5
 Complex Number2:
6.2 + j7.2
 Complex Number3:
 8.5+ j1.7
```

(b) Write a friend function for adding the two different distances and display its sum, using two classes.

Code:

```
#include<iostream.h>
#include<conio.h>
class distance2;
class distance1
{
 int feet;
 int inch;
 public:
void getData()
{
 cout<<"\nEnter feet: "; cin>>feet; cout<<"\nEnter inches: ";
 cin>>inch;
}
void showData()
{
 cout<< feet <<"'-" <<inch<<"\"";
}
 friend void sum(distance1, distance2);
};
class distance2
{
 int feet,inch;
 public:
void getData()
{
 cout<<"\nEnter feet: "; cin>>feet; cout<<"\nEnter inches: ";
 cin>>inch;
}
void showData()
{
 cout<< feet<<"'-"<<inch <<"\"" ;
}
  friend void sum(distance1, distance2);
};
 void sum(distance1 d1, distance2 d2)
  {
    int f=d1.feet+d2.feet;
    int i=d1.inch+d2.inch;
    if(i>=12)
            {
             i=i-12;
             f++;
            }
   cout<< f<<"'-"<<i <<"\"" ;
  }

void main()
{
 clrscr();
 distance1 obj1;
```

```
 distance2 obj2;
 cout<<"\nEnter Data for 1st Distance \n";
 cout<<" ---------- ";
 obj1.getData();
 cout<<"\nEnter Data for 2nd Distance \n";
 cout<<" ---------- ";
 obj2.getData();
 cout<<"\nDistance1: ";
 obj1.showData();
 cout<<"\nDistance2: ";
 obj2.showData();
 cout<<"\nDistance3: ";
 sum(obj1,obj2);
 getch();
}
```

Output:

```
Enter Data for 1st Distance
-------------------------
Enter feet: 16
Enter inches: 5
Enter Data for 2nd Distance
-------------------------
Enter feet: 6
Enter inches: 8
Distance1: 16'-5"
Distance2: 6'-8"
Distance3: 23'-1"
```

(c) Write a friend function for adding the two matrix from two different classes and display its sum.

Code:

```
#include<iostream.h>
#include<conio.h>
class matrix2;
class matrix1
{
 int a[3][3];
 public:
          void getData()
          {
           for(int i=0;i<3;i++)
            {
             for(int j=0;j<3;j++)
              cin>>a[i][j];
            }
          }
          void showData()
          {
           for(int i=0;i<3;i++)
            {
             for(int j=0;j<3;j++)
             cout<<a[i][j]<<" ";
              cout<<endl;
            }
          }
  friend void sum(matrix1, matrix2);
};
class matrix2
{
 int a[3][3];
public:
void getData()
{
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
cin>>a[i][j];
}
}
void showData()
{
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
cout<<a[i][j]<<" ";
cout<<endl;
}
}
friend void sum(matrix1, matrix2);
};
void sum(matrix1 m1, matrix2 m2)
{
```

```
    int a[3][3];
    for(int i=0;i<3;i++)
    {
    for(int j=0;j<3;j++)
    {
    a[i][j]=m1.a[i][j] + m2.a[i][j]; cout<<a[i][j]<<" ";
    }
    cout<<endl;
}
void main()
{
clrscr();
matrix1 obj1;
matrix2 obj2;
cout<<"\nEnter Data for 1st Matrix \n";
cout<<"----------\n";
obj1.getData();
cout<<"\nEnter Data for 2nd Matrix \n";
cout<<"----------\n";
obj2.getData();
cout<<"\nMatrix1: \n";
obj1.showData();
cout<<"\nMatrix2: \n";
obj2.showData();
cout<<"\nMatrix3: \n";
sum(obj1,obj2);
 getch();}
```

Output:

```
Enter Data for 1st Matrix
-------------------------
3 2 1
6 5 4
9 8 7
Enter Data for 2nd Matrix
-------------------------
7 8 9
4 5 6
1 2 3
Matrix1:
 3 2 1
6 5 4
   9 8 7
   Matrix2:
   7 8 9
   4 5 6
   1 2 3
   Matrix3:
    10 10 10
    10 10 10
    10 10 10
```

(3) Implement the following:

(a) Design a class Complex for adding the two complex numbers and also show the use of constructor.

Code:

```
#include<iostream.h>
 #include<conio.h>
class complex
{
 float n,m;
 public:
 complex()
{
 n=0; m=0;
}
complex(int a, int b)
{
 n=a; m=b;
}
void showData()
{
 cout<< n <<" + j"<< m ;
}
complex sum(complex c1)
{
 complex c3;
 c3.n=n+c1.n;
 c3.m=m+c1.m;
 return c3;
}
};
void main()
{
 clrscr();
 complex obj1(4,3);
 complex obj2(5,6);
 complex obj3;
 obj3=obj1.sum(obj2);
  cout<<"\nComplex Number1: ";
 obj1.showData();
 cout<<"\nComplex Number2: ";
 obj2.showData();
 cout<<"\nComplex Number3: ";
 obj3.showData();
 getch();
}
```

Output:

```
Complex Number1: 4 + j3
Complex Number2: 5 + j6
Complex Number3: 9 + j9
```

(b) Design a class Geometry containing the methods area() and volume() and also overload the area() function .

Code:

```
#include<iostream.h>
 #include<conio.h>
class geometry
{
 int l,b;
 public:
int area(int x)
{
 l=b=x;
return(l*b);
}
int area(int x, int y)
{
 l=x;
 b=y;
 return(l*b);
}
int volume(int x)
{
 l=x;
 return(l*l*l);
}
};
void main()
{
 clrscr();
geometry g;
cout<<"\nArea of square= "<<g.area(15);
cout<<"\nArea of rectangle= "<<g.area(12,10);
cout<<"\nVolume of cube= "<<g.volume(5);
getch();
}
```

Output:

```
Area of square= 225
Area of rectangle= 120
Volume of cube= 125
```

(c) Design a class Static Demo to show the implementation of static variable and static function.

Code:

```
#include<iostream.h>
 #include<conio.h>
 #include<stdio.h>
class student
{
     int roll_no;
    char name[30];
    float percent;
   static int c;
   public:
  void get()
{
cout<<"\nEnter Name:";
gets(name);
cout<<"\nEnter percentage:";
cin>>percent;
roll_no=++c;
}
void show()
{
cout<<"\nRoll No:"<<roll_no; cout<<"\nName:"<<name;
cout<<"\nPercentage:"<<percent;
cout<<"\n\n\tTotal number of students admitted:"<<c;
}
};
int student::c;
void main()
{
    clrscr();
    student s1,s2;
   s1.get();
    s2.get();
    cout<<"\n Object 1 Data";
    cout<<"\n***************";
    s1.show();
    cout<<"\nObject 2 Data";
    cout<<"\n***************";
    s2.show();
    getch();
}
```

Output:

```
Enter Name: Sagar Sinhal
 Enter percentage: 78
Enter Name: Rankesh patel
Enter percentage: 96
Object 1 Data
***************
Roll No: 1
Name: Sagar Sinhal
Percentage: 78
```

Total number of students admitted: 2
Object 2 Data
****
Roll No: 2
Name: Rankesh patel
Percentage: 96
Total number of students admitted: 2

(4) Implement the following

(a) Overload the operator unary(-) for demonstrating operator overloading.

Code:

```
#include<iostream.h>
 #include<conio.h>
class abc
{
 int a,b,c;
 public:
        void get()
        {
         cout<<"\nEnter three numbers: ";
          cin>>a>>b>>c;
        }
        void show()
        {
         cout<<"\n\nA= "<<a<<"\tB= "<<b<<"\tC= "<<c;
        }
        void operator -()
        {
         a= -a;
         b= -b;
         c= -c;
        }
};
void main()
{
 clrscr();
  abc a1;
  a1.get();
 cout<<"\n\n Original contents";
 a1.show();
 -a1;
 cout<<"\n\n After Negation";
 a1.show();
 getch();
}
```

Output:

```
Enter three numbers: 7
-9
8
Original contents
A= 7  B= -9 C= 8
After Negation
A= -7 B= 9 C= -8
```

(b) Overload the operator + for adding the timings of two clocks, And also pass objects as an argument.

Code:

```
#include<iostream.h>
#include<conio.h>
class time
{
 int hrs,min,sec;
 public:
  void get()
  {
   cout<<"\n\tEnter time (in hrs:minutes:seconds form): ";
   cin>>hrs>>min>>sec;
  }
  void show()
  {
   cout<<"\n"<<hrs<<":"<<min<<":"<<sec;
  }
 time operator +(time t2)
  {
  time t3;
  t3.sec=sec + t2.sec;
  t3.min=min + t2.min + (t3.sec/60);
   t3.sec=t3.sec%60;
  t3.hrs=hrs + t2.hrs + (t3.min/60);
  t3.min=t3.min%60;
  return t3;
  }
 };
  void main()
  {
   clrscr();
   time t1,t2,t3;
    t1.get();
   t2.get();
   t1.show();
   t2.show();
   t3=t1 + t2;
   t3.show();
   getch();
  }
```

Output:

```
Enter time (in hrs:minutes:seconds form): 10 30 40
Enter time (in hrs:minutes:seconds form): 5 40 30
10:30:40                 // time 1
5:40:30                  // time2
16:11:10                 // time1 + time2
```

(c) Overload the + for concatenating the two strings. For e.g “c” + “++” = c++

Code:

```
#include<iostream.h>
#include<conio.h>
 #include<string.h>
class string
{
 char str[60];
 public:
  void get()
  {
   cout<<"\n\tEnter a string: "; cin>>str;
  }
  void show()
  {
   cout<<"\n"<<str;
  }
 string operator +(string s2)
  {
  string s3; strcpy(s3.str,str); strcat(s3.str,s2.str); return
  s3;
  }
 };
 void main()
{
   clrscr();
   string s1,s2,s3;
   s1.get();
   s2.get();
   cout<<"\nString 1"; s1.show(); cout<<"\nString 2";
   s2.show();
   cout<<"\nAfter concatenation String 3";
    s3=s1 + s2;
   s3.show();
   getch();
  }
```

Output:

```
 Enter a string: Sanjay
Enter a string: Sinhal
String 1 Sanjay
String 2 Sinhal
After concatenation
String 3 SanjaySinhal
```

(5) Implement the following

(a) Design a class for single level inheritance using public and private type derivation.

(1) Using public type derivation:

Code:

```
#include<iostream.h>
#include<conio.h>
class base
int n;
public:
    void get()
    {
     cout<<"\nEnter value for n:"; cin>>n;
    }
    void show()
    {
     cout<<"\n\t\tN="<<n;
    }
};
  class derived:public base
  {
   int b; public:
    void get()
    {
     base::get();
     cout<<"\nEnter value for b: "; cin>>b;
    }
  };
  void main()
  {
   clrscr();
  derived d1;
  d1.get();
 d1.show();
 getch();
  }
```

Output:

```
Enter value for n:5
Enter value for b: 7
        N=5
```

(2) Using private type derivation:

Code:

```
#include<iostream.h>
 #include<conio.h>
class base
{
   int n;
 public:
    void get()
    {
     cout<<"\nEnter value for n:"; cin>>n;
    }
    void show()
```

```
        {
         cout<<"\n\t\tN="<<n;
        }
  };
    class derived:private base
    {
     int b;
     public:
          void get()
          {
           base::get();
           cout<<"\nEnter value for b: "; cin>>b;
          }
          void display()
          {
           show();
          }
    };
void main()
    {
     clrscr();
     derived d1;
     d1.get();
   // d1.show(); not accessible as its scope is private
   d1.display();
     getch();
    }
```

Output:

```
Enter value for n:5 Enter value for b: 6
          N=5
```

(b) Design a class for multiple inheritance.

Code:

```
#include<iostream.h>
#include<conio.h>
  class internal
  {
   int n;
   public:
         void get()
         {
          cout<<"\nEnter n: "; cin>>n;
         }
         int n_return()
         {
          return n;
         }
         void show()
         {
          cout<<"\n\nInternal marks: "<<n;
         }
  };
  class external
  {
   int m;
   public:
         void get()
         {
          cout<<"\nEnter m: "; cin>>m;
         }
         int m_return()
         {
           return m;
         }
         void show()
         {
          cout<<"\nM: "<<m;
         }
  };
class final:public internal, public external
         {
          float tot; public:
    void get()
    {
         internal::get();
         external::get();
    }
    void show()
    {
         tot=internal::n_return()+external::m_return();
          cout<<"\nTotal: "<<tot;
    }
};
```

```
void main()
{
 clrscr(); final t; t.get();
 t.show();
 getch();
 }
```

Output:

```
Enter n: 4
Enter m: 4
Total: 8
```

(c) Implement the hierarchical inheritance.

Code:

```
#include<iostream.h>
#include<conio.h>
class person
{
 char name[30];
 int age;
 public:
        void getdata()
        {
         cout<<"\nEnter name and age: ";
          cin>>name>>age;
        }
        void showdata()
        {
         cout<<"\nName: "<<name; cout<<"\nAge: "<<age;
        }
};
 class student:public person
 {
  int marks;
  public:
  void get()
{
         getdata();
         cout<<"\nEnter marks: ";
         cin>>marks;
        }
        void show()
        {
         showdata();
         cout<<"\nMarks: "<<marks;
        }
 };
 class employee:public person
 {
  char design[30]; public:
        void get()
        {
         getdata();
         cout<<"\nEnter designation: "; cin>>design;
        }
void show()
        {
         showdata(); cout<<"\nDesignation: "<<design;
        }
 };
  void main()
  {
   clrscr();
   student s;
```

```
    employee e;
    cout<<"\nEnter student's data";
     s.get();
    cout<<"\nEnter employee's data";
     e.get();
    cout<<"\n\nstudent's Data";
    s.show();
     cout<<"\n\nEmployee's Data";
    e.show();
    getch();
   }
```

Output:

```
Enter student's data
Enter name and age: Sanjeev 33
Enter marks: 72

Enter employee's data
Enter name and age: Rakesh 35
Enter designation: Manager

student's Data Name: Sanjeev
 Age: 33
Marks: 72
Employee's Data Name: Rakesh Age: 35
Designation: Manager
```

(6) Implement the following

(a) Implement the concept of method overriding. #include<iostream.h>
#include<conio.h>

Code:

```
class employee
  {
   int emp_code,age;
   char name[30], qualification[30];
   public:
         void get()
         {
          cout<<"\nEnter employee id: "; cin>>emp_code;
          cout<<"\nEnter employee name: "; cin>>name;
          cout<<"\nEnter employee age: "; cin>>age;
          cout<<"\nEnter employee qualification: "; cin>>qualification;
         }
         void show()
         {
          cout<<"\n\nEmployee id: "<<emp_code;
          cout<<"\tName: "<<name;
          cout<<"\nAge: "<<age<<"\t\tQualification: "<<qualification;
         }
  };
  class contract_employee: public employee
  {
   int contract_id;
    public:
         void get()
         {
          cout<<"\nEnter contract_id: ";
           cin>>contract_id;
         }
void show()
         {
          cout<<"\nContract ID: "<<contract_id;
         }
  };
  void main()
  {
   clrscr();
   contract_employee ce;
   ce.get();
   ce.show();
   getch();
   }
```

Output:

```
Enter contract_id: 123
Contract ID: 123
```

(b) Show the use of virtual function

Code:

```
#include<iostream.h>
#include<conio.h>
class base
{
public:
 virtual void display()
 {
  cout<<"\nDisplay of base class called";
 }
};
class derived:public base
{
public:
 void display()
 {
  cout<<"\nDisplay of derived class called";
 }
};
void main()
{
 clrscr();
 base *b;
 derived d;
  b=&d;
 b->display();
 getch();
}
```

Output:

```
Display of derived class called
```

(c) Show the implementation of abstract class.

Code:

```
#include<iostream.h>
#include<conio.h>
// Using abstract methods and classes. class Figure
{
 public:
 double dim1;
 double dim2;
  Figure(double a, double b)
 {
  dim1 = a; dim2 = b;
 }
// pure virtual function virtual double area()=0;
};
class Rectangle:public Figure
 {
public:
 Rectangle(double a, double b):Figure(a,b)
 {
 }
// implement area for rectangle
double area()
 {
 cout<<"\nInside Area for Rectangle:"; return dim1 * dim2;
 }
};
class Triangle:public Figure
{
public:
Triangle(double a, double b):Figure(a,b)
 {
 }
 // implement area for right triangle
 double area()
  {
  cout<<"\nInside Area for Triangle:";
  return dim1 * dim2 / 2;
 }
};
void main()
{
clrscr();
Rectangle r(9, 5);
Triangle t(10, 8);
cout<< r.area();
cout<< t.area();
 getch();
 }
```

Output:

```
Inside Area for Rectangle:45
 Inside Area for Triangle:40
```

(7) Implement the following:
(a)String operation for string length ,string concatenation and string reverse.

Code:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
void main()
{
clrscr();
char a[20],b[20],c[20];
int len;
cout<<"Enter the first name:";
cin>>a;
cout<<"Enter the second name:";
cin>>b;
len=strlen(a);
cout<<"Length of first name:"<<len<<"\n";
len=strlen(b);
cout<<"Length of second  name:"<<len<<"\n";
strcpy(c,b);
cout<<"Copied string is:"<<c;
strrev(c);
cout<<"Reverse of second string is:"<<c<<"\n";
if(strcmp(a,b)==0)
{
cout<<"String are same";
}
else
{
cout<<"String are not same";
}
strcat(a," ");
strcat(a,b)'
cout<<"Concatenation of two string is:"<<a;
getch();
}
```

Output:
Enter first name:Sayali
Enter second name:Patil
Length of first name:6
Length of second name:5
Copied string is:Patil
Reverse of second string is:litaP
String are not same
Concatenation of two different string:SayaliPatil

(b)Console formatting functions.

Example 1:

Without using iomanip setw() method.

Code:

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
char s[10];
cout<<"Enter your name: "; cin>>s;
cout<<endl<<s; getch();
}
```

Output:

```
Enter your name: SanjeelaSagar SanjeelaSagar
```

With using iomanip setw() method.

Code:

```
#include<iostream.h>
 #include<conio.h>
 #include<iomanip.h>
void main()
{
clrscr(); char s[10];
cout<<"Enter your name: "; cin>>setw(10)>>s; cout<<endl<<s;
getch();
}
```

Output:

```
Enter your name: SanjeelaSagar SanjeelaS
```

(8) Implement the following:

(a) Show the implementation of exception handling #include<iostream.h>

```
void main()
{
 float percent;
 cout<<"Enter your percentage: ";
cin>>percent;
try
{
if(percent<0 || percent>100)
throw(percent);
else
cout<<endl<<"Your percentage: "<<percent;
}
catch(int p)
{
cout<<endl<<"Invalid percentage: "<<p;
}
}
```

Output:

Enter percentage: 150
Invalid percentage: 150

(b)Show the implementation for exception handling for strings

Code:

```
#include<iostream>
#include<string>
 using namespace std;
void main()
{
string s;
cout<<"Enter the name of your course: ";
cin>>s;
else
try
{
if(s=="B.Sc - IT" || s=="BMS" || s=="B.Com")
cout<<endl<<"Your have chosen Course: "<<s;
throw(s);
}
catch(string ss)
{

cout<<endl<<"Oh!!!!!!!!! you have chosen the course that we don't provide: "<<ss;

}
```

Output:

1st Run:
Enter the name of your course: MCA
Oh!!!!!!!!! you have chosen the course that we don't provide: MCA
2nd Run:
Enter the name of your course: BMS
You have chosen Course: BMS

(9) Show the implementation

(a) Design a class FileDemo, open a file in read mode and display the total number of words and lines in the file.

Code:

```
#include<iostream.h>
 #include<conio.h>
 #include<fstream.h>
void main()
{
 clrscr();
 ifstream fread("WordLineCount.txt"); int wc=1,lc=1;
 char c; while(fread)
 {
   fread.get(c);
   if(c==' '|| c=='\n')
 wc++;
  if(c=='\n')
   lc++;
 }
 fread.close();
 cout<<"\n Total no. of words in the file: "<<wc;
  cout<<"\n Total no. of lines in the file: "<<lc;
 getch();
}
```

Output:

```
Contents of the file
---------------------------------
This is the first line of the file. This is the second line of the file.
This is the third line of the file.
 Total no. of words in the file: 24 Total no. of lines in the file: 3
```

(b) Design a class to handle multiple files and file operations

Code:

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
clrscr();
ofstream fwrite("Alphabets.txt");
fwrite<<"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
fwrite.close();
ifstream fread("Alphabets.txt");
ofstream fwrite1("Vowels.txt");
ofstream fwrite2("Consonants.txt");
char c;
while(fread)
{
fread.get(c);
if(c=='A' || c=='E' || c=='I' || c=='O' || c=='U')
fwrite1<<c;
else
fwrite2<<c;
}
fread.close();
fwrite1.close();
fwrite2.close();
fread.open("Alphabets.txt");
ifstreamfread1(“Vowels.txt”);
ifstreamfread2("Consonants.txt”);
cout<<"\n\nContents of Alphabets File\n";
cout<<”-----------\n";
while(fread)
{
fread.get(c);
cout<<c;
}
fread.close();
cout<<"\n\nContents of Vowels File\n";
 cout<<"----------\n";
while(fread1)
{
fread1.get(c); cout<<c;
}
fread1.close();
cout<<"\n\nContents of Consonants File\n"; cout<<"   \n";
while(fread2)
{
fread2.get(c); cout<<c;
}
fread2.close(); getch();
}
```

Output:

Contents of Alphabets File

---------- ABCDEFGHIJKLMNOPQRSTUVWXYZ

Contents of Vowels File

AEIOU
Contents of Consonants File

BCDFGHJKLMNPQRSTVWXYZ

(c) Design a editor for appending and editing the files

Code:

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class student
{
     char name[30];
     int age;
     float percent;
      public:
void getdata()
 {
     cout<<endl<<"Enter name: "; cin>>name; cout<<endl<<"Enter
     age: "; cin>>age;
     cout<<endl<<"Enter percentage: "; cin>>percent;
 }
void showdata()
 {
     cout<<endl<<name;
     cout<<"\t\t"<< age;
     cout<<"\t\t"<<percent;
 }
};
void main()
{
 clrscr();
 student st;
fstream freadwrite("Student.txt", ios::ate | ios::in | ios::out);
freadwrite.seekg(0,ios::beg);
cout<<endl<<"Current contents of file";
while(freadwrite.read((char*)&st,sizeof(st)))
st.showdata();
freadwrite.clear();
cout<<endl<<"Enter details for one more student";
 st.getdata();
char c;
cin.get(c);
freadwrite.write((char*)&st, sizeof(st));
freadwrite.seekg(0);
cout<<endl<<"After addition of one more student";
cout<<endl<<"Name \t\t Age \t\t Percentage";
 while(freadwrite.read((char*)&st, sizeof(st)))
 {
     st.showdata();
 }
int n = freadwrite.tellg() / sizeof(st);
 cout<<endl<<"Total no. of student record: "<<n;

cout<<endl<<"Enter student number to be updated: "; int num;
cin>>num;
cin.get(c);
```

```
int l=(num-1) * sizeof(st);
if(freadwrite.eof())
freadwrite.clear();
freadwrite.seekp(l);
    cout<<endl<<"Enter new values for the student";
    st.getdata();
    cin.get(c);
   freadwrite.write((char*)&st, sizeof(st))<<flush; freadwrite.seekg(0);
 cout<<endl<<"After updation contents are";
 cout<<endl<<"Name \t\t Age \t\t Percentage";
 while(freadwrite.read((char*)&st, sizeof(st)))
 {
  st.showdata();
 }
freadwrite.close();
getch();
}
```

Output:

```
Current contents of file
     Name              Age      Percentage
Sanjeela               33              78
Rakesh                 34              82
Sagar                  22              67
Enter details for one more student Enter name: Kashyap
Enter age: 19
Enter percentage: 89
After addition of one more student
     Name         Age          Percentage
Sanjeela               33              78
Rakesh                 34              82
Sagar                  22              67
Kashyap                      19              89
Total no. of student record: 4
Enter student number to be updated:
4 Enter new values for the student
Enter name:Kashyap
Enter Age:19
Enter percentage: 80
After updation contents are

    Name          Age              Percentage
     Sanjeela            33              78
      Rakesh             34              82
      Sagar              22              67
     Kashyap             20              80
```

(10)Show the implementation for the following

(a) Show the implementation of template class library for swap function.

Code:

```
#include<iostream.h>
#include<conio.h>
template<class A>
void swap(A &a, A &b)
  {
   A t=a; a=b; b=t;
  };
  void main()
  {
   clrscr();
   int n,m; float f1,f2;
   char c,d;
   cout<<endl<<"Enter two integers: "; cin>>n>>m;
   cout<<endl<<"Enter two floats: "; cin>>f1>>f2;
   cout<<endl<<"Enter two characters: "; cin>>c>>d;
   cout<<endl<<"Integers before swapping\n";
   cout<<" ------------------";
   cout<<endl<<"N= "<<n<<"\tM= "<<m; swap(n,m);
   cout<<endl<<"\nIntegers after swapping\n";
    cout<<"------------------ ";
   cout<<endl<<"N= "<<n<<"\tM= "<<m;
   cout<<endl<<"\nFloats before swapping\n";
   cout<<" ------------------";
  cout<<endl<<"F1= "<<f1<<"\tF2= "<<f2;
  swap(f1,f2);
   cout<<endl<<"\nFloats after swapping\n";
    cout<<"------------------";
   cout<<endl<<"F1= "<<f1<<"\tF2= "<<f2;
   cout<<endl<<"\nCharacters before swapping\n";
    cout<<"---------------------";
   cout<<endl<<"C= "<<c<<"\tD= "<<d;
   swap(c,d);
   cout<<endl<<"\nCharacters after swapping\n";
   cout<<" ------------------";
   cout<<endl<<"C= "<<c<<"\tD= "<<d;
   getch();
  }
```

Output:

```
Enter two integers: 10   20
Enter two floats: 10.5  11.5
Enter two characters: c  d
Integers before swapping
-----------------------
 N= 10 M= 20
Integers after swapping
----------------------
N= 20 M= 10
Floats before swapping
```

```
-----------------------
 F1= 10.5    F2= 11.5
Floats after swapping
-----------------------
 F1= 11.5    F2= 10.5
Characters before swapping
-------------------------
C= c  D= d
Characters after swapping
-------------------------
C= d  D= c
```

(b) Design the template class library for sorting ascending to descending and vice-versa

Code:

```
#include<iostream.h>
#include<conio.h>
template<class A>
void sort_asc(A *a, int n)
{
int i,j;
A t;
for(i=0;i<n-1;i++)
{
for(j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
t=a[i]; a[i]=a[j];
a[j]=t;
}
}
}
};
void main()
{
clrscr();
int a[10],i;
float f[10];
 char c[10];
cout<<endl<<"Enter 10 integers\n";
for(i=0;i<10;i++)
cin>>a[i];
cout<<endl<<"Enter 10 floats\n";
 for(i=0;i<10;i++)
cin>>f[i];
cout<<endl<<"Enter 10 characters\n";
for(i=0;i<10;i++)
cin>>c[i];
 cout<<endl<<"Integers before sorting\n";
 for(i=0;i<10;i++)
cout<<a[i]<<"\t";
 sort_asc(a,10);
cout<<endl<<"Integers after sorting\n";
for(i=0;i<10;i++)
cout<<a[i]<<"\t";
cout<<endl<<"Floats before sorting\n";
```

```
 for(i=0;i<10;i++)
cout<<f[i]<<"\t"; sort_asc(f,10);
cout<<endl<<"Floats after sorting\n";
 for(i=0;i<10;i++)
cout<<f[i]<<"\t";
cout<<endl<<"Characters before sorting\n";
for(i=0;i<10;i++)
cout<<c[i]<<"\t";
sort_asc(c,10);
cout<<endl<<"Characters after sorting\n";
for(i=0;i<10;i++)
cout<<c[i]<<"\t";
getch();
    }
```

Output:

```
Enter 10 integers
10 9 8 7 6 5 4 3 2 1

Enter 10 floats
10.9 9.8 8.7 7.6 6.5 5.4 4.3 3.2 2.1 1.1

Enter 10 characters z x c v b n m a s d

Integers before sorting
10    9    8    7    6    5    4    3    2    1

Integers after sorting
1    2    3    4    5    6    7    8    9    10
Floats before sorting
10.9  9.8   8.7   7.6   6.5   5.4   4.3   3.2   2.1   1.1

Floats after sorting
1.1   2.1   3.2   4.3   5.4   6.5   7.6   8.7   9.8   10.9

Characters before sorting
z    x    c    v    b    n    m    a    s    d

Characters after sorting
a    b    c    d    m    n    s    v    x    z
```