PRACTICAL

FILE

Department: Computer Science and Engineering

Session: July - December

Subject: Object Oriented Programming Using C++ Lab

Subject Code: BTCS-309

Semester: 3rd

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

Syllabus

1.[Classes and Objects]Write a program that uses a class where the member functions are defined inside a class.

2.[Classes and Objects]Write a program that uses a class where the member functions are defined outside a class.

3.[Classes and Objects]Write a program to demonstrate the use of static data members.

4.[Classes and Objects]Write a program to demonstrate the use of const data members.

5.[Constructors and Destructors]Write a program to demonstrate the use of zero argument and parameterized constructors.

6. [Constructors and Destructors]Write a program to demonstrate the use of dynamic constructor.

7.[Constructors and Destructors]Write a program to demonstrate the use of explicit constructor.

8.[Initializer Lists]Write a program to demonstrate the use of initializer list.

9.[Operator Overloading]Write a program to demonstrate the overloading of increment and decrement operators.

10.[Operator Overloading]Write a program to demonstrate the overloading of binary arithmetic operators.

11.[Operator Overloading]Write a program to demonstrate the overloading of memory management operators.

12.[Typecasting]Write a program to demonstrate the typecasting of basic type to class type.

13.[Typecasting]Write a program to demonstrate the typecasting of class type to basic type.

14.[Typecasting]Write a program to demonstrate the typecasting of class type to class type.

15.[Inheritance]Write a program to demonstrate the multilevel inheritance.

16.[Inheritance]Write a program to demonstrate the multiple inheritance.

17.[Inheritance]Write a program to demonstrate the virtual derivation of a class.

18.[Polymorphism]Write a program to demonstrate the runtime polymorphism.

19.[Exception Handling]Write a program to demonstrate the exception handling.

20.[Templates and Generic Programming]Write a program to demonstrate the use of function template.

21.[Templates and Generic Programming]Write a program to demonstrate the use of class template.

22.[File Handling]Write a program to copy the contents of a file to another file byte by byte. The name of the source file and destination file should be taken as command-line arguments,

23.[File Handling]Write a program to demonstrate the reading and writing of mixed type of data.

24.[File Handling]Write a program to demonstrate the reading and writing of objects.

SRI SUKHMANI INSTITUTE OF ENGINEERING & TECHNOLOGY

Affiliated to PTU, & Approved by AICTE

List of Practical

Sr. No.	Topic
1	Write a program that uses a class where the member functions are defined inside a class.
2	Write a program that uses a class where the member functions are defined outside a class.
3	Write a program to demonstrate the use of static data members.
4	Write a program to demonstrate the use of const data members.
5	Write a program to demonstrate the use of zero argument and parameterized constructors.
6	Write a program to demonstrate the use of dynamic constructor.
7	Write a program to demonstrate the use of explicit constructor.
8	Write a program to demonstrate the use of initializer list.
9	Write a program to demonstrate the overloading of increment and decrement operators.
10	Write a program to demonstrate the overloading of binary arithmetic operators.
11	Write a program to demonstrate the overloading of memory management operators.
12	Write a program to demonstrate the typecasting of basic type to class type.
13	Write a program to demonstrate the typecasting of class type to basic type.
14	Write a program to demonstrate the typecasting of class type to class type.
15	Write a program to demonstrate the multilevel inheritance.
16	Write a program to demonstrate the multiple inheritance.
17	Write a program to demonstrate the virtual derivation of a class.
18	Write a program to demonstrate the runtime polymorphism.
19	Write a program to demonstrate the exception handling.
20	Write a program to demonstrate the use of function template

21	Write a program to demonstrate the use of class template.
22	WAP to copy the contents of a file to another file.
23	WAP to demonstrate the read and write of file operation
24	Write a program to demonstrate the reading and writing of objects.
25.	*Write a program Concept of Abstraction, Data Hiding, and Encapsulation.

*Learning Beyond Syllabus Write a program Concept of Abstraction, Data Hiding, and Encapsulation.

Experiment 1

AIM: Write a program that uses a class where the member functions are defined inside a class.

```
#include <iostream.h>
#include<conio.h>
class book
{
Private:

int bookno;
char bname[30];
char auname[30];
float bprice;

public:

void getdata()
{
cout<<”enter the details”<<endl;
cout<<”enter the book no”;
cin>>bookno;
cout<<”enter the book name”;
cin>>bname;
cout<<”enter author name”;
cin>>auname;
cout<<”enter the book price”;
cin>>bprice;
}

void display()
{
cout<<”enter the record”<<endl;
cout<<”book no”<<book no<<endl;
cout<<”book name”<<bname<<endl;
cout<<”author name”<<auname<<endl;
cout<<”book price”<<bprice<<endl;
}
};

void main()
{
class book obj;
clrscr();
obj.getdata();
```

```
obj.display();
getch();
}
```

Output:

```
enter the details
enter the book no 145
enter the book name c
enter author name abc
enter book price 162

enter the record
book no 145
book name c
author name abc
book price 162
```

Experiment 2

AIM: Write a program that uses a class where the member functions are defined outside a class.

```
#include <iostream.h>
#include<conio.h>
class book
{
private:

int bookno;
char bname[30];
char auname[30];
float bprice;

public:

void getdata();
void display();
};

void book::void getdata()
{
cout<<"enter the details"<<endl;
cout<<"enter the book no";
cin>>bookno;
cout<<"enter the book name";
cin>>bname;
cout<<"enter author name";
cin>>auname;
cout<<"enter the book price";
cin>>bprice;
}

void book ::void display()
{
cout<<"enter the record"<<endl;
cout<<"book no"<<book no<<endl;
cout<<"book name"<<bname<<endl;
cout<<"author name"<<auname<<endl;
cout<<"book price"<<bprice<<endl;
}
void main()
{
class book obj;
```

```
clrscr();
obj.getdata();
obj.display();
getch();
}
```

Output:

```
enter the details
enter the book no 147
enter the book name oops
enter author name abc
enter book price 200

enter the record
book no 147
book name oops
author name abc
book price 200
```

Experiment 3

AIM: Write a program to demonstrate the use of static data members.

```
#include <iostream.h>
#include<conio.h>
class item
{
private:

static int count;
int n;

public:

void in(int x)
{
n=x;
count++;
}
void displaycount()
{
cout<<"count=";
cout<<count<<endl;
}
};
int item::count;
void main()
{  item p1,p2;
clrscr();
p1.displaycount();
p2.displaycount();
p1.in(7);
p2.in(8);
cout<<"after input value"<<endl;
p1.displaycount();
p2.displaycount();
getch();
}
```

Output:

```
count=0
count=0
after input value
count=1
count=2
```

Experiment 4

AIM: Write a program to demonstrate the use of const data members.

```
#include <iostream.h>
#include<conio.h>
class date
{
private:

int month, day, year;
date()
{
}
public:

date (int nm, int nd, int ny)
{
setdate(nm,nd,ny);
}

void setdate (int nm, int nd, int ny)
{
month = nm;
day = nd;
year = ny;
}

int getmonth() const
{
return month;
}

int getday() const
{
return day;
}

int getyear() const
{
return year;
}
};

void printdate (const date &cdate)
{
```

```
cout<<cdate.getmonth();
cout<<cdate.getday();
cout<<cdate.getyear();
cout<<"/"<<cdate.getyear()<<endl;}
int main ()
{
clrscr();
const date cdate (1,10,2013)
printdate (cdate);
getch();
return 0;
}
```

Output:

1/10/2013

Experiment 5

AIM: Write a program to demonstrate the use of zero argument and parameterized constructors.

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

class demo
{
private:

int  m,n;

public:

demo (int a, int b)
{
m=a;
n= b;
}

void display()
{
cout<<" The values are";
cout<<m<<endl<<n;
}
};

void main()
{
Demo obj (5,6);
obj.display();
}
```

Output:

```
The values are
5
6
```

Experiment 6

AIM: Write a program to demonstrate the use of dynamic constructor.

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

class dyncons
{

private:

int  *p;

public:

dyncons()
{
p=new int;
*p=10;
}

dyncons(int v)
{
p= new int;
*p=v;
}

int dis()
{
return(*p);
}};

void main()
{
clrscr();
dyncons o,o1(9);
cout<<"the value of object o's p is:";
cout<<o.dis();
cout<<"\n the value of object of o1's p is:"<<o1.dis();
getch();
}
```

Output:

```
The value of object o's p is: 10
The value of object o1's p is: 9
```

Experiment 7

AIM: Write a program to demonstrate the use of explicit constructor.

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

class explicit
{

Int data;

public:

explicit (int a):data(a)
{
cout<<”A::constructor….\n”;
cout<<”value of data :=”<<data<<endl;
}
};

int main()
{
explicit a1=37;
return 0;
getch();
}
```

Output:

```
A::constructor……..
Value of data:=37
```

Experiment 8

AIM: Write a program to demonstrate the use of initializer list.

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

class point

{

private:

int x,y;

public:

point (int i=0, int j=0) : x(i),y(j){}
int getx() const{return x;}
int get y() const {return y;}
};

int main()

{

clrscr();
point t1(10,15);
cout<<"x="<<ti.getx()<<"," ;
cout<<"y="<<t1.gety();
getch();
return 0;

}
```

Output:

x = 3
y =27

Experiment 9

AIM: Write a program to demonstrate the overloading of increment and decrement operators.

```
#include<iostream>
#include<conio.h>
class overloading
{
 int value;

public:

 void setValue(int temp)
 {
     value = temp;
 }

overloading operator+(overloading ob)
 {
  overloading t;
  t.value=value+ob.value;
  return(t);
  }
void display()
{
 cout<<value<<endl;
}
};

int main()
{
   overloading obj1,obj2,result;
   int a,b;
   cout<<"Enter the value of Complex Numbers a,b:";
   cin>>a>>b;
   obj1.setValue(a);
   obj2.setValue(b);
   result = obj1+obj2;
   cout<<"Input Values:\n";
   obj1.display();
   obj2.display();
   cout<<"Result:";
   result.display();
   getch();
   return 0;
}
```

Output:

```
Enter the value of Complex Numbers a,b:10
5
Input Values:
10
5
Result:15
```

Experiment 10

AIM:Write a program to demonstrate the overloading of binary arithmetic operators.

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

class airthmatic
{
float n;

 public:
 void get()
{
cout<"\n enter number:\n";
cin>>n;
}
arithmetic operator +(airthmatic &a)
{
aritjmatic t;
t.n=n+a.n;
return t;
}
arithmetic operator -(airthmatic &a)
{
aritjmatic t;
t.n=n-a.n;
return t;
}
arithmetic operator *(airthmatic &a)
{
aritjmatic t;
t.n=n*a.n;
return t;
}
arithmetic operator /(airthmatic &a)
{
aritjmatic t;
t.n=n/a.n;
return t;
}
void display()
{
cout<<n;
}
};
```

```
void main()
{
arithmetic a1,a2,a3;
clrscr();
a1.get();
a2.get();
a3 = a1+a2;
cout<<"\n addition of two number:";
a3.display();
a3 = a1-a2;
cout<<"\n subtraction of two number:";
a3.display();
a3 = a1*a2;
cout<<"\n multiplication of two number:";
a3.display();
a3 = a1/a2;
cout<<"\n division of two number:";
a3.display();
getch();
}
```

Output:

```
Enter number 12
Enter number 3
Addition of two number : 15
Subtraction of two number : 9
Multiplication of two number : 36
Division of two number : 4
```

Experiment 11

AIM: Write a program to demonstrate the overloading of memory management operators.

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

class op
{
 public:
 void *operator new(size_tsize,char const*file, int line);
void operator delete(void *p);
}
void *op::operator new (size_tsize,char const*file, int line)
{
void *p=malloc(size);
cout<<"\n new called:"<<file<<"\n line"<<line;
cout<<"\n size"<<size<<"\n p:"<<p<<endl;
return p;
}
void op::operator delete (void *p)
{
cout<<"\n deletecalled:"<<p<<endl;
free(p);
}
void main()
{
clrscr();
op *x= new(--file,--,--Line--)op;
delete x;
getch();
}
```

Output:

```
new called 11.cpp
line 25
size 1
p:ox8fc20de4
delete called ox8fc20de4
```

Experiment 12

AIM: Write a program to demonstrate the typecasting of basic type to class type.

```
#include<iostream>
#include<conio.h>
class distance
{
int feet,inch;
public:
distance()
{
m=m*100;
m=m*393700787;
inch=int(m)%12;
feet= m/12;
}
operator float()
{
float m;
m=((feet*12)+inch)*2.54;
m=m/100;
return(m);
}
void putdata()
{
cout<<feet<<"feet";
cout<<inch<<"inch";
}
};
void main()
{
clrscr();
float meter;
cout<<"\n enter  the length in meter:";
cin>>meter;
distance d1;
d1=meter;
cout<<"\n basic to class conversion";
d1.putdata();
getch();
}
```

Output:

```
enter the length in meter: 45
basic to calss conversion
147 feet 7 inch
```

Experiment 13

AIM: Write a program to demonstrate the typecasting of class type to basic type.

```
#include<iostream.h>
#include<conio.h>
class distance
{
int feet,inch;
public:
distance()
{
feet=0;
inch=0;
}
void getdata()
{
cout<<"\nEnter the feet and inch ";
cin>>feet>>inch;
}
operator float()
{
float m;
m=((feet*12)+inch)*2.54;
m=m/100;
return(m);
}
};
void main()
{
clrscr();
distance d2;
d2.getdata();
float meter=d2;
cout<<"\nClass to Basic conversion\nlength= "<<meter;
getch();
}
```

Output:

```
enter the feet and inch 6
4
class to basic conversion
length = 4.9784
```

Experiment 14

AIM: Write a program to demonstrate the typecasting of class type to class type.

```
# include <iostream.h>
# include <conio.h>
class in1
{
int code,items;
float price;
public:
in1(int a,int b,int c)
{
code=a;
items=b;
price=c;
}
void putdata()
{
cout<<"CODE= "<<code<<endl;
cout<<"ITEMS= "<<items<<endl;
cout<<"VALUE= "<<price<<endl;
}
int getcode()
{
return code;
}
int getitems()
{
return items;
}
int getprice()
{
return price;
}
operator float ()
{
return items*price;
}
};

class in2
{
int code;
float value;
public:
```

```
in2()
{
code=0;
value=0;
}
in2(int x,float y)
{
code=x;
value=y;
}
void putdata()
{
cout<<"CODE= "<<code<<endl;
cout<<"VALUE= "<<value<<endl;
}
in2(in1 p)
{
code=p.getcode();
value=p.getitems()*p.getprice();
}
};

void main()
{
clrscr();
in1 s1(100,51,145.0);
float tot_value;
in2 d1;
tot_value=s1;
d1=in1(s1);
cout<<"PRODUCT DETAILS INVENT-1 TYPES:->"<<endl;
s1.putdata();
cout<<"STOCK VALUE"<<endl;
cout<<"VALUE= "<<tot_value<<endl;
cout<<"PRODUCT DETAILS INVENT-2 TYPES:->"<<endl;
d1.putdata();
getch();
return 0;
}
```

Output:

```
Product details invent-1 types:
Code=100
Items=51
Value=145
Stock value = 7395
Product details invent- 2 types:
Code=100
Value=7395
```

Experiment 15

AIM: Write a program to demonstrate the multilevel inheritance.

```
#include<iostream.h>
#include<conio.h>
class top //base class
{
public :
int a;
void getdata()
{
cout<<"\n\nEnter first Number :::\t";
cin>>a;
}
void putdata()
{
cout<<"\nFirst Number Is :::\t"<<a;
}
};
class middle :public top // class middle is derived_1
{
public:
int b;
void square()
{
getdata();
b=a*a;
cout<<"\n\nSquare Is :::"<<b;
}
};
class bottom :public middle // class bottom is derived_2
{
public:
int c;
void cube()
{
square();
c=b*a;
cout<<"\n\nCube :::\t"<<c;
}
};
int main()
{
clrscr();
bottom b1;
b1.cube();
getch();
```

```
return 0;
}
```

Output:

```
Enter first number:12
Squae is: 144
Cube is:1728
```

Experiment 16

AIM: Write a program to demonstrate the multiple inheritance.

```
#include<iostream.h>
#include<conio.h>
class student
{
protected:
int rno,m1,m2;
public:
void get()
{
cout<<"Enter the Roll no :";
cin>>rno;
cout<<"Enter the two marks :";
cin>>m1>>m2;
}
};
class sports
{
protected:
int sm; // sm = Sports mark
public:
void getsm()
{
cout<<"\nEnter the sports mark :";
cin>>sm;

}
};
class statement:public student,public sports
{
int tot,avg;
public:
void display()
{
tot=(m1+m2+sm);
avg=tot/3;
cout<<"\n\n\tRoll No : "<<rno<<"\n\tTotal : "<<tot;
cout<<"\n\tAverage : "<<avg;
}
};
void main()
{
clrscr();
statement obj;
obj.get();
obj.getsm();
obj.display();
```

```
getch();
}
```

Output:

```
Enter the roll no :1251634
Enter the two marks:56
65
The sports marks:64
Roll no: 1251634
Total: 190
Avg: 63
```

Experiment 17

AIM: Write a program to demonstrate the virtual derivation of a class.

```
#include<iostream.h>

#include<conio.h>

class student
{
   int rno;
  public:
   void getnumber()
   {
            cout<<"Enter Roll No:";
            cin>>rno;
   }
   void putnumber()
   {
            cout<<"\n\n\tRoll No:"<<rno<<"\n";
   }
};

class test:virtual public student
{

  public:
   int part1,part2;
   void getmarks()
   {
            cout<<"Enter Marks\n";
            cout<<"Part1:";
            cin>>part1;
            cout<<"Part2:";
            cin>>part2;
   }
   void putmarks()
   {
            cout<<"\tMarks Obtained\n";
            cout<<"\n\tPart1:"<<part1;
            cout<<"\n\tPart2:"<<part2;
   }
};

class sports:public virtual student
{

  public:
    int score;
```

```
    void getscore()
    {
            cout<<"Enter Sports Score:";
            cin>>score;
    }
    void putscore()
    {
            cout<<"\n\tSports Score is:"<<score;
    }
};

class result:public test,public sports
{
    int total;
  public:
   void display()
   {
     total=part1+part2+score;
     putnumber();
     putmarks();
     putscore();
     cout<<"\n\tTotal Score:"<<total;
   }
};

void main()
{
   result obj;
   clrscr();
   obj.getnumber();
   obj.getmarks();
   obj.getscore();
   obj.display();
   getch();
}
```

Output:

```
Enter Roll No: 200

Enter Marks
 Part1: 90
Part2: 80
Enter Sports Score: 80


Roll No: 200
Marks Obtained
Part1: 90
Part2: 80
Sports Score is: 80
Total Score is: 250
```

Experiment 18

AIM: WAP to demonstrate Run time Polymorphism

```
#include<iostream.h>
#include<conio.h>
class Account
{
protected:
int acc_no;
public:
Account(int ac)
{
acc_no = ac;
}
virtual void display()
{ } //Empty Virtual Function
};

class Saving: public Account
{
int sav_amount;
public:
Saving(int ac, int s_am):Account(ac)
{
sav_amount = s_am;
}
void display();
};
void Saving::display()
{
cout<<"The Saving Account No : " <<acc_no<<endl;
cout<<"The Saving Account Amount : "<<sav_amount<<endl;
}

class Current: public Account
{
int cur_amount;
public:
Current(int ac, int c_am): Account(ac)
{
cur_amount = c_am;
}
```

```
void display();
};

void Current::display()
{
cout<<"The Current Account No : "<<acc_no<<endl;
cout<<"The Current Account Amount : "<<cur_amount<<endl;
}

void main()
{
Saving sav(01, 5000);
Current cur(02, 10000);
clrscr();
Account *acc; //Base Class Pointer

acc = &sav;
acc->display(); //display() From Saving Class

acc = &cur;
acc->display(); //display() From Current Class.
getch();
}
```

Output:

```
The Saving Account No : 1
The Saving Account Amount : 5000
The Current Account No : 2
The Current Account Amount : 10000
```

Experiment 19

AIM: WAP to demonstrate the exception handling.

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int a,b,c;
  float  d;
  clrscr();
  cout<<"Enter the value of a:";
  cin>>a;
  cout<<"Enter the value of b:";
  cin>>b;
  cout<<"Enter the value of c:";
  cin>>c;

  try
  {
        if((a-b)!=0)
        {
          d=c/(a-b);
          cout<<"Result is:"<<d;
        }
        else
        {
          throw(a-b);
        }
  }

  catch(int i)
  {
        cout<<"Answer is infinite because a-b is:"<<i;
  }

  getch();
}
```

Output:

```
Enter the value for a: 20
Enter the value for b: 20
Enter the value for c: 40

Answer is infinite because a-b is: 0
```

Experiment 20

AIM: WAP to demonstrate the use of function template.

```
#include<iostream.h>
#include<conio.h>
 template<class t>

void swap(t &x,t &y)
{
   t temp=x;
   x=y;
   y=temp;
}
 void fun(int a,int b,float c,float d)
{
   cout<<"\na and b before swaping :"<<a<<"\t"<<b;
   swap(a,b);
   cout<<"\na and b after swaping  :"<<a<<"\t"<<b;
   cout<<"\n\nc and d before swaping :"<<c<<"\t"<<d;
   swap(c,d);
   cout<<"\nc and d after swaping  :"<<c<<"\t"<<d;
}
void main()
{
    int a,b;
    float c,d;
    clrscr();
    cout<<"Enter A,B values(integer):";
    cin>>a>>b;
    cout<<"Enter C,D values(float):";
    cin>>c>>d;
    fun(a,b,c,d);
    getch();
}
```

Output:

```
Enter A, B values (integer): 10  20
Enter C, D values (float):    2.50  10.80

A and B before swapping: 10 20
A and B after swapping:  20 10

C and D before swapping: 2.50  10.80
C and D after  swapping: 10.80  2.50
```

Experiment 21

AIM: WAP to demonstrate the use of class template.

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

template<class t>
class cuboid
{
private:
    t a;t b;t c;
public:
    cuboid(t x,t y,t z)
    {
    a=x;
    b=y;
    c=z;
    }
    void volume()
    {
    t v;
    v=a*b*c;
    cout<<"Volume is : "<<v<<endl;
    }
};

void main()
{
clrscr();
cuboid <int> c1(2,3,4);
c1.volume();
cuboid<float>c2(5.6,1.4,3.2);
c2.volume();
getch();
}
```

OUTPUT:

```
Volume is : 24
Volume is : 25.087999
```

Experiment 22

AIM: WAP to copy the contents of a file to another file.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#include<ctype.h>
#include<fstream.h>

void main( )
{
        ofstream outfile;
        ifstream  infile;
        char fname1[10],fname2[20];
        char ch,uch;
        clrscr( );
        cout<<"Enter a file name to be copied ";
        cin>> fname1;
        cout<<"Enter new file name";
        cin>>fname2;
        infile.open(fname1);

if( infile.fail( ) )
        {
                cerr<< " No such a file Exit";
                getch();
                exit(1);
        }
        outfile.open( fname2);
                if(outfile.fail( ))
                {
                        cerr<<"Unable to create a file";
                        getch();
                        exit(1);
                }
        while( !infile.eof( ) )
        {
                ch = (char) infile.get( );
                uch = toupper(ch);
                outfile.put(uch);
        }
```

```
        infile.close( );
       outfile.close( );
       getch( );
}
```

OUTPUT:

Enter a file name to be copied.
C:\text1.txt
Enter new file name
D:\new.txt

Input file
Asbcdefghijklmnopqrstuvwxyz
Output file
ASBCDEFGHIJKLMNOPQRSTUVWXYZ

Experiment 23

AIM:WAP to demonstrate the read and write of file operation.

```
#include<fstream.h>
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream.h>
#include<conio.h>
void main()
{
        char c,u;
        char fname[10];
        clrscr();
        ofstream out;
        cout<<"Enter File Name:";
        cin>>fname;
        out.open(fname);
        cout<<"Enter the text(Enter # at end)\n";   //write contents to file
        while((c=getchar())!='#')
        {
                u=c-32;
                out<<u;
        }
        out.close();
        ifstream in(fname);        //read the contents of file
        cout<<"\n\n\t\tThe File contains\n\n";
        while(in.eof()==0)
        {
                in.get(c);
                cout<<c;
        }
        getch();

}
```

Output:

```
Enter File Name: two.txt
Enter contents to store in file (enter # at end)
oops programming
The File Contains
OOPS PROGRAMMING
```

Experiment 24

AIM: WAP to demonstrate the reading and writing of objects.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
class student
{
int roll_no;
char name[20];
public:
void read()
{
cin>>roll_no>>name;
}
void show()
{
cout<<roll_no<<name;
}
};
void main()
{
clrscr();
student s1;
s1.read();
ofstream out("stu.txt",ios::binary);
out.write((char*)&s1,sizeof(s1));
ifstream in("stu.txt",ios::binary);
in.read((char*)&s1,sizeof(s1));
s1.show();
getch();
}
```

Output:

```
1140831
radha
1140831 radha
```

Experiment 25

AIM: Write a program Concept of Abstraction, Data Hiding, and Encapsulation.

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

class quadratic
{
    private:
    double a,b,c;
    public:
            void read();
            void funcroots();
};
          void quadratic::read()
      {
          cout<<"Enter coffiecients of Quadratic Eq. a,b,c";
          cin>>a>>b>>c;
      }
         void quadratic::funcroots()
     {
        if(a==0)
         {
             cout<<"\nEquations is not quadratic";
             exit(0);
        }
       else
       {
            double disc;
            disc=b*b-4 *a*c;
            if(disc<0)
              cout<<"\nRoots are Imaginary";
            else if(disc==0)
                 {
                      double x=-b/(2*a);
                      cout<<"\nRoots are Equal,x= u"<<x;
                 }
            else
                {
                      double m,n,t;
                      t=sqrt(disc);
                      m=( -b+t)/(2*a);
                      n=(-b-t)/(2*a);
                      cout<<"\nRoots are Real and Unequal"<<endl;
```

```
                cout<<"m="<<m<<endl;
                cout<<"n="<<n;
            }
        }
}


void main()
    {
        clrscr();
        quadratic q1 ;
        q1.read();
        q1.funcroots();
        getch();
    }
```

Output:

```
Enter Coefficients of quadratic eq a,b,c10
20
65
Roots are imaginary
```