Lab Sheet 8
Understanding the Concept of Console and File Input/Output

Console Input/Output

Within console we can perform unformatted and formatted input/output. For unformatted input/output stream functions like put(), get(), getline(), write(), read() etc are used. For formatted input/output the stream objects cin and cout are used along with ios functions and flags and manipulators.
The ios functions that can be used for formatting are

- **width()**
- **fill()**
- **precision()**
- **setf()**
- **unsetf()**
- **flags() etc**

However to use the setf(), unsetf() and flags() functions one should know the flags available in ios class.
File Handling

There are three classes for handling files.

- **ifstream - for handling input files ofstream - for handling output files fstream - for handling input as well as output files.**

In all three classes, passing a filename as the first parameter in the constructor itself can open a file.
e.g ifstream infile("test.txt") opens the file test.txt in the input mode.
The constructors for all these classes are defined in the header file <fstream>, which are as follows
ifstream(const char *path, int mode=ios::in)

ofstream(const char *path, int mode=ios::out)

fstream(const char *path, int mode=ios::in|ios::out)

where path specifies the file to be opened, mode specifies the mode in which the file is to be opened.

File opening can also be done explicitly by calling the member function open() of the file stream classes. The open() function has similar prototype as the constructors.

After opening, the file contents can be written or read by using the stream operators with the file objects as

```
ofstream ofile("test.txt");

ofile<<"C++ lab class";
```

This statement writes "C++ lab class" in the file "test.txt"

Reading and Writing A class Object

The Binary input and output functions read( ) and write are designed to handle the entire structure of an object as a single unit, using the computer's internal representation of data. The function write copies a class object from memory byte by byte with no conversion.

Binary output and input functions take the following form

```
ipfile.read(reinterpret_cast<char*>(&obj),sizeof(obj) );

opfile.write(reinterpret_cast<char*>(&obj),sizeof(obj ));
```

Example

```
#include<i          int main()
ostream>            {
#include<f
stream>
#include<iomani
p>                        int b;
using namespace          public:
std;
                          demofile(){}
class demofile
                          demofile(int x,int
{                        y){a=x;b=y;}
  private:
   int a;
```

```
demofile                    file.write(reinterpret_cast<char*
de(10,20);                  >(&de),sizeof(de));
clrscr();                   file.seekg(0);
fstream file;               file.read(reinterpret_cast<char*>
file.open("demo.            (&de),sizeof(de));
txt",ios::in|               de.display();
ios::out);
                            file.close();
                            return 0;

   void display()   }
   { cout<<"a=
"<<a<<endl<<"b
=
"<<b<<endl;}
};
```

Exercises

1. Write a program to demonstrate the use of different ios flags and functions to format the output. Create a program to generate the bill invoice of a department store by using different formatting.

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
int main()
{
  ofstream bill("bill.txt",ios::out);
  bill << setw(40) << "SUJIT COMPANY LIMITED"
<< endl;
  cout << setw(40) << "SUJIT COMPANY LIMITED"
<< endl;
  int sno = 1;
  char part[20];
  int qt;
  float price;
  float subtotal;
  float total = 0;
```

```
  char ans;
  bill << setw(4) << "Sno " << setw(20) << left <<
"Particulars "<< setw(10) << "Quantity " << setw(6)
<< "Price "<< setw(10) <<"Sub total " <<
  endl; do{
    cout << "Particulars:"
    ; cin >> part;
    cout << "Quantity:";
    cin >> qt;
    cout << "Price:";
    cin >> price;
    cout << "Sub Total:";
    subtotal = qt * price;
    cout << subtotal << endl;
    total += subtotal;
    bill << setw(4) << sno++ << setw(20) << left
<< part << setw(10) << qt << setw(6) << price <<
setw(10) << subtotal << endl;
    cout << "Do you want to continue
    (y/n)"; cin >> ans;
  }
  while(ans == 'y');
  cout << "total " << total;
  bill << setw(34) << "total" << setw(10) << total
<< endl;
  bill.close();
  return 0;
}

#include<iostream>//or
#include<conio.h>
#include<iomanip>
using namespace std;
class Bill
{
    string *item_name;
    int *item_quantity,n;
    float *item_rate,*total_price;
public:
    void getdata()
    {
        cout<<"Enter the no of items:";
        cin>>n;
        item_name = new string[n];
        item_quantity = new int[n];
        item_rate=new float[n];
        total_price= new float[n];
```

```
        for(int i=0;i<n;i++)
        {
            cin.ignore();
            cout<<endl<<"Enter item name:\t";
            getline(cin,item_name[i]);
            cout<<endl<<"Enter item quantity:\t";
            cin>>item_quantity[i];
            cout<<endl<<"Enter item rate:\t";
            cin>>item_rate[i];

total_price[i]=item_quantity[i]*item_rate[i];
        }
    }
    void display()
    {
        int total=0;
        system("cls");

cout<<right<<setfill('*')<<setw(62)<<"WELCOME"<<setw(
55)<<' '<<endl;
        cout<<left<<setfill('
')<<setw(15)<<"S.N"<<setw(25)<<"Item
name"<<setw(25)<<"Item quantity"<<setw(25)<<"Item
rate"<<setw(25)<<"Total price"<<endl;
      for(int i=0;i<n;i++)
        {

cout<<left<<setw(15)<<i+1<<setw(25)<<item_name[i]<<se
tw(25)<<item_quantity[i]<<setw(25)<<item_rate[i]<<set
w(25)<<total_price[i]<<endl;
            total+=total_price[i];
        }
         cout<<right<<setw(96)<<"----------"<<endl;
         cout<<right<<setw(91  )<<total;
    }
};
int main()
{
    Bill b;
    b.getdata();
    b.display();
}
```

2.Write a program to create a userdefined manipulator that will format the output by setting the width, precision and fill character at the same time by passing arguments.

```
#include <iostream>
#include <iomanip>
using namespace std;
class Testmani{
 private:
  int width, precision;
  char fill;
 public:
  Testmani(int w, int p , char
f):width(w),precision(p),fill(f){};
  friend ostream& operator<<(ostream &str, Testmani
obj);
};
ostream& operator<<(ostream &str, Testmani obj)
{
  str << setw(obj.width)<< setfill(obj.fill) <<
setprecision(obj.precision);
  return str;
}
Testmani setwpf(int w, int p, char f)
{
  return Testmani(w,p,f);
}
int main()
{
  cout << setwpf(5,3,'$') <<
  5.630009; return 0;
}
```

```
#include<iostream>\\or
#include<iomanip>
using namespace std;
class manip
{
    int Width,Precision;
    char fill_char;
public:
```

```
    manip(int width,int precision,char
fillchar):Width(width),Precision(precision),fill_char
(fillchar){};
    friend ostream& operator <<(ostream& output,
manip obj);
};
  ostream & operator<<(ostream& output,manip obj)
    {
        output<<setw(obj.Width);
        output<<setprecision(obj.Precision);
        output<<setfill(obj.fill_char);
        return output;
    }
    manip setvalue(int width,int precision,char
fillchar)
    {
        return manip(width,precision,fillchar);
    }
int main()
{
 cout << setvalue(10,3,'*')<<28.66565544;
}
```

3. Write a program to overload stream operators to read complex number and display the complex number in a+ib format.

```
#include <iostream>
using namespace std;
class Complex{
 private:
  int real, img;
 public:
  Complex(int r, int i):real(r),img(i){};
  friend ostream& operator<<(ostream& a, Complex c);

};
ostream& operator<<(ostream& a, Complex c)
{
    a << c.real << "+i" << c.img;
}
int main()
{
Complex w(1,3);
  cout << w;
```

```
  return 0;
}

#include<iostream>//or
using namespace std;
class complex
{
   int real,imag;
   public:
       complex(){};
       friend istream& operator >>
(istream&,complex&);
       friend ostream& operator
<<(ostream&,complex&);
};
istream& operator>>(istream& in,complex& obj)
{
    cout<<"Enter the complex nunmber:"<<endl;
    cout<<"Real part:";
    in>>obj.real;
    cout<<"Imaginary part:";
    in>>obj.imag;
    return in;
}
ostream& operator<< (ostream& out,complex& obj)
{
    out<<"Entered complex number:"<<obj.real<<" +
"<<obj.imag<<" i";
    return out;
}
int main()
{
    complex num;
    cin>>num;
    cout<<num;
}
```

4. Write a program that stores the information about students (name, student id,department and address) in a structure and then transfers the information to a file in your directory. Finally, retrieve the information from your file and print in the proper format on your output screen.

```
#include <iostream>
#include <cstring>
#include <fstream>
using namespace
```

```
std; class Student{
private:
    char name[20], id[10], department[20],
address[20];
 public:
  Student(){};
  Student(char n[], char i[], char d[], char a[])
  {
    strncpy(name,n,20);
    strncpy(id,i,20);
    strncpy(department,d,20);
    strncpy(address,a,20);
  }
  void display(){
    cout << "Name: " << name << endl;
    cout << "Id: " << id << endl ;
    cout << "Department: " << department << endl;
    cout << "Address: " << address << endl;
  }
};

int main()
{
  ofstream data;
  data.open("Student.dat",ios::out|ios::binary);

  char name[20], id[10], department[20],
  address[20]; char ans;
  do
  {
    cout << "Enter Student information" << endl;
    cout << "Name: ";
    cin >> name;
    cout << "Id: " ;
    cin >> id;
    cout << "Department: " ;
    cin >> department;
    cout << "Address: " ;
    cin >> address;

    Student newStudent(name,id,department,address);

    data.write(reinterpret_cast<char
*>(&newStudent),sizeof(newStudent));
    cout << "Do you want to continue adding student
data y/n";
    cin >> ans;
```

```
  }
  while(ans == 'y');
  data.close();

  ifstream info;
  info.open("Student.dat",ios::in|ios::binary);
  while(!info.eof())
  {
    Student newStudent;
    info.read(reinterpret_cast<char
*>(&newStudent),sizeof(newStudent));
    if (info)
      newStudent.display();
  }
  info.close();

  return 0;
}

#include<iostream>//or
#include<fstream>
#include<iomanip>
using namespace std;
class student
{
    char name[20],department[20],address[20];
    int id;
public:
    void getdata()
    {
        cout<<endl<<"Name:\t";
        cin.getline(name,20);
        cout<<endl<<"Address:\t";
        cin.getline(address,20);
        cout<<endl<<"Department:\t";
        cin.getline(department,20);
        cout<<endl<<"Id:\t";
        cin>>id;
        cin.ignore();
    }
    void displaydata()
    {

cout<<setiosflags(ios::left)<<setw(20)<<name<<setw(20
)<<address<<setw(20)<<department<<setw(20)<<id<<endl;

    }
};
```

```
int main()
{
    ofstream file_write;
    student st;
    student disp;
    char c;

file_write.open("student.txt",ios::out|ios::binary);
    cout<<"Enter students data:"<<endl;
    do
    {
      st.getdata();

file_write.write(reinterpret_cast<char*>(&st),sizeof
(st));
        cout<<"Do you want to add more(y/n)?";
        cin>>c;
        cin.ignore();
    }while(c!='n');
    file_write.close();
    ifstream file_read;
   file_read.open("student.txt",ios::in|ios::binary);

cout<<endl<<setiosflags(ios::left)<<setfill('*')<<set
w(62)<<'*'<<endl;
    cout<<setfill('
')<<setw(20)<<"Name"<<setw(20)<<"Address"<<setw(20)<<
"Department"<<setw(20)<<"Id"<<endl;

cout<<setiosflags(ios::left)<<setfill('*')<<setw(62)<
<'*'<<setfill(' ')<<endl;

while(file_read.read(reinterpret_cast<char*>(&disp),s
izeof(disp)))
    {
            disp.displaydata();
    }
    file_read.close();
}
```

5. Write a program for transaction processing that write and read object randomly to and from a random access file so that user can add, update, delete and display the account information (accountnumber, lastname, firstname, totalbalance).

```
#include <iostream>
```

```
#include <fstream>
#include <cstring>
using namespace std;
class Transaction{
 private:
  char firstName[20], lastName[20];
 int  accountNumber,  totalBalance;
 public:
  Transaction(){};
  Transaction(char fn[], char ln[], int an, int
tb):accountNumber(an),totalBalance(tb){
    strncpy(firstName,fn,20);
    strncpy(lastName,ln,20);
  }
  friend istream& operator>>(istream& in, Transaction
&tr);
  friend ostream& operator<<(ostream& out,
Transaction tr);

};
istream& operator>>(istream& in, Transaction &tr)
{
  cout  <<  "-----------------------------------------
" << endl;
  cout << "First Name:
  "; in >> tr.firstName;
  cout << "Last Name: ";
  in >> tr.lastName;
  cout << "Account Number: ";
  in >> tr.accountNumber;
  cout << "Total Balance: ";
  in >> tr.totalBalance;
  cout  <<  "-----------------------------------------
" << endl;
  return in;
}

ostream &operator<<(ostream &out, Transaction tr)
{
```

```
  cout  <<  "----------------------------------------
" << endl;
  out << "First Name: " << tr.firstName << endl;
  out << "Last Name: " << tr.lastName << endl;
  out << "Account Number: " << tr.accountNumber <<
endl;
  out << "Total Balance: " << tr.totalBalance <<
endl;
  cout  <<  "----------------------------------------
" << endl;
  return out;
}
int main()
{
  int ans;
  do
  {
    cout << "Menu" << endl;
    cout << "1.create record" << endl;
    cout << "2.add record" << endl;
    cout << "3.delete record" << endl;
    cout << "4.edit record" << endl;
    cout << "5.display record" <<
    endl; cout << "Enter your choice";

    cin >> ans;
    fstream acc;
    if(ans== 1)
    {
      char ans;
      acc.open("account.dat",ios::out|ios::binary);
    do
    {
      Transaction tr;
      cin >> tr;
      acc.write(reinterpret_cast<char
*>(&tr),sizeof(tr));
      if (!acc)
      {
```

```
  cerr                                          << "Couldnot
                                                  write   the
                                                   data    to
                                                          the
file";
        return 1;
      }
      cout << "Do you want to continue y/n ";
      cin >> ans;
    }
    while(ans=='y');
    acc.close();
  }
  else if(ans == 2)
  {
    char ans;

acc.open("account.dat",ios::out|ios::app|ios::binary)
;
    do
    {
      Transaction tr;
      cin >> tr;
      acc.write(reinterpret_cast<char
*>(&tr),sizeof(tr));
      if (!acc)
      {
        cerr  <<  "Couldnot  modify  the  data  of  the
file";
        return 2;
      }
      cout << "Do you want to continue y/n ";
      cin >> ans;
    }
    while(ans=='y');
    acc.close();
  }
  else if(ans == 3)
  {
```

```
    char ns;
    Transaction tr;
    acc.open("account.dat",ios::in|ios::binary);
    fstream newacc;
    newacc.open("tmp.dat",ios::out|ios::binary);
    if(!newacc)
    {
      cerr << "couldnot create tmp record file";
    }
    while(!acc.eof())
    {
      acc.read(reinterpret_cast<char
*>(&tr),sizeof(tr));
      if (acc)
      {
        cout << tr;
        cout << "Do you want to delete this record
y/n ";
        cin >> ns;
        if (ns != 'y')
        {
          newacc.write(reinterpret_cast<char
*>(&tr),sizeof(tr));
          if(!newacc)
          {
            cerr << "unable to write to a temp
            file"; return 3;
          }
        }
        else
        {
          cout << "record deleted" << endl;
        }
      }
    }

    acc.close();
    newacc.close();
    remove("account.dat");
```

```
    rename("tmp.dat","account.dat");
  }
  else if(ans == 4)
  {
    char ans;

acc.open("account.dat",ios::in|ios::binary|ios::out);
    while(!acc.eof())
    {
      Transaction tr;
      int pos = acc.tellg();
      acc.read(reinterpret_cast<cha
      r
*>(&tr),sizeof(tr));
      if (acc)
      {
        cout << tr;
        cout << "Do you want to edit this record
y/n";
        cin >> ans;
        if (ans == 'y')
        {
          cin >> tr;
          acc.seekp(pos);
          acc.write(reinterpret_cast<char
*>(&tr),sizeof(tr));
          if(acc)
          {
            cout << "Record sucessfully edited"<<
endl;
          }
          else
          {
            cerr<< "Unable to modify the record";
            return 4;
          }
        }
      }
    }
```

```
    acc.close();
  }
  else if (ans == 5)
  {
    acc.open("account.dat",ios::in|ios::binary);
    while(!acc.eof())
    {
      Transaction tr;
      acc.read(reinterpret_cast<cha
      r
*>(&tr),sizeof(tr));
      if(acc)
      {
        cout << tr;
      }
      else
      {
    cerr  <<  "Couldnot  read  through  the  file"  <<
endl;
      }
    }
    acc.close();
  }
  }
  while(ans <= 5 && ans > 0
  ); return 0;
}

#include<iostream>//or
#include<fstream>
#include<iomanip>
using namespace std;
class transaction
{
    long int account_number;
    char lastname[20],firstname[20];
    float total_balance;
public:
    void getdata()
    {
```

```
        cout<<endl<<"Enter the following
information:"<<endl;
        cout<<"Account number:\t";
        cin>>account_number;
        cin.ignore();
        cout<<endl<<"First name:\t";
        cin.getline(firstname,20);
        cout<<endl<<"Last name:\t";
        cin.getline(lastname,20);
        cout<<endl<<"Total balance:\t";
        cin>>total_balance;
    }
    void display()
    {

cout<<setiosflags(ios::left)<<setw(25)<<account_numbe
r<<setw(20)<<firstname<<setw(20)<<lastname<<setw(20)<
<total_balance<<endl;
    }
};
int main()
{
    transaction tr;
    fstream file;
    int choice;
    while(1){
    cout<<"\n\nEnter your choice:"<<endl;
    cout<<"1) Create record\n";
    cout<<"2) Add record\n";
    cout<<"3) Update record\n";
    cout<<"4) Delete record\n";
    cout<<"5) Display record\n";
    cout<<"6) Exit\n";
    cout<<"Enter your choice:";
    cin>>choice;
    int record_pos,location;
    char ch;
    switch(choice)
    {
    case 1:{

file.open("transaction.txt",ios::out|ios::binary);
```

```
        do
        {
           tr.getdata();

file.write(reinterpret_cast<char*>(&tr),sizeof(tr));
            cout<<"Do you want to continue (y/n)?";
            cin>>ch;
        }while(ch!='n');
        file.close();
        break;
    }
    case 2:{

file.open("transaction.txt",ios::out|ios::app|ios::bi
nary);
        do
        {
            tr.getdata();

file.write(reinterpret_cast<char*>(&tr),sizeof(tr));
            cout<<"Do you want to continue (y/n)?";
            cin>>ch;
        }while(ch!='n');
        file.close();
        break;
    }
    case 3:{

file.open("transaction.txt",ios::out|ios::in|ios::bin
ary);
        cout<<"Enter record number to be updated:";
        cin>>record_pos;
        location=(record_pos-1)*sizeof(tr);
        tr.getdata();
        file.seekp(location);

file.write(reinterpret_cast<char*>(&tr),sizeof(tr));
        file.close();
        break;
    }
    case 4:{

file.open("transaction.txt",ios::in|ios::binary);
```

```
        ofstream
file_new("temp.txt",ios::out|ios::binary);
        cout<<"Enter record number to be deleted:";
        cin>>record_pos;
        location=(record_pos)*sizeof(tr);

while(file.read(reinterpret_cast<char*>(&tr),sizeof(t
r)))
        {
            if(file.tellg()!=location)
            {

file_new.write(reinterpret_cast<char*>(&tr),sizeof(tr
));
            }
            else
            {
                continue;
            }
        }
        file.close();
        file_new.close();
        remove("transaction.txt");
        rename("temp.txt","transaction.txt");
        break;
    }
    case 5:{

file.open("transaction.txt",ios::in|ios::binary);

cout<<endl<<setiosflags(ios::left)<<setfill('*')<<set
w(78)<<'*'<<endl;
        cout<<setfill(' ')<<setw(25)<<"Account
number"<<setw(20)<<"First name"<<setw(20)<<"Last
name"<<setw(20)<<"Total balance"<<endl;

cout<<setiosflags(ios::left)<<setfill('*')<<setw(78)<
<'*'<<setfill(' ')<<endl;

while(file.read(reinterpret_cast<char*>(&tr),sizeof(t
r)))
        {
            tr.display();
        }
```

```
        file.close();
         break;
    default:
         exit(0);
    }
    }
    }
}
```