UNIT 8
FILE HANDLING AND STREAMS

LH – 6HRS

PRESENTED BY:

ER. SHARAT MAHARJAN

OOP IN C++

CONTENTS (LH – 6HRS)

8.1 Stream Class Hierarchy for Console Input/Output (fstream base, ifstream, ofstream and fstream) (binary vs character files),

8.2 Unformatted Input/Output,

8.3 Formatted Input/Output with ios Member functions,

8.4 Formatting with Manipulators,

8.5 File Input/Output with Streams,

8.6 Opening and Closing files (open() and close() member function),

8.7 Read/Write from File (put() and get(), read() and write() member functions),

8.8 File Access Pointers and their Manipulators (seekg(), seekp(), tellg(), tellp(), offset, ios::beg, ios::cur, ios::end)

8.9 Sequential, Random Access to File

8.10 Testing Errors during File Operations (eof(), fail(), bad(), good())

8.11 Stream Operator Overloading (overloading extraction and insertion operators)

8.1 Stream Class Hierarchy

- So far, we have been using the **iostream** standard library, which provides **cin** and **cout** methods for reading from standard input and writing to standard output respectively.
- To read from a file and write into a file, requires another standard C++ library called **fstream**.

1. **ofstream**

- This data type represents the **output file stream** and is used to create files and to write information to files.

2. **ifstream**

- This data type represents the **input file stream** and is used to read information from files.

3. **fstream**

- This data type represents the file stream generally, and has the **capabilities of both ofstream and ifstream** which means it can create files, write information to files, and read information from files.

✓To perform file processing in C++, header files **<iostream> and <fstream>** must be included in C++ source file.

8.2 Unformatted Input/Output

- It is most basic form of input/output.
- It transfers internal representation of data directly between memory and file.

Examples:

✓get, put and getline methods

✓read and write methods

LAB 1: WAP to read the contents of file D:\\abc.txt.

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
        char ch;
        ifstream fin;
        fin.open("D:\\abc.txt");
        while(!fin.eof()){
                fin.get(ch);
                cout.put(ch);
        }
        fin.close();
        return 0;
}
```

LAB 2: WAP to write into and read from file with content name and age.

```
#include <fstream>
#include <iostream>
using namespace std;
int main () {
  char name[100];
  int age;
  ofstream outfile;             // open a file in write mode.
  outfile.open("afile.txt");
  cout << "Enter your name: ";
  cin.getline(name, 100);
  outfile << name << endl;  // write inputted data into the file.
  cout << "Enter your age: ";
  cin >> age;
  outfile << age << endl;     // again write inputted data into the file.
  outfile.close();              // close the opened file.
  ifstream infile;              // open a file in read mode.
  infile.open("afile.txt");
  cout << "Reading from the file" << endl;
  infile.getline(name,100);
  cout << name << endl;      // write the data at the screen.
  infile >> age;                // again read the data from the file and display it.
  cout << age << endl;
  infile.close();               // close the opened file.
  return 0;}
```

Reading and Writing by using read() and write() member functions:

Syntax for write():

✓fileobject.write((char *)&object, sizeof(object));

Syntax for read():

✓fileobject.read((char *)&object, sizeof(object));

LAB 3: WAP to write multiple objects to a file.

```
#include<iostream>
#include<fstream>
using namespace std;
class student{
        int roll;
        char name[20];
        char address[50];
        public:
                void setData(){
                        cout<<"Enter roll:"<<endl;
                        cin>>roll;
                        cout<<"Enter name:"<<endl;
                        cin>>name;
                        cout<<"Enter address:"<<endl;
                        cin>>address;}
                void getData(){
                        cout<<"Roll:"<<roll<<endl<<"Name:"<<name<<endl<<"Address:"<<address<<endl;
                }};
int main(){
        student s;
        ofstream fout;
        fout.open("D:\\student.txt");
        for(int i=0;i<5;i++){
                s.setData();
                fout.write((char*)&s,sizeof(student));}
        fout.close();
        cout<<"Write completed."<<endl;
        return 0;}
```

Lab 4: WAP to read multiple objects from file.

```
#include<iostream>
#include<fstream>
using namespace std;
class student{
        int roll;
        char name[20];
        char address[50];
        public:
                void setData(){
                        cout<<"Enter roll:"<<endl;
                        cin>>roll;
                        cout<<"Enter name:"<<endl;
                        cin>>name;
                        cout<<"Enter address:"<<endl;
                        cin>>address;
                }
                void getData(){
                        cout<<"Roll:"<<roll<<endl<<"Name:"<<name<<endl<<"Address:"<<address<<endl;
                }};
int main(){
        student s;
        ifstream fin;
        fin.open("D:\\student.txt");
        for(int i=0;i<5;i++){
                fin.read((char *)&s,sizeof(student)); //fin.seekg( sizeof(s)*2, ios::cur );-read only 3rd object from 5 objects
                s.getData();}
        fin.close();
        cout<<"Read completed."<<endl;
        return 0;}
```

8.3 Formatted Input/Output

- Formatted output converts internal binary representation of data into ASCII format and writes data to the file and formatted input reads data from file and converts it into internal binary format.
- Formatting of data can be done in two ways:

✓By using overloaded stream operators(<<-insertion >>-extraction)

✓By using manipulators(setw(),etc.)

LAB 5: Program to write content to the file.

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
        ofstream fout;
        fout.open("D:\\hello.txt");
        fout<<"Hello world.";
        fout<<"\nHello again.";
        fout.close();
        return 0;
}
```

8.8 File Access Pointers

- Both istream and ostream provide member functions for repositioning the file-position pointer. These member functions are **seekg** ("**seek get**") for istream and **seekp** ("**seek put**") for ostream.
- The argument to seekg and seekp normally is a long integer. A second argument can be specified to indicate the seek direction. The seek direction can be **ios::beg (the default)** for positioning relative to the beginning of a stream, **ios::cur** for positioning relative to the current position in a stream or **ios::end** for positioning relative to the end of a stream.

// position to the nth byte of fileObject (assumes ios::beg)

- fileObject.seekg(n);

// position n bytes forward in fileObject

- fileObject.seekg(n, ios::cur);

// position n bytes back from end of fileObject

- fileObject.seekg(n, ios::end);

// position at end of fileObject

- fileObject.seekg(0, ios::end);

8.10 Testing Errors during File Operations

- eof(): returns non-zero (true value) if end-of-file is encountered while reading, otherwise returns zero (false value).
- fail(): returns non-zero when an input or output operation has failed.
- good(): returns non-zero if no error has occurred.
- bad(): returns non-zero if an invalid operation is attempted.

THANK YOU FOR YOUR ATTENTION