Object-oriented Programming
File management
Version 1.0

In this particular lab session, the very basics and fundamentals of File management will be revealed towards you. With this session, you will be able to know various things about File management.

File management:

A file is a collection of related data stored in a particular area on the disk. Many real life problems handle large volumes of data and, in such situations, we need to use the concept of files.

The I/O system of C++ contains a set of classes that define the file handling methods. These include **ifstream**, **ofstream**, and **fstream**. These classes are contained in the header file *fstream*.

Opening and Closing a file:

A file can be opened in two ways:

1. Using constructor function of the class.
2. Using member function open() of the class.

The first method is useful when we use only one file in the stream whereas the second method is useful to manage multiple files.

In closing a file, we can simply use member function close() of the class.

Opening files using constructor:

We know that a constructor is used to initialize an object while it is being created. Here a file name is used to initialize the file stream objects. Let us see a program of creating files with constructor functions:

```
#include<iostream>
#include<fstream>
using namespace std;

class X{
    int cost;
    char name[20];
    public:
    void file_write()
    {
      ofstream outf("item.txt");
      cout<<"item name:";
      cin>>name;
```

```
      outf<<name<<"\n";

      cout<<"item cost:";
      int cost;
      cin>>cost;
      outf<<cost<<"\n";
      outf.close();
    }
    void file_read(){
      ifstream inf("item.txt");
      inf>>name;
      inf>>cost;
      cout<<"\n";
      cout<<"item:"<<name<<"\n";
      cout<<"expenditure:"<<cost<<"\n";
      inf.close();
    }
};
int main()
{
      X ob1;
      ob1.file_write();
      ob1.file_read();
      return 0;
}
```

In the above program, the file stream objects outf and inf are created for writing to and reading from file item.txt respectively. The item.txt file from outf is disconnected by using the following statement:

outf.close();

And the item.txt file from inf is disconnected by using the following statement:

inf.close();

Opening files using open():

As stated earlier, the function open() can be used to open multiples files that use the same stream object. This is done as follows:

file-stream-class stream-object;

stream-object.open("filename");

Example:

ofstream outfile;

outfile.open(“Data1.txt”);

……..

……..

outfile.close();

outfile.open(“Data2.txt”);

……..

……..

outfile.close();

……..

……..

Let us see a program of creating files with open():

```
#include<iostream>
#include<fstream>
using namespace std;

class X{
    public:
    void file_write()
    {
    ofstream fout;
    fout.open("country.txt");
    fout<<"USA\t";
    fout<<"UK";
    fout.close();
    fout.open("capital.txt");
    fout<<"Washington\t";
    fout<<"London\n";
    fout.close();
    }
    void file_read(){
    char line[50];
    ifstream fin;
    fin.open("country.txt");
    cout<<"The contents of the country file:\n";
    while(fin)
    {
        fin.getline(line,50);
```

```
        cout<<line;

    }
    fin.close();
    fin.open("capital.txt");
    cout<<"\nThe contents of the capital file:\n";
    while(fin)
    {
        fin.getline(line,50);
        cout<<line;

    }
    fin.close();
    }
};
int main()
{
      X ob1;
      ob1.file_write();
      ob1.file_read();
      return 0;
}
```

Detecting end-of-file:

Detection of the end-of-file condition is necessary for preventing any further attempt to read data from the file.

Suppose, the following statement

```
while(fin)
{
}
```

An **ifstream** object, such as **fin**, returns a value of 0 if any error occurs in the file otherwise non-zero value.

There is another approach using **eof()** member function of **ios** class. It returns a non-zero value when end of file is encountered, and zero, otherwise. This can be done using the following statement:

```
if( ! fin.eof() )
{
……..
}
```

File modes using open():

In order to open a file with a stream object we use its member function `open` :

```
open (filename, mode);
```

Where `filename` is a string representing the name of the file to be opened, and `mode` is an optional parameter with a combination of the following flags:

`ios::in`	Open for input operations.
`ios::out`	Open for output operations.
`ios::binary`	Open in binary mode.
`ios::ate`	Set the initial position at the end of the file. If this flag is not set, the initial position is the beginning of the file.
`ios::app`	All output operations are performed at the end of the file, appending the content to the current content of the file.
`ios::trunc`	If the file is opened for output operations and it already existed, its previous content is deleted and replaced by the new one.

Example:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  string line;
  ofstream myfile ("example.txt", ios::app);

  if (myfile.is_open())
  {
      cin>>line;
      myfile<<line;
      //value of line variable is appended to the file

       myfile.close();
  }

  else cout << "Unable to open file";

  return 0;
}
```

Another Example:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  string line;
  ifstream myfile ("example.txt", ios::in);

  if (myfile.is_open())
  {
    while ( getline (myfile,line) )
    {
      cout << line << '\n';
    }
    myfile.close();
  }
  else cout << "Unable to open file";

  return 0;
}
```