FILE ORGANISATION

Topic 4-term 2

Introduction to files and streams

Introduction

Many programs use large data sets that are stored in secondary memory
The data is stored in files.

There are two basic types of files:

- Text files: The data in the file are characters
- Binary files: The data in the file are in binary form

Files can also be grouped into:

- Sequential files: The data is accessed in the order in which it is stored
- Random files: The data is accessed randomly

Stream

A stream is a sequence of characters. It is the one that links a program to a file. In object-oriented programs, stream are objects. In C++ streams are objects of type fstream.

fstream is the name of a class that is stored in fstream.h file.

An object of type fstream is declared as shown below:
fstream inputfile;
fstream outputfile;
where: fstream is a keyword, *inputfile* and *outputfile* are names of the streams that are to be linked to files.
Connecting a file to a stream is known as opening a file,

1. using a constructor function
2. using open member function

1. Opening a file using constructor function.

The syntax of opening a file using constructor is:
fstream fstreamname("file_name", opening_mode);
where:
fstream is a reserved word
file_name is the name of the file being opened
opening_mode specifies whether a stream is being used for input, output or both
It can be one of the following specifiers:
ios:: in -open for reading

- ios::out -open for writing
- ios::app -append at the end of the file
- ios::ate -go to the end of the file
- ios::binary -open as a binary file

* ios::trunc - in content of a file being opened that existed earlier is discarded

For example:

fstream outputfile(“myfile1.cpp”, ios::out); //output file that is opened for writing

fstream inputfile(“myfile2.cpp”, ios::in); //input file that is opened for reading

Where
myfile1.cpp and myfile2.cpp are the names of files that are being opened.

2. Using open() function

The syntax to open a file using open() function is

fstreamname.open("file_name", open_mode);
For example.

fstream myfile; //a call to a fstream class. fstream is a reserved word.

myfile.open("Data1.cpp", ios::out); //open() is a member function of fstream class. It is being called using myfile reference variable (This is an already defined and initialized variable that has been assigned a value).

Data1.cpp is the argument that is being passed to the open() function or It is the name of the file that is opened.

ios::out is the file open mode.

Checking for successful opening

It is always advisable to check whether a file has been opened successful before carrying out any operation on the file. There are many things that can make opening of a file to fail.

For example, a file could be in a different directory or it could not be existing at all. Any attempt to carry out any operation on a file that has not been opened properly will fail.

fail() function is used to check whether a file has been opened successfully.

fail() function returns true(1) if the opening has failed and false(0) if the opening has succeeded

example

```
fstream inputfile("myfile.cpp", ios::in);

if( inputfile.fail())                    //display message if the opening has failed
cout<<……;
….
}
```

In such a case the program is terminated through the use of exit() function.

This function is passed 1 to indicate that the program has not ended successfully.

```
if( inputfile.fail())                    //display message if the opening has failed

exit(1);
}
```

Closing a file

A file that has been opened should be closed after it has been processed. This is done through the use of close() function.

This function disconnect a stream from a file that it has been connected with.

Example:

```
fstream inputfile(“myfile.cpp”, ios::in);
inputfile.close();
```

Input from and output to a file

Once a file has been connected to an input stream, the input operator >> can be used to input data from the file just like the way this operator is used to input data from a keyboard.

Example:

fstream inputfile(“myfile.cpp”, ios::in);
inputfile>>variable1>>variable2>>...variableN;

Similarly, once a file has been connected to an output stream, the output operator << can be used to output data to the file just like the way this operator is used to output data to a keyboard.

Example:

fstream outputfile(“myfile.cpp”, ios::out);
outputfile<<variable1<<variable2<<...variableN;

Detecting end of a file

It is always advisable to keep on checking whether one has reached the end of a file particularly when the file is an input.

This is done using

eof() function

This function returns true(nonzero) value when it has come to the end of a file and zero otherwise(when hasn't gotten to the end).

C++ features that support object oriented programming

* It gives the easiest way to handle the data handling and encapsulation with the help of powerful keywords such as class, private, public and protected
* C++ has inheritance as one of the most powerful design concepts
* It has polymorphism through virtual functions, virtual base classes and virtual destructors
* C++ provides overloading of operators and functions
* It focusses on functions and class templates for handling parameterized data types
* C++ provides friends, static, methods, constructors and destructors for class objects

Ways of coping with emerging trends in OOP

* By Upgrading the softwares online
* Use the licensed softwares as their updates are easily available
* continuously train for the new modules or packages