BCA SEMESTER-II

Object Oriented Programming using C++

Paper Code: BCA CC203

Unit :4

File Handling in C++

By: Ms. Nimisha Manan

Assistant Professor

Dept. of Computer Science

Patna Women's College

File Handling in C++

At the time of execution, every program is loaded in main memory to execute. Since main memory is volatile , the data would be lost once the program is terminated. If we need the same data again, we have to store the data in a file on the disk.

- A file is sequential stream of bytes ending with an end-of-file marker.
- A File is a collection of data stored on a secondary storage device like hard disk.
- File represents storage medium for storing data or information.

Types of file supported by C++:

- **Text Files :-** Text file is human readable because everything is stored in terms of text.

- **Binary Files :-** In binary file everything is written in terms of 0 and 1, therefore binary file is not human readable.

Streams in C++

A stream refers to a sequence of bytes. A stream is a logical interface to the devices that are connected to the computer. It is widely used as a logical interface to a file such as a disk file, the computer screen, keyboard etc. In Files we store data i.e. text or binary data permanently and use these data to read or write in the form of input output operations by transferring bytes of data. So we use the term File Streams/File handling.

The three standard streams in C++ are console input(cin),console output(cout) and console error(cerr)

Console input(cin) or standard input(stdin) :- Standard input is the stream from which the program receives its data. Unless redirected, input for a program is expected from the keyboard.

Console output(cout) or standard output (stdout) :- Standard Output is the stream where the program writes its output data. The program requests data transfer using thw write operation.

Console error(cerr) or standard error(stderr) :- Standard error is basically an output stream used by programs to report error messages.

A stream is linked to a file using an open operation and disconnected from a file using a close operation.

C++'s standard library called fstream, defines the following classes to support file handling. fstream is used to read and write on files.

- **ofstream class :** Provides methods for writing data into file. Such as, open(), put(), write(), seekp(), tellp(), close(), etc.
- **ifstream class :** Provides methods for reading data from file. Such as, open(),get(), read(), seekg(), tellg(), close(), etc.
- **fstream class :** Provides methods for both writing and reading data from file. The fstream class includes all the methods of ifstream and ofstream class.

Operations in File Handling:

- Opening a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

Opening a file

We need to tell the computer the purpose of opening our file. For e.g.- to write on the file, to read from the file, etc.

The open() function takes file-name argument. The purpose of opening the file i.e, whether for reading or writing, depends on the object associated with open() function.

These are the different modes in which we can open a file.

Mode	Purpose
ios::in	Open a text file for reading.

ios::out	Open a text file for writing. If it doesn't exist, it will be created.
ios::ate	Open a file and move the pointer at the end-of-file.
ios::app	Open a text file for appending. Data will be added at the end of the existing file. If file doesnt exist, it will be created.
ios::binary	Open file in a binary mode.
ios::nocreate	The file must already exist. If file doesn't exist, it will not create new file.
ios::trunc	If the file already exists, all the data will lost.

Program for opening a file.

```
#include <iostream>

#include <fstream>

int main(){

  ofstream file;

  file.open ("example.txt");

  return 0;

}
```

We have opened the file 'example.txt' to write on it. 'example.txt' file must be created in your working directory. We can also open the file for both reading and writing purposes.

Example :-

```
#include <iostream>
```

```
#include <fstream>

int main(){

  fstream file;

  file.open ("example.txt", ios::out | ios::in );

  return 0;

}
```

Reading & Writing data

We can read data from file and write data to file in four ways.

- Reading or writing characters using get() and put() member functions.
- Reading or writing formatted I/O using insertion operator (<<) and extraction operator (>>).
- Reading or writing object using read() and write() member functions.
- Reading and writing on a file

Example :-

```
#include <iostream>
#include <fstream>
int main(){
  char text[200];
  fstream file;
  file.open ("example.txt", ios::out | ios::in );
  cout << "Write text to be written on file." << endl;
  cin.getline(text, sizeof(text));

 // Writing on file
  file << text << endl;

  // Reading from file
  file >> text;
```

```
    cout << text << endl;


   //closing the file
    file.close();
    return 0;
  }
```

Closing a file

C++ automatically close and release all the allocated memory. But a programmer should always close all the opened files. Let's see how to close it.

```
#include <iostream>
#include <fstream>
int main(){
  ofstream file;
  file.open ("example.txt");
  file.close();
  return 0;
}
```