Lab 16
File Handling

What is file handling in C++?

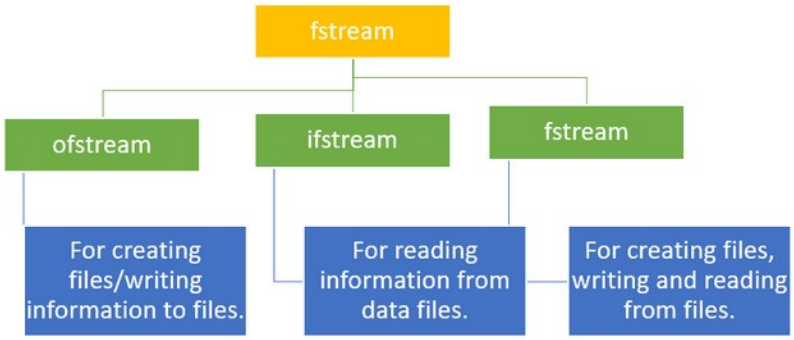
Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file. A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length. This will be determined by their usage. C++ provides you with a library that comes with methods for file handling.

The fstream Library

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:



To use the above classes of the fstream library, you must include it in your program as a header file.

How to Open Files

Before performing any operation on a file, you must first open it. If you need to write to the file, open it using fstream or ofstream objects. If you only need to read from the file, open it using the ifstream object.

The three objects, that is, fstream, ofstream, and ifstream, have the open() function defined in them. The function takes this syntax:

```
open (file_name, mode);
```

- The file_name parameter denotes the name of the file to open.
- The mode parameter is optional. It can take any of the following values:

Value	Description
ios:: app	The Append mode. The output sent to the file is appended to it.
ios::ate	It opens the file for the output then moves the read and write control to file's end.
ios::in	It opens the file for a read.
ios::out	It opens the file for a write.
ios::trunk	If a file exists, the file elements should be truncated prior to its opening.

How to Close Files

Once a C++ program terminates, it automatically

- flushes the streams
- releases the allocated memory
- closes opened files.

However, as a programmer, you should learn to close open files before the program terminates. The fstream, ofstream, and ifstream objects have the close() function for closing files. The function takes this syntax:

```
void close();
```

How to Write to Files

You can write to file right from your C++ program. You use stream insertion operator (<<) for this. The text to be written to the file should be enclosed within double-quotes.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
        fstream my_file;
        my_file.open("my_file.txt", ios::out);
        if (!my_file) {
                cout << "File not created!";
        }
        else {
                cout << "File created successfully!";
                my_file << "Guru99";
                my_file.close();
        }
        return 0;
}
```

How to Read from Files

You can read information from files into your C++ program using stream extraction operator (>>). You use the operator in the same way you use it to read user input from the keyboard. However, instead of using the cin object, you use the ifstream/ fstream object.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
        fstream my_file;
        my_file.open("my_file.txt", ios::in);
        if (!my_file) {
                cout << "No such file";
        }
        else {
                char ch;

                while (1) {
                        my_file >> ch;
                        if (my_file.eof())
                                break;

                        cout << ch;
                }

        }
        my_file.close();
        return 0;
}
```

Special operations in a File

There are few important functions to be used with file streams like:

- `tellp()` - It tells the current position of the put pointer.
  Syntax: filepointer.tellp()
- `tellg()` - It tells the current position of the get pointer.
  Syntax: filepointer.tellg()
- `seekp()` - It moves the put pointer to mentioned location.
  Syntax: filepointer.seekp(no of bytes,reference mode)
- `seekg()` - It moves get pointer(input) to a specified location.
  Syntax: filepointer.seekg((no of bytes,reference point)
- `put()` - It writes a single character to file.
- `get()` - It reads a single character from file.

Tasks

Problem No: 1
Write a C++ program to write number 1 to 100 in a data file NOTES.txt.

Problem No: 2
Write a C++ program, which initializes a string variable to the content "**Time is a great teacher but unfortunately it kills all its pupils. Berlioz**" and write the string to a file OUT.txt. Then write a user-defined function in C++ to read the content from the text file OUT.txt, count and display the number of alphabets present in it.

Problem No: 3
Write a function to count number of words in the text file "OUT.txt"created in previous task.

Problem No: 4
Write a C++ program to create a file “STORY.txt” which contains following lines:
“The rose is red.
A girl is playing there.
There is a playground.
An aeroplane is in the sky.
Numbers are not allowed in the password.”
Then write a function in C++ to count and display the number of lines not starting with alphabet 'A' present in a text file.

Problem No: 5

Assuming that a text file named FIRST.TXT contains some text written into it, write a function named copyupper(), that reads the file FIRST.TXT and creates a new file named SECOND.TXT contains all words from the file FIRST.TXT in uppercase.