File Handling

- File represents storage medium for storing data or information.
- In files, we store data i.e. text or binary data and use these data to read or write in the form of Input/ Output Operations.
- So we use the term file Handling. We use the header file <fstream>

fstream Libarary Provides C++ programmers with three classes for working with files.

These classes include :-

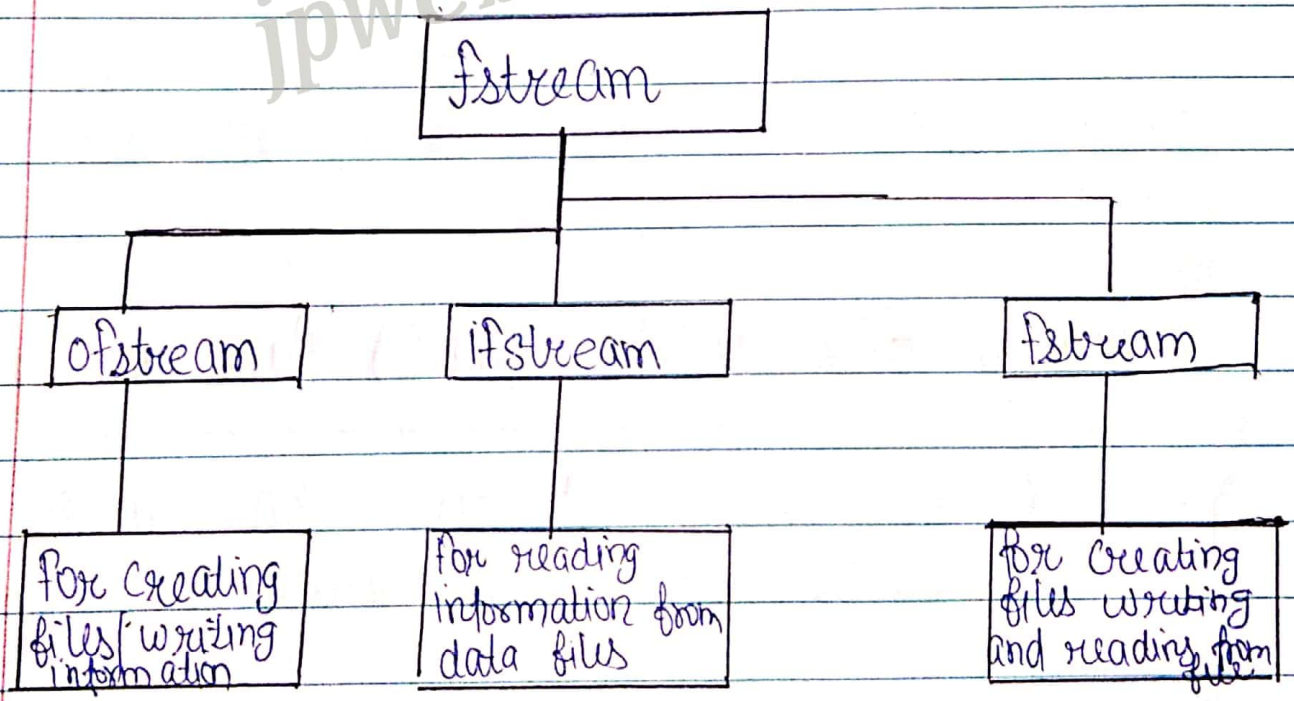
1. Ofstream
2. Ifstream
3. fstream

1. Ofstream :- These classes represents an output stream.

It is used to create files and to write information to files.

2. **Ifstream**:- These classes represents the input file stream and it is used to read information from files.

3. **fstream**:- It comes with ofstream | ifstream capabilities This stream class to both read and write from | to files.



(To use the above classes of the fstream Libarary, you must include it in your program as a header file)

Opening and Closing the files :-

1. Opening a file :-

- Before opening the file, input, output and stream must be created.
  - → To create an input stream, declare the ifstream class.
  - → To create an output stream, declare ofstream.
  - → To create an input/output stream, declare the fstream class.
- After creating the stream, open the file for processing.

Opening file with open() function

The Open() is used to open multiple files that use the same stream object.

Syntax:- file_stream-class stream_object;
stream_object.open("filename");

for example, to read input data from "Raj.txt" file, the open function with its stream can be used as:

```
ifstream infile;
infile.open("Raj.txt");
```

Similarly to write the data to the "Raj.txt" file, the open() function with its output stream

```
ofstream outfile;
outfile.open("Raj.txt");
```

Open() function modes :- The file mode tells that how to use a file i.e. How to read it, write it and how to append data to file etc.

File Mode	Description
ios::app	Append mode. All output to that file to be appended to the end. (add more data)
ios::ate	Open file for output & move to the read/write control to the end of the file
ios::in	Open a file for reading.
ios::out	Open a file for writing
ios::trunc	This will delete the perviaus content of the end of the file when file opened

file Mode	Description
ios::binary	This causes file open, Binary
ios::nocreate	This causes the open() to fail if the file does not already exist (It will not create a new file with that name)
ios::noreplace	This is used when you want to create a new file at the same time

Some examples in different file modes are:-

```
infile.open("abc.txt", ios::in);   // open for reading
outfile.open("abc.text", ios::out);   " " writing
```

We can be combined file modes by using C++ bitwise operator (|).

```
fstream afile;
afile.open("Ram.txt", ios::out | ios::in);
```

Closing a file

When a C++ program terminates it automatically closes all the opened files and release all the allocated memory.

But it is always a good practice that a programmer should close all the opened files before program termination.

Syntax:

```
void close();
```

* Example

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
ofstream file("filename.txt");
file.close();                    // close the fil
getch();
}
```

// Create and
// Open a text

Reading and Writing a file :-

1. **Write To a file)** - To write to the file, use the insertion operator (<<).

The only difference is we use ofstream and fstream object instead of cout object.

2. **Reading from files)** - To read from the file, use the extraction operator (>>).

However, instead of using the cin object, you use the ifstream/ fstream object.

Example :-

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void main()
{
ofstream file("filename.txt");
file << "Writing to a file ....";
file.close();
getch();
}
```