Data File Handling in C++

File. The information / data stored under a specific name on a storage device, is called a file.

Stream. It refers to a sequence of bytes.

Text file. It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (End of Line) character or delimiter character. When this EOL character is read or written, certain internal translations take place.

Binary file. It is a file that contains information in the same format as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

Classes for file stream operation

ofstream: Stream class to write on files
ifstream: Stream class to read from files
fstream: Stream class to both read and write from/to files.

Opening a file

OPENING FILE USING CONSTRUCTOR

```
ofstream outFile("sample.txt");     //output only
ifstream inFile(“sample.txt”);  //input only
```

OPENING FILE USING open()

Stream-object.open(“filename”, mode)

```
ofstream outFile;
outFile.open("sample.txt");

ifstream inFile;
inFile.open("sample.txt");
```

File mode parameter	Meaning
ios::app	Append to end of file
ios::ate	go to end of file on opening
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only
ios::nocreate	open fails if the file does not exist
ios::noreplace	open fails if the file already exist
ios::trunc	delete the contents of the file if it exist

All these flags can be combined using the bitwise operator OR (|). For example, if we want to open the file example.bin in binary mode to add data we could do it by the following call to member function open():

```
fstream file;
file.open ("example.bin", ios::out | ios::app | ios::binary);
```

Closing File

```
outFile.close();
inFile.close();
```

INPUT AND OUTPUT OPERATION

put() and get() function
the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated stream.
example :

```
file.get(ch);
file.put(ch);
```

write() and read() function
write() and read() functions write and read blocks of binary data.
example:

```
file.read((char *)&obj, sizeof(obj));
file.write((char *)&obj, sizeof(obj));
```

ERROR HANDLING FUNCTION

FUNCTION	RETURN VALUE AND MEANING
eof()	returns true (non zero) if end of file is encountered while reading; otherwise return false(zero)
fail()	return true when an input or output operation has failed
bad()	returns true if an invalid operation is attempted or any unrecoverable error has occurred.
good()	returns true if no error has occurred.

File Pointers and Their Manipulation

All i/o streams objects have, at least, one internal stream pointer:
ifstream, like istream, has a pointer known as the get pointer that points to the element to be read in the next input operation.

ofstream, like ostream, has a pointer known as the put pointer that points to the location where the next element has to be written.

Finally, fstream, inherits both, the get and the put pointers, from iostream (which is itself derived from both istream and ostream).

These internal stream pointers that point to the reading or writing locations within a stream can be manipulated using the following member functions:

seekg()	moves get pointer(input) to a specified location
seekp()	moves put pointer (output) to a specified location
tellg()	gives the current position of the get pointer
tellp()	gives the current position of the put pointer

The other prototype for these functions is:

seekg(offset, refposition);
seekp(offset, refposition);

The parameter offset represents the number of bytes the file pointer is to be moved from the location specified by the parameter refposition. The refposition takes one of the following three constants defined in the ios class.

ios::beg start of the file
ios::cur current position of the pointer
ios::end end of the file

example:

```
file.seekg(-10, ios::cur);
```