File Pointers

A pointer is used to handle and keep track of the files being accessed. Every file maintains two pointers called get_pointer (in input mode file) and put_pointer (in output mode file), which tells the current position where reading or writing will take place with the use of opening modes and their respective manipulators.

These pointers help attain random access to the file. Like Moving at once to any area withinside the record rather than shifting through it sequentially.

Functions of file handling:

seekg():-

It is used to move the get pointer to the desired location concerning a reference point.

Syntax: file_pointer.seekg (number of bytes ,Reference point);
Example: fin.seekg(10,ios::beg);

tellg():-

tellg() is used to realize which they get pointer is in a file.

Syntax: file_pointer.tellg();
Example: int posn = fin.tellg();

seekp():-

seekp() is used to move the put pointer to the desired location concerning a reference point.

Syntax: file_pointer.seekp(number of bytes ,Reference point);
Example: fout.seekp(10,ios::beg);

tellp():-

tellp()is used to realize which they get pointer is in a file.

Syntax: file_pointer.tellp();
Example: int posn=fout.tellp();

What are the reference points available?
The reference points are:

ios::beg – These are used from the beginning of the file

ios::end – These are used from the end of the file

ios::cur – These are used from the current position in the file.

Files Usage

Data Preservation: Storing files in documents allows to hold records even though this system terminates.

Easy Data Access: Accessing records will become easy whilst there may be a large number of records and it’s far saved with inside the record, then these files may be accessed the use of the C++ commands.

Portability: It will become less difficult to switch records from one pc to any other with no changes.
Operations On Files
Various operations that can be performed on a file are:

Creation of a new file (fopen with attributes as "a")
Opening an existing file (fopen).
Reading from file (fscanf or fgets).
Writing to a file (fprintf or fputs).
Moving to a specific location in a file (fseek, rewind).
Closing a file (fclose).
fstream library
Before diving into each sub-topics, let us first learn about the header file we will be using to gain access to the file handling method. In C++, the fstream library is used to handle files, and it is dealt with with the help of three classes known ofstream, ifstream, and fstream.

ofstream:

This class helps create and write the data to the file obtained from the program's output.

ifstream:

We use this class to read data from files and also known as the input stream.

fstream:

This class is the combination of both ofstream and ifstream. It provides the ability of creating, writing, and reading a file.

Object type that identifies a stream and contains the information needed to control it, including a pointer to its buffer, its position indicator and all its state indicators.

FILE objects are usually created by a call to either fopen or tmpfile, which both return a pointer to one of these objects.

File pointer is a pointer returned by **fopen()** library function. It is used to identify a file. It is passed to a **fread()** and **fwrite()** function.

Example of file pointer:

```
FILE *fp;

fp = fopen("sample.txt,"a");

fprintf( fp, "Welcome to GFG");

fclose(fp);
```

Example

```
/* FEOF example */
#include <stdio.h>

int main()
{
  FILE * pFile;
  char buffer [100];

  pFile = fopen ("myfile.txt" , "r");
  if (pFile == NULL) perror ("Error opening file");
  else
  {
    while ( ! feof (pFile) )
    {
      if ( fgets (buffer , 100 , pFile) == NULL ) break;
      fputs (buffer , stdout);
    }
    fclose (pFile);
  }
  return 0;
}
```

	File Pointer	**File descriptor**

1.	File pointer is allocated with fopen function call
	FILE *fp;
	fp = fopen("sample.txt,"a");
		File descriptor is allocated with open system call
		int fd = open(filePath, mode);

2.	It is generally, use for the application which is doing extensive read or write from a file	It is generally used for the application that do frequently random access of file
3.	It is a pointer	It is an integer value like 0, 1, 2
4.	The file pointer is buffered	The file descriptor is not buffered
5.	It is highly portable and efficient	It is less portable and efficient in comparison with the file pointer
6.	It is passed to fread() and fwrite() function	It is passed to read() and write() function
7.	It is not suitable for inter-process communication	It is suitable for inter-process communication
8.	Library functions generally use file pointer	System call generally use the file descriptor