**FILE HANDLING**

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).  
How to achieve the File Handling  
For achieving file handling we need to follow the following steps:-  
 STEP 1-Naming a file  
 STEP 2-Opening a file  
 STEP 3-Writing data into the file  
 STEP 4-Reading data from the file  
 STEP 5-Closing a file.

**Classes for File stream operations :-**

The I/O system of C++ contains a set of classes which define the file handling methods.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.   
**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.

1. ifstream:-

* This class provides input operations.
* It contains open() function with default input mode.
* Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

2. ofstream:-

* This class provides output operations.
* It contains open() function with default output mode.
* Inherits the functions put(),  write(), seekp() and tellp() functions from the ostream.

3. fstream:-

* This class provides support for simultaneous input and output operations.
* Inherits all the functions from istream and ostream classes through iostream.

**UNFORMATED I/O:**

Unformatted input/output (I/O) in C++ refers to reading and writing raw bytes of data, rather than formatted text or values. Unformatted I/O is typically used for binary file I/O or for transferring data between different systems or applications.

C++ provides a set of unformatted I/O functions that allow you to read and write raw bytes of data. Some of the commonly used functions for unformatted I/O include:

* + read() and write(): These functions are used for reading and writing raw bytes of data from and to a file or stream. The read() function reads a specified number of bytes from a file or stream into a buffer, while the write() function writes a specified number of bytes from a buffer to a file or stream.
  + get() and put(): These functions are used for reading and writing single characters from and to a file or stream. The get() function reads a single character from a file or stream, while the put() function writes a single character to a file or stream.
  + readsome(): This function is used for reading a specified number of characters from a stream into a buffer, without blocking if the requested number of characters is not immediately available.
  + getline(): This function is used for reading a line of text from a stream and storing it in a buffer.

**Formated I/O with ios member functions:**

In formatted I/O operation we use following functions to make output in perfect alignment. In industrial programming all the output should be perfectly formatted due to this reason in C++ provides many function to connect any file into perfect aligned format. These functions are available in header file <iomanip>. iomanip refers input output manipulators.

* + Width(x): This function is used to set width of the output.

Syntax: cout<<setw(int x);

* + fill(): This function is used to fill specified character at unused

space. Syntax cout<<setfill(‘character’)<<var;

* + precision(x): This method is used for setting floating point of the output. Syntax : cout<<setprecision(int x)<<var;
  + setflag(arg1,arg2): This function is used for setting format flag for

output. Syntax : setiosflags(arg1,arg2);

* + unsetflag(arg2): This function is used to reset set flags for output.

syntax: resetiosflag(arg2);

* + setbase(arg): This function is used to set base field of the flag.

Syntax: setbase(arg);

**File access Pointer and their manipulator:**

In C++, file access pointers are used to keep track of the current position within a file. These pointers are also known as file position indicators or file cursors.

The three standard file access pointers in C++ are:

1. **ios::beg**: This is the beginning of the file. It is the default position when the file is first opened.
2. **ios::cur**: This is the current position within the file.
3. **ios::end**: This is the end of the file.

You can manipulate these pointers using the following functions:

1. **tellg()**: This function returns the current position of the input pointer.
2. **tellp()**: This function returns the current position of the output pointer.
3. **seekg()**: This function moves the input pointer to a specified position in the file.
4. **seekp()**: This function moves the output pointer to a specified position in the file.

For example, let's say you want to read a file and print the contents from the middle of the file. You can use the following code to achieve this:

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream file("example.txt");

file.seekg(0, ios::end); // Move the pointer to the end of the file

int length = file.tellg(); // Get the length of the file

file.seekg(length/2, ios::beg); // Move the pointer to the middle of the file

string line;

while (getline(file, line)) { // Read and print the contents from the middle of the file

cout << line << endl;

}

file.close();

return 0; }