C++ Files and Streams

So far, we have been using the **iostream** standard library, which provides **cin**

and **cout** methods for reading from standard input and writing to standard output respectively.

fstream has three new data types −

|  |  |
| --- | --- |
| **Sr.No** | **Data Type & Description** |
| 1 | **ofstream**  This data type represents the output file stream and is used to create files and to write information to files. |
| 2 | **ifstream**  This data type represents the input file stream and is used to read information from files. |
| 3 | **fstream**  This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files. |

To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file.

## **Opening a File**

A file must be opened before you can read from it or write to it. Either **ofstream** or **fstream** object may be used to open a file for writing. And ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

void open(const char \*filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

|  |  |
| --- | --- |
| **Sr.No** | **Mode Flag & Description** |
| 1 | **ios::app**  Append mode. All output to that file to be appended to the end. |
| 2 | **ios::ate**  Open a file for output and move the read/write control to the end of the file. |
| 3 | **ios::in**  Open a file for reading. |
| 4 | **ios::out**  Open a file for writing. |
| 5 | **ios::trunc**  If the file already exists, its contents will be truncated before opening the file. |

You can combine two or more of these values by **OR-**ing them together. For example if you want to open a file in write mode and want to truncate it in case that already exists, following will be the syntax −

ofstream outfile;

outfile.open("file.dat", ios::out | ios::trunc );

Similar way, you can open a file for reading and writing purpose as follows −

fstream afile;

afile.open("file.dat", ios::out | ios::in );

## **Closing a File**

When a C++ program terminates it automatically flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

void close();

## **Writing to a File**

While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an **ofstream** or **fstream** object instead of the **cout**object.

## **Reading from a File**

You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an **ifstream** or **fstream** object instead of the **cin** object.

## **Read and Write Example**

Following is the C++ program which opens a file in reading and writing mode. After writing information entered by the user to a file named afile.dat, the program reads information from the file and outputs it onto the screen −

## **Implementation by using ifsream & ofstream classes:**

/\* File Handling with C++ using ifstream & ofstream class object\*/

/\* To write the Content in File\*/

/\* Then to read the content of file\*/

#include <iostream>

/\* fstream header file for ifstream, ofstream,

  fstream classes \*/

#include <fstream>

using namespace std;

// Driver Code

int main()

{

    // Creation of ofstream class object

    ofstream fout;

    string line;

    // by default ios::out mode, automatically deletes

    // the content of file. To append the content, open in ios:app

    // fout.open("sample.txt", ios::app)

    fout.open("sample.txt");

    // Execute a loop If file successfully opened

    while (fout) {

        // Read a Line from standard input

        getline(cin, line);

        // Press -1 to exit

        if (line == "-1")

            break;

        // Write line in file

        fout << line << endl;

    }

    // Close the File

    fout.close();

    // Creation of ifstream class object to read the file

    ifstream fin;

    // by default open mode = ios::in mode

    fin.open("sample.txt");

    // Execute a loop until EOF (End of File)

    while (fin) {

        // Read a Line from File

        getline(fin, line);

        // Print line in Console

        cout << line << endl;

    }

    // Close the file

    fin.close();

    return 0;

}

## **Implementation by using fstream class**:

/\* File Handling with C++ using fstream class object \*/

/\* To write the Content in File \*/

/\* Then to read the content of file\*/

#include <iostream>

/\* fstream header file for ifstream, ofstream,

   fstream classes \*/

#include <fstream>

using namespace std;

// Driver Code

int main()

{

    // Creation of fstream class object

    fstream fio;

    string line;

    // by default openmode = ios::in|ios::out mode

    // Automatically overwrites the content of file, To append

    // the content, open in ios:app

    // fio.open("sample.txt", ios::in|ios::out|ios::app)

    // ios::trunc mode delete all conetent before open

    fio.open("sample.txt", ios::trunc | ios::out | ios::in);

    // Execute a loop If file successfully Opened

    while (fio) {

        // Read a Line from standard input

        getline(cin, line);

        // Press -1 to exit

        if (line == "-1")

            break;

        // Write line in file

        fio << line << endl;

    }

    // Execute a loop untill EOF (End of File)

    // point read pointer at beginning of file

    fio.seekg(0, ios::beg);

    while (fio) {

        // Read a Line from File

        getline(fio, line);

        // Print line in Console

        cout << line << endl;

    }

    // Close the file

    fio.close();

    return 0;

}

## **File Position Pointers**

Both **istream** and **ostream** provide member functions for repositioning the file-position pointer. These member functions are **seekg** ("seek get") for istream and **seekp** ("seek put") for ostream.

The argument to seekg and seekp normally is a long integer. A second argument can be specified to indicate the seek direction. The seek direction can be **ios::beg** (the default) for positioning relative to the beginning of a stream, **ios::cur** for positioning relative to the current position in a stream or **ios::end** for positioning relative to the end of a stream.

The file-position pointer is an integer value that specifies the location in the file as a number of bytes from the file's starting location. Some examples of positioning the "get" file-position pointer are −

// position to the nth byte of fileObject (assumes ios::beg)

fileObject.seekg( n );

// position n bytes forward in fileObject

fileObject.seekg( n, ios::cur );

// position n bytes back from end of fileObject

fileObject.seekg( n, ios::end );

// position at end of fileObject

**fileObject.seekg( 0, ios::end );**

**Seek Direction:**

**ios::beg (default)**

**ios::cur**

**ios::end**

**Seek function:**

**seekg()**

**seekp()**

## **Examples:**

### Create File

#include<iostream>

#include<conio>

#include <fstream>

using namespace std;

int main()

{

fstream st; // Step 1: Creating object of fstream class

st.open("E:\studytonight.txt",ios::out); // Step 2: Creating new file

if(!st) // Step 3: Checking whether file exist

{

cout<<"File creation failed";

}

else

{

cout<<"New file created";

st.close(); // Step 4: Closing file

}

getch();

return 0;

}

### Writing to a File

#include <iostream>

#include<conio>

#include <fstream>

using namespace std;

int main()

{

fstream st; // Step 1: Creating object of fstream class

st.open("E:\studytonight.txt",ios::out); // Step 2: Creating new file

if(!st) // Step 3: Checking whether file exist

{

cout<<"File creation failed";

}

else

{

cout<<"New file created";

st<<"Hello"; // Step 4: Writing to file

st.close(); // Step 5: Closing file

}

getch();

return 0;

}

Here we are sending output to a file. So, we use ios::out. As given in the program, information typed inside the quotes after **"FilePointer <<"** will be passed to output file.

### Reading from a File

#include <iostream>

#include<conio>

#include <fstream>

using namespace std;

int main()

{

fstream st; // step 1: Creating object of fstream class

st.open("E:\studytonight.txt",ios::in); // Step 2: Creating new file

if(!st) // Step 3: Checking whether file exist

{

cout<<"No such file";

}

else

{

char ch;

while (!st.eof())

{

st >>ch; // Step 4: Reading from file

cout << ch; // Message Read from file

}

st.close(); // Step 5: Closing file

}

getch();

return 0;

}

Here we are reading input from a file. So, we use ios::in. As given in the program, information from the output file is obtained with the help of following syntax **"FilePointer >>variable"**.

### Close a File

It is done by FilePointer.close().

#include <iostream>

#include<conio>

#include <fstream>

using namespace std;

int main()

{

fstream st; // Step 1: Creating object of fstream class

st.open("E:\studytonight.txt",ios::out); // Step 2: Creating new file

st.close(); // Step 4: Closing file

getch();

return 0;

}

## **Special operations in a File**

here 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.
* ***Note:****For seekp and seekg three reference points are passed:****ios::beg****- beginning of the file****ios::cur****- current position in the file****ios::end****- end of the file*

Below is a program to show importance of tellp, tellg, seekp and seekg:

#include <iostream>

#include<conio>

#include <fstream>

using namespace std;

int main()

{

fstream st; // Creating object of fstream class

st.open("E:\studytonight.txt",ios::out); // Creating new file

if(!st) // Checking whether file exist

{

cout<<"File creation failed";

}

else

{

cout<<"New file created"<<endl;

st<<"Hello Friends"; //Writing to file

// Checking the file pointer position

cout<<"File Pointer Position is "<<st.tellp()<<endl;

st.seekp(-1, ios::cur); // Go one position back from current position

//Checking the file pointer position

cout<<"As per tellp File Pointer Position is "<<st.tellp()<<endl;

st.close(); // closing file

}

st.open("E:\studytonight.txt",ios::in); // Opening file in read mode

if(!st) //Checking whether file exist

{

cout<<"No such file";

}

else

{

char ch;

st.seekg(5, ios::beg); // Go to position 5 from begning.

cout<<"As per tellg File Pointer Position is "<<st.tellg()<<endl; //Checking file pointer position

cout<<endl;

st.seekg(1, ios::cur); //Go to position 1 from beginning.

cout<<"As per tellg File Pointer Position is "<<st.tellg()<<endl; //Checking file pointer position

st.close(); //Closing file

}

getch();

return 0;

}

Output:

New file created

File Pointer Position is 13

As per tellp File Pointer Position is 12

As per tellg File Pointer Position is 5

As per tellg File Pointer Position is 6

## Q1: Read a characters from a file as well as to count the number lines of any particular file consist of

First create a file called my-input-file.txt which will contain some text. For example:

welcome to  
linuxconfig.org  
c++

#include <iostream>  
#include <fstream>  
  
**using** **namespace** std;  
  
int main() {  
  
ifstream fin;  
fin.open("my-input-file.txt", ios::in);  
  
char my\_character ;  
int number\_of\_lines = 0;  
  
 **while** (!fin.eof() ) {  
  
 fin.get(my\_character);  
 cout << my\_character;  
 **if** (my\_character == '\n'){  
 ++number\_of\_lines;  
 }  
 }  
cout << "NUMBER OF LINES: " << number\_of\_lines << endl;  
  
}

NOTE: while loop and fin.get will add extra new line character so you may start with :

int number\_of\_lines = -1

**OUTPUT:**

welcome to  
linuxconfig.org  
c++  
  
NUMBER OF LINES: 4

# Functions

# is\_open

**Check if file is open**

Returns whether the stream is currently associated to a file.  
  
Streams can be associated to files by a successful call to member [open](http://www.cplusplus.com/basic_fstream::open) or directly on construction, and disassociated by calling [close](http://www.cplusplus.com/basic_fstream::close) or on destruction.

**bool is\_open() const;**

### Parameters

none

### Return Value

true if a file is open and associated with this *stream* object.  
false otherwise.

### Example

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | // fstream::is\_open  #include <iostream> // std::cout  #include <fstream> // std::fstream  int main () {  std::fstream fs;  fs.open ("test.txt");  if (fs.is\_open())  {  fs << "lorem ipsum";  std::cout << "Operation successfully performed\n";  fs.close();  }  else  {  std::cout << "Error opening file";  }  return 0;  } | [Edit & Run](http://www.cplusplus.com/reference/fstream/basic_fstream/is_open/) |

Possible output:

|  |
| --- |
| Operation successfully performed |

# C++ program to read file word by word

**Example :**

Input: And in that dream, we were flying.

Output:

And

in

that

dream,

we

were

flying.

|  |
| --- |
| // C++ implementation to read  // file word by word  #include <bits/stdc++.h>  using namespace std;    // driver code  int main()  {      // filestream variable file      fstream file;      string word, t, q, filename;        // filename of the file      filename = "file.txt";        // opening file      file.open(filename.c\_str());        // extracting words from the file      while (file >> word)      {          // displaying content          cout << word << endl;      }        return 0;  } |

Output:

geeks

for

geeks.

# C++ program to append content of one text file to another

**Approach:**  
**1)**Open **file.txt** in inputstream and **file2.txt** in outputstream with append option, so that the previous content of file are not deleted.  
**2)**Check if there’s an error in opening or locating a file. If yes, then throw an error message.  
**3)**If both the files are found, then write content from source file to destination file.  
**4)**Display the content of destination file.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C++ implementation to append  // content from source file to  // destination file  #include <bits/stdc++.h>  using namespace std;    // driver code  int main()  {      fstream file;        // Input stream class to      // operate on files.      ifstream ifile("file.txt", ios::in);        // Output stream class to      // operate on files.      ofstream ofile("file2.txt", ios::out | ios::app);        // check if file exists      if (!ifile.is\_open()) {            // file not found (i.e, not opened).          // Print an error message.          cout << "file not found";      }      else {          // then add more lines to          // the file if need be          ofile << ifile.rdbuf();      }      string word;        // opening file      file.open("file2.txt");        // extracting words form the file      while (file >> word) {            // displaying content of          // destination file          cout << word << " ";      }        return 0;  } |

Output:

geeks for geeks

# C++ program to print unique words in a file

|  |
| --- |
| // C++ program to print unique words in a string  #include <bits/stdc++.h>  using namespace std;    // Prints unique words in a file  void printUniquedWords(char filename[])  {      // Open a file stream      fstream fs(filename);        // Create a map to store count of all words      map<string, int> mp;        // Keep reading words while there are words to read      string word;      while (fs >> word)      {          // If this is first occurrence of word          if (!mp.count(word))              mp.insert(make\_pair(word, 1));          else              mp[word]++;      }        fs.close();        // Traverse map and print all words whose count      //is 1      for (map<string, int> :: iterator p = mp.begin();           p != mp.end(); p++)      {          if (p->second == 1)              cout << p->first << endl;      }  }    // Driver program  int main()  {      // Create a file for testing and write something in it      char filename[] = "test.txt";      ofstream fs(filename, ios::trunc);      fs << "geeks for geeks quiz code geeks practice for qa";      fs.close();        printUniquedWords(filename);      return 0;  } |

Output:

code

practice

qa

quiz

<https://www.geeksforgeeks.org/csv-file-management-using-c/>