C++ File Handling: How to Open, Write, Read, Close Files in C++

What is file handling in C++?

Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length. This will be determined by their usage. C++ provides you with a library that comes with methods for file handling. Let us discuss it.

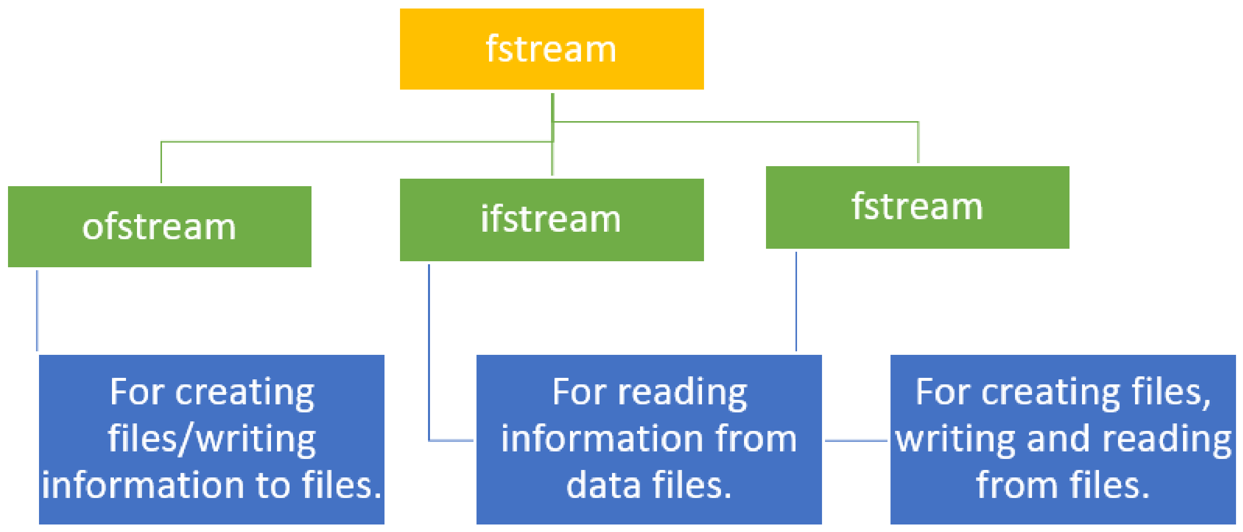
In this c++ tutorial, you will learn:

The fstream Library

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:


fstream
ofstream
ifstream
fstream
For creating files/writing information to files.
For reading information from data files.
For creating files, writing and reading from files.

fstream library

To use the above classes of the fstream library, you must include it in your program as a header file. Of course, you will use the #include preprocessor directive. You must also include the iostream header file.

How to Open Files

Before performing any operation on a file, you must first open it. If you need to write to the file, open it using fstream or ofstream objects. If you only need to read from the file, open it using the ifstream object.

The three objects, that is, fstream, ofstream, and ifstream, have the open() function defined in them. The function takes this syntax:

```
open (file_name, mode);
```

- The file_name parameter denotes the name of the file to open.
- The mode parameter is optional. It can take any of the following values:

Value	**Description**
ios:: app	The Append mode. The output sen
ios::ate	It opens the file for the output then
ios::in	It opens the file for a read.

ios::out	It opens the file for a write.
ios::trunk	If a file exists, the file elements sho

It is possible to use two modes at the same time. You combine them using the | (OR) operator.

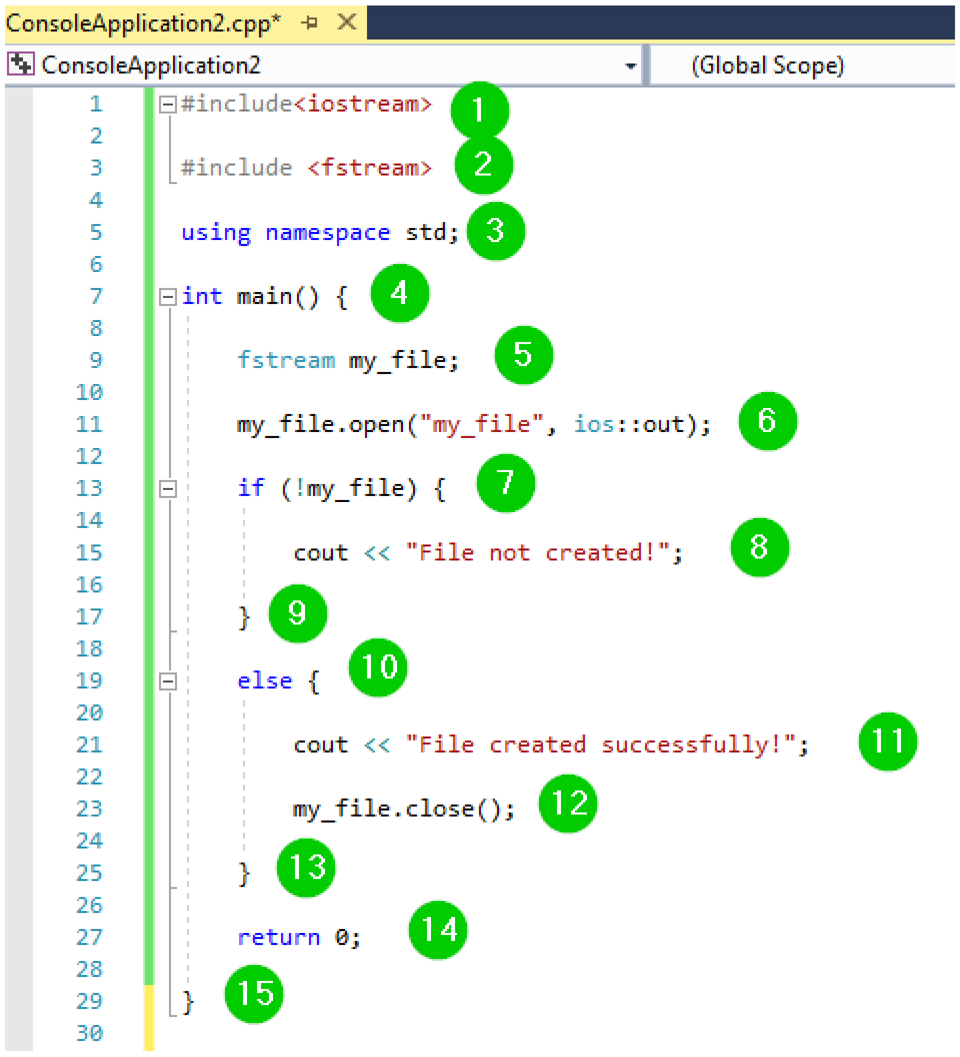
Example 1:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
	fstream my_file;
	my_file.open("my_file", ios::out);
	if (!my_file) {
		cout << "File not created!";
	}
	else {
		cout << "File created successfully!";
		my_file.close();
	}
	return 0;
}
```

Output:

```
File created successfully!
```

Here is a screenshot of the code:



```cpp
#include<iostream>

#include <fstream>

using namespace std;

int main() {

	fstream my_file;

	my_file.open("my_file", ios::out);

	if (!my_file) {

		cout << "File not created!";

	}

	else {

		cout << "File created successfully!";

		my_file.close();

	}

	return 0;

}
```

Code Explanation:

1. Include the iostream header file in the program to use its functions.
2. Include the fstream header file in the program to use its classes.
3. Include the std namespace in our code to use its classes without calling it.
4. Call the main() function. The program logic should go within its body.
5. Create an object of the fstream class and give it the name my_file.
6. Apply the open() function on the above object to create a new file. The out mode allows us to write into the file.
7. Use if statement to check whether file creation failed.
8. Message to print on the console if the file was not created.
9. End of the body of if statement.
10. Use an else statement to state what to do if the file was created.
11. Message to print on the console if the file was created.
12. Apply the close() function on the object to close the file.
13. End of the body of the else statement.
14. The program must return value if it completes successfully.
15. End of the main() function body.

How to Close Files

Once a C++ program terminates, it automatically

- flushes the streams
- releases the allocated memory
- closes opened files.

However, as a programmer, you should learn to close open files before the program terminates.

The fstream, ofstream, and ifstream objects have the close() function for closing files. The function takes this syntax:

```
void close();
```

How to Write to Files

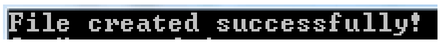
You can write to file right from your C++ program. You use stream insertion operator (<<) for this. The text to be written to the file should be enclosed within double-quotes.

Let us demonstrate this.

Example 2:

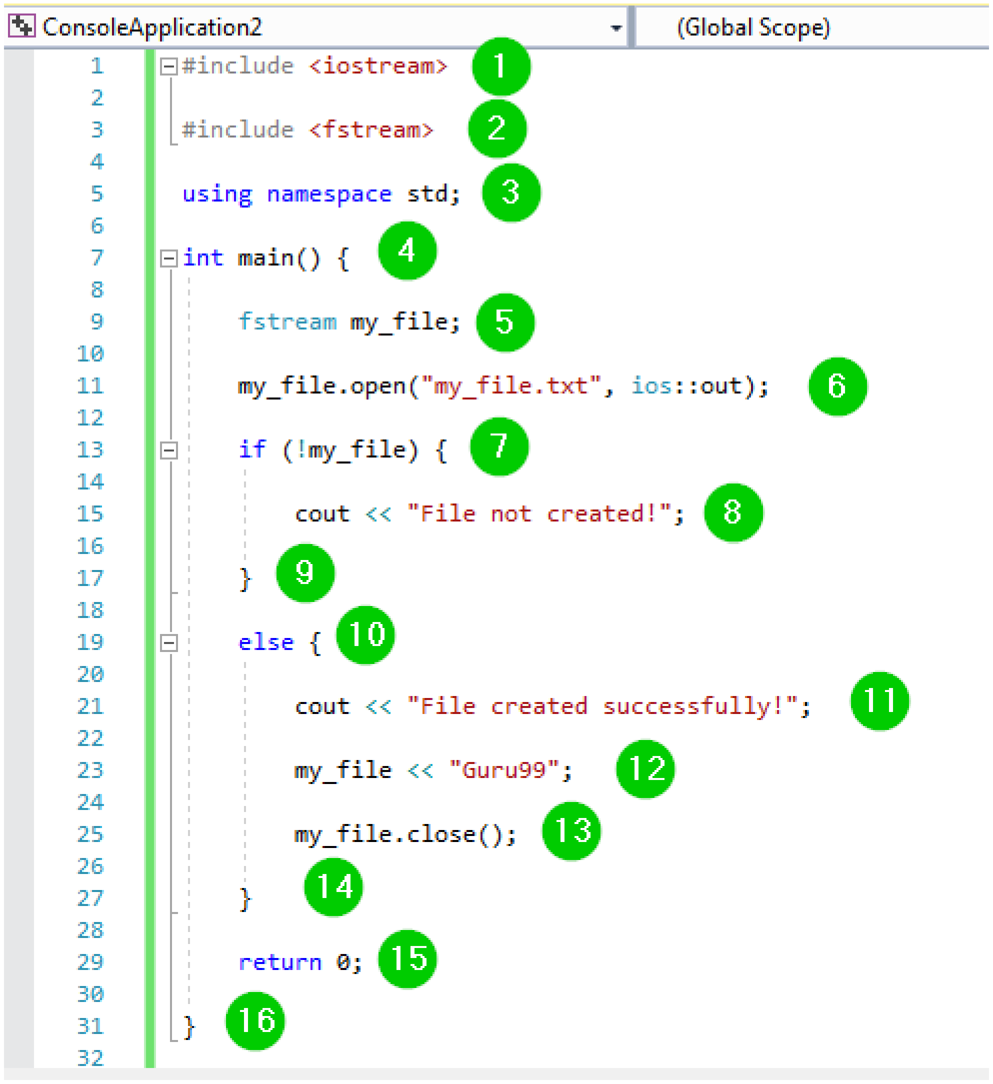
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
        fstream my_file;
        my_file.open("my_file.txt", ios::out);
        if (!my_file) {
                cout << "File not created!";
        }
        else {
                cout << "File created successfully!";
                my_file << "Guru99";
                my_file.close();
        }
        return 0;
}
```

Output:



```
File created successfully!
```

Here is a screenshot of the code:



```cpp
#include <iostream>

#include <fstream>

using namespace std;

int main() {

	fstream my_file;

	my_file.open("my_file.txt", ios::out);

	if (!my_file) {

		cout << "File not created!";

	}

	else {

		cout << "File created successfully!";

		my_file << "Guru99";

		my_file.close();

	}

	return 0;

}
```

Code Explanation:

1. Include the iostream header file in the program to use its functions.
2. Include the fstream header file in the program to use its classes.
3. Include the std namespace in the program to use its classes without calling it.
4. Call the main() function. The program logic should be added within the body of this function.
5. Create an instance of the fstream class and give it the name my_file.
6. Use the open() function to create a new file named my_file.txt. The file will be opened in the out mode for writing into it.
7. Use an if statement to check whether the file has not been opened.
8. Text to print on the console if the file is not opened.
9. End of the body of the if statement.
10. Use an else statement to state what to do if the file was created.
11. Text to print on the console if the file was created.
12. Write some text to the created file.

13. Use the close() function to close the file.
14. End of the body of the else statement.
15. The program must return value upon successful completion.
16. End of the body of the main() function.

How to Read from Files

You can read information from files into your C++ program. This is possible using stream extraction operator (>>). You use the operator in the same way you use it to read user input from the keyboard. However, instead of using the cin object, you use the ifstream/ fstream object.

Example 3:

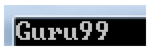
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
	fstream my_file;
	my_file.open("my_file.txt", ios::in);
	if (!my_file) {
		cout << "No such file";
	}
	else {
		char ch;

		while (1) {
			my_file >> ch;
			if (my_file.eof())
				break;

			cout << ch;
		}

	}
	my_file.close();
	return 0;
}
```

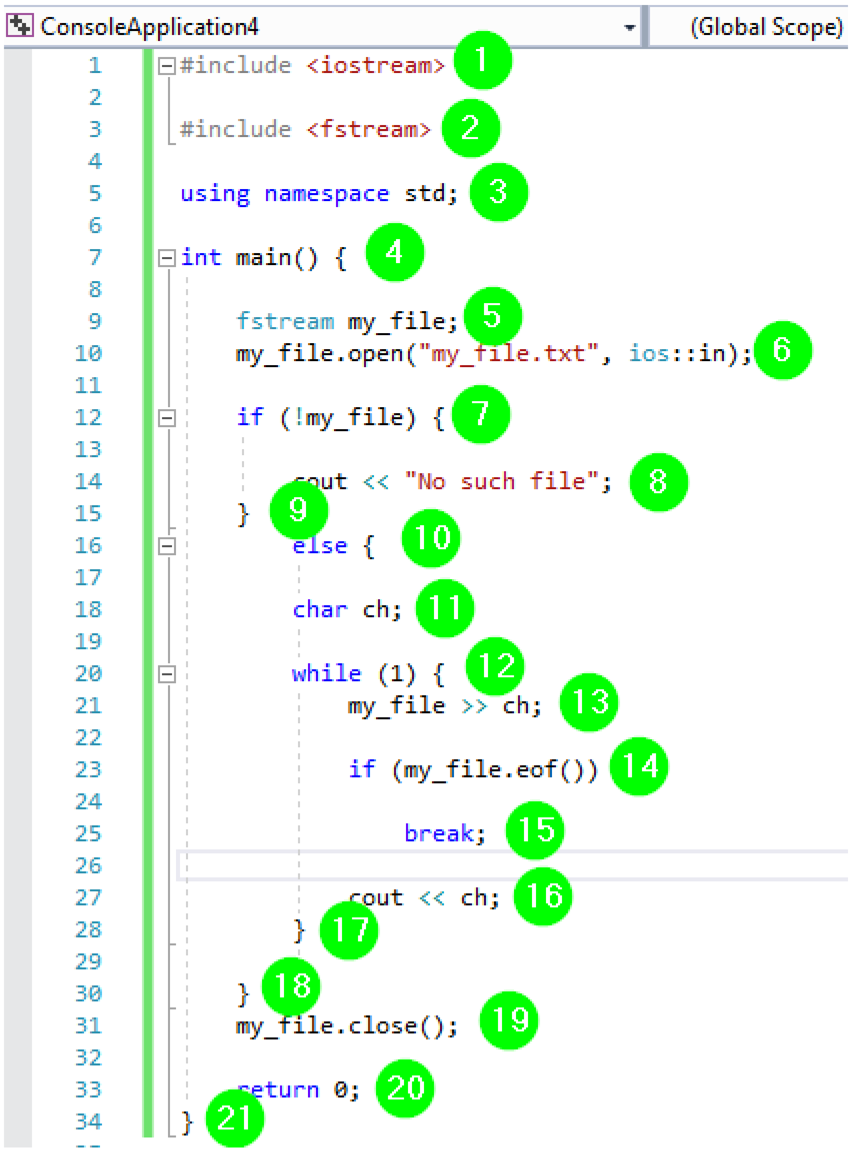
Output:



```
Guru99
```

No such file

Here is a screenshot of the code:



```cpp
#include <iostream>

#include <fstream>

using namespace std;

int main() {

    fstream my_file;
    my_file.open("my_file.txt", ios::in);

    if (!my_file) {

        cout << "No such file";
    }
        else {

        char ch;

        while (1) {
            my_file >> ch;

            if (my_file.eof())

                break;


            cout << ch;
        }

    }
    my_file.close();

    return 0;
}
```

Code Explanation:

1. Include the iostream header file in the program to use its functions.
2. Include the fstream header file in the program to use its classes.
3. Include the std namespace in the program to use its classes without calling it.
4. Call the main() function. The program logic should be added within the body of this function.
5. Create an instance of the fstream class and give it the name my_file.
6. Use the open() function to create a new file named my_file.txt. The file will be opened in the in mode for reading from it.
7. Use an if statement to check whether the file does not exist.
8. Text to print on the console if the file is not found.
9. End of the body of the if statement.
10. Use an else statement to state what to do if the file is found.
11. Create a char variable named ch.
12. Create a while loop for iterating over the file contents.

13. Write/store contents of the file in the variable ch.
14. Use an if condition and eof() function that is, end of the file, to ensure the compiler keeps on reading from the file if the end is not reached.
15. Use a break statement to stop reading from the file once the end is reached.
16. Print the contents of variable ch on the console.
17. End of the while body.
18. End of the body of the else statement.
19. Call the close() function to close the file.
20. The program must return value upon successful completion.
21. End of the body of the main() function.

Example 4 (Read Lines):

```
#include <iostream>
#include <fstream>
#include<string>
using namespace std;
int main() {
    string lines;
    ifstream myfile ("mytext.txt" , ios::in);
    if(!myfile.is_open())
    {
        cerr << "Can not open file!";
        myfile.close();
        return 0;
    }
    else
    {
        while (!myfile.eof())
        {
            string line;
            getline(myfile,line);
            lines += line + "\n";
        }
        lines = lines.substr(0,lines.length()-1);
        cout << lines;
    }
    return 0;
}
```

Positioning:

- Tellg() : The **tellg()** function is used with input streams, and returns the current “get” position of the pointer in the stream.

```cpp
#include <iostream>
#include <string>
#include <sstream>

int main()
{
    std::istringstream in("Hello, World!");
    std::string word1, word2, word3, word4, word5;

    in >> word1;
    in.seekg(0, std::ios_base::beg); // <- rewind
    in >> word2;
    in.seekg(1, std::ios_base::cur); // -> seek from cur pos toward the end
    in >> word3;
    in.seekg(-6, std::ios_base::cur); // <- seek from cur pos (end) toward begin
    in >> word4;
    in.seekg(-6, std::ios_base::end); // <- seek from end toward begin
    in >> word5;

    std::cout << "word1 = " << word1 << '\n'
              << "word2 = " << word2 << '\n'
              << "word3 = " << word3 << '\n'
              << "word4 = " << word4 << '\n'
              << "word5 = " << word5 << '\n';
}
```

Summary:

- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The fstream library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the #include preprocessor directive.