File Handling in C++ | How to Open, Read and Close

What is File Handling in C++?

File handling in C++ is a mechanism to store the output of a program in a file and help perform various operations on it. Files help store these data permanently on a storage device.

The term “Data” is commonly referred to as known facts or information. In the present era, data plays a vital role. It helps to describe, diagnose, predict or prescribe. But to achieve all this, we need to store it somewhere. You all would argue that there are so many text editors like ‘Notepad’ and ‘MS Office’, which help us store data in the form of text. You are right! But here we are discussing at a level of programming. In contrast, text editors like ‘Notepad’ and ‘MS Office’ are pre-built and cannot be accessed at the programming level to store data. File Handling is a hot topic when it comes to storing such programming data.

Almost every programming language has a ‘File Handling’ method to deal with the storage of data. In this article, we will learn about file handling in C++.

Now, This topic of file handling is further divided into sub-topics:

- Create a file
- Open a file
- Read from a file
- Write to a file
- Close a file

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++, fstream library is used to handle files, and it is dealt with the help of three classes known as ofstream, ifstream and fstream.

ofstream:

This class helps create and write the data to the file obtained from the program's output. It is also known as the input stream.

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 capability of creating, writing and reading a file.

To access the following classes, you must include the fstream as a header file like how we declare iostream in the header.

Example:

```
#include<iostream>

#include<fstream>
```

After including the header file, there comes a question saying do we need to create the file within the program or else do we need to use an existing file. But this isn't that difficult to answer because, in C++, we get four different methods to handle files. Let's discuss them one by one.

File Operations in C++

C++ provides us with four different operations for file handling. They are:

1. **open()** – This is used to create a file.
2. **read()** **–** This is used to read the data from the file.
3. **write() –** This is used to write new data to file.
4. **close() –** This is used to close the file.

We will look into each of these and try to understand them better.

Opening files in C++

To read or enter data to a file, we need to open it first. This can be performed with the help of 'ifstream' for reading and 'fstream' or 'ofstream' for writing or appending to the file. All these three objects have open() function pre-built in them.

Syntax:

```
1	open( FileName , Mode );
```

Here:

FileName – It denotes the name of file which has to be opened.

Mode – There different mode to open a file and it explained in this article.

Mode	**Description**
iso::in	File opened in reading mode

iso::out	File opened in write mode
iso::app	File opened in append mode
iso::ate	File opened in append mode but read and write performed at the end of the file.
iso::binary	File opened in binary mode
iso::trunc	File opened in truncate mode
iso::nocreate	The file opens only if it exists
iso::noreplace	The file opens only if it doesn't exist

In C++, we can use two modes simultaneously with the help of | (OR) operator.

Program for Opening File:

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
    fstream FileName;
    FileName.open("FileName", ios::out);
    if (!FileName){
        cout<<"Error while creating the file";
    }
    else{
        cout<<"File created successfully";
        FileName.close();
    }
    return 0;
}
```

Explanation of above code

1. Here we have an iostream library, which is responsible for input/output stream.

2. We also have a fstream library, which is responsible for handling files.
3. Creating an object of the fstream class and named it as ‘FileName’.
4. On the above-created object, we have to apply the open() function to create a new file, and the mode is set to ‘out’ which will allow us to write into the file.
5. We use the ‘if’ statement to check for the file creation.
6. Prints the message to console if the file doesn’t exist.
7. Prints the message to console if the file exists/created.
8. We use the close() function on the object to close the file.

Output

```
File created successfully
```

Writing to File

Till now, we learned how to create the file using C++. Now, we will learn how to write data to file which we created before. We will use fstream or ofstream object to write data into the file and to do so; we will use stream insertion operator (<<) along with the text enclosed within the double-quotes.

With the help of open() function, we will create a new file named ‘FileName’ and then we will set the mode to ‘ios::out’ as we have to write the data to file.

Syntax:

```
FileName<<"Insert the text here";
```

Program for Writing to File:

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
    fstream FileName;
    FileName.open("FileName.txt", ios::out);
    if (!FileName) {
        cout<<" Error while creating the file ";
    }
    else {
        cout<<"File created and data got written to file";
        FileName<<"This is a blog posted on Great Learning";
        FileName.close();
    }
    return 0;
}
```

Explanation of above code

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library, which is responsible for handling files.
3. Creating an object of the fstream class and named it as ‘FileName’.
4. On the above-created object, we have to apply the open() function to create a new file, and the mode is set to ‘out’ which will allow us to write into the file.
5. We use the ‘if’ statement to check for the file creation.
6. Prints the message to console if the file doesn’t exist.
7. Prints the message to console if the file exists/created.
8. Writing the data to the created file.
9. We use the close() function on the object to close the file.

Output

```
File created and data got written to file
```

Reading from file in C++

Getting the data from the file is an essential thing to perform because without getting the data, we cannot perform any task. But don’t worry, C++ provides that option too. We can perform the reading of data from a file with the CIN to get data from the user, but then we use CIN to take inputs from the user's standard console. Here we will use fstream or ifstream.

Syntax:

```
FileName>>Variable;
```

Content of FileName.txt:

Hello World, Thank You for Visiting Great Learning.

Program for Reading from File:

```
#include<iostream>
#include <fstream>
using namespace std;
int main() {
    fstream FileName;
    FileName.open("FileName.txt", ios::in);
    if (!FileName) {
        cout<<"File doesn’t exist.";
    }
    else {
        char x;
        while (1) {
            FileName>>x;
            if(FileName.eof())
                break;
            cout<<x;
        }
    }
    FileName.close();
    return 0;
}
```

Explanation of above code

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library which is responsible for handling files.
3. Creating an object of the fstream class and named it ‘FileName’.

4. On the above-created object, we have to apply the open() function to create a new file, and the mode is set to ‘in’ which will allow us to read from the file.
5. We use the ‘if’ statement to check for the file creation.
6. Prints the message to console if the file doesn’t exist.
7. Creating a character(char) variable with the named x.
8. Iterating of the file with the help of while loop.
9. Getting the file data to the variable x.
10. Here we are using if condition with eof() which stands for the end of the file to tell the compiler to read till the file’s end.
11. We use the ‘break’ statement to stop the reading from file when it reaches the end.
12. The print statement to print the content that is available in the variable x.
13. We use the close() function on the object to close the file

Output

Hello World, Thank You for Visiting Great Learning.

Infile C++

We can also use inFile to read from the file. Here, inFile >> S takes in the file stream, which is your file data, and uses a space delimiter (breaks it up by whitespace) and then puts the contents in the variable S.

Example:

If we had a file that had the data:

“My Cat is Hungry”

and we used inFile >> S here, i.e.:

```
ifstream inFile("file.txt")
string words;
while(inFile >> words) {
   cout << words << endl;
}
```

We will get the output:

```
My

Cat

is

Hungry
```

The inFile >> words will continue to return 'true' until there are no more items separated by whitespace.

Closing a file in C++

Closing a file is a good practice, and it is must to close the file. Whenever the C++ program comes to an end, it clears the allocated memory, and it closes the file. We can perform the task with the help of close() function.

Syntax:

```
FileName.close();
```

Program to Close a File:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    fstream FileName;
    FileName.open("FileName.txt", ios::in);
    if (!FileName) {
        cout<<"File doesn’t exist";
    }
    else {
        cout<<"File opened successfully";
        }
    }
    FileName.close();
    return 0;
}
```

Explanation of above code

1. Here we have an iostream library, which is responsible for input/output stream.
2. We also have a fstream library, which is responsible for handling files.
3. Creating an object of the fstream class and named it as ‘FileName’.
4. On the above-created object, we will apply the open() function to create a new file, and the mode is set to ‘out’ which allows us to write into the file.
5. We use the ‘if’ statement to check for the file creation.
6. Prints the message to console if the file doesn’t exist.
7. Prints the message to console if the file opened or not.
8. We use the close() function on the object to close the file.

File Position Pointers

We can reposition the file-position pointer in istream and ostream using its special member functions. These member functions are ‘seekg’ and ‘seekp’. ‘seekg’ or ‘seek get’ is used for istream and ‘seekp’ or ‘seek put’ is used for ostream.

Both these member functions take long integer as arguments. A second argument is used to specify the direction of seek. The seek directions can be ios::beg(for positioning in the beginning of a stream), ios::cur(for positioning relative to a current position of a stream) and ios::end(to position relative to the end of a stream).