----------------------------------------

e1: Video - Introduction

----------------------------------------

fstream: header file.

ofstream: Stream class to write on files. [Output File Stream]

ifstream: Stream class to read from files. [Input File Stream]

Hi and welcome to the next topic of this module. In this topic, we will explore how to deal with files in C++.

Well, handling files in C++ is a huge topic in itself. And we can have an entire training dedicated on file handling. But since this a beginner level training, so we will just explore the basics of how to create a file, write to it and read from a file.

**MM** So let’s get started.

To handle files in C++, we basically use two classes. The first class is **ofstream** and the second class is **ifstream**.

Now ofStream stands for Output File Stream and is used to write on a file.

And ifstream stands for input file stream and is used to read from a file. Nice and simple.

Now these two classes belong to a **MMM** header file known as **fstream** which stands for File Streams. So while handling files, we will include this header file in our program.

**UP NEXT**

In the next video, we will explore how to create a file, and write on it.

---------------------------------------------------

e2: Video - Create and Write on a file

---------------------------------------------------

// Create an ofstream (Output File Stream) object.

// Create a file (if it doesn't exist) and open it.

// Write on the file.

// Close the opened file.

Let us now proceed and explore **MM** how to create a file and write on it.

**DEMO**

Before we start, we need to include a header file <fstream> in our program.

TVT

So let’s do it.

**#include** <fstream>

Well, these are the four basic steps that we will follow in this video. Let’s explore these steps one by one.

CALLOUT

We will first, create an object of ofstream.

TVT

ofstream oFile;

Like this.

CALLOUT

Then we will, create a file if it doesn’t exist and open it. So that we can write on it.

TTT

I will use oFile object ---- and call open function ---- and provide the name of the file that we want to open.

oFile.open("my-note.txt");

So this statement will open a file named -- my-note.txt --- If this file doesn’t exist then it will first create it. Fine? And to name this file, I am using Kebab case naming convention. So please note that.

Our next step will be to write on the opened file.

TTT

I will use oFile object ---- and use the insertion operator. ---- And write the statement that we want to write on the file.

oFile << "Hi! "; [WRITE THIS ONLY]

TVT

If you want to write another statement, then you can do that as well.

oFile << "I love to travel. ";

Now our last step will be to close the opened file.

TTT

We will use oFile object and call close function.

oFile.close();

Now here please note that it is considered a good programming practice to always close the file if you have opened it. This will allow the system to clean up memory space that was once utilized by the file which we opened.

So always remember to close the file, if you have opened it. Fine?

Let us run the program.

Let us restore our project explorer.

And after you run your program, you will find a file created named my-note.txt in your project folder.

Now if you run the program for the first time, there is a possibility that you might not see this file over here. So just re-run the program, and then you will find the file for sure here.

Well, it just happened with me a couple of times, so I thought I should tell you too. Fine? So please re-run your program, in case you face the same problem.

Now if you open this file, you will find, we have our text written Hi! I love to travel.

How cool is that? Using C++ language, we are able to create a file and write to it as well.

Now if you want these two statements in two separate lines, then we need to use newline characters.

**BACK TO PROGRAM.**

So here, I can use a new line and here as well let’s use a newline. Well, it truly depends on how you want your text to appear in your file.

Run the program again.

**SHOW FILE**

Great.

**BACK TO PROGRAM**

Now so far we have written only strings as text on our file. But we can write a number, or character, in a similar way as we did in our print statements.

So we can write I am 25 years old. In this way.

oFile << "I am " << 25 << " years old. \n";

Run the program. So here is the output. Great.

So yes that’s all for this video.

**UP NEXT**

Now in the next video, let’s explore how to read from a file.

string myText;

// Read from the text file

ifstream iFile;

iFile.open("my-note.txt");

// Use a while loop together with the getline() function to read the file line by line

**while** (getline(iFile, myText)) {

// Output the text from the file

cout << myText;

}

// Close the file

iFile.close();

---------------------------------------------------

e3: Video - Read from a File

---------------------------------------------------

Let us now proceed and explore how to read data from a file.

**DEMO**

Here, I have updated the code. So make sure to update the code from the link present below the video.

Now, reading from a file is as simple as writing on the file.

**SHOW FILE**

Basically, in this video, we will read the data line by line from this file such as line 1, line 2 and line 3 using a loop and print them one by one in our output console.

**COME BACK TO PROGRAM**

So to read from a file, we must include the fstream header file. Fine?

**SCROLL AND ALIGN CODE**

Then here at the top, I have defined a string which will hold the text present in each line of the file.

Then here, I am creating an input file stream object.

Then I am opening the file. Which means I am making my file ready so that I can read data from it.

Now make sure your file is present in your project folder. Otherwise your code won’t work.

Then we have while loop. Well, this is bit tricky to understand, so for a moment let’s forget that we have a while loop here.

Let’s focus on this getline function. ---- in this function we have the first parameter as an input file stream and second parameter is the string variable which we have defined here.

If you remember, initially we explored how to get string user input using this getline function.

TVT

// getline(cin, str)

That time we used getline function and passed the first parameter as cin, which was used to read the input stream entered by the user. And we stored the user input in second parameter which is a string.

Similarly, here as well, our first parameter is a file input stream, and once we read the data from the file, we will store it in string str. Fine? I hope you are able to recall things which we already explored.

REVERT TVT DELETE getling(cin, str)

Anyways, so this statement, at a time basically reads one line of text from the file.

**SHOW FILE**

Which means for the first time, getline function will read the first line of text present on the file.

**BACK TO PROGRAM.**

So once it reads the first line, it stores that line in the string variable str. Fine?

And then we are printing the text in the output console. How cool is that?

So when these two are executed for the first time,

**SHOW FILE**

it has only read and printed the first line of text in the output console. So we need to move on to the next line, read it and print the next line.

And repeat the cycle one more time for the last line as well. Fine?

**BACK TO PROGRAM**

So to repeat the cycle, we are using while loop.

So this while loop will be executed until all the lines on the file are read and printed in the output console.

So for that here I am using a condition.

**DELETE CONDITION**

First I am using the input file stream object and calling EOF function.

Well, EOF stands for End Of File. This is something you need to remember.

Well, this function returns true, if the end of file is reached. Otherwise, it returns false if it finds a valid text to read from the file.

**SHOW FILE**

i.e. while executing while loop, if we find text in the first line then eof function returns false. For second line as well, it will return false, for third as well it will return false. But when it reach this point, where we don’t have any text or line, then it means we have reached END OF LINE. so when it reaches here, then eof function will return true.

**BACK TO PROGRAM**

So this means to execute the while loop for valid lines, we need to use a NOT Operator.

So I will use an exclamation mark here.

Therefore, this condition as a whole ---- is true until we reach the end of the file.

So in this way, these two statements will read each line and print each line one by one.

Nice and simple.

And finally, we are closing the file.

Perfect.

Let us run the program.

So here is the output.

Here we can print each line in a new line.

So we can put a newline at the end here. And run the program again. Perfect.

So yes, this is how we read data from a file. That’s all for this video.

**UP NEXT**

Up next you will find a simple code challenge based on File handling. Go ahead and do it.

---------------------------------------------------

e4: Video - Summary

---------------------------------------------------

So finally we have reached the end of the topic.

**MM** let us summarize our topic.

**MMM MM**

To handle files in C++, we need to include File Streams header file in our program.

**MM**

We need to use the Output File stream class to write on a file.

**MM**

Next, we use the Input File Stream class object to read from a file.

Moving on,

**MMM**

we used a few functions while dealing with files in this topic.

**MM** The first is the open function, that basically opens the file and makes it ready so that we can write and read from the file.

**MM**

Next, is close close function, which is used to close an opened file. Well, it is very important to close the opened file so as to allow the system to efficiently clean up memory which was once allocated to the opened file.

**MM**

Last but not least, eof function is used to determine the End of File. It returns true if the end of file is reached, otherwise it returns false.

And the full form of EOF is the end of file. So please note that.

So yes that’s pretty much it for this topic.

UP NEXT you will get a small quiz to test your understanding of the topic. All the best.

1. File Streams header file: <fstream>
2. Output File Stream class: ofstream
3. Input File Stream class: ifstream
4. open() function is used to open a file so that we can write and read from the file.
5. close() function is used to close the opened file. This is important to clean up memory.
6. eof() function is used to determine the End Of File.