


# FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

## Lecture33

### Lecture 33: File Handling - Append and other Mathematical Operations

#### Example: Appending to a File in C++

 `is_open()` method checks whether the file has been successfully opened. If the file is open, the program proceeds to write to it.

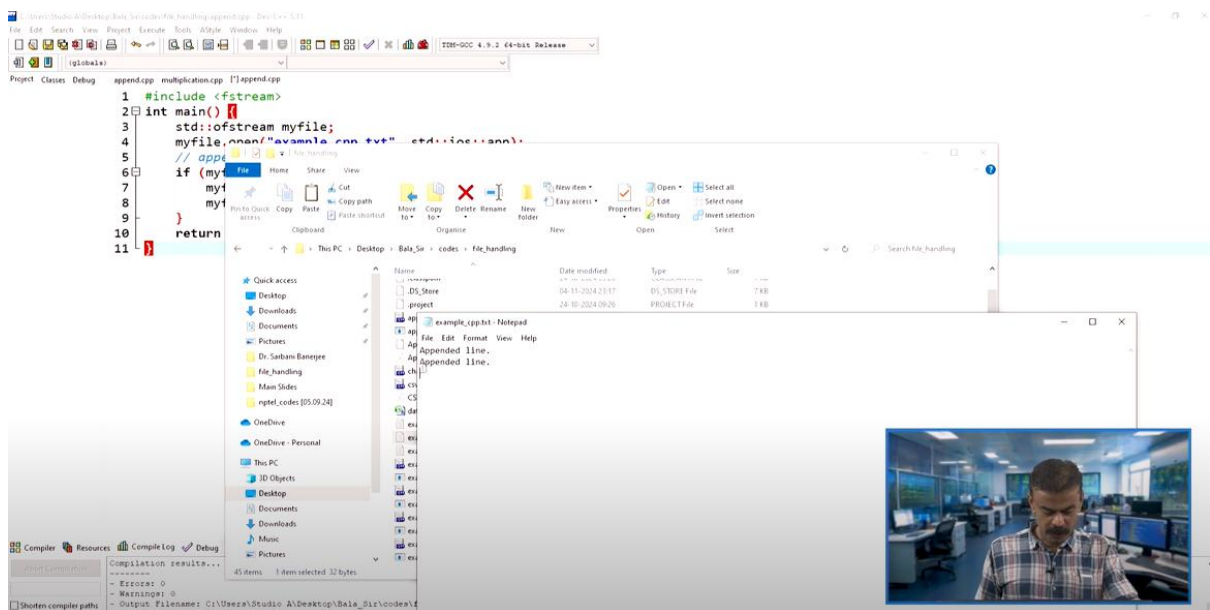
```
1 #include <fstream>
2 int main() {
3     std::ofstream myfile;
4     myfile.open("example.txt", std::ios::app);
5     // append mode
6     if (myfile.is_open()) {
7         myfile << "Appended line.\n";
8         myfile.close();
9     }
10    return 0;
11 }
```



Welcome to Lecture 33: File Handling. So, in the In the last lecture, we saw how to read and write from a file, right? How to write to a file and how to read from a file. So, in today's lecture, I will start with how to append to a file in C++, okay. So, assume that you have a text file, let's say example.txt, and from there, how we are going to Assume that there are three or four lines there, and then here, what we are doing is appending one line, right? An appended line.

Let's see how it works, all right. So, you have file stream, include file stream fstream, and then myfile is an object under output file stream ofstream, and myfile.open, all right? Myfile.open. So, that means we are going to append to example.txt. So, that is the meaning over here. So, myfile is invoking the open member function, right? And then we are going to do the appending operation in example.txt.

So, now, as I said, is\_open, if you look at the start of the slide. So, is\_open is a member function or a method. So, that will be checking whether the file has been successfully opened or not, right? So, if it is open, then the program will proceed. Otherwise, what you can do is use a try-throw-catch, right? If, suppose, you are not able to open it, right.



So, now it is very trivial. We have seen in the last lecture and the last two lectures, in fact, several try-throw-catch options. In fact, we will study them in today's lecture also. So, now if you look, if myfile, which is invoking the underscore open member function, right? If it is successful, then it will print, right? In that file, it will print myfile, right. So, that will print an appended line. So, let us assume we have three or four lines already there.

So, let us see with an example, right? We will go to the program and we will check after the appended line. So, let us see what will happen. So, then you have to close the file. myfile will invoke close. So, it will be closed.

So, after updating this particular, right? Or you can use whatever you want, right? Appended line and then a new line. In fact, the cursor will go to the next line, and then you will see when you open the file now, example.txt. So, you will see the appended line. So, now let us go to the program. So, append.cpp, yes.

So, here we have example\_cpp.txt. Let us open that file. Let us open the file and see what the content of example\_cpp.txt is, right. So, we have an appended line, appended, maybe we will write something else, right. One is an appended line, the second line we will put 'Welcome to file handling,' 'Welcome to,' and the next line 'Welcome to IITR,' right.

So now we will save this file, and what we can do in the program, we have a new appended line. Yes, that is good. So, line number 7. So now we will run the code. Now when you run the code, right, so it is successfully running.

Now we can open that file. So when you open the file, previously we had only three lines. Now you can see the fourth line also, right. So like this, you can append a new line. So, you can open the new line.

## Example: Appending to a File in Java

```
1 import java.io.FileWriter;
2 import java.io.IOException;
3
4 public class AppendToFile {
5     public static void main(String[] args) {
6         try {
7             FileWriter writer = new FileWriter(
8                 "example.txt", true);
9             // append mode
10            writer.write("Appended_line.\n");
11            writer.close();
12        } catch (IOException e) {
13            e.printStackTrace();
14        }
15    }
16 }
```



I hope it is clear for everyone. So, in the case of Java, the same program if you do it in Java. So, here I have used the IOException, right? So, you have the IO file writer. So, the main program, and you can see writer is an object under the class FileWriter, right?

So, you have the constructor FileWriter, and that is going to handle example.txt, right? So, true in the sense if the file exists or not, right? So, it will check. If, in that case, the file does not exist, all right. So, then you will go to the catch block, right?

So, it will throw the exception, and it will go to the catch block, and then you have the error exception, all right. So, now assume that the file exists, OK? So, then the writer will invoke write, and then you are going to have an appended line. So, now once you append the line, and then you are closing the file. Let us quickly go to it. It is almost similar now. I can quickly go into the program because it is not very difficult once you know it in C++.

So, learning Java is not very difficult. So, we will make it a new appended line, all right, okay. And now we will check the file, which file we have, example.txt, we will see what its content is, yeah. So, we have so many here with a blank space also. So, hello world, this is the example file, it includes multiple lines, punctuation, and spaces, there is a blank space, there is no line after a blank space, there is a big blank space, and then a blank line, end of file, okay.

So, now we will see how this works, this is a good example. So, now we run the code, right, once you run the code, it is successfully executing, yes, because we do not have any output. Now, if you look at example.txt, yes, you can see the new appended line, okay, fine. So, this is how you can append a line, right, to a file in C++ as well as Java. So, now we have seen one true, right, in line number 7 and 8, example.txt, and true. So, that is indicating that the file should be opened in append mode, right. So, this is a new thing we are seeing because in the case of writing and reading, we have not seen this. So, the true is indicating that it is in append mode.

So, now we will manipulate some mathematical cases. For example, the addition of two numbers. I am giving one input file that contains 5 and 10, right? Because most of you may be working with AI and ML, you have to handle so much data. And then you have to manipulate the data and get the output. Manipulation, in the sense, you have to write a proper algorithm. And you have to get the output, right? This is what many data scientists are doing. So now, this program will be useful for beginners. So what we do is we will take two integers. And keep it in input.txt, 5, right?

## Example: Appending to a File in Java

- 👉 The second item 'true' of FileWriter, indicates that the file should be opened in append mode.



And then what we are going to do is we are going to multiply two integers. Let us say I am taking 20 and 30, right? So these 20 and 30 will be multiplied, and the resultant will be 600, right? So that will be stored in output.txt. So, you know, this program is a simple program when you write it, but when you are handling the file, I am giving input.txt.

## Example: Multiplying Two Numbers from a File

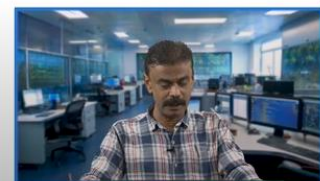
- 📖 In this example, we will demonstrate reading two numbers from an input file, multiplying them, and writing the result to an output file.
- 📖 The example is implemented in both C++ and Java.
- 📖 **Steps:**
  - 📖 Read two integers from 'input.txt'.
  - 📖 Multiply the two integers.
  - 📖 Write the result to 'output.txt'.



So, this program, as I said, is for beginners. So, suppose you are handling, let us say, huge data, you call it big data, right? And you are handling the file, you can use any language, right? And then you do the algorithm. For example, several algorithms are there in machine learning. For example, you want to use feature extraction or classification, and then you have to get the output data. So, the output data you are storing in a different file.

## C++: Reading, Multiplying, and Writing Result

```
1 #include <fstream>
2 #include <iostream>
3
4 int main() {
5     int num1, num2;
6
7     // Step 1: Read two numbers from input.txt
8     std::ifstream inputFile("input.txt");
9     if (inputFile.is_open()) {
10         inputFile >> num1 >> num2;
11         inputFile.close();
12     } else {
13         std::cerr << "Unable to open input file";
14         return 1;
15     }
```



So, this program will be useful particularly for beginners. So, we take two integers, put them in one input dot txt, and multiply these two integers. You have to read both the numbers, and we are going to do the multiplication. After multiplication, you will

get the output, right? That output should be stored in output dot txt. Let us see how we are going to do it. So, here we are including fstream and iostream, number 1 and number 2, num 1 and num 2 are two integer variables, alright.

So, here you have an input file, alright, under ifstream, under the class ifstream, alright. So, it will handle input dot txt. If the input file is invoking the is\_open method or is\_open member function, alright. Right. So, if it is able to open, right, the input file is handling input dot txt. If you are able to open it, then we have stored number 1 and number 2, right.

So, these number 1 and number 2 will be read. That is what line number 10 inputFile number 1 and number 2 will be read, and then you are closing the file, right? inputFile will invoke close. So, suppose you are not able to open it, It will throw an exception. So, the exception will be, right. In fact, if you are not able to open it, it is not an exception.

If and else, simple if and else statement. So if you are not able to open, then cerr, unable to open the file, right, unable to open the input file. So we are continuing. Result is number 1 into number 2, right? Result is multiplication, right?

So, now ofstream, the output file stream, we have done this, right? We are going to write it, right? So, output file is an object and ofstream is the class and then it is handling output dot txt. So, if output file, right? When it is invoking is\_open, if you are able to open, right?

So, you are writing the result. Result is what? Number 1 into number 2. So, then you are closing the file otherwise if you are not able to open the file output dot txt then it will throw the error cerr right unable to open the output file all right. So, then line number 14 you have one cout statement.



## C++: Reading, Multiplying, and Writing Result

```
1 // Step 2: Multiply the numbers
2 int result = num1 * num2;
3
4 // Step 3: Write the result to output.txt
5 std::ofstream outputFile("output.txt");
6 if (outputFile.is_open()) {
7     outputFile << result;
8     outputFile.close();
9 } else {
10     std::cerr << "Unable to open output file";
11     return 1;
12 }
13
14 std::cout << "Result written to output.txt: " <<
15     result << std::endl;
16 return 0;
```



So, this is for our information in the console. So, in the console, it will tell the result to return to output.txt, all right. So, this information we will get the message in the console, all right, and then it will tell. Also, it will tell the result if I cout, you are putting the result, and then endl, the cursor will go to the next line. So, here, even in the console, you are printing what the result is, and then also, this is an optional line, line number 14 is an optional line, all right.

So, but when you are running the code. I mean, it is always better to check whether it is running properly or not, right. So that we can check. So, for that, it will be useful. So, line number 14 will print the result to return to output.txt, and whatever the result, number 1 multiplied by number 2, right, that will be printed, and the next line is nothing but endl, I mean, the new line.

So, return 0. So, let us quickly go to the program. And check this, right. So, multiplication of two numbers, slightly a big program. So, you can see that we have input.txt.

Line number 8, we have input.txt. So we will see what is input.txt. We have two numbers, 10 and 5, right. So I am using two lines. First line I am writing 10, then press enter for 5.

So input.txt. So we have taken a very small data, all right. So slowly I am talking about data, all right. So 10 and 5, 10 in the first line, 5 in the next line. So, I explained all line numbers 8 to 15.

Now, the result is nothing but number 1 multiplied by number 2. So, 10 multiplied by 5 we should get 50, and also I have talked about the output\_CPP. So, here, what will happen? So, we have to just go down in the program, yeah, number 1 multiplied by

number 2, yeah. So, here the result returns to output\_CPP.txt, and then the console will print the result also.

So, when we run the code, because we have seen this code now, I will run the code. Yes, when you compile and run, you can see the result returned to output\_cpp.txt. Also, you are seeing the result 50. So, one number is 10, another number is 5. 10 multiplied by 5 is 50. So, here we are getting, and also we will check output\_cpp.txt. Now we will open it. Yes, you are getting 50 here. So, now what we can do is we will run one more test case, right? Let it be 50, right? We are not disturbing this output.

So, the input file will give 150 or any number. Let us take the input file. Let us consider 100, and here let us take 1729. Fine, yeah, hopefully, it will not go out of the range. Let us save this file and execute it, right. So, in fact, you can see the output 1729 00 And let us open now the output file output\_cpp.txt. So, we will get 172900, right. So, now you can handle the data, right.

So, assume that this is the basic code for the input file and output file. So, now you can do whatever problems you want. So, just go to a slightly more advanced level, right. So, take data of length, let us say, 1000 lines. And then you are manipulating something from that data.

All right. For example, I mean, the whole world is talking about machine learning or deep learning. Right. So in that case, when you are getting the output from somewhere or even the input file, when you are handling a text file, a numerical data file, an image file, or a video file, it's a huge amount of data. Right.

It has numbers. And then you are manipulating like this and you are reading the file. Right, and then you are making some modifications or you are doing some processing, and when you are getting the output, so where is it giving the output? Right? So can we analyze those outputs? Right? So for that, we have introduced basic manipulation on multiplying the two numbers. I hope it is clear for everyone, right? So this is what we have done: input file handling. So, input.txt, we have two integers, and then you have seen those were multiplied in the program, and then the output file, it was stored in output.txt, right?

So we have, in fact, tested with two cases, and then it was working fine. So now, in the case of Java, In the case of Java, what do you have to do? So, you have number 1, let us say 0, number 2, 0, you are assigning. So, java.io.\*, I put all the packages, \* represents, and now you have a try block, all right.



## JAVA: Reading, Multiplying, and Writing Result

```
1 import java.io.*;
2
3 public class MultiplyNumbers {
4     public static void main(String[] args) {
5         int num1 = 0, num2 = 0;
6
7         // Step 1: Read two numbers from input.txt
8         try (BufferedReader reader = new BufferedReader(new
9             FileReader("input.txt"))) {
10             num1 = Integer.parseInt(reader.readLine());
11             num2 = Integer.parseInt(reader.readLine());
12         } catch (IOException e) {
13             System.out.println("Unable to open input file")
14             ;
15             e.printStackTrace();
16         }
17         return;
18     }
19 }
```



So, the reader object under `BufferedReader`, we had seen this in the last class, right? So, you have a memory allocation with a constructor, and then you have a `FileReader`, all right. So, the `FileReader` will handle this `input.txt`. So, now you are taking two inputs from the user, all right. So, it is like in the file you are taking.

So, it is not like your usual one. So, here you are getting `Integer.parseInt`, all right, and then you are reading from the line. Let us say the first line is 10, and the second line you have 5, correct? So, from the `input.txt`, we have the two numbers. So, in case line number 8, you are not able to read that `input.txt` file, or if it is not getting opened.

So, then it will throw an exception. So, it will be handled from line number 12 to 13, and then you are returning it fine. So, let us see how this works. In fact, yeah, the program is continuing, right? Result is equal to result; we have already, yeah, here we are declaring it as also `int`. So, `num1` multiplied by `num2`.

## JAVA: Reading, Multiplying, and Writing Result

```
1      // Step 2: Multiply the numbers
2      int result = num1 * num2;
3
4      // Step 3: Write the result to output.txt
5      try (BufferedWriter writer = new BufferedWriter(new
6              FileWriter("output.txt"))) {
7          writer.write(Integer.toString(result));
8      } catch (IOException e) {
9          System.out.println("Unable to open output file"
10              );
11          e.printStackTrace();
12      }
13
14      System.out.println("Result written to output.txt: "
15          + result);
16  }
```



So, now the output file again you are doing the similar thing, right. So, here you have a `BufferedWriter`. So, there it was a `BufferedReader`. So, here it is a `BufferedWriter` class under this writer is an object, small w, and allocating memory, and you have a constructor `BufferedWriter`, and then `FileWriter` is handling the `output.txt`. So, if you are able to handle this `output.txt`, no problem, it will further continue.

Otherwise, alright, so you have a catch exception. It will say unable to open the output file, alright. Now, line number 6, writer is invoking write. So, write is nothing but integer to string; in fact, the integer is invoking the `toString` method, and then the result, right. So, the result, suppose you are multiplying 10 by 550, right.

So, that will be displayed; the write function will be displayed, and also for our convenience, we are printing like exactly what we have done in C++. So, the result returned to `output.txt` will be printed along with the result, right. So, it is more or less the same as what we have seen in C++, alright, and here the only difference is here when you are using the constructor or creating an object with dynamic memory allocation you have to be a little careful; otherwise, both the programs, alright, will give you the same result. Let us quickly go to the Java console, and we will run the code, right? So here you go, here is the same program as what I have told.

The only thing we will check is the input file, right? `Input.txt`, what is there? We will run two cases: the same 100 and 1729, right. So, these two numbers are there in `input.txt`, or we will change it because we already have the output file with the same result, right. So, here is what we will do: change the first number to 1000 and keep the second number as it is. So, we have to get 1729 followed by three zeros as the output. So, now, for the rest of them, I will explain. Let us quickly run the code. Yes,

it is running, and then you can see 1729 followed by three zeros. What is the output file name? Let us see.

So, here you can see the result returned to output\_java.txt. OK, here it is: output\_java.txt. Correct, output\_java.txt. So, if we see that output\_java.txt. So, we are getting 1729 followed by 000. I hope it is clear for everyone. So, in the case of Java, we can also multiply.

So, when you look at the BufferedReader, which is a class. So, here we have an object reader. So, that is able to read the input.txt. So, it is reading the two integers. We have given 1000 and 1729.

Right. And in case. Right. So, this you can test. So, what you can do.

I mean, do not keep any input.txt in the directory. And then try. You will get the IOException. It is not very difficult. Right.

So, because here we are positively going to see what exactly will happen. And then it will parse the string. The meaning of parsing the string. You have two numbers. Right.

So, 1000 and 1729. All right. So that will be separated. And then these will be useful. Right.

So for multiplying. So one will be number one. Another one will be number two. Right. So that is what?

Parse the strings read from the file as integers. Parsing. This is called parsing. Right. So on the first line, you have 1000.

The second line you have 1729. So these will be parsed. And then output file handling. So the BufferedWriter class. So, that is used to write the result to output.txt.

So, if you are not able to open this output.txt, then you will have the problem, right? That means there will be a try-throw-catch exception thrown. So, then it will be caught, and you can say, 'Unable to open output file,' all right. So, this is how it works in both C++ and Java, all right. Another example: writing formatted data in C++. So, you may require some format, right?

So, even when you are handling, let us say, a spreadsheet. So, you will put, let us say, the student marks, and then you want to do the left alignment, right alignment,

or center alignment, right? So, in a similar way. So, we can fix. So, suppose I want the value, the real value, right.

Assume that you are entering 123.456, but I want only the precision 2, right? Set precision 2. So, in that case, assume that I want 123.46, right. So, these kinds of formats, we can easily handle both in C++ and Java. So, let us take this simple example. So, I have included fstream, file stream.

And then I have included input-output manipulation, IO manipulation. Right. And you have a class called ofstream. Right. You have a class called ofstream.

So, ofstream as the object myfile. Right. Data dot txt. Right. So, it is handling data dot txt.

So if myfile is invoking is\_open, if you are able to open it. Right. If you are able to open it. So it is having this std fixed. Right.

So, that means it is going to fix with the set precision 2, right? It is going to fix with the set precision 2, and myfile will print, let us say, the value, right. So, assume that the value is 123.456. So, in that case, I will get the output 123.46. So, I am doing the set precision 2. If you put 1. So, it may be like 123.5, something like that. Maybe we can try.

So, then you are closing this. So, let us run this code. So, formatted dot CPP explains this same 123.456 we are using. So, let us run the code here. So, data dot txt we have 123.46 value 123.46.

So, we are printing, right? So, we are printing. So, here you can see we have to print. My file is printing the value. The value is in double quotes. So, the value is coming with a colon, and then this 123.456 is becoming 123.46.

So, if you look at line number 6, we are using fixed and then set precision. So, set precision to 2. So, let us change it to, let us say, 1, set precision to 1, so that you will understand better. Now, we run the code. It is successfully running.

Now, we will see data.txt. So, data.txt, if you look, see 123.5. So, it is rounding off 456 to become 123.5. I hope it is clear. So, we will see one more program, writing formatted data.

## Explanation of Java Code

### 📖 Input File Handling:

- 📖 Uses `BufferedReader` to read two integers from 'input.txt'.
- 📖 Catches and handles any `IOException` that occurs during file operations.

### 📖 Processing:

- 📖 Parses the strings read from the file as integers and multiplies them.

### 📖 Output File Handling:

- 📖 Uses `BufferedWriter` to write the result to 'output.txt'.
- 📖 Catches and handles any `IOException` that occurs during file operations.



So, here, as I said, when you are accessing some spreadsheet, you want to have the left alignment, right alignment, or alignment in the middle. So, in that case, right, how are you going to do it? So, in the previous program, we had seen fixed. All right. And then set precision.

So here, you are going to see the left alignment. All right. And set W is nothing but set width. So for this, you have `fstream`, IO manipulation, `iostream`, and `string`. `String` we have used already.

`iostream` we have to start with. So you have `stream` and IO manipulation. Right. So, in fact, in the previous example, we used IO manipulation. In fact, I started with IO manipulation.

So, that means this library is providing the functionalities for manipulating input and output formats. So, you require the format, right? So, for example, I want to have the left alignment, right? So, here in this case, we will see we will take a simple example, right? So, you have an output file which is handling `output.txt` under the class `ofstream`.

If the output file is able to open, that means it is open. Assume that if it is true, then the output file has the left alignment, right? `STD` double colon, right? Or the scope resolution operator `left` is a left alignment. And then you have set the width for 20 characters, right?

## Example: Writing Formatted Data in C++

📖 `<iomanip>` library provides functionalities for manipulating input and output formats.

```
1 #include <fstream>
2 #include <iomanip> // for formatted output
3 int main() {
4     std::ofstream myfile("data.txt");
5     if (myfile.is_open()) {
6         myfile << std::fixed << std::setprecision(2);
7         myfile << "Value:" << 123.456 << "\n";
8         myfile.close();
9     }
10    return 0;
11 }
```



Set the width for 20 characters. So, name 20 characters and set the width for 10 characters. For the score, you are setting the width for 10 characters, right? Right. So, you are doing the first one is allies.

Left setw 20 allies. Right. And setw 10 95 marks. The score is 95 marks. Similarly for Bob.

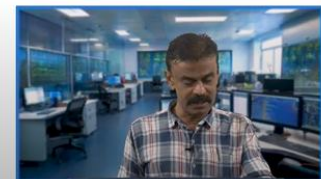
Right. 87. Charlie 78. Right. So then you are closing the file.

If you are not able to open the file. So then it will throw an error. It will tell unable to open output file. Right. So when we run the code.

## C++: Writing Formatted Data

**Task:** Write a list of names and scores to `output.txt`, aligning the data in columns.

```
1 #include <fstream>
2 #include <iomanip>
3 #include <iostream>
4 #include <string>
5
6 int main() {
7     std::ofstream outputFile("output.txt");
8     if (outputFile.is_open()) {
9         outputFile << std::left << std::setw(20) << "Name" << std::setw(10) << "Score" << std::endl;
10        outputFile << std::left << std::setw(20) << "Alice" << std::setw(10) << 95 << std::endl;
11        outputFile << std::left << std::setw(20) << "Bob" << std::setw(10) << 87 << std::endl;
12        outputFile << std::left << std::setw(20) << "Charlie" << std::setw(10) << 78 << std::endl;
13        outputFile.close();
14    } else {
15        std::cerr << "Unable to open output file";
16    }
17    return 0;
18 }
```



We will quickly go into the Java console, right? So, here you go, all right. So, this is the program, yes, that I already explained. Suppose if we run the code, let us see



what the output we are getting is, right? So, it is running successfully, and when you look at output.txt, we are getting this, right? So, the name set width is 20 characters.

So, you can now change it. So, put it at 15. So, put it at 25. Right. So that you can play with it.

And then you can see that. And then you can see the output. I will open the output.txt. Right. So you can see that.

Right. Scores are 95, 87, 78. So here, the width is 10. Right. So, like this.

So, you can manipulate this. You can change it to 20, 25, 15. Something like that. And then, see the output. So now, in the last two programs, we have seen the formatted one.

Right. So, how can you do that? Formatting. So, when you want to have the output data in a particular format, All right.

So you might have practically done that. Right. Some of the spreadsheets, as I said, keep on seeing. So suppose I want to enter certain marks or certain data. So you keep the left alignment or right alignment or even in the document file when you are using, you do the left alignment, right alignment, et cetera.

Right. So the same thing with the help of a program. Is it possible to do that? Yes. When you are handling the file and then, I mean, you have some streams.

Right. How to use the output file stream so I can put the left alignment like a left. Previous case fixed set precision right all these inbuilt you can use and then you can make the output the formatted one right. So, in today's lecture we talked about how to append the line and also the multiplication of two numbers right using the file and then When you are getting the output so can I do the formatted one right so that we had seen.

So, with this, I am concluding this lecture. Thank you.