

SEARCH

RESOURCES

CONCEPTS

1. Intro
2. CODE: Write and Run Your First C...
3. Compiled Languages vs Scripted L...
4. C++ Output and Language Basics
5. CODE: Send Output to the Console
6. How to Store Data
7. Bjarne Introduces C++ Types
8. Primitive Variable Types
9. What is a Vector?
10. C++ Vectors
11. C++ Comments
12. Using Auto
13. CODE: Store a Grid in Your Progr...
14. Getting Ready for Printing
15. Working with Vectors
16. For Loops
17. Functions
18. CODE: Print the Board
19. If Statements and While Loops
20. Reading from a File
21. CODE: Read the Board from a File
22. Processing Strings
23. Adding Data to a Vector
24. CODE: Parse Lines from the File
25. CODE: Use the ParseLine Function
26. Formatting the Printed Board
27. CODE: Formatting the Printed Bo...
28. CODE: Store the Board using the ...
29. Great Work!

File Input Streams

Creating an Input Stream Object

In C++, you can use the `std::ifstream` object to handle input file streams. To do this, you will need to include the header file that provides the file streaming classes: `<fstream>`.

Once the `<fstream>` header is included, a new input stream object can be declared and initialized using a file path `path`:

```
std::ifstream my_file;  
my_file.open(path);
```

Alternatively, the declaration and initialization can be done in a single line as follows:

```
std::ifstream my_file(path);
```

C++ `ifstream` objects can also be used as a boolean to check if the stream has been created successfully. If the stream were to initialize successfully, then the `ifstream` object would evaluate to `true`. If there were to be an error opening the file or some other error creating the stream, then the `ifstream` object would evaluate to `false`.

The following cell creates an input stream from the file `"files/1.board"`:

```
In [ ]: #include <fstream>  
#include <iostream>  
#include <string>  
  
int main()  
{  
    std::ifstream my_file;  
    my_file.open("files/1.board");  
    if (my_file) {  
        std::cout << "The file stream has been created!" << '\n';  
    }  
}
```

Compile & Execute

Explain

Loading terminal (id_7qs09au), please wait...

Reading Data from the Stream

If the input file stream object has been successfully created, the lines of the input stream can be read using the `getline` method. In the cell below, a while loop has been added to the previous example to get each line from the stream and print it to the console.

```
In [ ]: #include <fstream>  
#include <iostream>  
#include <string>  
  
int main() {  
    std::ifstream my_file;  
    my_file.open("files/1.board");  
    if (my_file) {  
        std::cout << "The file stream has been created!" << '\n';  
        std::string line;  
        while (getline(my_file, line)) {  
            std::cout << line << '\n';  
        }  
    }  
}
```

Compile & Execute

Explain

Loading terminal (id_6gag5b0), please wait...

Recap

That's it! To recap, there are essentially four steps to reading a file:

1. Include `<fstream>`
2. Create a `std::ifstream` object using the path to your file.
3. Evaluate the `std::ifstream` object as a `bool` to ensure that the stream creation did not fail.
4. Use a `while` loop with `getline` to write file lines to a string.

On to an Exercise

Have a careful look at the code above for reading file lines into the code. In the next exercise, you will write a function to do this in your program.