

Lesson 2:
Introduction to the C++ Language

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!

Processing Strings

SEND FEEDBACK

Streaming ints from a string with istringstream

In C++ strings can be streamed into temporary variables, similarly to how files can be streamed into strings. Streaming a string allows us to work with each character individually.

One way to stream a string is to use an input string stream object `istringstream` from the `<sstream>` header.

Once an `istringstream` object has been created, parts of the string can be streamed and stored using the "extraction operator": `>>`. The extraction operator will read until whitespace is reached or until the stream fails. Execute the following code to see how this works:

```
In [ ]: #include <iostream>
#include <sstream>
#include <string>

using std::istringstream;
using std::string;
using std::cout;

int main ()
{
    string a("1 2 3");

    istringstream my_stream(a);

    int n;
    my_stream >> n;
    cout << n << "\n";
}
```

Compile & Execute

Explain

Loading terminal (id_8kne4px), please wait...

The `istringstream` object can also be used as a boolean to determine if the last extraction operation failed - this happens if there wasn't any more of the string to stream, for example. If the stream still has more characters, you are able to stream again. See the following code for an example of using the `istringstream` this way:

```
In [ ]: #include <iostream>
#include <sstream>
#include <string>

using std::istringstream;
using std::string;
using std::cout;

int main()
{
    string a("1 2 3");

    istringstream my_stream(a);

    int n;

    // Testing to see if the stream was successful and printing results.
    while (my_stream) {
        my_stream >> n;
        if (my_stream) {
            cout << "That stream was successful: " << n << "\n";
        }
        else {
            cout << "That stream was NOT successful!" << "\n";
        }
    }
}
```

Compile & Execute

Explain

Loading terminal (id_wdhrmkw0), please wait...

The extraction operator `>>` writes the stream to the variable on the right of the operator and returns the `istringstream` object, so the entire expression `my_stream >> n` is an `istringstream` object and can be used as a boolean! Because of this, a common way to use `istringstream` is to use the entire extraction expression in a while loop as follows:

```
In [ ]: #include <iostream>
#include <sstream>
#include <string>

using std::istringstream;
using std::string;
using std::cout;

int main () {
    string a("1 2 3");

    istringstream my_stream(a);

    int n;

    while (my_stream >> n) {
        cout << "That stream was successful: " << n << "\n";
    }
    cout << "The stream has failed." << "\n";
}
```

Compile & Execute

Explain

Loading terminal (id_q8np8yh), please wait...

Strings with Mixed Types

In the stream example above, the string contained only whitespaces and characters which could be converted to `int`s. If the string has mixed types, more care is needed to process the string. In the following example, the type `char` is used, which is a type that can hold only a single ASCII character.

```
In [ ]: #include <iostream>
#include <sstream>
#include <string>

using std::istringstream;
using std::string;
using std::cout;

int main()
{
    string b("1,2,3");

    istringstream my_stream(b);

    char c;
    int n;

    while (my_stream >> n >> c) {
        cout << "That stream was successful:" << n << " " << c << "\n";
    }
    cout << "The stream has failed." << "\n";
}
```

Compile & Execute

Explain

Loading terminal (id_g5k09ja), please wait...

In that example, notice that the `3` was not printed! The expression:

```
my_stream >> n >> c
```

tried to stream an `int` followed by a `char`. Since there was no `char` after the `3`, the stream failed and the `while` loop exited.

Loading [MathJax]/extensions/Safe.js

Menu Shrink

NEXT