

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!

Formatting the Printed Board

SEND FEEDBACK

Enums

C++ allows you to define a custom type which has values limited to a specific range you list or "enumerate". This custom type is called an "enum".

Suppose you were writing a program that stores information about each user's car, including the color. You could define a `Color` enum in your program, with a fixed range of all the acceptable values:

- white
- black
- blue
- red

This way, you can be sure that each color is restricted to the acceptable set of values.

Here is an example:

In []:

```
#include <iostream>
using std::cout;

int main()
{
    // Create the enum Color with fixed values.
    enum class Color {white, black, blue, red};

    // Create a Color variable and set it to Color::blue.
    Color my_color = Color::blue;

    // Test to see if my car is red.
    if (my_color == Color::red) {
        cout << "The color of my car is red!" << "\n";
    } else {
        cout << "The color of my car is not red." << "\n";
    }
}
```

Compile & Execute

Explain

Loading terminal (id_6jgm71y), please wait...

Note: In the example above, the keyword `enum` is followed by the keyword `class` and then the class name `Color`. This creates what are called "scoped" enums. It is also possible, but [not advisable](https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md#Renum-class) (https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md#Renum-class), to omit the `class` keyword and thus create "unscoped" enums. [More information is available at cppreference.com](https://en.cppreference.com/w/cpp/language/enum) (https://en.cppreference.com/w/cpp/language/enum).

Example with a switch

Below is another example of an enum being used. Here, a custom type `Direction` is created with four possible values: `kUp`, `kDown`, `kLeft`, `kRight`. One of these values is then stored in a variable and used in the switch statement.

In []:

```
#include <iostream>
using std::cout;

int main()
{
    enum class Direction {kUp, kDown, kLeft, kRight};

    Direction a = Direction::kUp;

    switch (a) {
        case Direction::kUp : cout << "Going up!" << "\n";
            break;
        case Direction::kDown : cout << "Going down!" << "\n";
            break;
        case Direction::kLeft : cout << "Going left!" << "\n";
            break;
        case Direction::kRight : cout << "Going right!" << "\n";
            break;
    }
}
```

Compile & Execute

Explain

Loading terminal (id_9w0djk), please wait...

When you feel like you have understood the example above, try modifying the code to test different values, or define another enum

Loading [MathJax]/extensions/Safe.js

Menu

Shrink

NEXT