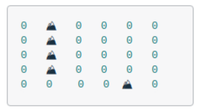
Formatting the Printed Board



In the previous exercises, you stored and printed the board as a `vector<vector<int>>`, where only two states were used for each cell: `0` and `1`. This is a great way to get started, but as the program becomes more complicated, there will be more than two possible states for each cell. Additionally, it would be nice to print the board in a way that clearly indicates open areas and obstacles, just as the board is printed above.

To do this clearly in your code, you will learn about and use something called an `enum`. An `enum`, short for enumerator, is a way to define a type in C++ with values that are restricted to a fixed range. For an explanation and examples, see the notebook below.

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

↑ Menu Expand

On to an Exercise

In the next exercise, you will start the process of storing the board using a custom `enum` type. To get started with this process, you will write some code to convert the custom type values to strings for printing.