

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!

Great Work!

https://video.udacity-data.com/topher/2019/September/5d71ad21_c-nd-c1-l02-outro/c-nd-c1-l02-outro_720p.mp4

Summary

Great work! You now have a C++ program which reads board data, stores the board in your program, and prints the board with nice formatting. You are ready to begin writing the A* search function.

Here is the outline summarizing what you have learned in this lesson:

Lesson Recap

- Compile and run a simple C++ program
- Send output to the terminal
- Variables and containers
 - Variable types
 - Vectors
 - auto
- Functions and control structures
 - Conditionals
 - Loops
 - Functions
- Data Input
 - Read data from a file
 - Parse data and process strings
- Define your own types with enums

Review

At this point, your program should look like the program below. You have built this short program from scratch, so now is great time to review the code to ensure you recall how each part of it works.

```
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
using std::cout;
using std::ifstream;
using std::istringstream;
using std::string;
using std::vector;

enum class State {kEmpty, kObstacle};

vector<State> ParseLine(string line) {
    istringstream sline(line);
    int n;
    char c;
    vector<State> row;
    while (sline >> n >> c && c != ',') {
        if (n == 0) {
            row.push_back(State::kEmpty);
        } else {
            row.push_back(State::kObstacle);
        }
    }
    return row;
}

vector<vector<State>> ReadBoardFile(string path) {
    ifstream myfile (path);
    vector<vector<State>> board{};
    if (myfile) {
        string line;
        while (getline(myfile, line)) {
            vector<State> row = ParseLine(line);
            board.push_back(row);
        }
    }
    return board;
}

string CellString(State cell) {
    switch(cell) {
        case State::kObstacle: return "⚠ ";
        default: return "0 ";
    }
}

void PrintBoard(const vector<vector<State>> board) {
    for (int i = 0; i < board.size(); i++) {
        for (int j = 0; j < board[i].size(); j++) {
            cout << CellString(board[i][j]);
        }
        cout << "\n";
    }
}

int main() {
    auto board = ReadBoardFile("1.board");
    PrintBoard(board);
}
```