Build an OpenStreetMap Route Planner

REVIEW | CODE REVIEW 2 | HISTORY

Meets Specifications

Congratulations on successfully completing this project. The way entire algorithm was implemented as well as the precautions that were taken to ensure that this submission runs with success is quite commendable.

Some resources for further reading in your free time:

- A great video on Pointers: What Are Pointers? (C++)
- A quick overview of Object Oriented Programming in C++
- C++ docs
- C++17 features
- Vector and its attributes
- Unordered map

Keep learning and stay udacious!

Compiling and Testing

✓ **The project code must compile without errors using `cmake` and `make`.**

✓ Good Job! The code compiles correctly using `cmake` and `make` without any errors. 💯

✓ **Code must pass tests that are built with the `./test` executable from the `build` directory of the project. See the project submission instructions for more details on how to run the tests.**

✓ The code passes tests built with `cmake_tests ..` commands from the `build` directory. Great work!

User Input

✓ **A user running the project should be able to input values between 0 and 100 for the start x, start y, end x, and end y coordinates of the search, and the project should find a path between the points.**

✓ Excellent! Your implementation of the route planning successfully finds a path between two points specified on the map. I tested it with few points and I must say it is robust

✓ **The coordinate (0, 0) should roughly correspond with the lower left corner of the map, and (100, 100) with the upper right.**

Note that for some inputs, the nodes might be slightly off the edges of the map, and this is fine.

✓ The start and end points are mapped correctly from the relative coordinates.

Code Efficiency

✓ **Your code does not need to sacrifice comprehension, stability, or robustness for speed. However, you should maintain good and efficient coding practices when writing your functions.**

Here are some things to avoid. This is not a complete list, but there are a few examples of inefficiencies.

- **Running the exact same calculation repeatedly when you can run it once, store the value and then reuse the value later.**
- **Loops that run too many times.**
- **Creating unnecessarily complex data structures when simpler structures work equivalently.**
- **Unnecessary control flow checks.**

✓ Your implementations completely stays clear of unnecessary complex structures, avoids loops

that run repeatedly, avoids running exact calculations severally and also unnecessary control flow checks are avoided.
Good Job !!!

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS

RETURN TO PATH