

Lesson 3:
A* Search

SEARCH

RESOURCES

CONCEPTS

1. Intro

2. Motion Planning

3. Maze

4. Maze 2

5. Coding the Shortest Path Algorithm

6. A* Search

7. Lesson Code Structure

8. CODE: Starting A* Search

9. CODE: Writing the A* Heuristic

10. Pass by Reference in C++

11. CODE: Adding Nodes to the Ope...

12. CODE: Initialize the Open Vector

13. CODE: Create a Comparison Fun...

14. CODE: Write a While Loop for the...

15. CODE: Check for Valid Neighbors

16. Constants

17. CODE: Expand the A* Search to ...

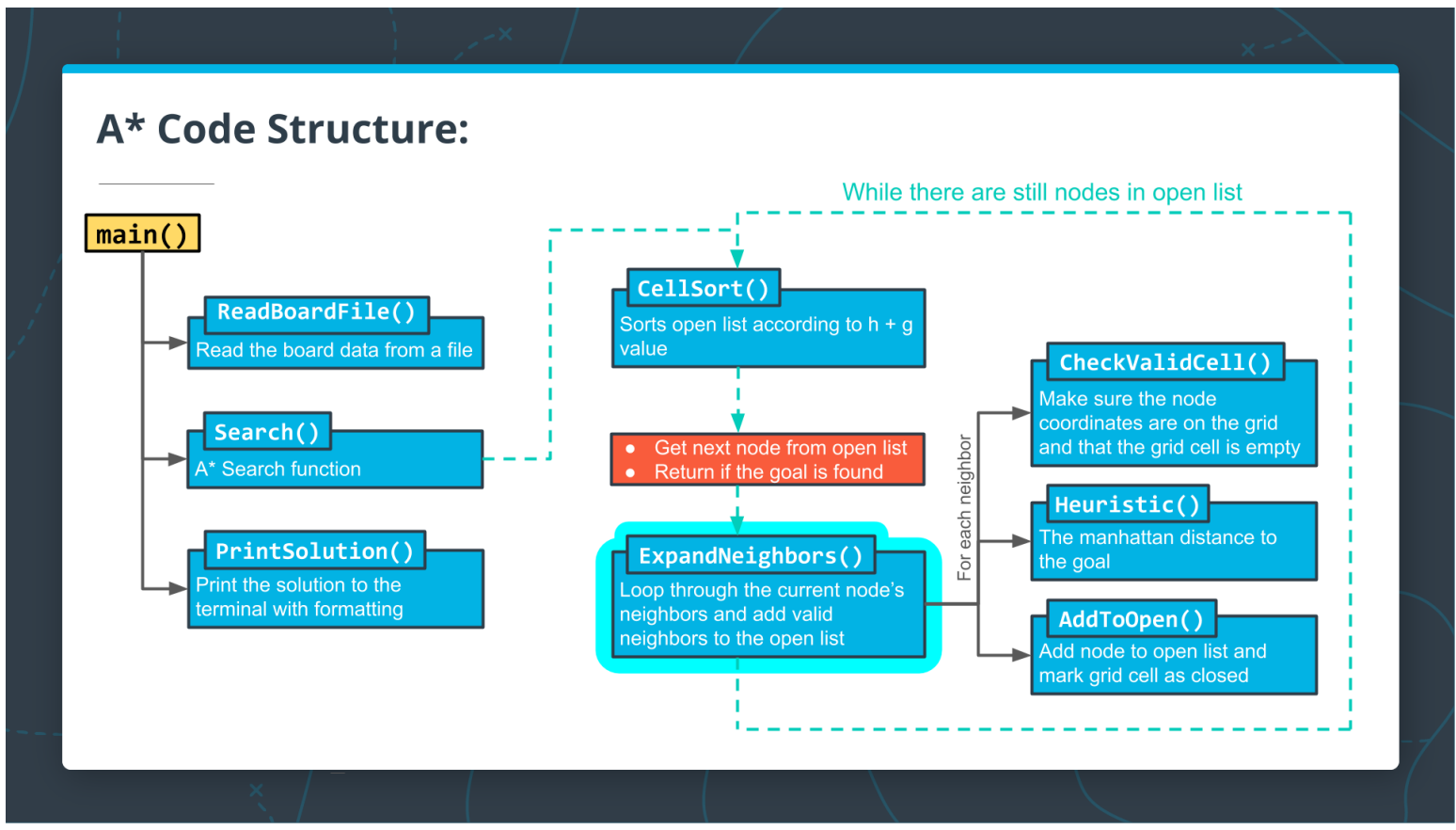
18. Arrays

19. CODE: Adding a Start and End to ...

20. Congratulations!!

21. How to Become More Proficient ...

Expand the A* Search to Neighbors



Writing the `ExpandNeighbors()` function

You have now reached the final step of the A* algorithm! You are ready to expand your A* search to neighboring nodes and add valid neighbors to the open vector. In this exercise, you will write an `ExpandNeighbors` function that takes care of this functionality for you.

To Complete This Exercise:

Write a `void ExpandNeighbors` function that accepts references to the following:

- The current node,
- the open vector,
- the grid, and
- an int array for the goal coordinates.

The `ExpandNeighbors` function should implement the functionality given in the pseudocode below:

```
// TODO: ExpandNeighbors {  
  
    // TODO: Get current node's data.  
  
    // TODO: Loop through current node's potential neighbors.  
  
    // TODO: Check that the potential neighbor's x2 and y2 values are on the grid and not closed.  
  
    // TODO: Increment g value, compute h value, and add neighbor to open list.  
  
} // TODO: End function
```

Note: we have provided directional deltas in the form of a 2D `array`. An array is a C++ container much like a vector, although without the ability to change size after initialization. Arrays can be accessed and iterated over just as vectors.

In the exercise, you can iterate over these `delta` values to check the neighbors in each direction:

```
// directional deltas  
const int delta[4][2]={{-1, 0}, {0, -1}, {1, 0}, {0, 1}};
```

< +

> home > workspace

1.board

main.cpp

solution.cpp

test.cpp

main.cpp

```
1 * #include <algorithm> // for sort  
2 #include <iostream>  
3 #include <fstream>  
4 #include <sstream>  
5 #include <string>  
6 #include <vector>  
7 using std::cout;  
8 using std::ifstream;  
9 using std::istringstream;  
10 using std::sort;  
11 using std::string;  
12 using std::vector;  
13  
14 const int delta[4][2] = {{-1, 0}, {0, -1}, {1, 0}, {0, 1}};
```

+ BASH

```
root@e5fec8d5be4:/home/workspace#
```

↑ Menu

↗ Expand