## CONCEPTS ✓ 1. Intro

SEARCH

RESOURCES

2. Motion Planning

Lesson 3:

A\* Search

- ☑ 3. Maze
- ✓ 4. Maze 2

5. Coding the Shortest Path Algorithm

✓ 8. CODE: Starting A\* Search

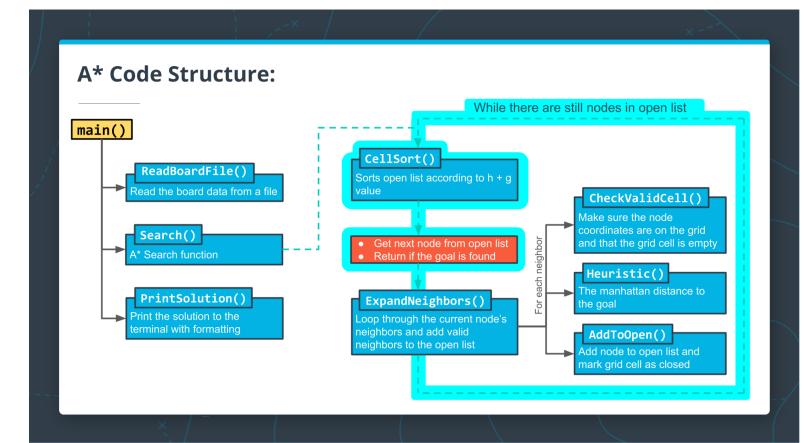
☑ 7. Lesson Code Structure

- 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 th...
- 15. CODE: Check for Valid Neighbors
- 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 ...

## Write a While Loop for the A\* Algorithm



Writing the while loop for the search

Great work so far! Now on to some of the core functionality of the A\* search algorithm. A\* search works by sorting the open list using the f-value, and using the node with the lowest f-value as the next node in the search. This process continues until the goal node has been found or the open list runs out of nodes to use for searching.

In this exercise, you will implement the primary while loop in the algorithm which carries out the process described above:

## To Complete This Exercise:

Complete all of the TODOs in the pseudocode below. These are also marked directly in the exercise code.

```
// TODO: while open vector is non empty {
   // TODO: Sort the open list using `CellSort`, and get the current node.

// TODO: Get the x and y values from the current node,
   // and set grid[x][y] to kPath.

// TODO: Check if you've reached the goal. If so, return grid.

// If we're not done, expand search to current node's neighbors. This step will be completed in a late
   // ExpandNeighbors

//} // TODO: End while loop
```

## **Note:** We've included a header and a function to sort the open vector:

- #include <algorithm>
- std::sort
- CellSort

The CellSort function uses the Compare function you wrote previously to determine the sorting order. The CellSort function contains two operators that you haven't seen before: \* and ->. These operators have to do with C++ pointers, which you will learn about in the next lesson. Don't worry about them for now!

