Some Coding Suggestions

1. Realize that writing code is mostly debugging and testing!

2. Use naming conventions and be consistent

3. Use function names and variable names that make sense and are understandable by others (not only you!).

4. Writing clean code:
   makes others understand your code,
   makes you understand your own code,
   and reduces possibility of introducing bugs!

5. Solve a simple version of the problem first and then go for the "actual" problem.

6. Use the console for debugging.

7. Use the debugger for debugging.

8. Every now and then compile in release mode to make sure you don't have a bug that is not caught in debug mode. The larger a project, the harder it gets to find a bug that shows up in release mode.

9. Always assume someone else will end up having to look in your code and find his/her way around, because that will happen.

10. Do NOT hack. A hack is a solution that is not correct and does not handle all the possible cases.
    If you have to, make sure to put a tag like //HACK: so when someone looks in the code they can make additional considerations

11. Do not leak memory

12. Solve a problem first and then optimize the solution.

13. Have about 1 to 10 commenting ratio. More when needed, less when not needed

14. Use "malloc" with "free" and "new" with "delete."

15. Use factories to recycle objects and avoid unnecessary creation and deletion. Don't always allocate things on the stack or always on the heap. Know the difference.

16. Avoid nested for loops when possible.

17. Use macros carefully when they will improve code, but remember that overuse can bloat code.

18. Avoid unnecessary dependencies! A path planner should not depend on a GUI!

19. Always know whether or not the function you are using is part of the ANSI C standard. You never know when you'll need to port your code!

20. Make your projects have reasonable directory structure. It should be easy to remove the source files and distribute the project.

21. Avoid global functions and especially global variables when possible. If you do use them, make sure you know what "extern" does.

22. Be careful about statically allocated variables/objects. They stay around during the entire run of the app!

23. Be careful about dynamically allocated memory; it is costly to allocate/free.

24. When possible avoid over loaded operators. They tend to reduce readability.

25. Use libraries like STL for prototyping and then hand craft your data structures if they prove to be a bottleneck.

26. Don't assume others' code is perfect. Use it and check when needed.

27. Don't be afraid to use extra parentheses. Too few of them can send you looking around for a long time, whereas extra ones are ignored during compile time.

28. Use white space to make your code easier to read.

29. Avoid negative logic. For example, use (A && B) rather than !(!A || !B).

30. When possible, try to write intuitive code. It makes it easier for you and others to see what is going on.

31. It is often safer to initialize all variables in the constructor (to NULL, 0, etc.)

32. Know bitwise operators like: & | ~ >> << ^

33. Add debugging masks/flags to your code

34. Get used to going to other code sources for help. Often times, the best solution to your problem is very similar to some tutorial or article, but not the exact problem. Learn to use these to guide you on your way through challenges.