SOURCE CONTROL STUFF

FIGURES 1-3

FIGURE 5

SETTING UP TESTING FOR C++

FIGURE 6

A cursory examination of the source code showed a major missing component for any refactoring: unit testing. Testing provides a means through which we can confirm that any refactoring done to the project does not affect the overall functionality and usability of the project. By providing a test suite to run against the various methods in the project, we can quickly and efficiently narrow down any errors in the code.

FIGURES 7-9

Given the expansive, naïve nature of the source code, we created tests with visions of how we would like the code to be organized, rather than tests that would work with the current source code. As a result, we created several new objects and methods that we would like to see in the refactored version of the code. As a result, however, this test suite cannot be compiled until these new objects and methods are written. To begin, we created methods for InitializeBoards(), SetupBoards(), Fire(), and CheckWin().

FIGURE 10

FIGURE 11

Next, we refactored the code to use a multi-dimensional 8x8 array of Booleans, rather than the array of integers currently being used. This refactoring improves the performance and execution time of the code and better models the real-world example of a Battleship game. A Message string, which would hold output to the console, was also created.

FIGURE 12

C++ allows a user to include a header file to be used in testing, which holds a class that includes all its variables and methods. Unfortunately, the source code had not created a class for the application, so we created a Battleship class to hold all the variables and methods.

FIGURE 13

In the process of creating our tests, we also decided that it would be beneficial to have a method that would print the contents of the boards. Such a method would allow the user to easily view the contents of either board at the request of the user. As such, we created a PrintBoards() method.

FIGURE 16

With these three major changes, the test suite can now compile. All that remained was to add the ‘Battleship.cpp’ file to the test project. Running the tests should produce the failed outputs shown below. We purposefully fail these tests to model the “red, green, refactor” method of refactoring and test writing.

FIGURES 17-19

References

<https://www.cprogramming.com/cgi-bin/source/source.cgi?action=Category&CID=2>

<https://msdn.microsoft.com/en-us/library/jj620919.aspx>

<https://blogs.msdn.microsoft.com/vcblog/2017/04/19/cpp-testing-in-visual-studi/>

<http://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/>

<https://answers.yahoo.com/question/index?qid=20110315131034AA08bPe>

<https://stackoverflow.com/questions/10274162/how-to-find-2d-array-size-in-c>

https://stackoverflow.com/questions/19886397/how-to-solve-the-error-lnk2019-unresolved-external-symbol-function