Dev Log Battleship

By Nick Rodgers

Description of the solution:

My version of battleship isn't the cleanest code-wise out there, but I am pleased with my results, coming in at 400 lines of code. This game is menu based, playable by 2 people, is easy to use and prints out 2 boards per player. I started with the Printboard functions. I took the rudimentary tic-tac-toe printboard function that we learned in a lab one week, and expanded it to print a full size, 10x10 battleship board for 2 players. This proved fairly simple, as enumeration and a switch case made printing the boards fairly easy. Putting the numbers on the top and side as coordinates was a little more challenging, but it works now.

Next came the initialize function which was also very simple, and I made it so that it would fill the arrays with "water" without me having to initialize every single 2D array. The Setships function came next, which proved very challenging, but once I made the direction print the ships the correct way, by iterating through a row or column using a counter, everything else fell into place. I made it so that the player can only place 4 ships, and only one of each because during testing I realized it was too easy to cheat and fill the board with only patrols or only carriers or the like.

After this came the Playeraketurn function, which was not too complicated, requiring only a check to see what value each board had in it, and then replacing that with the appropriate value. Checkvictory came second to last, once everything else was in place, and was the final piece in the game until I realized that during testing, I was simply doing the same ship setup over and over again to make testing fast, but during gameplay the players would have to remember every coordinate they put in and whether it was a hit or a miss. Thus, I created the Printupperboard functions, to show the player where they had hit and where they had missed, just like the real game has. With this, I tested everything and it worked as it should, and finally Battleship was complete.

Functions in the program and what they all do:

Many of the functions I describe are going to be for Player 1 and 2, because they are pretty much identical

Menu:

Prints out a title screen, scans a number from the user and using a switch case statement decides whether to start the game, read the rules, or quit the program. Calls the Rules function if 2 is pressed, and the Reset function if 1 is pressed.

Rules:

Prints the rules of the game, and waits for any button to be pressed before returning to the menu.

Printboardplayer:

Clears the screen to get rid of the menu text, and declares 4 variables. 2 of these are two go through the 2D arrays that are the boards, the other two decide when to print new lines to make the board square. Uses a switch case statement that reads the enumerated value in the board and prints the appropriate symbol to the screen, which is necessary for the user to see what is going on.

Calls the Printupperboard function to make sure that both upper and lower boards are printed each time they need to be printed.

Printupperboard:

At its core the same as Printboardplayer, with one fewer case in the switch case statement, the FULL case, because the hit/miss board each player has should not print ships. The purpose of this function is simply to do that, print the hit/miss board for each player so that they know where they have attempted so far.

Playertaketurn:

This reads a coordinate from the user and then puts it into the other player's board, checks to see what the enumerated type is there, and replaces the value accordingly. It uses two variables that become the coordinates that the user enters, and the switch case affects both the other player's board and the current player's hit/miss board. Calls the Printboardplayer function to print the upper and lower board, and then when it finishes it calls the playertaketurn function of the next player, but not before calling the Checkvictory function to see if the player has won yet.

Reset:

Initializes all 4 boards with an i and j variable and two for loops that shift through every array, setting each entry to EMPTY. Calls the Setshipsplayer1 function to start the game.

Setshipsplayer

Calls Printboardplayer to print a blank board so that the player can see where they can put their ships, creates 8 variables, and initializes 5 of them. The reason for this many variables, and what they do are as follows: i,j are for shifting through the array, count iterating through the array when the ships are places, putting them horizontally or vertically, loop is to determine how many ships the player can still place, and c,b,d,p are the Carrier, Battleship, Destroyer and Patrol variables, that tell the user which ships are still available for placement. While not strictly necessary, it ensures that only one of each ship can be places and helps the player remember which ships they've placed so far. The two large if/else statements are for determining which ships are left, as well as printing the ships in the right direction. The player enters 4 numbers: the size of the ship, the X coordinate, the Y coordinate, and the direction they want the ship to be placed, horizontally or vertically. Calls the second players Setshipsplayer function after player 1 has finished.

Checkvictory:

Goes through both players upper boards and checks for the number of hits, if the number of hits is 10 it declares that player the winner and goes back to the menu, where the players can start again or quit if they want to.

Main:

Calls the menu function.