Project Two

Title

Battleship

Course

CSC-17C

Due Date

12/4/2022

Author

Austin Wroblos

Introduction

For project two I made a few adjustments on my battleship game from project one. For this project I already knew what I wanted to do for a few of the requirements before I had started. I didn’t change the flow of the game at all, but I did change the backend logic of a few of the functions. On top of the logic adjustments, I added a shot tracker that tracks all positions that each player shot at. Another idea I had was to have the players’ actual name displayed rather than referring to them as player one and player two.

Development Process

I thought it would be a cool idea to change the way the coordinate inputs were validated. So, I set up a graph to hold all the real coordinate values and checked if the given input was inside the graph. It’s an interesting way to utilize a graph but it did end up working well. I improved upon the player names by having the user enter their name into a hash table and referring to them by that name for the duration of the game. I didn’t have to really add anything for recursion since I had already implemented it in many places in my program. Lastly, I added a record of all the locations that were shot at and put them into a tree to print them out at the end of the game. I used AVL balancing on the tree before printing out the values at the end of the game. Lastly, I removed the stacks and queue’s part for this project since the player names were stored in a hash table now.

Game Rules

The rules are simple, two separate players are given a board and a total of five ships that vary in length. Once you have all the required pieces to start the game you both begin by placing your ships on a board secretly so the other player cannot the see the position of your ships. Once both players have finished placing there ships the attack process will begin. The attack process is performed by stating a position on your opponent’s map and shooting a missile there to hit a ship. If a ship is in the position of the coordinate called, then the other player must tell the player what ship has been hit by the attack. Once all positions on a ship have been hit that ship is now sunk and is no longer in play. This attack process is repeated until either one of the players sinks all of the opposing players ships. If a player sinks all opposing ships they are deemed the winner of that game. Whenever a ship is hit we will mark it on the board with an X and whenever we miss a shot we will mark it on the board as O. The official rule book can also be found [Here](https://www.hasbro.com/common/instruct/battleship.pdf).

Code Description

Player class:

Variables:

1.pMap-Holds position of players ships

2.stat-Holds the players stats

3.ships-Holds all the data about the ships

4.shots-Holds the places that the players has shot at

Text

Description automatically generated

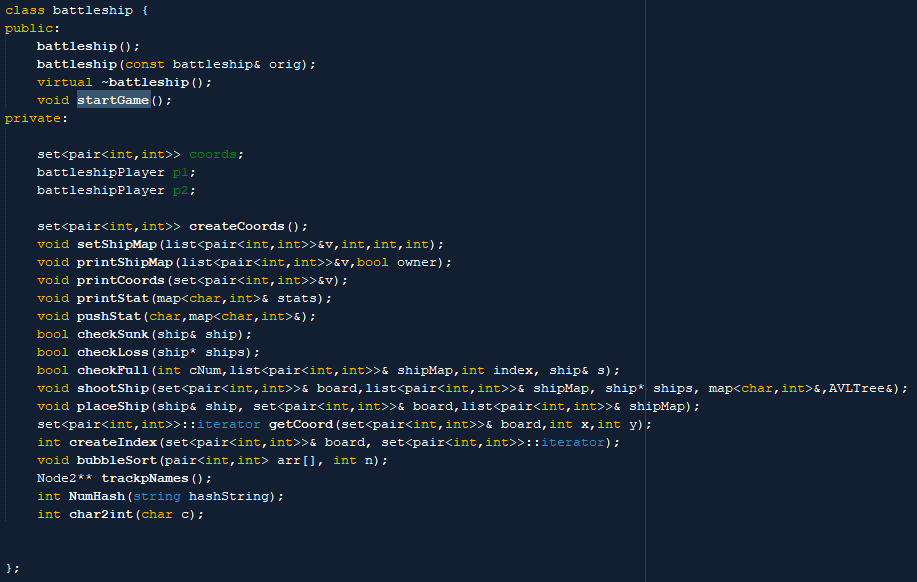
Game Class:

Variables:

1.coords-holds all coords on the board

2.p1- Holds all player one info

3.p2- Holds all player two info



Ship Structure:

Variables:

1.number - Holds the ship #

2.size - Holds the size of the ship

3.pegs - Bool array stating which pegs have been hit

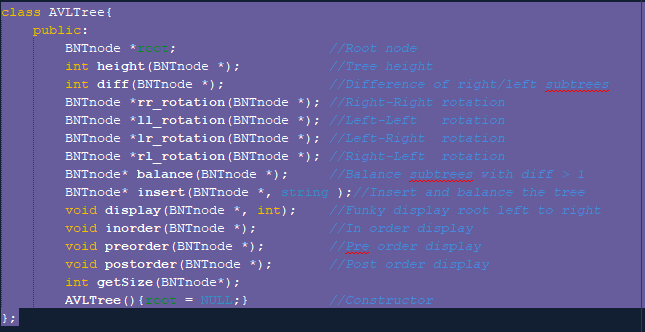
4.sunk - Bool stating whether the ship has sunk or not

A picture containing shape

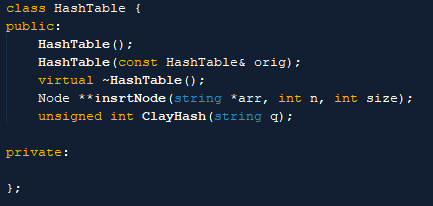
Description automatically generated

AVLTree Class:

1.root – Holds root node of Binary tree



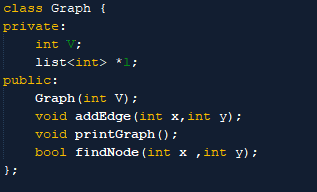
HashTable Class:



Graph Class:

1.V-Vertex Count.

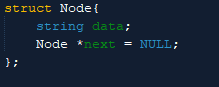
2.\*l-Nods connected to vertices.



Node Struct:

1.data-Holds data for that node

2.next-Links to next node in list.

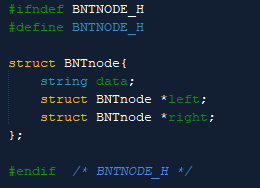


BNTnode:

1.Data-Holds data for that node

2.left-Links to left child

3.right-Links to right child



Sample Input/Output

Notes: Most sample input images were taken before changes were made from project one so player names aren’t tracked. In the actual program it works as planned.

Game Start:

Objective: Enter player names

Result: Once player names have been entered the ship placement process begins

A picture containing text

Description automatically generated

Edge Cases:

Objective: Place ship on edge of board

Result: Gets caught by checkText

Description automatically generated

Objective: Place ship where another ship is placed

Result: Gets caught by check

Text

Description automatically generated

Objective: Place ship off board

Result: Gets caught by check

Text

Description automatically generated

Ship placement:

Objective: Place all ships in coords.

Results: Display Map of ships

Background pattern

Description automatically generated

Shooting:

Objective: Enter Coordinate to shoot at and hit target

Results: Displays which ship you hit and set it on the map with a X

Text

Description automatically generated with medium confidence

Objective: Enter Coordinate to shoot at and missing target

Results: Displays which ship you hit and set it on the map with a O

A picture containing calendar

Description automatically generated

Objective: Shoot all pegs on a ship.

Result: Notify player that ship has been sunk

Background pattern

Description automatically generated

Win Game:

Objective: Sink all opposing ships.

Result: Display winner and match stats

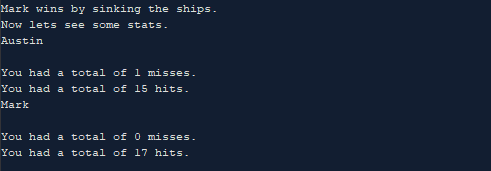
Text

Description automatically generated

Updated Samples:

Names:

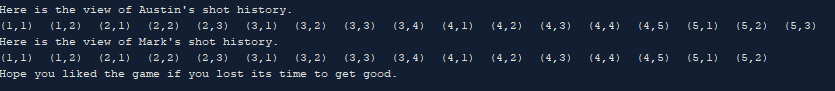
Stats with names:



Placing ships with names:



Shooting Ships with names: 

Tracking Player Shots with Tree:

CheckList:

Graphs:

1.Graphs Class

2.battleship.cpp-288

3. battleship.cpp -433

Trees:

1.battleship.cpp-482

2.battleship.cpp 182-192

Recursion:

1.Recursion can be found all over my program specifically in the placeShip and shootShip function in the battleship class.

Hashing:

1.battleship.cpp 126-203

2.HashTable Class