Project One

Title

Battleship

Course

CSC-17C

Due Date

11/6/2022

Author

Austin Wroblos

Introduction

For project one I decided to go ahead and make the well know game battleship. In CSC-17A I thought about making battleship but rather instead I made blackjack. I thought it would be fun for this class to go ahead and give battleship a try. This project took me just over two weeks to do with a total of 720 lines. It consists of tow two classes and a structure. For the two classes I have a game class and a player class. The player class holds each player’s ship map, ships, and stats. The game class holds the board coordinates and all the driver code for the game. I made a structure to hold all the ship information inside of it such as its length and number. You can find my project at any time on GitHub [Here](https://github.com/aw2884463/CSC-17C_Repo).

Development Process

In my early versions I began by writing directly with the new stl concepts learned over the past couple of weeks and it was getting a bit challenging, so I had to scrap that idea. So around version three my approach changes juristically, I thought it would be best to write the game with vectors to start so I could understand how I wanted the game to run. I had a major issue with tuples being immutable which made me switch my approach to the ships completely. This forced me to make my ships a structure which made the process so much easier. Once I got the game logic down it was time to convert all the vectors to other stl containers. For the most part I switched the vectors into list since they aren’t too different from each other. Once I converted everything from vectors I added the players stat tracker and began making everything into a class. Once I made the battleship class I noticed that it would be cleaner for me to make another player class so I wouldn’t have duplicate sets of variables for each player.

Game Rules

The rules are fairly 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 can not 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 opponents 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 will we mark it on the board with and 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

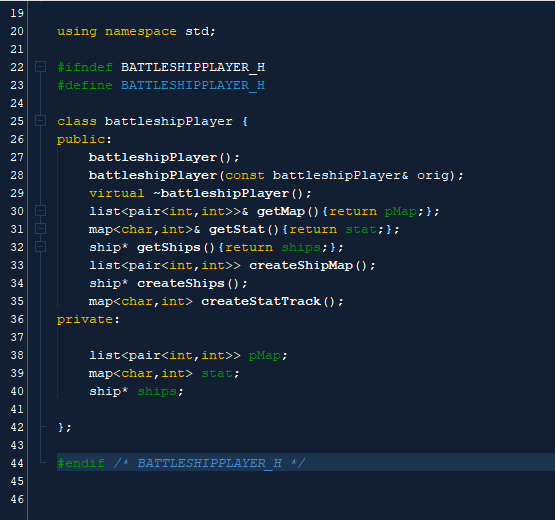
Player class:

Variables:

1.pMap-Holds position of players ships

2.stat-Holds the players stats

3.ships-Holds all the data about the ships



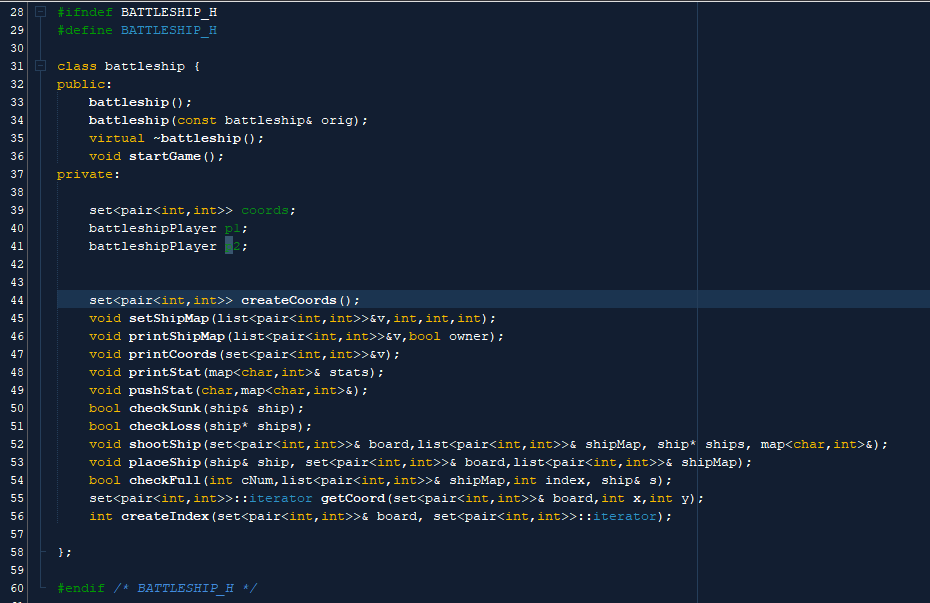
Game Class:

Variables:

1.coords-Holds the board coordinates.

2.p1-Holds player ones data

3.p2-Holds players twos data



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



Sample Input/Output

Game Start:

Objective: Enter player names

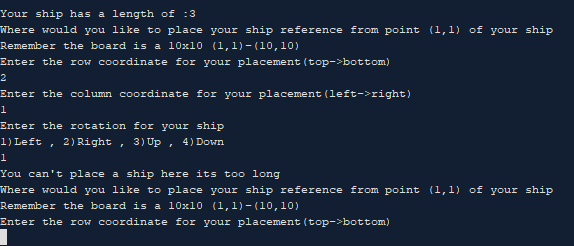
Result: Once player names have been entered the ship placement process begins

A picture containing text

Description automatically generated

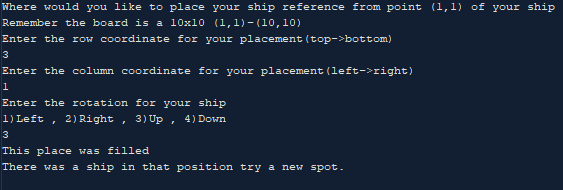
Edge Cases:

Objective: Place ship on edge of board

Result: Gets caught by check

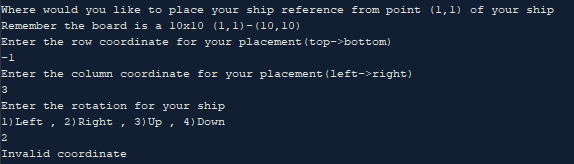
Objective: Place ship where another ship is placed

Result: Gets caught by check



Objective: Place ship off board

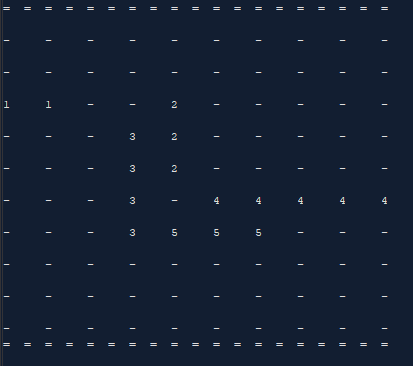
Result: Gets caught by check



Ship placement:

Objective: Place all ships in coords.

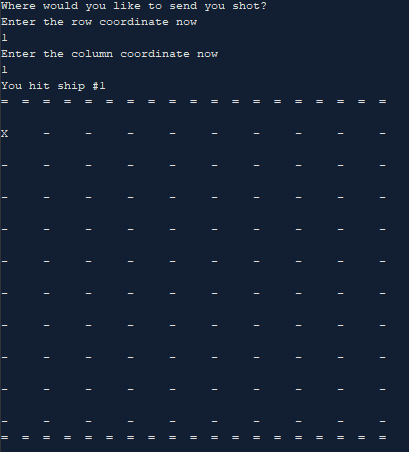
Results: Display Map of ships



Shooting:

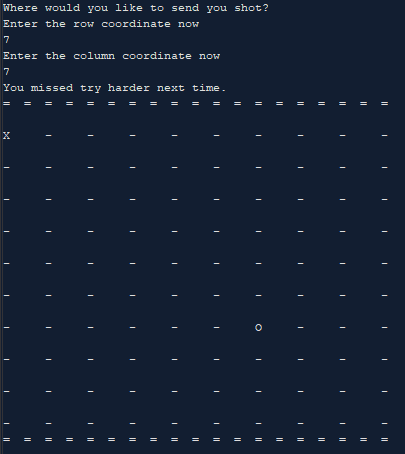
Objective: Enter Coordinate to shoot at and hit target

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



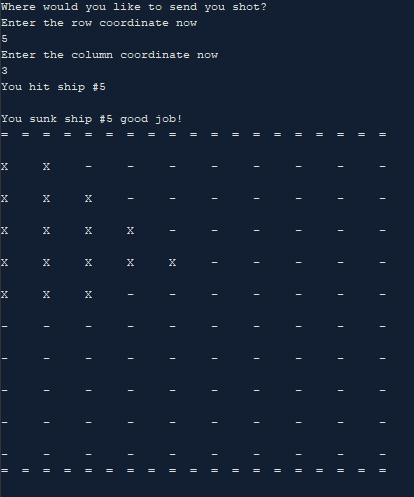
Objective: Enter Coordinate to shoot at and missing target

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



Objective: Shoot all pegs on a ship.

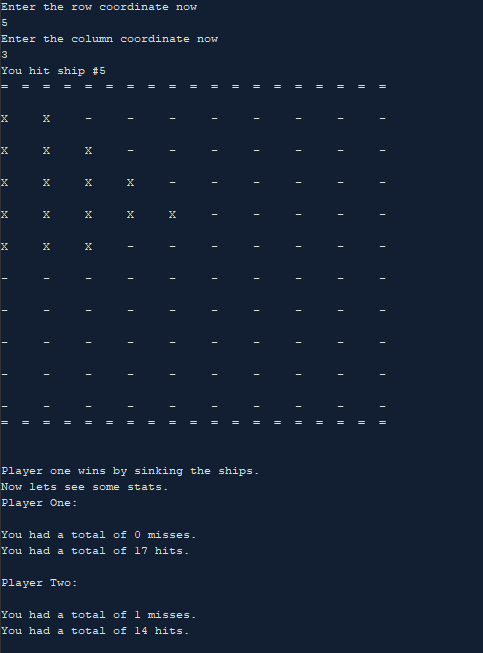
Result: Notify player that ship has been sunk



Win Game:

Objective: Sink all opposing ships.

Result: Display winner and match stats



Checklist

Containers

1.Sequences

**List**- They are used all over the program but the main use for them is to hold the ship map values which. The ship map holds the location of the players ships.

“battleshipPlayer.cpp”- Line 35-Creation

2.Associative Containers

**Sets** -I use a set in my program to hold the coordinates of the system.

“battleship.cpp”- Line 421-Creation

**Maps**-I use a map in my program to hold the stats of each player

“battleshipPlayer.cpp”- Line 27-Creation

3.Container Adaptors

**Stack**-I use a stack to read the names of all past players from a file.

“battleship.cpp”-Line 51-Creation

**Queue**-I use a queue to store the names of the current players to a file.

“battleship.cpp”-Line 33-Creation

Iterators

**Output iterators**- I used them to print the coordinates and to print out a players ship map.

“battleship.cpp”-Line 440,480-Printing Values

**Input iterators**- I used them to assign values to the players ship map when shooting and placing the ships

“battleship.cpp”-Line 217,432-Assigning Values

**Algorithms**

1.non-Mutating

Find- I used find in order to get the location of the coordinate inside of a set.

“battleship.cpp”-Line 346-Get iterator to get distance from beginning

2.Mutating

Fill-I used the fill algorithm to fill the players ship map with pairs

“battleshipPlayer.cpp”-Line 35-Creating ship map

3.Organization