**Project 1**

**<Battleship: The Board Game>**

**CIS-17A 48290**

**Name: Baca, Daniel**

**Date: 11/01/2022**

**Introduction**

Title: Battleship - The Board Game

The board game, Battleship, has been around since 1967, inspired by the events of the time with World War I and II still lingering in the minds of the people, the game started off as a pencil and paper game that children would play at their leisure. Over the course of the past 50 years, the game has seen countless adaptations and new ways to play. My first introduction with the game was with the modern Hasbro set of the classic board game, given to me and my brother as a gift for Christmas at a young age. Since then, I have had fond memories of the game and when given the opportunity to recreate the game from scratch using only the power of modern-day C++ programming I stood up to the challenge.

**Summary**

Project size: 870 + Lines

The number of variables: 24

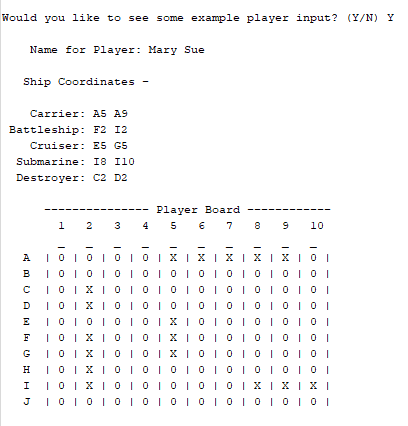
Overall, I am satisfied with how the entire project has turned out. It certainly was a challenge at the beginning but as time went on more and more of the pieces for how the game worked just fell into place and started to flow pretty well. I was not able to hit every topic on the checklist, but I did hit a majority of them and managed to implement each of them in a way I felt like made sense. The entire project took me more or less two full weeks, working on different versions and iterating on what would eventually become the complete thing. It took a good amount of time simply establishing the foundation for the way the game would work, but with time I managed to get there piece by piece and eventually create a fully functional game of Battleship.

**Description**

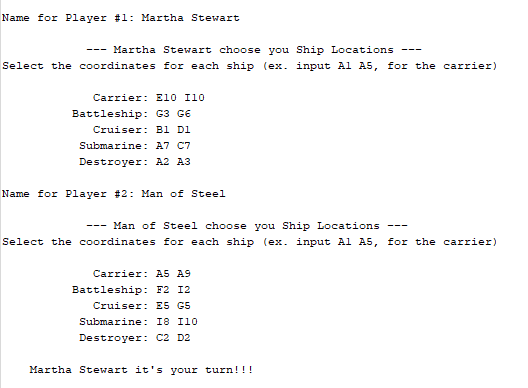
The layout of the program is somewhat simple once unfolded, the program begins by using a function called, game\_start(), in this function both players can input their names, enter their ship’s coordinates and also see example inputs for each portion. The function then returns to main where the function ,coor\_to\_num(), converts the input of ship coordinates from each player into usable number coordinates to use for the program and the rest of the game. The next function, numToboard(), takes these recently converted number coordinates and then uses them to accurately place each of the ships onto the player’s boards. From here the program will loop between four functions, disp\_board1(), shooting1(), disp\_board2(), and shooting2() ,essentially having each player keep shooting each other’s boards, displaying to the players their updated boards and then indicating where and what shots have been taken. During the course of this loop, with every players turn their moves will be recorded in two separate Text Files using the prntTofile() function. Eventually either shooting1() or shooting2() will come back with a flag to indicate that the game has ended, causing the loop to break. From this point the game is effectively over, but there are still two remaining functions, prntToBinFile() and BinFileToprnt(). The players are asked if they would like a recap of information based on the game that was just played, if they would like to see it then these two functions will work to output the desired information.

**Sample Inputs / Outputs**

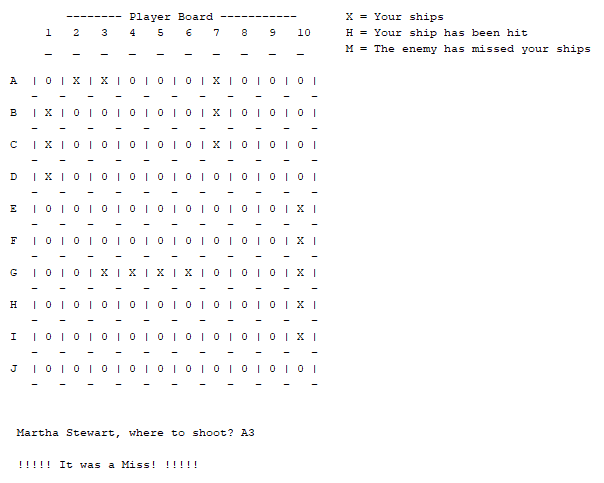
When you first start the game it will prompt the player, asking if they would like to see some example input for how to input their name and ship coordinates, the image below shows exactly this.



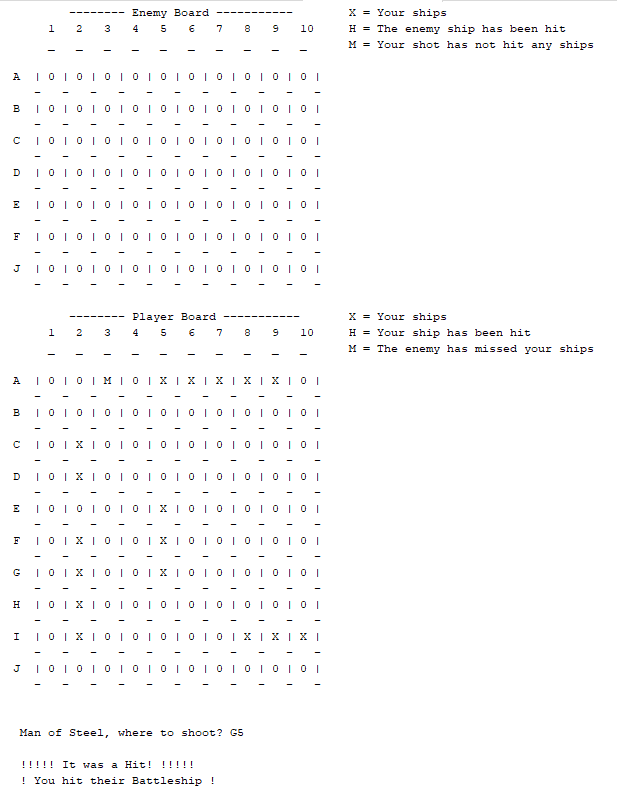
After the user is prompted about the example input, the two players can then input their names and coordinates. Names are written normally using the keyboard, but ship coordinates must be in the format, capital letter and number, corresponding to the way Battleship is normally played on a grid with the rows and columns being assigned letters and numbers. Example inputs are show in the following image.



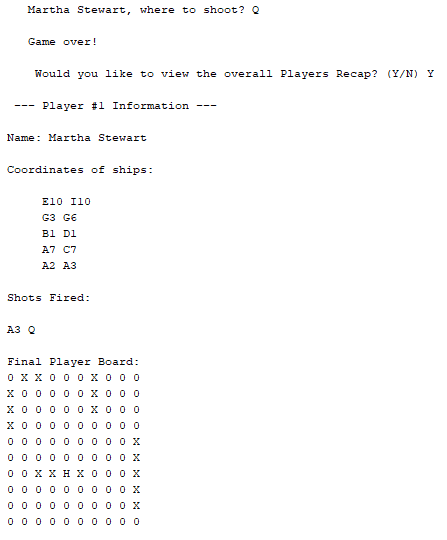
This image shows the act of the individual user shooting a shot at the enemy and then missing.

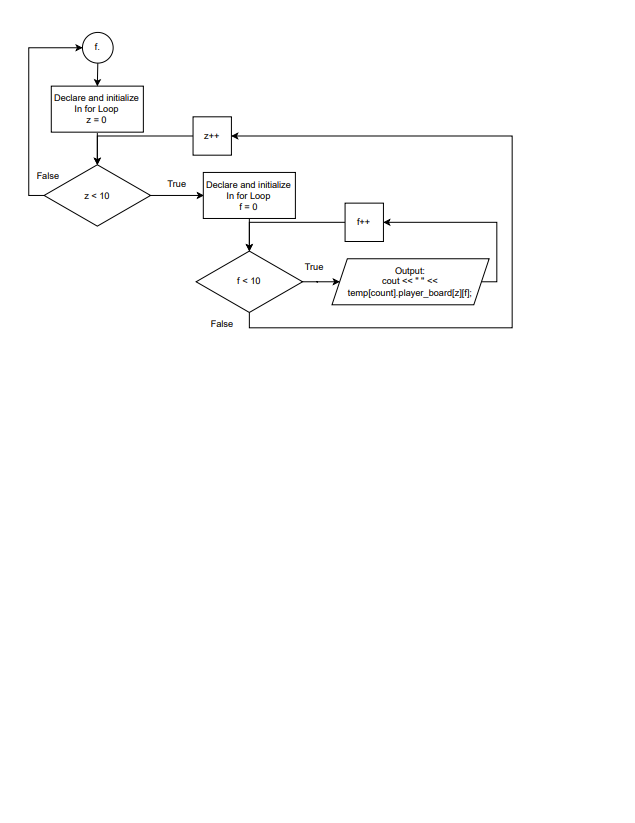
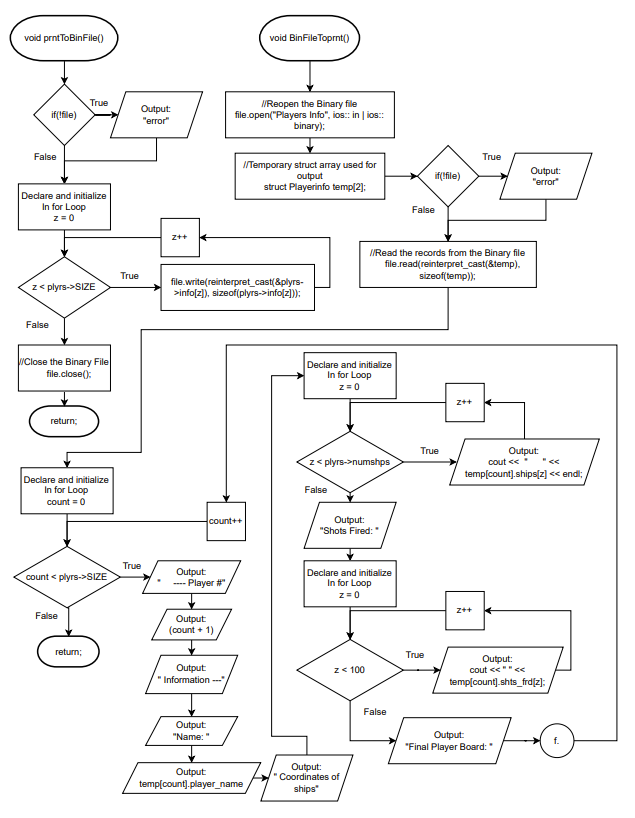
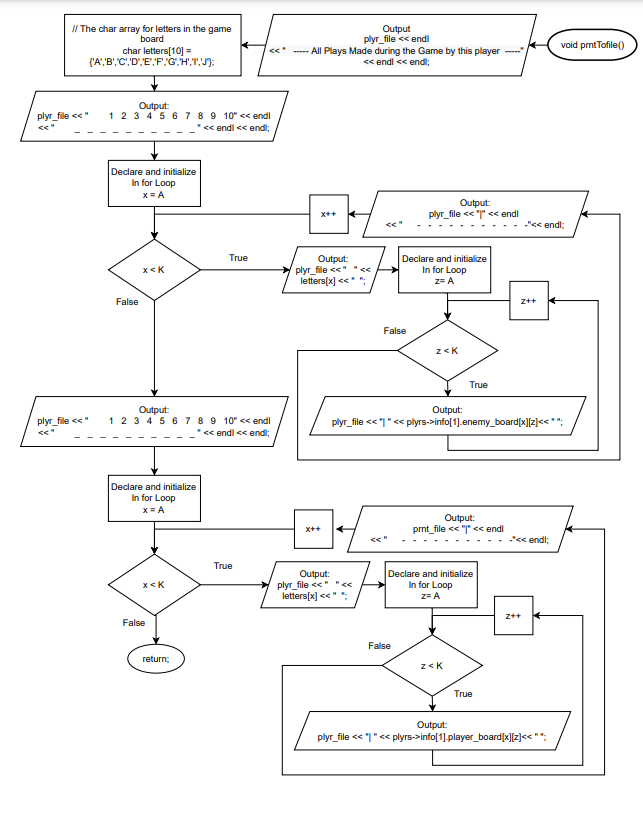
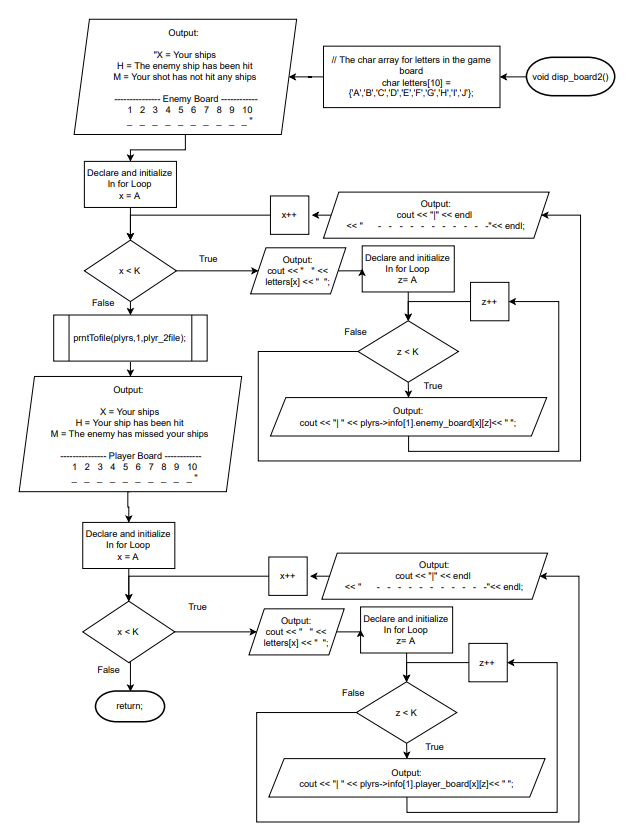
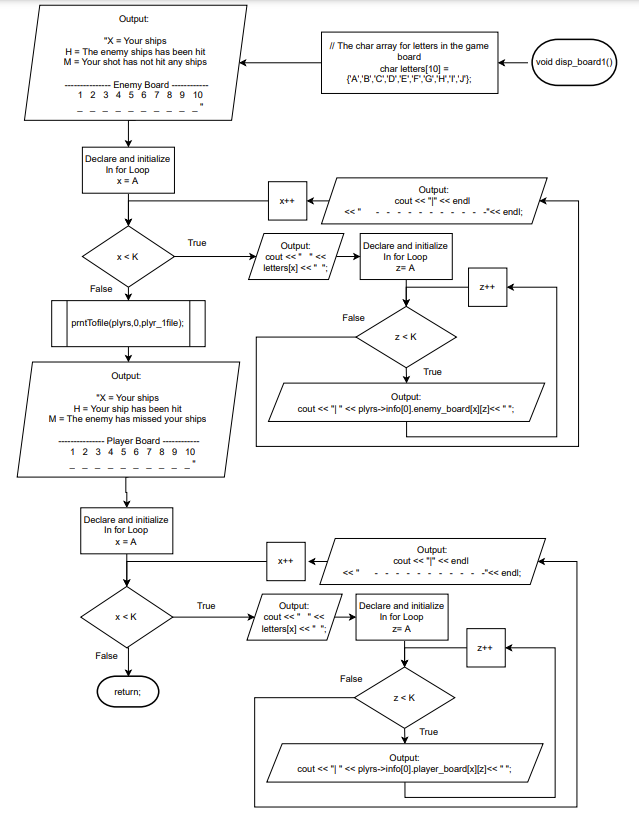
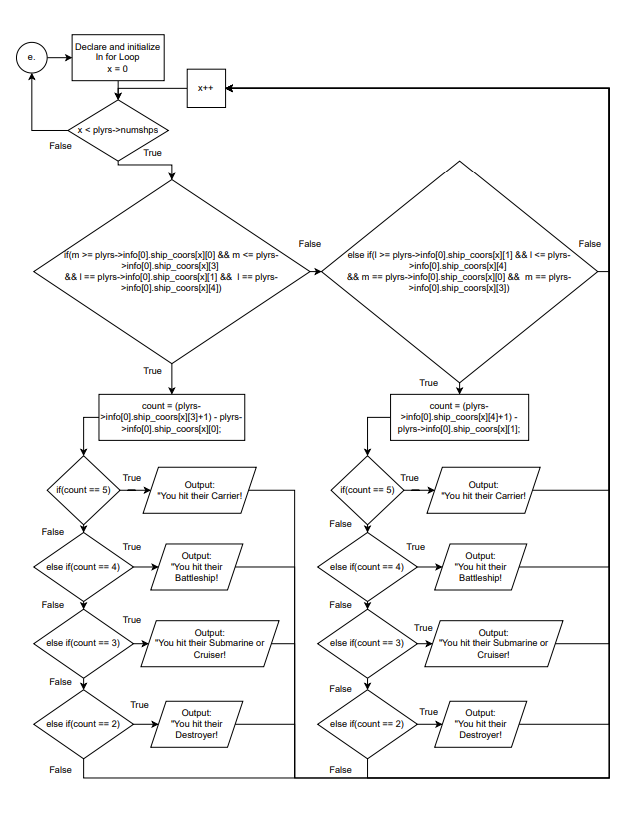
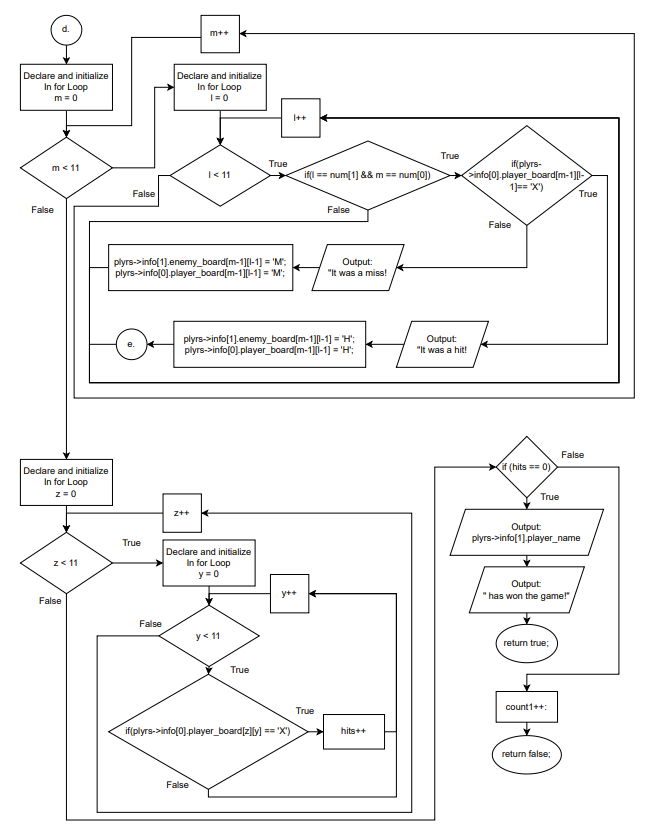
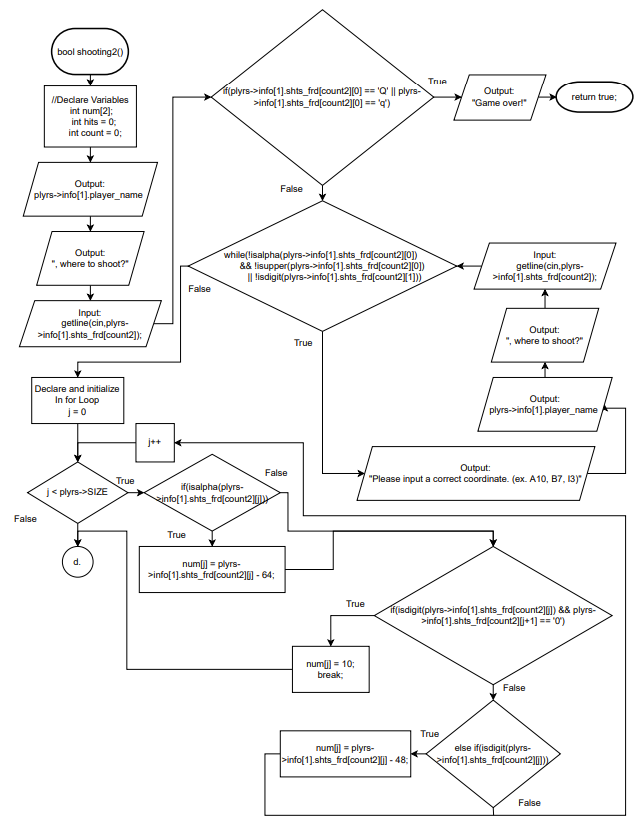
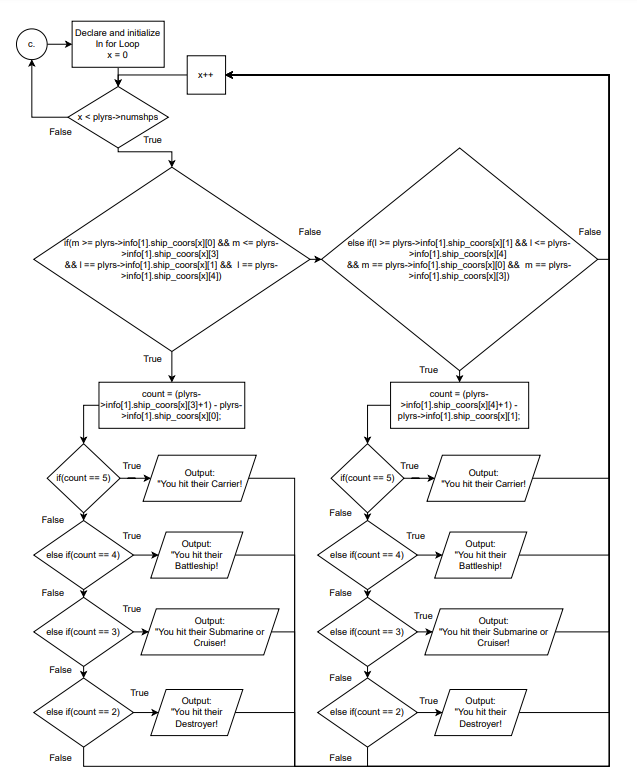
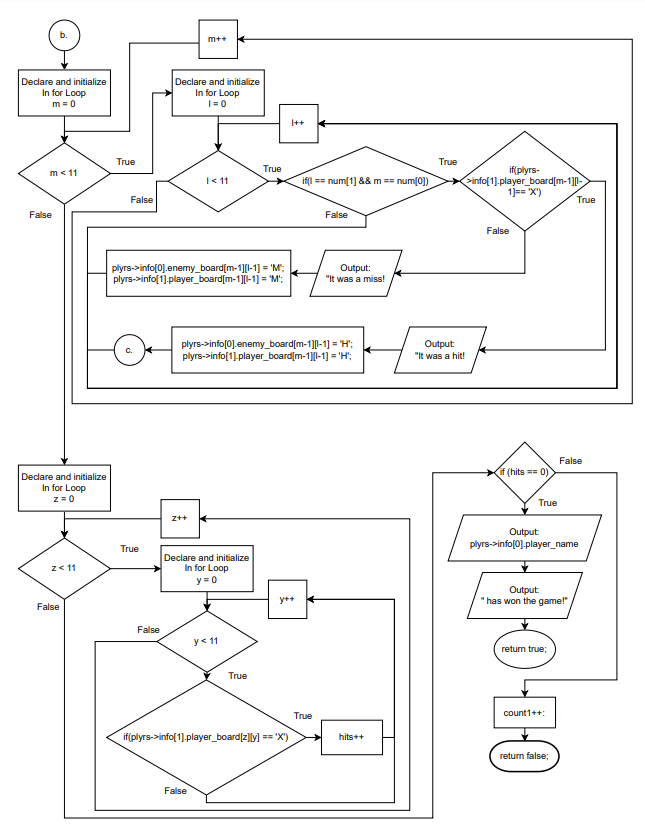
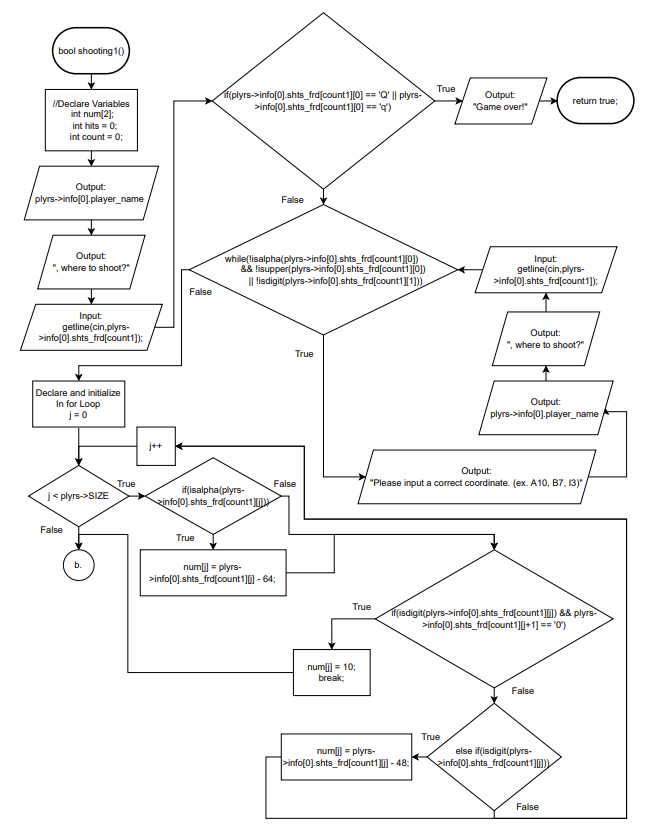
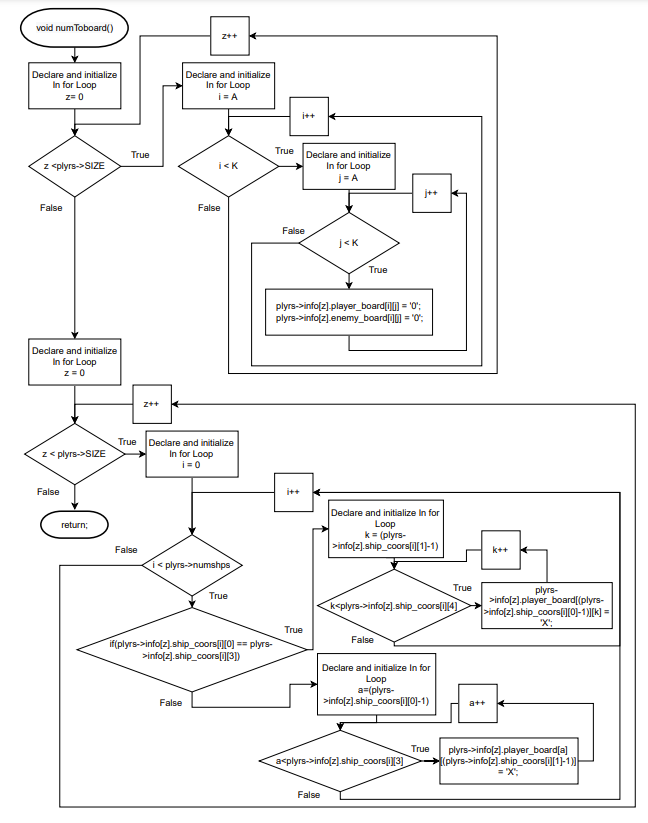
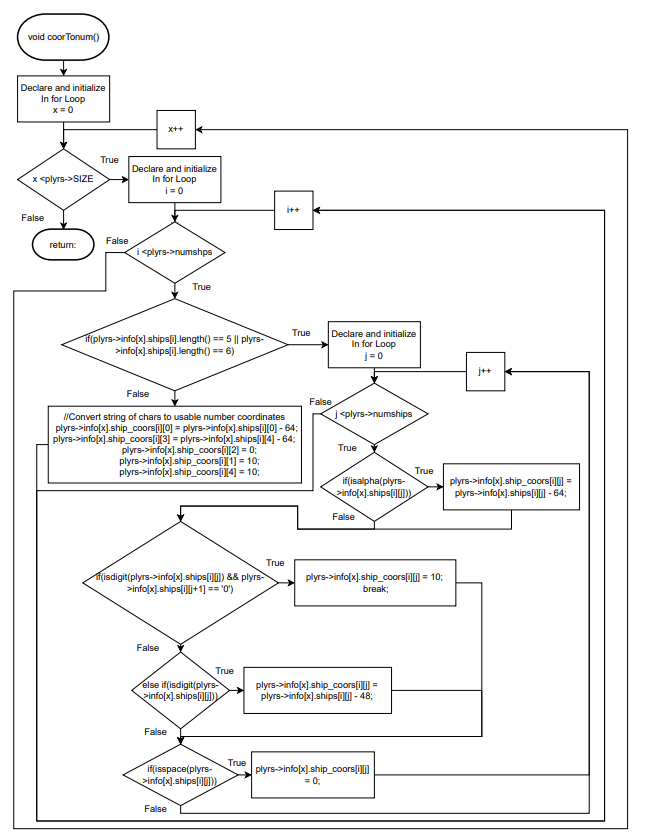
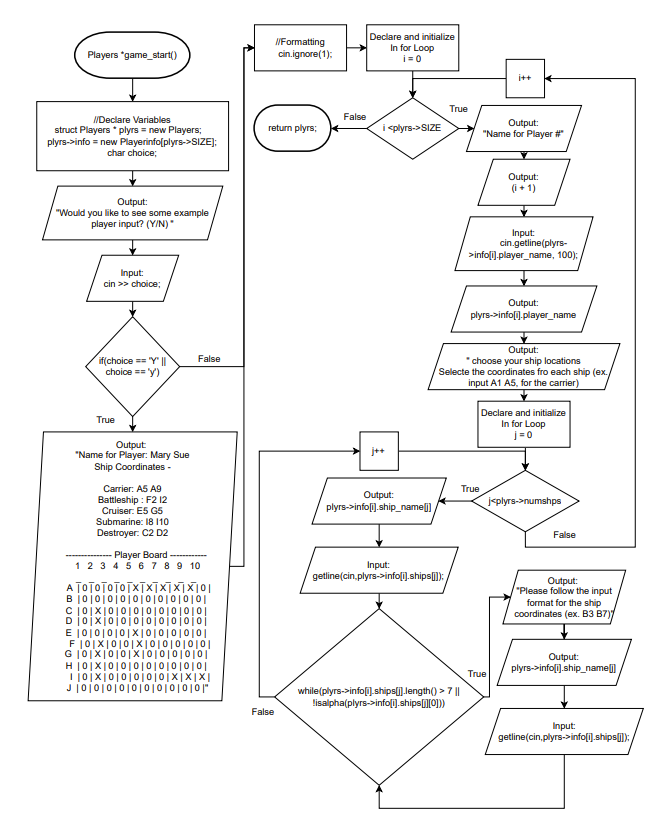
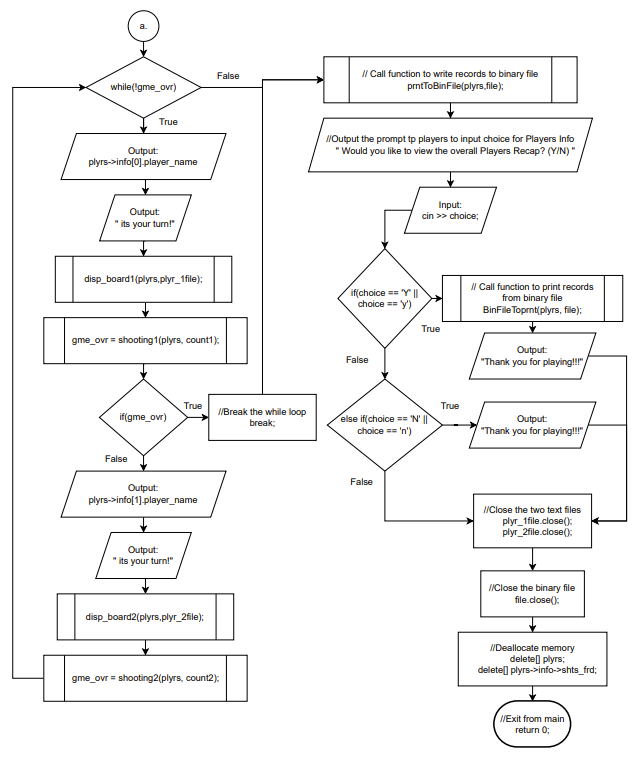
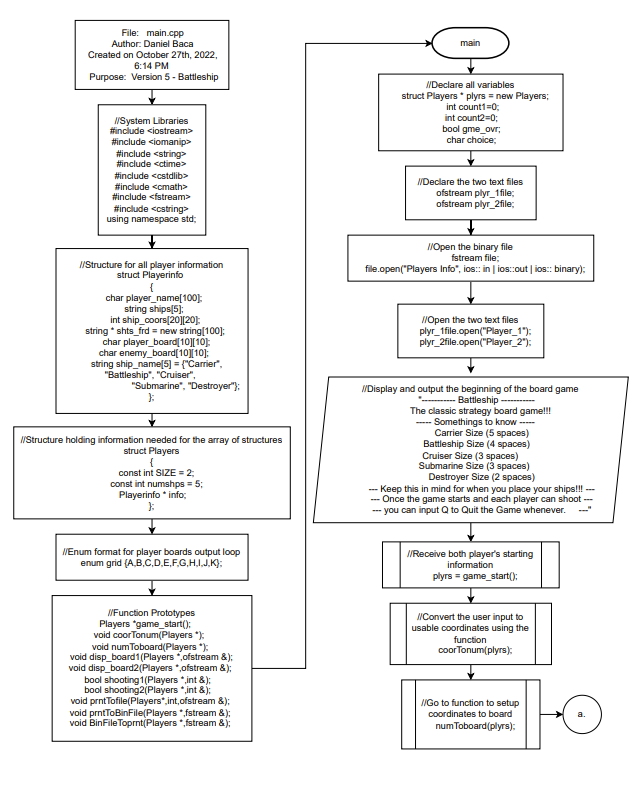


The following image shows the entire board when it is a players turn, they will be able to see both their player board and the enemy board where shots the player has made on the other player will be displayed and updated with every turn. Also when shooting, if the player has landed a shot on one of the enemy’s ships then it will indicated which ship was hit, just like in the real life game.



Lastly, once the game has ended either from one of the players sinking all of their opponents ships or voluntairly by quitting the game as done in the following image, the player will be prompted if they would like to view the endgame player recap information, if yes, then the players can view their names, ship coordinates, shots fired and even their final player boards.



**FlowChart**

**Pseudocode**

- Inlclude all System Libraries (ex. #include <iostream>, #include <iomanip>)

struct Playerinfo - Structure for all player information

{

Set char player\_name[100];

Set string ships[5];

Set int ship\_coors[20][20];

Set string \* shts\_frd = new string[100];

Set char player\_board[10][10];

Set char enemy\_board[10][10];

Set string ship\_name[5] = {"Carrier", "Battleship", "Cruiser",

"Submarine", "Destroyer"};

};

struct Players - Structure holding information needed for the array of structures

{

Set const int SIZE = 2;

Set const int numshps = 5;

Set Playerinfo \* info;

};

- Enum format for player boards output loop

enum grid {A,B,C,D,E,F,G,H,I,J,K};

- Function Prototypes

Players \*game\_start();

void coorTonum(Players \*);

void numToboard(Players \*);

void disp\_board1(Players \*,ofstream &);

void disp\_board2(Players \*,ofstream &);

bool shooting1(Players \*,int &);

bool shooting2(Players \*,int &);

void prntTofile(Players\*,int,ofstream &);

void prntToBinFile(Players \*,fstream &);

void BinFileToprnt(Players \*,fstream &);

int main(int argc, char\*\* argv)

{

Set struct Players \* plyrs = new Players;

Set int count1=0;

Set int count2=0;

Set bool gme\_ovr;

Set char choice;

Set ofstream plyr\_1file;

Set ofstream plyr\_2file;

Set fstream file;

Open the binary file, file, for input and output in binary mode

Open both the text files, plyr\_1file and plyr\_2file

Output "----------- Battleship -----------

The classic strategy board game!!!

----- Somethings to know -----

Carrier Size (5 spaces)

Battleship Size (4 spaces)

Cruiser Size (3 spaces)

Submarine Size (3 spaces)

Destroyer Size (2 spaces)

--- Keep this in mind for when you place your ships!!! ---

--- Once the game starts and each player can shoot ---

--- you can input Q to Quit the Game whenever. ---"

Call the game\_start function and input it into the plyrs struct

Call the coor\_to\_num function to convert the user input to number coordinates

Call the numToboard function to setup each player's board

while the game is not over

{

Output "It is players 1's turn!!!"

Call the disp\_board1() function to show player's board

Call the shooting1() function and receive whether or not the game is over

If the game is over

break out of the loop

Output "It is players 2's turn!!!"

Call the disp\_board2() function to show player's board

Call the shooting2() function and receive whether or not the game is over

}

Call the prntToBinFile() function to prnt the info to the binary file

Output "Would you like to view the overall Players Recap? (Y/N) "

//Use choice input to determine what to input for end of program

If choice is equal to 'Y' or 'y'

{

Call to BinFileToprnt() function to prnt info from the binary file

Output "Thank you for Playing!!! "

}

Else if choice is equal to 'N' or 'n'

cout << endl << " Thank you for Playing!!! " << endl;

Close both text files, plyr\_1file and plyr\_2file

Close the binary file, file

Deallocate the memory for plyrs and plyrs->info->shts\_frd

Return 0 and end the program

}

Players \*game\_start()

{

Set struct Players \* plyrs = new Players;

Set plyrs->info = new Playerinfo[plyrs->SIZE];

Set char choice;

Output "Would you like to see some example player input? (Y/N) "

If choice is equal to 'Y' or 'y'

{

Output " Name for Player: Mary Sue

Ship Coordinates -

Carrier: A5 A9

Battleship: F2 I2

Cruiser: E5 G5

Submarine: I8 I10

Destroyer: C2 D2

--------------- Player Board ------------

1 2 3 4 5 6 7 8 9 10

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

A | 0 | 0 | 0 | 0 | X | X | X | X | X | 0 |

B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

C | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

D | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

E | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 |

F | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 |

G | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 |

H | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

I | 0 | X | 0 | 0 | 0 | 0 | 0 | X | X | X |

J | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |"

}

for int i = 0 to plyrs->SIZE

{

Output "Name for the player # " and (i+1)

Input the name of the player in plyrs->info[i].player\_name

Output plrs->info[i].player\_name and "choose you Ship Locations ---

Select the coordinates for each ship

(ex. input A1 A5, for the carrier)"

for int j = 0 to plyrs->numships

{

Output plyrs->info[i].ship\_name[j] and ": "

Input the ship coordinates into plyrs->info[i].ships[j]

while plyrs->info[i].ships[j] is longer than 7 elements

and plyrs->info[i].ships[j][0] is not a letter

{

Output " Please follow the input format for the ship coordinates. (ex. B3 B7)"

Output plyrs->info[i].ship\_name[j] and ": "

Input the ship coordinates into plyrs->info[i].ships[j]

}

}

}

Return the plyrs structure

}

void coorTonum(struct Players \* plyrs)

{

for int x = 0 to plyrs->SIZE

{

for int i = 0 to plyrs->numshps

{

if plyrs->info[x].ships[i] is exactly 5 elements long or plyrs->info[x].ships[i] is exactly 6 elements long

{

for int j=0 to plyrs->numshps

{

if plyrs->info[x].ships[i][j] is a letter

Convert plyrs->info[x].ships[i][j] to an int to put in plyrs->info[x].ship\_coors[i][j]

if plyrs->info[x].ships[i][j] is a number and plyrs->info[x].ships[i][j+1] is equal to '0'

{

Set plyrs->info[x].ship\_coors[i][j] equal to 10

Break the loop

}

else if plyrs->info[x].ships[i][j] is a number

{

Convert plyrs->info[x].ships[i][j] to an int to put in plyrs->info[x].ship\_coors[i][j]

}

if plyrs->info[x].ships[i][j] is a space

Set plyrs->info[x].ship\_coors[i][j] equal to 0

}

}

else

{

Convert plyrs->info[x].ships[i][0] to an int to put into plyrs->info[x].ship\_coors[i][0]

Convert plyrs->info[x].ships[i][4] to an int to put into plyrs->info[x].ship\_coors[i][3]

Set plyrs->info[x].ship\_coors[i][2] equal to 0

Set plyrs->info[x].ship\_coors[i][1] equal to 10

Set plyrs->info[x].ship\_coors[i][4] equal to 10

}

}

}

}

void numToboard(struct Players \* plyrs)

{

for int z = 0 to plyrs->SIZE

for int i = A to K

for int j = A to K

{

Set plyrs->info[z].player\_board[i][j] equal to 0

Set plyrs->info[z].enemy\_board[i][j] equal to 0

}

for int z = 0 to plyrs->SIZE

{

for int i = 0 to plyrs->numshps

{

if plyrs->info[z].ship\_coors[i][0] is equal to plyrs->info[z].ship\_coors[i][3]

for int k = (plyrs->info[z].ship\_coors[i][1]-1) to plyrs->info[z].ship\_coors[i][4]

Set plyrs->info[z].player\_board[(plyrs->info[z].ship\_coors[i][0]-1)][k] euqal to 'X'

else

for int a = (plyrs->info[z].ship\_coors[i][0]-1) to plyrs->info[z].ship\_coors[i][3]

Set plyrs->info[z].player\_board[a][(plyrs->info[z].ship\_coors[i][1]-1)] equal to 'X'

}

}

}

bool shooting1(struct Players \* plyrs,int &count1)

{

Set int num[2];

Set int hits = 0;

Set int count = 0;

Output plyrs->info[0].player\_name and ", where to shoot? ";

Input the shot into plyrs->info[0].shts\_frd[count1]

if plyrs->info[0].shts\_frd[count1][0] is equal to 'Q' or 'q'

{

Output "Game over!"

Return that the game is over

}

while plyrs->info[0].shts\_frd[count1][0] is not a letter and

plyrs->info[0].shts\_frd[count1][0] is not an uppercase letter or

plyrs->info[0].shts\_frd[count1][1] is not a number

{

Output "Please input a correct coordinate. (ex. A10, B7, I3)"

Output plyrs->info[0].player\_name and ", where to shoot? ";

Input the shot into plyrs->info[0].shts\_frd[count1]

}

for int j = 0 to plyrs->SIZE

{

if plyrs->info[0].shts\_frd[count1][j] is a letter

Convert plyrs->info[0].shts\_frd[count1][j] to an int and put it into num[j]

if plyrs->info[0].shts\_frd[count1][j] is a number and plyrs->info[0].shts\_frd[count1][j+1] is equal to '0'

{

Set num[j] equal to 10

Break the loop

}

else if plyrs->info[0].shts\_frd[count1][j] is a number

{

Convert plyrs->info[0].shts\_frd[count1][j] to an int an put it into num[j]

}

}

// Change the game board to show missed and hit shots and output it

for int m = 0 to 11

{

for int l = 0 to 11

{

if l is equal to num[1] and m is equal to num[0]

{

if plyrs->info[1].player\_board[m-1][l-1] is equal to 'X'

{

Output " !!!!! It was a Hit! !!!!!"

Set plyrs->info[0].enemy\_board[m-1][l-1] equl to 'H'

Set plyrs->info[1].player\_board[m-1][l-1] equal to 'H'

for int x=0 to plyrs->numshps

{

if m is greater than or equal to plyrs->info[1].ship\_coors[x][0] and m is less than or equal to plyrs->info[1].ship\_coors[x][3]

and l is equal to plyrs->info[1].ship\_coors[x][1] and l is equal to plyrs->info[1].ship\_coors[x][4])

{

Perform (plyrs->info[1].ship\_coors[x][3]+1) - plyrs->info[1].ship\_coors[x][0] and put the result in count

if count is equal to 5

Output " ! You hit their Carrier !"

else if count is equal to 4

Output " ! You hit their Battleship !"

else if count is equal to 3

Output " ! You hit their Submarine or Cruiser !"

else if count is equal to 2

Output " ! You hit their Destroyer !"

}

else if(l is greater than or equal to plyrs->info[1].ship\_coors[x][1] and l is less than or equal to plyrs->info[1].ship\_coors[x][4]

and m is equal to plyrs->info[1].ship\_coors[x][0] and m is equal to plyrs->info[1].ship\_coors[x][3])

{

Set count equal to (plyrs->info[1].ship\_coors[x][4]+1) - plyrs->info[1].ship\_coors[x][1]

if count is equal to 5

Output " ! You hit their Carrier !"

else if count is equal to 4

Output " ! You hit their Battleship !"

else if count is equal to 3

Output " ! You hit their Submarine or Cruiser !"

else if count is equal to 2

Output " ! You hit their Destroyer !"

}

}

}

else

{

Output " !!!!! It was a Miss! !!!!!"

Set plyrs->info[0].enemy\_board[m-1][l-1] equal to 'M'

Set plyrs->info[1].player\_board[m-1][l-1] equal to 'M'

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for int z = 0 to 11

for int y = 0 to 11

if plyrs->info[1].player\_board[z][y] is equal to 'X'

Add 1 to hits

if hits is equal to 0

{

Output plyrs->info[0].player\_name and " has won the game!"

Return that the game is over

}

Add 1 to count1

Return that the game is not over yet

}

bool shooting2(struct Players \* plyrs,int &count2)

{

Set int num[2];

Set int hits = 0;

Set int count = 0;

Output plyrs->info[1].player\_name and ", where to shoot? ";

Input the shot into plyrs->info[1].shts\_frd[count1]

if plyrs->info[1].shts\_frd[count1][0] is equal to 'Q' or 'q'

{

Output "Game over!"

Return that the game is over

}

while plyrs->info[1].shts\_frd[count1][0] is not a letter and

plyrs->info[1].shts\_frd[count1][0] is not an uppercase letter or

plyrs->info[1].shts\_frd[count1][1] is not a number

{

Output "Please input a correct coordinate. (ex. A10, B7, I3)"

Output plyrs->info[1].player\_name and ", where to shoot? ";

Input the shot into plyrs->info[1].shts\_frd[count1]

}

for int j = 0 to plyrs->SIZE

{

if plyrs->info[1].shts\_frd[count1][j] is a letter

Convert plyrs->info[1].shts\_frd[count1][j] to an int and put it into num[j]

if plyrs->info[1].shts\_frd[count1][j] is a number and plyrs->info[1].shts\_frd[count1][j+1] is equal to '0'

{

Set num[j] equal to 10

Break the loop

}

else if plyrs->info[1].shts\_frd[count1][j] is a number

Convert plyrs->info[1].shts\_frd[count1][j] to an int an put it into num[j]

}

// Change the game board to show missed and hit shots and output it

for int m = 0 to 11

{

for int l = 0 to 11

{

if l is equal to num[1] and m is equal to num[0]

{

if plyrs->info[0].player\_board[m-1][l-1] is equal to 'X'

{

Output " !!!!! It was a Hit! !!!!!"

Set plyrs->info[1].enemy\_board[m-1][l-1] equl to 'H'

Set plyrs->info[0].player\_board[m-1][l-1] equal to 'H'

for int x=0 to plyrs->numshps

{

if m is greater than or equal to plyrs->info[0].ship\_coors[x][0] and m is less than or equal to plyrs->info[0].ship\_coors[x][3]

and l is equal to plyrs->info[0].ship\_coors[x][1] and l is equal to plyrs->info[0].ship\_coors[x][4])

{

Perform (plyrs->info[0].ship\_coors[x][3]+1) - plyrs->info[0].ship\_coors[x][0] and put the result in count

if count is equal to 5

Output " ! You hit their Carrier !"

else if count is equal to 4

Output " ! You hit their Battleship !"

else if count is equal to 3

Output " ! You hit their Submarine or Cruiser !"

else if count is equal to 2

Output " ! You hit their Destroyer !"

}

else if(l is greater than or equal to plyrs->info[0].ship\_coors[x][1] and l is less than or equal to plyrs->info[0].ship\_coors[x][4]

and m is equal to plyrs->info[0].ship\_coors[x][0] and m is equal to plyrs->info[0].ship\_coors[x][3])

{

Set count equal to (plyrs->info[0].ship\_coors[x][4]+1) - plyrs->info[0].ship\_coors[x][1]

if count is equal to 5

Output " ! You hit their Carrier !"

else if count is equal to 4

Output " ! You hit their Battleship !"

else if count is equal to 3

Output " ! You hit their Submarine or Cruiser !"

else if count is equal to 2

Output " ! You hit their Destroyer !"

}

}

}

else

{

Output " !!!!! It was a Miss! !!!!!"

Set plyrs->info[1].enemy\_board[m-1][l-1] equal to 'M'

Set plyrs->info[0].player\_board[m-1][l-1] equal to 'M'

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for int z = 0 to 11

for int y = 0 to 11

if plyrs->info[0].player\_board[z][y] is equal to 'X'

Add 1 to hits

if hits is equal to 0

{

Output plyrs->info[1].player\_name and " has won the game!"

Return that the game is over

}

Add 1 to count2

Return that the game is not over yet

}

void disp\_board1(struct Players \* plyrs, ofstream &plyr\_1file)

{

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

Output " -------- Enemy Board -----------

X = Your ships

1 2 3 4 5 6 7 8 9 10

H = The enemy ship has been hit

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

M = Your shot has not hit any ships"

for int x = A to K

{

Output " " and letters[x] and " "

for int z = A to K

Output "| " and plyrs->info[0].enemy\_board[x][z] and " "

Output "|" and " - - - - - - - - - - -"

}

Call prntToFile function to prnt player moves to text file

Output " -------- Player Board -----------

X = Your ships

1 2 3 4 5 6 7 8 9 10

H = Your ship has been hit

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

M = The enemy has missed your ships"

for int x = A to K

{

Output " " and letters[x] and " "

for int z = A to K

Output "| " and plyrs->info[0].player\_board[x][z] and " "

Output "|" and " - - - - - - - - - - -"

}

}

void disp\_board2(struct Players \* plyrs, ofstream &plyr\_2file)

{

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

Output " -------- Enemy Board -----------

X = Your ships

1 2 3 4 5 6 7 8 9 10

H = The enemy ship has been hit

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

M = Your shot has not hit any ships"

for int x = A to K

{

Output " " and letters[x] and " "

for int z = A to K

Output "| " and plyrs->info[1].enemy\_board[x][z] and " "

Output "|" and " - - - - - - - - - - -"

}

Call prntToFile function to prnt player moves to text file

Output " -------- Player Board -----------

X = Your ships

1 2 3 4 5 6 7 8 9 10

H = Your ship has been hit

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_

M = The enemy has missed your ships"

for int x = A to K

{

Output " " and letters[x] and " "

for int z = A to K

Output "| " and plyrs->info[1].player\_board[x][z] and " "

Output "|" and " - - - - - - - - - - -"

}

}

void prntTofile(struct Players \* plyrs, int play\_num, ofstream &plyr\_file)

{

Output to plyr\_file " ----- All Plays Made during the Game by this player -----"

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

Output to plyr\_file " 1 2 3 4 5 6 7 8 9 10" and

" \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

for int x = A to K

{

Output to plyr\_file " " and letters[x] and " "

for int z = A to K

Output to plyr\_file "| " and plyrs->info[play\_num].enemy\_board[x][z] and " "

Output to plyr\_file "|" and " - - - - - - - - - - -"

}

Output to plyr\_file " 1 2 3 4 5 6 7 8 9 10" and

" \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

for int x = A to K

{

Output to plyr\_file " " and letters[x] and " "

for int z = A to K

Output to plyr\_file "| " and plyrs->info[play\_num].player\_board[x][z] and " "

Output to plyr\_file "|" and " - - - - - - - - - - -"

}

}

void prntToBinFile(struct Players \* plyrs, fstream &file)

{

if file is not open

Output "error"

for int z=0 to plyrs->SIZE

Use the plyrs->info array of structures to write into the binary file

Close the binary file, file

}

void BinFileToprnt(struct Players \* plyrs, fstream &file)

{

Reopen the binary file, file, in input mode and in binary mode

Set struct Playerinfo temp[2];

if file is not open

Output "error"

Use the temp array of structures to read from the binary file

for int count = 0 to plyrs->SIZE

{

Output " --- Player #" and (count+1) and " Information ---" and " Name: "

and temp[count].player\_name

Output " Coordinates of ships:"

for int z = 0 to plyrs->numshps

Output " " and temp[count].ships[z]

Output " Shots Fired: "

for int z=0 to 100

Output " " and temp[count].shts\_frd[z]

Output " Final Player Board:"

for int z=0 to 10

{

for int f = 0 to 10

Output " " and temp[count].player\_board[z][f];

}

}

}

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| Integer | ship\_coors[][] | 2D Integer array holding numeric ship coordinates | Lines: 73 (struct Playerinfo{}) |
| Const int | SIZE | Size of the array of structures | Lines: 88 (struct Players{}) |
| Const int | numshps | Variable for the number of ships | Lines: 89 (struct Players{}) |
| Integer | count1 | Counter for the number of times Player 1 has shot at Player 2 | Lines: 124 and 382(main() and shooting1()) |
| Integer | count2 | Counter for the number of times Player 2 has shot at Player 1 | Lines: 126 and 523(main() and shooting2()) |
| Integer | num[] | Array holding the converted user number coordinates | Lines: 385 and 526 (shooting1() and shooting2()) |
| Integer | hits | Number of ships coordinates not yet shot | Lines: 386 and 527 (shooting1() and shooting2()) |
| Integer | count | Counter used to determine what kind of ship was hit | Lines: 387 and 528 (shooting1() and shooting2()) |
| Char | player\_name[] | C-string array holding the names of each player | Lines: 70 (struct Playerinfo{}) |
| Char | player\_board[][] | C-string 2D array holding the players board moves | Lines: 77 (struct Playerinfo{}) |
| Char | enemy\_board[][] | C-string 2D array holding the enemy board moves | Lines: 78 (struct Playerinfo{}) |
| Char | choice | The character input for whether for not to have endgame review | Lines: 130 (main()) |
| Char | choice | The character input for whether or not to display example input | Lines: 228 (game\_start()) |
| Char | letters[] | Char array for letters on the game board | Lines: 664, 720, 781 (disp\_board1(), disp\_board2() and prntTofile()) |
| String | ships[] | String array holding the ship coordinates input by the player | Lines: 71 (struct Playerinfo{}) |
| String | shts\_frd[] | Dynamic String Array holding the shots fired by the player | Lines: 75 (struct Playerinfo{}) |
| String | ship\_name[] | String array holding the names of each ship | Lines: 81 (struct Playerinfo{}) |
| Bool | gme\_ovr | Bool variable gotten from the shooting functions telling whether or not the game has ended | Lines: 128 (main()) |
| Struct | \* plyrs | Dynamic Pointer to the structure variable from struct Players{} | Lines: 122 (main()) |
| Struct | \* info | Structure Pointer to the Playerinfo array of structures | Lines: 90 (struct Player{}) |
| enum | grid | Format for player board rows output loop | Lines: 94 Global() |
| Text File | plyr\_1file | File recording all the moves made by player 1 | Lines: 134, 661, 773 (main(), disp\_board1() and prntTofile()) |
| Text File | plyr\_2file | File recording all the moves made by player 2 | Lines: 135, 717, 773(main(), disp\_board2() and prntTofile()) |
| Binary File | file | Binary File holding all of the players information at the end of the game | Lines: 138, 817, 832(main(), prntToBinFile() and BinFileTOprnt()) |

**C++ Concepts**

|  |  |  |
| --- | --- | --- |
| Chapter | New syntax and Keywords | Location |
| 9 | Pointer Variables - string \* shts\_frd,  struct Playerinfo \* info, struct Players \* plyrs, | Lines: 75 (struct Playerinfo{}), 90 (Struct Players{}), 122 (main()) |
| Arrays/Pointers - Playerinfo \* info,  struct Players \* plyrs = new Players;  plyrs->info = new Playerinfo[plyrs->SIZE]; | Lines: 90 (struct Players{}), 226-227 (game\_start()) |
| Function Parameters - void coorTonum(struct Players \* plyrs), void numToboard(struct Players \* plyrs), bool shooting1(struct Players \* plyrs,int &count1) | Lines: 304 (coorTonum()), 354 (numToboard()), 382 (shooting1()) |
| Memory allocation -  string \* shts\_frd = new string[100];  struct Players \* plyrs = new Players;  struct Players \* plyrs = new Players;  plyrs->info = new Playerinfo[plyrs->SIZE]; | Lines: 75 (struct Playerinfo{}), 122 (main()), 227 (game\_start()) |
| Return Parameters –  Players \*game\_start();  return plyrs; | Lines: 97, 223 (game\_start()), 299 (game\_start()) |
| 10 | Testing –  if(isalpha(plyrs->info[x].ships[i][j]))  else if(isdigit(plyrs->info[x].ships[i][j]))  if(isspace(plyrs->info[x].ships[i][j])) | Lines: 320(coorTonum()), 330(coorTonum()), 335(coorTonum()) |
| C-strings –  char player\_name[100];  char player\_board[10][10];  char enemy\_board[10][10];  char letters[10] = {'A','B','C','D','E','F','G','H','I','J'}; | Lines: 70 ,77, 78 (all in struct Playerinfo{}), 664 (disp\_board1()) |
| Strings –  string ships[5];  string \* shts\_frd = new string[100];  string ship\_name[5] = {"Carrier", "Battleship", "Cruiser", "Submarine", "Destroyer"}; | Lines: 71, 75, 81 (all in struct Playerinfo{}) |
| 11 | Data –  struct Playerinfo  {  char player\_name[100];  string ships[5];  int ship\_coors[20][20];  string \* shts\_frd = new string[100];  char player\_board[10][10];  char enemy\_board[10][10];  string ship\_name[5] = {"Carrier", "Battleship", "Cruiser", "Submarine", "Destroyer"};  }; | Lines: 68 – 83 (struct Playerinfo{}) |
| Access –  cin.getline(plyrs->info[i].player\_name, 100);  getline(cin,plyrs->info[i].ships[j]);  cout << setw(20) << plyrs >info[i].ship\_name[j] << ": "; | Lines: 268, 282 and 292 (all in game\_start()) |
| Arrays –  struct Players \* plyrs = new Players; plyrs->info = new Playerinfo[plyrs->SIZE];  struct Playerinfo temp[2]; | Lines: 226 - 227 (game\_start()), and 838 (BinFileToprnt()) |
| Nested –  struct Playerinfo  {  char player\_name[100];  string ships[5];  int ship\_coors[20][20];  string \* shts\_frd = new string[100];  char player\_board[10][10];  char enemy\_board[10][10];  string ship\_name[5] = {"Carrier", "Battleship", "Cruiser", "Submarine", "Destroyer"};  };    struct Players  {  const int SIZE = 2;  const int numshps = 5;  Playerinfo \* info;  }; | Lines: 90 (struct Players{}) |
| Function Parameters –  void coorTonum(struct Players \* plyrs)  void numToboard(struct Players \* plyrs)  bool shooting1(struct Players \* plyrs,int &count1) | Lines: 304 (coorTonum()), 354 (numToboard()), 382 (shooting1()) |
| Function Return –  Players \*game\_start()  {  struct Players \* plyrs = new Players;  plyrs->info = new Playerinfo[plyrs->SIZE];  return plyrs;  } | Lines: 223, 226-227, and 299 (all in game\_start()) |
| Pointers –  Playerinfo \* info;  struct Players \* plyrs = new Players; | Lines: 90 (struct Players{}), 122 (main()) |
| Enumeration –  enum grid {A,B,C,D,E,F,G,H,I,J,K};  for(int z=A;z<K;z++)  cout << "| " << plyrs >info[0].enemy\_board[x][z]<< " ";  for(int z=A;z<K;z++)  plyr\_file << "| " << plyrs->info[play\_num].enemy\_board[x][z]<< " "; | Lines: 94 (Global), 679 - 680 (disp\_board1()), 791 and 792 (prnTofile()) |
| 12 | File Operations –  file.open("Players Info", ios:: in | ios::out | ios:: binary);  plyr\_1file.open("Player\_1");  plyr\_2file.open("Player\_2");  file.open("Players Info", ios:: in | ios:: binary); | Lines: 139,142-143 (all in main()), and 835 (BinFileToprnt()) |
| Formatting –  plyr\_file << endl << " ----- All Plays Made during the Game by this player -----"  << endl << endl;  plyr\_file << " 1 2 3 4 5 6 7 8 9 10" << endl  << " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl << endl; | Lines: 776-778, and 783-784 (all in prntTofile()) |
| Function Parameters –  void prntTofile(struct Players \* plyrs, int play\_num, ofstream &plyr\_file)  void prntToBinFile(struct Players \* plyrs, fstream &file)  void BinFileToprnt(struct Players \* plyrs, fstream &file) | Lines: 773 (prntTofile()), 817 (prntToBinFile(), 832 (BinFileToprnt) |
| Member Functions – N/A | Lines: N/A |
| Multiple Files –  ofstream plyr\_1file;  ofstream plyr\_2file;  fstream file;  file.open("Players Info", ios:: in | ios::out | ios:: binary);  plyr\_1file.open("Player\_1");  plyr\_2file.open("Player\_2"); | Lines: 134-135, 138, 139, 142-143 (all in main()) |
|  | Binary Files –  fstream file;  file.open("Players Info", ios:: in | ios::out | ios:: binary); | Lines: 138, 139 (main()) |
|  | Records with structures –  for(int z=0;z<plyrs->SIZE;z++)  file.write(reinterpret\_cast<char \*>(&plyrs->info[z]), sizeof(plyrs->info[z]));  file.read(reinterpret\_cast<char \*>(&temp), sizeof(temp)); | Lines: 824 -825 (prntToBinFile()), and 845 (BinFileToprnt) |
|  | Random Access files – N/A | Lines: N/A |
|  | Input/Output Simultaneous –  fstream file;  file.open("Players Info", ios:: in | ios::out | ios:: binary);  for(int z=0;z<plyrs->SIZE;z++)  file.write(reinterpret\_cast<char \*>(&plyrs->info[z]), sizeof(plyrs->info[z]));  file.read(reinterpret\_cast<char \*>(&temp), sizeof(temp)); | Lines: 139 (main()), 824 -825 (prntToBinFile()), and 845 (BinFileToprnt) |

**References**

1. Gaddis, Tony “Starting Out with C++ from Control Structures to Objects – 9th Edition”
2. Various C++ Programming Websites (<https://www.geeksforgeeks.org/>, <https://stackoverflow.com/>, <https://cplusplus.com/> )
3. Source Used as Reference Material for Rules of Battleship Board Game: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwi1qd2g5IT7AhX0MUQIHVIADQUQFnoECBQQAQ&url=https%3A%2F%2Fwww.hasbro.com%2Fcommon%2Finstruct%2Fbattleship.pdf&usg=AOvVaw2XhO4v8ZigPERaaI8L0_K4>

**Program**

/\*

\* File: main.cpp

\* Author: Daniel Baca

\* Created on October 27th, 2022, 6:14 PM

\* Purpose: Version 5 - Battleship

\*

\* Create a functional game of BattleShip, implementing many of the programming

\* concepts from Chapter 9 to 12, while following the rules of the original

\* board game which are:

\*

\* 1. Players place their 5 ships on their player board, the rules here are:

\*

\* - Place each ship in any horizontal or vertical position, but not

\* diagonally.

\* - Do not place a ship so that it overlaps the edge of the board or another

\* ship.

\* - Do not change the position of any ship once the game has begun.

\*

\* 2. Decide who will go first, each player will alternate turns, calling out

\* one shot per turn to try and hit each other's ships.

\*

\* 3. On your turn, pick a target hole on your enemy board and call out its

\* location by the letter and number. Each target hole has a letter-number

\* coordinate that corresponds with the same coordinate on your

\* enemy's board. To determine each coordinate, find its

\* corresponding letter on the left side of the target board and its number

\* on the top of the board. For example, when you call a shot, your opponent

\* must tell you whether your shot is a hit or a miss.

\*

\* 4. It's a Hit - If you call out a shot location that is occupied by a ship on

\* your enemy's board, your shot is a hit! Your opponent tells you which ship

\* you have hit (cruiser, submarine, etc.). Your shots will be recorded for

\* you on each of the boards. Your opponent's board will be updated in the

\* corresponding hole of the ship you have hit on his or her board.

\*

\* 5. It's a Miss - If you call out a shot location not occupied by a ship on

\* your opponent's board, it's a miss. The board will record this move, so

\* you cannot call this shot again.

\*

\* Play continues in this manner, with you and your opponent calling one shot

\* per turn.

\*

\* 6. Once all the holes in any one ship are hit then the ship is considered

\* sunk. The owner of the ship must announce which ship was sunk.

\*

\* 7. If you're the first player to sink your opponents entire fleet of 5 ships,

\* you win the game!

\*

\*/

//System Libraries

#include <iostream>

#include <iomanip>

#include <string>

#include <ctime>

#include <cstdlib>

#include <cmath>

#include <fstream>

#include <cstring>

using namespace std;

//User Libraries

//Global Constants Only

//Well known Science, Mathematical and Laboratory Constants

//Structure for all player information

struct Playerinfo

{

char player\_name[100]; //C-string Array holding the name of the player

string ships[5]; //String Array holding the ship coordinates input

//by the player

int ship\_coors[20][20]; //Number Array with the number coordinates converted

//from user input

string \* shts\_frd = new string[100]; //Dynamic String Array holding the

//shots fired by the player

char player\_board[10][10]; //C-string array holding the players board moves

char enemy\_board[10][10]; //C-string array holding the enemy board moves

//String array holding the names of each ship

string ship\_name[5] = {"Carrier", "Battleship", "Cruiser",

"Submarine", "Destroyer"};

};

//Structure holding information needed for the array of structures

struct Players

{

const int SIZE = 2; //Size of the array of structures

const int numshps = 5; //Variable for number of ships

Playerinfo \* info; //Pointer holding the array of structures

};

//Enum format for player boards output loop

enum grid {A,B,C,D,E,F,G,H,I,J,K};

//Function Prototypes

Players \*game\_start(); //Get both player's names and ship coordinates

void coorTonum(Players \*); //Convert string input to usable coordinates

void numToboard(Players \*); //Input converted coordinates to match player

//boards

void disp\_board1(Players \*,ofstream &); //Display Player 1's two game

//boards

void disp\_board2(Players \*,ofstream &); //Display Player 2's two game

//boards

bool shooting1(Players \*,int &); //Have Player 1 shoot at the enemy's board,

//show what was hit or missed and end game

//if necessary

bool shooting2(Players \*,int &); //Have Player 2 shoot at the enemy's board,

//show what was hit or missed and end game

//if necessary

void prntTofile(Players\*,int,ofstream &); //Print each player's move to their

//file

void prntToBinFile(Players \*,fstream &); //Print the overall players

//information to the Binary file

void BinFileToprnt(Players \*,fstream &); //Read from the binary file and output

//the player's information

//Execution of Code Begins Here

int main(int argc, char\*\* argv)

{

//Declare all variables

struct Players \* plyrs = new Players; //Dynamic Structure for Players

int count1=0; //A counter for the number of times players 1 has shot at

//Player 2

int count2=0; //A counter for the number of times players 2 has shot at

//Player 1

bool gme\_ovr; //The bool variable gotten from the shooting functions

//telling whether or not the game has ended or not

char choice; //The character input for whether or not to have endgame

//review

//Declare the two text files

ofstream plyr\_1file;

ofstream plyr\_2file;

//Open the binary file

fstream file;

file.open("Players Info", ios:: in | ios::out | ios:: binary);

//Open the two text files

plyr\_1file.open("Player\_1");

plyr\_2file.open("Player\_2");

//Display the beginning of the board game

cout << " ----------- Battleship -----------" << endl

<< " The classic strategy board game!!!" << endl;

cout << " ----- Somethings to know -----" << endl

<< " Carrier Size (5 spaces)" << endl

<< " Battleship Size (4 spaces)" << endl

<< " Cruiser Size (3 spaces)" << endl

<< " Submarine Size (3 spaces)" << endl

<< " Destroyer Size (2 spaces)" << endl << endl

<< " --- Keep this in mind for when you place your ships!!! ---"

<< endl << " --- Once the game starts and each player can shoot ---"

<< endl << " --- you can input Q to Quit the Game whenever. ---"

<< endl << endl;

//Receive both player's starting information

plyrs = game\_start();

//Convert the user input to usable coordinates using the function

coorTonum(plyrs);

//Go to function to setup coordinates to board

numToboard(plyrs);

//Loop for each player's turn, shooting their shots until someone wins

while(!gme\_ovr)

{

//Start Player 1's turn

cout << endl << " " <<plyrs->info[0].player\_name << " it's your turn!!!"

<< endl << endl;

disp\_board1(plyrs,plyr\_1file);

gme\_ovr = shooting1(plyrs, count1);

//End game if necessary

if(gme\_ovr)

break;

//Start Player 2's turn

cout << endl << " " <<plyrs->info[1].player\_name << " it's your turn!!!"

<< endl << endl;

disp\_board2(plyrs,plyr\_2file);

gme\_ovr = shooting2(plyrs, count2);

}

// Call function to write records to binary file

prntToBinFile(plyrs,file);

//Prompt players to input choice for Players Info

cout << " Would you like to view the overall Players Recap? (Y/N) ";

cin >> choice;

//Use choice input to determine what to input for end of program

if(choice == 'Y' || choice == 'y')

{

// Call function to print records from binary file

BinFileToprnt(plyrs, file);

cout << endl << " Thank you for Playing!!! " << endl;

}

else if(choice == 'N' || choice == 'n')

cout << endl << " Thank you for Playing!!! " << endl;

//Close the two text files

plyr\_1file.close();

plyr\_2file.close();

//Close the binary file

file.close();

//Deallocate memory

delete[] plyrs;

delete[] plyrs->info->shts\_frd;

//Clean up the code, close files, deallocate memory, etc....

//Exit stage right

return 0;

}

// Function for receiving all player information including name and ship

// coordinates

Players \*game\_start()

{

//Declare Variables

struct Players \* plyrs = new Players; //Dynamic array structure

plyrs->info = new Playerinfo[plyrs->SIZE]; //for function

char choice;

//Ask player if they would like to see example input

cout << "Would you like to see some example player input? (Y/N) ";

cin >> choice;

//Use choice input to determine what to input for end of program

if(choice == 'Y' || choice == 'y')

{

cout << endl << " Name for Player: Mary Sue" << endl << endl

<< " Ship Coordinates -" << endl << endl

<< " Carrier: A5 A9" << endl

<< " Battleship: F2 I2" << endl

<< " Cruiser: E5 G5" << endl

<< " Submarine: I8 I10" << endl

<< " Destroyer: C2 D2" << endl << endl

<< " --------------- Player Board ------------" << endl

<< " 1 2 3 4 5 6 7 8 9 10" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl

<< " A | 0 | 0 | 0 | 0 | X | X | X | X | X | 0 | " << endl

<< " B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " C | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " D | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " E | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " F | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " G | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | " << endl

<< " H | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< " I | 0 | X | 0 | 0 | 0 | 0 | 0 | X | X | X | " << endl

<< " J | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | " << endl

<< endl;

}

//Formatting

cin.ignore(1);

//Loop to get input for each player

for(int i=0; i<plyrs->SIZE;i++)

{

// Get Name of Player

cout << endl << "Name for Player #" << (i+1) << ": ";

cin.getline(plyrs->info[i].player\_name, 100);

cout << endl;

// Prompt user to input ship coordinates

cout << " --- " << plyrs->info[i].player\_name

<< " choose you Ship Locations ---" << endl;

cout << "Select the coordinates for each ship "

<< "(ex. input A1 A5, for the carrier)" << endl << endl;

//Loop for ship coordinates input

for(int j=0;j<plyrs->numshps;j++)

{

//Input the ship coordinates

cout << setw(20) << plyrs->info[i].ship\_name[j] << ": ";

getline(cin,plyrs->info[i].ships[j]);

//Input Validation for Ship Coordinates

while(plyrs->info[i].ships[j].length() > 7 || !isalpha(plyrs->info[i].ships[j][0]))

{

//Prompt the user to enter valid input

cout << endl << " Please follow the input format for the ship "

<< "coordinates. (ex. B3 B7)" << endl << endl;

//Get the input again

cout << setw(20) << plyrs->info[i].ship\_name[j] << ": ";

getline(cin,plyrs->info[i].ships[j]);

}

}

}

//Return the plyrs pointer and the array of structures

return plyrs;

}

//Function used to convert players string input to usable int number coordinates

//for the rest of the program

void coorTonum(struct Players \* plyrs)

{

// Loop for each player and each subsequent string input for their ship

// coordinates

for(int x=0;x<plyrs->SIZE;x++)

{

for(int i=0;i<plyrs->numshps;i++)

{

//See if the string is a certain length and choose option based on it

if(plyrs->info[x].ships[i].length() == 5 || plyrs->info[x].ships[i].length() == 6)

{

for(int j=0;j<plyrs->numshps;j++)

{

// Determine if the portion of the string is a letter and convert to

// an int

if(isalpha(plyrs->info[x].ships[i][j]))

plyrs->info[x].ship\_coors[i][j] = plyrs->info[x].ships[i][j] - 64;

// Determine if the portion of the string is a number and convert to

// an int

if(isdigit(plyrs->info[x].ships[i][j]) && plyrs->info[x].ships[i][j+1] == '0')

{

plyrs->info[x].ship\_coors[i][j] = 10;

break;

}

else if(isdigit(plyrs->info[x].ships[i][j]))

{

plyrs->info[x].ship\_coors[i][j] = plyrs->info[x].ships[i][j] - 48;

}

// Determine if the portion of the string is a space

if(isspace(plyrs->info[x].ships[i][j]))

plyrs->info[x].ship\_coors[i][j] = 0;

}

}

else

{

//Convert string of chars to usable number coordinates

plyrs->info[x].ship\_coors[i][0] = plyrs->info[x].ships[i][0] - 64;

plyrs->info[x].ship\_coors[i][3] = plyrs->info[x].ships[i][4] - 64;

plyrs->info[x].ship\_coors[i][2] = 0;

plyrs->info[x].ship\_coors[i][1] = 10;

plyrs->info[x].ship\_coors[i][4] = 10;

}

}

}

}

//Function used to have the converted coordinates format the player boards with

//each of their ships

void numToboard(struct Players \* plyrs)

{

// Initialize empty board layout

for(int z=0;z<plyrs->SIZE;z++)

for(int i=A;i<K;i++)

for(int j=A;j<K;j++)

{

plyrs->info[z].player\_board[i][j] = '0';

plyrs->info[z].enemy\_board[i][j] = '0';

}

// Place X's that represent ships on the player's boards

for(int z=0;z<plyrs->SIZE;z++)

{

for(int i=0;i<plyrs->numshps;i++)

{

if(plyrs->info[z].ship\_coors[i][0] == plyrs->info[z].ship\_coors[i][3])

for(int k=(plyrs->info[z].ship\_coors[i][1]-1);k<plyrs->info[z].ship\_coors[i][4];k++)

plyrs->info[z].player\_board[(plyrs->info[z].ship\_coors[i][0]-1)][k] = 'X';

else

for(int a=(plyrs->info[z].ship\_coors[i][0]-1);a<plyrs->info[z].ship\_coors[i][3];a++)

plyrs->info[z].player\_board[a][(plyrs->info[z].ship\_coors[i][1]-1)] = 'X';

}

}

}

//Function used to have player 1 shoot at player 2, output hits and ships hit or

//missed, and decide if the game is over yet

bool shooting1(struct Players \* plyrs,int &count1)

{

//Declare Variables

int num[2]; //Array holding converted user number coordinates

int hits = 0; //Number of ship coordinates not yet shot

int count = 0; //Counter used to determine what kind of ship was hit

// Initial shot user input

cout << endl << " " << plyrs->info[0].player\_name << ", where to shoot? ";

getline(cin,plyrs->info[0].shts\_frd[count1]);

cout << endl;

//Check to see if player wants to quit the game

if(plyrs->info[0].shts\_frd[count1][0] == 'Q' || plyrs->info[0].shts\_frd[count1][0] == 'q')

{

cout << " Game over!" << endl << endl;

return true;

}

// Input Validation for user shot input

while(!isalpha(plyrs->info[0].shts\_frd[count1][0])

&& !isupper(plyrs->info[0].shts\_frd[count1][0])

|| !isdigit(plyrs->info[0].shts\_frd[count1][1]))

{

cout << "Please input a correct coordinate. (ex. A10, B7, I3)" << endl;

cout << endl << " " << plyrs->info[0].player\_name << ", where to shoot? ";

getline(cin,plyrs->info[0].shts\_frd[count1]);

cout << endl;

}

// Convert the input coordinates into usable number coordinates

for(int j=0;j<plyrs->SIZE;j++)

{

if(isalpha(plyrs->info[0].shts\_frd[count1][j]))

num[j] = plyrs->info[0].shts\_frd[count1][j] - 64;

if(isdigit(plyrs->info[0].shts\_frd[count1][j]) && plyrs->info[0].shts\_frd[count1][j+1] == '0')

{

num[j] = 10;

break;

}

else if(isdigit(plyrs->info[0].shts\_frd[count1][j]))

{

num[j] = plyrs->info[0].shts\_frd[count1][j] - 48;

}

}

// Change the game board to show missed and hit shots and output it

for(int m=0;m<11;m++)

{

for(int l=0;l<11;l++)

{

if(l == num[1] && m == num[0])

{

if(plyrs->info[1].player\_board[m-1][l-1]== 'X')

{

//Output that a ship has been hit

cout << " !!!!! It was a Hit! !!!!!" << endl;

plyrs->info[0].enemy\_board[m-1][l-1] = 'H';

plyrs->info[1].player\_board[m-1][l-1] = 'H';

//Determine what ship has been shot at

for(int x=0;x<plyrs->numshps;x++)

{

if(m >= plyrs->info[1].ship\_coors[x][0] && m <= plyrs->info[1].ship\_coors[x][3]

&& l == plyrs->info[1].ship\_coors[x][1] && l == plyrs->info[1].ship\_coors[x][4])

{

//Length of the ship shot used for output

count = (plyrs->info[1].ship\_coors[x][3]+1) - plyrs->info[1].ship\_coors[x][0];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

else if(l >= plyrs->info[1].ship\_coors[x][1] && l <= plyrs->info[1].ship\_coors[x][4]

&& m == plyrs->info[1].ship\_coors[x][0] && m == plyrs->info[1].ship\_coors[x][3])

{

//Length of the ship shot used for output

count = (plyrs->info[1].ship\_coors[x][4]+1) - plyrs->info[1].ship\_coors[x][1];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

}

}

else

{

//Output that the shot has missed hitting any ships and

//record it

cout << " !!!!! It was a Miss! !!!!!" << endl;

plyrs->info[0].enemy\_board[m-1][l-1] = 'M';

plyrs->info[1].player\_board[m-1][l-1] = 'M';

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for(int z=0;z<11;z++)

for(int y=0;y<11;y++)

if(plyrs->info[1].player\_board[z][y] == 'X')

hits++;

// Condition to see if the game has ended yet

if(hits == 0)

{

cout << endl << " " << plyrs->info[0].player\_name

<< " has won the game!" << endl << endl;

return true;

}

// Increase count for recording shots

count1++;

// Return false if the game hasn't ended

return false;

}

//Function used to have player 2 shoot at player 1, output hits and ships hit or

//missed, and decide if the game is over yet

bool shooting2(struct Players \* plyrs,int &count2)

{

//Declare Variables

int num[2]; //Array holding converted user number coordinates

int hits = 0; //Number of ship coordinates not yet shot

int count = 0; //Counter used to determine what kind of ship was hit

// Initial shot user input

cout << endl << " " << plyrs->info[1].player\_name << ", where to shoot? ";

getline(cin,plyrs->info[1].shts\_frd[count2]);

cout << endl;

//Check to see if user wants to quit the game

if(plyrs->info[1].shts\_frd[count2][0] == 'Q' || plyrs->info[1].shts\_frd[count2][0] == 'q')

{

cout << " Game over!" << endl << endl;

return true;

}

// Input Validation for user shot input

while(!isalpha(plyrs->info[1].shts\_frd[count2][0])

&& !isupper(plyrs->info[1].shts\_frd[count2][0])

|| !isdigit(plyrs->info[1].shts\_frd[count2][1]))

{

cout << "Please input a correct coordinate. (ex. A10, B7, I3)" << endl;

cout << endl << " " << plyrs->info[1].player\_name << ", where to shoot? ";

getline(cin,plyrs->info[1].shts\_frd[count2]);

cout << endl;

}

// Convert the input coordinates into usable number coordinates

for(int j=0;j<plyrs->SIZE;j++)

{

if(isalpha(plyrs->info[1].shts\_frd[count2][j]))

num[j] = plyrs->info[1].shts\_frd[count2][j] - 64;

if(isdigit(plyrs->info[1].shts\_frd[count2][j]) && plyrs->info[1].shts\_frd[count2][j+1] == '0')

{

num[j] = 10;

break;

}

else if(isdigit(plyrs->info[1].shts\_frd[count2][j]))

num[j] = plyrs->info[1].shts\_frd[count2][j] - 48;

}

// Change the game board to show missed and hit shots and output it

for(int m=0;m<11;m++)

{

for(int l=0;l<11;l++)

{

if(l == num[1] && m == num[0])

{

if(plyrs->info[0].player\_board[m-1][l-1]== 'X')

{

//Output that a ship has been hit

cout << " !!!!! It was a Hit! !!!!!" << endl;

plyrs->info[1].enemy\_board[m-1][l-1] = 'H';

plyrs->info[0].player\_board[m-1][l-1] = 'H';

//Determine what ship has been shot at

for(int x=0;x<plyrs->numshps;x++)

{

if(m >= plyrs->info[0].ship\_coors[x][0] && m <= plyrs->info[0].ship\_coors[x][3]

&& l == plyrs->info[0].ship\_coors[x][1] && l == plyrs->info[0].ship\_coors[x][4])

{

//Length of the ship shot used for output

count = (plyrs->info[0].ship\_coors[x][3]+1) - plyrs->info[0].ship\_coors[x][0];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

else if(l >= plyrs->info[0].ship\_coors[x][1] && l <= plyrs->info[0].ship\_coors[x][4]

&& m == plyrs->info[0].ship\_coors[x][0] && m == plyrs->info[0].ship\_coors[x][3])

{

//Length of the ship shot used for output

count = (plyrs->info[0].ship\_coors[x][4]+1) - plyrs->info[0].ship\_coors[x][1];

//Output which ship was hit

if(count == 5)

cout << " ! You hit their Carrier !"

<< endl;

else if(count == 4)

cout << " ! You hit their Battleship !"

<< endl;

else if(count == 3)

cout << " ! You hit their Submarine or"

<< " Cruiser !" << endl;

else if(count == 2)

cout << " ! You hit their Destroyer !"

<< endl;

}

}

}

else

{

//Output that the shot has missed hitting any ships and

//record it

cout << " !!!!! It was a Miss! !!!!!" << endl;

plyrs->info[1].enemy\_board[m-1][l-1] = 'M';

plyrs->info[0].player\_board[m-1][l-1] = 'M';

}

}

}

}

// Detect to see if there are any ships coordinates not shot yet

for(int z=0;z<11;z++)

for(int y=0;y<11;y++)

if(plyrs->info[0].player\_board[z][y] == 'X')

hits++;

// Condition to see if the game has ended yet

if(hits == 0)

{

cout << endl << " " << plyrs->info[1].player\_name

<< " has won the game!" << endl << endl;

return true;

}

// Increase count for recording shots

count2++;

// Return false if the game hasn't ended

return false;

}

// Function used to display the boards for each player on their turns

void disp\_board1(struct Players \* plyrs, ofstream &plyr\_1file)

{

// The char array for letters in the game board

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

// Format the output and general layout of the game boards

cout << " -------- Enemy Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = The enemy ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = Your shot has not hit any ships" << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

cout << " " << letters[x] << " ";

for(int z=A;z<K;z++)

cout << "| " << plyrs->info[0].enemy\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

// Use the print to file function to record players moves

prntTofile(plyrs,0,plyr\_1file);

// Format the output and general layout of the game boards

cout << " -------- Player Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = Your ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = The enemy has missed your ships" << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

cout << " " << letters[x] << " ";

for(int z=A;z<K;z++)

cout << "| " << plyrs->info[0].player\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

}

// Function used to display the boards for each player on their turns

void disp\_board2(struct Players \* plyrs, ofstream &plyr\_2file)

{

// The char array for letters in the game board

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

// Format the output and general layout of the game boards

cout << " -------- Enemy Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = The enemy ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = Your shot has not hit any ships" << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

cout << " " << letters[x] << " ";

for(int z=A;z<K;z++)

cout << "| " << plyrs->info[1].enemy\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//Formatting

cout << endl;

// Use the print to file function to record players moves

prntTofile(plyrs,1,plyr\_2file);

// Format the output and general layout of the game boards

cout << " -------- Player Board ----------- "

<< "X = Your ships" << endl

<< " 1 2 3 4 5 6 7 8 9 10"

<< " H = Your ship has been hit" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ "

<< " M = The enemy has missed your ships" << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

cout << " " << letters[x] << " ";

for(int z=A;z<K;z++)

cout << "| " << plyrs->info[1].player\_board[x][z]<< " ";

cout << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

// Formatting

cout << endl;

}

// Function to print the players moves to the text files

void prntTofile(struct Players \* plyrs, int play\_num, ofstream &plyr\_file)

{

//File Setup text

plyr\_file << endl

<< " ----- All Plays Made during the Game by this player -----"

<< endl << endl;

// Format the output and general layout of the game boards

char letters[10] = {'A','B','C','D','E','F','G','H','I','J'};

plyr\_file << " 1 2 3 4 5 6 7 8 9 10" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

plyr\_file << " " << letters[x] << " ";

for(int z=A;z<K;z++)

plyr\_file << "| " << plyrs->info[play\_num].enemy\_board[x][z]<< " ";

plyr\_file << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

//More game board input to the file

plyr\_file << endl << endl;

plyr\_file << " 1 2 3 4 5 6 7 8 9 10" << endl

<< " \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ " << endl << endl;

// Loop for more accurate board displays

for(int x=A;x<K;x++)

{

plyr\_file << " " << letters[x] << " ";

for(int z=A;z<K;z++)

plyr\_file << "| " << plyrs->info[play\_num].player\_board[x][z]<< " ";

plyr\_file << "|" << endl

<< " - - - - - - - - - - -"<< endl;

}

}

// Function to write the records to the binary files

void prntToBinFile(struct Players \* plyrs, fstream &file)

{

// Check to see if the file is open and available

if(!file)

cout << endl << "error" << endl;

// Write both structure records to the binary file

for(int z=0;z<plyrs->SIZE;z++)

file.write(reinterpret\_cast<char \*>(&plyrs->info[z]), sizeof(plyrs->info[z]));

//Close the Binary File

file.close();

}

// Function to print the records from the binary file

void BinFileToprnt(struct Players \* plyrs, fstream &file)

{

//Reopen the Binary file

file.open("Players Info", ios:: in | ios:: binary);

//Temporary struct array used for output

struct Playerinfo temp[2];

// Check to make sure the file is open and available

if(!file)

cout << endl << "error" << endl;

//Read the records from the Binary file

file.read(reinterpret\_cast<char \*>(&temp), sizeof(temp));

// Output all of the records

for(int count=0;count<plyrs->SIZE;count++)

{

cout << endl <<" --- Player #" << (count+1) << " Information ---"

<< endl << endl << " Name: " << temp[count].player\_name << endl;

cout << endl << " Coordinates of ships:" << endl << endl;

for(int z=0;z<plyrs->numshps;z++)

cout << " " << temp[count].ships[z] << endl;

cout << endl << " Shots Fired: " << endl << endl;

for(int z=0;z<100;z++)

cout << " " << temp[count].shts\_frd[z];

cout << endl << " Final Player Board:" << endl;

for(int z=0;z<10;z++)

{

for(int f=0;f<10;f++)

cout << " " << temp[count].player\_board[z][f];

cout << endl;

}

}

}