**War of Ships**

**CSC 17A 43950**

**Nornubari Kanabolo**

**05/02/2015**

**Introduction**

In this game there is a naval war taking place between the British Empire (User) and Germany (Enemy) in 1839. They are fighting over control of the section of the Mediterranean Sea that has a less treacherous and more lucrative trade route, which can maximize only one of the country’s wealth. The program is similar to Battleship, and allows the user to decide the size of the board and how many ships should be used. The user chooses the spot on the board where he thinks the enemy ship is at by entering a row then a column within the bounds of the board. A hit ship is denoted by a “+.” A missed ship is denoted by an “X.” The spots of the users ships are denoted by an “O.”This continues until either the user or enemy has destroyed all the ships on their opponent’s board. The game then ends and displays the winner of the war.

**Summary**

Lines of code: 309

Comment Lines: 41

White spaces: 37

This project used most of the concepts and constructs we learned so far in this course. It uses a series of for loops, while, if-else-if statements and do while loops to control the moves, points, and winner of the game. The game uses structures for the board and players which had the components for the size of the board and the user and enemy ships, respectively. It also used functions with pass by references and dynamically allocated arrays to keep displaying the position of the two fleets’ ships during the war. It was difficult to figure out how to exactly nest some of the for loops since each one has a specific placement on the board.

**Pseudo Code**

**Game board and user input functions**

***Call begin function***

***Get board row and column size and number of ships from user***

***Print empty game board***

***Initialize i***

***For i<a.columns display empty space***

***Initialize i***

***Initialize j***

***For i<a.rows display empty space***

***i++***

***For j<a.columns display border section***

***j++***

***Initialize i***

***For i<a.columns display end of border section***

***i++***

***Dynamically allocate memory for arrays of the players***

***Call Player structure***

***Declare \*\*arr***

***Initialize variables***

***Set random number seed***

***Set spots where each ship will be blank***

***Do***

***Input tRow***

***while(tRow <= 0 ||tRow > b.rows)***

***Check if position is valid***

***Input tCol***

***while(tRow <= 0 ||tRow > b.columns)***

***check if position is valid***

***Initialize i***

***For i < b.ships***

***Do***

***Set random positions for enemy***

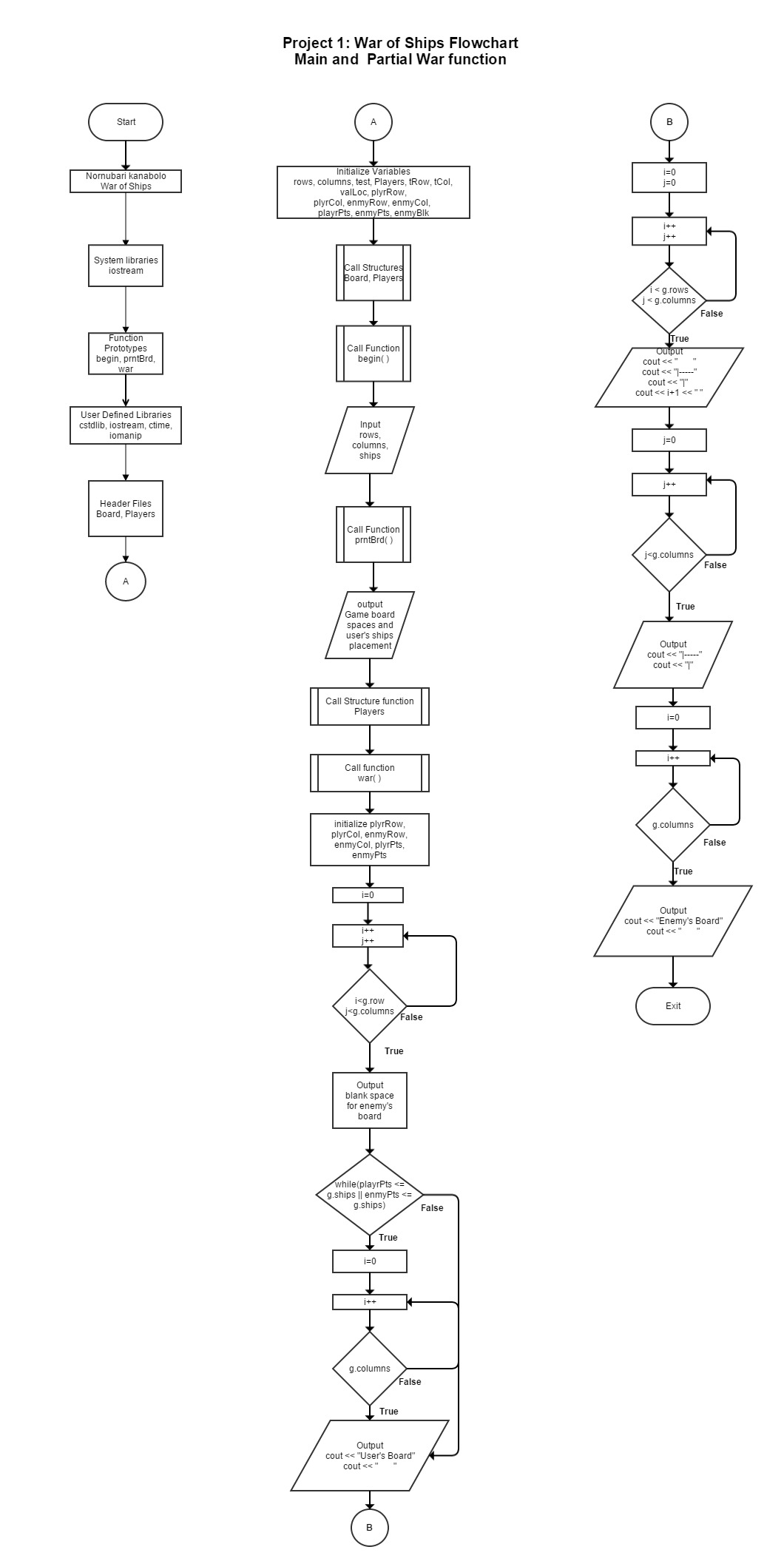
***while(valLoc == false)***

***Validate positions***

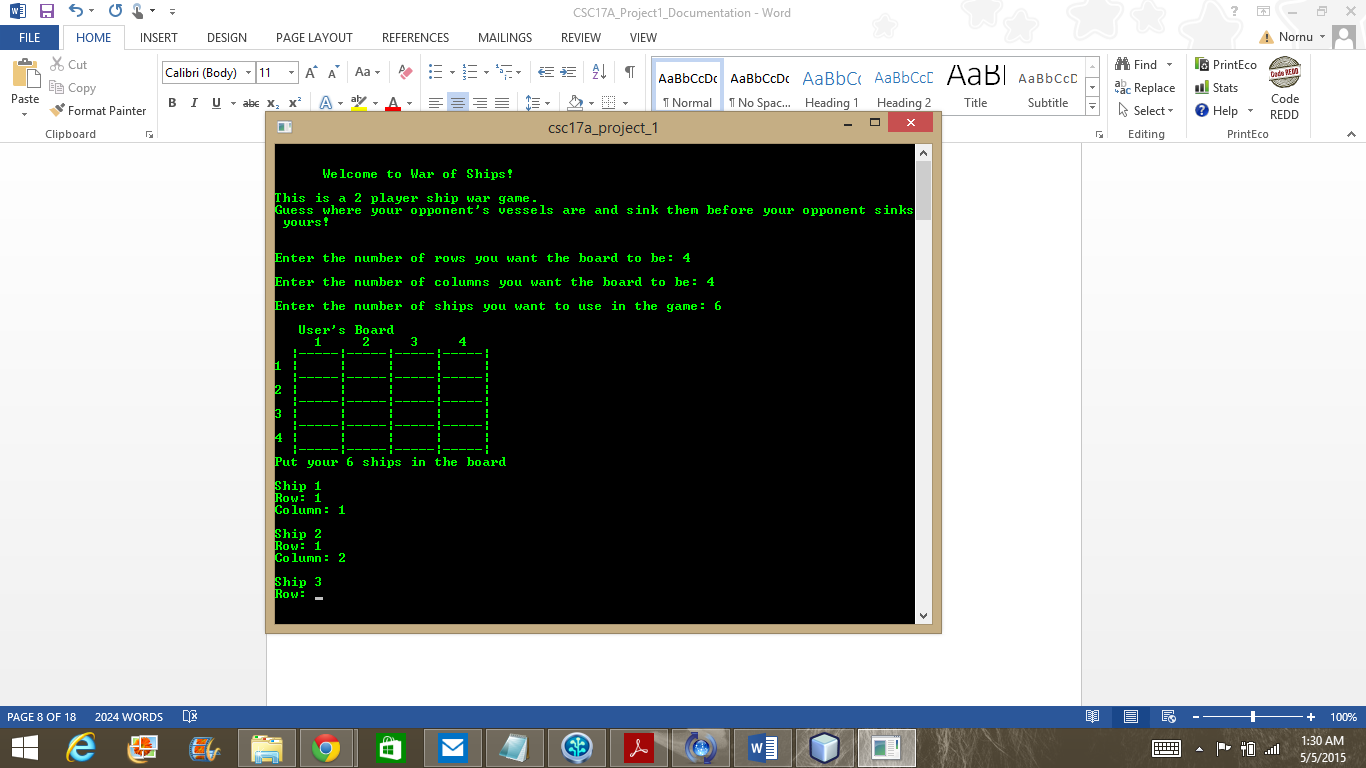
***Return arr***

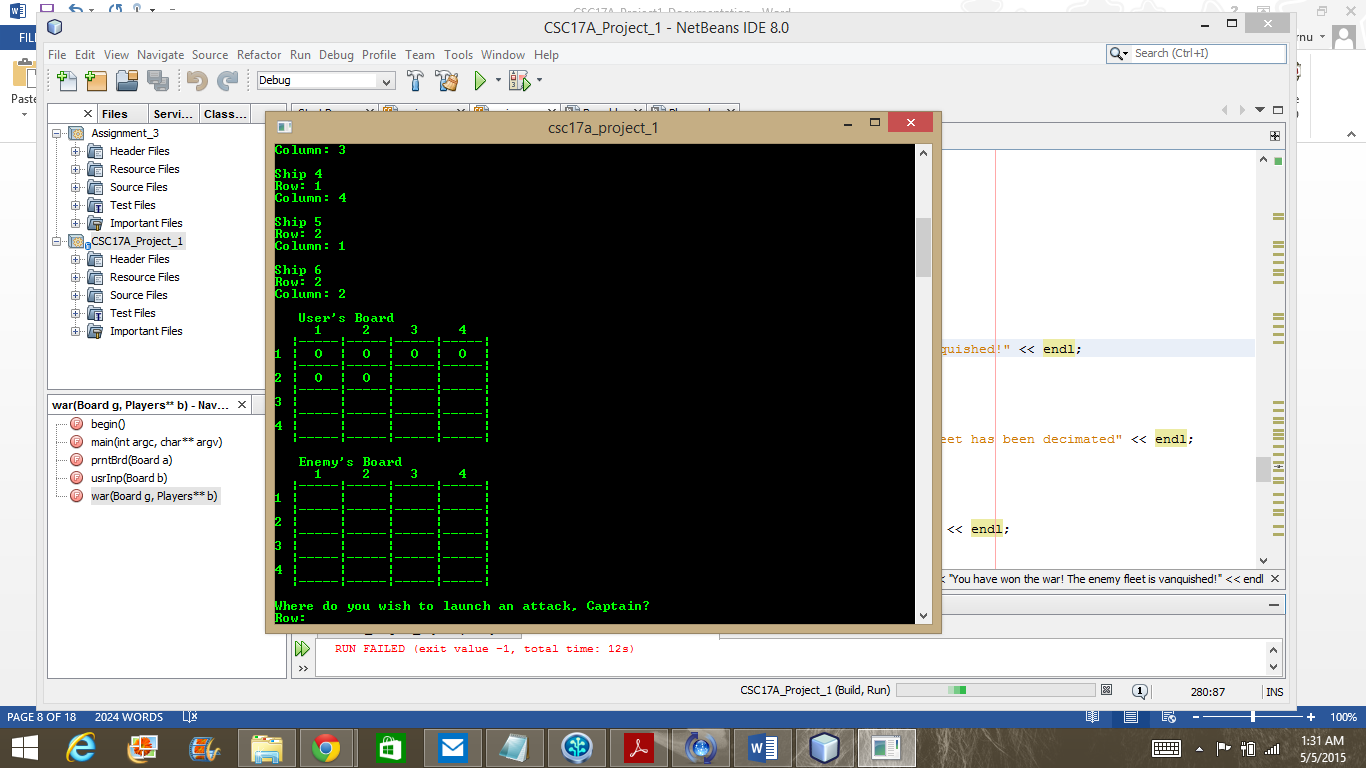
|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Location |
| int | rows | Number of rows on board | In Board header file |
|  | columns | Number of columns on board | In Board header file |
|  | ships | Number of ships on board | In Board header file |
|  | tRow | temporary value for the row | In usrInp function |
|  | tCol | temporary value for the column | In usrInp function |
|  | plyrRow | player rows | War function |
|  | plyrCol | player columns | War function |
|  | enmyRow | enemy rows | War function |
|  | enmyCol | enemy rows | War function |
|  | playrPts | counter for each ship destroyed by user | War function |
|  | enmyPts | counter for each ship destroyed by enemy | War function |
| char | test | Blank space | In prntBrd function |
|  | plyrBrd | Array with all the player tokens | In Players header file |
|  | enmyBrd | Array with all the enemy’s tokens | In Players header file |
|  | enmyBlk | Blank board for the enemy | In usrInp function |
| bool | valLoc | check for valid input | In usrInp function |

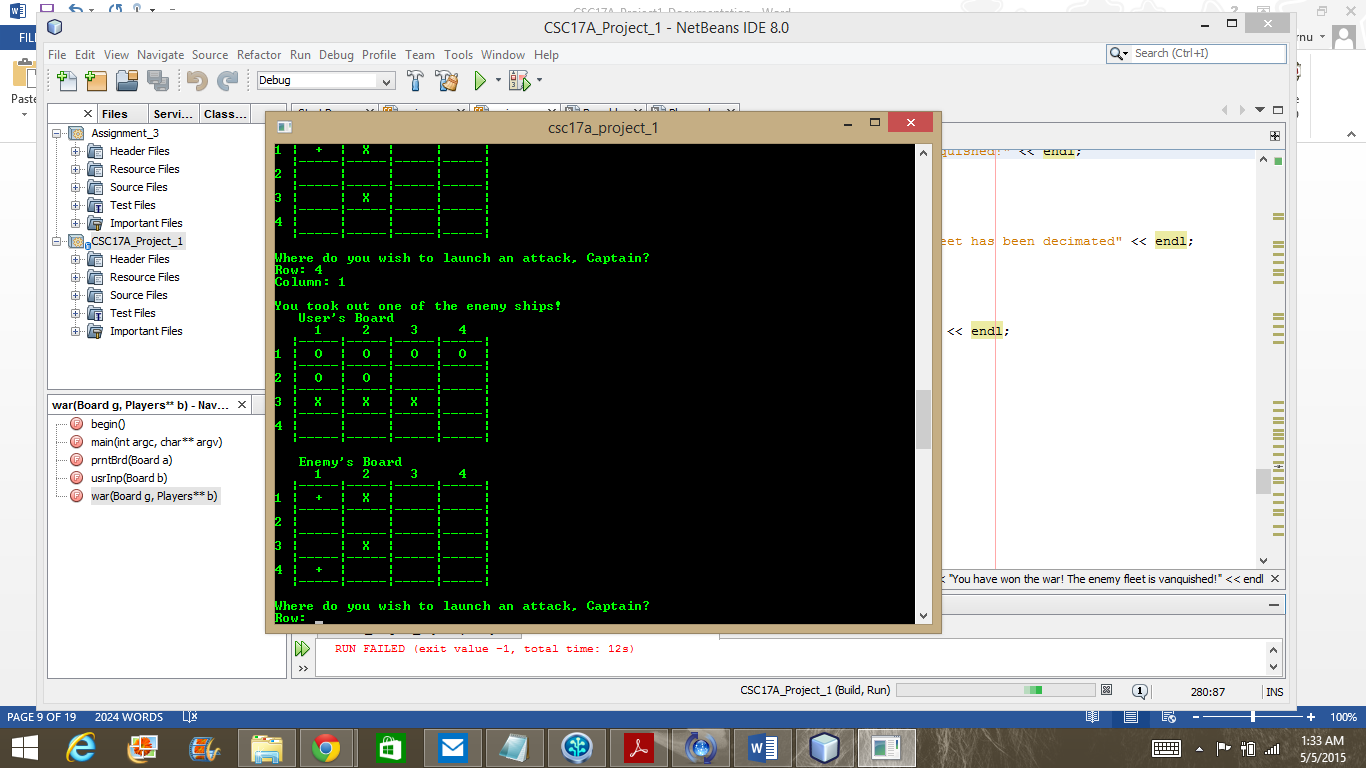
|  |  |  |
| --- | --- | --- |
| Chapter | Constructs/Syntax | Location |
| 2 | Equality and relational operators  (&&, ||, ==,>=,>,<=,<,-,+) | Comparing positions of ships |
|  | bool | Check validity of positions |
|  | If | Where ship positions are compared to board positions |
|  | If else | Where ship positions are compared to board positions |
|  | While | Checks if points exceed the number of ships |
|  | Do-while | Checks if points are in boundaries of number of ships |
|  | for | Prints out boards, ships, and their positions |
| 7 | Array | Board and player ships arrays |
| 11 | Dynamic array | References the array of each player |
|  | Structure | In header Board, and Player file |

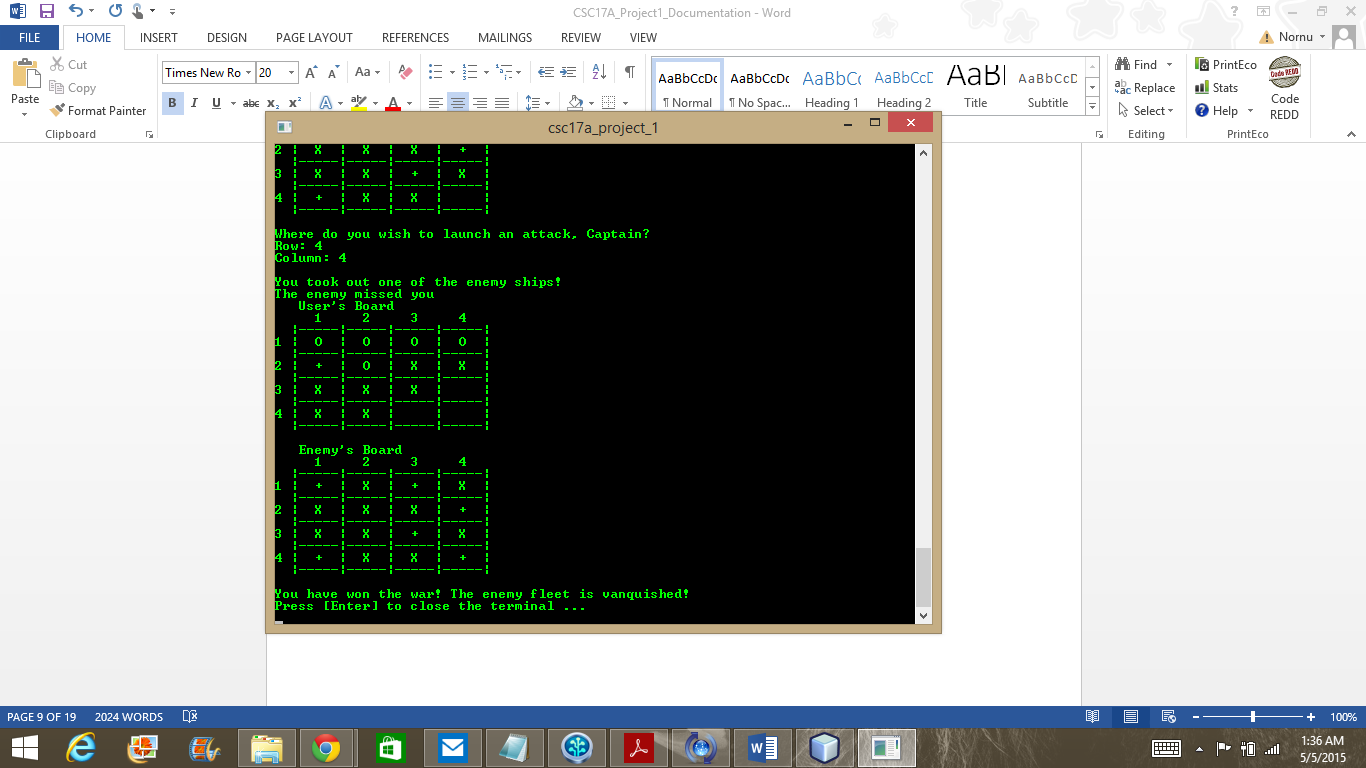


**Example Code**









**Program Code**

**Board Header File**

|  |
| --- |
|  |
| #ifndef BOARD\_H |
|  | #define BOARD\_H |
|  | #include <cstdlib> |
|  | #include <iostream> |
|  | #include <iomanip> |
|  | #include <ctime> |
|  |  |
|  | struct Board |
|  | { |
|  | int rows; //The number of rows in the board |
|  | int columns; //The number of columns in the board |
|  | int ships; //The number of ships in the board |
|  | }; |
|  |  |
|  | #endif /\* BOARD\_H \*/ |
|  |  |

**Players Header File**

|  |
| --- |
| #ifndef PLAYERS\_H |
|  | #define PLAYERS\_H |
|  | #include <cstdlib> |
|  | #include <iostream> |
|  | #include <iomanip> |
|  | #include <ctime> |
|  |  |
|  | struct Players |
|  | { |
|  | char playrBd; //Array with all the player tokens |
|  | char enemyBd; //Array with all the enemy tokens |
|  | }; |
|  |  |
|  | #endif /\* PLAYERS\_H \*/ |

**Main**

|  |
| --- |
| /\* |
|  | \* File: main.cpp |
|  | \* Author: Nornubari Kanabolo |
|  | \* Project 1 - War of Ships |
|  | \*/ |
|  |  |
|  | //User Defined Libraries |
|  | #include <cstdlib> |
|  | #include <iostream> |
|  | #include <iomanip> |
|  | #include <ctime> |
|  | #include "Board.h" |
|  | #include "Players.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | //Function Prototypes |
|  |  |
|  | //Prints the beginning to the game |
|  | void begin(); |
|  | //Prints the starting board |
|  | void prntBrd(Board); |
|  | //Gets the ships position in the board |
|  | Players \*\*usrInp(Board); |
|  | //Where the war is |
|  | void war(Board, Players\*\*); |
|  |  |
|  | //Execution Begins Here |
|  |  |
|  | int main(int argc, char\*\* argv) { |
|  | //Declare structures |
|  | struct Board game; //Structure that holds the board dimensions |
|  | struct Players \*\*boards; //Structure that composes the players boards |
|  |  |
|  | //Call beginning function |
|  | begin(); |
|  |  |
|  | //Getting input to make the board |
|  | cout << endl << "Enter the number of rows you want the board to be: "; |
|  | cin >> game.rows; |
|  | cout << endl << "Enter the number of columns you want the board to be: "; |
|  | cin >> game.columns; |
|  | cout << endl << "Enter the number of ships you want to use in the game: "; |
|  | cin >> game.ships; |
|  |  |
|  | //Calls the function that prints the board |
|  | prntBrd(game); |
|  |  |
|  | //Allocate new array for resizing board |
|  | boards = new Players\*[game.rows]; |
|  | for(int i = 0; i < game.rows; ++i) |
|  | boards[i] = new Players[game.columns]; |
|  |  |
|  | //Retrieve the user's input |
|  | boards = usrInp(game); |
|  |  |
|  | //Calling the war function |
|  | war(game, boards); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | void begin() |
|  | { |
|  | //Output the introduction |
|  | cout << endl << endl; |
|  | cout << right << setw(30) << "Welcome to War of Ships!" << endl ; |
|  | cout << endl << "This is a 2 player ship war game.\n"; |
|  | cout << "Guess where your opponent's vessels are and sink them before your opponent sinks yours!\n"; |
|  | cout << endl; |
|  | } |
|  |  |
|  |  |
|  | void prntBrd(Board a) |
|  | { |
|  | //Prints the empty board |
|  | //Declare variables |
|  | char test = ' '; |
|  |  |
|  | cout << endl << " User's Board" << endl; |
|  | for(int i = 0; i < a.columns; i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < a.rows; i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < a.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < a.columns; j++) |
|  | { |
|  | cout << "| " << test << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < a.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  |  |
|  | } |
|  |  |
|  | Players \*\*usrInp(Board b) |
|  | { |
|  | //Declare and allocate dynamic array |
|  | Players \*\*arr; |
|  | arr = new Players\*[b.rows]; |
|  | for(int i = 0; i < b.rows; ++i) |
|  | arr[i] = new Players[b.columns]; |
|  |  |
|  | int tRow = 0; //temporary value for the row |
|  | int tCol = 0; //temporary value for the column |
|  | bool valLoc = true;//check for valid input |
|  |  |
|  | srand(time(0)); |
|  |  |
|  | //Set array spaces to blank spaces |
|  | for(int i = 0; i < b.rows; i++) |
|  | { |
|  | for(int j = 0; j < b.columns; j++) |
|  | { |
|  | arr[i][j].playrBd = ' '; |
|  | arr[i][j].enemyBd = ' '; |
|  | } |
|  | } |
|  |  |
|  | //Get locations for user's ships |
|  | cout << "Put your "<< b.ships << " ships in the board" << endl; |
|  |  |
|  | for(int i = 0; i < b.ships; i++) |
|  | { |
|  | do |
|  | { |
|  | cout << endl << "Ship " << i+1 << endl; |
|  | cout << "Row: "; |
|  | cin >> tRow; |
|  | while(tRow <= 0 ||tRow > b.rows) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Row: "; |
|  | cin >> tRow; |
|  | } |
|  | cout << "Column: "; |
|  | cin >> tCol; |
|  | while(tCol <= 0 ||tCol > b.columns) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Row: "; |
|  | cin >> tCol; |
|  | } |
|  |  |
|  | //Check if position is valid |
|  | if(arr[tRow-1][tCol-1].playrBd == 'O') |
|  | { |
|  | valLoc = false; |
|  | cout << endl << "You already have a ship there" << endl; |
|  | } |
|  | else |
|  | { |
|  | arr[tRow-1][tCol-1].playrBd = 'O'; |
|  | } |
|  | }while(valLoc == false); |
|  | } |
|  |  |
|  | //Enemy randomly selects ship position |
|  | for(int i = 0; i < b.ships; i++) |
|  | { |
|  | do |
|  | { |
|  | tRow = (rand() % b.rows); |
|  | tCol = (rand() % b.columns); |
|  |  |
|  | if(arr[tRow][tCol].enemyBd == 'O') |
|  | { |
|  | valLoc = false; |
|  | } |
|  | else |
|  | { |
|  | arr[tRow][tCol].enemyBd = 'O'; |
|  | } |
|  | }while(valLoc == false); |
|  | } |
|  |  |
|  |  |
|  | return arr;//Return array with position of ship |
|  | } |
|  |  |
|  | //Function where all moves and calculations take place |
|  | void war(Board g, Players \*\*b) |
|  | { |
|  | //Declare variables |
|  | int plyrRow = 0; //Initialize player rows |
|  | int plyrCol = 0; //Initialize player columns |
|  |  |
|  | int enmyRow = 0; //Initialize enemy rows |
|  | int enmyCol = 0; //Initialize enemy columns |
|  |  |
|  | int playrPts = 0; //Initialize counter for each ship destroyed by user |
|  | int enmyPts = 0; //Initialize counter for each ship destroyed by enemy |
|  | char enmyBlk[g.rows][g.columns];//Blank board for the enemy |
|  |  |
|  | //Setting enmyBlk array with blank spaces |
|  | for(int i = 0; i < g.rows; i++) |
|  | { |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | enmyBlk[i][j] = ' '; |
|  | } |
|  | } |
|  |  |
|  | //Loop to make the moves |
|  | while(playrPts <= g.ships || enmyPts <= g.ships) |
|  | { |
|  | cout << endl << " User's Board" << endl; |
|  | for(int i = 0; i < g.columns; i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < g.rows; i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "| " << b[i][j].playrBd << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl << endl; |
|  |  |
|  | cout << " Enemy's Board" << endl; |
|  | for(int i = 0; i < g.columns; i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < g.rows; i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "| " << enmyBlk[i][j] << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < g.columns; j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  |  |
|  | //Check for the winner |
|  | if(playrPts == g.ships) |
|  | { |
|  | cout << endl << "You have won the war! The enemy fleet is vanquished!" << endl; |
|  | break; |
|  | } |
|  | else if(enmyPts == g.ships) |
|  | { |
|  | cout << endl << "You have been defeated by the enemy. Your fleet has been decimated" << endl; |
|  | break; |
|  | } |
|  |  |
|  | //Get coordinates to attack the enemy |
|  | cout << endl << "Where do you wish to launch an attack, Captain?" << endl; |
|  | cout << "Row: "; |
|  | cin >> plyrRow; |
|  | while(plyrRow <= 0 ||plyrRow > g.rows) |
|  | { |
|  | cout << endl << "Invalid input. Try again" << endl; |
|  | cout << "Row: "; |
|  | cin >> plyrRow; |
|  | } |
|  | cout << "Column: "; |
|  | cin >> plyrCol; |
|  |  |
|  | //Confirm user's move |
|  | while(plyrCol <= 0 || plyrCol > g.columns) |
|  | { |
|  | cout << endl << "Invalid input. Try again" << endl; |
|  | cout << "Column: "; |
|  | cin >> plyrCol; |
|  | } |
|  |  |
|  | if(b[plyrRow-1][plyrCol-1].enemyBd == ' ') |
|  | { |
|  | b[plyrRow-1][plyrCol-1].enemyBd = 'X'; |
|  | enmyBlk[plyrRow-1][plyrCol-1] = 'X'; |
|  | cout << endl << "You missed the enemy"; |
|  | } |
|  | else if(b[plyrRow-1][plyrCol-1].enemyBd == 'O') |
|  | { |
|  | b[plyrRow-1][plyrCol-1].enemyBd = '+'; |
|  | enmyBlk[plyrRow-1][plyrCol-1] = '+'; |
|  | cout << endl << "You took out one of the enemy ships!"; |
|  | playrPts++; |
|  | } |
|  | else if(b[plyrRow-1][plyrCol-1].enemyBd == 'X' || |
|  | b[plyrRow-1][plyrCol-1].enemyBd == '+') |
|  | { |
|  | cout << endl << "That spot has already been attacked. That was a waste of a turn"; |
|  | } |
|  |  |
|  | //Confirm enemy's move |
|  | enmyRow = (rand() % g.rows); |
|  | enmyCol = (rand() % g.columns); |
|  |  |
|  | if(b[enmyRow][enmyCol].playrBd == 'O') |
|  | { |
|  | b[enmyRow][enmyCol].playrBd = '+'; |
|  | cout << endl << "Noooo! You've been hit!"; |
|  | enmyPts++; |
|  | } |
|  | else if(b[enmyRow][enmyCol].playrBd == ' ') |
|  | { |
|  | b[enmyRow][enmyCol].playrBd = 'X'; |
|  | } |
|  | else if(b[enmyRow][enmyCol].enemyBd == 'X' || |
|  | b[enmyRow][enmyCol].enemyBd == '+') |
|  | { |
|  | cout << endl << "The enemy missed you"; |
|  | } |
|  | } |
|  | } |