**War of Ships**

**V. 2**

**CSC 17A 43950**

**Nornubari Kanabolo**

**06/10/2015**

**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 battleship game is an object oriented program. It uses two class objects one derived from the other and a template. In the main file it has five functions to perform different tasks. First the program ask the user for three inputs: the number of rows in the board, the number of columns and the number of ships. Then the program initializes a class object from the User class to store those values. The User class is a derived class form the base class Board. After storing the user inputs in the Board class the program runs an exception to check for errors in the inputs. Then the program prints a reference board using the user inputs and calls the functions getUser from the User class to get the positions in the board to place the ships, it places an “O” in those places. It does the same with the enemy board calling the getEnemy function but the positions are randomly selected.

The program then runs a while loop. Inside the loop a new player and enemy board is printed. The player board displays the ships and the enemy board is blind. Coordinates are requested to launch attack into the board, the player enters inputs for the rows and the columns. Then the program uses those numbers in the enemy array to check if there are any ships in that position. If there is a ship that spot is marked with the “#” symbol and a one is added to a counter called userPoints, the program keeps track of the number of ships destroyed. If the attacked space is blank an “X” is placed in that position. The game then ends and displays the winner of the war.

**Summary**

Lines of code: 591

Comment Lines: 41

White spaces: 37

This project used most of the concepts and constructs we learned in this course. It uses a series of for loops, while, if-else-if statements and do while loops to control the moves, points, constructors, destructors, accessors, and mutators and winner of the game. The game uses classes and headers 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<battle.columns display empty space***

***Initialize i***

***Initialize j***

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

***i++***

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

***j++***

***Initialize i***

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

***i++***

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

***Call User class***

***Declare battle, \*\*arr***

***Initialize variables***

***Set random number seed***

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

***Do***

***Input tempRow***

***while(tempRow <= 0 ||tempRow > battle.rows)***

***Check if position is valid***

***Input tempCol***

***while(tempRow <= 0 ||tempRow > battle.columns)***

***check if position is valid***

***Initialize i***

***For i < battle.ships***

***Do***

***Set random positions for enemy***

***while(validLoc == false)***

***Validate positions***

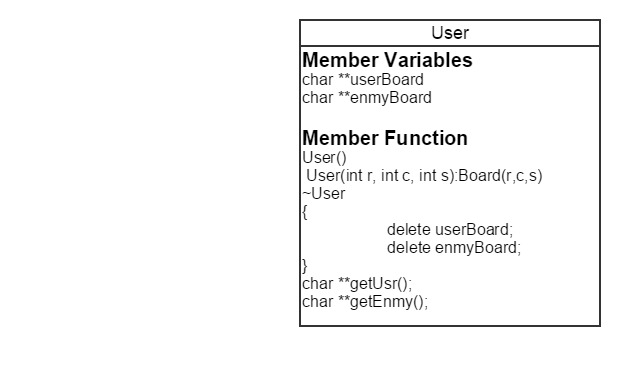
***Return arr***

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Location |
| int | rows | Number of rows on board | In Gameboard header file |
|  | columns | Number of columns on board | In Gameboard header file |
|  | ships | Number of ships on board | In Gameboard header file |
|  | setRows | Mutator to transform rows | In Gameboard header file |
|  | setCols | Mutator to transform columns | In Gameboard header file |
|  | setShips | Mutator to transform ships | In Gameboard header file |
|  | tempRow | temporary value for the row | In User.cpp |
|  | tempCol | temporary value for the column | In User.cpp |
|  | usrRow | player rows | Main function |
|  | usrColumn | player columns | Main function |
|  | enmyRow | enemy rows | Main function |
|  | enmyCol | enemy rows | Main function |
|  | usrPnts | counter for each ship destroyed by user | Main function |
|  | enmyPnts | counter for each ship destroyed by enemy | Main function |
| char | start | Blank space | In prntBrd function |
|  | usrBoard | Array with all the player tokens | In User header file |
|  | enmyBoard | Array with all the enemy’s tokens | In User header file |
|  | invisBrd | Invisible board for the enemy | Main function |
| bool | validLoc | check for valid input | Main function |

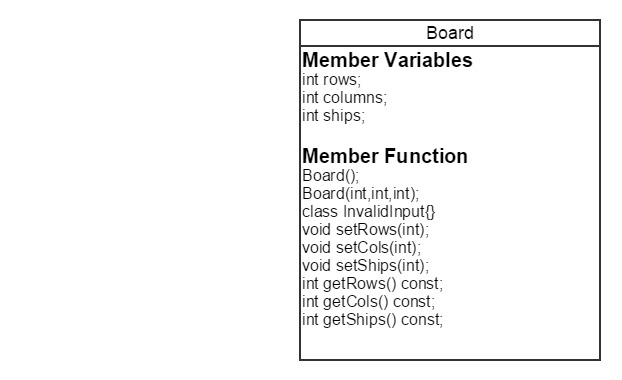
|  |  |  |
| --- | --- | --- |
| Chapter | Constructs/Syntax | Location |
| 2 | Equality and relational operators  (&&, ||, ==,>=,>,<=,<,-,+) | Comparing positions of ships |
|  | bool | Check validity of positions |
|  | If | Where ship positions are compared to board positions |
|  | If else | Where 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 |
|  | Class | Gameboard, User, and Victor header  Gameboard, User, and Victor souce files |

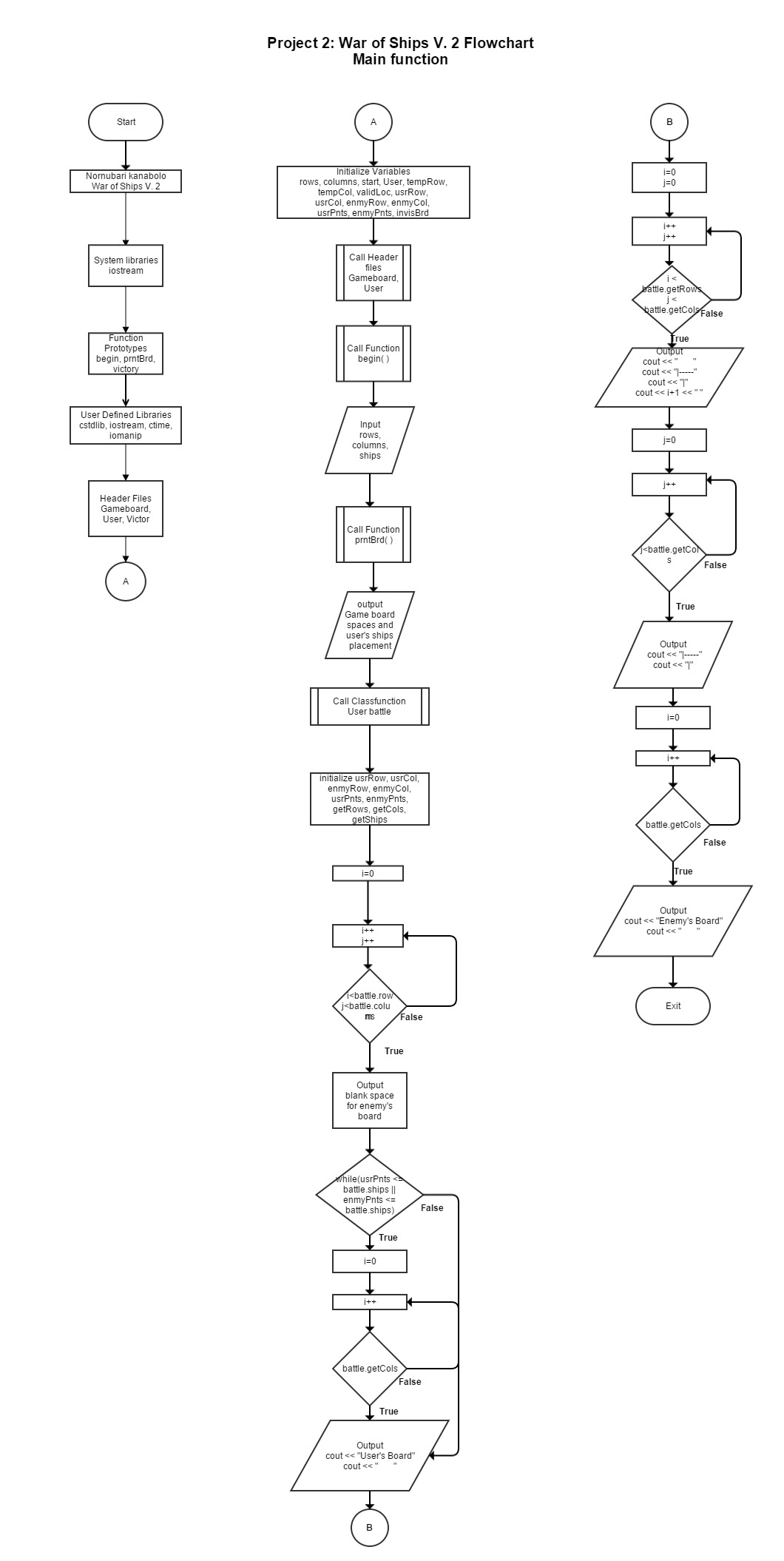
**Class Diagrams**

**User**

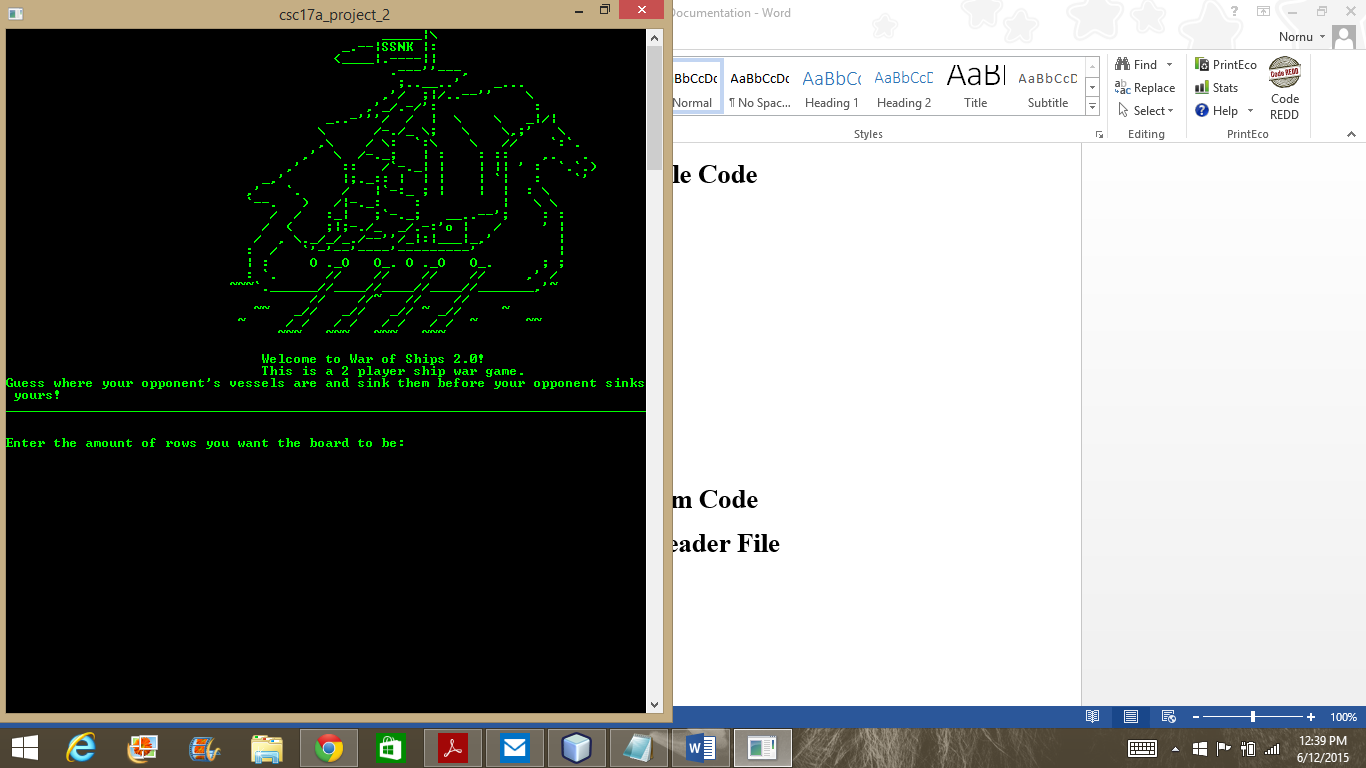


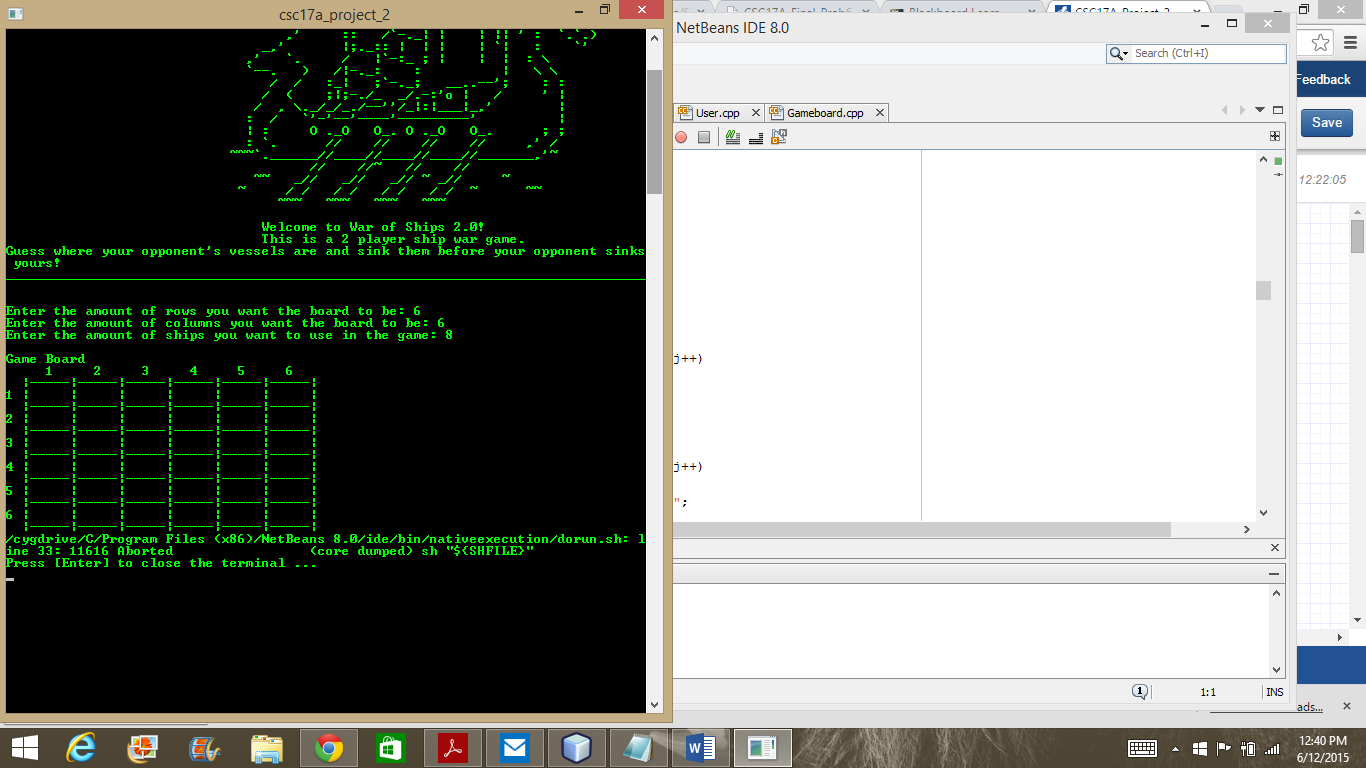
**Gameboard**

****



**Example Code**





Was unable to get my code to work, but I could not find errors in my code. It seems to be right.

**Program Code**

**Gameboard Header File**

|  |
| --- |
|  |
| |  | | --- | | #ifndef GAMEBOARD\_H | |  | #define GAMEBOARD\_H | |  |  | |  | class Board | |  | { | |  | private: | |  | int rows; //Amount of rows in the board | |  | int columns; //Amount of columns in the board | |  | int ships; //Amount of ships in the board | |  | public: | |  | //Constructor | |  | Board(); //Default constructor | |  | Board(int,int,int); //Constructor using input | |  | //Exceptions | |  | class InvalidInput{}; | |  |  | |  | //Mutators | |  | void setRows(int); | |  | void setCols(int); | |  | void setShips(int); | |  |  | |  | //Accessors | |  | int getRows() const; | |  | int getCols() const; | |  | int getShips() const; | |  |  | |  | }; | |  |  | |  |  | |  | #endif /\* GAMEBOARD\_H \*/ | |  |

**Gameboard Source file**

|  |
| --- |
| #include "Gameboard.h" |
|  |  |
|  | Board::Board() |
|  | { |
|  | rows = 0; |
|  | columns = 0; |
|  | ships = 0; |
|  | } |
|  | Board::Board(int ro,int co,int sh) |
|  | { |
|  | rows = ro; |
|  | columns = co; |
|  |  |
|  | //Exception validation |
|  | if(sh < (ro\*co)) |
|  | { |
|  | ships = sh; |
|  | } |
|  | else |
|  | throw InvalidInput(); |
|  | } |
|  |  |
|  | //Mutators |
|  | void Board::setRows(int ro) |
|  | { |
|  | rows = ro; |
|  | } |
|  | void Board::setCols(int co) |
|  | { |
|  | columns = co; |
|  | } |
|  | void Board::setShips(int sh) |
|  | { |
|  | //Exception validation |
|  | if( sh < (rows \* columns)) |
|  | { |
|  | ships = sh; |
|  | } |
|  | else |
|  | throw InvalidInput(); |
|  | } |
|  |  |
|  | //Accessors |
|  | int Board::getRows() const{ return rows;} |
|  | int Board::getCols() const { return columns;} |
|  | int Board::getShips() const { return ships;} |

**User Header File**

|  |
| --- |
| #ifndef USER\_H |
|  | #define USER\_H |
|  |  |
|  | #include "Gameboard.h" |
|  |  |
|  | class User : public Board |
|  | { |
|  | private: |
|  | char \*\*userBoard; //Array with all the player tokens |
|  | char \*\*enmyBoard; //Array with all the enemy tokens |
|  | public: |
|  | User(){} |
|  | User(int r, int c, int s):Board(r, c, s){} |
|  |  |
|  | ~User() |
|  | { |
|  | delete [] userBoard; |
|  | delete [] enmyBoard; |
|  | } |
|  |  |
|  | char \*\*getUsr(); |
|  | char \*\*getEnmy(); |
|  | }; |
|  |  |
|  | #endif /\* USER\_H \*/ |

**User Source File**

|  |
| --- |
| #include <iostream> |
|  | #include <cstdlib> |
|  | #include "User.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | //Ask for user input and put in an array |
|  | char \*\*User::getUsr() |
|  | { |
|  | //Getting input form the Board object |
|  | int rows = getRows(); |
|  | int columns = getCols(); |
|  | int ships = getShips(); |
|  |  |
|  | //Resize pointer array |
|  | userBoard = new char\*[rows]; |
|  | for(int i = 0; i < rows; ++i) |
|  | { |
|  | userBoard[i] = new char[columns]; |
|  | } |
|  |  |
|  | int tempRow = 0; //Temporary int for row |
|  | int tempCol = 0; //Temporary int for column |
|  | bool validLoc = true;//Bool to check for valid input |
|  |  |
|  |  |
|  |  |
|  | //Put array spaces to ' ' |
|  | for(int i = 0; i < rows; i++) |
|  | { |
|  | for(int j = 0; j < columns; j++) |
|  | { |
|  | userBoard[i][j] = ' '; |
|  | } |
|  | } |
|  |  |
|  | //Get places for user's ships |
|  | cout << endl << "Put your "<< ships << " ships on the board" << endl; |
|  |  |
|  | for(int i = 0; i < ships; i++) |
|  | { |
|  | do |
|  | { |
|  | cout << endl << "Ship " << i+1 << endl; |
|  | cout << "Row: "; |
|  | cin >> tempRow; |
|  | while(tempRow <= 0 ||tempRow > rows) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Row: "; |
|  | cin >> tempRow; |
|  | } |
|  | cout << "Column: "; |
|  | cin >> tempCol; |
|  | while(tempCol <= 0 ||tempCol > columns) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Column: "; |
|  | cin >> tempCol; |
|  | } |
|  |  |
|  | //Check if space is valid |
|  | if(userBoard[tempRow-1][tempCol-1] == 'O') |
|  | { |
|  | validLoc = false; |
|  | cout << endl << "You already have a ship there!" << endl; |
|  | } |
|  | else |
|  | { |
|  | userBoard[tempRow-1][tempCol-1] = 'O'; |
|  | } |
|  | }while(validLoc == false); |
|  | } |
|  |  |
|  | return userBoard; |
|  | } |
|  |  |
|  | //Generates random input and stores in an array |
|  | char \*\*User::getEnmy() |
|  | { |
|  | //Getting input form the Board object |
|  | int rows = getRows(); |
|  | int columns = getCols(); |
|  | int ships = getShips(); |
|  |  |
|  | //Resize pointer array |
|  | enmyBoard = new char\*[rows]; |
|  | for(int i = 0; i < rows; ++i) |
|  | { |
|  | enmyBoard[i] = new char[columns]; |
|  | } |
|  |  |
|  | //Setting all the array spaces to ' ' |
|  | for(int i = 0; i < rows; i++) |
|  | { |
|  | for(int j = 0; j < columns; j++) |
|  | { |
|  | enmyBoard[i][j] = ' '; |
|  | } |
|  | } |
|  |  |
|  | int tempRow = 0; //Temp int for the row |
|  | int tempCol = 0; //Temp int for the column |
|  | bool validLoc = true;//Check if input valid |
|  |  |
|  |  |
|  | //Enemy randomly selects its ships position |
|  | for(int i = 0; i < ships; i++) |
|  | { |
|  | do |
|  | { |
|  | tempRow = (rand() % rows); |
|  | tempCol = (rand() % columns); |
|  |  |
|  | if(enmyBoard[tempRow][tempCol] == 'O') |
|  | { |
|  | validLoc = false; |
|  | } |
|  | else |
|  | { |
|  | enmyBoard[tempRow][tempCol] = 'O'; |
|  | } |
|  | }while(validLoc == false); |
|  | } |
|  |  |
|  | return enmyBoard; |
|  | } |

**Victor Header File**

|  |
| --- |
| #ifndef VICTOR\_H |
|  | #define VICTOR\_H |
|  |  |
|  | class Victor |
|  | { |
|  | private: |
|  | int usrPnts; |
|  | int enmyPnts; |
|  | public: |
|  | Victor() |
|  | { |
|  | usrPnts = 0; |
|  | enmyPnts = 0; |
|  | } |
|  | Victor(int us, int en){ usrPnts = us; enmyPnts = en;} |
|  | void setUsrPnts(int us){usrPnts = us;} |
|  | void setEnmyPnts(int en){enmyPnts = en;} |
|  | void getRslt(); |
|  | }; |
|  |  |
|  |  |
|  | #endif /\* VICTOR\_H \*/ |

**Victor Source File**

|  |
| --- |
| #include <iostream> |
|  | #include "Victor.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | void Victor::getRslt() |
|  | { |
|  |  |
|  | if(usrPnts > enmyPnts) |
|  | { |
|  | cout<<"You sunk " << enmyPnts << " enemy ships" << endl; |
|  | cout<<"The enemy only sunk " << usrPnts << " of your ships" << endl; |
|  | } |
|  | if(usrPnts < enmyPnts) |
|  | { |
|  | cout<<"You only sunk " << enmyPnts << " enemy ships" << endl; |
|  | cout<<"The enemy only sunk " << usrPnts << " of your ships" << endl; |
|  | } |
|  | } |

**Main**

|  |
| --- |
| /\* |
|  | \* File: main.cpp |
|  | \* Author: Nornubari Kanabolo |
|  | \* CSC 17A Project 2 - War of Ships V. 2 |
|  | \*/ |
|  |  |
|  |  |
|  |  |
|  | using namespace std; |
|  | //User Defined Libraries |
|  | #include <iomanip> |
|  | #include <cstdlib> |
|  | #include <ctime> |
|  | #include <iostream> |
|  | #include <fstream> |
|  |  |
|  | //Classes |
|  | #include "User.h" |
|  | #include "Victor.h" |
|  |  |
|  | //Function prototypes |
|  |  |
|  | //Prints the beginning to the game |
|  | void begin(); |
|  | //Prints the starting board |
|  | void prntBrd(User); |
|  | //Gets input and prints user board |
|  | void userBoard(int, int, char\*\*); |
|  | //Gets input and prints enemy board |
|  | void enmyBoard(int, int, char\*\*); |
|  | //Checks for a victor |
|  | bool victory(int, int, int); |
|  |  |
|  | //Execution begins here |
|  | int main(int argc, char\*\* argv) { |
|  | //Declare variables |
|  | int row = 0; //User input for rows |
|  | int column = 0; //User input for columns |
|  | int ship = 0; //User input for ships amount |
|  | int usrRow = 0; //User rows |
|  | int usrColumn = 0; //User columns |
|  | int enmyRow = 0; //Enemy rows |
|  | int enmyCol = 0; //Enemy columns |
|  | char \*\*usr; //Pointer array for User board |
|  | char \*\*enemy; //Pointer array for enemy board |
|  | int usrPnts = 0; //Counter for number of ships sunk by user |
|  | int enmyPnts = 0; //Counter for number of ships sunk by enemy |
|  |  |
|  | //Set random number seed |
|  | srand(time(0)); |
|  |  |
|  | //Calling the begin function |
|  | begin(); |
|  |  |
|  | //Getting input to make the board |
|  | cout << endl << "Enter the amount of rows you want the board to be: "; |
|  | cin >> row; |
|  | cout << "Enter the amount of columns you want the board to be: "; |
|  | cin >> column; |
|  | cout << "Enter the amount of ships you want to use in the game: "; |
|  | cin >> ship; |
|  |  |
|  | //Initializing class object |
|  | User battle; |
|  |  |
|  | //Declaring class object with exceptions |
|  | try |
|  | { |
|  | battle.setRows(row); |
|  | battle.setCols(column); |
|  | battle.setShips(ship); |
|  | } |
|  | catch(Board::InvalidInput) |
|  | { |
|  | cout << endl << "You can't have that many ships. Enter a valid amount" << endl; |
|  | cout << "Please enter how many ships you want: "; |
|  | cin >> ship; |
|  | battle.setShips(ship); |
|  | } |
|  |  |
|  | //Calls the function that prints the reference board |
|  | prntBrd(battle); |
|  |  |
|  | //Initializing the blind board for the enemy to hide the real board |
|  | char invisBrd[battle.getRows()][battle.getCols()]; |
|  |  |
|  | //Resizing the pointer array to hold the user board |
|  | usr = new char\*[battle.getRows()]; |
|  | for(int i = 0; i < battle.getRows(); ++i) |
|  | { |
|  | usr[i] = new char[battle.getCols()]; |
|  | } |
|  | //Calling the getUsr function from the class User and store the value into the user pointer array |
|  | usr = battle.getUsr(); |
|  |  |
|  | //Resizing the pointer array to hold the enemy board |
|  | enemy = new char\*[battle.getRows()]; |
|  | for(int i = 0; i < battle.getRows(); ++i) |
|  | { |
|  | enemy[i] = new char[battle.getCols()]; |
|  | } |
|  | //Calling the getEnmy function from the class User and store the value into the enemy pointer array |
|  | enemy = battle.getEnmy(); |
|  |  |
|  | //Setting invisBrd array with empty spaces |
|  | for(int i = 0; i < battle.getRows(); i++) |
|  | { |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | invisBrd[i][j] = ' '; |
|  | } |
|  | } |
|  |  |
|  | //Move loop |
|  | while(usrPnts <= battle.getShips() || enmyPnts <= battle.getShips()) |
|  | { |
|  | userBoard(battle.getRows(), battle.getCols(), usr); |
|  |  |
|  | //Printing the user board |
|  | cout << endl << "User Board" << endl; |
|  | for(int i = 0; i < battle.getCols(); i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < battle.getRows(); i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "| " << usr[i][j] << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl << endl; |
|  |  |
|  | enmyBoard(battle.getRows(), battle.getCols(), enemy); |
|  |  |
|  | //Printing the enemy board |
|  | cout << "Enemy Board" << endl; |
|  | for(int i = 0; i < battle.getCols(); i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < battle.getRows(); i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "| " << invisBrd[i][j] << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < battle.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  |  |
|  | //Finding victor by calling the victory function |
|  | if(victory(usrPnts, enmyPnts, battle.getShips()) == true) |
|  | { |
|  | break; |
|  | } |
|  |  |
|  | //Getting coordinates to attack |
|  | cout << endl << "What are the coordinates to launch an attack ?" << endl; |
|  | cout << "Row: "; |
|  | cin >> usrRow; |
|  | while(usrRow <= 0 ||usrRow > battle.getRows()) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Row: "; |
|  | cin >> usrRow; |
|  | } |
|  | cout << "Column: "; |
|  | cin >> usrColumn; |
|  | while(usrColumn <= 0 || usrColumn > battle.getCols()) |
|  | { |
|  | cout << endl << "Invalid input!" << endl; |
|  | cout << "Column: "; |
|  | cin >> usrColumn; |
|  | } |
|  |  |
|  | //Validating user moves |
|  | if( enemy[usrRow-1][usrColumn-1] == ' ') |
|  | { |
|  | enemy[usrRow-1][usrColumn-1] = 'X'; |
|  | invisBrd[usrRow-1][usrColumn-1] = 'X'; |
|  | cout << endl << "You missed!"; |
|  | } |
|  | else if( enemy[usrRow-1][usrColumn-1] == 'O') |
|  | { |
|  | enemy[usrRow-1][usrColumn-1] = '#'; |
|  | invisBrd[usrRow-1][usrColumn-1] = '#'; |
|  | cout << endl << "Great job, captain!You hit one"; |
|  | usrPnts++; |
|  | } |
|  | else if( enemy[usrRow-1][usrColumn-1] == 'X' || enemy[usrRow-1][usrColumn-1] == '#') |
|  | { |
|  | cout << endl << "Captain, we chose that spot already. Get your head in the game!"; |
|  | } |
|  |  |
|  | //Getting randomly generated input for the enemy moves |
|  | enmyRow = (rand() % battle.getRows()); |
|  | enmyCol = (rand() % battle.getCols()); |
|  |  |
|  | //Validating enemy moves |
|  | if( usr[enmyRow][enmyCol] == 'O') |
|  | { |
|  | usr[enmyRow][enmyCol] = '#'; |
|  | cout << endl << "We've been hit, Captain!"; |
|  | enmyPnts++; |
|  | } |
|  | else if( usr[enmyRow][enmyCol] == ' ') |
|  | { |
|  | usr[enmyRow][enmyCol] = 'X'; |
|  | } |
|  | else if(enemy[enmyRow][enmyCol] == 'X' || |
|  | enemy[enmyRow][enmyCol] == '#') |
|  | { |
|  | cout << endl << "They missed, make them pay for their mistakes"; |
|  | } |
|  | } |
|  |  |
|  | cout << endl << "The crew thanks yee for the leadership you have shown us. Maybe our paths will cross again one day." << endl; |
|  |  |
|  | //Template |
|  | //Victor result(usrPnts, enmyPnts); |
|  | //result.getRslt(); |
|  |  |
|  | //Delete user pointer arrays |
|  | for (int i = 0; i < battle.getCols(); i++) |
|  | { |
|  | delete[] usr[i]; |
|  | } |
|  | delete [] usr; |
|  | //Delete enemy pointer arrays |
|  | for (int i = 0; i < battle.getCols(); i++) |
|  | { |
|  | delete[] enemy[i]; |
|  | } |
|  | delete [] enemy; |
|  | return 0; |
|  | } |
|  | void begin() |
|  | { |
|  | string line; |
|  | ifstream begin ("begin.txt"); |
|  |  |
|  | if (begin.is\_open()) |
|  | { |
|  | while ( getline (begin,line) ) |
|  | { |
|  | cout << line << '\n'; |
|  | } |
|  | begin.close(); |
|  | } |
|  | else cout << "Unable to open file"; |
|  | } |
|  |  |
|  | void prntBrd(User a) |
|  | { |
|  | //Prints board for User to place ships |
|  | //Declare variables |
|  | char start = ' '; |
|  |  |
|  | cout << endl << "Game Board" << endl; |
|  | for(int i = 0; i < a.getCols(); i++) |
|  | { |
|  | cout << " " << i+1; |
|  | } |
|  | cout << endl; |
|  | for(int i = 0; i < a.getRows(); i++) |
|  | { |
|  | cout << " "; |
|  | for(int j = 0; j < a.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  | cout << i+1 << " "; |
|  | for(int j = 0; j < a.getCols(); j++) |
|  | { |
|  | cout << "| " << start << " "; |
|  | } |
|  | cout << "|" << endl; |
|  | } |
|  | cout << " "; |
|  | for(int j = 0; j < a.getCols(); j++) |
|  | { |
|  | cout << "|-----"; |
|  | } |
|  | cout << "|" << endl; |
|  |  |
|  | } |
|  | //Function that checks for a victor |
|  | bool victory(int usr, int enemy, int ships) |
|  | { |
|  | bool vic = false; |
|  |  |
|  | if(usr == ships) |
|  | { |
|  | cout << endl << "Great job, Captain!! You are victorious!" << endl; |
|  | vic = true; |
|  | } |
|  | if(enemy == ships) |
|  | { |
|  | cout << endl << "Captain...How could you allow the enemy to win?" << endl; |
|  | vic = true; |
|  | } |
|  |  |
|  | return vic; |
|  | } |
|  |  |
|  | void userBoard(int, int, char\*\*){ |
|  |  |
|  | } |
|  | //Gets input and prints enemy board |
|  | void enmyBoard(int, int, char\*\*){ |
|  |  |
|  | } |