|  |
| --- |
| Battleship: The Game |
| Project 2 |
| Ryan Melendez CSC 17A 12/12/15 |

**Introduction:**

The project I created is a two player battleship game simulator in which two players must guess the locations of the other persons ships placed on a 10 x 15 game board. At the start of the game, the program prompts the user, based off player numbers, to input the location of the 5 ships of varying sizes. As each turn is taken, the opposing players board is stored in memory and the proper board pertaining to that player is displayed. I chose to create this program because it was a game that I remember playing growing up and felt that my personal experience from playing this game would allow me to recreate it with accuracy. In this version structs have been replaced with classes to create a more secure and accurate game.

Ship Size: Cruiser: 2 units, Frigate: 3 units, Submarine: 3 units, Escort: 4 units, Battleship: 5 units

Game board is 10 x 15:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

0 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

8 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

**Summary:**

Program size: 1011 lines

Number of variables: around 32

Number of Classes: 8

Number of functions: 7

IDE: Visual Studios 13

Overall this program took me around 1 week to create and for the most part gave me the opportunity to incorporate the materials covered in Chapters 9-12 with the addition of Chapters 13-16. When it came to the few exceptions, I was able to find a way to demonstrate how they were used in somewhat of a relevant manner.

I enjoyed creating this project because it allowed me to recreate a game I played early on and my life and still to this day. The fact that I now have a playable game on my computer is something I look forward to showing to friends and family as the opportunities arise. Some of the issues I ran into while creating the project were mainly dealing with dynamic memory allocations and pointers solely based on the fact that I had not fully grasped the concepts. Transferring everything over to classes too a little bit more time than I thought to create but it allowed me to create a more organized and efficient game.

**Description:**

Refer to attached FlowChart.

**Psuedocode:**

*Start*

*-preset ship values*

*-using classes*

*-create board*

*-using an array[][]*

*Input*

*-ask players to input location of ships*

*- asks for placement dealing with 2 separate boards*

*-after users input their ships, the game begins*

*- prompts players to input coordinates of attack on the opposing players board*

*- based off x and y coordinates*

*-all of this loops until all of one players ships are sunk*

*Output*

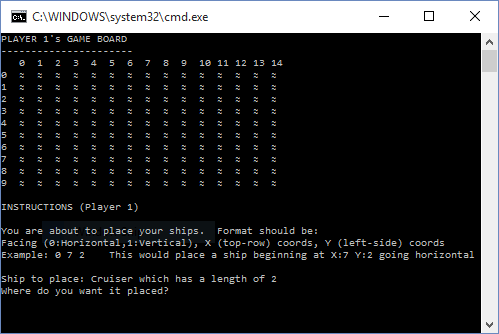
*-outputs the players names and the winner of the two*

*-outputs the total hits and misses*

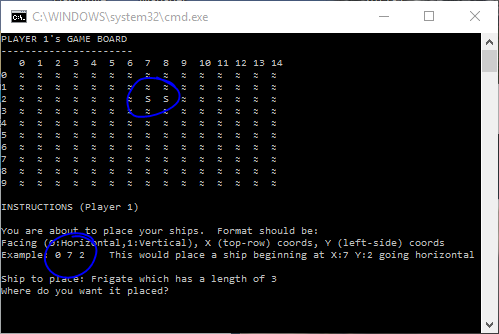
*-outputs them to both txt and dat files*

**Gameplay examples:**

Simple picture of how the game play is initialized

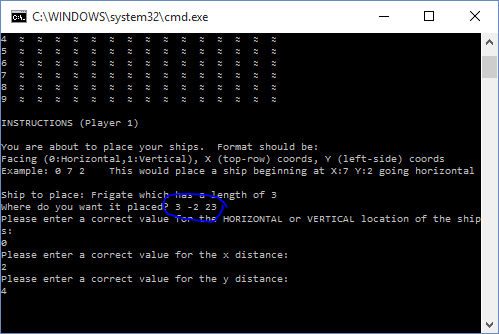
****

Example case: inputting the coordinates of 0 (Horizontal) 7 (X-direction) 2 (Y-direction)



Error Checking: if the user inputs incorrect values, user it asked to input them again

In this case, the incorrect values of (3, -2, 23)



**New Concepts:**

-Chapter 9 Pointers

* Line 36 OutInfo ptr

-Chapter 10 CStrings

* String Objects
  + Line 335 – compared string objects to alphabetize them

-Chapter 11 Structured Data

* Enumerated Data:
  + Line 31- enum data

-Chapter 12 Advanced File Operations

* Reading from a .txt file
  + Line 60 - calls the function dispIntro
  + Line 378 - within the function accessing the WelcomeMessage file
* Outputting to a .txt file
  + Line 183 – sends info to the function otpt
  + Line 349 – majority of the function outputs the game stats
* Outputting to a .dat file
  + Line 323 – creates dat file
  + Line 324 - sends the names to that dat file

- Chapter 13 Classes

* Placeship
* Ship
* Point
* Possib
* OutInfo
* Direction
* Player
* Name

- Chapter 14

* Copy Constructors
  + Line 11; Direction.h
  + Line 14; Placeship.h
  + Line 7; Player.h
  + Line 8; Point.h
  + Line 16; Ship.h
* Overload operators
  + Line 12-13; Direction.h
  + Line 12-13; OutInfo.h
  + Line 23-25; Placeship.h
  + Line 25-26; Ship.h

- Chapter 15 Inheritance

* Polymorphism
  + Line 8; Placeship.h
  + Line 25; Possib.h
* Virtual Functions
  + Line 10; Direction.h
  + Line 6; OutInfo.h
  + Line 13 ; Placeship.h
  + Line 15; Possib.h
  + Line 13; Ship.h

- Chapter 16 Exceptions

* Exceptions
  + Line 368; main.cpp
* Vectors
  + Line 50; main.cpp

**References**

* Gaddis 8th edition: Getting started with C++
* <http://www.hasbro.com/common/instruct/Battleship.PDF> : Game instructions

**Code**

/\*

Ryan Melendez

Battleship Game

Project 2

CSC-17A

Visual Studios 13

10/23/15

\*/

#include <iostream>

#include <string>

#include <fstream>

#include <cctype>

#include <vector>

#include "Ship.h"

#include "Placeship.h"

#include "Player.h"

#include "Possib.h"

#include "OutInfo.h"

using namespace std;

const char isWATER = 247; // ASCII Character Code

const char isHIT = 'X'; // Constant chars used to display the board

const char isSHIP = 'S'; // chose constant since it would be referenced more than once

const char isMISS = '0'; // and it would allow the user to customize the game board to their preference

enum DIRECTION { HORIZONTAL, VERTICAL }; // keep track of which direction to place the ship

;

bool gameRunning = false;

//Functions

void LoadShips(vector<Ship>& sp);

void ResetBoard(Player[]);

void DrawBoard(int, Player[]);

bool UserInputAttack(int&, int&, int);

bool GameOverCheck(int, Player []);

void otptStat(int, int, int);

void dispIntro();

int main()

{

vector<Ship> spVec(5);

Player ply[3];

int missCount = 0;

Placeship pS;

int hitCount = 0;

Possib pZ;

OutInfo \*ptr, otFile; // pointer to a struct

ptr = &otFile;

dispIntro(); // reads input from a txt file

LoadShips(spVec);

ResetBoard(ply);

//"PLACE SHIPS" phase of game

//Loop through each player...

for (int aplyr = 1; aplyr<3; ++aplyr)

{

//Loop through each ship type to place

for (int thisShip = 0; thisShip<5; thisShip++)

{

//Display gameboard for player

DrawBoard(aplyr, ply);

//Give instructions

cout << "\n";

cout << "INSTRUCTIONS (Player " << aplyr << ")\n\n";

cout << "You are about to place your ships. Format should be:\n";

cout << "Facing (0:Horizontal,1:Vertical), X (top-row) coords, Y (left-side) coords\n";

cout << "Example: 0 7 2 This would place a ship beginning at X:7 Y:2 going horizontal\n\n";

cout << "Ship to place: " << spVec[thisShip].getN() << " which has a length of " << spVec[thisShip].getL() << "\n";

cout << "Where do you want it placed? ";

//Get input from user and loop until good data is returned

pS.onGrid[0].setX(-1);

while (pS.onGrid[0].getX() == -1)

{

pS.UserInputShipPlacement(); // do this while

cout << pS.getDirect() << " " << pS.onGrid[0].getX() << " " << pS.onGrid[0].getY();

//cout << pS.getDirect() << " " << pS.getX() << " " << pS.getY();

}

//Combine user data with "this ship" data

pS.setL(spVec[thisShip].getL());

pS.setN(spVec[thisShip].getN());

cout << pS.getL();

//Add the FIRST grid point to the current player's game board

ply[aplyr].grid[pS.onGrid[0].getX()][pS.onGrid[0].getY()] = isSHIP;

//Determine ALL grid points based on length and direction

for (int i = 1; i<pS.getL(); i++)

{

if (pS.getDirect() == HORIZONTAL){

pS.onGrid[i].setX(pS.onGrid[i - 1].getX()+1);

pS.onGrid[i].setY(pS.onGrid[i - 1].getY());

}

if (pS.getDirect() == VERTICAL){

pS.onGrid[i].setY(pS.onGrid[i - 1].getY()+1);

pS.onGrid[i].setX(pS.onGrid[i - 1].getX());

}

//Add the remaining grid points to our current players game board

ply[aplyr].grid[pS.onGrid[i].getX()][pS.onGrid[i].getY()] = isSHIP;

}

//Loop back through each ship type

}

//Loop back through each player

}

//Ready to play the game

gameRunning = true;

int thisPlayer = 1;

do {

//Because we are attacking , the

//opposite players board is the displayed

int enemyPlayer;

if (thisPlayer == 1)

enemyPlayer = 2;

if (thisPlayer == 2)

enemyPlayer = 1;

system("cls");

DrawBoard(enemyPlayer, ply);

//Get attack coords from this player

bool crctIn = false;

int x, y;

while (crctIn == false)

{

crctIn = UserInputAttack(x, y, thisPlayer);

}

//Check board; if a ship is there, set as HIT.. otherwise MISS

if (ply[enemyPlayer].grid[x][y] == isSHIP)

{

ply[enemyPlayer].grid[x][y] = isHIT;

missCount++;

ptr->setMiss(missCount); // keeps track of how many were hit

}

if (ply[enemyPlayer].grid[x][y] == isWATER)

{

ply[enemyPlayer].grid[x][y] = isMISS;

hitCount++;

ptr->setMiss(hitCount); // keeps track of how many were missed

}

//Check to see if the game is over

//If 0 is returned, nobody has won yet

int aWin = GameOverCheck(enemyPlayer, ply);

if (aWin != 0) {

gameRunning = false;

break;

}

if (missCount = 16)

{

pZ.dispOutro();

}

//Alternate between each player as we loop back around

thisPlayer = (thisPlayer == 1) ? 2 : 1;

} while (gameRunning);

system("cls");

cout << "\n\nCONGRATULATIONS!!! PLAYER " << thisPlayer << " HAS WON THE GAME!\n\n\n\n";

// output files to GameStats.txt file

// outputs usernames to a binary file

otptStat(ptr->getMiss(), ptr->gethit(), thisPlayer); // sending miss count and hit count as pointers.

system("pause");

return 0;

}

bool GameOverCheck(int enemyPLAYER, Player p[])

{

bool winner = true;

//Loop through enemy board

for (int w = 0; w<15; w++)

{

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

{

//If any ships remain, game is NOT over

if (p[enemyPLAYER].grid[w][h] = isSHIP)

{

winner = false;

return winner;

}

}

}

//If we get here, somebody won, game over!

return winner;

}

bool UserInputAttack(int& x, int& y, int theplayer)

{

cout << "\nPLAYER " << theplayer << ", ENTER COORDINATES TO ATTACK: ";

bool goodInput = false;

cin >> x >> y;

if (x<0 || x >= 15)

return goodInput;

if (y<0 || y >= 10)

return goodInput;

goodInput = true;

return goodInput;

}

void LoadShips(vector<Ship>& sp)

{

// Sets the default data for the ships

// we plan to include in the game

sp[0].setN("Cruiser"); sp[0].setL(2);

sp[1].setN("Frigate"); sp[1].setL(3);

sp[2].setN("Submarine"); sp[2].setL(3);

sp[3].setN("Escort"); sp[3].setL(4);

sp[4].setN("Battleship"); sp[4].setL(5);

}

void ResetBoard(Player p[])

{

//Loop through each player

for (int plyr = 1; plyr<3; plyr++)

{

//For each grid point, set contents to 'water'

for (int w = 0; w<15; w++)

{

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

{

p[plyr].grid[w][h] = isWATER;

}

} //Loop back to next player

}

}

void DrawBoard(int thisPlayer, Player p[3])

{

//Draws the board for a player (thisPlayer)

cout << "PLAYER " << thisPlayer << "'s GAME BOARD\n";

cout << "----------------------\n";

//Loop through top row (board\_width) and number columns

cout << " ";

for (int w = 0; w<15; w++) {

if (w < 10)

//Numbers only 1 character long, add two spaces after

cout << w << " ";

else if (w >= 10)

//Numbers 2 characters long, add only 1 space after

cout << w << " ";

}

cout << "\n";

//Loop through each grid point and display to console

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

{

for (int w = 0; w<15; w++)

{

//If this is the FIRST (left) grid point, number the grid first

if (w == 0)

cout << h << " ";

//If h was 1 character long, add an extra space to keep numbers lined up

if (w<10 && w == 0)

cout << " ";

//Display contents of this grid (if game isn't running yet, we are placing ships

//so display the ships

if (gameRunning == false)

cout << p[thisPlayer].grid[w][h] << " ";

//Don't show ships, BUT show damage if it's hit

if (gameRunning == true && p[thisPlayer].grid[w][h] != isSHIP)

{

cout << p[thisPlayer].grid[w][h] << " ";

}

else if (gameRunning == true && p[thisPlayer].grid[w][h] == isSHIP)

{

cout << isWATER << " ";

}

//If we have reached the border.. line feed

if (w == 15 - 1)

cout << "\n";

}

}

}

void otptStat(int miss, int hits, int winner) // writes output files to a text file & dat binary file

{

// keep track of how many times the shots are fired, missed, names of the two players.

ofstream outFile;

fstream file;

string first, last;

string name1, name2;

cout << "Player 1 please enter your name: ";

getline(cin, name1);

cout << "Player 2 please enter your name: ";

getline(cin, name2);

outFile.open("GameStats.txt", ios::out | ios::app); // file for txt

file.open("GameStats.dat", ios::out | ios::binary); // creates a binary file that mimics the .txt file

file.write(reinterpret\_cast<char \*>(&name1), sizeof(name1));

file.write(reinterpret\_cast<char \*>(&name2), sizeof(name2));

// can be used to keep track of the people who use the game

// for say maybe in an arcade setting

outFile << "============================================" << endl;

// alphabetizing the string objects using operators

if (name1 > name2)

{

last = name1;

first = name2;

outFile << " " << first << " and " << last << " " << endl;

}

else if (name2 > name1)

{

first = name1;

last = name2;

outFile << " " << first << " and " << last << " " << endl;

}

outFile << " Thanks for playing Battleship! " << endl;

outFile << "============================================" << endl;

outFile << "Player 1: " << name1 << endl;

outFile << "Player 2: " << name2 << endl;

outFile << "The total amount of shots missed was: " << miss << endl;

outFile << "The total amount of shots hit was: " << hits << endl;

outFile << "The winner was Player " << winner + 1 << " !!" << endl;

outFile.close();

system("cls");

}

void dispIntro() // reads input from a txt file

{

string getCntnt;

ifstream openFile;

openFile.exceptions(std::ios::failbit);

// now any operation that sets the failbit error flag on file throws

try {

openFile.open("C:\\WelcomeMessage.txt", ios::in);

}

catch (std::ios\_base::failure &fail) {

// opening the file failed! do your stuffs here

}

// disable exceptions again as we use the boolean conversion interface

openFile.exceptions(std::ios::goodbit);

openFile.open("WelcomeMessage.txt");

while (openFile >> getCntnt)

{

cout << getCntnt << " ";

}

cout << endl << endl;

openFile.close();

}