**Project 1**

<Battleship>

CSC 5 - 41202

Name: David Macias

Date: 02/01/2016

**Introduction**

Title: Battleship

Usually, Battleship is played on square grids where ships of different sizes are secretly placed either vertical or horizontal without overlapping. After each player arranges the ships on their grid the game can start. The players take turns calling out the column (represented by a letter) and the row number that they wish to strike on the opponent’s grid. As the strike locations are being called, the player records it so that the location is not called again. If the attacking player calls all the coordinates that a single ship fills, that ship is destroyed and sunk. To win the game, a player must sink all the opponent’s ships.

For this game, ships are randomly arranged by the computer and the user inputs the coordinates. The game ends when the user has sunk all the ships or has entered the command to quit.

**Development Summary**

Project size: 402 lines

Number of Variables: 31

This project illustrates the basic ideas of the game and can be improved with new concepts to be learned in class.

The project took about a week working on it 1-3 hours per day.

The most difficult part of the project was creating an algorithm so that ships do not overlap when they are randomly arranged by the program on the grid. It is still not complete and overlapping still occurs in some instances.

The only new thing I had to learn which was not taught in class was the reading in of data from a file using the fstream library for the rank chart.

One neat concept I learned in this project is that data from one point when the program was ran can be stored, then reused when the program is ran at a different time when using external files.

**Description**

This project illustrates the concepts we’ve learned so far in class.

The 7 constructs:

-Independent If statements

-Dependent if statements

-Ternary operators

-Switch-Case

-While loops

-Do-While loops

-For loops

As well as different system libraries

-cstdlib

-ctime

-iomanip

-fstream

-string

**Flow Chart**

**Pseudo Code**

Initialize

While modeSet is false

Display the Intro Menu

If 1 is entered

Display the difficulties

If 1 is entered

Set the grid size X and Y to 10

Else if 2 is entered

Set the grid size X and Y to 15

Else

Set the grid size X and Y to 20

Else if 3 is entered

Open in file “Battleship Scores.dat”

Display its contents

Close in file

Else

Set ModeSet to true

Set char array shpGrid and usrGrid according to rows and columns and difficulty

Randomly generate values for the X and Y that is within the rows and columns for each ship

For each index in the ship array

If the ship’s X and Y matches the index

Randomly generate a 1 or 2

If 2 is generated

draw the ship for each row index for each integer of the ship length

increment the row by 1

else

draw the ship for each column for each integer of the ship length

increment the column by 1

For each index In the ship array

If the index contains a ship character

Add 1 to ship count

For each index in the ship array

If the index does not contain a ship character

Input a non ship character

For each index in the user array

Input a non ship character

Do

Display the user grid

If the row and column inputted by the user matches a ship character in the ship array

Decrease the ship count by 1

Increase the strike count by 1

Increase the hit count by 1

Else

Input and ‘X’ in that row and column on the user array

Increase the strike count by 1

Prompt the user to input the row and column to strike

If user enters “00”

Set quit to true

While quit is false

If quit is true

Display ship array

Display total strikes and hits and calculate ratio

Prompt user for initials

Print initials, difficulty, strikes, hits, ratio to outfile “Battleships Scores”

Read in contents from infile “Battleships Scores”

Exit

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| character | usrGrid[] | Array that the user sees | main() |
|  | shpGrid[] | Array that contains the ships | Main() |
|  | lttrIn | User character input for the array location in the column | Main() |
| short | rows | Number of rows in both grids | Main() |
|  | clmns | Number of columns in both grids | Main() |
|  | shipX(a) | Column location of ship(a) a=[1-5] | Main() |
|  | shipY(a) | Row location of ship(a) a=[1-5] | Main() |
|  | shipL(a) | The number of characters the ship(a) is. a=[1-5] | Main() |
|  | shipCnt | Counter for the number of ship characters in the ship array | Main() |
|  | hitCnt | Counter for the ships that are hit | Main() |
|  | rowIn | User input for the row location | Main() |
|  | dfflty | Number input to change the array sizes | Main() |
|  | menuNum | Number option for start game, change difficulty, or see ranks | Main() |
| Boolean | hit | Becomes true when user column, row input matches ship column, row | Main() |
|  | quit | Becomes true when user inputs the command to quit the game | Main() |
|  | jstStrt | Becomes false after the user grid is printed | Main() |
|  | modeSet | Becomes true after the option to play the game is inputted by user | Main() |
| string | intl | User’s initials | Main() |
|  | fLine | Lines read from the external in file | Main() |

**Reference**

1. Textbook
2. <http://www.cplusplus.com/forum/lounge/52458/> -Reading in From file

**Program**

/\*

\* Author: David Macias

\* Created on January 25, 2016, 11:28 PM

\* Purpose: CSC 5 Project 1: Battleship

\*/

//System Libraries

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

//User Libraries

//Global Constants

//Function Prototypes

//Execution Begins Here

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

//Set the random time seed

srand(static\_cast<int>(time(0)));

//Declare and initialize variables

unsigned short rows; //Number of Rows in both Grids

unsigned short clmns; //Number of Columns in both Grids

unsigned short shipX1, shipX2, shipX3, shipX4, shipX5; //Second index for the 2 dimensional array

unsigned short shipY1, shipY2, shipY3, shipY4, shipY5; //First index for the 2 dimensional array

unsigned short shipL1 = 6, shipL2 = 4; //The length for ships 1 and 2

unsigned short shipL3 = 4, shipL4 = 3; //The length for ships 3 and 4

unsigned short shipL5 = 2; //The length for ship 5

unsigned short shpCnt = 0; //Ship counter

unsigned short ortn; //To determine orientation of ships in the ship grid

unsigned short strkCnt = 0; //Strike Counter

unsigned short hitCnt = 0; //Hit Counter

unsigned short rowIn; //User input for the row

unsigned short dfflty = 1; //Difficulty of the game / Automatically set to 1 at the start of game

unsigned short menuNum; //Initial menu option input

char lttrIn; //User input for the column

bool hit = false; //Boolean value becomes true when a ship is hit

bool quit = false; //Boolean value becomes true when the user quits the game

bool jstStrt = true; //Boolean value becomes false after first input is made

bool modeSet = false; //Boolean value becomes true after the initial menu

string intl; //User's initials

string fLine;

ofstream out; //Output stream Game scores

ifstream in;

//Open the Scores file

out.open("Battleship Scores.dat",ios::app);

//Start the Game!

while(!modeSet){

cout <<setw(30) <<"BATTLESHIP!" <<endl;

cout <<endl <<" 1.Difficulty: "

<<(dfflty == 1?"Easy":

dfflty == 2?"Medium":

dfflty == 3?"Hard":"Difficulty not set");

cout <<" 2.Start Game"

" 3.Ranks" <<endl;

cout <<"Enter 1 to change Difficulty or 2 to start the Game." <<endl;

cin >>menuNum;

//Change Difficulty

if(menuNum == 1){

cout <<"Enter Difficulty: 1-Easy" <<endl;

cout <<" 2-Medium" <<endl;

cout <<" 3-Hard" <<endl;

cin >>dfflty;

}else if(menuNum == 3){

in.open("Battleship Scores.dat");

while(getline (in,fLine)){

cout << fLine << '\n';

}

in.close();

}else{

modeSet = true;

}

//Set array index according to difficulty

switch(dfflty){

case 1:{

rows = 10;

clmns = 10;

break;

}case 2:{

rows = 15;

clmns = 15;

break;

}case 3:{

rows = 20;

clmns = 20;

break;

}default: {

rows = 10;

clmns = 10;

break;

}

}

}

//Set the index sizes of the arrays

char shpGrid [rows][clmns] = {}; //Size of the 2 dimensional ship array

char usrGrid [rows][clmns] = {}; //Size of the 2 dimensional User array

//Randomize ship locations

shipX1 = rand()%(clmns-shipL1+1)+1;

shipY1 = rand()%(rows-shipL1+1)+1;

shipX2 = rand()%(clmns-shipL2+1)+1;

shipY2 = rand()%(rows-shipL2+1)+1;

shipX3 = rand()%(clmns-shipL3+1)+1;

shipY3 = rand()%(rows-shipL3+1)+1;

shipX4 = rand()%(clmns-shipL4+1)+1;

shipY4 = rand()%(rows-shipL4+1)+1;

shipX5 = rand()%(clmns-shipL5+1)+1;

shipY5 = rand()%(rows-shipL5+1)+1;

//Draw Ship 1 in Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(j == shipX1 && i == shipY1){

ortn = rand()%2+1;

if(ortn == 2){

while(i < shipY1+shipL1){

shpGrid[i][j] = 'O';

i++;

}

}else{

while(j < shipX1+shipL1){

shpGrid[i][j] = 'O';

j++;

}

}

}

}

}

//Draw Ship 2 Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(j == shipX2 && i == shipY2){

ortn = rand()%2+1;

if(ortn == 2){

if(shpGrid[i][j] == 'O'){

do{

j++;

}while(shpGrid[i][j] == 'O');

}

while(i < shipY2+shipL2){

shpGrid[i][j] = 'O';

i++;

}

}else{

if(shpGrid[i][j] == 'O'){

do{

i++;

}while(shpGrid[i][j] == 'O');

}

while(j < shipX2+shipL2){

shpGrid[i][j] = 'O';

j++;

}

}

}

}

}

//Draw Ship 3 Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(j == shipX3 && i == shipY3){

ortn = rand()%2+1;

if(ortn == 2){

if(shpGrid[i][j] == 'O'){

do{

j++;

}while(shpGrid[i][j] == 'O');

}

while(i < shipY3+shipL3){

shpGrid[i][j] = 'O';

i++;

}

}else{

if(shpGrid[i][j] == 'O'){

do{

i++;

}while(shpGrid[i][j] == 'O');

}

while(j < shipX3+shipL3){

shpGrid[i][j] = 'O';

j++;

}

}

}

}

}

//Draw Ship 4 Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(j == shipX4 && i == shipY4){

ortn = rand()%2+1;

if(ortn == 2){

if(shpGrid[i][j] == 'O'){

do{

j++;

}while(shpGrid[i][j] == 'O');

}

while(i < shipY4+shipL4){

shpGrid[i][j] = 'O';

i++;

}

}else{

if(shpGrid[i][j] == 'O'){

do{

i++;

}while(shpGrid[i][j] == 'O');

}

while(j < shipX4+shipL4){

shpGrid[i][j] = 'O';

j++;

}

}

}

}

}

//Draw Ship 5 Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(j == shipX5 && i == shipY5){

ortn = rand()%2+1;

if(ortn == 2){

if(shpGrid[i][j] == 'O'){

do{

j++;

}while(shpGrid[i][j] == 'O');

}

while(i < shipY5+shipL5){

shpGrid[i][j] = 'O';

i++;

}

}else{

if(shpGrid[i][j] == 'O'){

do{

i++;

}while(shpGrid[i][j] == 'O');

}

while(j < shipX5+shipL5){

shpGrid[i][j] = 'O';

j++;

}

}

}

}

}

//Read how many O's there are to determine game end

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(shpGrid[i][j] == 'O'){

shpCnt++;

}

}

}

//Fill the rest of the Ship Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

if(!(shpGrid[i][j] == 'O')){

shpGrid[i][j] = '~';

}

}

}

//Fill out the User Grid

for(int i = 1; i <= rows; i++){

for(int j = 1; j <= clmns; j++){

usrGrid[i][j] = '~'; //Character to fill the board with

}

}

//Start the time

int begT = static\_cast<int>(time(0));

//Print out the User Grid and Play the Game!

do{

cout <<" ";

for(int k=1, f=65; k <= clmns; k++, f++){

cout <<" " <<static\_cast<char>(f); //Print Letters for Columns

}

cout <<endl;

for(int i = 1, t = 1; i <= rows; i++, t++) {

cout <<setw(2) <<t; //Print the number indicating the row

for(int j = 1; j <= clmns; j++) {

if(i == rowIn && j == static\_cast<int>(lttrIn)-64){

if(shpGrid[rowIn][static\_cast<int>(lttrIn)-64] == 'O'){

usrGrid[i][j] = 'H';

shpCnt--;

strkCnt++;

hitCnt++;

hit = true;

}else{

usrGrid[i][j] = 'X';

strkCnt++;

}

}

cout <<" " <<usrGrid[i][j];

}

cout <<(i == 2?" HITS NEEDED TO SINK ENEMY SHIPS":

i == 3?" Ship 1 -6 Hits Ship 2 -4 Hits Ship 3 -4 Hits":

i == 4?" Ship 4 -3 Hits Ship 5 -2 Hits":

i == 6?" LEGEND":

i == 7?" H - Hit":

i == 8?" X - Miss":"");

cout <<endl;

}

//Check if all the ship indices were hit

if(shpCnt == 0){

cout <<"You won! You sunk all the enemies ships!" <<endl;

}

//Check if a hit was made

if (hit){

cout <<endl;

cout <<"You got a hit! That leaves " <<shpCnt <<" hits still needed"

" to win!" <<endl;

hit = false;

}else{

if(jstStrt){

cout <<endl <<"There are " <<shpCnt

<<" hits needed to win!" <<endl;

jstStrt = false;

}else{

cout <<endl <<"Miss! There are still " <<shpCnt

<<" hits needed to win!" <<endl;

}

}

//Input the index of the strike that is to be made

cout <<"Enter the Letter of the column and row number you wish to strike"

" (i.e. A1)." <<endl;

cout <<"Or Enter two zeros to quit." <<endl;

cin >>lttrIn >>rowIn;

if (static\_cast<int>(lttrIn) == 48 && rowIn == 0){

quit = true;

}

}while(!quit);

//End the time

int endT = static\_cast<int>(time(0));

//If user quits show them the ship Grid

if(quit){

cout <<"Good try! GAME OVER" <<endl;

cout <<" ";

for(int k=1, f=65; k <= clmns; k++, f++){

cout <<" " <<static\_cast<char>(f); //Print Letters for Columns

}

cout <<endl;

for(int i = 1, t = 1; i <= rows; i++, t++) {

cout <<setw(2) <<t; //Print the number indicating the row

for(int j = 1; j <= clmns; j++) {

cout <<" " <<shpGrid[i][j];

}

cout <<endl;

}

}

//Output the hit to strike ratio

cout <<strkCnt <<" Strikes were ordered and " <<hitCnt

<<" hits were made." <<endl;

cout <<"That means for 1 hit " <<strkCnt\*1.0f/hitCnt

<<" strikes were made." <<endl;;

//Get the user's initials for the Scores

cout <<endl <<"Enter your initials" <<endl;

cin >>intl;

//Output scores to file

out<<setw(8) <<intl <<setw(15)

<<(dfflty == 1?"Easy":

dfflty == 2?"Medium":"Hard")<<setw(20) <<strkCnt <<setw(10)

<<hitCnt <<setw(10) <<strkCnt\*1.0f/hitCnt <<setw(10)

<<endT-begT <<endl;

//Display the Ranks on the Screen

in.open("Battleship Scores.dat");

while(getline (in,fLine)){

cout << fLine << '\n';

}

in.close();

//Exit stage right

out.close();

return 0;

}