Battleship Board Game

Christopher Delgado

CSC 17A

Summer 2021

45398

Summary

Lines of code: 428

Number of Variables: 16

This assignment gave me more trouble than I initially expected. I decided to take the battleship game I coded in CSC 5 out of the closet and give it new life for this project. I came in assuming that it would be easy to build on top of the foundation I built and that I would be in and out quickly. I quickly realized what the confusion that comes with switching out sets of variables that were passed in and out of multiple functions with a structure feels like. Eventually, after lots of trial and error, I was able to add the new concepts that I learned into my game of Battleship.

I found that the structures added a lot of utility to my program. I was able to trim down and eliminate 3 functions all together with the use of structures. Dynamic memory allocation also allowed me to add new player customization to my game. With the use of pointer structure arrays and the 'new' operator , I was able to allow the player to change the size of the board at the beginning of the program. Before this I was only able to change this in the source code itself, which wasn't really ideal. This was able to provide me with yet another reason why structures are so powerful.

In the last version of my program I added the ability to compare your score to previous high scores and store them in a file. I also utilized a binary file to store the information of a structure which held basic info of the most recent player.

Overall, it was cool to go back and improve on old code that I wrote a year back. It allowed me to see the improvements I have made in my programming skills while still presenting a challenging yet rewarding experience.

Pseudo Code

Initialize

Enter the "start screen" function

Display start screen and instructions for player

Get player name

Enter "find size" function

User inputs size of the gameboard

Enter the "fill boat" function

Randomly assign values to 10 boats by using for loop

Use nested loops to stop computer from picking duplicate numbers

Enter "fill board" function

Give values to board starting at 1 and incrementing by 1 every time

Enter "User Boat" function

Asks user to input numbers to act as locations for their boats on the grid

Input validation

Enter "Display Board" function

For loop that goes from zero to desired length of board

If it is beginning of loop display current turn and max turn

Display game board using structure

Display hits and misses

For loop that displays users boats location

For loop that goes from zero to desired length of board

If it is beginning of loop display "User Board"

Display game board using structure

Display hits and misses

Enter do while loop

Enter "Hit Miss" function

Ask the user where the want to hit

Enter loop which checks if player choice is equal to boat location

If user choice is equal to computers boat it's a hit

If user choice is not equal to computers boat it's a miss

Enter "Computer Choice" function

*Set computer choice to a random number within the board *

For loop that prevents the computer from repeating previous numbers it guessed

If computer choice is equal to user boat then it is a hit

If computer choice is not equal to user boat it is a miss

Enter "Display Board" function from earlier

loop=Function "Win Lose"

Function returns false if max turns is reached

Function returns false if every user boat gets sunk

Function returns false if every computer boat gets sunk

If none of these are true increment turn and return true

while(loop==true)

If game was won

Enter "Stat File" function

Open a binary file

Input player stats into file then close file

Open binary file again

Output player stats from file back into structure then close file

Output player stats into program

Enter "High Score" function

Open file which holds high score and transfer that value to int named check

Close file

If you beat high score write new high score into the file

If you did not beat high score display high score

If you tied high score display message informing player

Cleanup dynamic memory

Return 0

Program

```
/*
 * File:   main.cpp
 * Author: Christopher Delgado
 * Created on July 17th, 2021
 * Purpose: Battleship
 */

//System Level Libraries
#include <iostream>   //I/O Library
#include <ctime>      //For random number generator
#include <iomanip>    //Formatting library
#include <string>     //string library to hold players name
#include <fstream>    //File library
using namespace std;  //Libraries compiled under std

//User Level Libraries
struct Game{
   int compBrd;
   int userBrd;
   int guess;
};

struct Stats{
   short nTurn;
   string name;
};

//Global Constants - Science/Math Related
//Conversions, Higher Dimensions
```

```
//Function Prototypes
void srtScrn(char, const int, Stats &);                    //Displays starting screen and
instructions
void findSze(Game *&, short&, int &);                      //Gets size of game board
void filBoat(int [], const int, short);                    //Fills computers boats with random
values
void fillBrd(Game[], short);                               //Sets up the board
void usrBoat(int [], const int, short);                    //Fills player choice for boats
void dsplyBd(int[], const int, Game [], short, Stats &, int, int, int[]);//Displays user and
computers board
void hitMiss(Game [], int [], short, const int, int);      //Function that figures out if player
got a hit or a miss
void cmpChce(Game [], int [], short, const int);           //Function that figures out if
computer got a hit or a miss
bool winLose(int [], int[], int &, int, bool, bool &);     //Determines whether or not the
game has been won or lost
void hiScore(ofstream &, ifstream &, int);                 //Opens high score file and checks
if you have high score
void statFle(Stats &, ofstream &, ifstream &, int);        //Puts Stats structure into a
binary file

//Execution Begins Here!
int main(int argc, char** argv) {
   //Random Number Seed Set Here
   srand(static_cast<unsigned int>(time(0)));

   //Variable Declarations
   const int NBOATS=10;          //Number of boats
   short size;                   //Holds size of game board
   Game *gameBrd=0;              //Structure array pointer for game board
   int boats[NBOATS];            //Array that holds value of boats
   int pBoats[NBOATS];           //Array that holds value of player boats
   int choice;                   //User choice of where to shoot on board
   int turn, maxTurn;            //Used for the amount of turns and max turns in game
   bool loop;                    //Used for do while loop
   char start;                   //Used to start game
   ofstream highScr;             //Used to open and read the high score file
   ifstream inptScr;             //Input your high score into the high score file
   ofstream outStat;             //Outputs stats structure from a file
   ifstream inptSta;             //Inputs stats structure in file
   Stats gmeStat;                //Game stats structure
   bool win;                     //States if game was won

   //Variable Initialization
   turn=1;                       //Player starts on turn 1
```

```
win=false;

//Mapping Process Inputs to Outputs
//Presents start screen and instructions
srtScrn(start, NBOATS, gmeStat);

//User inputs the size of the board they want to play on
findSze(gameBrd, size, maxTurn);

//Assigns random value to boats
filBoat(boats, NBOATS, size);

//Fills user and computer board
fillBrd(gameBrd, size);

//Fills user boats
usrBoat(pBoats, NBOATS, size);

//Displays both boards before any turns have been made
dsplyBd(pBoats, NBOATS, gameBrd, size, gmeStat, turn, maxTurn, boats);

do{
   //Game play function that determines a hit or a miss
   hitMiss(gameBrd, boats, size, NBOATS, choice);

   //Computers Choice
   cmpChce(gameBrd, pBoats, size, NBOATS);

   //Displays user and computers altered board
   dsplyBd(pBoats, NBOATS, gameBrd, size, gmeStat, turn, maxTurn, boats);

   //Function either returns false or true
   loop=winLose(boats, pBoats, turn, maxTurn, loop, win);

}while(loop);

//If you win then it will display game stats and high score
if(win==true){
   statFle(gmeStat, outStat, inptSta, turn);
   hiScore(highScr, inptScr, turn);
}


//Clean Up
delete []gameBrd;
```

```
    //Exit stage right!
    return 0;
}

//Displays start screen and instructions
void srtScrn(char start, const int NBOATS, Stats &name){
    //Asks user to start or exit
    cout<<"Welcome to Battleship\n";
    cout<<"Press Y to play or press N to exit\n";
    cin>>start;
    while(start!='y'&&start!='Y'&&start!='n'&&start!='N'){   //Input validation using while loop
        cout<<"Please enter either Y or N\n";
        cin>>start;
    }
    switch(start){   //Pressing Y leads to rest of code while N exits program
        case('Y'):
        case('y'):
            break;
        case('N'):
        case('n'):
            exit (0);
            break;
    }
    //Instructions for game
    cout<<"You will have to sink all of the computers ships in order to win!\n";
    cout<<"Each row is labeled from 1-10, 11-20, etc.\n";
    cout<<"The computer will randomly place "<<NBOATS<<" boats 1 space big and "
        <<"it is your objective to find them and sink them.\n";
    cout<<"A hit will be indicated by a \"x\" and a miss will be indicated by a \"o\".\n\n";

    //Gets name of the player
    cout<<"Please enter players first name\n";
    cin.ignore();
    getline(cin, name.name);
}

//Gets size of of game board
void findSze(Game *&gameBrd, short &size, int &maxTurn){
    cout<<"Enter the size of the game board \n";
    //Size has to be 15 because the number of boats is 10
    cout<<"The game board must be at least 15 spaces big\n";
    cin>>size;
    while(size<15){
        cout<<"The game board must be at least 15 spaces big\n";
```

```
        cin>>size;
    }
    maxTurn=size;
    gameBrd= new Game[size];
}

//Gives Values to array starting at 1 to board size limit
void fillBrd(Game g[], short size){
    for(int n=0; n<size; n++){
        g[n].compBrd=n+1;
        g[n].userBrd=n+1;
    }
}

//Display the game board
void dsplyBd(int boat[], const int NBOATS, Game g[], short size, Stats &name, int turn, int
maxTurn, int boats[]){
    //Displays computers boats for testing purposes
    for(int n=0; n<10; n++){
        cout<<boats[n]<<",";
    }
    cout<<endl;

    for (int n=0; n<size; n++){
        //Displays heading of game board
        if(n==0){
            cout<<"Turn#: "<<turn<<"                    ";
            cout<<"Max Turn#: "<<maxTurn<<endl;
            cout<<setw(27)<<"Computer Board"<<endl;
        }
        //Outputs Values of game board and accounts for hits and misses
        if(g[n].compBrd==0)
            cout<<"   "<<'x';
        else if(g[n].compBrd==-1)
            cout<<"   "<<'o';
        else if(g[n].compBrd<10)
            cout<<"   "<<g[n].compBrd;
        else
            cout<<"  "<<g[n].compBrd;
        if(n%10==(10-1))
            cout<<endl;
    }
    cout<<endl;

    cout<<endl;
```

```
    cout<<"Your boats#: ";
    for(int i=0; i<NBOATS; i++){
        if(i==NBOATS-1){
            if(boat[i]==0)
                cout<<"";
            else
                cout<<boat[i];
        }
        else{
            if(boat[i]==0)
                cout<<"";
            else
                cout<<boat[i]<<", ";
        }
    }
    //Displays the users board accounting for hits and misses
    cout<<endl;
    for (int n=0; n<size; n++){
        if(n==0){
            cout<<setw(19)<<name.name<<"'s Board"<<endl;
        }
        if(g[n].userBrd==0)
            cout<<"   "<<'x';
        else if(g[n].userBrd==-1)
            cout<<"   "<<'o';
        else if(g[n].userBrd<10)
            cout<<"   "<<g[n].userBrd;
        else
            cout<<"  "<<g[n].userBrd;
        if(n%10==(10-1))
            cout<<endl;
    }
}

//Adds values to boats 1-5
void filBoat(int a[], const int NBOATS, short size){
    for(int n=0; n<NBOATS; n++){
        a[n]=rand()%size+1;
    }
    //Keeps computer from repeating numbers when picking boats
    for(int i=0; i<=10000; i++){
        for(int n=0; n<NBOATS; n++){
            for(int j=0; j<NBOATS; j++){
                if(n==j)
                    cout<<"";
```

```
            else if(a[n]==a[j])
               a[n]=rand()%size+1;
         }
      }
   }
}

//Holds value for user boats
void usrBoat(int a[], const int NBOATS, short size){
   cout<<"Please enter "<<NBOATS<<" values for your boats from 1-"<<size<<endl;
   for(int n=0; n<NBOATS; n++){
      cout<<"Boat #"<<n+1<<") ";
      cin>>a[n];
   }
   //Input validation for all scenarios
   for(int i=0; i<=10000; i++){
      for(int n=0; n<NBOATS; n++){
         for(int j=0; j<NBOATS; j++){
            if(n==j)
               cout<<"";
            else if(a[n]==a[j]){
               cout<<"You entered "<<a[n]<<" twice, please enter a new number"<<endl;
               cin>>a[n];
            }
         }
         while(a[n]>size||a[n]<1){
         cout<<a[n]<<" is not between 1 and "<<size<<endl;
         cout<<"Please enter a new number"<<endl;
         cin>>a[n];
         }
      }
   }
}

//Function informs user whether or not they got a hit or a miss
void hitMiss(Game g[], int boats[], short size, const int NBOATS, int choice){
   cout<<endl<<"Which spot do you wanna shoot"<<endl;
   cin>>choice;
   while(choice<1||choice>size){     //Input validation for keeping choice input between 1-30
      cout<<"Please pick a number between 1-"<<size<<endl;
      cin>>choice;
   }
   bool hit=false;

   for(int n=0; n<NBOATS; n++){
```

```
        if(choice==boats[n]){
            boats[n]=0;
            hit=true;
        }
    }
    if(hit==true){
        cout<<"-----------------------------------------"<<endl
            <<"You got a hit"<<endl;
        for(int i=0; i<size; i++){
            if(choice==g[i].compBrd){
                g[i].compBrd=0;
            }
        }
    }
    if(hit==false){
        for(int j=0; j<size; j++){
            if(choice==g[j].compBrd){
                cout<<"-----------------------------------------"<<endl
                    <<"That was a miss"<<endl;
                g[j].compBrd=-1;
            }
        }
    }
}

//Used for the computers turn
void cmpChce(Game g[], int pBoats[], short size, const int NBOATS){
    bool hit=false;
    int cChoice=rand()%size+1;

    //Prevents computer from choosing same number
    for(int i=0; i<size; i++){
        for(int j=0; j<size; j++){
            if(cChoice==g[j].guess)
                cChoice=rand()%size+1;
        }
    }
    //If choice is equal to one of the boats then it is a hit
    for(int n=0; n<NBOATS; n++){
        if(cChoice==pBoats[n]){
           pBoats[n]=0;
            hit=true;
        }
    }
    if(hit==true){
```

```
        cout<<"Enemy got a hit at "<<cChoice<<endl;
        //Marks on board where the hit occurred
        for(int i=0; i<size; i++){
            if(cChoice==g[i].userBrd){
                g[i].guess=g[i].userBrd;
                g[i].userBrd=0;
            }
        }
    }
    if(hit==false){
        cout<<"Enemy missed their shot at "<<cChoice<<endl;
        //Marks where miss occurred
        for(int j=0; j<size; j++){
            if(cChoice==g[j].userBrd){
                g[j].guess=g[j].userBrd;
                g[j].userBrd=-1;
            }
        }
    }
}

//Function that informs if game has been won or lost
bool winLose(int boats[], int pBoats[], int &turn, int maxTurn, bool loop, bool &win){
    //Informs the player that they have sunken all the ships
    if(boats[0]==0&& boats[1]==0&& boats[2]==0&& boats[3]==0&& boats[4]==0,
       boats[5]==0&& boats[6]==0&& boats[7]==0&& boats[8]==0&& boats[9]==0){
        cout<<"\nYou sunk all the ships!\n";
        cout<<"You won!\n";
        win=true;
        return false;
    }
    //If all your boats get sunk you lose
    else if(pBoats[0]==0&& pBoats[1]==0&& pBoats[2]==0&& pBoats[3]==0&&
pBoats[4]==0,
            pBoats[5]==0&& pBoats[6]==0&& pBoats[7]==0&& pBoats[8]==0&& pBoats[9]==0){
        cout<<"\nAll your ships have been sunk.\n";
        cout<<"You Lost! :(\n";
        return false;
    }
    //Informs the player that they have run out of turns and lost
    else if(turn==maxTurn){
        cout<<"\nYou ran out of turns.\n";
        cout<<"You Lost! :(\n";
        return false;
    }
```

```
    else{
        turn++;
        return true;
    }
}

void hiScore(ofstream &oFile, ifstream &iFile, int turn){
    //Get the high score and put it into the variable check
    int check;
    iFile.open("Highscore.txt");
    iFile>>check;
    iFile.close();

    //Check if you got a high score
    if(check<turn){
        cout<<"You did not beat the high score of "<<check<<" turns"<<endl;
    }
    if(check>turn){
        cout<<"You hold the new high score of winning in "<<turn<<" turns"<<endl;
        oFile.open("Highscore.txt", ios::trunc);
        oFile<<turn;
        oFile.close();
    }
    if(check==turn){
        cout<<"You tied the high score of "<<turn<<" turns"<<endl;
    }
}

void statFle(Stats & game, ofstream &oFile, ifstream &iFile, int turn){
    game.nTurn=turn;

    //Opening the file and writing structure to file
    oFile.open("Stats.txt", ios::out | ios::binary);
    oFile.write(reinterpret_cast<char *>(&game), sizeof(game));
    oFile.close();

    //Writing the data back into the structure an displaying it
    iFile.open("Stats.txt", ios::in | ios::binary);
    if(iFile){
        iFile.read(reinterpret_cast<char *>(&game), sizeof(game));
        cout<<"\nPlayer Stats:\n_____________";
        cout<<"\nPlayer name: "<<game.name<<endl;
        cout<<"Turn taken to win: "<<game.nTurn<<endl<<endl;
    }
    else
```

```
        cout<<"\nThe file was not opened properly\n";
    iFile.close();
}

    if(rr==2)n=2;
    if(rr==3)n=3;
    sGuess[n]+=1;

    aGuess[++guess]=sGuess;//Save the result

    //Return the result
    return sGuess;
}
```