Project 2

**Battleship (extended)**

Joseph Alipranti

CSC-17A 49285

12/14/2020

**Introduction**

The game I have created with the implementation of the concepts learned in the previous chapters is the classic game, Battleship. I created this game because it used to be a childhood favorite game of mine, and even though the rules and gameplay are straightforward, I saw it as a challenge to program, and took it upon myself to create it. The rules for the game are simple. Take turns with the computer on guessing where the other player has their ships placed, and the first person to sink all of their enemy’s ships wins the game.

**Summary**

Lines of Code: almost 1200

Number of Variables: roughly 70 (some variables repeat in different functions, including objects from classes)

Utilizes concepts from Chapters 13 through 16, as well as concepts from Chapters 9 through 12

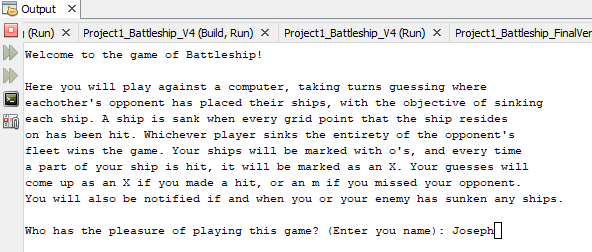
Programming this project was quite a challenge, as there were many things to validate and ensure certain actions/decisions made by the user or computer would not step out of bounds of the grid. For this second version of the original project, I was fortunately able to tackle two of the bigger problems that I noticed with the last version. The first, being that the computer’s guesses were entirely random, and that when the computer got a hit on the user, the computer would guess a completely different coordinate instead of going one spot over to hit the ship again. I was able to enhance this part of the computer’s guessing mechanism to give the user more competition. The other problem I was able to tackle, while implementing concepts from previous chapters, was my faulty “Ship has sank” declaration, that would sometimes go off when a ship has only been hit once or twice. Fortunately I was able to figure out a way to work this into a class with arrays, and it no longer gives the unwarranted message.

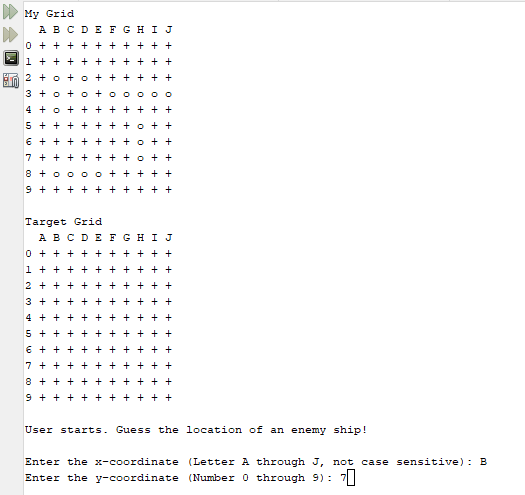
**Description**

For programming the solution to the problem (game), I brushed up on the basic rules of Battleship, the pieces/grids used by each player, and the methodology of the gameplay and how the grids get marked for reference. Programming this project took me a few days on top of the original program to complete, slowly everyday chipping away at the little functions and tasks that I needed to get done in order to have a near-replica of the actual game.

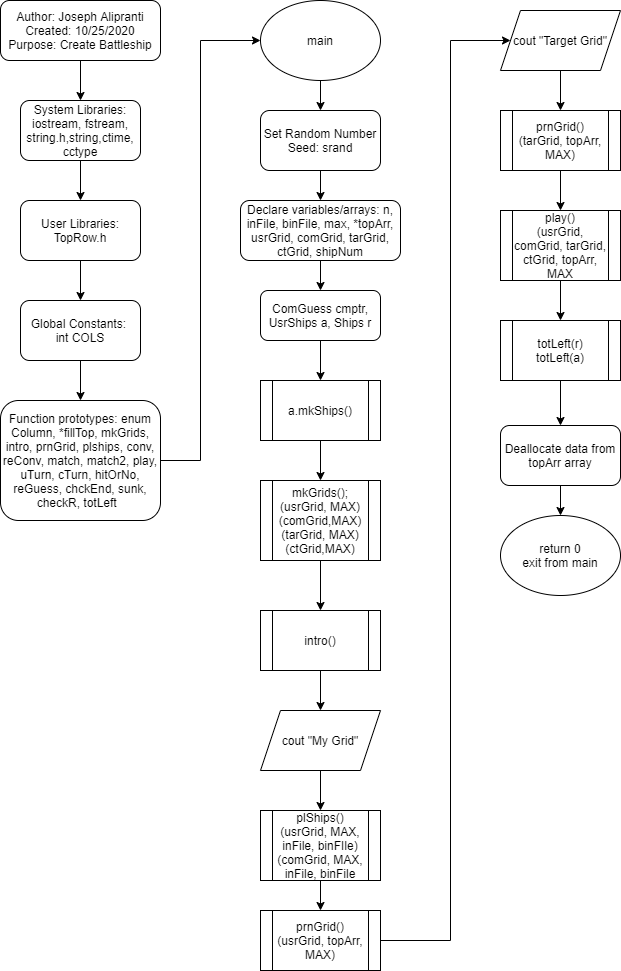
**Sample Input/Output**

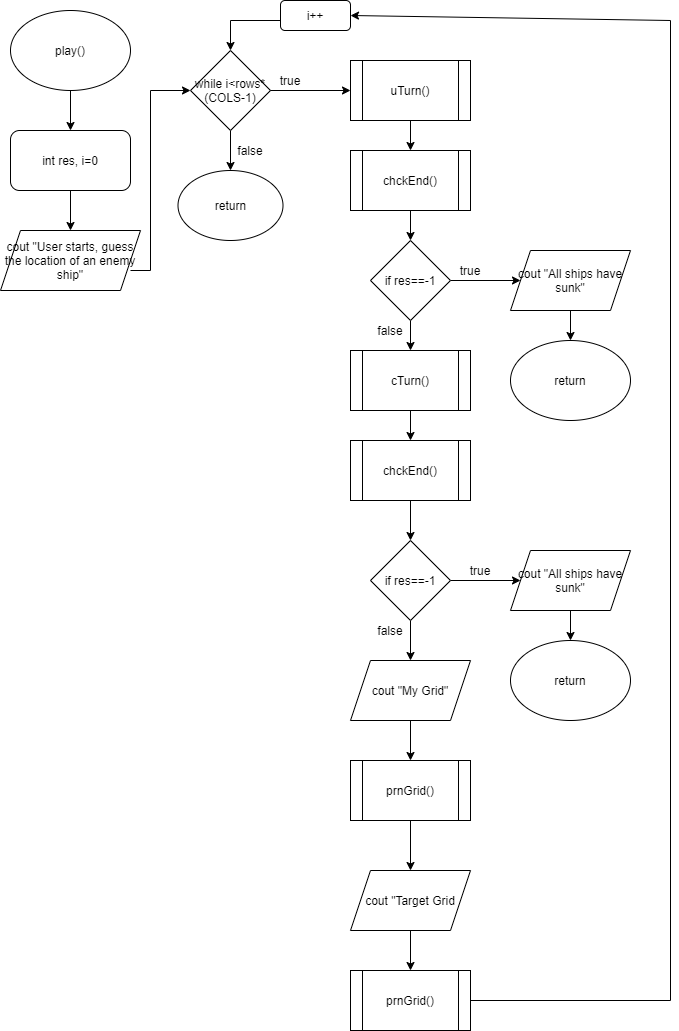
The only thing that you (the user) will be entering in the program for play, is your name at the beginning, and then the coordinates that are asked of you by the program for your guesses:

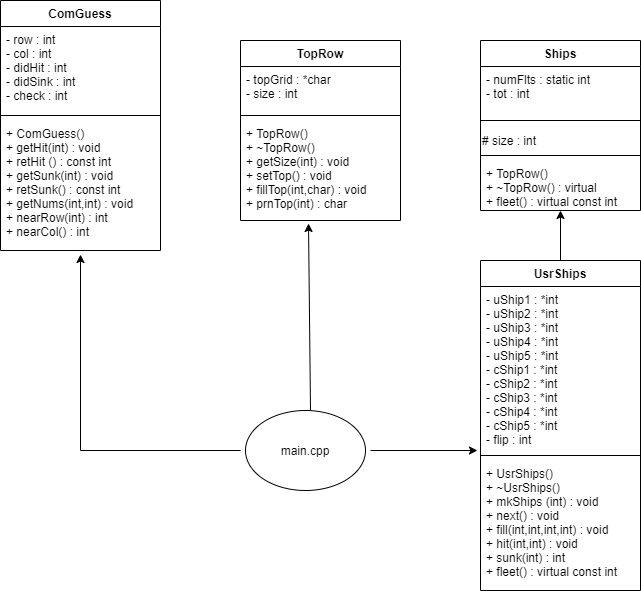




**Flowcharts/UML**







**Variables**

***Const int COLS***: global, above function prototypes to pass 2D arrays to functions

***Const int MAX***: main(), for the rows of each grid

***Ifstream inFile***: main(), to retrieve the ship pieces from a text file

***Char getC***: \*fillTop(), to get character and store in array after converting an integer

***Int shipNum***: holds the amount of ships per player

***Int lBound***: mkGrids(), to turn integers into characters using ASCII conversion for grid axis

***Int ships***: plShips(), number of ships to be read in from text file

***Int r***: Every function using input from user during game, to store the row chosen

***Int c***: Every function using input from user during game, to store the col chosen

***Int test, test1***: plShips(), test if ship placement has gone off grid or is overlapping

***Int ret***: all return functions, to hold the return integer for every variable that called it

***Int res, res2***: uTurn(), cTurn(), to hold the return values checking for ships sunk

***ComGuess Cmptr***: of ComGuess class, to link main.cpp to the header file and its functions

***UsrShips a***: of UsrShips class, to link main.cpp to the header file and its functions

***Ships r***: of Ships class, to link main.cpp to the header file and its functions

***int hit***: holds the return integer to determine if a ship has been hit or not

***int check***: checks through a loop for the computer guessing mechanism

***char cChar***: used to switch between char and int datatypes

***int dHit***: get return from retHit function in ComGuess class

***int dSunk***: get return from retSunk function in ComGuess class

***string eString***: string used to throw exception if faulty user input is detected

**Concepts**

Included in checkoff sheet

**References**

All code is original code, however, I have used cplusplus.com and w3schools.com for reference on syntax and use.

**Program (main.cpp)**

/\*

\* File: main.cpp

\* Author: Joseph Alipranti

\* Created on December 14th, 2020, 4:25 PM

\* Purpose: Create a game of Battleship between the user and the computer

\*

\* Final Version: With the program basically complete with a debugged "sunk"

\* function, all that is left is to incorporate a few more

\* concepts if and where applicable.

\*/

//System Libraries

#include <iostream> //I/O Library

#include <fstream> //File Stream Library

#include <string.h> //Char Array Manipulation Library

#include <cctype> //For toupper operation

#include <ctime> //For rand operation

#include <string> //String Library

using namespace std;

//User Libraries

#include "TopRow.h"

#include "ComGuess.h"

#include "UsrShips.h"

//Global Constants Only

//Well known Science, Mathematical and Laboratory Constants

const int COLS = 11;

//Function Prototypes

enum Column {A,B,C,D,E,F,G,H,I,J};

topRow \*fillTop(int); //Create top row from a pointer within a class

void mkGrids(char [][COLS],const int); //Create the grid framework

void intro(); //Display intro text

void prnGrid(char [][COLS],topRow \*,const int); //Print grids

void plShips(char [][COLS],const int,ifstream &,UsrShips &); //Place the ships on the grids

int conv(char &); //Convert char input into integer to search through arrays

char revConv(int &); //Reverse convert int input into char to display guess

int match(const int [],const int,const int); //P1 of matching input for ship placement

int match2(const char [][COLS],const int,const int,const int); //P2 of matching input for ship placement

void play(char [][COLS],char [][COLS],char [][COLS],char [][COLS],topRow \*,const int,ComGuess,UsrShips &); //Function stepping through game

void uTurn(char [][COLS],char [][COLS],UsrShips &); //Function for user to make their guesses

void cTurn(char [][COLS],char [][COLS],ComGuess &,UsrShips &); //Function for computer to make their guesses

int hitOrNo(char [][COLS],const int,const int); //Determine if hit or miss

int reGuess(char [][COLS],const int,const int); //Evaluate if user/computer guess is repeated

int chckEnd(char [][COLS],const int); //Check if all ships sunk

int sunk(int,UsrShips &); //Check if a ship has sunk

int checkR(int); //Error testing with exception

void totLeft(const Ships &); //Display total fleet left

//Execution of Code Begins Here

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

//Set the random number seed here

srand(time(0));

//Declare all variables for this function

int n=0;

int shipNum=5;

ifstream inFile;

const int MAX = 10;

topRow \*topArr=fillTop(MAX);

char usrGrid[MAX][COLS]; //Define user grid

char comGrid[MAX][COLS]; //Define computer grid

char tarGrid[MAX][COLS]; //Define target grid

char ctGrid[MAX][COLS]; //Define computer target grid

ComGuess cmptr;

UsrShips a;

Ships r;

a.mkShips(shipNum);

//Initialize all known variables

//Process Inputs to Outputs -> Mapping Process

//Maps known values to the unknown objectives

mkGrids(usrGrid,MAX);

mkGrids(comGrid,MAX);

mkGrids(tarGrid,MAX);

mkGrids(ctGrid,MAX);

//Display the Inputs/Outputs

intro();

cout<<"My Grid"<<endl;

plShips(usrGrid,MAX,inFile,a);

a.next();

plShips(comGrid,MAX,inFile,a);

prnGrid(usrGrid,topArr,MAX);

cout<<endl<<"Target Grid"<<endl;

prnGrid(tarGrid,topArr,MAX);

cout<<endl;

play(usrGrid,tarGrid,comGrid,ctGrid,topArr,MAX,cmptr,a);

cout<<"To begin the game, there were ";

totLeft(r);

cout<<"To end the game, there are ";

totLeft(a);

cout<<endl<<"Thanks for playing!"<<endl;

//Clean up the code, close files, deallocate memory, etc....

delete []topArr;

//Exit stage right

return 0;

}

topRow \*fillTop(int max){

int n=1;

int f=0;

int c=1;

char getC;

topRow \*a=new topRow[n];

a[f].getSize(max); //Set size in topRow class

a[f].setTop(); //Set new size for topGrid array in topRow class

for(int i=A;i<=J;i++){

getC=revConv(c);

a[f].fillTop(i,getC); //Fill topGrid array with chars in topRow class

c++;

}

return a; //return the dynamic array

}

//Function Implementations

void mkGrids(char grid[][COLS],const int rows){

int lBound=47, //Set to ASCII decimal of the char before 0

first=0;

for(int r=0;r<rows;r++){

lBound++;

grid[r][first]=lBound; //y-axis of grid made up of numbers in the char array

for(int c=1;c<COLS;c++){

grid[r][c]='+'; //Fill rest of grid with +'s

}

}

}

void intro(){

int max=35;

string name;

char title[35]="Welcome to the game of Battleship!";

//Introductory comments, displaying basic rules and objectives of the game

//and how it will run

cout<<title<<endl<<endl<<

"Here you will play against a computer, taking turns guessing where"<<endl<<

"eachother's opponent has placed their ships, with the objective of sinking"<<endl<<

"each ship. A ship is sank when every grid point that the ship resides"<<endl<<

"on has been hit. Whichever player sinks the entirety of the opponent's"<<endl<<

"fleet wins the game. Your ships will be marked with o's, and every time"<<endl<<

"a part of your ship is hit, it will be marked as an X. Your guesses will"<<endl<<

"come up as an X if you made a hit, or an m if you missed your opponent."<<endl<<

"You will also be notified if and when you or your enemy has sunken any ships."<<endl<<endl;

cout<<"Who has the pleasure of playing this game? (Enter you name): ";

getline(cin,name);

cout<<endl;

//Comments leading to the display of the starting version of user's grid

//and target grid

cout<<name<<", here is your grid ('My Grid'), as well as the grid where you guess the"<<endl<<

"location of your opponent's ships ('Target Grid'). You both will have"<<endl<<

"your ships set up on the grid randomly, so you may begin playing."<<endl<<endl;

}

void prnGrid(char grid[][COLS],topRow \*arr,const int rows){

int n=0;

cout<<" ";

for(int i=0;i<rows;i++)

cout<<" "<<arr[n].prnTop(i); //Format the top x-axis of the grid

cout<<endl;

for(int i=0;i<rows;i++){

for(int j=0;j<COLS;j++){

cout<<grid[i][j]<<" "; //Format the rest of the grid below

}

cout<<endl;

}

}

void plShips(char grid[][COLS], const int rows, ifstream &file,UsrShips &record){

file.open("ships.txt");

int max=6, //Max length of char array if including null terminator

ships=5, //Number of ships used for game

size=0, //Size of ship, determined when read from file

r=0, //Initializing rows

c=0, //Initializing columns

j=0, //Initializing increment variable for 1st part of if/else

k=0; //Initializing increment variable for 2nd part of if/else

int test,

test1;

int \*rTaken=new int[max-1]; //1D Dynamic Array to hold random row designations as they come

for(int i=0;i<(max-1);i++)

rTaken[i]=0; //Fill array with 0's to be out of range of the flag

char ship[max]; //Char array to get input from file

if(file){

for(int i=0;i<ships;i++){

int s=i;

file.getline(ship,max); //Read line from file

size=strlen(ship); //Determine size of line to get the size of ship

if(size>3){

r=rand()%9+0; //Random row from 0 to 9 to avoid exceeding grid

test=match(rTaken,(max-1),r); //Compare coordinate to array to avoid overlapping ships

if(test==-1){

while(test==-1){

r=rand()%9+0; //Repeat of lines 143/144 until open to place ship on grid

test=match(rTaken,(max-1),r);

}

}

rTaken[j]=r; //Update the dynamic array with a new value to compare with after each iteration

c=rand()%6+1; //Random column from 1 to 6 to stay within grid

int l=0;

for(int i=c;i<(c+size);i++){

grid[r][i]='o'; //Add current ship to the grid

record.fill(r,i,s,l);

l++;

}

j++;

}

else{

c=rand()%10+1; //Random column from 1 to 10 to stay within grid

r=rand()%7+0; //Random row from 0 to 7 to stay within grid

test=match2(grid,rows,r,c); //Compare random coordinates to coordinates already occupied

if(test==-1){

while(test==-1||test1==-1){

c=rand()%10+1; //Repeat lines 158/159/160

r=rand()%7+0;

test=match2(grid,rows,r,c);

}

}

int l=0;

for(int i=r;i<(r+size);i++){

grid[i][c]='o'; //Add current ship to the grid

record.fill(i,c,s,l);

l++;

}

k++;

}

}

}

file.close(); //Close file

delete []rTaken; //Deallocate memory from dynamic array

}

int conv(char &a){

char b=toupper(a); //Make every character capitalized

int r=static\_cast<int>(b-64); //Subtract from decimal ASCII value to get grid axis

return r;

}

char revConv(int &a){

int b=a+64; //Add 64 to find char correspondent

char r=static\_cast<char>(b); //Cast the decimal int as a char

return r;

}

int match(const int array[],const int tot,const int n){

int ret=n;

for(int i=0;i<tot;i++){

if(n==array[i])

ret=-1; //If random coordinate already used/placed in array, return -1

}

return ret;

}

int match2(const char grid[][COLS],const int max,const int r,const int c){

int ret=0;

if(grid[r-1][c]=='o'||grid[r][c]=='o'||grid[r+1][c]=='o'||grid[r+2][c]=='o'||grid[r+3][c]=='o')

ret=-1; //If random coordinates are already used/placed in array, return -1

if(grid[r-1][c+1]=='o'||grid[r][c+1]=='o'||grid[r+1][c+1]=='o'||grid[r+2][c+1]=='o'||grid[r+3][c+1]=='o')

ret=-1; //If random coordinates are already used/placed in array, return -1

if(grid[r-1][c-1]=='o'||grid[r][c-1]=='o'||grid[r+1][c-1]=='o'||grid[r+2][c-1]=='o'||grid[r+3][c-1]=='o')

ret=-1; //If random coordinates are already used/placed in array, return -1

return ret;

}

void play(char uGrid[][COLS],char tGrid[][COLS],char cGrid[][COLS],char ctGrid[][COLS],topRow \*topArr,const int rows,ComGuess cmptr,UsrShips &a){

int res;

int i=0;

cout<<"User starts. Guess the location of an enemy ship!"<<endl;

while(i<(rows\*(COLS-1))){

//Go to function for user's turn

uTurn(tGrid,cGrid,a);

//Check if user has sunk all computer's ships

res=chckEnd(cGrid,rows);

if(res==-1){

cout<<endl<<"You have sunk all of the computer's ships, and won the game!"<<endl;

return;

}

//Go to function for computer's turn

cTurn(uGrid,ctGrid,cmptr,a);

//Check if computer has sunk all user's ships

res=chckEnd(uGrid,rows);

if(res==-1){

cout<<endl<<"Oh no! The computer has sunk your fleet! Computer has won the game."<<endl;

return;

}

cout<<"My Grid"<<endl;

prnGrid(uGrid,topArr,rows);

cout<<endl<<"Target Grid"<<endl;

prnGrid(tGrid,topArr,rows);

i++;

}

}

void uTurn(char tGrid[][COLS],char cGrid[][COLS],UsrShips &a){

int r, //Row input

t, //temporary for row

c, //Column input

res, //Get return from reGuess function

res2, //Get return from sunk function

hit, //Get return from hitOrMiss

num=0; //Initializing number to use later for calculation

char cChar; //Hold char to switch between integer values

cout<<endl<<"Enter the x-coordinate (Letter A through J, not case sensitive): ";

cin>>cChar;

if(cChar<65||cChar>74&&cChar<97||cChar>106){ //If char received is not decimal representation of A to J

while(cChar<65||cChar>74&&cChar<97||cChar>106){

cout<<"Not a valid x-coordinate. Please enter a letter A through J: ";

cin>>cChar;

}

}

c=conv(cChar); //Convert char to integer

cout<<"Enter the y-coordinate (Number 0 through 9): ";

cin>>r;

//Beginning of exception

try{

//Send row to function to see if it is in range

t=checkR(r);

}

catch(string eString){

//Catches exception thrown from function since input is faulty

cout<<eString;

cin>>r;

//Stay in loop until user gives good input

while(r<0||r>9){

cout<<"Not a valid y-coordinate. Please enter a number 0 through 9: ";

cin>>r;

}

}

//Check if guess is a repeat

res=reGuess(tGrid,r,c);

if(res==-1){

while(res==-1){

cout<<"You have previously guessed that coordinate. Please enter a new coordinate"<<endl<<

"Enter the x-coordinate (Letter A through J, not case sensitive): ";

cin>>cChar;

if(cChar<65||cChar>74&&cChar<97||cChar>106){ //If char received is not decimal representation of A to J

while(cChar<65||cChar>74&&cChar<97||cChar>106){

cout<<"Not a valid x-coordinate. Please enter a letter A through J: ";

cin>>cChar;

}

}

c=conv(cChar);

cout<<"Enter the y-coordinate (Number 0 through 9): ";

cin>>r;

if(r<0||r>9){ //If integer received is not between 0 and 9

while(r<0||r>9){

cout<<"Not a valid y-coordinate. Please enter a number 0 through 9: ";

cin>>r;

}

}

//Check if guess is a repeat

res=reGuess(tGrid,r,c);

}

}

cout<<endl;

int k=1;

hit=hitOrNo(cGrid,r,c); //Call function to determine if guess is a hit or not

if(hit!=-1){

//Calculate values that correspond to the data in the ship arrays

num=r+(c\*10);

//Determine which ship, and what part of the ship was hit

a.hit(num,k);

cout<<"You have hit an enemy boat at ("<<cChar<<

","<<r<<")!"<<endl;

cGrid[r][c]='X'; //Mark x on computer's grid if ship is hit

res2=sunk(k,a);

if(res2==-1)

cout<<"You have sunk an enemy ship!"<<endl;

tGrid[r][c]='X'; //Mark x for hit in target grid

}

else{

cout<<"You missed at ("<<cChar<<","<<r<<")."<<endl;

tGrid[r][c]='m'; //Mark m for miss in target grid

}

}

void cTurn(char uGrid[][COLS],char ctGrid[][COLS],ComGuess &cmptr,UsrShips &a){

int r, //Row input

c, //Column input

res, //Get return from reGuess function

res2, //Get return from sunk function

hit, //Get return from hitOrMiss

check=1, //To check which count in loop between 1 and 4

num=0; //Initializing number to use later for calculation

char cChar; //Hold char to switch between integer values

//Get didHit from retHit function in ComGuess class

int dHit=cmptr.retHit();

//Get didSink from retSunk function in ComGuess class

int dSunk=cmptr.retSunk();

//If previous guess was a hit, but didn't sink the ship

if(dHit==1 && dSunk!=1){

//Get new row from ComGuess class for computer to make accurate guess

r=cmptr.nearRow(check);

//Get new col from ComGuess class for computer to make accurate guess

c=cmptr.nearCol();

cChar=revConv(c); //Convert x-axis integer to char

}

else{

c=rand()%10+1; //Computer picks random x-axis between 1 and 10 (A and J)

cChar=revConv(c); //Convert x-axis integer to char

r=rand()%9+0; //Computer picks random y-axis between 0 and 9

}

//Check if guess is a repeat

res=reGuess(ctGrid,r,c);

if(res==-1){

//If previous guess was a hit, but didn't sink the ship

if(dHit==1 && dSunk!=1){

check=2;

while(check<5&&res==-1){

//Get new row from ComGuess class for computer to make accurate guess

r=cmptr.nearRow(check);

//Get new col from ComGuess class for computer to make accurate guess

c=cmptr.nearCol();

cChar=revConv(c); //Convert x-axis integer to char

//Check if guess is a repeat

res=reGuess(ctGrid,r,c);

check++;

}

}

while(res==-1){

c=rand()%10+1; //Computer picks random x-axis between 1 and 10 (A and J)

cChar=revConv(c); //Convert x-axis integer to char

r=rand()%9+0; //Computer picks random y-axis between 0 and 9

//Check if guess is a repeat

res=reGuess(ctGrid,r,c);

}

}

//Set didHit back to 0

cmptr.getHit(0);

//Set didSink back to 0

cmptr.getSunk(0);

int k=0;

hit=hitOrNo(uGrid,r,c); //Call to determine if guess is hit or miss

if(hit!=-1){

cmptr.getHit(1); //Set didHit in ComGuess class to 1

cmptr.getNums(r,c); //Copy r and c to row and col in ComGuess class

//Calculate values that correspond to the data in the ship arrays

num=r+(c\*10);

//Determine which ship, and what part of the ship was hit

a.hit(num,k);

cout<<"The computer has hit your boat at ("<<cChar<<

","<<r<<")!"<<endl;

uGrid[r][c]='X'; //Mark x in user's grid if user ship is hit

res2=sunk(k,a);

if(res2==-1){

cout<<"The enemy has sunk one of your ships!"<<endl<<endl;

cmptr.getSunk(1); //Set didSink in ComGuess class to 1

}

ctGrid[r][c]='X'; //Fill computer's target grid

}

else{

cout<<"The computer missed at ("<<cChar<<","<<r<<")."<<endl<<endl;

ctGrid[r][c]='m'; //Fill computer's target grid

}

}

int hitOrNo(char grid[][COLS],const int row,const int col){

int ret=1;

if(grid[row][col]=='o'){ //If part of ship resides in the guessed coordinate

grid[row][col]='X'; //Mark x in grid if ship is hit

}

else

ret=-1; //Return -1 if miss

return ret;

}

int reGuess(char grid[][COLS],const int r,const int c){

int ret=1;

if(grid[r][c]=='X'||grid[r][c]=='m')

ret=-1; //If coordinate shows up as x or m, coordinate has been previously guessed

return ret;

}

int chckEnd(char grid[][COLS],const int rows){

int ret=-1;

for(int i=0;i<rows;i++){

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

if(grid[i][j]=='o') //If no coordinate holds an 'o', then all ships have sunk

ret=1;

}

}

return ret;

}

//Altered Sunk function to use classes

int sunk(int n,UsrShips &a){

int ret;

//Get result from Sunk function in UsrShips class

ret=a.sunk(n);

return ret;

}

//Function to throw exception if not in range

int checkR(int r){

if(r<0||r>9){

string eString = "Not a valid y-coordinate. Please enter a number 0 through 9: ";

//throw eString to catch()

throw eString;

}

else

return r; //return as is if in range

}

//Display the fleet started with, and ended with.

void totLeft(const Ships &n){

cout<<n.fleet()<<" total units on the board."<<endl;

}