|  |
| --- |
|  |
| Battleship Game |
| CSC5 45277 |

|  |
| --- |
| Alec Nguyen  Summer 2016 |

Table of Contents

[Introduction 2](#_Toc457572140)

[Rules 2](#_Toc457572141)

[Objective 2](#_Toc457572142)

[How to Play 2](#_Toc457572143)

[Approach 2](#_Toc457572144)

[Similarities 2](#_Toc457572145)

[Differences 2](#_Toc457572146)

[Pseudo-Code 3](#_Toc457572147)

[Flowcharts 4](#_Toc457572148)

[Constructs & Concepts 9](#_Toc457572149)

[Source Code 10](#_Toc457572150)

# **Introduction**

Battleship is a popular pen and paper guessing game from the 1930’s. Two players play on four grids and must correctly guess where the other player’s ships are in a coordinate system placed on top on the grid. The grid system and random placement of the ships made this game an obvious choice to implement multidimensional arrays and random number generators into a game. This is a simplified version where the player plays against the computer, without them marking the player board.

# **Rules**

## **Objective**

To sink all of the opposing player’s battleships before running out of moves.

## **How to Play**

1. The player selects the difficulty setting of the game, by generating the number of plots taken on the game board.
2. The player then must input the (y, x) coordinates into the computer to attack the computer’s board.
3. This continues until the player eliminates all enemy ships or runs out of moves.

# **Approach**

## **Similarities**

* The game is a one sided version of the actual game with two players, in that the user player must randomly select the coordinates to find the computer’s ships.

## **Differences**

* To add a fun element other than attacking a random board, I added a point system based on how many plots are on the board, with a lower total possible point making the game more difficult.
* In this game the player also has a limited amount of moves rather than playing until one player loses, this forces the player to choose between sinking ships or playing for points.

# **Pseudo**-**Code**

*Intro screen1*

*Stat sheet prompt1*

*Player enters name to recorded on their stat sheet*

*Blank boards are filled one representing the user, the other representing the computer5*

*Difficulty Info text displayed1*

*Player determines difficulty by choosing the enemy fleet size3*

*Control info text displayed1*

*Game begins2, and loops until enemy is defeated or runs out of moves*

*Enemy board is shown*

*User inputs (y, x) coordinates to choose a coordinate to attack*

*New updated board is printed with a marker indicating a hit or miss*

*The game determines whether or not the player lost4*

*Displays final board5*

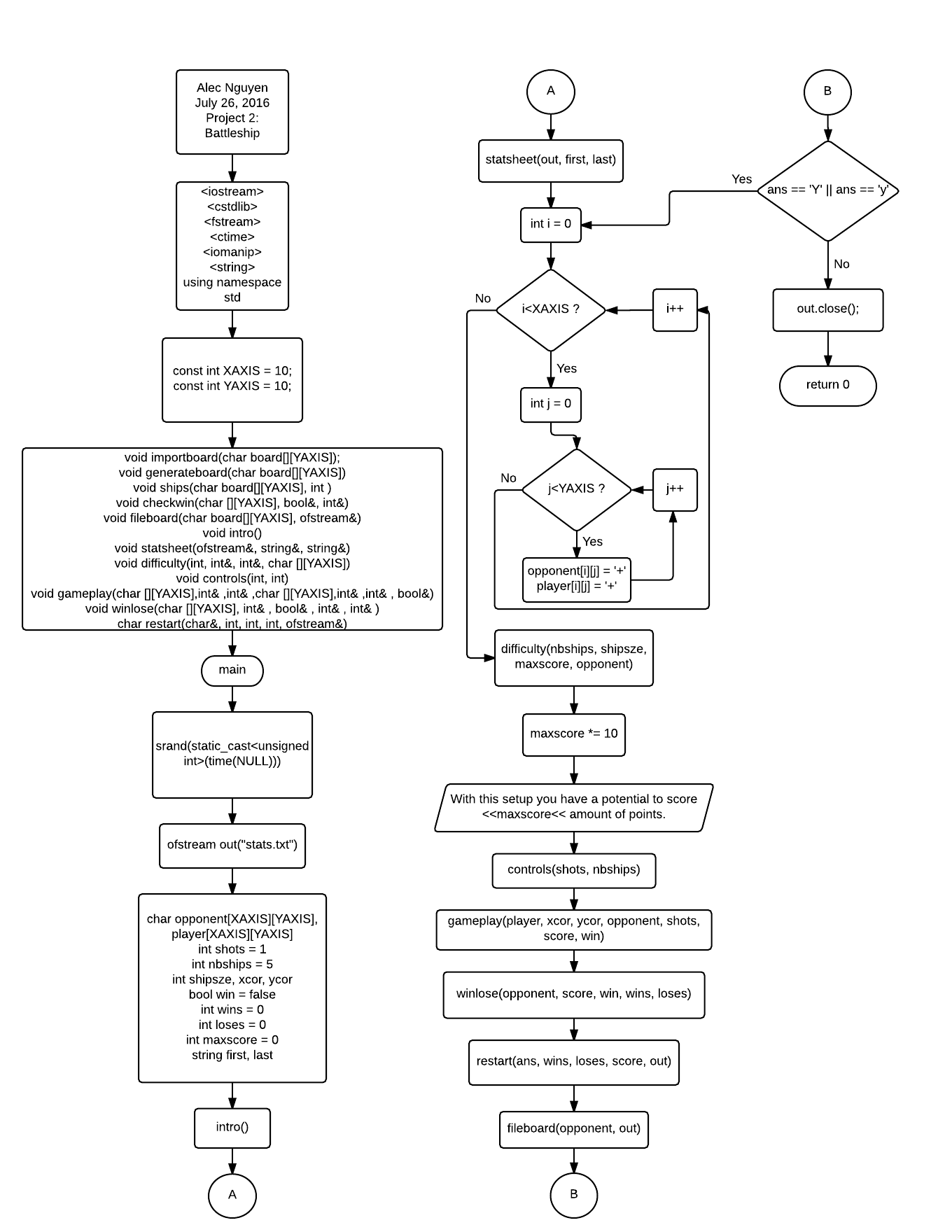
*Displays final score*

*Writes data to stat sheet file5*

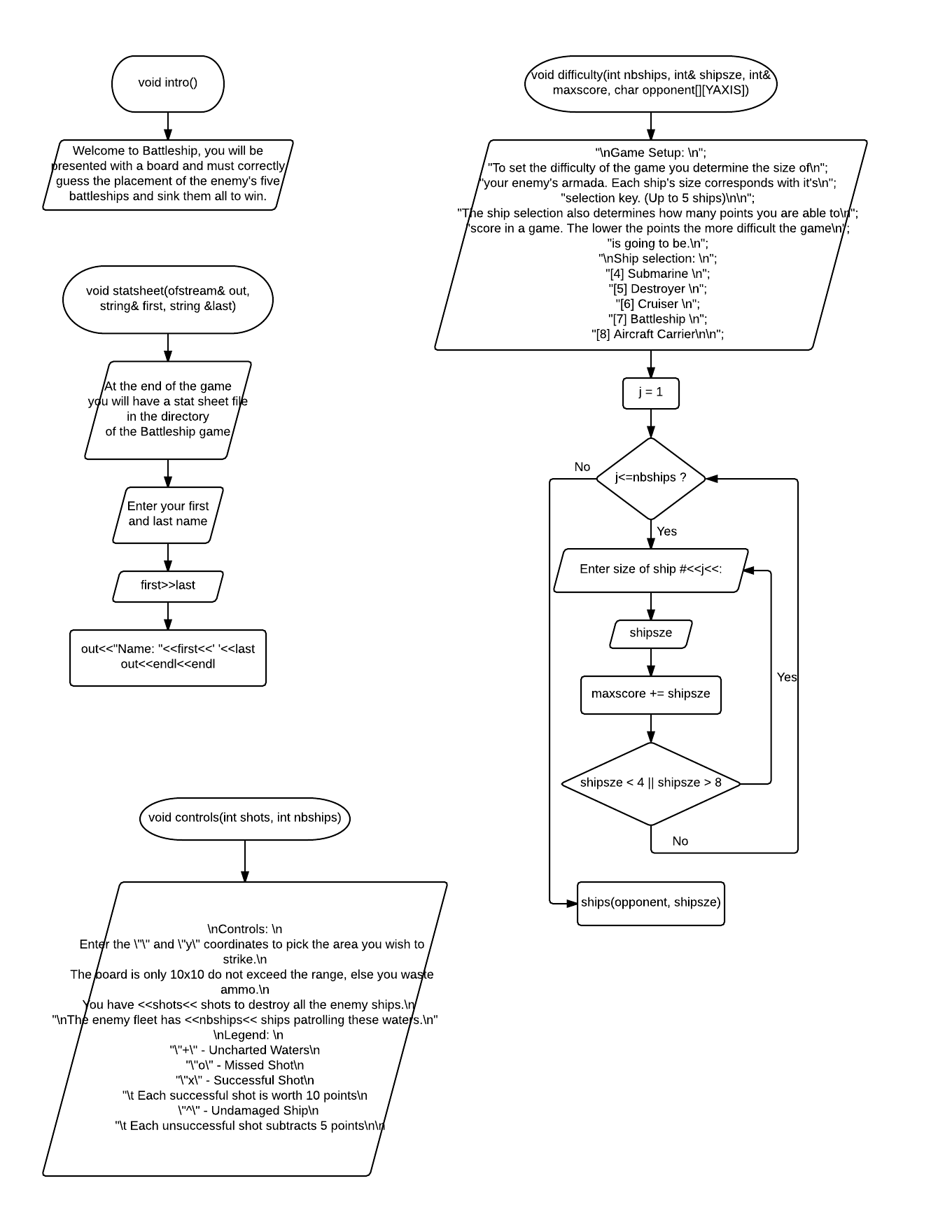
*Games asks user to play again5 looping back to the difficulty setting1*

(Pseudo-Code is marked to show where each specific function is used in the following flowcharts, full-sized flowcharts are included within the project folder.)

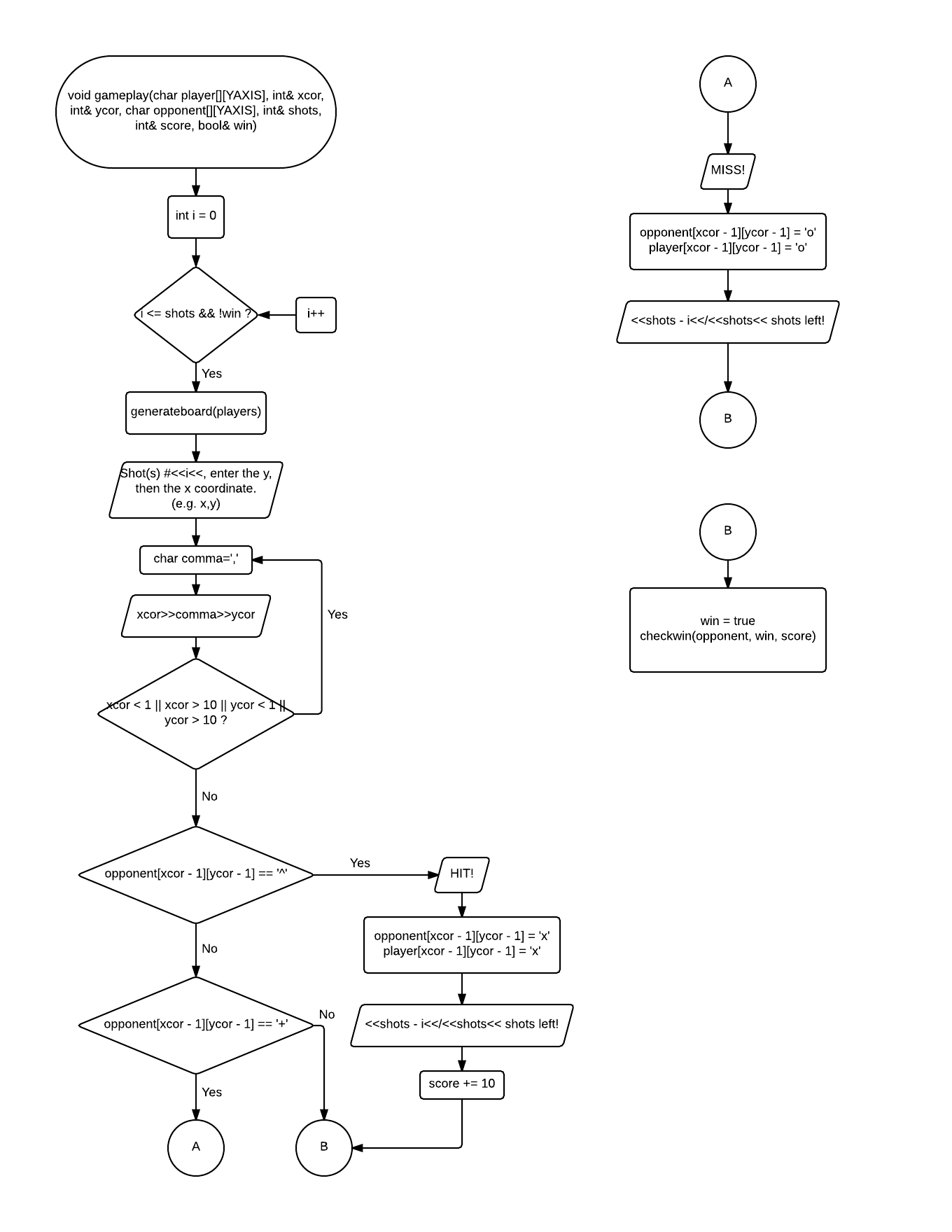
# **Flowcharts**



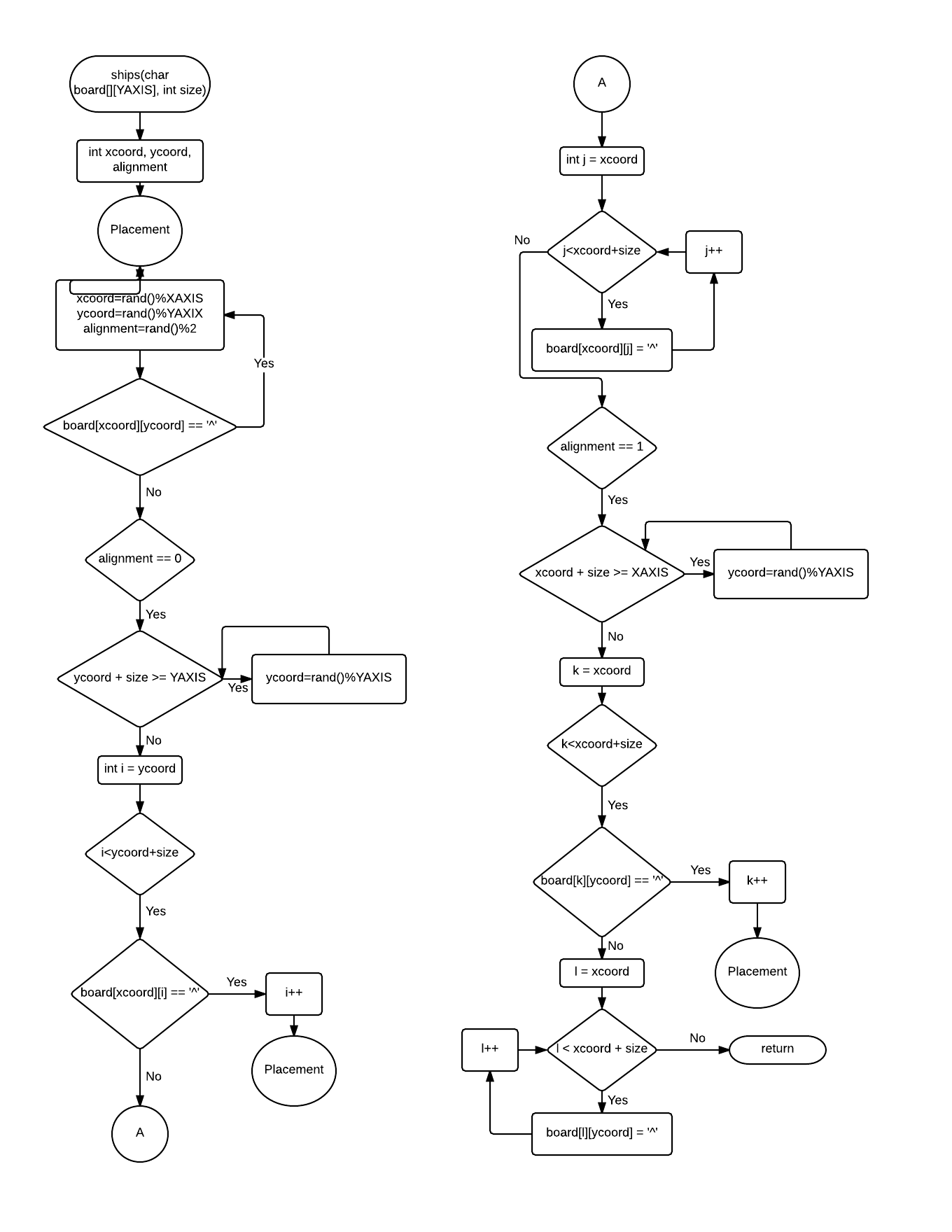
[1]



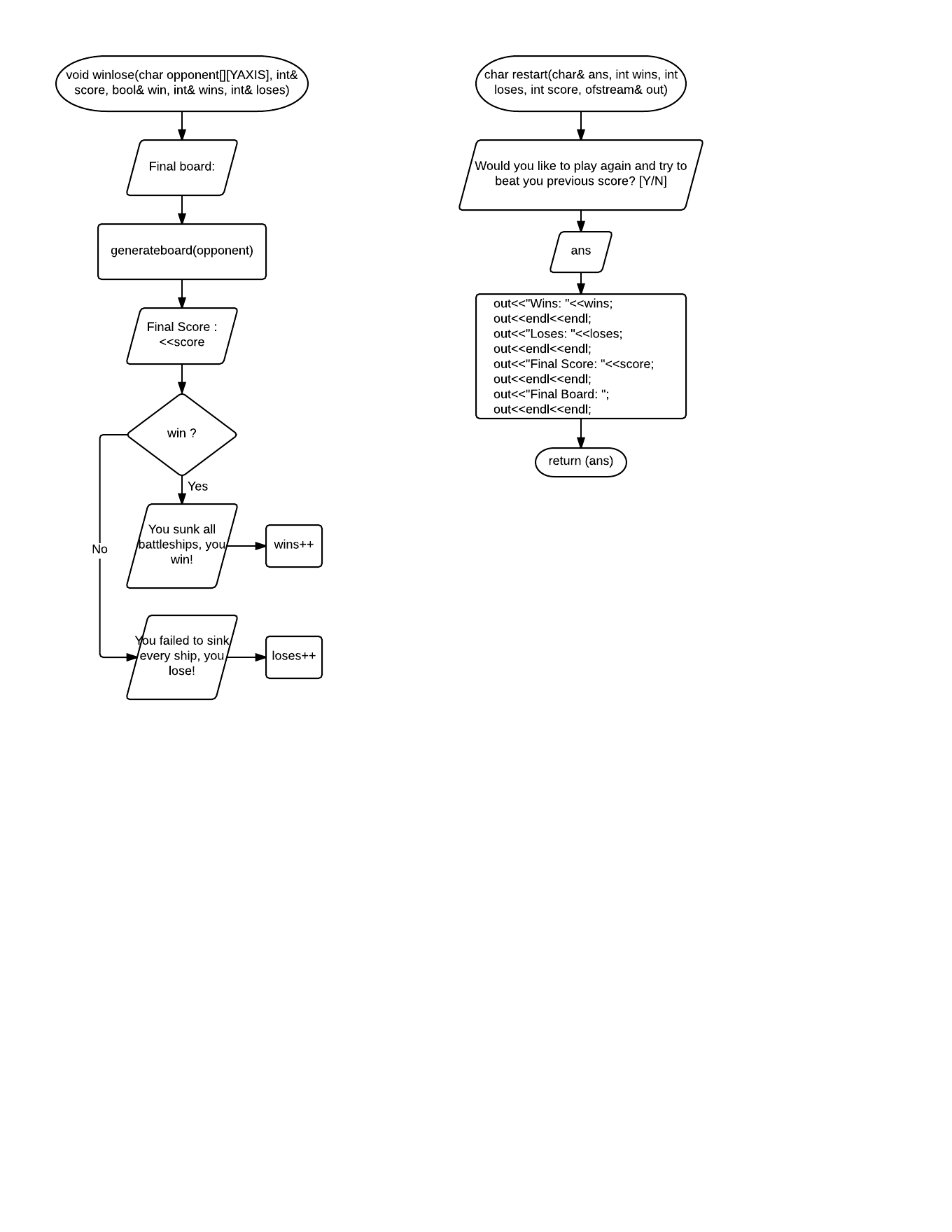
[2]



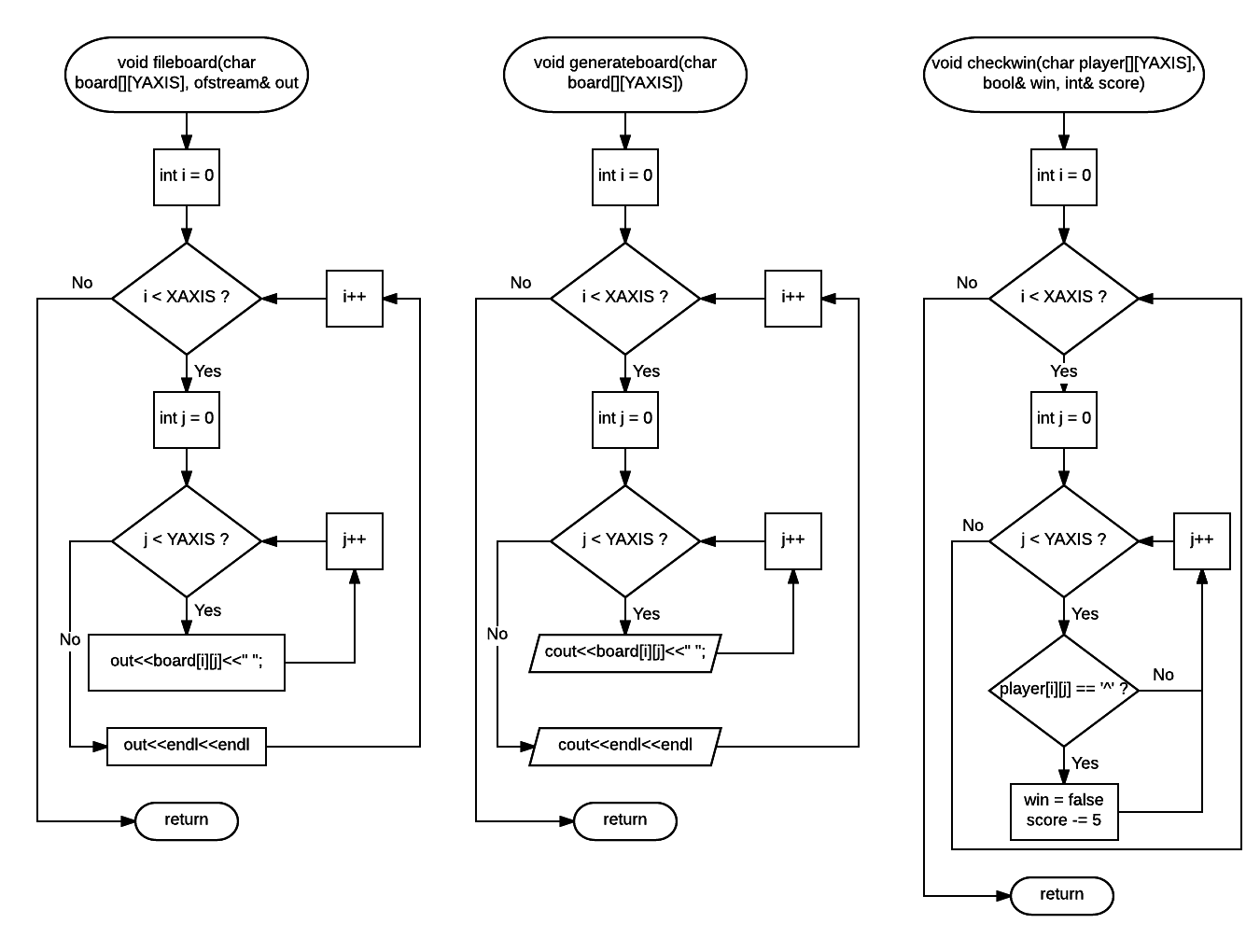
[3]



[4]



[5]



# **Constructs** **&** **Concepts**

**iostream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| static\_cast |  | *different variable* | Line 39 |
| cout |  |  | Throughout |
| cin |  |  | Throughout |

**cstdlib Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| srand() |  |  | 39 |
| rand() |  |  | 39 |

**ctime Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| time() |  |  | 39 |

**iomanip Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| setw() |  |  | 233, 241 |

**string Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| string |  |  | 56 |

**fstream Library**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Frequency | Description | Location |
| out.close() |  |  | 93 |
| out | 14 | *write to file* | 101, 103, 266-273 |
| ofstream | 1 | *variable* | 56 |

# **Source** **Code**

/\*

\* File: main.cpp

\* Author: Alec Nguyen

\* Created on July 26, 2016

\* Purpose: Project 2 - Battleship

\*/

//system libraries

#include <iostream> //Input/Output Library

#include <cstdlib> //Random Number

#include <fstream> //file streamsR

#include <ctime> //Time

#include <iomanip> //Formatting

#include <string>

using namespace std;//Namespace of the System Libraries

//User Libraries (Libraries created by the user)

//Global Constants

const int XAXIS = 10; //Number of Rows

const int YAXIS = 10; //Number of Columns to create 10X10 playspace

//Function Prototypes

void importboard(char board[][YAXIS]); //User imported board

void generateboard(char board[][YAXIS]); //Prints out board

void ships(char board[][YAXIS], int ); //Ship placement

void checkwin(char [][YAXIS], bool&, int&); //checks if user wins

void fileboard(char board[][YAXIS], ofstream&); //writes board to file

void intro();

void statsheet(ofstream&, string&, string&);

void difficulty(int, int&, int&, char [][YAXIS]);

void controls(int, int);

void gameplay(char [][YAXIS],int& ,int& ,char [][YAXIS],int& ,int& , bool&);

void winlose(char [][YAXIS], int& , bool& , int& , int& );

char restart(char&, int, int, int, ofstream&);

//Execution Begins Here!

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

//Set random number seed

srand(static\_cast<unsigned int>(time(NULL)));

//Files stream

ofstream out("stats.txt");

//Declare Variables

char opponent[XAXIS][YAXIS], player[XAXIS][YAXIS]; //Separate Player/Computer boards

int shots = 1; //Number of attempts user has to hit enemy ships

int nbships = 5; //Number of Battleships

int shipsze; //"Peg Length" of each ship

int xcor, ycor; //coordinate selection

bool win = false; //if win is true then player wins

int wins = 0; //win counter

int loses = 0; //loss counter

char ans = 'y'; //Retry screen

int score = 0; //Score

int maxscore = 0; //Max score

string first, last; //Player name

//Intro Text

intro();

//Stat Sheet Prompt

statsheet(out, first, last);

do{

//Fills player and computer 'blank board'

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

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

opponent[i][j] = '+';

player[i][j] = '+';

}

}

//Difficulty Setting

difficulty(nbships, shipsze, maxscore, opponent);

maxscore \*= 10;

cout<<"\nWith this setup you have a potential to score "<<maxscore<<" amount";

cout<<" of points.\n";

//Game Controls

controls(shots, nbships);

//Game-play

gameplay(player, xcor, ycor, opponent, shots, score, win);

//Win-Lose Screen

winlose(opponent, score, win, wins, loses);

//Restart Screen

restart(ans, wins, loses, score, out);

//Write Board to stat.txt file

fileboard(opponent, out);

}while(ans == 'Y' || ans == 'y');

out.close();

//Exit Stage Stage!

return 0;

}

//Function writes a copy of the final board to the file

void fileboard(char board[][YAXIS], ofstream& out){

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

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

out<<board[i][j]<<" ";

}

out<<endl<<endl;

}

}

//Function prints the play field, after each move and for the final board.

void generateboard(char board[][YAXIS]){

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

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

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

}

cout<<endl<<endl;

}

}

//Function will randomly place the ships chosen by the user

void ships(char board[][YAXIS], int size){

int xcoord;

int ycoord;

int alignment;

//Randomize X and Y coordinates

Placement: //goto and 'Placement' label allow this function to repeat and restart as necessary

do{

xcoord = rand() % XAXIS;

ycoord = rand() % YAXIS;

alignment = rand() % 2;

}while(board[xcoord][ycoord] == '^');

if(alignment == 0){

while(ycoord + size >= YAXIS){

ycoord = rand() % YAXIS;

}

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

if(board[xcoord][i] == '^')

goto Placement;

}

for(int j = ycoord; j <ycoord + size; j++){

board[xcoord][j] = '^';

}

}

if(alignment == 1){

while(xcoord + size >= XAXIS){

xcoord = rand() % XAXIS;

}

for(int k = xcoord; k < xcoord + size; k++){

if(board[k][ycoord] == '^')

goto Placement;

}

for(int l = xcoord; l < xcoord + size; l++){

board[l][ycoord] = '^';

}

}

return;

}

//Function checks for undamaged ships

void checkwin(char player[][YAXIS], bool& win, int& score){

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

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

if(player[i][j] == '^'){

win = false;

score -= 5;

}

}

}

}

void intro(){

cout<<"Welcome to Battleship, you will be presented with a board and must \n";

cout<<"correctly guess the placement of the enemy's five battleships and sink\n";

cout<<"them all to win.\n\n";

}

void statsheet(ofstream& out, string& first, string &last){

cout<<"At the end of the game you will have a stat sheet file in the directory\n";

cout<<"of the Battleship game.\n";

cout<<"Enter your first and last name\n";

cin>>first>>last;

out<<"Name: "<<first<<' '<<last;

out<<endl<<endl;

}

void difficulty(int nbships, int& shipsze, int& maxscore, char opponent[][YAXIS]){

cout<<"\nGame Setup: \n";

cout<<"To set the difficulty of the game you determine the size of\n";

cout<<"your enemy's armada. Each ship's size corresponds with it's\n";

cout<<"selection key. (Up to 5 ships)\n\n";

cout<<"The ship selection also determines how many points you are able to\n";

cout<<"score in a game. The lower the points the more difficult the game\n";

cout<<"is going to be.\n";

cout<<"\nShip selection: \n";

cout<<"[4] Submarine \n";

cout<<"[5] Destroyer \n";

cout<<"[6] Cruiser \n";

cout<<"[7] Battleship \n";

cout<<"[8] Aircraft Carrier\n\n";

//Loop will cycle for the size of each individual ship

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

do{

cout<<"Enter size of ship #"<<j<<": ";

cin>>shipsze;

maxscore += shipsze;

}while(shipsze < 4 || shipsze > 8);

ships(opponent, shipsze);

}

}

void controls(int shots, int nbships){

cout<<"\nControls: \n";

cout<<"Enter the \"\" and \"y\" coordinates to pick the area you wish to strike.\n";

cout<<"The board is only 10x10 do not exceed the range, else you waste ammo.\n";

cout<<"You have "<<shots<<" shots to destroy all the enemy ships.\n";

cout<<"\nThe enemy fleet has "<<nbships<<" ships patrolling these waters.\n";

cout<<"\nLegend: \n";

cout<<"\"+\" - Uncharted Waters\n";

cout<<"\"o\" - Missed Shot\n";

cout<<"\"x\" - Successful Shot\n";

cout<<"\t Each successful shot is worth 10 points\n";

cout<<"\"^\" - Undamaged Ship\n";

cout<<"\t Each unsuccessful shot subtracts 5 points\n\n";

}

void gameplay(char player[][YAXIS], int& xcor, int& ycor, char opponent[][YAXIS], int& shots, int& score, bool& win){

for(int i = 1; i <= shots && !win; i++){

generateboard(player);

cout<<"\nShot(s) #"<<i<<", enter the y, then the x coordinate.\n";

cout<<"(e.g. x,y)\n";

do{

char comma = ',';

cin>>xcor;

cin>>comma;

cin>>ycor;

}while(xcor < 1 || xcor > 10 || ycor < 1 || ycor > 10);

//Successful Hit

if(opponent[xcor - 1][ycor - 1] == '^'){

cout<<setw(3)<<"\nHIT!"<<setw(3)<<endl;

opponent[xcor - 1][ycor - 1] = 'x';

player[xcor - 1][ycor - 1] = 'x';

cout<<setw(3)<<shots - i<<"/"<<shots<<" shots left!\n";

score += 10;

//Unsuccessful Hit

}else if(opponent[xcor - 1][ycor - 1] == '+'){

cout<<setw(3)<<"\nMISS!"<<setw(3)<<endl;

opponent[xcor - 1][ycor - 1] = 'o';

player[xcor - 1][ycor - 1] = 'o';

cout<<setw(3)<<shots - i<<"/"<<shots<<" shots left!\n";

}

win = true;

checkwin(opponent, win, score);

}

}

void winlose(char opponent[][YAXIS], int& score, bool& win, int& wins, int& loses){

cout<<"Final board: \n";

generateboard(opponent);

cout<<"Final Score : "<<score;

if(win){

cout<<"\nYou sunk all battleships, you win!\n";

wins++;

}else{

cout<<"\nYou failed to sink every ship, you lose!\n";

loses++;

}

}

char restart(char& ans, int wins, int loses, int score, ofstream& out){

cout<<"\nWould you like to play again and try to beat your \n";

cout<<"previous score? [Y/N]\n";

cin>>ans;

out<<"Wins: "<<wins;

out<<endl<<endl;

out<<"Loses: "<<loses;

out<<endl<<endl;

out<<"Final Score: "<<score;

out<<endl<<endl;

out<<"Final Board: ";

out<<endl<<endl;

return (ans);

}