Project 2

<Battleship>

CSC-5 (41375)

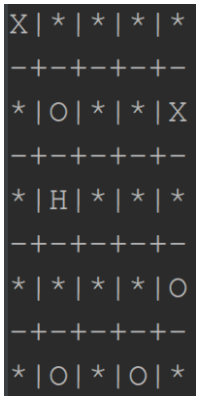
Christian Daniel

02/07/2021

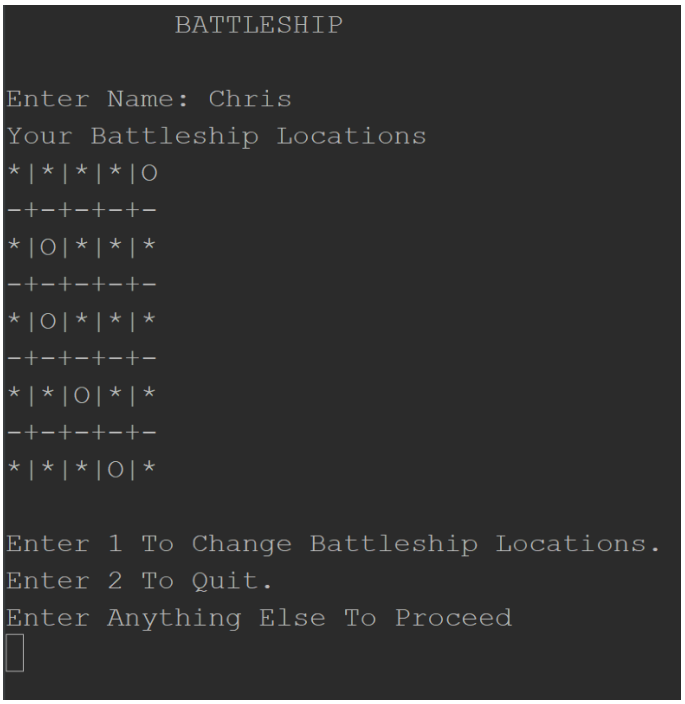
Introduction

Tittle: Battleship

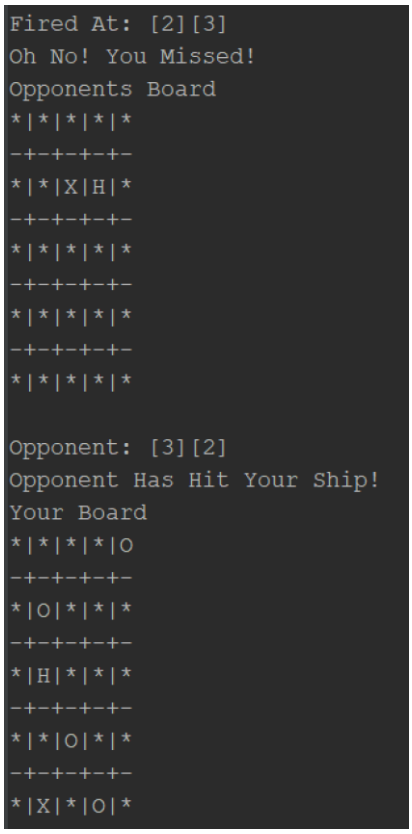
This game of Battleship is played on a 5x5 grid. Empty spaces are represented by ‘*’, ships are represented by ‘O’, hits are represented by ‘H’, and misses are represented by ‘X’.



The user is first asked to enter their name. The program then randomly generates five battleship locations on the users and opponents board. After outputting the users board, the user can decide whether they would like to keep their battleship locations or randomly generate new ones.



Once the user confirms they would like to proceed, the game starts. The users is prompted to enter the location they wish to fire at, by entering the row and column. If the input is invalid, the program will loop and the user will be asked to enter a new row and column. After validating the input, a message will be outputted letting the user know if they hit or miss. Right below that, both players updated boards will be outputted as well as a message notifying the user whether the opponent missed or hit one of their ships.



The game will continue looping until one player wins. If the user wins, their score and number of misses will be outputted, and their name and score get added to the scoreboard text file. If the user losses, a miniature board is outputted revealing the opponents ship locations.

```
Fired At: [5][4]
YOU SUNK AN OPPONENTS SHIP!

:D You Win!
Chris Got A Score Of 97.1
Number Of Misses: 2
```

```
You Lose :(

Opponents Ships
X|*|X|*|*
*|H|*|O|*
*|O|X|X|*
*|*|O|X|*
*|*|*|O|*
```

Scoreboard

```
Player Names                              Score
Chris.....................................100.0
John......................................94.6
Mike......................................37.0
Daniel....................................30.4
```

Summary

Project size: about 300 lines

Number of variables: about 20

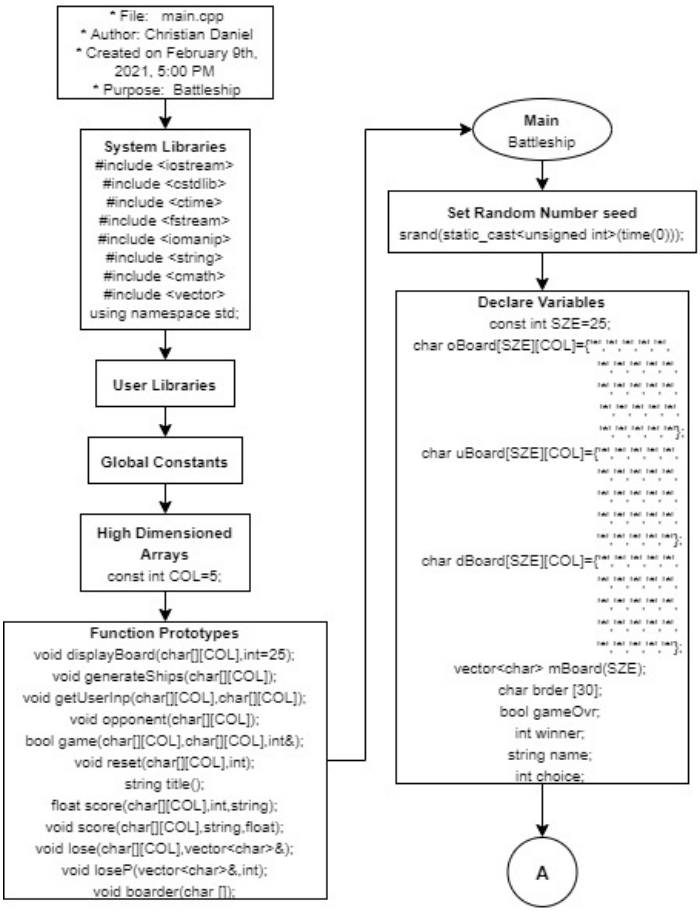
This project is an improved version of Project 1. I added a second dimension to the board by using new concepts we learned, such as arrays and functions.

The coding took me about 15 hours. Since I created a battleship game for project one, I already had an idea on how I wanted to code this game; So, I sat down and coded almost the whole game in version 1, testing it as I implemented new code. At this point, the game is playable, but was still under the 250 minimum line count and was missing some things from the check off sheet. For version 2, I added a scoring system and a scoreboard which saves the users name and score in a text file named “scoreboard.txt”. I also changed my code a bit to incorporate more things from the check off sheet. In version 3, I then continued to clean up my code, adding more comments and making small tweaks to my code. These changes include reading from a file, overloading functions, and adding a mini board that reveals the enemy ships after losing the game.

Major Variables

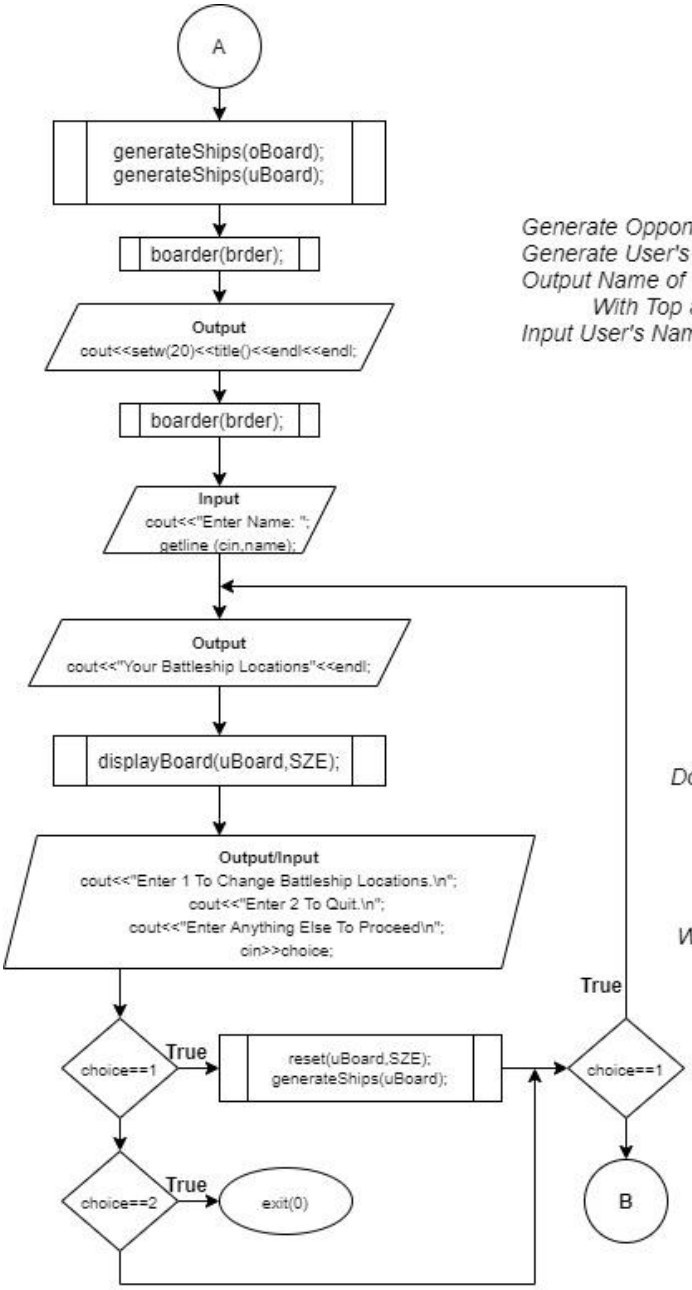
Type	Variable Name	Description
const int	COL	Number of Columns
	SZE	Size of Game Board Array
char	oBoard [SZE][COL]	Opponent Game Board
	uBoard [SZE][COL]	User's Game Board
	dBoard [SZE][COL]	Opponents Displayed Board
	mBoard(SZE)	Miniature Opponent Board Displayed If User Loses
bool	gameOvr	Game Over
int	winner	Winner
float	score	Score Out of 100

Battleship Flowchart



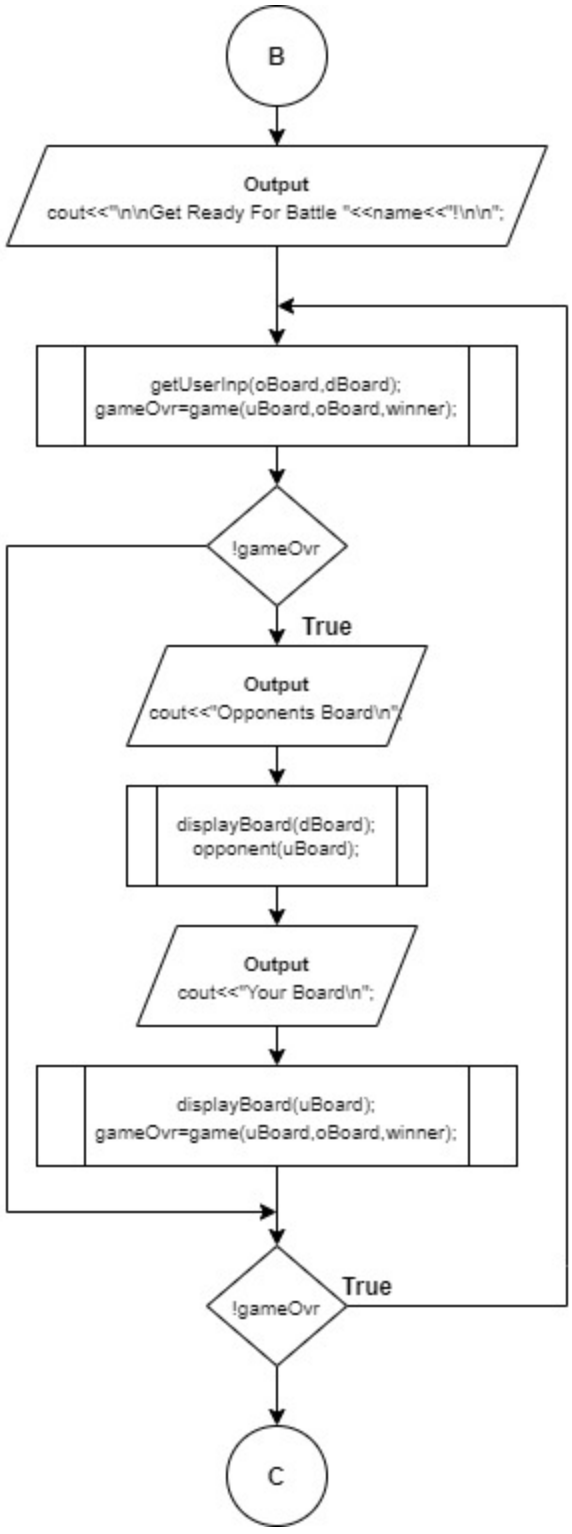
Pseudocode

Bring In System Libraries
Declare High Dimension Array Constant
Declare Function Prototypes
Enter Main Then,
    Set Random Number Seed
    Declare Variables and Initialize Game Boards



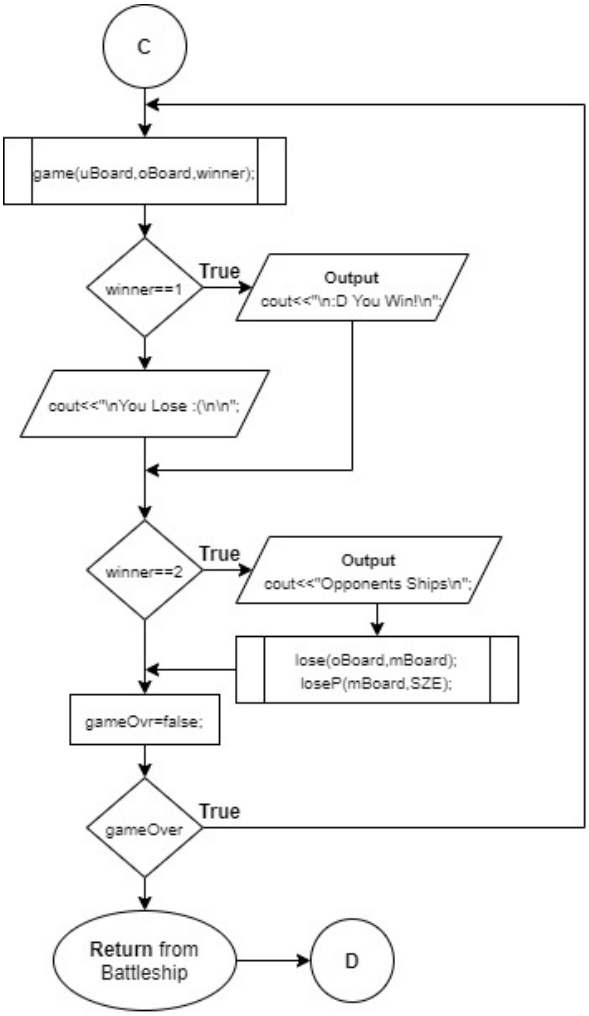
Generate Opponent's Ships
Generate User's Ships
Output Name of Game,
With Top and Bottom Boarder
Input User's Name

Do,
Display User's Board
Input Choice
Enter 1 To Generate New Battleship Locations
Enter 2 To Exit Program
Enter Anything Else To Continue
While Choice Is 1

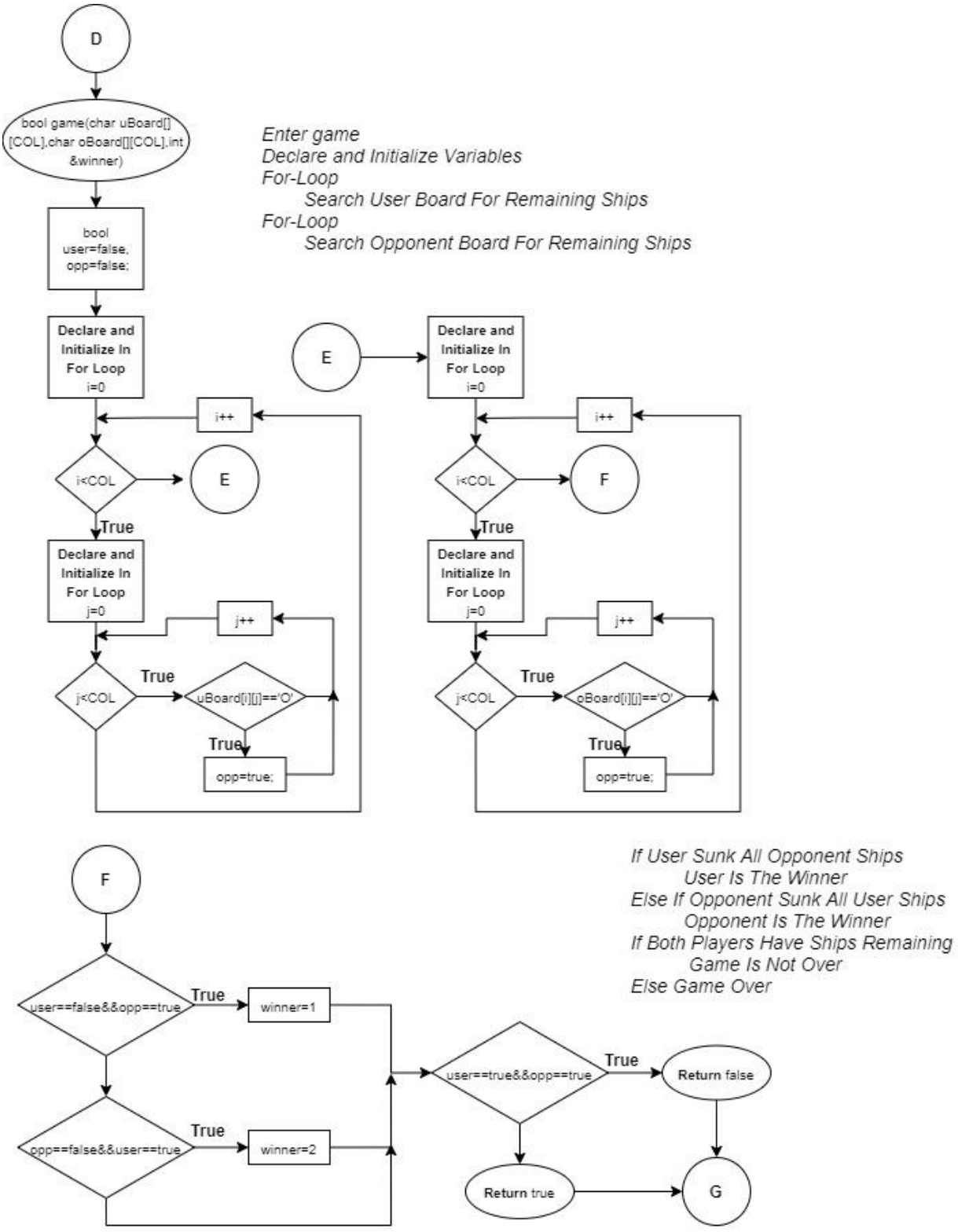


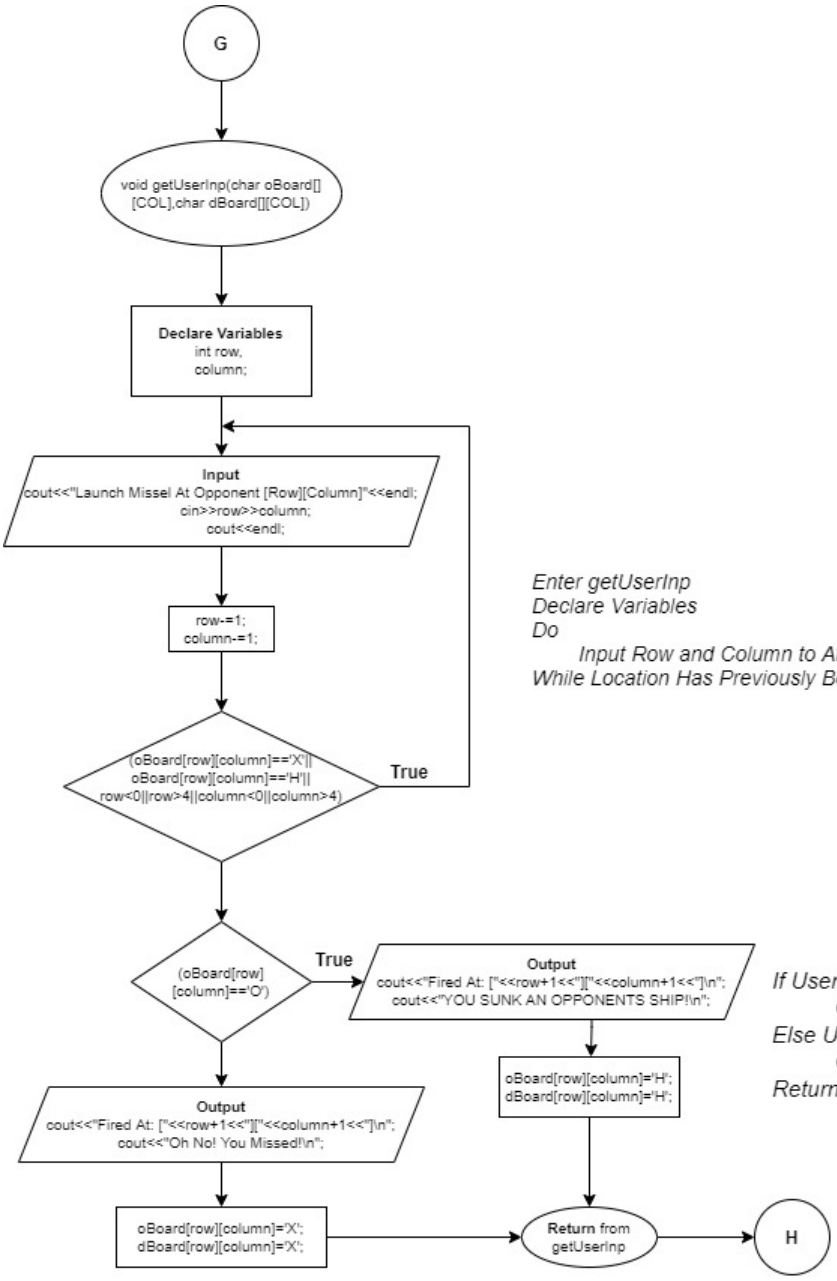
Output Message

Do
Get User Input,
Input Row and Column To Attack
Check If Game Is Over
If Game Is Not Over
Display Opponents Board
Opponent's Turn To Attack
Display User's Board
Check If Game Is Over
While Game Is Not Over



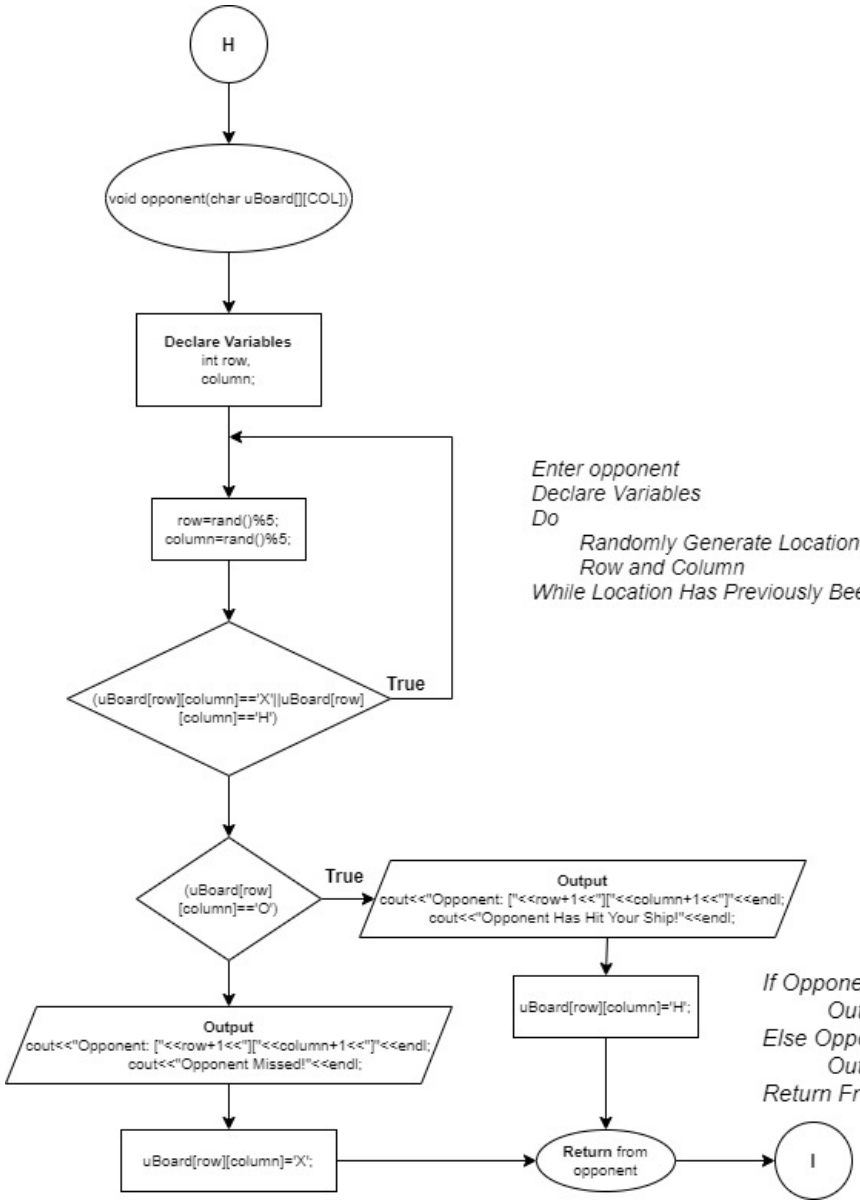
While Game Is Over
Check Who Won
If User Wins
Output Message
Else Output Message
If User Wins
Calculate and Output Score
Output Number Of Misses
If Opponent Wins
Output Mini Board Revealing Opponent Ships
Set GameOvr To False To Exit Loop


D
bool game(char uBoard[][COL],char oBoard[][COL],int &winner)
bool user=false, opp=false;
Declare and Initialize In For Loop i=0
i++
i<COL
E
True
Declare and Initialize In For Loop j=0
j++
j<COL
True
uBoard[i][j]=='O'
True
opp=true;
E
Declare and Initialize In For Loop i=0
i++
i<COL
F
True
Declare and Initialize In For Loop j=0
j++
j<COL
True
oBoard[i][j]=='O'
True
opp=true;
F
user==false&&opp==true
True
winner=1
opp==false&&user==true
True
winner=2
user==true&&opp==true
True
Return false
Return true
G
Enter game
Declare and Initialize Variables
For-Loop
Search User Board For Remaining Ships
For-Loop
Search Opponent Board For Remaining Ships
If User Sunk All Opponent Ships
User Is The Winner
Else If Opponent Sunk All User Ships
Opponent Is The Winner
If Both Players Have Ships Remaining
Game Is Not Over
Else Game Over



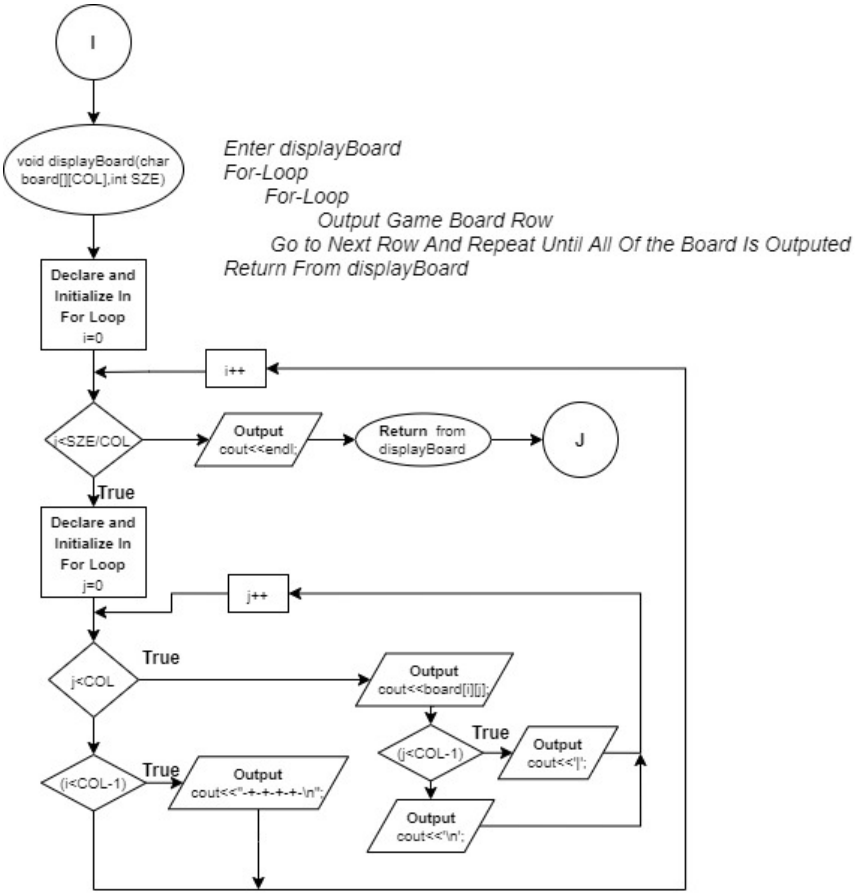
Enter getUserInp
Declare Variables
Do
Input Row and Column to Attack
While Location Has Previously Been Attacked

If User Hit Opponents Ship
Output Message and Set Location to 'H'
Else User Missed
Output Message and Set Location to 'X'
Return From getUserInp

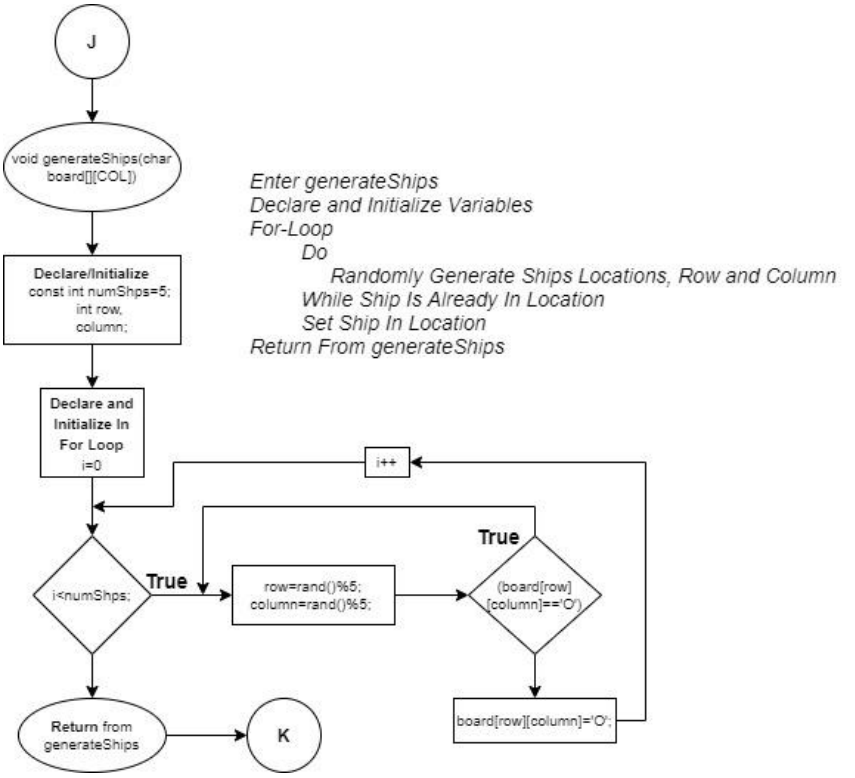


Enter opponent
Declare Variables
Do
Randomly Generate Location To Attack
Row and Column
While Location Has Previously Been Attacked

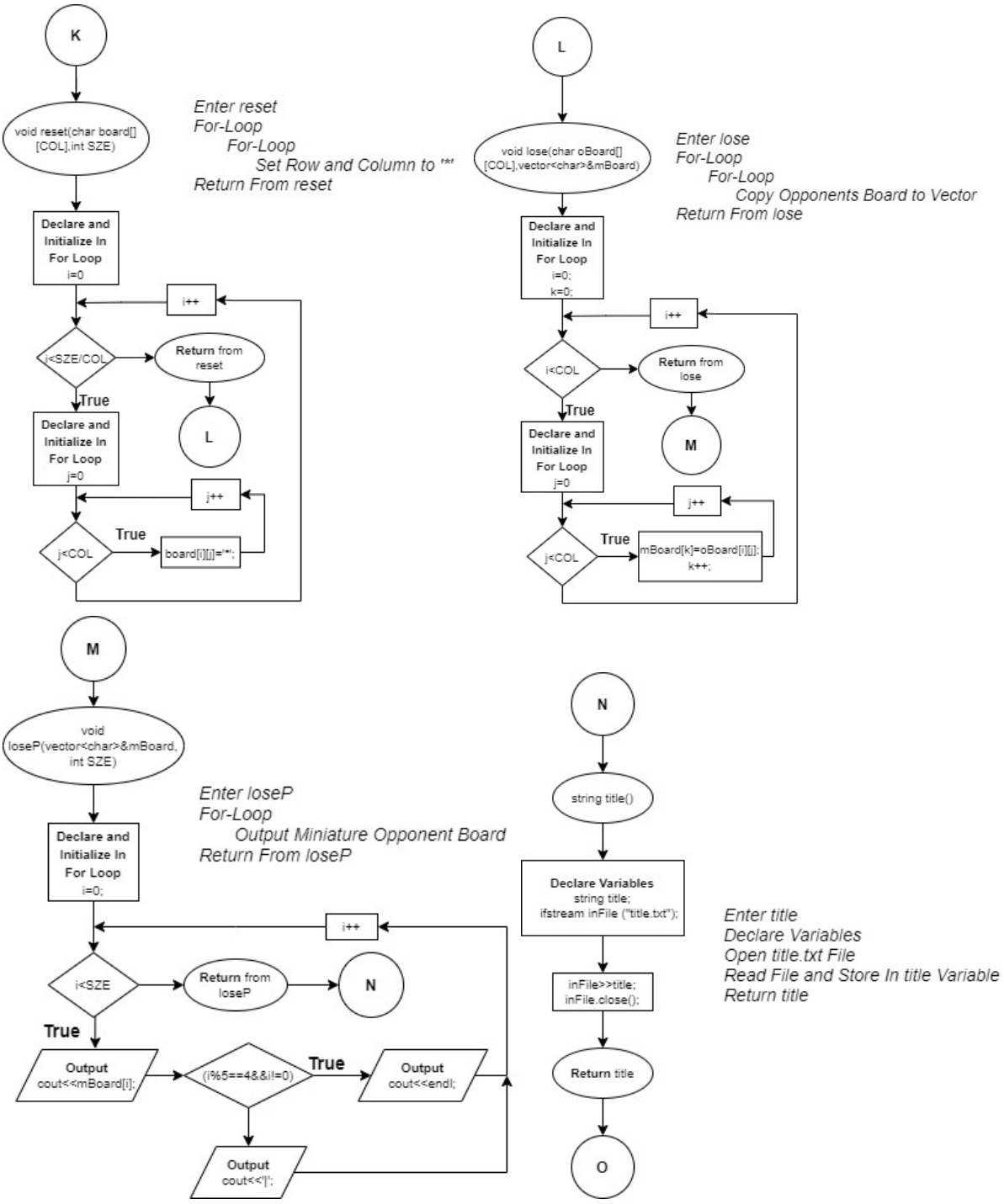
If Opponent Hit Users Ship
Output Message and Set Location to 'H'
Else Opponent Missed
Output Message and Set Location to 'X'
Return From opponent

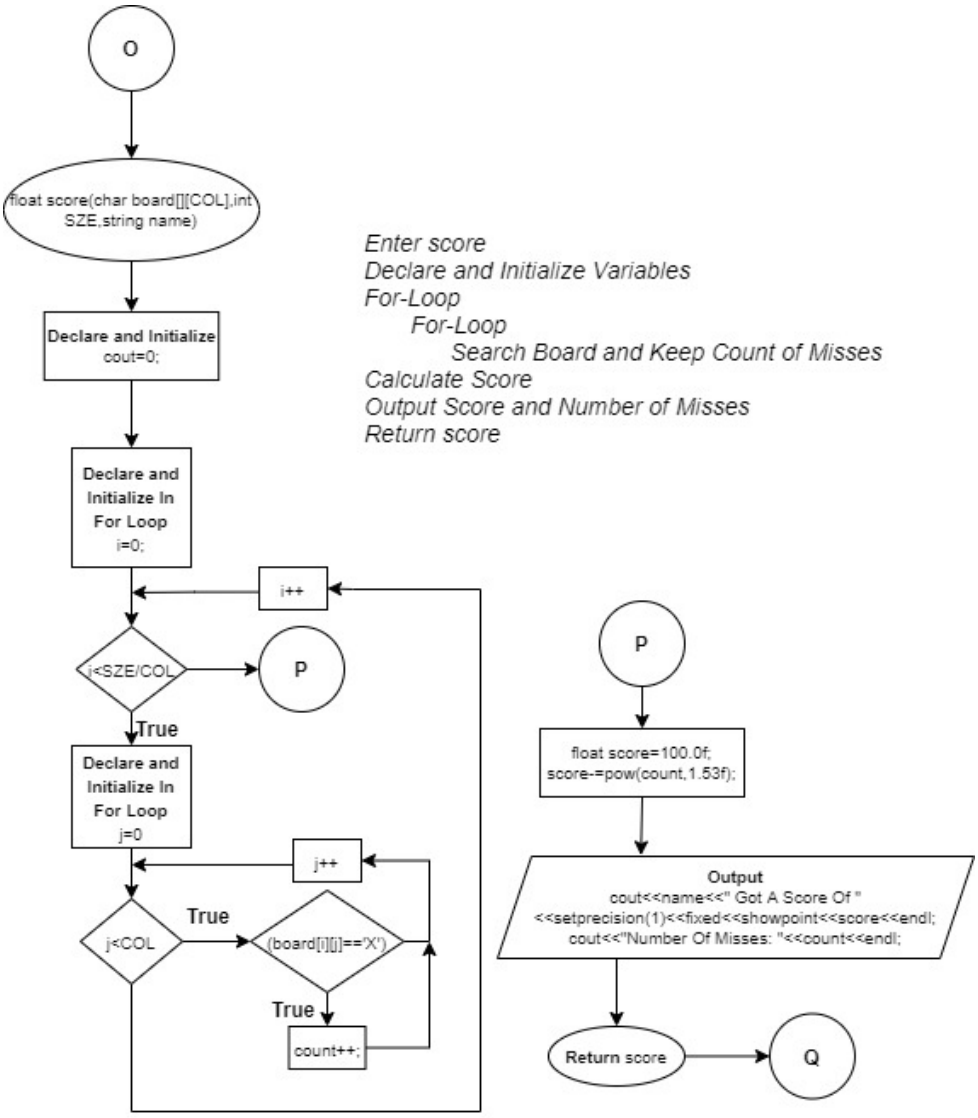


Enter displayBoard
For-Loop
For-Loop
Output Game Board Row
Go to Next Row And Repeat Until All Of the Board Is Outputed
Return From displayBoard

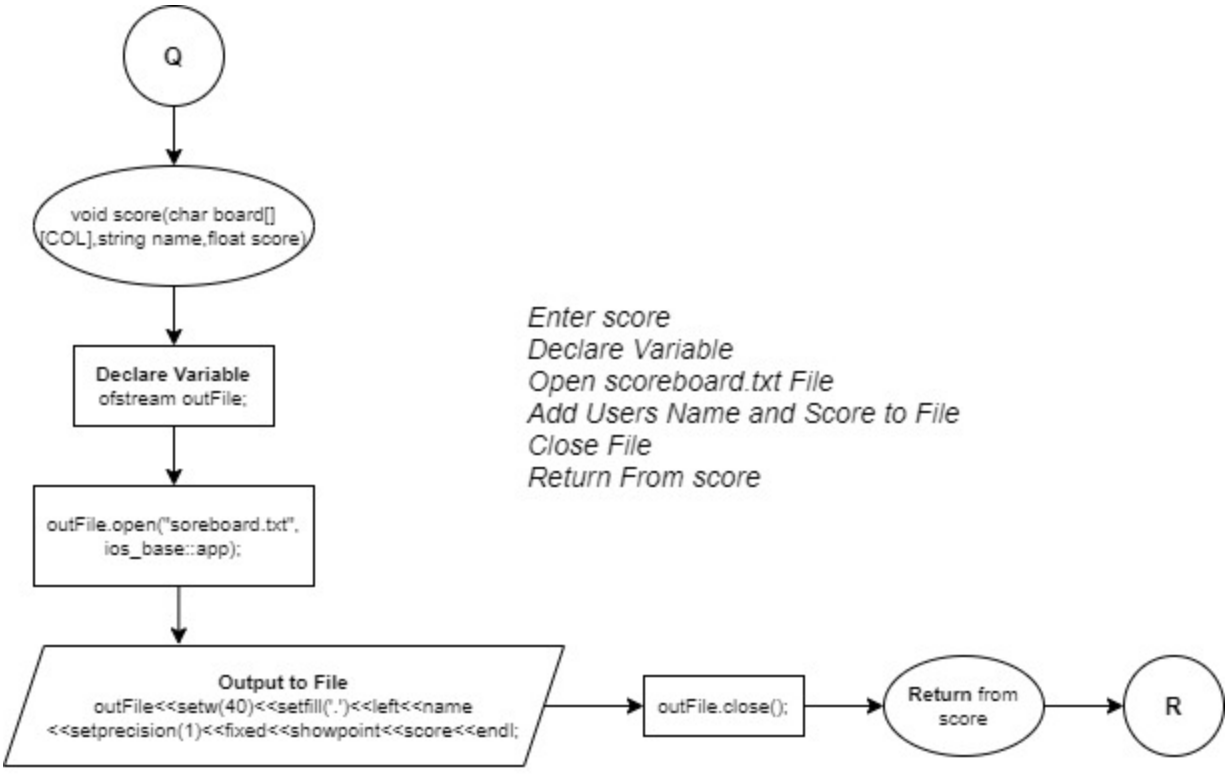


Enter generateShips
Declare and Initialize Variables
For-Loop
Do
Randomly Generate Ships Locations, Row and Column
While Ship Is Already In Location
Set Ship In Location
Return From generateShips

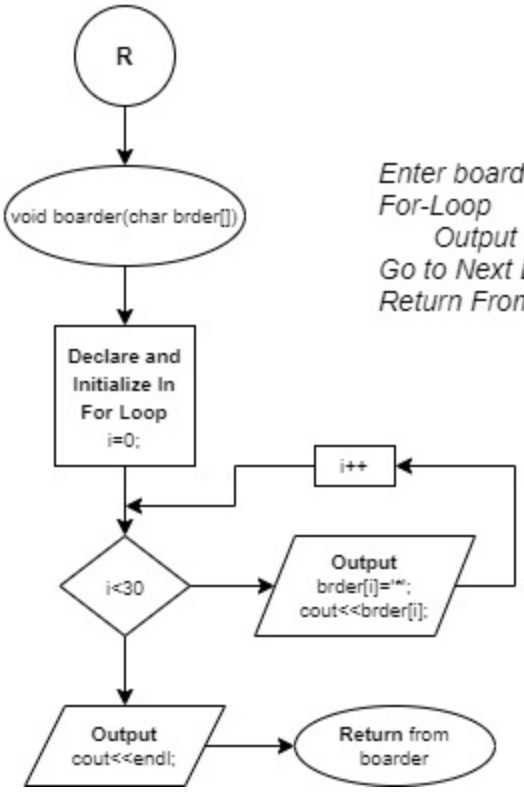

K
void reset(char board[][COL],int SZE)
Declare and Initialize In For Loop i=0
i<SZE/COL
Return from reset
L
True
Declare and Initialize In For Loop j=0
j<COL
board[i][j]='*';
i++
j++
Enter reset
For-Loop
For-Loop
Set Row and Column to '*'
Return From reset
L
void lose(char oBoard[][COL],vector<char>&mBoard)
Declare and Initialize In For Loop i=0; k=0;
i<COL
Return from lose
M
mBoard[k]=oBoard[i][j];
k++;
Enter lose
For-Loop
For-Loop
Copy Opponents Board to Vector
Return From lose
M
void loseP(vector<char>&mBoard, int SZE)
Declare and Initialize In For Loop i=0;
i<SZE
Return from loseP
N
Output cout<<mBoard[i];
(i%5==4&&i!=0)
Output cout<<endl;
Output cout<<'|';
Enter loseP
For-Loop
Output Miniature Opponent Board
Return From loseP
N
string title()
Declare Variables string title; ifstream inFile ("title.txt");
inFile>>title; inFile.close();
Return title
O
Enter title
Declare Variables
Open title.txt File
Read File and Store In title Variable
Return title



Enter score
Declare and Initialize Variables
For-Loop
For-Loop
Search Board and Keep Count of Misses
Calculate Score
Output Score and Number of Misses
Return score



Enter score
Declare Variable
Open scoreboard.txt File
Add Users Name and Score to File
Close File
Return From score



Enter boarder
For-Loop
Output 30 '' Characters*
Go to Next Line
Return From boarder

Program

```
/*
 * File:   main.cpp
 * Author: Christian Daniel
 * Created on February 9th, 2021, 5:00 PM
 * Purpose:  Battleship
 */

//System Libraries
#include <iostream>  //I/O Library
#include <cstdlib>  //Random Function
#include <ctime>  //Time Library
#include <fstream>  //File I/O
#include <iomanip>  //Formatting Library
#include <string>  //String Library
#include <cmath>  //Math Library
#include <vector>  //Vector Library
using namespace std;

//User Libraries

//Global Constants
//Math, Science, Universal, Conversions, High Dimensioned Arrays
const int COL=5;  //Number of Columns

//Function Prototypes
void displayBoard(char[][COL],int=25);  //Display Game Board
void generateShips(char[][COL]);  //Randomly Generate Battleship Locations
void getUserInp(char[][COL],char[][COL]);  //Get User Input
void opponent(char[][COL]);  //Opponent Attacks
bool game(char[][COL],char[][COL],int&);  //Checks If Game Over And Returns Winner
void reset(char[][COL],int);  //Resets Game Board
string title();  //Name Of Game
float score(char[][COL],int,string);  //Get Score
void score(char[][COL],string,float);  //Add to Score Board
void lose(char[][COL],vector<char>&);  //Copy Opponent Board
void loseP(vector<char>&,int);  //Output Mini Opponent Board
void boarder(char []);  //Output Boarder

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

```
//Declare And Initialize Variables
const int SZE=25;  //Size of Array
char oBoard[SZE][COL]={'*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*'};  //Initialize Opponents Hidden Board
char uBoard[SZE][COL]={'*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*'};  //Initialize Users Board
char dBoard[SZE][COL]={'*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*',
                       '*','*','*','*','*'};  //Initialize Opponents Displayed Board
vector<char> mBoard(SZE);
char brder [30];
bool gameOvr;
int winner;
string name;
int choice;

//Generate Ships
generateShips(oBoard);  //Generate Opponent Ships Randomly
generateShips(uBoard);  //Generate Users Ships Randomly

//Output Title
boarder(brder);
cout<<setw(20)<<title()<<endl<<endl;
boarder(brder);

//Input Name
cout<<"Enter Name: ";
getline (cin,name);

//Choose Battleship Locations
do{
   cout<<"Your Battleship Locations"<<endl;
   displayBoard(uBoard,SZE);
   cout<<"Enter 1 To Change Battleship Locations.\n";
   cout<<"Enter 2 To Quit.\n";
   cout<<"Enter Anything Else To Proceed\n";
```

```
        cin>>choice;  //Input Choice
        switch (choice){
            case 1: reset(uBoard,SZE);  //Reset Board And Generate New Ships
                    generateShips(uBoard);break;
            case 2: exit(0);  //Exit Program
        }
    }while (choice==1);  //While Choice Is To Generate New Ship Locations

    //Game Begins
    cout<<"\n\nGet Ready For Battle "<<name<<"!\n\n";
    do{
        getUserInp(oBoard,dBoard);  //Get USer Input
        gameOvr=game(uBoard,oBoard,winner);
        if(!gameOvr){  //Check If Game Is Over
            cout<<"Opponents Board\n";
            displayBoard(dBoard);  //Output Opponents Board
            opponent(uBoard);  //Opponents Turn
            cout<<"Your Board\n";
            displayBoard(uBoard);  //Display Users Board
            gameOvr=game(uBoard,oBoard,winner);  //Check if Game Is Over
        }
    }while(!gameOvr);  //Loop While Game Is Not Over

    //Game Over
    while(gameOvr){  //While Game Is Over
        game(uBoard,oBoard,winner);  //Check Who Won
        (winner==1)?cout<<"\n:D You Win!\n":  //Output If Win
                cout<<"\nYou Lose :(\n\n";  //Output If Lose
        if(winner==1){  //If User Wins
            float scr=score(oBoard,SZE,name);  //Output Score
            score(oBoard,name,scr);  //Add Score To Scoreboard
        }
        if(winner==2){  //If Opponent Wins
            cout<<"Opponents Ships\n";
            lose(oBoard,mBoard);  //Copy Opponent Board
            loseP(mBoard,SZE);  //Output Mini Opponent Board
        }
        gameOvr=false;  //Exit Loop
    }
    return 0;
}

bool game(char uBoard[][COL],char oBoard[][COL],int &winner){
    //Declare/Initialize
```

```
    bool user=false,
         opp=false;

    //Search Boards
    //Loop Through Every Index And Look For Remaining User Ships
    for(int i=0;i<COL;i++){
       for(int j=0;j<COL;j++){
          if (uBoard[i][j]=='O'){  //If User Ships Are Left
             opp=true;
          }
       }
    }
    //Loop Through Every Index And Look For Remaining Opponent Ships
    for(int i=0;i<COL;i++){
       for(int j=0;j<COL;j++){
          if (oBoard[i][j]=='O'){  //If Opponent Ships Are Left
             user=true;
          }
       }
    }
    //Results
    if (user==false&&opp==true)winner=1;  //If User Sunk All Opponent Ships, User Wins
    else if (opp==false&&user==true)winner=2;  //If Opponent Sunk All User Ships, Opp Wins
    if (user==true&&opp==true) return false;  //If Both Players Are Still In Game, Game Is Not Over
    else return true;  //Else Game Over
}

void getUserInp(char oBoard[][COL],char dBoard[][COL]){
    //Declare
    int row,
     column;

    //Get User Input To Launch Missel
    do{
       cout<<"Launch Missel At Opponent [Row][Column]"<<endl;
          cin>>row>>column;  //Input Row and Column
          cout<<endl;
          row-=1;
          column-=1;
    }while(oBoard[row][column]=='X'||
          oBoard[row][column]=='H'||
          row<0||row>4||column<0||column>4);  //Input Validation
    //Hit or Miss
    if (oBoard[row][column]=='O'){  //If User Hits Opponents Ship
```

```
      cout<<"Fired At: ["<<row+1<<"]["<<column+1<<"]\n";
      cout<<"YOU SUNK AN OPPONENTS SHIP!\n";
      oBoard[row][column]='H';
      dBoard[row][column]='H';
   }else{ //Else User Missed
      cout<<"Fired At: ["<<row+1<<"]["<<column+1<<"]\n";
      cout<<"Oh No! You Missed!\n";
      oBoard[row][column]='X';
      dBoard[row][column]='X';
   }
}

void opponent(char uBoard[][COL]){
   //Declare
   int row,
    column;
   //Generate Opponents Missel Launch Location
   do{
      row=rand()%5;
      column=rand()%5;
   }while(uBoard[row][column]=='X'||uBoard[row][column]=='H');  //Validation
   //Hit or Miss
   if (uBoard[row][column]=='O'){  //If Opponent Hits users Ship
      cout<<"Opponent: ["<<row+1<<"]["<<column+1<<"]"<<endl;
      cout<<"Opponent Has Hit Your Ship!"<<endl;
      uBoard[row][column]='H';
   }else{  //Else Opponent Missed
      cout<<"Opponent: ["<<row+1<<"]["<<column+1<<"]"<<endl;
      cout<<"Opponent Missed!"<<endl;
      uBoard[row][column]='X';
   }
}

void displayBoard(char board[][COL],int SZE){
   //Displays Any Board Passed In
   for(int i=0; i<SZE/COL; i++){
      for (int j=0; j<COL; j++){
         cout<<board[i][j];
         if (j<COL-1){cout<<'|';
         }else cout<<'\n';
      }
   if (i<COL-1) cout<<"-+-+-+-+-\n";
   }cout<<endl;
}
```

```
void generateShips(char board[][COL]){
   //Declare/Initialize
   const int numShps=5;
   int row,
    column;
   //Generate 5 Random Ship Locations
   for(int i=0; i<numShps; i++){
      do{
         row=rand()%5;  //Generate Random Row
         column=rand()%5;  //Generate Random Column
      }while (board[row][column]=='O');  //While Ship Already In Location
      //Set Ship In Generated Location
      board[row][column]='O';
   }
}

void reset(char board[][COL],int SZE){
   //Clear Any Board Passed In
   for(int i=0; i<SZE/COL; i++){
      for (int j=0; j<COL; j++){
         board[i][j]='*';
      }
   }
}

void lose(char oBoard[][COL],vector<char>&mBoard){
   for(int i=0,k=0;i<COL;i++){  //Copy Opponent Board To Vector
      for(int j=0;j<COL;j++){
         mBoard[k]=oBoard[i][j];
         k++;
      }
   }
}

void loseP(vector<char>&mBoard,int SZE){
   for(int i=0;i<SZE;i++){  //Output Mini Opponent Board
      cout<<mBoard[i];
      if (i%5==4&&i!=0)cout<<endl;
      else cout<<'|';
   }
}

string title(){
```

```
   //Declare
   string title;
   ifstream inFile ("title.txt"); //Open File
   inFile>>title;  //Read Name Of Game
   inFile.close();  //Close File
   return title;  //Return The Name
}

float score(char board[][COL],int SZE,string name){  //Calculate Score
   int count=0;  //Misses Count
   for(int i=0; i<SZE/COL; i++){
      for (int j=0; j<COL; j++){
         if (board[i][j]=='X'){  //Count All Misses
            count++;
         }
      }
   }
   float score=100.0f;  //Top Score (0 Misses)
   score-=pow(count,1.53f);  //Misses Exponentially Deduct From Score
   cout<<name<<" Got A Score Of "  //Output Score
         <<setprecision(1)<<fixed<<showpoint<<score<<endl;
   cout<<"Number Of Misses: "<<count<<endl;  //Output Number Of Misses
   return score;  //Return The Score
}

void score(char board[][COL],string name,float score){  //Add To Scoreboard
   //Declare
   ofstream outFile;
   outFile.open("soreboard.txt", ios_base::app);  //Open File
   outFile<<setw(40)<<setfill('.')<<left<<name    //Add Name And Score To Scoreboard
         <<setprecision(1)<<fixed<<showpoint<<score<<endl;
   outFile.close();  //Close File
}

void boarder(char brder[]){
   //Output Boarder
   for(int i=0;i<30;i++){
      brder[i]='*';
      cout<<brder[i];
   }cout<<endl;
}
```