Project 1

<1-Line Battleship>

CSC-5 (41375)

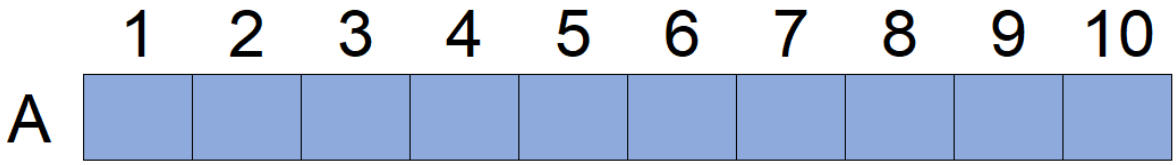
Christian Daniel

02/07/2021

Introduction

Tittle: 1-Line Battleship

This game is just like the regular Battleship game most are familiar with, except it is not played on a 10x10 grid. This game only uses the top row ‘A’ but keeps the same number of columns [1-10].



The user is first asked to input their name, and the location of their two battleships. The program will generate the opponent’s battleship locations at random each game. After validating the users input, the user will be asked to choose a location on the opponent’s grid to launch a missel at. A message will be outputted letting the user know whether or not they hit a battleship. Immediately after, the program will generate a random number, and will launch a missel at the user’s grid. A message will be outputted letting the user know where on the grid the missel was launched, and if the opponent has sunk one of their battleships.

The program will continue looping until the user wins (sinking both opponent’s ships), or until the user loses (opponent sinks both of the users ships). This program was coded so that the opponent nor user will be allowed to launch a missel at the same location more than once.

Summary

Project size: about 180 lines

Number of variables: about 35

This project includes most of the concepts covered in the first half of the semester. New concepts can also be implemented into this game, such as arrays and functions, that will make the program easier to read and to make changes, for Project 2. I plan to expand the battleship grid into 2 dimensions like it was meant to be played.

The coding took me about 20 hours. (v1) I started off with just trying to figure out how to let the user input where to launch a missel at and then output if it was a hit or miss. (v2) I then added input validation. (v3) After that, I set a random number seed so that the opponents battleship location changes after every new game. (v4) I then added some more input validation so that the user or opponent cannot shoot the same battleship more than once. (v5) I added more input validation so that the user can not launch a missel at the same spot more than once, regardless of if it was a hit or miss, and (v6) did the same for the opponent. (v7) I then cleaned up the code a bit and added a scoreboard which saves the users name and score in a text file named “score_board.txt”. Old scores will remain in the file and not be overwritten by new scores.

Major Variables

Type	Variable Name	Description
unsigned short int	fire	Location on opponents' grid where user desires to fire missel at
	cfire	Randomly generated location on users' grid that opponent fires at
	ship1	Users 1st battleship location on grid
	ship2	Users 2nd battleship location on grid
	cship1	Opponents 1st battleship location on grid
	cship2	Opponents 2nd battleship location on grid
char	tries	Keeps track of number of launches
bool	s1, s2	Users ship 1 & 2, true means it has not been hit, false means it's been hit and sunk
	cS1, cS2	Opponents ship 1 & 2, true means it has not been hit, false means it's been hit and sunk
	a1, a2, a3, a4, a5, a6, a7, a8, a9, a10	All possible battleship locations that user can launch missel at, false means the user has already launched at that location, true means they have not
	cA1, cA2, cA3, cA4, cA5, cA6, cA7, cA8, cA9, cA10	All possible battleship locations that opponent can launch missel at, false means the opponent has already launched at that location, true means they have not

Pseudo Code

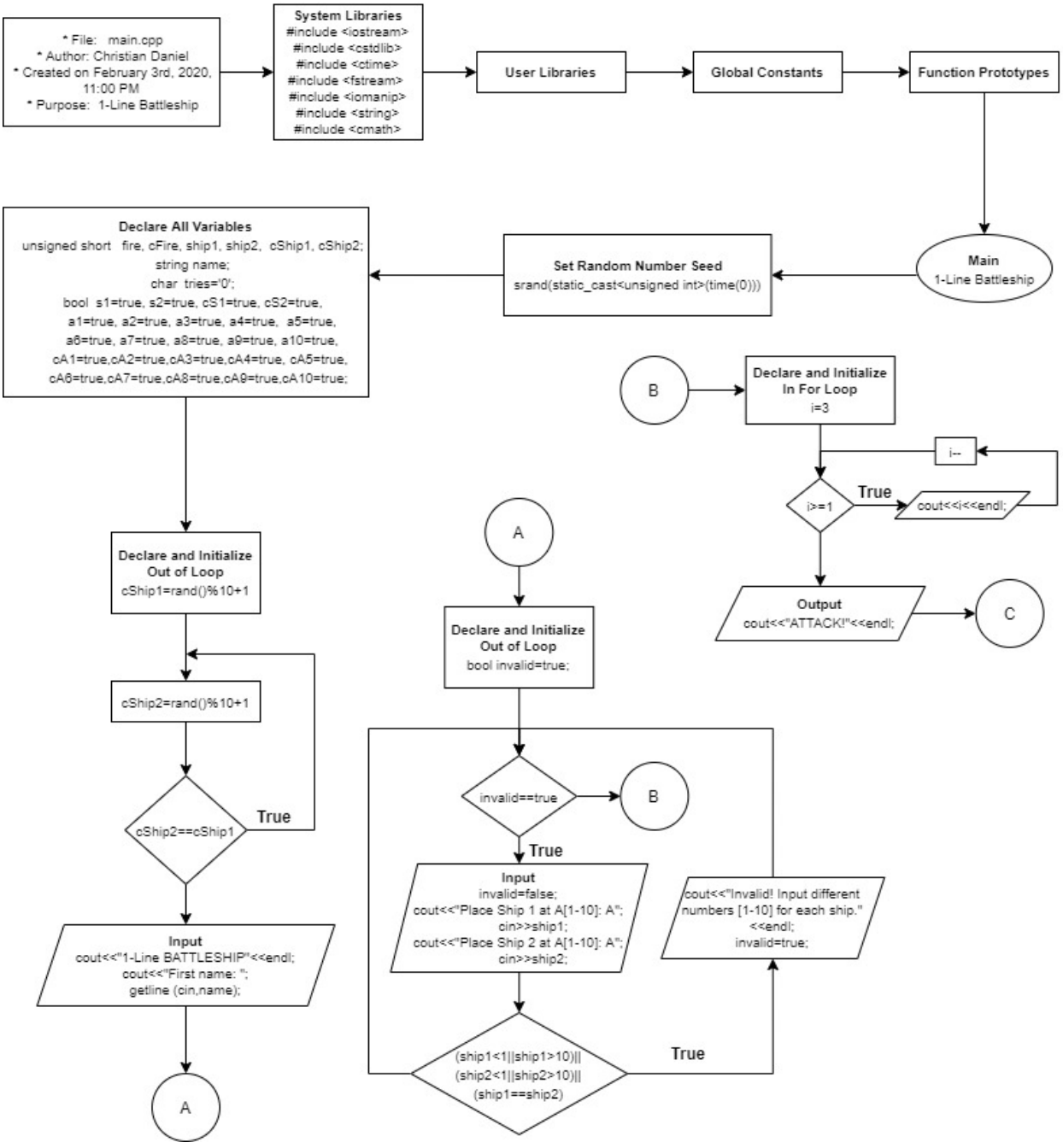
```
//Initialize Variables
Generate Random Location for Opponents Ship 1
Do
   Generate Random Location for Opponents Ship 2
While Both Ships Are in Same Location

Output Name of Game
Input Name

//Input Both Ships Location
While Inputs Are Invalid
   Input Ship 1 Location [1-10]
   Input Ship 2 Location [1-10]
   If Input Invalid
      Output Message

//Countdown To Begin Battle
Output 3 2 1 ATTACK!

Do
   Do
      Input Where to Fire at [1-10]

      If Invalid, Number Must Be 1-10, Can't Fire At Same Spot Twice
         Output Message
      Else Record Previous Inputs
         Set Location To False After Being Fired At
   While Input Is Invalid
   Add 1 to Tries Count

   //Hit or Miss
   If Ship 1 Gets Hit
      Output Message
      Set Opponent Ship 1 Location to False (sunk)
   Else If Ship 2 Gets Hit
      Output Message
      Set Opponent Ship 2 Location to False (sunk)
   Else Output Message (miss)
   Skip a Line

   //Opponent Fires
   If Game is Still Going
      Do
         Do
```

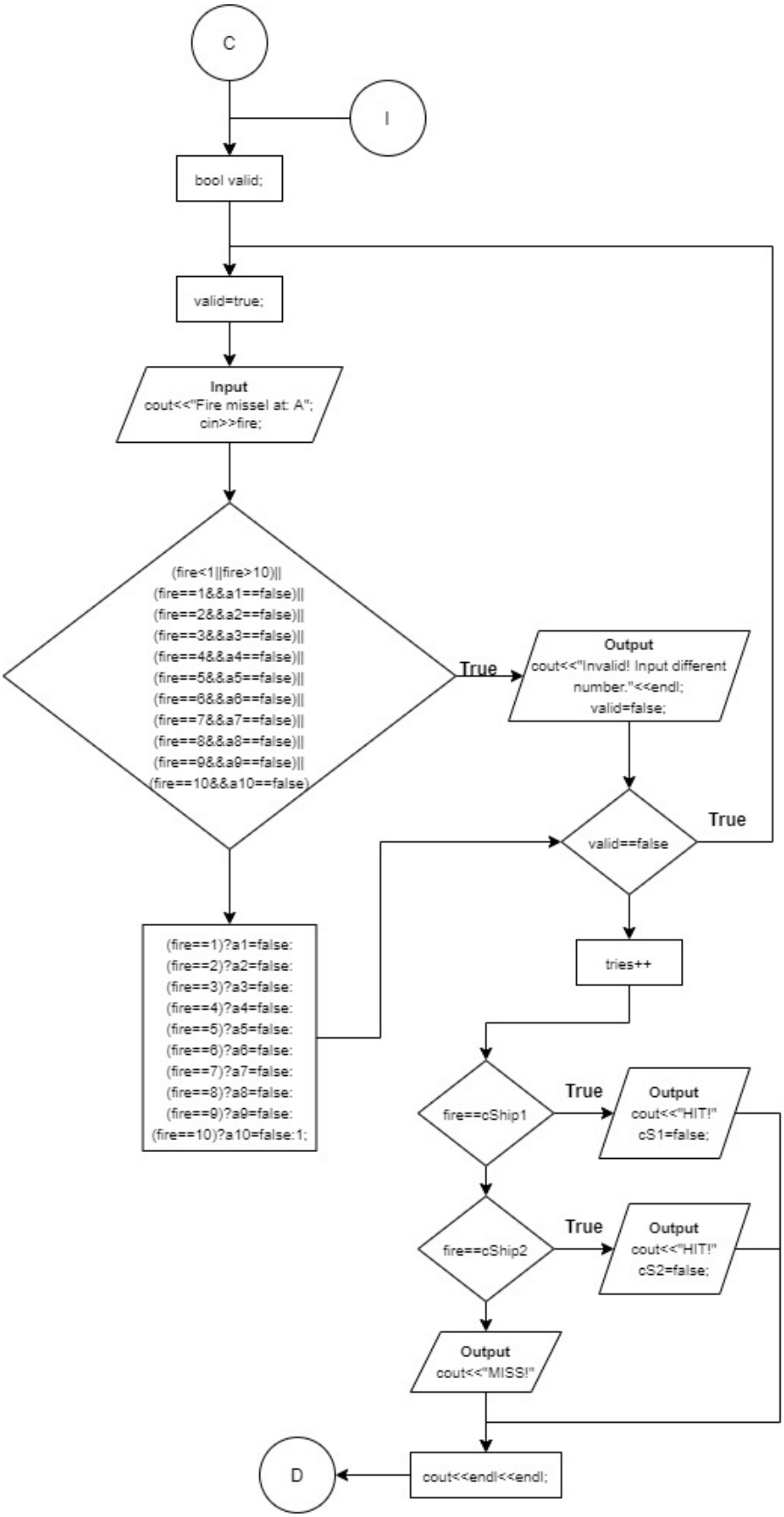
```
            Random Number Chosen for Opponent
            If Invalid, Same Number Is Generated More Than Once It's Invalid
                Set to Invalid
            If Valid, Record Location Opponent Fired At
                Set Location To False After Being Fired At
        While Invalid

        //Opponents Fire Hit or Miss
        If Ship 1 Got Hit
            Output Message
            Set Ship 1 Location to False (sunk)
        Else If Ship 2 Got Hit
            Output Message
            Set Ship 2 Location to False (sunk)
        Else If No Ship Was Hit
            Output Message
        Else Set to Invalid
    While Invalid
    Skip 2 Lines
While Game Is Still Going (Starts Over)

//Output Results
If All Opponent Ships Are Sunk
   Output Message And Score (win)
   Open txt File
   Write Name and Score Into txt File
   Close txt File
Else Output Message (lose)

//Exit stage right!
```

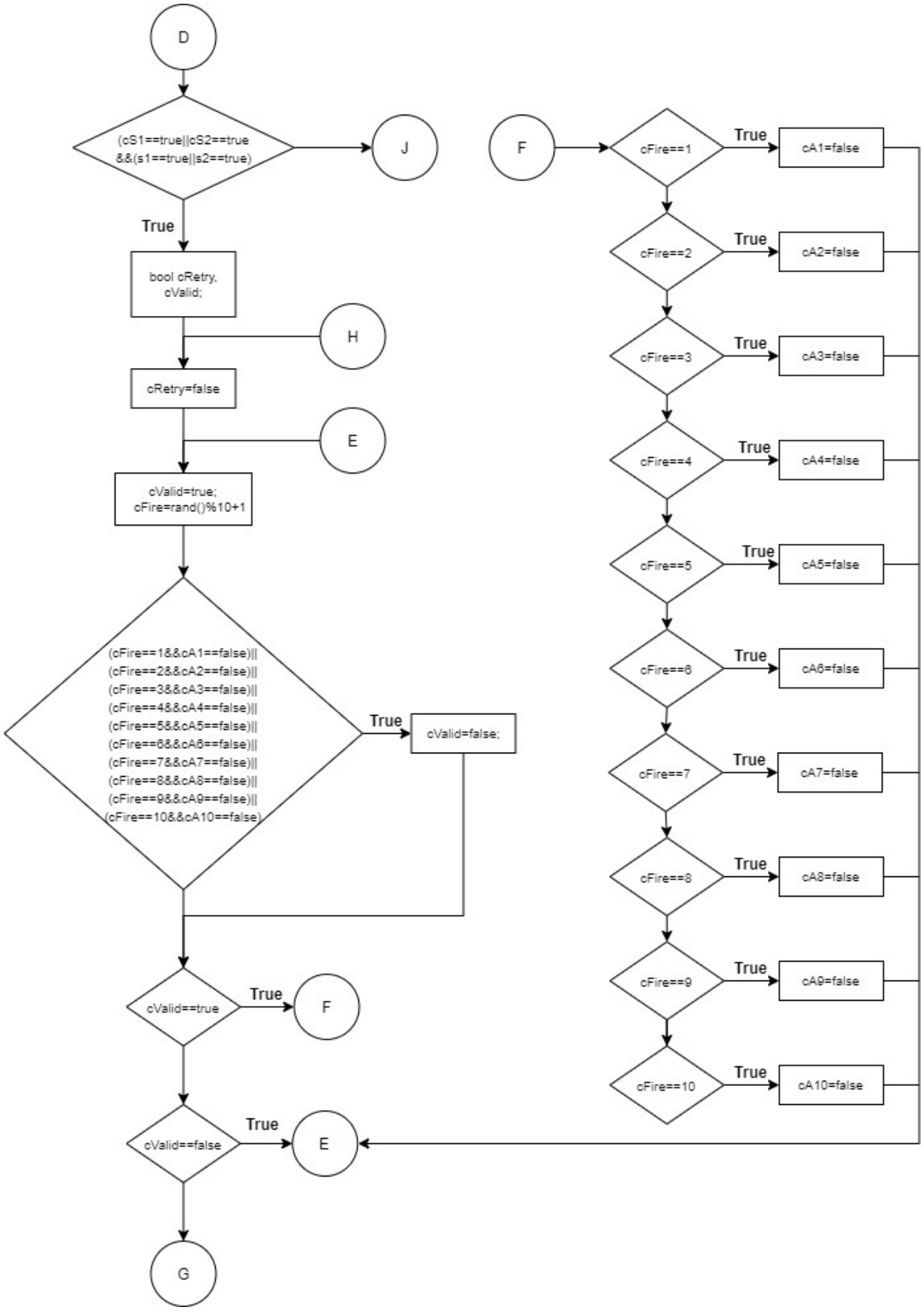
Project 1

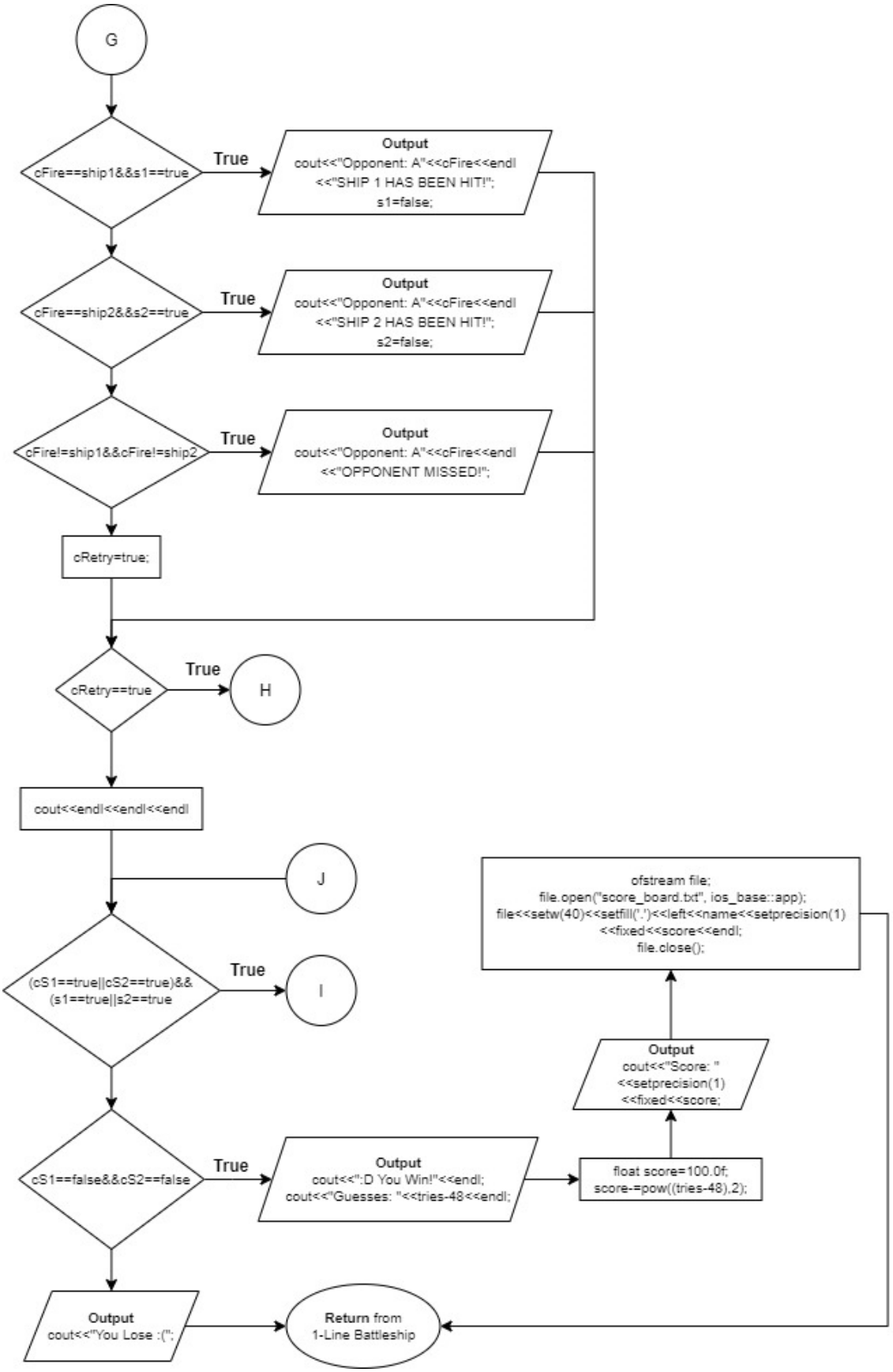
1-Line Battleship

Christian Daniel

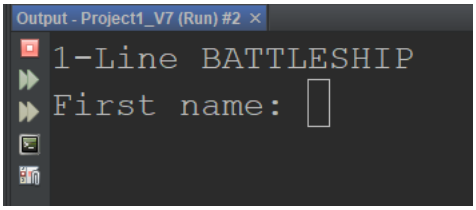


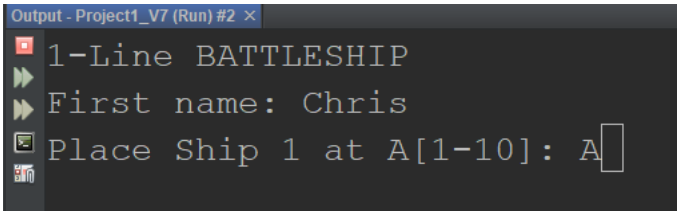

C
I
bool valid;
valid=true;
Input
cout<<"Fire missel at: A";
cin>>fire;
(fire<1||fire>10)||
(fire==1&&a1==false)||
(fire==2&&a2==false)||
(fire==3&&a3==false)||
(fire==4&&a4==false)||
(fire==5&&a5==false)||
(fire==6&&a6==false)||
(fire==7&&a7==false)||
(fire==8&&a8==false)||
(fire==9&&a9==false)||
(fire==10&&a10==false)
True
Output
cout<<"Invalid! Input different number."<<endl;
valid=false;
valid==false
True
(fire==1)?a1=false:
(fire==2)?a2=false:
(fire==3)?a3=false:
(fire==4)?a4=false:
(fire==5)?a5=false:
(fire==6)?a6=false:
(fire==7)?a7=false:
(fire==8)?a8=false:
(fire==9)?a9=false:
(fire==10)?a10=false:1;
tries++
fire==cShip1
True
Output
cout<<"HIT!"
cS1=false;
fire==cShip2
True
Output
cout<<"HIT!"
cS2=false;
Output
cout<<"MISS!"
cout<<endl<<endl;
D

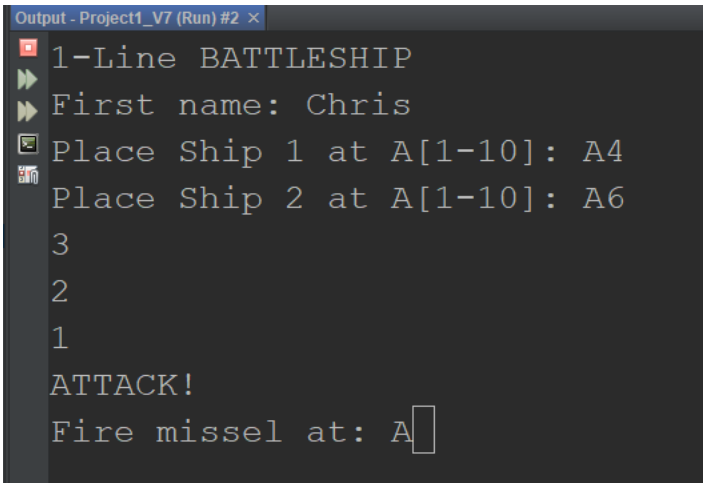

D
(cS1==true||cS2==true
&&(s1==true||s2==true)
J
True
bool cRetry,
cValid;
H
cRetry=false
E
cValid=true;
cFire=rand()%10+1
(cFire==1&&cA1==false)||
(cFire==2&&cA2==false)||
(cFire==3&&cA3==false)||
(cFire==4&&cA4==false)||
(cFire==5&&cA5==false)||
(cFire==6&&cA6==false)||
(cFire==7&&cA7==false)||
(cFire==8&&cA8==false)||
(cFire==9&&cA9==false)||
(cFire==10&&cA10==false)
True
cValid=false;
cValid==true
True
F
cValid==false
True
E
G
F
cFire==1
True
cA1=false
cFire==2
True
cA2=false
cFire==3
True
cA3=false
cFire==4
True
cA4=false
cFire==5
True
cA5=false
cFire==6
True
cA6=false
cFire==7
True
cA7=false
cFire==8
True
cA8=false
cFire==9
True
cA9=false
cFire==10
True
cA10=false

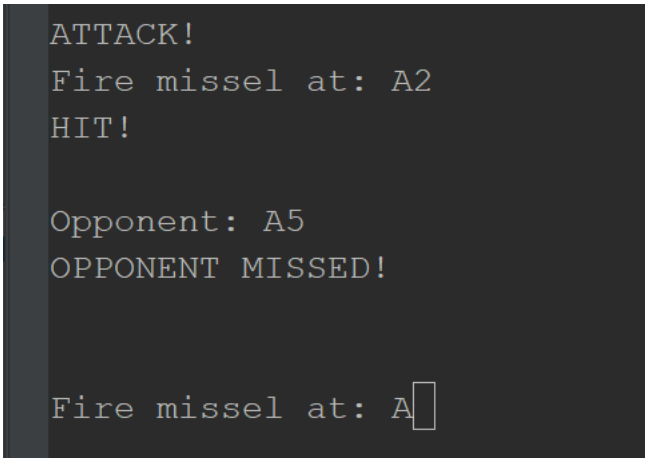

G
cFire==ship1&&s1==true
True
Output
cout<<"Opponent: A"<<cFire<<endl
<<"SHIP 1 HAS BEEN HIT!";
s1=false;
cFire==ship2&&s2==true
True
Output
cout<<"Opponent: A"<<cFire<<endl
<<"SHIP 2 HAS BEEN HIT!";
s2=false;
cFire!=ship1&&cFire!=ship2
True
Output
cout<<"Opponent: A"<<cFire<<endl
<<"OPPONENT MISSED!";
cRetry=true;
cRetry==true
True
H
cout<<endl<<endl<<endl
J
(cS1==true||cS2==true)&&
(s1==true||s2==true
True
I
cS1==false&&cS2==false
True
Output
cout<<":D You Win!"<<endl;
cout<<"Guesses: "<<tries-48<<endl;
float score=100.0f;
score-=pow((tries-48),2);
Output
cout<<"Score: "
<<setprecision(1)
<<fixed<<score;
ofstream file;
file.open("score_board.txt", ios_base::app);
file<<setw(40)<<setfill('.')<<left<<name<<setprecision(1)
<<fixed<<score<<endl;
file.close();
Output
cout<<"You Lose :(";
Return from
1-Line Battleship

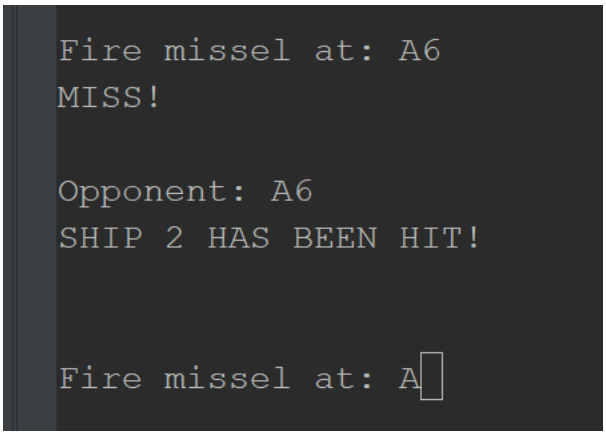
Example Inputs/Outputs



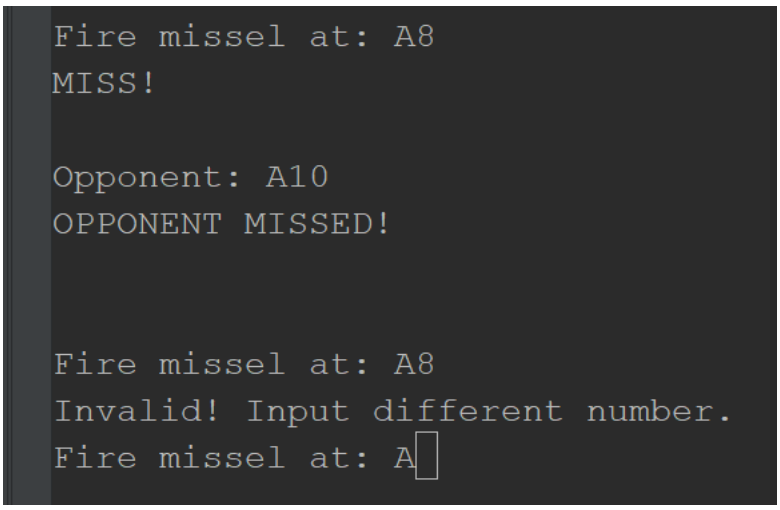




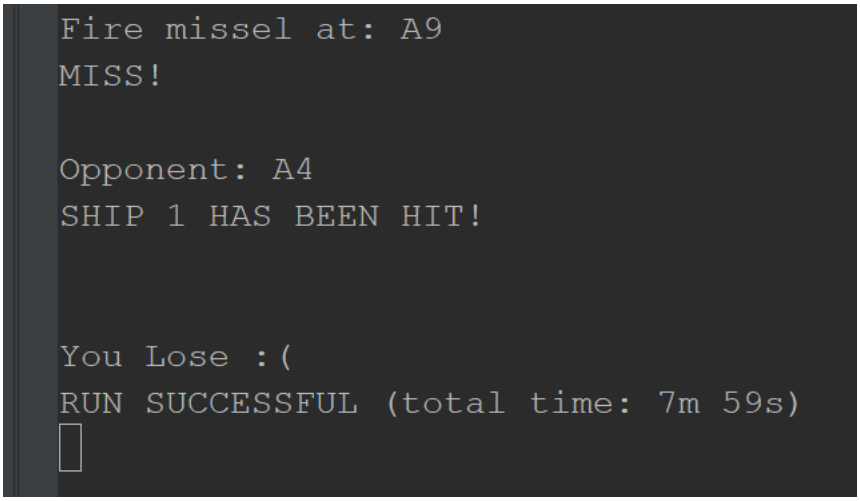




Inputting same number more than once outputs Invalid message



Lose example



Win example

```
Fire missel at: A3
HIT!

:D You Win!
Guesses: 3
Score: 91.0
RUN SUCCESSFUL (total time: 15s)
```

Score board

Score Board

Name	Score
Christian	64.0
Mark	91.0
Mike	19.0
John	84.0
Daniel	19.0
Chris	64.0

Program

```
/*
 * File:   main.cpp
 * Author: Christian Daniel
 * Created on February 3rd, 2020, 11:00 PM
 * Purpose:  1-Line Battleship
 */

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

using namespace std;

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

   //Declare Variable Data Types and Constants
   unsigned short   fire,  //Location To Fire At
                 cFire,  //Location Opponent Fires At
                 ship1,  //Ship 1 Location
                 ship2,  //Ship 2 Location
                 cShip1, //Opponent Ship 1
                 cShip2; //Opponent Ship 2
   string name;
   char  tries='0';  //Number of Attempts
   bool  s1=true, //Ship 1 floating or sunk
         s2=true, //Ship 2 floating or sunk
        cS1=true, //Opponent Ship 1 floating or sunk
        cS2=true, //Opponent Ship 2 floating or sunk
        a1=true, a2=true, a3=true, a4=true,  a5=true,  //All Possible Battleship Locations
        a6=true, a7=true, a8=true, a9=true, a10=true,  //True Means Location Has Not Been Fired
At
       cA1=true,cA2=true,cA3=true,cA4=true, cA5=true,  //All Possible Opponent Battleship
Locations
       cA6=true,cA7=true,cA8=true,cA9=true,cA10=true;  //True Means Location Has Not Been
Fired At

   //Initialize Variables
```

```
cShip1=rand()%10+1;  //Generate Random Location for Opponents Ship 1
do{  //Do
   cShip2=rand()%10+1;  //Generate Random Location for Opponents Ship 2
}while (cShip2==cShip1);  //While Both Ships Are in Same Location

cout<<"1-Line BATTLESHIP"<<endl;  //Output Name of Game
cout<<"First name: ";  //Input Name
getline (cin,name);

//Input Both Ships Location
bool invalid=true;
while (invalid==true){   //While Inputs Are Invalid
   invalid=false;
   cout<<"Place Ship 1 at A[1-10]: A";
   cin>>ship1;  //Input Ship 1 Location [1-10]
   cout<<"Place Ship 2 at A[1-10]: A";
   cin>>ship2;  //Input Ship 2 Location [1-10]
   if ((ship1<1||ship1>10)||(ship2<1||ship2>10)||(ship1==ship2)){  //If Input Invalid
      cout<<"Invalid! Input different numbers [1-10] for each ship."<<endl;  //Output Message
      invalid=true;
   }
}

//Countdown To Begin Battle
for(int i=3;i>=1;i--){  //3 2 1 ATTACK!
   cout<<i<<endl;
}
cout<<"ATTACK!"<<endl;

do{  //Do
   bool valid;
   do{  //Do
      valid=true;
      cout<<"Fire missel at: A";
      cin>>fire;  //Input Where to Fire at [1-10]

      if ((fire<1||fire>10)||  //If Invalid, Number Must Be 1-10
         (fire==1&&a1==false)||  //Can't Fire At Same Spot Twice
         (fire==2&&a2==false)||
         (fire==3&&a3==false)||
         (fire==4&&a4==false)||
         (fire==5&&a5==false)||
         (fire==6&&a6==false)||
         (fire==7&&a7==false)||
         (fire==8&&a8==false)||
         (fire==9&&a9==false)||
```

```
        (fire==10&&a10==false)){

        cout<<"Invalid! Input different number."<<endl; //Output Message
        valid=false;

    }else{  //Else Record Previous Inputs
        (fire==1)?a1=false:  //Set Location To False After Being Fired At
        (fire==2)?a2=false:
        (fire==3)?a3=false:
        (fire==4)?a4=false:
        (fire==5)?a5=false:
        (fire==6)?a6=false:
        (fire==7)?a7=false:
        (fire==8)?a8=false:
        (fire==9)?a9=false:
        (fire==10)?a10=false:1;
    }

}while (valid==false);  //While Input Is Invalid
 tries++;  //Add 1 to Tries Count

 //Hit or Miss
 if (fire==cShip1){  //If Ship 1 Gets Hit
    cout<<"HIT!";  //Output Message
    cS1=false;  //Set cS1 to false (sunk)
 }else if (fire==cShip2){  //If Ship 2 Gets Hit
    cout<<"HIT!";  //Output Message
    cS2=false;  //Set cS2 to false (sunk)
 }else cout<<"MISS!";  //Else Output Message
 cout<<endl<<endl;  //Skip a Line

 //Opponent Fires
 if ((cS1==true||cS2==true)&&
     (s1==true||s2==true)){  //If Game is Still Going
    bool cRetry,
         cValid;
    do{  //Do
        cRetry=false;
        do{  //Do
            cValid=true;
            cFire=rand()%10+1;  //Random Number Chosen for Opponent
            if ((cFire==1&&cA1==false)||  //If Invalid
                (cFire==2&&cA2==false)||
                (cFire==3&&cA3==false)||
                (cFire==4&&cA4==false)||
                (cFire==5&&cA5==false)||
```

```
                (cFire==6&&cA6==false)||
                (cFire==7&&cA7==false)||
                (cFire==8&&cA8==false)||
                (cFire==9&&cA9==false)||
                (cFire==10&&cA10==false)){
                cValid=false;  //If Same Number Is Generated More Than Once It's Invalid
            }
            if (cValid==true){  //If Valid, Record Location Opponent Fired At
               switch(cFire){
                  case 1:cA1=false;break;  //Set Location To False After Being Fired At
                  case 2:cA2=false;break;
                  case 3:cA3=false;break;
                  case 4:cA4=false;break;
                  case 5:cA5=false;break;
                  case 6:cA6=false;break;
                  case 7:cA7=false;break;
                  case 8:cA8=false;break;
                  case 9:cA9=false;break;
                  case 10:cA10=false;break;
               }
            }
         }while(cValid==false);  //While Invalid

         //Opponents Fire Hit or Miss
         if (cFire==ship1&&s1==true){  //If Ship 1 Got Hit
            cout<<"Opponent: A"<<cFire<<endl
               <<"SHIP 1 HAS BEEN HIT!";  //Output Message
            s1=false;  //Set s1 to false (sunk)
         }else if (cFire==ship2&&s2==true){  //Else If Ship 2 Got Hit
            cout<<"Opponent: A"<<cFire<<endl
               <<"SHIP 2 HAS BEEN HIT!";  //Output Message
            s2=false;  //Set s2 to false (sunk)
         }else if (cFire!=ship1&&cFire!=ship2){  //Else If No Ship Was Hit
            cout<<"Opponent: A"<<cFire<<endl
               <<"OPPONENT MISSED!";  //Output Message
         }else cRetry=true;
      }while (cRetry==true);  //While Invalid
      cout<<endl<<endl<<endl;  //Skip 2 Lines
   }
}while ((cS1==true||cS2==true)&&(s1==true||s2==true));  //While Game Is Still Going

//Output Results
if (cS1==false&&cS2==false){  //If All Opponent Ships Are Sunk
   cout<<":D You Win!"<<endl;  //Output Message And Score
   cout<<"Guesses: "<<tries-48<<endl;
   float score=100.0f;
```

```
        score-=pow((tries-48),2);
        cout<<"Score: "<<setprecision(1)<<fixed<<score;
        ofstream file;
        file.open("score_board.txt", ios_base::app);  //Open txt File
        file<<setw(40)<<setfill('.')<<left<<name<<setprecision(1)<<fixed<<score<<endl;  //Write
Name and Score Into txt File
        file.close();  //Close txt File
    }else cout<<"You Lose :(";  //Else Output Message

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