Mastermind.

Project2

CSC- 17a – 48983 C++

Jonathan Balisky

25 – October – 2015

Content

1. Introduction

Rules and Gameplay

This version of mastermind allows for two modes of gameplay easy or normal. In the easy mode the player has to guess a 3 digit pin, in normal a 4 digit pin. Regardless of which mode is selected the program will make each digit, of the pin, some number between 0-9. Each time after the player guesses, the program will tell the player how many digits are correct and are in the right place, and how many digits are the right digits, but not in the right place. The program will then display how many guess the player has left. If the player fails to guess the right pin in 9 tries, the program will display all the guesses that the player submitted, then sort and display them with and numerical sort and ask the player if they would like to try again.

Thoughts after Program

Next version of this game should start to rewrite the program in a modular fashion. Roping in related variables into constructs with constructors and de-structers . Then tying into that functions rather then in the program itself. I believe this will greatly “clean” up the code and make it easier for future editors to understand. Finally I could place a log in a construct which would contain the players name, age, amount of games played, and the average of how many number of guesses it took to guess the answer. The log would then be updated each and every time the player plays and the user can then check his ranking against other players before or after he/she plays.

2. Development

Approach Strategy

In updating Mastermind my main purpose was to implement concepts I had recently learned in class. The goal was to take a already working piece of code and implement more advance functions and tools to remove clutter and reduce the lines of code

Because most of the logic was already there from the previous version game I went through the code again looking for places to implement several changes. These items included more the use of dat files, advance read write functions, character array, structures, and pointers to structures. One of the main improvements was the use of classes. This brings the program into a more segmented approach. Doing this minimizes the programs variability and even makings it easier to read and understand. File that where once available to be change by the user are now hidden away in protected sectors. All functions and applications are tucked in header files and .cpp files. Final arrays have been switched to dynamic arrays and the answer array has been made into a template. This allows the answer array to be changed to the appropriate data type when needed.

3. Research

I. Classes
Created a game class in which none of the variables used by that class are accessible from main. This created a more secure streamline program

II. Template file
Instead of declaring the array type beforehand I used a template then declared the answers as a char later when that was needed.

III. Batch writing to Binary files
Wanted a way to shorten up the lines of code needed for writing data to a file. By changing the format of my files from .txt to .dat and writing in binary format I was able to write a whole array of answers from the user in one line. No for loop needed Nice!

4. Variables list

Type	Variable Name	Description		Line
int	Level	How many digits the pin is		33
	counter	How many times the player guessed		23
	*answer	Array for the answer		35
	Indx	Location for the table		20
char	** table	Dynamic array for the players guesses		43
string	Temp	Temporary place holder for the file		36
	usrG	For the player to input guess		39
bool	*aMatch	check whether user digits matches answer		34
	*gMatch	Check if the guess has been matched		
	Swap	Check to see if swap was made		21
const int	SIZE	Size of arrays		27
ofstream	Output			37

5. Topic Covered (Checklist)

Chapter	Type	Code	File	Line
Constructor		Game (int level);	Game.h	
Destructor		~Game();	Game.h	
Memory Allocation	T	answer = new T [level];	Game.h	157
	Char	sAnswer = new char [level];	Game.h	158

	Char	table = new char *[guess]; //Creating 2 d dynamic array
		for (int i = 0; i < guess; i++) { //
		table[i] = new char[level];
			Game.h	179-183

	Bool	gMatch = new bool [level];	Game.h	160
Template		template <class T>	Game.h	17
				33
Base class		Class IsValid{	IsValid.h	16
Derived class in heritance		class Game :public IsValid{	Game.h	18

Delete 2d Dynamic array		for (int i = 0; i<9; i++){
		delete table[i];
		}
		delete[] table;
			Game.h	205-209

Delete array		delete[] sAnswer;	Game.h	201
Class pointer		T *answer;	Game.h	20
Deletion of class pointer		Delete[] answer;	Game.h	200
array	int array	Int answer[SIZE]		34
	bool aray	Bool mathc[SIZE]		35
	char**	table = new char *[guess]; ...		195
String Objects	string	string temp;		44
Conversion	toupper	toupper(usrG[0]) == 'Y')		157
Char array	Char array[]	char sAnswer[SIZE];		42
Dot operator	nmbr.xs	cout << "X(s)=" << nmbr.xs << endl;		130

Structure initialization	Struct Numbers	struct Numbers{
		int xs; //How many x's
		short os; //How many o'x
		};
				21-24

Structures in Arguments	nPtr->xs == level	if (nPtr->xs == level) {		137
Binary files	"Answer.dat"	inOut.open("Answer.dat", ios::out | ios::binary | ios::app);		103
12 Writing to binary	inOut.write	inOut.write(sAnswer, sizeof (sAnswer));		107
12 Batch Writing to binary with array	inOut.write	inOut.write(sAnswer, sizeof (sAnswer));		107

12 Reading binary files	"instructions.dat", ios::in	inOut.open("instructions.dat", ios::in);		61

6. Libraries included

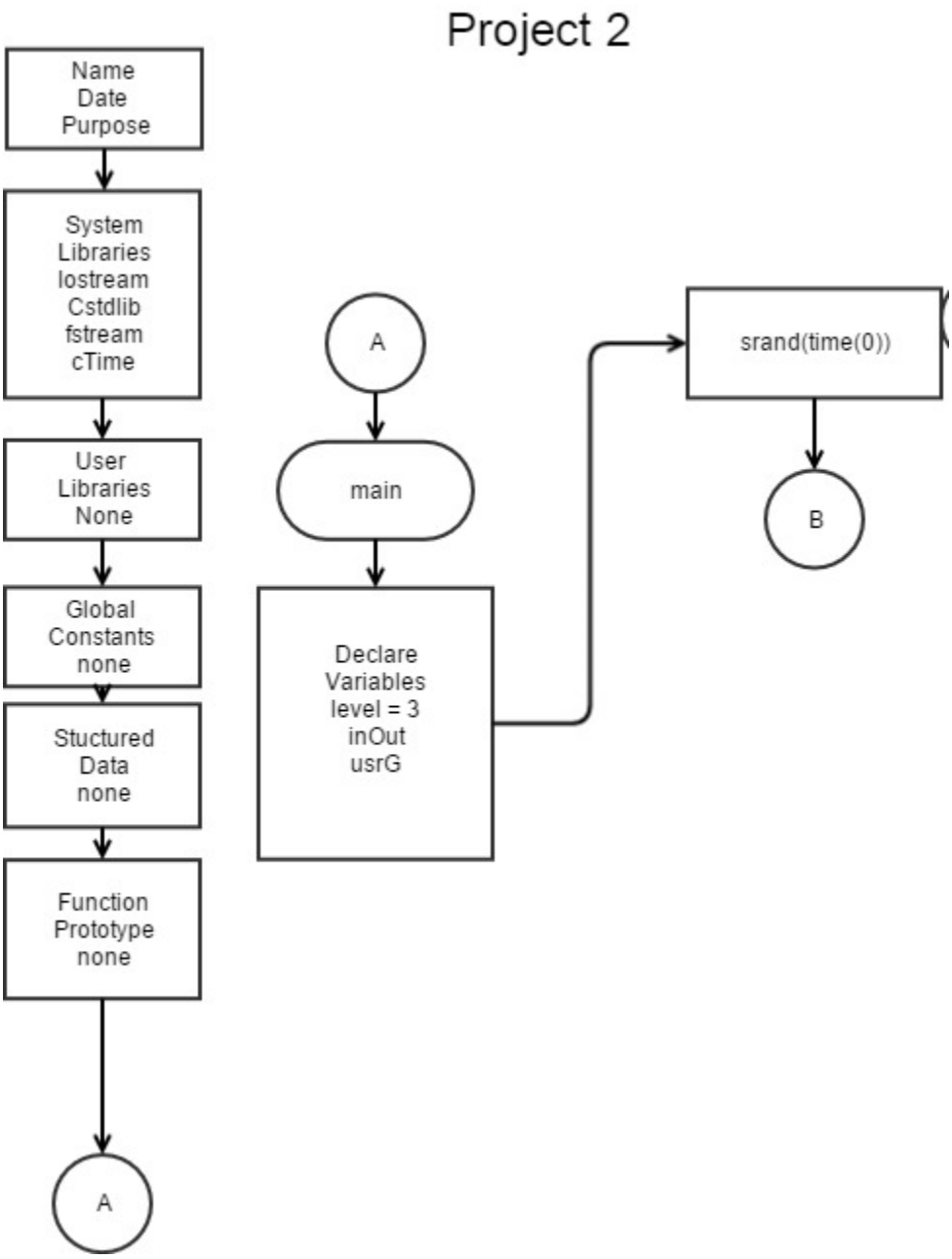
- <cstdlib>
- <cstdlib>
- <iostream>
- <ctime>
- <fstream>
- <string>
- <cstring>
- “Game”
- “IsValid”

7. Pseudo code

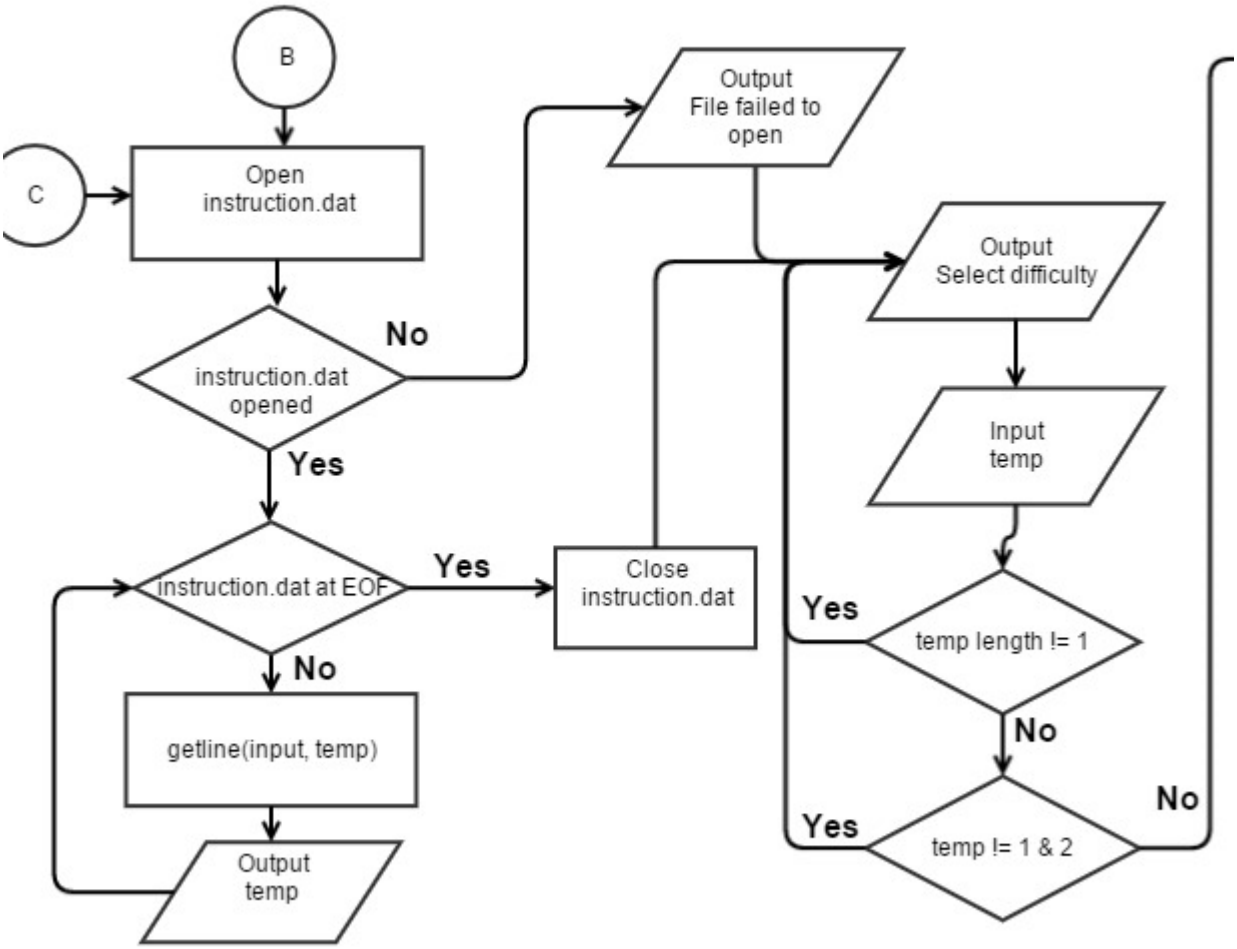
Set time seed
Output instruction from file
Do
    Ask user for easy or normal
Do
    Call prepare function
        Random answer and initialize other variables
        Create table
    Output the answer to file
    Game start
    do
        Input guess number
        Call isvalid to check validation
        Call gssHst
        Call compare function
        Display Xs and Ox (result)
        Guess -1
    While guess>0 and guess answer is not correct
    Sort answers
    Delete table
    If x==level output win
    Else output lose
    Ask for another new game
While(Yes)

7. Flowchart

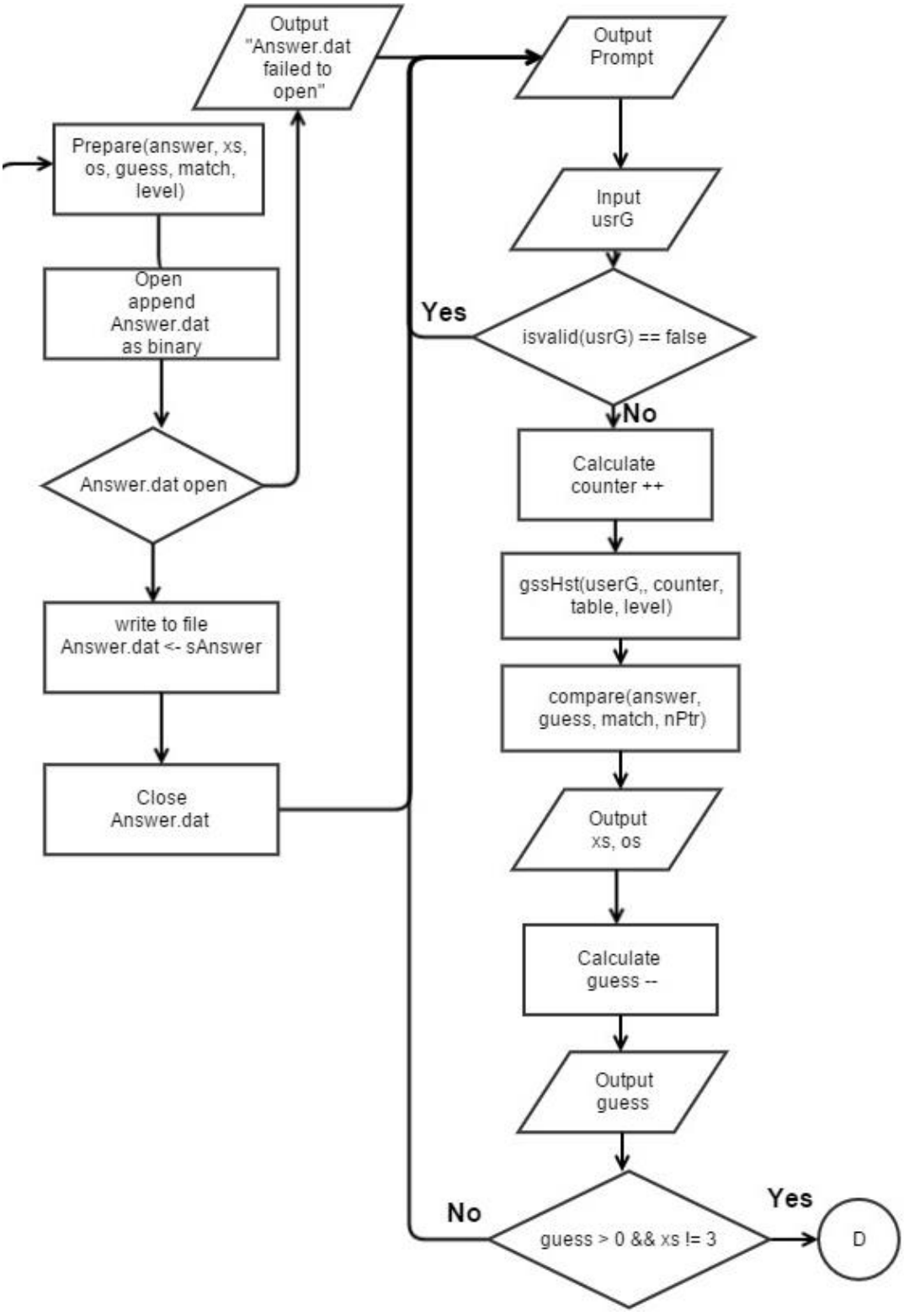
Main

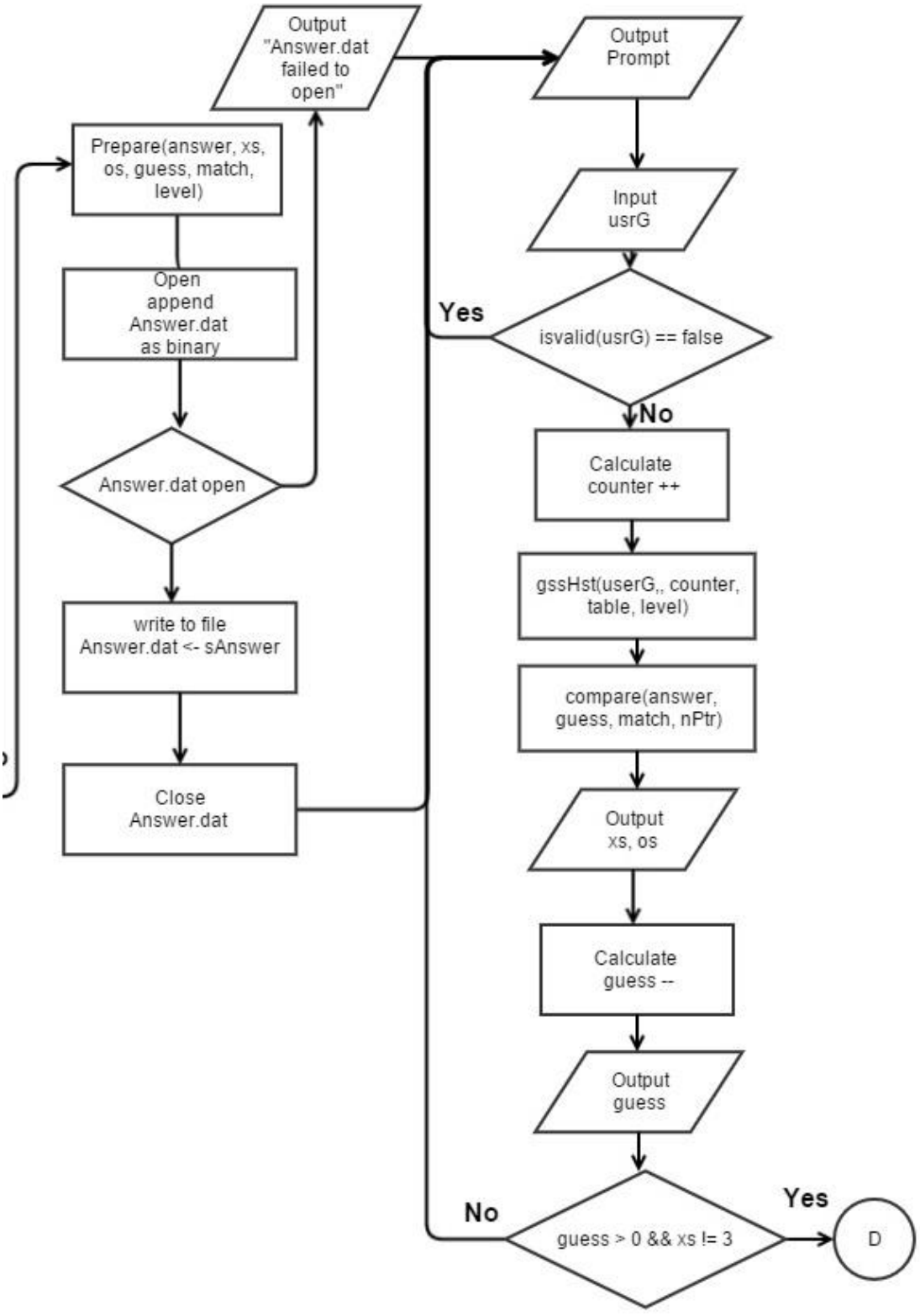


Main

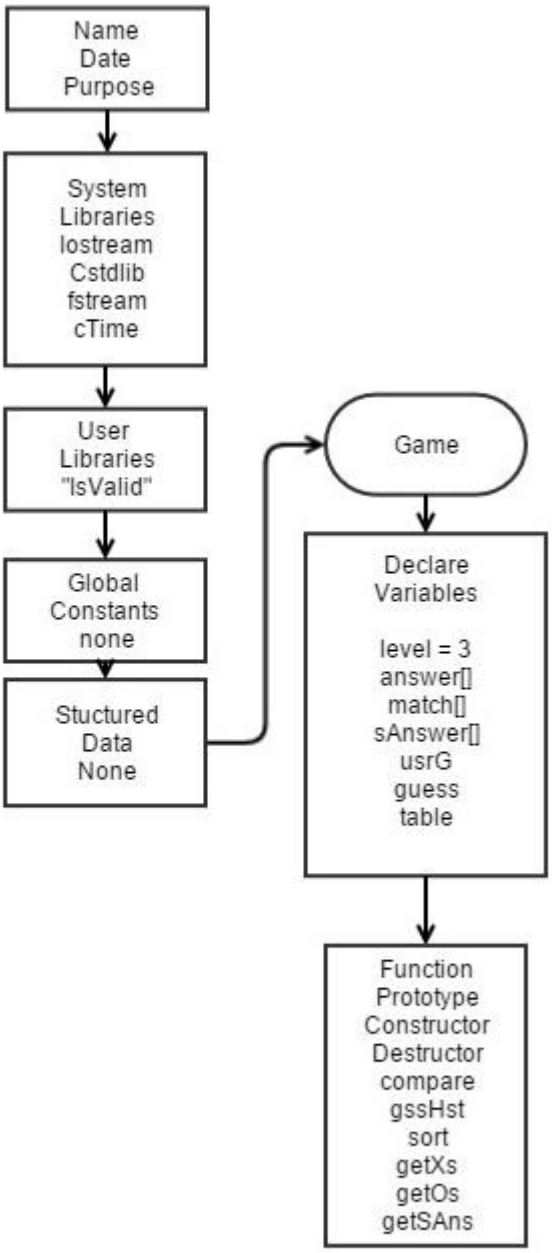

B
C
Open
instruction.dat
instruction.dat
opened
No
Output
File failed to
open
Yes
instruction.dat at EOF
Yes
Close
instruction.dat
No
getline(input, temp)
Output
temp
Output
Select difficulty
Input
temp
Yes
temp length != 1
No
Yes
temp != 1 & 2
No

Main

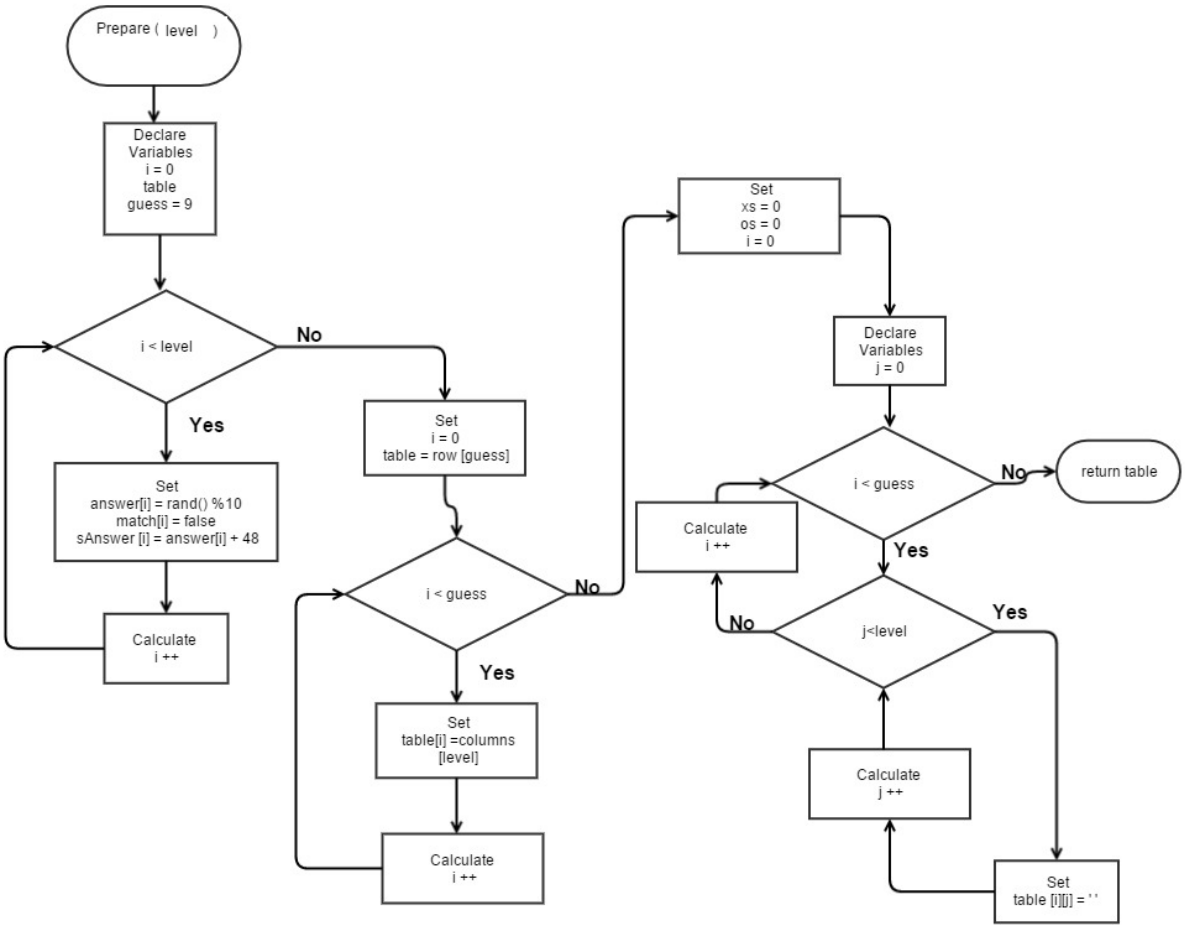

Output "Answer.dat failed to open"
Output Prompt
Prepare(answer, xs, os, guess, match, level)
Input usrG
Open append Answer.dat as binary
Yes
isvalid(usrG) == false
No
Answer.dat open
Calculate counter ++
write to file Answer.dat <- sAnswer
gssHst(userG,, counter, table, level)
compare(answer, guess, match, nPtr)
Close Answer.dat
Output xs, os
Calculate guess --
Output guess
No
guess > 0 && xs != 3
Yes
D


Output "Answer.dat failed to open"
Output Prompt
Prepare(answer, xs, os, guess, match, level)
Input usrG
Open append Answer.dat as binary
Yes
isvalid(usrG) == false
No
Answer.dat open
Calculate counter ++
write to file Answer.dat <- sAnswer
gssHst(userG,, counter, table, level)
compare(answer, guess, match, nPtr)
Close Answer.dat
Output xs, os
Calculate guess --
Output guess
No
guess > 0 && xs != 3
Yes
D

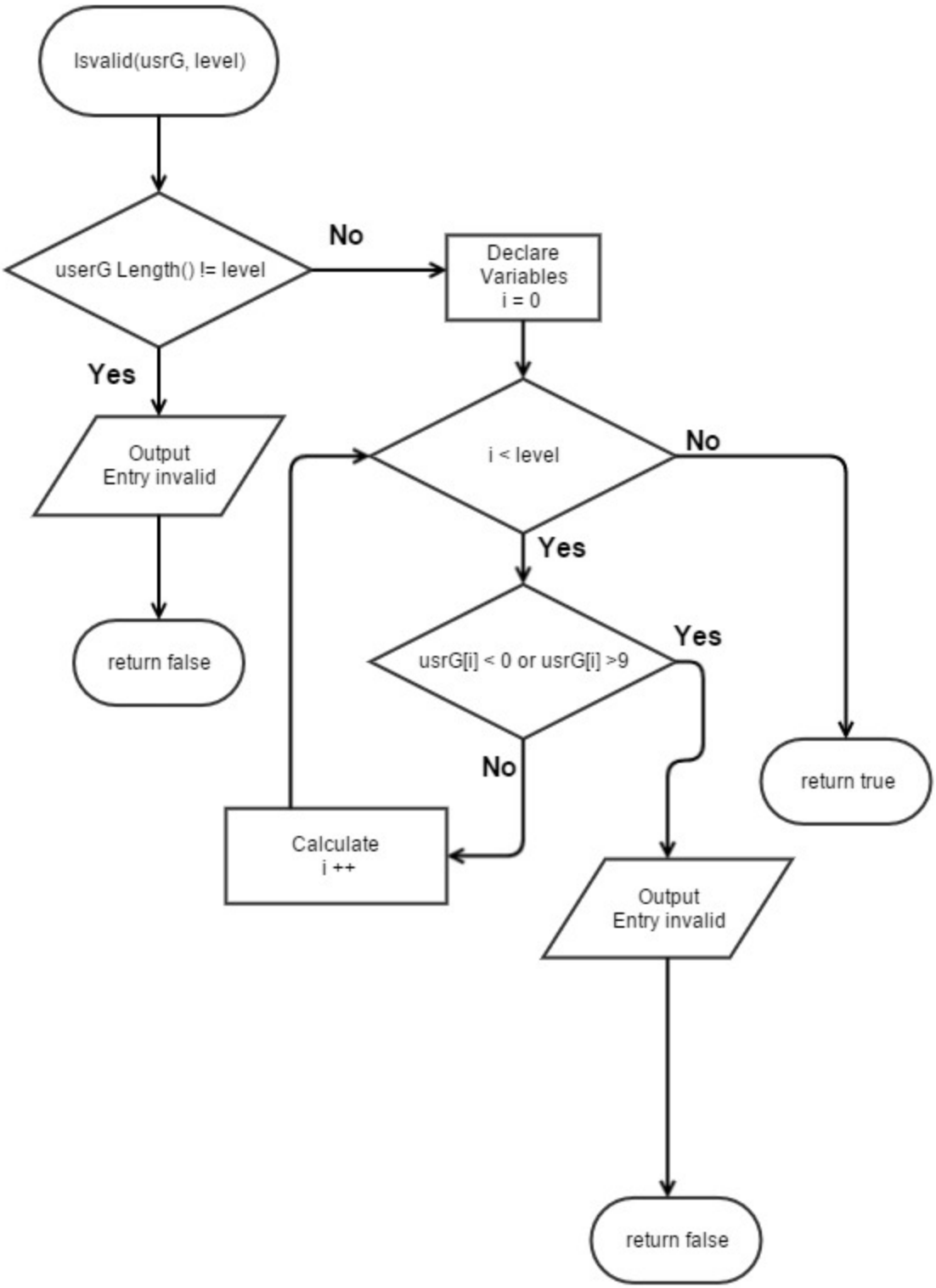
Game.h


Name
Date
Purpose
System
Libraries
Iostream
Cstdlib
fstream
cTime
User
Libraries
"IsValid"
Global
Constants
none
Stuctured
Data
None
Game
Declare
Variables
level = 3
answer[]
match[]
sAnswer[]
usrG
guess
table
Function
Prototype
Constructor
Destructor
compare
gssHst
sort
getXs
getOs
getSAns

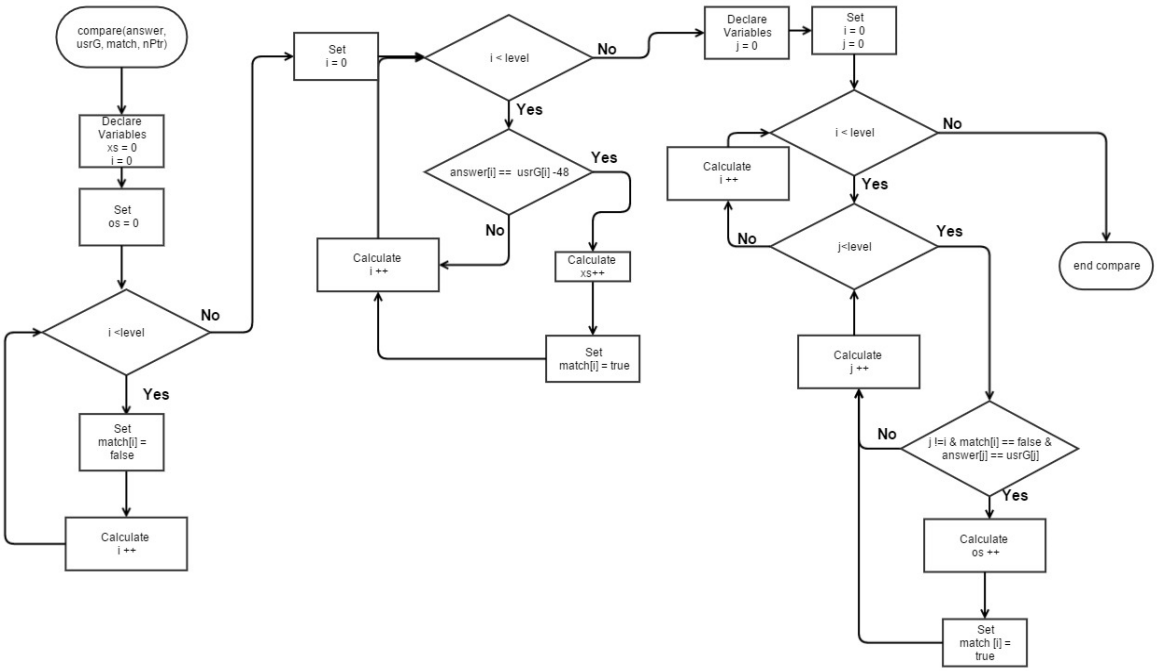
Game Cons.


Prepare (level)
Declare
Variables
i = 0
table
guess = 9
i < level
No
Yes
Set
answer[i] = rand() %10
match[i] = false
sAnswer [i] = answer[i] + 48
Calculate
i ++
Set
i = 0
table = row [guess]
i < guess
No
Yes
Set
table[i] =columns
[level]
Calculate
i ++
Set
xs = 0
os = 0
i = 0
Declare
Variables
j = 0
i < guess
No
return table
Yes
Calculate
i ++
No
j<level
Yes
Calculate
j ++
Set
table [i][j] = ' '

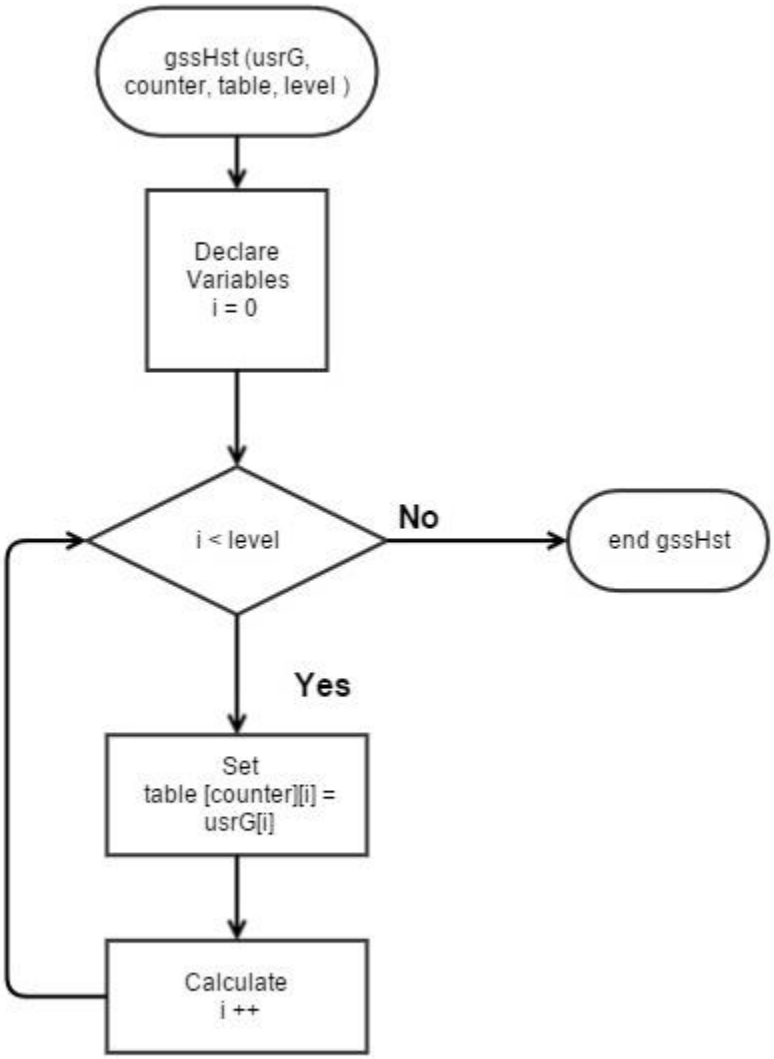
Validate


Isvalid(usrG, level)
userG Length() != level
No
Yes
Declare
Variables
i = 0
Output
Entry invalid
return false
i < level
No
Yes
usrG[i] < 0 or usrG[i] >9
Yes
No
return true
Calculate
i ++
Output
Entry invalid
return false

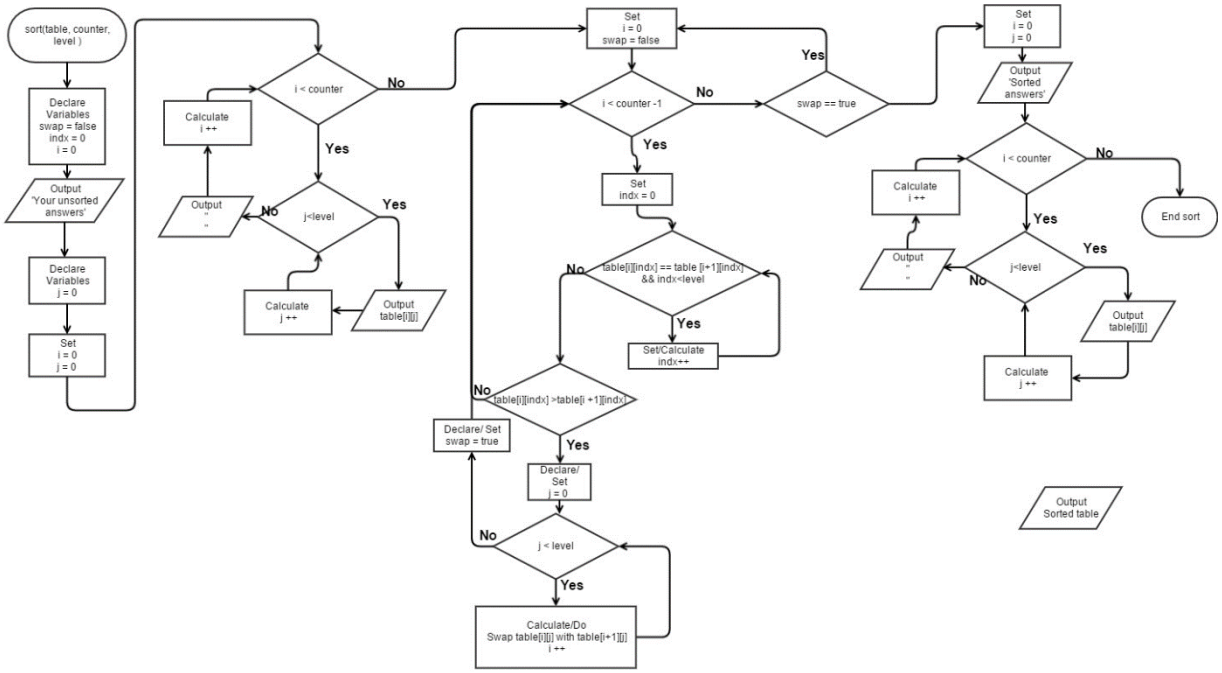
Compare


compare(answer, usrG, match, nPtr)
Declare Variables xs = 0 i = 0
Set os = 0
i <level
No
Yes
Set match[i] = false
Calculate i ++
Set i = 0
i < level
No
Yes
answer[i] == usrG[i] -48
Yes
No
Calculate xs++
Calculate i ++
Set match[i] = true
Declare Variables j = 0
Set i = 0 j = 0
i < level
No
Yes
Calculate i ++
No
j<level
Yes
end compare
Calculate j ++
No
j !=i & match[i] == false & answer[j] == usrG[j]
Yes
Calculate os ++
Set match [i] = true

gssHst


gssHst (usrG, counter, table, level)
Declare Variables i = 0
i < level
No
end gssHst
Yes
Set table [counter][i] = usrG[i]
Calculate i ++

Sort


sort(table, counter, level)
Declare Variables swap = false indx = 0 i = 0
Output 'Your unsorted answers'
Declare Variables j = 0
Set i = 0 j = 0
i < counter
No
Yes
Calculate i ++
Output " "
j<level
No
Yes
Calculate j ++
Output table[i][j]
Set i = 0 swap = false
i < counter -1
No
Yes
swap == true
Set indx = 0
table[i][indx] == table [i+1][indx] && indx<level
No
Yes
Set/Calculate indx++
table[i][indx] >table[i +1][indx]
No
Yes
Declare/ Set swap = true
Declare/ Set j = 0
j < level
No
Yes
Calculate/Do Swap table[i][j] with table[i+1][j] j ++
Set i = 0 j = 0
Output 'Sorted answers'
i < counter
No
Yes
End sort
Calculate i ++
Output " "
j<level
No
Yes
Output table[i][j]
Calculate j ++
Output Sorted table

8. Input output (raw)

```
**********************************************************************************
**************
**                                  **Mastermind jr.**                            **
**          You have to guess numbers that are between 0-9 within 9 guesses!          **
**            First select either easy or normal. easy = 3 numbers normal = 4          **
**                          First type in three or four numbers as your guess          **
**       Then the game will show you the amount of Os and Xs you have for that guess.     **
**   An O means that one of your numbers is correct but it is in the wrong position       **
**       An X means that you have a number correct and in the right possition!          **
**   So guess away! But remember you only have 9 guess. Run out of guesses and you lose.   **
**                Do you have what it takes? Can you decode the sequence?             **
**                              We will soon find out...;)                            **
**********************************************************************************
***************What difficulty would you like to play? 1 for Easy or 2 for Hard: 2
Input your guess:
12314
Please enter 4 numbers.
Input your guess:
1231215
Please enter 4 numbers.
Input your guess:
dfe
Please enter 4 numbers.
Input your guess:
1234
x=0
o=3
Guesses left: 8
Input your guess:
9999
x=1
o=0
Guesses left: 7
Input your guess:
5689
x=0
o=1
Guesses left: 6
Input your guess:
342
Please enter 4 numbers.
Input your guess:
2315
x=2
o=1
```

Guesses left: 5
Input your guess:
2912
x=4
o=0
Guesses left: 4
Congrats you won!
Unsorted answers from you!!
1234
9999
5689
2315
2912
Sorted answers via bubble sorts !!
1234
2315
2912
5689
9999
Play again Y/N?
1
Play again Y/N?
4
Play again Y/N?
s
Play again Y/N?
Y
****
****
**   **Mastermind jr.**   **
**   You have to guess numbers that are between 0-9 within 9 guesses!   **
**   First select either easy or normal. easy = 3 numbers normal = 4   **
**   First type in three or four numbers as your guess   **
**   Then the game will show you the amount of Os and Xs you have for that guess.   **
** An O means that one of your numbers is correct but it is in the wrong position   **
**   An X means that you have a number correct and in the right possition!   **
** So guess away! But remember you only have 9 guess. Run out of guesses and you lose. **
**   Do you have what it takes? Can you decode the sequence?   **
**   We will soon find out...;)   **
****
**********What difficulty would you like to play? 1 for Easy or 2 for Hard: 1
Input your guess:
123
x=0
o=0
Guesses left: 8

Input your guess:
123
x=0
o=0
Guesses left: 7
Input your guess:
123
x=0
o=0
Guesses left: 6
Input your guess:
1234
Please enter 3 numbers.
Input your guess:
543
x=0
o=0
Guesses left: 5
Input your guess:
674
x=1
o=1
Guesses left: 4
Input your guess:
865
x=1
o=0
Guesses left: 3
Input your guess:
456
x=0
o=0
Guesses left: 2
Input your guess:
345
x=0
o=0
Guesses left: 1
Input your guess:
86
Please enter 3 numbers.
Input your guess:
567
x=1
o=1
Guesses left: 0

Sorry but you ran out of guesses. :(
Unsorted answers from you!!
123
123
123
543
674
865
456
345
567
Sorted answers via bubble sorts !!
123
123
123
345
456
543
567
674
865
Play again Y/N?
N
See you again next time!

RUN SUCCESSFUL (total time: 1m 21s)

Code
Main

```
/*
 File:   Game.cpp
 Author: Jonathan Balisky
 Created on July 25, 2015, 9:18 pM
 Purpose:  Mastermind .
 */

//Libraries



#include <iostream>
#include <string>
#include <cstdlib>
#include <fstream>
#include <ctime>

//User libraries
#include"IsValid.h"
#include "Game.h"

using namespace std;
//structured data
//struct Numbers{
//    int xs; //How many x's
//    short os; //How many o'x
//};

//Functions Prototypes

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

    int level = 3; //Difficulty  of gam2
    string temp; //For the file output
    fstream inOut; // for file stream
```

```
string usrG; //The users guess or input


srand(time(0)); //setting time seed
do {

  // counter = 0;



   cout<<"Call prepare."<<endl;//For diagonostics
   inOut.open("instructions.dat", ios::in);
   if (inOut.is_open()) {
      while (getline(inOut, temp)) {
         cout << temp;
      }
      inOut.close();
   } else {
      cout << "Instructions failed to open" << endl;

   }
   do {
      do {
         cout << "What difficulty would you like to play? 1 for Easy or 2 for normal: ";

         getline(cin, temp);
      } while (temp.length() != 1); // User did not enter 1 digit
   } while (temp[0] != 49 && temp[0] != 50); // User did not enter 1 or 2

   if (temp[0] == 49) { //Level is easy
      level = 3;
   } else { //User selected hard level
      level = 4;
   }
   //Game decleration
```

```
//      iV.setLevel(level);//setting the level (Maybe this should be put into a class function?)
        //Game
        Game<char> game(level); //Game varriable
       // game.setLevel(level);

       // table = prepare(answer, guess, match, level, sAnswer, nPtr); //Initialize (shouldnt need
this)

        // cout<<"Call prepare."<<endl;//For diagonostics
        //   //Out Put answer to a file
        //      inOut.open("Answer.dat", ios::out);
        //
        //      if(inOut.is_open()){
        //
        //          for (int i = 0; i<level; i++){
        //           inOut<<answer[i];
        //           }
        //          inOut.close();
        //      }
        //      else{
        //          cout<<"Failed to write answer to file";
        //      }
        //
        //Output answer as Binary
        inOut.open("Answer.dat", ios::out | ios::binary | ios::app);

        if (inOut.is_open()) {
           inOut<<endl;
           inOut.write(game.getSAns(), sizeof (game.getSAns()));
           inOut.close();
        }
        else {
           cout << "Failed to write answer to file" << endl;
        }
```

```
//      for(int i=0;i<3;i++){
//cout<<answer[i]; //For diagonostics
//      }

do {
   do {
      cout << "Input your guess: " << endl; //User enter guess
      getline(cin, usrG); //This should be removed And simply call a class function to do this

     // iV.setUserG(usrG); //Setting user guess in the is valid...
      //cin.ignore();

   } while (game.validate(usrG) == false); //Loop until user enters valid answer
   cout<<usrG<<endl;
//      game.setUserG(usrG);//Setting user guess in the Game class

   game.gssHst(usrG); //Storing the user's guess
  // counter++; //gssHst Ran

   game.compare(usrG);
  //compare(answer, usrG, match, level, nPtr);

   cout << "X(s)=" << game.getXs() << endl; //Right numbers in right space
   cout << "O(s)=" << game.getOs() << endl; //How many Correct number but in the
incorrect space
   //guess--;
   cout << "Guesses left: " << game.getGuess() << endl;



} while (game.getGuess() > 0 && game.getXs() != level); // Continue if user
// has more guesses or has not guess correctly yet

if (game.getXs() == level) { //user won
   cout << "Congrats you won!" << endl;
} else { //user lost
```

```
            cout << "Sorry but you ran out of guesses. :( " << endl;

        }


        game.sort();
       // sort(table, counter, level);

//        for (int i = 0; i < 9; i++) {
//            delete table[i];
//        }
//        delete[] table;
        do { //Checking for an input of Y or N
            do { //Checking for input over 1 char
                cout << "Play again Y/N?" << endl;
                getline(cin, usrG);
            } while (usrG.length() != 1);
        } while (toupper(usrG[0]) != 89 && toupper(usrG[0]) != 78);
        //}while(usrG[0] > 'Y' || usrG < 'N' || (usrG[0]>'N' && usrG[0]<'Y'));
        cout<<"Check"<<endl;
    } while (toupper(usrG[0]) == 'Y');

    cout << "See you again next time!" << endl;



    return 0;
}
```

Game.h

```
/*
 * File:   Game.h
 * Author: Jonathan
 * Purpose: changing  game to moduls
 * Created on November 21, 2015, 2:47 PM
 */

#ifndef GAME_H
#defineGAME_H
#include <iostream>
#include <string>
#include <cstring>
#include <ctime>
#include <cstdlib>
#include "IsValid.h"
using namespace std;
template <class T>
class Game :public IsValid{
private:
    T *answer; //randomly gen. answer
    int guess; // how many guesses the user has left
    int level; //what difficultly the player selected
    int counter; //Which guess number the user is on
//    string usrG; //the user's guess
    short xs, os; //Correct prosition and correct number
    char *sAnswer;
    char **table; //Table for answers
    bool *aMatch; //Array for answer matches
    bool *gMatch; //Array for the guess matches

public:
    //only constructor , one para
    Game (int level);
    ~Game();
    short getXs() const;
```

```
    short getOs() const;
    int getGuess() const;
    char * getSAns() const;
//    void setLevel(int);
//    void setUsrG(string);
     void compare(string usrG) {

//       cout<<"\n Compare ran \n";
//       cout<<usrG<<endl;
       //Reseting vaules = 0
      xs = 0;
      os = 0;

    for (int i = 0; i < level; i++) {
          aMatch[i] = false; //int all values to zero again
          gMatch[i] = false;
                //    cout<<"\n False int. ran \n";
       }



       //Checking for correct numbers in the right position
       for (int i = 0; i < level; i++) {
          // cout<<"usrG["<<i<<"] =" <<usrG[i]<<endl;
          if (answer[i] == (usrG[i] - 48)) { //subtracting 48 makes it an integer
             xs++;
//             cout<<"aMatch["<<i<<"] = true"<<endl;
             aMatch[i] = true;
             gMatch[i] = true;
          }
       }
//      cout<<"level = "<<level<<endl;
       //Checking for os
       for (int i = 0; i < level; i++) { //i is position of answer
//          cout<<"answer["<<i<<"] =" <<answer[i]<<endl<<"aMatch["<<i<<"] =
"<<aMatch[i]<<endl
```

```
//                  <<"userG["<<i<<"] = "<<usrG[i]<<endl;
          for (int j = 0; j < level; j++) { //j is position of usrG(user guess)
             if (j != i && aMatch[i] == false && gMatch[j] == false && answer[i] == usrG[j] - 48)
{
//                   cout<<" usrG["<<j<<"] = "<<usrG[j]<<endl
//                                  <<"gMatch["<<i<<"] = "<<gMatch[j]<<endl;
                os++;
//                  cout<<"os = "<<os<<endl;
                aMatch[i] = true;
                gMatch[j] = true;



             }

          }
       }
     //user has used a guess
     guess--;
   }

   void gssHst(string usrG){



       for (int i = 0; i < level; i++) {
         table[counter][i] = usrG[i];
      }
       counter++;
   }

   void sort(){
   bool swap = false; //For the bubble swap function
   int indx = 0; //Second index location for table

   cout << "Unsorted answers from you!!" << endl;
   for (int i = 0; i < counter; i++) {
```

```
      for (int j = 0; j < level; j++) {
         cout << table[i][j];
      }
      cout << endl;
   }

   do {
      swap = false;

      for (int i = 0; i < counter - 1; i++) { //counter -1 because can swap the last one with the one
after
         indx = 0;
         while (table[i][indx] == table[i + 1][indx] && indx < level)indx++; //Checking to see if
current
         //index is equal then going to next location if they
         //are equal.
         if (table[i][indx] > table[i + 1][indx]) { //If that row and col. not equal
            //then check if first is larger if it is then swap all numbers
            //cout << endl << "i = " << i << endl;
            for (int j = 0; j < level; j++) { //Swapping each 2 rows and their respective columns
until all the rows are swapped
               // cout << "Table[i] = " << table[i][j] << "table[i+1] = " << table[i + 1][j] << endl;
//For diagnostics
               table [i][j] = table [i][j]^table[i + 1][j]; //in place swap. to hopefully make Dr lehr
happy so he give me extra credit
               table [i + 1][j] = table [i][j]^table[i + 1][j];
               table [i][j] = table [i][j]^table[i + 1][j];
            }
            swap = true;
         }

      }
   } while (swap == true);

   cout << "Sorted answers via bubble sorts !!" << endl;
   for (int i = 0; i < counter; i++) {
      for (int j = 0; j < level; j++) {
```

```
            cout << table[i][j];
        }
        cout << endl;
    }

    }
  //  bool validate ();



};
template <class T>
Game<T>::Game(int level) :IsValid(level){

//    setLevel(level);
    //you can ask here and delete the level parameter which is level here
    //Game(int level) this level is a para!!!!!
//    level = l;

    //Dynamicly allocating memory
   answer = new T [level]; //
   sAnswer =  new char [level];
   aMatch = new bool [level]; //which collums are answers arematched
   gMatch = new bool [level]; //which guesses are right but wrong possition

//Setting default vaules
    xs = 0;
    os = 0;
    guess = 9;
    counter = 0;

    for (int i = 0; i < level; i++) {
        answer[i] = rand() % 10; //creating answer from 0-9
        sAnswer [i] = answer[i] + 48; //ascii equivalent to numbers
        aMatch[i] = false; //Set all to false
        gMatch[i] = false; //set all to false
```

```
        //cout<<"answer = "<<answer[i]; //For diagonostics
    }



    //
    table = new char *[guess]; //Creating 2 d dynamic array
    for (int i = 0; i < guess; i++) { //
        table[i] = new char[level];
    }
    // cout<<endl;


    for (int i = 0; i < guess; i++) {//Filling the array with empty spaces
        for (int j = 0; j < level; j++) {
            table[i][j] = ' ';
        }
    }



}
//Destructor
template <class T>
Game<T>::~Game(){
    //if Game was activated and answer was actually created then delete.
   // if(answer[0] != NULL){
        delete[] answer; //
        delete[] sAnswer;
        delete[] aMatch;
        delete[] gMatch;
        //Deleting 2d array
      for (int i = 0; i<9; i++){
          delete table[i];
     }
        delete[] table;
    //}
```

```
}
//setting level into the private variables
//void Game<T>::setLevel(int l){
//    level = l;
//}
//setting usrG into the private variables
//void Game<T>::setUsrG(string uG){
//
//    usrG = uG;
//}
//Getting xs
//need to put the return type at the beginning
//only return var, should be const right?
//just showing him you know the concept
template <class T>
short Game<T>::getXs() const{
   return xs;
}
//Getting os
template <class T>
short Game<T>::getOs() const {
   return os;
}
//Getting the amount of guesses left
//if you only have one line command inside the function, you can put it into head file
//called inline functino  cannot remember how to spell, sth inside the book
template <class T>
int Game<T>::getGuess() const{
   return guess;
}

//getting saved answer
template <class T>
char *Game<T>::getSAns()const{
   return sAnswer;
}#endif          /* GAME_H */
```

IsValid.h

```
/*
 * File:   IsValid.h
 * Author: Jonathan
 *
 * Created on November 19, 2015, 9:57 AM
 */


#ifndef ISVALID_H
#defineISVALID_H
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

class IsValid{
protected:
   string usrG; //users guess
   int level; //Difficulty of the game chosen by user
public:
   IsValid(int level){this->level=level;}
   void setUserG(string usrG);
   void setLevel(int l);
   bool validate (string);
};



#endif  /* ISVALID_H */
```

IsValid.cpp

```
/*
 * File:   IsValid.cpp
 * Author: Juanathan Balisky
 * Created on 11,19,2015
 * Purpose: Implementation isValid
 */

//User Library for the Specification
#include "IsValid.h"

void IsValid::setUserG(string usrG){

    cout<<this->usrG<<endl;
   // cout<<"IsValid was called and string was set"<<endl;
}

 void IsValid::setLevel(int l){
    level = l;
   // cout<<"IsValid was Called and Level was set"<<endl;
}

/****************************************isvalid*********************************
*****************
 * Purpose: To check whether or not the user entered correct amount of numbers
 * Input: usrG, level
 * Output: True or false


*******************************************************************************
*****************/

bool IsValid::validate(string a){
    // cout<<"Call isvalid."<<endl; //For diagonostics
//    for(int i = 0; i<level; i++){
//        cout<<"usrG["<<i<<"] = "<< usrG[i]<<endl;
//    }
```

```
    if (a.length() != level) {
        cout << "Please enter " << level << " numbers." << endl;
        return false;
    } else {
        for (int i = 0; i < level; i++) {
            if (a[i] < 48 || a[i] > 57) {
                cout << "Please enter numbers and numbers only." << endl;
                return false;
            }
        }
        // cout<<"number valid"<<endl;//For diagonostics
        return true;
    }


}
```