Project 2: Mastermind!

Write-Up

Lucas Banks
CSC 5 48102
Dr. Mark Lehr
12/12/16

Table Of Contents

Title Page_____ 1
Table of Contents_____ 2
Summary_____ 3
Flowcharts!_____ 4
Psuedocode_____ 6
Code _____ 7

Summary

This program runs a text based game called Mastermind. In this game, there will be a board with 4 spots. Each spot on the board has a letter assigned to it. The goal of the game is for the user to figure out in what order the letters are assigned to the board. There are 4 difficulty modes. Easy, intermediate, hard, and beginner. Beginner mode is for new players to help them get a grasp of the game. At that difficulty, there are 4 letter choices, which also means that there are only 4 possible letters that could take up any given spot on the board.

As the user guesses the order of the letters, the program will tell you exactly which spots are right, and which spots are wrong. The user can only guess 4 letters at a time with no repeats, which is a rule that applies to all difficulties. In Easy mode and up, the program won't tell you exactly which spots are right or wrong. Instead, it will tell you if you have a letter on the right spot, and if you have a letter that exists somewhere on the board, but not on the right spot. It won't tell you what order. In Easy mode, there are 6 possible letter choices. In intermediate, there are 7, and Hard, 8. There are only 4 board spots for each mode.

The program will save players' highscores & names into files. The user can choose to view the highscores, or even clear the scoreboard. It will only show the top 10 scores.

Flowcharts

Global & Main
https://www.gliffy.com/go/publish/11370307

string setBoard();
https://www.gliffy.com/go/publish/11370957

string checkInput();
https://www.gliffy.com/go/publish/11370701

Void printInstructions();
https://www.gliffy.com/go/publish/11371001

Void printScore();
https://www.gliffy.com/go/share/se04ukxece5qg511jdv5

Void sortScores();
https://www.gliffy.com/go/share/s0p7q1v7voyie5o41rlb

Void saveScores();
https://www.gliffy.com/go/share/so881x5i653eg0qmlcfx

Psuedocode!

Int main() -

User will be presented with a menu where they can select difficulty or choose other options such as viewing the scoreboard. They are also given the option to quit out of the program.

Game will vary in difficulty depending on the difficulty selected by the user.

The board spots are then assigned when the user selects a difficulty.

The user is allowed to make a guess.

The program will check to see if the user entered a guess that’s appropriate to the given difficulty. If it’s not, then it will prompt user to enter a valid guess.

When a valid guess is entered, the program will check how accurate it was.

If it’s wrong, the program will have the user guess again.

If the user selected beginner difficulty, the game will output the exact board spots that he/she guessed right or wrong.

If the user selected any other difficulty, the game will output o’s for every letter chosen by the user that exists on the board, but isn’t in the right spot, and O’s for letters that are in the right spot. But not in any particular order.

The user can input 1 during the game to quit out.

After a user guesses the spots correctly, the program will save a score and have the user enter a name.

The score and name is then saved.

The program will ask if you want to play again. From there, the user can go to the main menu and play again, or quit altogether.

String setBoard() -

Depending on what difficulty the user selected, the board will be assigned with a combination of letters ranging from 4 possible letter choices to 8 possible letter choices.

The board is then randomized with the possible letter choices.

No repeats letters will be assigned. As the program assigns board spaces, it will check to see if a letter has already been assigned to a previous space so that it doesn't assign that letter to another space.

Program will print out possible choices so that the player can know what the options are.

String checkInput() -

Program will know what the valid input choices are based off the difficulty selected.

User will enter the guess. If the guess is not exactly 4 characters or uses characters that are invalid, the user will be prompted to input again.

Allows for lowercase & uppercase letters.

void printInstructions() -

Prints the instructions & prompts user to enter anything to continue.

void saveScore() -

Saves the user's score & name into files.

void printScores() -

Tests to see if the user's score breaks the top 10. If it does, it gets placed in the scoreboard.

The scoreboard will print when the user selects print scoreboard.

void printScores() -

Will sort the scores so that there are in order.

CODE

```
#include <iostream> //input/output objects
#include <string> //enables string data type
#include <cstdlib> //stadndard library
#include <ctime> //enables rand
#include <iomanip> //enables stream manipulation
#include <fstream> //enables file io
#include <sstream> //to convert ints to string

using namespace std;

//Global Constants - Array Size Initialization
const int ROW=11,
          COL=2;

//Function prototypes
string setBoard(string user, string&);
string checkInput(unsigned short, bool&, string&);

void printInstructions();
void saveScore(float, string);
void printScore(float, string);
void sortScores(string[][COL]);

//execution begins here
int main()
{
//passes in the appropriate values to set the board for the difficulty
//selected by the user.
  string begnner(4, '.'),
         easy(6, '.'),
         interm(7, '.'),
         hard(8, '.');
    string guess, //the player's guess will be stored
           spot(4,'.'), //where letters will be randomly assigned
           chkSpot, //checks how accurate the player's guess is
```

```
          again="y", //variable that user input will determine if he
plays again
          choices, //Where list of valid choices will be stored for
the difficulty
          accuraO = (4,""), //where Os and os will be sorted
          accura=(4,""), //Where Os and os will be stored
          dif, //variable where user input to determine difficulty
will be stored
          cont,
          name; //User will enter their name to save their score.
    bool win, //Keeps game in a loop until character wins.
         quit; //keeps menu in loop until user chooses to quit
    unsigned short numOf, //number of valid choices
                   oCount=0, //where number of os will be stored
                   oOCount=0; //where number of Os will be stored
    float score=0; //where score will be kept
    ifstream file; //Where high-score data will be read from.

    RESTART: //will take back to beginning of menu loop
    do{
        //sets win = false so that the game will keep looping until
player
        //wins and/or decides to quit the program.
        win=false;

        //sets quit to false so menu will keep looping until player
decides to
        //leave
        quit=false;

        //Prompts user to select difficulty or leave program.
        cout<<"Welcome to MASTERMIND!"<<endl;
        cout<<"What difficulty do you want to select?"<<endl;
        cout<<"1 = Beginner"<<endl;
        cout<<"2 = Easy"<<endl;
        cout<<"3 = Intermediate"<<endl;
        cout<<"4 = Hard"<<endl;
        cout<<""<<endl;
        cout<<"5 = VIEW HIGH-SCORES!"<<endl;
        cout<<"9 = INSTRUCTIONS"<<endl;
        cout<<"- = DELETE HIGH-SCORES!"<<endl;
        cout<<"+ = Cheat! (Only works when you input it during a
game)"<<endl;
```

```
        cout<<""<<endl;
        cout<<"Input anything else to quit program: ";
        getline(cin,dif);

        //Passes in strings of a length relevant to the difficulty
selected.
        //Also sets score based on difficulty.
        switch (dif[0])
        {
            case '1': {spot=setBoard(begnner, choices);
            numOf=4;
            score=100;
            break;}
            case '2': {spot=setBoard(easy, choices);
            numOf=6;
            score=300;
            break;}
            case '3': {spot=setBoard(interm, choices);
            numOf=7;
            score=375;
            break;}
            case '4': {spot=setBoard(hard, choices);
            numOf=8;
            score=480;
            break;}
            case '5': {printScore(0,"\0");
            goto RESTART;
            break;}
            case '-': {
                      ofstream outfile("names.txt");
                      outfile.close();

                      ofstream outfile1("scores.txt");
                      outfile1.close();

                      cout<<"The scoreboard is now CLEARED"<<endl;
                      cout<<"Enter anything to continue: ";
                      getline(cin,cont);
            goto RESTART;
            break;}
            case '9': {printInstructions(); //calls function to print
instructions
            goto RESTART; //goes to beginning of menu loop
```

```
            break;}
            default: {
                win=true; //sets win to true to break out of loop
                again="n"; //sets again to n to break out of loop
            }
    }

    do{
        //this if will skip the whole block that checks the user's
input for the
        //right letters if user decides to quit.

        if (again!="n"){
        guess=checkInput(numOf, quit, spot);
        if(quit==true)
        {
            break;
        }

    //for loop checks each spot to see how it corresponds with the
player's
    //guess, and then will assign O or X to a variable that will
output
    //during the game so that the player can know how accurate his
guess was
        for (int j=0; j<4; j++)
        {
            (guess[j]==spot[j]) ? (chkSpot[j]='O') : (chkSpot[j]='X');

            if(guess[j]==spot[j])
            {
                accura+="O";
                oOCount++; //increases value per every O assigned
            }
            else if ((spot.find(guess[j])!=
string::npos)&&guess[j]!=spot[j])
            {
                accura+="o";
                oCount++; //increases value per every o assigned
            }
            else if (spot.find(guess[j])==string::npos)
            {
```

```
                accura+=" "; //puts in a blank when neither an O or o
isn't assigned
            }
        }

        //puts an amount of os corresponding to the oCount in the
front of string
        for(int i=0; i<oCount; i++)
        {
            accuraO+="o";
        }

        //after the os are put in, this does the same thing as above,
only with
        //Os. The Os will always be after the os
        for(int i=0; i<oOCount; i++)
        {
            accuraO+="O";
        }

    //if statement checks if user guessed the letters assigned to each
    //spot on the board correctly. If the player is successful, tell
em.
    //if not, tell em.
            if
(chkSpot[0]=='O'&&chkSpot[1]=='O'&&chkSpot[2]=='O'&&chkSpot[3]=='O')
            {
                cout<<"YOU WIN"<<endl;

                cout<<"|| [OOOO] || ";
                cout<<"== [OOOO]"<<endl;
                cout<<endl;

                //Outputs final score
                cout<<"Your final score is
"<<setprecision(4)<<score<<"!"<<endl;

                //Prompts user to type in the name that will be
associated with
                //the score.
                cout<<"Enter your name, you mastermind: ";

                getline(cin,name);
```

```
                while(name.size()>14)
                {
                    cout<<"Input up to 14 LETTERS only"<<endl;
                    cout<<"Enter your name: ";
                    getline(cin,name);

                }
                cout<<endl;

                //Calls saveScore function to save score.
                saveScore(score, name);


                //Sets win = true so that game loop will break
                win=true;
                cout<<"Would you like to play again on a new board?
Enter 'y' for yes, ";
                cout<<"or anything else to quit: ";

                getline(cin,again);

            }
            else
            {
                //Outputs the board to show what the user got right
and the
                //list of their choices
                cout<<"|| ["<<"?"<<"]["<<"?"<<"]["<<"?"<<"]["<<"?"<<"]
|| ";
                if(numOf!=4){
                    cout<<"== ["<<accuraO<<"]";
                }
                else
                {

                    cout<<"==
["<<chkSpot[0]<<chkSpot[1]<<chkSpot[2]<<chkSpot[3]<<"]";
                }
                cout<<"   -- CHOICES: "<<choices<<" -- Enter '1' to
return to the menu."<<endl;
            }
        //resets score, oCount, oOcount, accura & accuraO
        score-=score*0.07;
```

```
        oCount=0;
        oOCount=0;
        accuraO = (4, "");
        accura = (4, "");
        }
    }while(win==false);  //loop will keep players guessing until they
win.
}while(again=="y"||again=="Y"); //loop will keep players in menu until
they-
                                //decide to quit
    return 0;
}//end of main

//0000000111111111122222222223333333333444444444455555555566666666667
7777777778
//3456789012345678901234567890123456789012345678901234567890123456789
1234567890
//************************   setBoard
**************************************
//Purpose:  Sets the board appropriate to the difficulty selected by
the user.
//
//Inputs:   string user -> The difficulty that the user selects.
//          string &choices -> Where list of valid choices will be
stored.
//Output:   string spot -> Where the letters will be randomly
assigned.
//*****************************************************************
**********

string setBoard(string user, string &choices){
    string spot(user.size(),'.'), //where letters will be randomly
assigned
           chkSpot; //checks how accurate the player's guess is
    int boardSet, //where random number assignment (1-4) will be
stored
        repeat[user.size()]={}; //tests for number repetition
    bool exists=false;
    srand(time(NULL));

    //Will assign list of valid choices depending on what difficulty
the user
    //chooses
```

```
    switch(user.size())
    {
        case 4: {choices="Q W A S"; break;}
        case 6: {choices="Q W A S E D"; break;}
        case 7: {choices="Q W A S E D R"; break;}
        case 8: {choices="Q W A S E D R F"; break;}
    }

    //for loop for randomizing the board
    for (int i=0; i<4; i++)
    {
        //sets exist to true at the start of each loop.
        exists=true;

        //gets a random number in-between 1 and the number denoting
        //the length of the string passed in by the user selected
difficulty
        boardSet=(rand() % user.size()) + 1;

        //Tests a user character to see if it's already been used
        //(AKA, if it's already been stored in repeat[])
        for(int j=0; j<4; j++)
        {
            if(boardSet==repeat[j])
            {
                //if there is a repeat variable, set exists to false
                exists=false;
            }
        }

        //first if test will test if the random number is identical to
any previous random number.
        //if the random number IS equal to any previous ones, it will
decrement 'i'
        if (exists==false)
        {
            //(bgrList.find(guess[i])== string::npos) (PREVIOUS
VERSION ^)

            //decrements i so that the for loop will run an extra loop
so that it can
            //assign a number that hasn't already been taken
            i--;
```

```
        }
        //if the random number is unique, it assigns it to a spot on
the board
        else{
        //each number 1-4 corresponds to a letter that will be
assigned to a spot
        //on the board
        switch (boardSet)
        {
            case 1: {
                spot[i]='Q'; //Assigns Q to a spot
            break;
            }
            case 2: {
                spot[i]='W'; //Assigns W to a spot
                break;
            }
            case 3: {
                spot[i]='A'; //Assigns A to a spot
                break;
            }
            case 4: {
                spot[i]='S'; //Assigns S to a spot
                break;
            }
            case 5: {
                spot[i]='E'; //Assigns E to a spot
                break;
            }
            case 6: {
                spot[i]='D'; //Assigns D to a spot
                break;
            }
            case 7: {
                spot[i]='R'; //Assigns R to a spot
                break;
            }
            case 8: {
                spot[i]='F'; //Assigns F to a spot
                break;
            }
        }//ends switch
```

```
        chkSpot[i]=spot[i]; //Assigns letter to check for if the user
guessed correctly.
        repeat[i]=boardSet; //assigns number in repeat variable so it
can test for future repeats
        }//ends else statement

    }//ends for loop
    //outputs the board
    cout<<"|| ["<<"?"<<"]["<<"?"<<"]["<<"?"<<"]["<<"?"<<"] || ";
    cout<<"== [????]";
    cout<<"   -- CHOICES: "<<choices<<" -- Enter '1' to return to the
menu."<<endl;

    return spot; //returns the string which holds the positions of the
letters
                //on the board.
}

//0000000111111111122222222223333333333444444444455555555566666666667
7777777778
//3456789012345678901234567890123456789012345678901234567890123456789
1234567890
//************************   checkInput
**************************************
//Purpose:  Checks to see if player input was valid.

//Inputs:   unsigned short num -> where num of possible board space
choices are passed in
//          0 to 65,535
//          bool &quit -> bool allowing you to quit out of the game.
//          0 or 1
//          string &spot -> where the board is stored.

//Output:   string guess ->  where user's guess is stored
//************************************************************************
**********

string checkInput(unsigned short num, bool &quit, string &spot)
{
    bool check; //variable to check if input is valid
    string guess, //user's guess
           vldList, //where valid list of characters will be stored.
           repeat=(4, "");
```

```
    //assigns the string of valid characters to a variable that will
be searched
    //when it comes to checking if the user input is valid
    switch(num)
    {
        case 4: {vldList="QWASqwas1+"; break;}
        case 6: {vldList="QWASEDqwased1+"; break;}
        case 7: {vldList="QWASEDRqwasedr1+"; break;}
        case 8: {vldList="QWASEDRFqwasedrf1+"; break;}
    }

     do{
            //if nothing is invalid about the user input, this bool
allows
            //the program to pass through the loop
            check=true;

            //Prompts user to enter guess
            cout<<"Enter: ";
            getline(cin,guess);

            if(guess=="1")
            {
                quit=true;
            }
            else if(guess=="+")
            {
                cout<<"THE ANSWER IS: "<<spot<<endl;
            }

             //Checks for valid input
            while((guess.size()!=4)&&(guess!="1"))
            {
                cout<<"Invalid input. Please type in the four valid
letters only. NO REPEATS"<<endl;
                getline(cin,guess);
            }

            //converts lowercase inputs to uppercase
        for(int u=0; u<4; u++){
            switch(guess[u])
            {
            case 'q': {guess[u]='Q'; break;}
```

```
            case 'w': {guess[u]='W'; break;}
            case 'a': {guess[u]='A'; break;}
            case 's': {guess[u]='S'; break;}
            case 'e': {guess[u]='E'; break;}
            case 'd': {guess[u]='D'; break;}
            case 'r': {guess[u]='R'; break;}
            case 'f': {guess[u]='F'; break;}
            }
        }

            //Goes through the user inputted characters one by one,
searches
            //to see if any character doesn't match up with the valid
list of
            //characters.
            for(unsigned int i=0; i<guess.size(); i++)
            {
                if (vldList.find(guess[i])== string::npos)
                {
                    check=false;
                }
            }

            //check = false if user types in 2 of the same chars
            for (unsigned i=0; i<guess.size()-1; i++)
            {   if (guess.find(guess[i], i+1) != string::npos)
                check=false;
            }

            //if any erroneous character was found in previous for
loop, this while
            //will run giving the player an invalid input message
            if(check==false)
            {
                cout<<"Invalid input. Please type in four valid
letters only. NO REPEATS"<<endl;
            }

        }while(check==false);  //will loop if the input was invalid
        return guess; //returns the user's BACKGROUND CHECKED guess
}
```

```
//000000011111111112222222222333333333344444444455555555556666666666777
7777778
//345678901234567890123456789012345678901234567890123456789012345678901
1234567890
//*************************   printInstructions
*************************************
//Purpose:  Opens instructions file and prints the contents of it.

//Inputs:   Nothing is sent in.
//Output:   Nothing is returned out.
//*****************************************************************
**********
void printInstructions(){
    string cont,    //pauses after instructions print until user
inputs anything
           contents; //where file data will be stored
    ifstream textFile; //opens instructions.txt

    //opens file
    textFile.open("instructions.txt");
    if (textFile) { //If file opened successfully, continue with what
you wanna do
        while (textFile >> contents) { //While there is MORE input to
read from the file, do whatever
            cout << contents <<" "; //display data from file
        }
    } else cout << "Error opening file."; //If it doesn't open,
display error

    //close and clear file
    textFile.close();
    textFile.clear();
    cout<<endl;
    cout<<endl;
    //waits for user input to continue back to menu
    cout<<"Enter anything to continue: ";
    getline(cin,cont);

    return;
}

//000000011111111112222222222333333333344444444455555555556666666666777
7777778
```

```
//3456789012345678901234567890123456789012345678901234567890123456789012345678
90
//*************************   saveScore
**************************************
//Purpose:  Saves the score the player got (and his name) into files.

//Inputs:   float score -> The score that the player earned. 1.2E-38
to 3.4E+38
//          string name -> The name that the player inputs.
//Output:   Nothing is returned.
//*****************************************************************
**********
void saveScore(float score, string name){
    ofstream file;

    file.open("scores.txt", fstream::app);
    file<<score<<" \n";
    file.close();

    file.open("names.txt", fstream::app);
    file<<name<<" \n";
    file.close();

    return;
}

//0000000111111111122222222223333333333444444444455555555556666666666777
7777778
//3456789012345678901234567890123456789012345678901234567890123456789012345678
90
//*************************   printScore
**************************************
//Purpose:  Prints the scoreboard if user chooses to do so. Function
is also called
//          whenever the user wins to save his score/name and sort the
high scores.

//Inputs:   float score -> where the score will be passed in. 1.2E-38
to 3.4E+38
//          string name -> where the name will be passed in
//Output:   Nothing is returned.
//*****************************************************************
**********
```

```
void printScore(float score, string name){
    string hiScore[ROW][COL];//2D array where scores and names will be
held
    string cont;//variable to wait for user input to continue
    ifstream scores,//read into scores file
             names;//read into names file
    ofstream outfile, outfile1;//opens scores & names file

        //Opens files
        scores.open("scores.txt");
        names.open("names.txt");

        //stores values from file into array
        for(int i=0; i<ROW; i++)
        {
            scores>>hiScore[i][0];
            names>>hiScore[i][1];
        }

    //tests player's score w/ value in the 10th index of the first
dimension
    //of the array. That number will always be the lowest value in the
array.
    //if the score is bigger, it will replace that element.
    if(score>atoi(hiScore[9][0].c_str())&&score!=0)
    {
        //converts score value to a string so it can be stored in
string array
        ostringstream ss;
        ss<<score;
        hiScore[9][0]=ss.str();
        hiScore[9][1]=name;
    }


        //Calls function to sort scores
        sortScores(hiScore);

        //Closes files
        scores.close();
        names.close();
```

```
        //If score doesn't == 0, that means the function was called
from
        //a user winning the game. So this is here to clear the file
so that
        //it can be replaced with contents from the array with sorted
score data.
        //It wont run if the function was called for the purpose of
printing the
        //score. (In that case, score WOULD = 0, and this wouldn't
run)
        if(score!=0){
            //Clears files
            ofstream file("names.txt")  ;
                     file.close();
            ofstream file1("scores.txt");
                     file1.close();
                     cout<<endl;
                     //cout<<"CHECK TO SEE IF FILES WERE CLEARED: ";
                     //getline(cin,cont);
        }

        //If score == 0, the function was called from a user selecting
the menu
        //option to print the score. When that happens, the function
passes in 0
        //to score. So this WONT print the scoreboard if function was
called from
        //the user winning the game.
            //Prints scoreboard
            cout<<"                  SCOREBOARD"<<endl;
            for(int i=10; i>=1; i--)
            {
                cout<<setw(15)<<hiScore[i][1]<<"                ";

cout<<setprecision(4)<<atoi(hiScore[i][0].c_str())<<endl;
            }

            outfile.open("scores.txt");
            outfile1.open("names.txt");

            //Puts new sorted array into file.
            for(int i=ROW-1; i>0; i--){
```

```
            outfile<<hiScore[i][0]<<" \n";
            outfile1<<hiScore[i][1]<<" \n";
                }

            outfile1.close();
            outfile.close();

        cout<<"Enter anything to continue: ";
        getline(cin,cont);
        cout<<endl;
        cout<<endl;

        return;
        }


//0000000111111111122222222223333333333444444444455555555556666666666777
7777778
//3456789012345678901234567890123456789012345678901234567890123456789012
34567890
//*************************   sortScores
****************************************
//Purpose:  Where scores will be sorted.

//Inputs:   string scores[][] -> 2D array where scores & names will be
kept.
//Output:   Nothing is returned.
//*************************************************************************
**********
void sortScores(string scores[][COL]){

   //Modified selection sort.
    for(int i=0; i<(10); i++)
    {
        for(int j=i+1; j<11; j++)
        {
           static string temp, temp1;

            if(atoi(scores[i][0].c_str())>atoi(scores[j][0].c_str()))
            {
                temp=scores[i][0];
                scores[i][0]=scores[j][0];
                scores[j][0]=temp;
```

```
                temp1=scores[i][1];
                scores[i][1]=scores[j][1];
                scores[j][1]=temp1;
            }
        }
    }

    return;
}
```