**Project 1**

**~MASTERMIND MOD~**

**~The Game~**

**CIS-17a 48983**

**Name : Anh Hoang Le**

**Date : 10/25/2015**

**Introduction**

Title: Mastermind modify - The game

This is a code breaking game, originated from a board game with the same name but using color pegs coded and only 4 for the sequence.

The user will be ask to input a certain size of a sequence as they wish. Then he/she will input the randomize range for each number.

The computer will then generate number of numbers according to size and range of the random and will give you an array of mystery sequence.

The user is given 10 turns to break the sequence and get all of the numbers right . If they get all right in under 10 turns, they will be rewarded as the true Mastermind!

This is a modified version of the game using numbers and dynamic allocated the size for the flexible game play if the user wish to play a harder or guessing a longer sequence. It’s a small mini games for everyone to enjoy~.

**Summary**

Project size : about 380 lines

The number of variables: about 20 variables

The number of methods: 7

This project includes concepts that we learned from chapter 9 to chapter 12 in the book. It has many concepts from earlier chapters that we reviewed such as input and out file console, variables, function,etc.

The project took me about a week and a half because I tried to applied different logic to the game or even more complex ideas for the game. I ran into a lot of troubles and decided not to include it in the project.

Even though it’s a small project with many possibilities of cool programming idea, I still think it’s not completely satisfied me since I want to do more but I got out of time but overall, it was a great experience for reviewing my skills. I remember a lot of the things I forgot and found out new ways to approach the project that I think I will bring them to the next one.

Since this is a very simple game, I tried to applied all the concepts that we learned to the game and make them works.

I think that I need to practice more about object oriented programming and try to make my code even simpler and as well as applying on pointers.

**Description**

Before I write any of the code, I tried to look for ideas that really interest me. I actually decided to make a Pokemon fighting type game but I ran into a lot of troubles without finding a way to make the game suitable with all of the concepts that we learned and I decided to start of from scratch with a new idea. So I was kinda falling behind from the schedule that I set up. As I found this game, I spent about a day writing out pseudo code and outline of the game.

**Flow Chart**

I used Gliffy.com to flow charted my code and it’s a pretty big program to flowchart. It contains my main code as well as all the functions that I used inside the code.

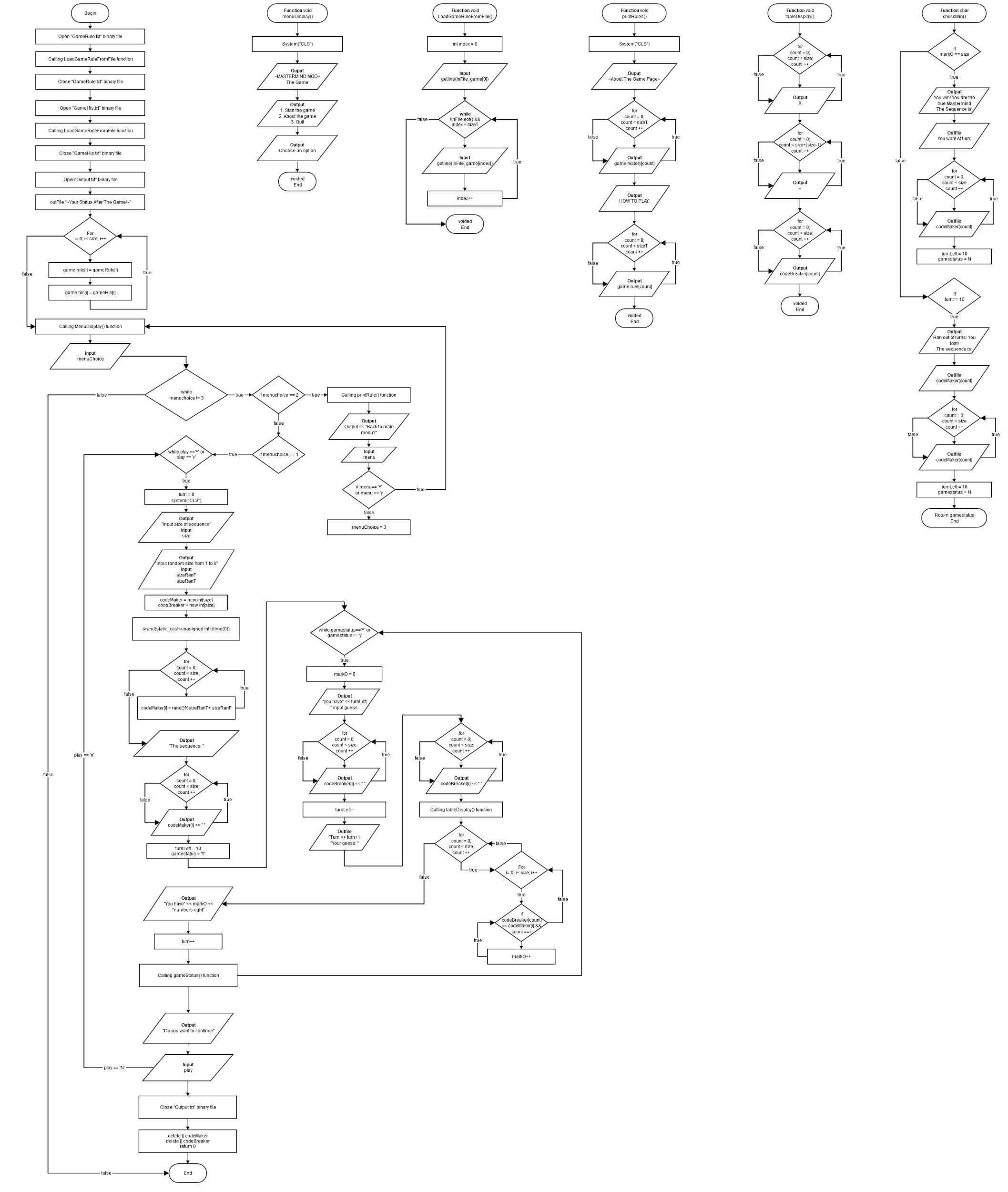
I posted it here for any readers to see but if you still can’t see the flowchart, You can use the link below:

<https://www.gliffy.com/go/html5/9263139?app=1b5094b0-6042-11e2-bcfd-0800200c9a66>

The link will give you direction to my gliffy page which I finalized the flow chart.

If it does not work for you, you can access it through my Github repository with this link:

<https://github.com/hoanganh1111/LeAnh_CSC17a_48983/tree/master/Proj>



**PSEUDO CODE**

***Initialize***

***Output the main menu***

***Input decision***

***while it’s not exit the code***

***{***

***if choice is 2***

***{***

***Display game rule and description***

***}***

***else if choice is 1***

***{***

***while play is yes***

***{***

***Input size of the sequence***

***Input the random from 1-9***

***Randomize the sequence***

***Output the table***

***while the match is running***

***{***

***Input the guess sequence***

***Displaying the table with the input***

***Check for win and return the status***

***Lose a turn***

***}***

***Match end ask if want to continue?***

***Input the decision***

***}***

***Start the match again or break out of loop***

***}***

***}***

***End the code***

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| ifstream | inFile | Input the binary file | init() |
| ofstream | outFile | Output the binary file | init() |
| Integer | sizeT | size of the binary file | init() |
|  | size | size of the sequence | init() |
|  | sizeRanF | randomize sequence from | init() |
|  | sizeRanT | randomize sequence to | init() |
|  | markO | numbers of right guesses | init() |
|  | turn | turns given | init() |
|  | turnLeft | turns left from losing | init() |
|  | \*codeMaker | holding the sequence numbers | init() |
|  | \*codeBreaker | holding the guess each turn | init() |
| char | play | The game is started | init() |
|  | menu | Back to main menu | init() |
|  | gamestatus | While the match is running | init() |
| string[] | gameRule[] | string holding the texts from the binary file | init() |
|  | gameHis[] | string holding the texts from the binary file | init() |
| struct | about  {  game.rule[]  game.his[]  } | Holding textures after loading them from the binary file | init() |

**C++ Concepts**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Syntax and Keywords** | **Location and Examples** |
| 2 | System Library | #include <iomanip> |
| cout objects | cout << endl |
| Escape sequences(\n, \t, \b) | cout << “\nTurn: ”; |
| Integers and string | int markO = 0;  string gameRule[] |
| Special characters(#, {}, <, >, ;, “”, ‘’, (), ) | #include <iostream>  cout <<, cin >>, “GameRule.txt”; |
| 3 | cin objects | cin >> menuChoice |
| Mathematical expressions | turn = 0  size+(size-1) |
| Use of cin.ignore() | cin >> codeBreaker[]  cin.ignore() |
| 4 | If/else selection structure | if(menuchoice == 2){}  else{} |
| Nested if statement | if(menuchoice == 2){}  if(menu==Y||menu ==y){} |
| If/else if statement | if(menu==Y||menu ==y){}  else if (menu==N){} |
| Menu using if/else statement | 1.Start the game  2.About the rule  3.Exit |
| Relational Operators(==, !=, <, >) | menu==Y,  menu!= N,  count < size,  turn > 10 |
| 5 | For repetition structure | for(int i=0;i<size,i++){} |
| Increment operator | turn++ |
| Decrement operator | turnLeft-- |
| while repetition structure | while(menuchoice != 3){} |
| 6 | Function prototyped | void menuDisplay(); |
| Calling a function | LoadGameRuleFromFile(inFile, gameRule, size) |
| Define a function | menuDisplay()  {  cout<< “1.Start the game”  cout << “2.About”  cout << “3.Exit”  } |
| Return a value from function | checkWin()  {  return gamestatus;  } |
| Global variable | const int sizeT = 12; |
| 7 | Arrays | string gameRule[size] |
| Array as function parameter | void LoadGameRuleFromFile(ifstream &inFile, string gameRule[], int size) |
| 9 | Pointers | int \*codeMaker  int \*codeBreaker |
| Dynamic Memory | codeMaker = new int[size]  codeBreaker = new int[size] |
| Release the dynamic array | delete [] codeMaker  delete[] codeBreaker |
| 10 | C-string | #include <string.h> |
|  | String objects | string gameRule[size]  string gameHis[size] |
| 11 | Structured data | struct about  {  string history[sizeT]  string rule[sizeT]  } |
| Accessing structure members | cout << game.rule[sizeT] |
| 12 | File Operations | ifstream inFile  ofstream outFile |
| Opening a binary file | inFile.open(“GameRule.txt”)inFile.open(GameHis.txt”) |
| Close a binary file | inFile.close() |
| Input/Output from/into a binary file | getline(inFile, game[index])  outFile << “Ran out of turns! You lost!” |
| Passing file stream into functions | void LoadGameRuleFromFile(ifstream &inFile, string game[], int sizeT); |

**Reference**

1. Gaddis text book

2. <http://www.cplusplus.com/>

3. <https://en.wikipedia.org/wiki/Mastermind_(board_game)>

**Program**

//MASTERMIND MOD - The Game

//System Libraries

#include <iostream> //Input/Output stream Library

#include <iomanip> //Format Library

#include <string.h> //String Library

#include <fstream> //Input/Output stream text files

#include <cstdlib> //Random Library

#include <ctime> //Time Library

#include <stdlib.h> //Standard Library

using namespace std; //Utilize standard name-space directly

//Global constants

const int sizeT = 12;

//Structure declaration

struct about //Structure about

{

string history[sizeT]; //String array

string rule[sizeT]; //string array

};

//Function prototypes

void menuDisplay();

void LoadGameRuleFromFile(ifstream &, string game[], int);

void printRules(struct about game, int sizeT);

void tableDisplay(int [], int);

char checkWin(ofstream &outFile, int, int, char, int, int [], int);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MASTERMIND MOD GAME \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

{

ifstream inFile; //Input binary file

ofstream outFile; //Output binary file

string gameRule[sizeT]; //string to retrieve game rule

string gameHis[sizeT]; //string to retrieve game description

about game; //structure game with description and rule

int size; //size of the sequence

int sizeRanF; //randomize number from 1

int sizeRanT; //to 9

int menuChoice; //choice of menu

char play = 'Y'; //while the game is running

int markO; //Number of right guesses

int turn; //Number of turn(max 10)

int turnLeft; //count down turns

char menu = 'Y'; //Going back to main menu

char gamestatus = 'Y'; //While the user wants to play again

int \*codeMaker; //Code maker

int \*codeBreaker; //Code breaker

//Open GameRule.txt file

inFile.open("GameRule.txt");

//Loading game rule from file

LoadGameRuleFromFile(inFile, gameRule, sizeT);

//Close the text file

inFile.close();

//Open GameHistory.txt file

inFile.open("GameHistory.txt");

//Load game description from file

LoadGameRuleFromFile(inFile, gameHis, sizeT);

//Close the text file

inFile.close();

//Open Output.txt file

outFile.open("Output.txt");

//Output headline to file

outFile << " ~YOUR STATUS AFTER THE GAME~\n\n";

//Setting the string inside the structure

for(int i = 0; i < sizeT; i++)

{

game.rule[i]= gameRule[i];

game.history[i]= gameHis[i];

}

//Main menu with choices 1, 2, 3

mainMenu:

{

menuDisplay(); //Display the menu

cin >> menuChoice; //Input choice

}

while (menuChoice != 3) //While choice is not 3(exit)

{

if(menuChoice == 2) //If user picks 2

{

//Print rules and description using the printRules function

printRules(game , sizeT);

//You want to back to the main menu

cout << "Back to main menu?(Y): \n";

cin >> menu; //Input the decision

if(menu == 'Y' || menu == 'y') //If they pick yes

goto mainMenu; //GO back to displaying main menu

else if (menu == 'N' || menu == 'n')

menuChoice = 3; //Set menu choice to 3(exit the code)

}

else if(menuChoice == 1) //If menu choice is 1

{

while(play == 'y' || play == 'Y') //While play is yes

{

turn = 0; //Reset the turn

system("CLS"); //Clear the screen

//Prompt for inputing the sequence size

cout << "Input the size of the sequence: ";

cin >> size; cout << endl; //Input the size

//Prompt for randomize range from 1

cout << "Input the random number size from(1): ";

cin >> sizeRanF; //Input 1

//to 2-9

cout << " to(2-9): ";

cin >> sizeRanT; cout << endl; //Input from 2-9

codeMaker = new int[size]; //Dynamic allocated codeMaker using input size

codeBreaker = new int[size]; //Dynamic allocated codeBreaker using input size

//Set random number seed to time.

srand(static\_cast<unsigned int>(time(0)));

//Randomize the number according to user's input

for(int i = 0; i < size; i++)

{

codeMaker[i] = rand()%sizeRanT+sizeRanF;

}

//Output sequence to binary file

outFile << "The sequence: ";

for(int i = 0; i < size; i++)

outFile << codeMaker[i] << " ";

turnLeft = 10; //Reset turnLeft

gamestatus = 'Y'; //Game status is yes playing

while(gamestatus == 'Y' || gamestatus == 'y')

{

markO = 0; //User's right guess reset

//Output how many turn user have left

cout << "You have " << turnLeft << " turns left!\n";

//Prompt for inputing the guess

cout << "Input the guess: ";

//Inputing the guess using codeBreaker array

for (int i = 0; i < size ; i++)

{

cin >> codeBreaker[i];

cin.ignore();

}cout << endl;

//Lost a turn

turnLeft--;

//Output their turn into the binary file

outFile << "\n\nTurn " << turn+1 << ": ";

outFile << "\nYour guess: ";

//Output the guess into the binary file

for(int i = 0; i < size; i++)

outFile << codeBreaker[i] << " ";

//Output how many turns left into the binary file

outFile << "\nTurn left: " << turnLeft << " turns.\n";

//Displaying table with the guesses and sequence size

tableDisplay(codeBreaker, size);

//Check for right guesses of the sequence

for(int count = 0; count < size; count++)

{

for (int i = 0; i < size ; i++)

{

//If the input match the default number by array position and value

if(codeBreaker[count] == codeMaker[i] && count == i)

{

markO++;

}

}

}

//Output how may numbers the user got right

cout << "You have " << markO << " numbers right!";

cout << endl;

turn++; //Adding a turn

//Check for win funtion to see if the user get all the numbers right

gamestatus = checkWin(outFile, markO, turn, gamestatus,

turnLeft, codeMaker, size);

}

//End the match. Prompt for continuation

cout << "\nDo you want to continue?(Y/N): ";

cin >> play; //Input the decision

}

}

//Other input choice other than 1,2 or 3

else

{

//Display invalid

cout << "\nInvalid input, please re-enter: ";

cin >> menuChoice; //Re-input the main menu choice

}

}

//Close the binary file Output.txt

outFile.close();

delete [] codeMaker; //Delete the array stored

delete [] codeBreaker; //Delete the array stored

//End code

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output the Main Menu \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Void menuDisplay();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void menuDisplay()

{

system("CLS");

cout << " ~MASTERMIND MOD~\n";

cout << " The Game \n";

cout << " 1. Start the game.\n";

cout << " 2. About the game.\n";

cout << " 3. Quit.\n";

cout << "\nChoose an Option: ";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\* Loading Game Rule & Description from binary files \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inputs

\* inFile -> input the binary file into the array

\* game[] -> Array to holds the string from the file

\* sizeT -> Size of the string

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LoadGameRuleFromFile(ifstream &inFile, string game[], int sizeT)

{

int index = 0; //Index set to 0

getline(inFile, game[0]); //Input the first line of the file

while(!inFile.eof() && index < sizeT) //While it's not end of the file

{ //and index < size of the file

getline(inFile, game[index]); //Inputing the sentence into the array

index++; //Increment index

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Output Game Rules & Descriptions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inputs

\* inFile -> input the binary file into the array

\* game[] -> Array to holds the string from the file

\* sizeT -> Size of the string

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printRules(struct about game, int sizeT)

{

system("CLS"); //Clear the screen

//Output the headline of the menu

cout << " ~About The Game Page~\n\n";

//Output the game description

for(int i = 0; i < sizeT; i++)

cout << game.history[i] << endl;

//Output how to play

cout << "HOW TO PLAY:\n";

//Output the game rule

for(int i = 0; i < sizeT; i++)

cout << game.rule[i] << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Display Table inside the game \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inputs

\* codeBreaker[] -> The user's guess input array

\* size -> Size of the sequence

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void tableDisplay(int codeBreaker[], int size)

{

//Output the X X X X with the size

for(int i = 0; i < size; i++)

{

cout << "X" << " ";

}cout << endl;

//Output the ----- with the size

for(int i = 0; i < size+(size-1); i++)

cout << "-";

cout << endl;

//Output spaces

for(int i = 0; i < size; i++)

cout << codeBreaker[i] << " ";

cout << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Check for The Solution \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Inputs

\* outFile -> output into the binary file

\* markO -> numbers of right degits

\* turn -> turn the user is on

\* gamestatus -> Game status to repeating the match

\* turnLeft -> Number of turn left

\* codeMaker[] -> The code ramdomize sequence

\* size -> Size of the sequence

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char checkWin(ofstream &outFile, int markO, int turn,

char gamestatus, int turnLeft, int codeMaker[],

int size)

{

//If the user guess all numbers right

if(markO == size)

{

//Output the winning and he/she is the true Master mind

cout << "\nYou win! You are the true MasterMind! \n";

cout << "The sequence is: ";

//Output that the user win into the binary file

outFile << "You won! At turn " << turn+1;

//Output the actual randomize sequence

for(int i=0; i < size; i++)

cout << codeMaker[i] << " ";

turnLeft = 10; //Reset turn left to 10

gamestatus = 'N'; //Game ended

}

//If user ran out of turn without guessing the right sequence

else if (turn == 10)

{

//Output that they lost and the actual sequence is

cout << "\nRan out of turns. You lost! What a Surprise! \n";

cout << "The sequence is: ";

//Output that they lost into the binary file

outFile << "You lost! Maximum of turns reached!";

//Output the actual randomize sequence

for(int i=0; i < size; i++)

cout << codeMaker[i] << " ";

turnLeft = 10; //Reset turn left to 10

gamestatus = 'N'; //Game ended

}

return gamestatus; //Return game status is N

}