**Project 2**

**~MASTERMIND MOD~**

**~TIC TAC TOE~**

**-The Game-**

**CIS-17a 48983**

**Name : Anh Hoang Le**

**Date : 12/12/2015**

**Introduction**

Title: Mastermind modify + Tic Tac Toe - The game

**Master Mind 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~.

**Tic Tac Toe**

The classic game that everyone knows how to play what what it is. It mimics an actual tic tac toe using 3 dimensional array to holds value of the players X and O. Display the board with the values X and O. Who ever reaches three in a row will win the match. The player can change name and restart the game or exit the code.

**Summary**

Project size : about 770 lines

The number of variables: about 40variables

The number of methods: 10

This project includes concepts that we learned from chapter 9 to chapter 16in 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.

Since these are very simple games, I tried to applied all the concepts that we learned to the games 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 and classes.

**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. Then I decided to write tic tac toe program and applied classes to it.

**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.

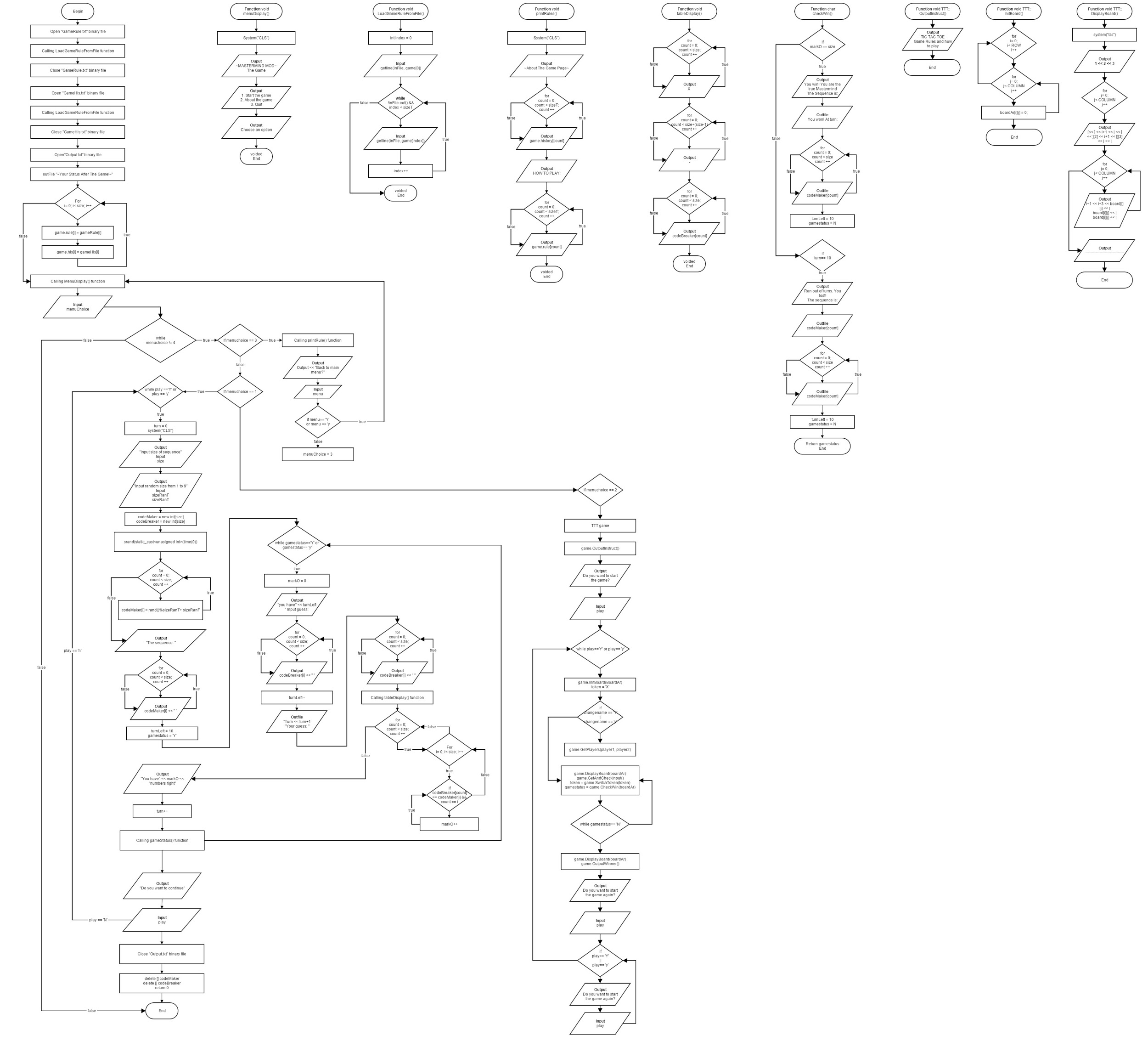
I posted it here for any readers to see but if you still can’t see the flowchart, You can use the link below:

<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 3***

***{***

***Display game rule and description***

***}***

***else if choice is 1***

***{***

***while play is yes***

***{***

***Input size of the sequence***

***Input the random from 1-4***

***Randomize the sequence***

***Convert the sequence into***

***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***

***}***

***else if choice is 2***

***{***

***Output the instruction***

***Ask for first player’s name***

***Ask for second player’s name***

***Reset the screen***

***Initialize the board***

***Display the board***

***Player’s input***

***Switch token***

***Check for winner or out of turns***

***Output the winner***

***Ask for the user decision to play the game again***

***If player pick yes***

***{***

***Ask if user want to change player’s names***

***Yes, input the player’s names***

***}***

***}***

***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() |
| Class | class TTT{  public:  OutInstruct()  CheckWin()  }; | Hold all functions that will be transfer variables into the functions and put out results | 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); |
| 13-15 | Using class | class TTT{}; |
| Public member functions | GetPlayers(string player1, string player2) |
| Define public class member functions | void TTT::GetAndCheckInput() |
| Class declaration | TTT game |
| Access the class functions | game.CheckWin |

**Reference**

1. Gaddis text book

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

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

4. https://en.wikipedia.org/wiki/Tic-tac-toe

**Program**

**//functions.h**

**#ifndef FUNCTIONS\_H\_**

**#define FUNCTIONS\_H\_**

**//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**

**//Constant variables**

**const int COLUMNS = 3;**

**const int ROWS = 3;**

**const int sizeT = 12;**

**//Structure declaration**

**struct about //Structure about**

**{**

**string history[sizeT]; //String array**

**string rule[sizeT]; //string array**

**};**

**//Class Declaration**

**class TTT**

**{**

**private:**

**public:**

**void OutputInstruct();**

**void InitBoard(char boardAr[][COLUMNS]);**

**void DisplayBoard(const char boardAr[][COLUMNS]);**

**void GetAndCheckInp(char boardAr[][COLUMNS], char token, string player1, string player2);**

**char SwitchToken(char token);**

**void GetPlayers(string& player1, string& player2);**

**char CheckWin(char boardAr[][COLUMNS]);**

**void OutputWinner(char whoWon, string player1, string player2);**

**};**

**//Function prototypes**

**void menuDisplay();**

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

**void printRules(struct about game, int sizeT);**

**void tableDisplay(char [], int);**

**char checkWinF(ofstream &outFile, int, int, char, int, char [], int);**

**#endif /\* FUNCTIONS\_H\_ \*/**

**//functions.cpp**

**#include "functions.h"**

**void menuDisplay()**

**{**

**system("CLS");**

**cout << " ~MASTERMIND MOD + TIC TAC TOE~\n";**

**cout << " The Game \n";**

**cout << " 1. Start Mastermind Modified.\n";**

**cout << " 2. Start Tic Tac Toe.\n";**

**cout << " 3. About the game(Mastermind).\n";**

**cout << " 4. Quit.\n";**

**cout << "\nChoose an Option: ";**

**}**

**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**

**}**

**}**

**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;**

**}**

**void tableDisplay(char 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;**

**}**

**char checkWinF(ofstream &outFile, int markO, int turn,**

**char gamestatus, int turnLeft, char codeMakerL[],**

**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 << codeMakerL[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 << codeMakerL[i] << " ";**

**turnLeft = 10; //Reset turn left to 10**

**gamestatus = 'N'; //Game ended**

**}**

**return gamestatus; //Return game status is N**

**}**

**//class.cpp**

**#include "functions.h"**

**void TTT::OutputInstruct()**

**{**

**cout << "\*TIC-TAC-TOE" << endl << endl;**

**cout << "\*There are two players 'X' and 'O'. " << endl << endl;**

**cout << "\*X always goes first." << endl << endl;**

**cout << "\*Players alternate placing Xs and Os on the board until " << endl;**

**cout << " either (a) one player has three in a row, horizontally, " << endl;**

**cout << " vertically or diagonally, or (b) all nine squares are " << endl;**

**cout << " filled." << endl << endl;**

**cout << "\*If a player is able to draw three X's or three O's in a " << endl;**

**cout << " row, that player wins." << endl << endl;**

**cout << "\*If all nine squares are filled and neither player has " << endl;**

**cout << " three in a row, the game is a draw." << endl << endl;**

**}**

**void TTT::InitBoard(char boardAr[][COLUMNS]) // tic-tac-toe board – OUT**

**{**

**for(int i = 0; i < ROWS; i ++)**

**{**

**for (int j = 0; j < COLUMNS; j ++)**

**boardAr[i][j] = ' ';**

**}**

**}**

**void TTT::DisplayBoard(const char boardAr[][COLUMNS])**

**{**

**int i;**

**int j;**

**system("cls");**

**cout << setw(10) << "1" << setw(8) << "2" << setw(9) << "3\n";**

**for (i = 0; i < 3; i++)**

**{**

**cout << setw(7) << "[" << i+1 << "][1] | " << "[" << i+1;**

**cout << "][2] | " << "[" << i+1 << "][3]" << endl;**

**cout << setw(14) << "|" << setw(9) << "|" << endl;**

**for (j = 0; j < 3; j++)**

**{**

**switch(j)**

**{**

**case 0: cout << i + 1 << setw(9) << boardAr[i][j];**

**cout << setw(4) << "|";**

**break;**

**case 1: cout << setw(4) << boardAr[i][j];**

**cout << setw(5) << "|";**

**break;**

**case 2: cout << setw(4) << boardAr[i][j] << endl;**

**break;**

**default: cout <<"ERROR!\n\n";**

**}**

**}**

**cout << setw(14)<< "|" << setw(10) << "|\n";**

**if (i != 2)**

**{**

**cout << setw(32) << "--------------------------\n";**

**}**

**}**

**cout << endl << endl;**

**}**

**void TTT::GetAndCheckInp(char boardAr[][COLUMNS], char token, string player1, string player2)**

**{**

**bool retry;**

**int column;**

**int row;**

**switch(token)**

**{**

**case 'X': cout << player1 << " it's your turn." << endl;**

**break;**

**case 'O': cout << player2 << " it's your turn." << endl;**

**break;**

**}**

**cout << "Enter your move: ";**

**cin >> row;**

**cin >> column;**

**do**

**{**

**if(row<1 || row>ROWS || column<1 || column>COLUMNS)**

**retry= true;**

**else if(boardAr[row-1][column-1] != ' ')**

**retry=true;**

**else**

**retry = false;**

**if(retry == true)**

**{**

**cout<<"Invalid move, try again ";**

**cin>>row;**

**cin>>column;**

**}**

**else**

**boardAr[row-1][column-1]= token;**

**}while(retry == true);**

**}**

**char TTT::SwitchToken(char token)**

**{**

**// If token is currently X, switch it to O**

**if(token == 'X')**

**{**

**token = 'O';**

**}**

**// If token is currently O, switch it to X**

**else**

**{**

**token = 'X';**

**}**

**return token;**

**}**

**void TTT::GetPlayers(string& player1, // Player X’s name**

**string& player2) // Player O’x name**

**{**

**cin.ignore(100, '\n');**

**cout << "\nEnter the first player's name: ";**

**getline(cin, player1);**

**cout << "\nEnter the second player's name: ";**

**getline(cin, player2);**

**}**

**char TTT::CheckWin(char boardAr[][COLUMNS])**

**{**

**int r=0; // Number for rows**

**int c=0; // Number for columns**

**char W; // Winning**

**// Check 3 rows of X**

**if (boardAr[r][c] =='X' && boardAr[r][c+1] == 'X'**

**&& boardAr[r][c+2]== 'X' )**

**W = 'X';**

**else if (boardAr[r+1][c]== 'X' && boardAr[r+1][c+1]== 'X'**

**&& boardAr[r+1][c+2]== 'X' )**

**W = 'X';**

**else if (boardAr[r+2][c] == 'X'&& boardAr[r+2][c+1]== 'X'**

**&& boardAr[r+2][c+2]== 'X')**

**W = 'X';**

**// Check 3 columns of X**

**else if (boardAr[r][c] =='X' && boardAr[r+1][c]== 'X'**

**&& boardAr[r+2][c]== 'X')**

**W = 'X';**

**else if (boardAr[r][c+1]=='X' && boardAr[r+1][c+1]=='X'**

**&& boardAr[r+2][c+1]== 'X')**

**W = 'X';**

**else if (boardAr[r][c+2]=='X' && boardAr[r+1][c+2] =='X'**

**&& boardAr[r+2][c+2]== 'X')**

**W ='X';**

**// Check diagonally for X**

**else if (boardAr[r][c] =='X' && boardAr[r+1][c+1]=='X'**

**&& boardAr[r+2][c+2]=='X')**

**W = 'X';**

**else if (boardAr[r+2][c]=='X' && boardAr[r+1][c+1]=='X'**

**&& boardAr[r][c+2]== 'X')**

**W = 'X';**

**// Check 3 rows for O**

**else if (boardAr[r][c]=='O' && boardAr[r][c+1]=='O'**

**&& boardAr[r][c+2]== 'O' )**

**W = 'O';**

**else if (boardAr[r+1][c] =='O'&& boardAr[r+1][c+1]=='O'**

**&& boardAr[r+1][c+2]== 'O')**

**W = 'O';**

**else if (boardAr[r+2][c]=='O' && boardAr[r+2][c+1]=='O'**

**&& boardAr[r+2][c+2]== 'O')**

**W = 'O';**

**// Check 3 columns for O**

**else if (boardAr[r][c]=='O' && boardAr[r+1][c]=='O'**

**&& boardAr[r+2][c]== 'O')**

**W = 'O';**

**else if (boardAr[r][c+1]=='O' && boardAr[r+1][c+1]=='O'**

**&& boardAr[r+2][c+1]== 'O')**

**W = 'O';**

**else if (boardAr[r][c+2]=='O' && boardAr[r+1][c+2]=='O'**

**&& boardAr[r+2][c+2]== 'O')**

**W = 'O';**

**// Check diagonally for O**

**else if (boardAr[r][c]=='O' && boardAr[r+1][c+1]=='O'**

**&& boardAr[r+2][c+2]== 'O')**

**W = 'O';**

**else if (boardAr[r+2][c]=='O' && boardAr[r+1][c+1]=='O'**

**&& boardAr[r][c+2]== 'O')**

**W = 'O';**

**// Check if the game is a tie**

**else if (boardAr[r][c]!= ' ' && boardAr[r][c+1] != ' '**

**&&boardAr[r][c+2]!= ' ' && boardAr[r+1][c]!= ' '**

**&& boardAr[r+1][c+1]!= ' ' && boardAr[r+1][c+2]!= ' '**

**&& boardAr[r+2][c]!= ' ' && boardAr[r+2][c+1]!= ' '**

**&& boardAr[r+2][c+2]!= ' ')**

**W = 'T';**

**// Check if the game is still playing**

**else**

**W = 'N';**

**return W;**

**}**

**void TTT::OutputWinner(char whoWon, // represents the winner or a value**

**// indicating a tied game. - IN**

**string player1, // player X’s name - OUT**

**string player2) // player O’x name - OUT**

**{**

**// Switch on winner**

**switch (whoWon)**

**{**

**// Player X wins**

**case 'X': cout << player1 << " Is The Winner!" << endl;**

**break;**

**// Player O wins**

**case 'O': cout << player2 << " Is The Winner!" << endl;**

**break;**

**// Tie**

**case 'T': cout << "The game is a tie, you are all losers." << endl;**

**break;**

**}**

**}**

**//main.cpp**

**#include "Functions.h"**

**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 = 1; //randomize number from 1**

**int sizeRanT = 4; //to 4**

**int menuChoice; //choice of menu**

**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**

**char \*codeMakerL;**

**char \*codeBreaker; //Code breaker**

**char boardAr[ROWS][COLUMNS]; //Array that holds the rows and columns of the board**

**char gameStatus = 'N'; //Character that is used to determine the status of game**

**char play = 'Y'; //Holds user's decision when prompted if they want to play**

**char token; //Holds the character of the current player**

**char changeName = 'Y'; //Holds user's decision when prompted to change name**

**string player1; //Holds player 1's name**

**string player2; //Holds player 2's name**

**//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**

**}**

**play = 'N';**

**while (menuChoice != 4) //While choice is not 4(exit)**

**{**

**if(menuChoice == 3) //If user picks 3**

**{**

**//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**

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

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

**codeMakerL = new char[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;**

**}**

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

**{**

**switch(codeMaker[i])**

**{**

**case 1: codeMakerL[i] = 'B'; break;**

**case 2: codeMakerL[i] = 'R'; break;**

**case 3: codeMakerL[i] = 'Y'; break;**

**case 4: codeMakerL[i] = 'W'; break;**

**}**

**}**

**//Output sequence to binary file**

**outFile << "The sequence: ";**

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

**outFile << codeMakerL[i] << " ";**

**turnLeft = 11; //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-1 << " 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] == codeMakerL[i] && count == i)**

**{**

**markO++;**

**}**

**}**

**}**

**//Output how may numbers the user got right**

**cout << "You have " << markO << " colors right!";**

**cout << endl;**

**turn++; //Adding a turn**

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

**gamestatus = checkWinF(outFile, markO, turn, gamestatus,**

**turnLeft, codeMakerL, size);**

**}**

**//End the match. Prompt for continuation**

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

**cin >> play; //Input the decision**

**}**

**}**

**else if(menuChoice == 2) //If the user picks 2**

**{**

**TTT game;**

**//Calling on OutputInstruct function**

**game.OutputInstruct();**

**//Prompting players if they want to play**

**cout << "Do you want to start the game (Y/N): ";**

**cin >> play;**

**cout << endl;**

**//While loop that runs the actual tic tac toe game**

**while(play == 'Y' || play == 'y')**

**{**

**// Initializes board with blanks**

**game.InitBoard(boardAr);**

**// Sets token so that first player is 'X'**

**token = 'X';**

**// Loop that runs if changeName is 'y' or 'Y'**

**if(changeName == 'Y' || changeName == 'y')**

**// Calling on GetPlayers function**

**game.GetPlayers(player1, player2);**

**do**

**{**

**game.DisplayBoard(boardAr);**

**game.GetAndCheckInp(boardAr, token, player1, player2);**

**token = game.SwitchToken(token);**

**gameStatus = game.CheckWin(boardAr);**

**}while(gameStatus == 'N');**

**game.DisplayBoard(boardAr);**

**game.OutputWinner(gameStatus, player1, player2);**

**cout << "Do you want to play again? ";**

**cin >> play;**

**cout << endl;**

**if (play == 'Y' || play == 'y')**

**{**

**cout << "Would you like to change your names? ";**

**cin >> changeName;**

**}**

**}**

**}**

**//Other input choice other than 1, 2, 3 or 4**

**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;**

**}**