|  |  |
| --- | --- |
| C:\Users\HTThanh\Desktop\Logo-KHTN 2009.jpg | TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM KHOA CÔNG NGHỆ THÔNG TIN  MÔN: **KỸ THUẬT LẬP TRÌNH** |

GIANG ĐỨC NHẬT  
MA THANH NHI  
ĐINH LÊ GIA NHƯ  
NGUYỄN PHÁT KIM NHUNG

ĐỒ ÁN MÔN HỌC  
CỜ CARO

TP.HCM, ngày 28 tháng 04 năm 2023

|  |  |
| --- | --- |
| C:\Users\HTThanh\Desktop\Logo-KHTN 2009.jpg | TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM KHOA CÔNG NGHỆ THÔNG TIN  MÔN: **KỸ THUẬT LẬP TRÌNH** |

GIANG ĐỨC NHẬT - 22120252  
MA THANH NHI - 22120256  
ĐINH LÊ GIA NHƯ - 22120257  
NGUYỄN PHÁT KIM NHUNG- 22120259

ĐỒ ÁN MÔN HỌC  
CỜ CARO

**GIÁO VIÊN HƯỚNG DẪN**

**TS. TRƯƠNG TOÀN THỊNH**

TP.HCM, ngày 28 tháng 04 năm 2023

Mục lục

[1 Tổ chức thư mục trong chương trình 5](#_Toc135772981)

[2 Những định nghĩa, biến được thêm vào 5](#_Toc135772982)

[2.1 Định nghĩa 5](#_Toc135772983)

[2.2 Cấu trúc 6](#_Toc135772984)

[2.3 Biến toàn cục 6](#_Toc135772985)

[3 Lưu đồ thuật toán của chương trình 8](#_Toc135772986)

[4 Hàm main() 9](#_Toc135772987)

[5 Xử lí lưu, tải trò chơi 9](#_Toc135772989)

[5.1 int CheckExistedFile(string fileName) 9](#_Toc135772990)

[5.2 void SaveGame() 9](#_Toc135772991)

[5.3 void SaveMatchInfo(string matchName) 11](#_Toc135772992)

[5.4 void StartGame() 12](#_Toc135772993)

[5.5 void ResetData() 12](#_Toc135772994)

[5.6 void LoadGame(string matchName) 13](#_Toc135772995)

[5.7 void DrawExistedData() 14](#_Toc135772996)

[6 Nhận biết thắng/thua/hòa 14](#_Toc135772997)

[6.1 int CheckBoard(int pX, int pY) 14](#_Toc135772998)

[6.2 int TestBoard() 15](#_Toc135772999)

[6.3 int CheckWin(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2) 19](#_Toc135773000)

[6.4 void ResetToCheck(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2) 20](#_Toc135773001)

[7 Xử lí hiệu ứng thắng/thua/hòa 20](#_Toc135773002)

[7.1 void HighlightWin(\_POINT a[], int& n) 20](#_Toc135773003)

[7.2 int ProcessFinish(int pWhoWin) 20](#_Toc135773004)

[8 Xử lí giao diện màn hình khi chơi 24](#_Toc135773005)

[8.1 void ShowGame() 24](#_Toc135773006)

[8.2 void DrawBoard() 30](#_Toc135773007)

[9 Xử lý màn hình chính 33](#_Toc135773008)

[9.1 void ShowMenu() 33](#_Toc135773009)

[10 Điều chỉnh các thuộc tính của màn hình console 36](#_Toc135773010)

[10.1 void FixConsoleWindows() 36](#_Toc135773011)

[10.2 void SetColor(int backgoundColor, int textColor) 36](#_Toc135773012)

[11 Các hàm dùng để vẽ, in 36](#_Toc135773013)

[11.1 In hình ảnh, icon 36](#_Toc135773014)

[11.3 In tiêu đề các lựa chọn ở trang menu 37](#_Toc135773015)

[11.4 In tiêu đề trang 37](#_Toc135773016)

[11.5 Hiệu ứng 37](#_Toc135773017)

[11.6 Vẽ giao diện các trang 37](#_Toc135773018)

[12 Chức năng Undo 38](#_Toc135773019)

[13 Chế độ đánh với máy 39](#_Toc135773020)

[14 Âm thanh 41](#_Toc135773021)

[15 Xóa file 42](#_Toc135773022)

[16 Tìm file 43](#_Toc135773025)

[Tài liệu tham khảo 48](#_Toc135773026)

[Video demo 48](#_Toc135773027)

# Tổ chức thư mục trong chương trình

Chương trình được chia ra 6 nhóm.

* Nhóm header gồm header.h, header.cpp: chứa các định nghĩa, biến toàn cục.
* Nhóm view gồm view.h, view.cpp: chứa các hàm dùng để vẽ các hình, logo, bàn cờ…
* Nhóm page gồm page.h, page.cpp: chứa các hàm dùng để hiển thị các trang như trang chủ, trang chơi game, trang thông tin…
* Nhóm model gồm model.h, model.cpp: chứa các hàm chức năng như lưu game, tải game, kiểm tra bàn cờ…
* Nhóm control gồm control.h, control.cpp: chứa các hàm di chuyển, điều chỉnh thuộc tính màn hình.
* Nhóm main gồm main.cpp để chạy toàn bộ chương trình.

Các file .cpp chứa các hàm, còn .h sẽ chứa các prototype của các hàm ấy .

Thêm vào đó là file lưu tên “game\_files.txt” và các file lưu dữ liệu trận đấu như “Match1.txt”, “Match2.txt”.

# Những định nghĩa, biến được thêm vào

## Định nghĩa

Các định nghĩa về kích thước và vị trí làm mốc:

#define WIDTH 232

#define HEIGHT 50

#define BOARD\_SIZE 16

#define LEFT 28

#define TOP 8

#define CENTER\_X 102

#define CENTER\_Y 26

#define GOAL 5

#define MENU\_SIZE 5

Các định nghĩa về màu sắc để dễ sử dụng:

#define BLACK 0

#define BLUE 1

#define GREEN 2

#define AQUA 3

#define RED 4

#define PURPLE 5

#define YELLOW 6

#define WHITE 7

#define GRAY 8

#define LIGHT\_BLUE 9

#define LIGHT\_GREEN 10

#define LIGHT\_AQUA 11

#define LIGHT\_RED 12

#define LIGHT\_PURPLE 13

#define LIGHT\_YELLOW 14

#define BRIGHT\_WHITE 15

## Cấu trúc

struct \_POINT { int x, y, c; };

struct \_LIST {

int x, y;

string item;

};

struct EDGE {

int top = 0, bottom = 0, left = 0, right = 0;

};

struct TURN\_BOT {

int sz;

\_POINT led[2];

};

## Biến toàn cục

\_POINT \_A[BOARD\_SIZE][BOARD\_SIZE];

\_POINT \_MENU[MENU\_SIZE] = {

\_POINT{CENTER\_X,CENTER\_Y - 3,1},

\_POINT{CENTER\_X,CENTER\_Y + 1,2},

\_POINT{CENTER\_X,CENTER\_Y + 5,3},

\_POINT{CENTER\_X,CENTER\_Y + 9,4},

\_POINT{CENTER\_X,CENTER\_Y + 13,5},

};

\_POINT \_LAST\_POINT;

bool isLoad = 0;

string \_file\_name = "";

string PLAYER1, PLAYER2;

int SCORE\_X, SCORE\_O;

int \_MODEPLAY;

bool \_TURN;

int \_COMMAND;

int \_X, \_Y;

int MODE;

bool NEW\_GAME;

ofstream \_MATCH\_LIST\_FILE;

int MATCH\_LIST\_SIZE;

vector<\_LIST> \_MATCH\_LIST;

int SUB\_ML\_SIZE;

vector<\_LIST> \_SUB\_ML;

int SOUND;

bool \_EXIT;

bool isLoad = 0 : dùng để lưu vô trận cũ.

string \_file\_name = "" : dùng để lưu vô trận cũ.

string PLAYER1, PLAYER2 : dùng để lưu tên người chơi.

int SCORE\_X, SCORE\_O : dùng để lưu điểm.

\_POINT \_LAST\_POINT : dùng để lưu bước đánh cuối, được dùng trong lưu/tải file và chức năng undo.

int \_MODEPLAY : là chế độ chơi, là 1 nếu chơi với người, là 3 nếu chơi với máy, tương ứng với thứ tự các option trong menu. Biến này được dùng khi mở trang game và lưu/tải file.

int MODE : là chế độ trang, dùng cho các hàm di chuyển trong các trang vì mỗi trang có giới hạn di chuyển khác nhau.

bool NEW\_GAME : dùng khi lưu/tải game để vẽ dữ liệu trong bàn cờ.

ofstream \_MATCH\_LIST\_FILE : dùng để mở file danh sách các file trận đấu.

vector<\_LIST> \_MATCH\_LIST : dùng để in tên các file khi mở trang chứa file đã lưu.

int SUB\_ML\_SIZE : độ dài mảng chứa tên trong phần tìm file.

vector<\_LIST> \_SUB\_ML : lưu các file tìm thấy khi nhập tên file trong phần tìm file.

int SOUND : dùng để bật/tắt âm thanh.

bool \_EXIT : đặt vào đầu các hàm mở các trang, khi chọn EXIT ngoài Menu thì \_EXIT = 1, lúc này sẽ lần lượt thoát về các hàm bằng return, cuối cùng return được về main() và thoát chương trình.

# Lưu đồ thuật toán của chương trình

FixConsoleWindow

GetMatchListSize

ShowMenu

Yes

If == ESC

Exit

About

No

Option

ShowAbout

\_COMMAND

New

Continue

NEW\_GAME=1

NEW\_GAME=0

ShowGame

NEW\_GAME

Yes

ResetData

DrawExistedData

No

\_COMMAND

\_COMMAND

Yes

“W”,“A”

“S”,“D”

“U”

No

Yes

Undo

If == ESC

\_COMMAND

Move

No

Enter

“H”

ESC

Ask

No

CheckBoard

ShowHelp

Yes

“L”

SaveGame

TestBoard

No

Finish

Yes

ProcessFinish

Yes

AskContinue

No

# Hàm main()

int main() {

\_EXIT = 0;

FixConsoleWindows();

GetMatchListSize();

ShowMenu();

system("cls");

return 0;

# }

# Xử lí lưu, tải trò chơi

Dữ liệu của các trận đấu được lưu trong các file “name.txt”, sau đó tên của các file này được lưu trong 1 file là “game\_files.txt”. Các file “name.txt” chứa dữ liệu của bàn cờ và lượt đánh cuối cùng.

Khi vào trận, nếu là trận mới thì dữ liệu bàn cờ sẽ trở về ban đầu, nếu là mở file đã lưu thì các dấu X O đã đánh sẽ được vẽ lên bàn cờ và con trỏ sẽ nằm ở vị trí đánh cuối cùng mà file đã lưu.

## int CheckExistedFile(string fileName)

int CheckExistedFile(string fileName) {

//0:không mở được file -1:file không tồn tại 1:file đã tồn tại

string line;

ifstream file("game\_files.txt");

if (!file) {

cout << "Cannot open game file";

return 0;

}

while (!file.eof()) {

getline(file, line);

if (CheckSameString(line, fileName + ".txt"))

return 1;

}

return -1;

}

Hàm này sẽ kiểm tra xem tên file mà người dùng nhập đã tồn tại chưa.

## void SaveGame()

|  |  |
| --- | --- |
|  | void SaveGame() { |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | string matchName; |
|  | while (1) { |
|  | Input\_Data(matchName, 10, CENTER\_X + 2, CENTER\_Y + 2, InputFileName); |
|  | if (matchName != "")break; |
|  | } |
|  | \_MATCH\_LIST\_FILE.open("game\_files.txt", ios::app); |
|  | if (!\_MATCH\_LIST\_FILE) { |
|  | GotoXY(CENTER\_X + 2, CENTER\_Y + 3); |
|  | cout << "Cannot save"; |
|  | return; |
|  | } |
|  | while (CheckExistedFile(matchName) == 1) { |
|  | matchName = ""; |
|  | GotoXY(CENTER\_X + 2, CENTER\_Y + 3); |
|  | cout << " File existed!"; |
|  | GotoXY(CENTER\_X + 2, CENTER\_Y + 2); |
|  | for (int i = 0;i < 10;++i) |
|  | cout << " "; |
|  | GotoXY(CENTER\_X + 2, CENTER\_Y + 2); |
|  | while (1) { |
|  | Input\_Data(matchName, 10, CENTER\_X + 2, CENTER\_Y + 2, InputFileName); |
|  | if (matchName != "")break; |
|  | } |
|  | } |
|  | if(MATCH\_LIST\_SIZE==0)\_MATCH\_LIST\_FILE<< matchName + ".txt"; |
|  | else { |
|  | \_MATCH\_LIST\_FILE.seekg(0, ios::end); |
|  | \_MATCH\_LIST\_FILE << endl << matchName + ".txt"; |
|  | } |
|  | SaveMatchInfo(matchName); |
|  | \_MATCH\_LIST\_FILE.close(); |
|  | GetMatchListSize(); |
|  | \_MATCH\_LIST\_FILE.close(); |
|  | } |

Hàm này nhận tên file mà người dùng nhập sau đó kiểm tra nó đúng định dạng chưa rồi lưu tên đó vào “game\_files.txt” và lưu dữ liệu của bàn cờ hiện tại vào file tên tương ứng.

Các kí tự được kiểm tra cùng lúc nhập vào. Nếu kí tự hợp lệ thì sẽ được thêm vào matchName và cout ra màn hình, nếu không thì bỏ qua kí tự đó.

Dòng 4 - 7: nhập tên file và kiểm tra theo quy tắc chỉ gồm các chữ cái, số và các kí tự như “\_”, “-“.

Dòng 8: mở file lưu tên các file dữ liệu trận đấu.

Dòng 14: kiểm tra xem tên file đã nhập đã tồn tại chưa.

Dòng 30: ghép tên file với đuôi “.txt” và lưu vào danh sách file.

Dòng 32: lưu dữ liệu trận đấu.

## void SaveMatchInfo(string matchName)

void SaveMatchInfo(string matchName) {

matchName += ".txt";

ofstream matchFile(matchName);

if (!matchFile) {

cout << " Cannot save!";

return;

}

cout << " Saved";

for (int i = 0;i < BOARD\_SIZE;i++) {

for (int j = 0;j < BOARD\_SIZE;j++) {

matchFile << \_A[i][j].c << endl;

}

}

matchFile << \_LAST\_POINT.x << endl << \_LAST\_POINT.y << endl;

if (\_MODEPLAY == \_MENU[2].c)matchFile << 0 << endl;

else matchFile << \_LAST\_POINT.c << endl;

matchFile << \_MODEPLAY << endl

<< PLAYER1 << endl

<< PLAYER2 << endl

<< SCORE\_X << endl

<< SCORE\_O;

matchFile.close();

}

Hàm này nhận vào tên rồi tạo file .txt tương ứng, sau đó lần lượt chạy hết ma trận bàn cờ để lưu dữ liệu ô đánh, cuối cùng sẽ lưu bước đi cuối và chế độ chơi của trận đó.

## void StartGame()

void StartGame() {

system("cls");

DrawBoard();

ShowCursor(true);

if (NEW\_GAME == 1 || \_LAST\_POINT.x==0&&\_LAST\_POINT.y==0)

ResetData();

else

DrawExistedData();

}

Đầu tiên sẽ vẽ ra khung bàn cờ. Có 1 biến toàn cục là NEW\_GAME, khi ở Menu thì NEW\_GAME = 1, khi vào Continue và chọn mở 1 file nào đó thì NEW\_GAME = 0. Hàm StartGame() sẽ kiểm tra biến này, nếu NEW\_GAME = 1 tức là tạo trận mới, dữ liệu bàn cờ trở về ban đầu, nếu NEW\_GAME = 0 tức là mở lại trận cũ, sẽ vẽ dữ liệu trong file lên bàn cờ.

## void ResetData()

void ResetData() {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

\_A[i][j].x = 4 \* j + LEFT + 2;

\_A[i][j].y = 2 \* i + TOP + 1;

\_A[i][j].c = 0;

}

}

\_TURN = true; \_COMMAND = -1;

\_X = \_A[0][0].x; \_Y = \_A[0][0].y;

GotoXY(\_X, \_Y);

}

Hàm này được gọi khi tạo trận đấu mới, dùng để duyệt hết ma trận bàn cờ và đặt các giá trị về 0, sau đó đặt con trỏ về vị trí đầu tiên của bàn cờ.

## void LoadGame(string matchName)

void LoadGame(string matchName) {

int i = 0, j = 0, m = 0, value = 0;

int numbers[BOARD\_SIZE \* BOARD\_SIZE + 4];

ifstream matchFile(matchName);

if (!matchFile) {

cout << "Cannot open match file";

return;

}

while (m< BOARD\_SIZE \* BOARD\_SIZE + 4) {

matchFile >> numbers[m++];

}

m = 0;

for (int i = 0;i < BOARD\_SIZE;i++)

for (int j = 0;j < BOARD\_SIZE;j++) {

\_A[i][j].x = 4 \* j + LEFT + 2;

\_A[i][j].y = 2 \* i + TOP + 1;

\_A[i][j].c = numbers[m];

++m;

}

\_LAST\_POINT.x = numbers[m];++m;

\_LAST\_POINT.y = numbers[m];++m;

\_LAST\_POINT.c = numbers[m];++m;

\_MODEPLAY = numbers[m];

matchFile >> PLAYER1;

matchFile >> PLAYER2;

matchFile >> SCORE\_X;

matchFile >> SCORE\_O;

if (\_LAST\_POINT.c == -1)

\_TURN = false;

else \_TURN = true;

\_COMMAND = -1;

}

Hàm này nhận vào tên file sau đó tải dữ liệu từ file vào ma trận bàn cờ và bước đánh cuối.

Đầu tiên tạo 1 mảng int numbers[] với độ dài bằng tổng số ô trong ma trận và thêm + 4 giá trị của bước đi cuối và chế độ chơi. Sau đó duyệt ma trận bàn cờ để gán giá trị vào từ mảng numbers[], gán giá trị của bước đi cuối vào biến toàn cục \_LAST\_POINT, gán chế độ trong file vào \_MODEPLAY và gán các thông tin trận đấu như tên 2 người chơi và tỉ số.

## void DrawExistedData()

void DrawExistedData() {

for (int i = 0;i < BOARD\_SIZE;i++)

for (int j = 0;j < BOARD\_SIZE;j++) {

if (\_A[i][j].c != 0) {

GotoXY(\_A[i][j].x, \_A[i][j].y);

switch (\_A[i][j].c) {

case -1: {

SetColor(BRIGHT\_WHITE, RED);

cout << "X"; break; }

case 1: {

SetColor(BRIGHT\_WHITE, GREEN);

cout << "O"; break; }

}

}

}

\_X = \_A[0][0].x; \_Y = \_A[0][0].y;

GotoXY(\_X, \_Y);

if (\_LAST\_POINT.x != 0 && \_LAST\_POINT.y != 0) {

\_X = \_LAST\_POINT.x; \_Y = \_LAST\_POINT.y;

GotoXY(\_X, \_Y);

}

}

Hàm này được gọi khi mở file trận đấu đã lưu, dùng để đánh dấu X O lên bàn cờ.

Duyệt từng ô trên bàn cờ, nếu ở đã được đánh thì in ra X hoặc O. Sau đó đưa con trỏ về ô đầu tiên nếu là trận mới, hoặc đưa con trỏ về điểm đã đánh trước đó nếu là mở lại file.

# Nhận biết thắng/thua/hòa

## int CheckBoard(int pX, int pY)

int CheckBoard(int pX, int pY) {

//Kiểm tra vị trí hiện tại có trống không

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

if (\_A[i][j].x == pX && \_A[i][j].y == pY && \_A[i][j].c == 0) {

if (\_TURN == true) \_A[i][j].c = -1;

else \_A[i][j].c = 1;

\_LAST\_POINT = { pX,pY,\_A[i][j].c };

return \_A[i][j].c;

}

}

}

return 0;

}

Hàm này kiểm tra vị trí hiện tại có trống không, nếu trống thì tùy vào lượt đánh mà gán giá trị c tại ô đó, đồng thời lưu lại vào bước đi cuối.

## int TestBoard()

|  |  |
| --- | --- |
|  | int TestBoard() { |
|  | //Kiểm tra thắng thua, người 1 hay \_Turn=true thắng thì trả về -1 người còn lại thắng trả về 1, |
|  | //hòa nhau trả về 0 còn chưa phân định ai thắng trả về 2 |
|  | \_POINT led1 = { 0,0,2 }, led2 = { 0,0,2 }; |
|  | int row = 0, col = 0; |
|  | int result = 2, n = 0, m = 0; |
|  |  |
|  | //Mảng lưu tọa độ đường chiến thắng |
|  | \_POINT a[9]; |
|  | \_POINT p = XYinMatrix(\_X, \_Y, row, col); |
|  | TURN\_BOT tb[4]; |
|  | a[n++] = p; |
|  |  |
|  | //Kiểm tra trên dòng |
|  | //Các đèn led tín hieu de kiem tra luat chan hai dau |
|  | for (int i = col - 1; i >= 0 && i < BOARD\_SIZE; i--) { |
|  | if (\_A[row][i].c == \_A[row][col].c) |
|  | a[n++] = \_A[row][i]; |
|  | else { |
|  | led1 = \_A[row][i]; |
|  | break; |
|  | } |
|  | } |
|  | for (int i = col + 1; i >= 0 && i < BOARD\_SIZE; i++) { |
|  | if (\_A[row][i].c == \_A[row][col].c) |
|  | a[n++] = \_A[row][i]; |
|  | else { |
|  | led2 = \_A[row][i]; |
|  | break; |
|  | } |
|  | } |
|  | if (\_MODEPLAY == \_MENU[2].c && \_TURN == true) |
|  | tb[m++] = { n,{led1,led2} }; |
|  | result = CheckWin(a, n, led1, led2); if (result != 2)return result; |
|  |  |
|  | //Kiểm tra trên cột |
|  | for (int i = row + 1; i >= 0 && i < BOARD\_SIZE; i++) { |
|  | if (\_A[i][col].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][col]; |
|  | } |
|  | else { |
|  | led1 = \_A[i][col]; |
|  | break; |
|  | } |
|  |  |
|  | } |
|  | for (int i = row - 1; i >= 0 && i < BOARD\_SIZE; i--) { |
|  | if (\_A[i][col].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][col]; |
|  | } |
|  | else { |
|  | led2 = \_A[i][col]; |
|  | break; |
|  | } |
|  | } |
|  | if (\_MODEPLAY == \_MENU[2].c && \_TURN == true) |
|  | tb[m++] = { n,{led1,led2} }; |
|  | result = CheckWin(a, n, led1, led2); if (result != 2)return result; |
|  |  |
|  | //Kiểm tra đường chéo chính |
|  | for (int i = row - 1, j = col - 1; i >= 0 && i < BOARD\_SIZE, j >= 0 && j < BOARD\_SIZE; i--, j--) { |
|  | if (\_A[i][j].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][j]; |
|  | } |
|  | else { |
|  | led1 = \_A[i][j]; |
|  | break; |
|  | } |
|  | } |
|  | for (int i = row + 1, j = col + 1; i >= 0 && i < BOARD\_SIZE, j >= 0 && j < BOARD\_SIZE; i++, j++) { |
|  | if (\_A[i][j].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][j]; |
|  | } |
|  | else { |
|  | led2 = \_A[i][j]; |
|  | break; |
|  | } |
|  | } |
|  | if (\_MODEPLAY == \_MENU[2].c && \_TURN == true) |
|  | tb[m++] = { n,{led1,led2} }; |
|  | result = CheckWin(a, n, led1, led2); if (result != 2)return result; |
|  |  |
|  | //Kiểm tra đường chéo phụ |
|  | for (int i = row + 1, j = col - 1; i >= 0 && i < BOARD\_SIZE, j >= 0 && j < BOARD\_SIZE; i++, j--) { |
|  | if (\_A[i][j].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][j]; |
|  | } |
|  | else { |
|  | led1 = \_A[i][j]; |
|  | break; |
|  | } |
|  |  |
|  | } |
|  | for (int i = row - 1, j = col + 1; i >= 0 && i < BOARD\_SIZE, j >= 0 && j < BOARD\_SIZE; i--, j++) { |
|  | if (\_A[i][j].c == \_A[row][col].c) { |
|  | a[n++] = \_A[i][j]; |
|  | } |
|  | else { |
|  | led2 = \_A[i][j]; |
|  | break; |
|  | } |
|  | } |
|  | if (\_MODEPLAY == \_MENU[2].c && \_TURN == true) { |
|  | tb[m++] = { n,{led1,led2} }; |
|  | PVC(key, tb); |
|  | } |
|  | result = CheckWin(a, n, led1, led2); if (result != 2)return result; |
|  | for (int i = 0; i < 4; i++) tb[i] = { 0,{{0,0},{0,0}} }; |
|  |  |
|  | int flag = 0; |
|  | for (int i = 0; i < BOARD\_SIZE; i++) |
|  | for (int j = 0; j < BOARD\_SIZE; j++) |
|  | if (\_A[i][j].c == 0)flag = 1; |
|  | if (flag == 0) |
|  | return 0; |
|  | return 2; |
|  | } |

Hàm này xét 8 đường ngang dọc chéo xem khi đánh 1 nước thì xét bắt đầu đếm xem có bao nhiêu nước đánh đó xung quanh.

Dòng 16 – 102: kiểm tra trên các dòng và đếm n số dấu kế nhau.

Các dòng result = CheckWin(a, n, led1, led2); if (result != 2)return result; để kiểm tra n đủ hay dư chưa.

## int CheckWin(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2)

int CheckWin(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2) {

if ((n == 5 && (led1.c == 0 || led2.c == 0))||n>5) {

\_PlaySound(6);

HighlightWin(a, n);

\_TURN == true ? SCORE\_X++ : SCORE\_O++;

if (SCORE\_X == 10 || SCORE\_O == 10)

ResetScore();

return(\_TURN == true ? 1 : -1);

}

ResetToCheck(a, n, led1, led2);

return 2;

}

Hàm này đếm xem n đủ 5 chưa, đủ rồi thì highlight nó lên rồi trả về người thắng thua hay chưa phân định vào hàm ProcessFinish(). Thêm vào đó là tăng điểm của người chơi đã thắng và lưu vào biến toàn cục SCORE\_X hoặc SCORE\_O.

## void ResetToCheck(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2)

void ResetToCheck(\_POINT a[], int& n, \_POINT& led1, \_POINT& led2) {

a[n] = { 0 };

n = 1;

led1 = { 0,0,2 }; led2 = { 0,0,2};

}

Hàm này reset lại các giá trị dùng trong hàm int CheckWin(\_POINT a[], int& n, POINT& led1, \_POINT& led2)

# Xử lí hiệu ứng thắng/thua/hòa

## void HighlightWin(\_POINT a[], int& n)

void HighlightWin(\_POINT a[], int& n) {

for (int i = 1; i<10; i++) {

int color = i; Sleep(600);

for (int j = 0; j < n; j++) {

GotoXY(a[j].x, a[j].y);

if(i==9)SetColor(LIGHT\_AQUA,PURPLE);

else SetColor(BRIGHT\_WHITE, color++);

if (\_TURN == true)cout << "X";

else cout << "O";

}

}

SetColor(BRIGHT\_WHITE, LIGHT\_AQUA);

Sleep(2500);

}

Hàm này tạo hiệu ứng làm nổi bật đường đánh thắng.

## int ProcessFinish(int pWhoWin)

int ProcessFinish(int pWhoWin) {

PrintRectangle2lines(0, 1, WIDTH - 1, HEIGHT - 1);

int top = 5, left = CENTER\_X - 30;

int bg\_color = BRIGHT\_WHITE, text\_color = LIGHT\_AQUA;

ShowCursor(false);

switch (pWhoWin) {

case 1: {

system("cls");

PrintRectangle2lines(0, 1, WIDTH - 1, HEIGHT - 1);

int old\_mode = \_setmode(\_fileno(stdout), \_O\_WTEXT);

SetColor(BRIGHT\_WHITE, BLACK);

wstring logo[12] = {

L" ██████╗ ██╗ █████╗ ██╗ ██╗███████╗██████╗ ",

L" ██╔══██╗██║ ██╔══██╗╚██╗ ██╔╝██╔════╝██╔══██╗ ",

L" ██████╔╝██║ ███████║ ╚████╔╝ █████╗ ██████╔╝ ",

L" ██╔═══╝ ██║ ██╔══██║ ╚██╔╝ ██╔══╝ ██╔══██╗ ",

L" ██║ ███████╗██║ ██║ ██║ ███████╗██║ ██║ ",

L" ╚═╝ ╚══════╝╚═╝ ╚═╝ ╚═╝ ╚══════╝╚═╝ ╚═╝ ",

L" ██╗ ██╗   ██╗ ██╗██╗███╗ ██╗ ",

L" ╚██╗██╔╝   ██║ ██╗ ██║██║████╗ ██║ ",

L" ╚███╔╝    ╚██╗████╗██╔╝██║██╔██╗██║ ",

L" ██╔██╗    ████╔═████║ ██║██║╚████║ ",

L" ██╔╝╚██╗   ╚██╔╝ ╚██╔╝ ██║██║ ╚███║ ",

L" ╚═╝ ╚═╝   ╚═╝ ╚═╝ ╚═╝╚═╝ ╚══╝ " };

PrintHelicopter(55, 15);

PrintHelicopter(155, 5);

int count = 0;

int color[] = { RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE, RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE,RED };

while (count++ < 8) {

SetColor(BRIGHT\_WHITE, color[count]);

for (int i = 0; i < 6; i++) {

GotoXY(left, top + i);

wcout << logo[i];

GotoXY(left, top + 8 + i);

wcout << logo[6 + i];

Sleep(100);

}

Sleep(250);

}

SetColor(BRIGHT\_WHITE, BLACK);

int current\_mode = \_setmode(\_fileno(stdout), old\_mode);

break;

}

case -1: {

system("cls");

PrintRectangle2lines(0, 1, WIDTH - 1, HEIGHT - 1);

int old\_mode = \_setmode(\_fileno(stdout), \_O\_WTEXT);

SetColor(bg\_color, text\_color);

wstring logo[12] = {

L" ██████╗ ██╗ █████╗ ██╗ ██╗███████╗██████╗ ",

L" ██╔══██╗██║ ██╔══██╗╚██╗ ██╔╝██╔════╝██╔══██╗ ",

L" ██████╔╝██║ ███████║ ╚████╔╝ █████╗ ██████╔╝ ",

L" ██╔═══╝ ██║ ██╔══██║ ╚██╔╝ ██╔══╝ ██╔══██╗ ",

L" ██║ ███████╗██║ ██║ ██║ ███████╗██║ ██║ ",

L" ╚═╝ ╚══════╝╚═╝ ╚═╝ ╚═╝ ╚══════╝╚═╝ ╚═╝ ",

L" █████╗    ██╗ ██╗██╗███╗ ██╗ ",

L" ██╔══██╗   ██║ ██╗ ██║██║████╗ ██║ ",

L" ██║ ██║   ╚██╗████╗██╔╝██║██╔██╗██║ ",

L" ██║ ██║   ████╔═████║ ██║██║╚████║ ",

L" ╚█████╔╝   ╚██╔╝ ╚██╔╝ ██║██║ ╚███║ ",

L" ╚════╝    ╚═╝ ╚═╝ ╚═╝╚═╝ ╚══╝ " };

PrintHelicopter(55, 15);

PrintHelicopter(155, 5);

int count = 0;

int color[] = { RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE, RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE,RED };

while (count++ < 8) {

SetColor(BRIGHT\_WHITE, color[count]);

for (int i = 0; i < 6; i++) {

GotoXY(left, top + i);

wcout << logo[i];

GotoXY(left, top + 8 + i);

wcout << logo[6 + i];

Sleep(100);

}

Sleep(250);

}

SetColor(BRIGHT\_WHITE, BLACK);

int current\_mode = \_setmode(\_fileno(stdout), old\_mode);

break;

}

case 0: {

system("cls");

PrintRectangle2lines(0, 1, WIDTH - 1, HEIGHT - 1);

int old\_mode = \_setmode(\_fileno(stdout), \_O\_WTEXT);

SetColor(bg\_color, text\_color);

wstring logo[12] = {

L" ██╗ ██╗ █████╗    ",

L" ╚██╗██╔╝██╔══██╗   ",

L" ╚███╔╝ ██║ ██║   ",

L" ██╔██╗ ██║ ██║   ",

L" ██╔╝╚██╗╚█████╔╝   ",

L" ╚═╝ ╚═╝ ╚════╝    ",

L"██████╗ █████╗ ██████╗ ██╗ ██╗██╗██╗██╗",

L"██╔══██╗██╔══██╗██╔══██╗ ██║ ██╗ ██║██║██║██║",

L"██║ ██║███████║██████╔╝ ╚██╗████╗██╔╝██║██║██║",

L"██║ ██║██╔══██║██╔══██╗ ████╔═████║ ╚═╝╚═╝╚═╝",

L"██████╔╝██║ ██║██║ ██║ ╚██╔╝ ╚██╔╝ ██╗██╗██╗",

L"╚═════╝ ╚═╝ ╚═╝╚═╝ ╚═╝ ╚═╝ ╚═╝ ╚═╝╚═╝╚═╝", };

PrintHelicopter(55, 15);

PrintHelicopter(155, 5);

int count = 0;

int color[] = { RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE, RED,BRIGHT\_WHITE,GREEN,BRIGHT\_WHITE,RED };

while (count++ < 8) {

SetColor(BRIGHT\_WHITE, color[count]);

for (int i = 0; i < 6; i++) {

GotoXY(left, top + i);

wcout << logo[i];

GotoXY(left, top + 8 + i);

wcout << logo[6 + i];

Sleep(100);

}

Sleep(250);

}

SetColor(BRIGHT\_WHITE, BLACK);

int current\_mode = \_setmode(\_fileno(stdout), old\_mode);

break;

}

case 2:

\_TURN = !\_TURN;

}

return pWhoWin;

}

Hàm này kiểm tra xem ai thắng và in hiệu ứng người chiến thắng hoặc hòa.

# Xử lí giao diện màn hình khi chơi

## void ShowGame()

|  |  |
| --- | --- |
|  | void ShowGame() { |
|  | if (\_EXIT) return; |
|  | else { |
|  | \_PlaySound(4); |
|  | StartGame(); |
|  | int row\_console = 0, column\_console = 0, row = 0, col |
|  | = 0, flag = 0, res = 0; |
|  | KEY\_EVENT\_RECORD keyevent; |
|  | EDGE bien = { TOP + 1,TOP + BOARD\_SIZE \* 2 - 1,LEFT + |
|  | 2,LEFT + BOARD\_SIZE \* 4 - 2 }; |
|  | bool validEnter = true; |
|  | while (1) { |
|  | ShowCursor(true); |
|  | ReadInputKey(keyevent); |
|  | if (keyevent.bKeyDown) { |
|  | KeyMove(&\_X, &\_Y, 4, 2, bien, keyevent); |
|  | switch (keyevent.wVirtualKeyCode) { |
|  | case (VK\_ESCAPE): case (0x4C): { |
|  | \_PlaySound(3); |
|  | ShowAsk(keyevent.wVirtualKeyCode); |
|  | if (\_EXIT) return; |
|  | break; |
|  | } |
|  | case (0x48): { //H: Help |
|  | \_PlaySound(4); |
|  | ShowHelp(); |
|  | if (\_EXIT) return; |
|  | break; } |
|  | case (VK\_RETURN): { |
|  | \_PlaySound(1); |
|  | switch (CheckBoard(\_X, \_Y)) { |
|  | case -1: { |
|  | SetColor(BRIGHT\_WHITE, RED); |
|  | cout << "X"; |
|  | //Hiệu ứng đổi lượt |
|  | SetColor(BRIGHT\_WHITE, GRAY); |
|  | PrintX(TOP + 4, LEFT + BOARD\_SIZE \* |
|  | 4 + 12, 2); |
|  | SetColor(BRIGHT\_WHITE, GREEN); |
|  | PrintO(TOP + ((BOARD\_SIZE \* 2 - (4 |
|  | \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | GotoXY(\_LAST\_POINT.x, |
|  | \_LAST\_POINT.y); |
|  | if(\_MODEPLAY==\_MENU[2].c)Sleep(200); |
|  | break; |
|  | } |
|  | case 1: { |
|  | if (\_MODEPLAY == \_MENU[0].c) { |
|  | SetColor(BRIGHT\_WHITE, GREEN); |
|  | cout << "O"; |
|  | //Hiệu ứng đổi lượt |
|  | SetColor(BRIGHT\_WHITE, RED); |
|  | PrintX(TOP + 4, LEFT + BOARD\_SIZE \* |
|  | 4 + 12, 2); |
|  | SetColor(BRIGHT\_WHITE, GRAY); |
|  | PrintO(TOP + ((BOARD\_SIZE \* 2 - (4 |
|  | \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | GotoXY(\_LAST\_POINT.x, |
|  | \_LAST\_POINT.y); } |
|  | break; |
|  | } |
|  | case 0: validEnter = false; |
|  | } |
|  | if (validEnter == true) { |
|  | switch |
|  | (ProcessFinish(TestBoard(keyevent))) { |
|  | case -1:case 1: { |
|  | \_PlaySound(4); |
|  | flag = AskContinue(); |
|  | if (\_EXIT) return; |
|  | break; |
|  | } |
|  | case 0: { |
|  | flag = AskContinue(); |
|  | if (\_EXIT) return; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | break; |
|  | } |
|  | case (0x55): { |
|  | if (\_MODEPLAY == \_MENU[0].c) { |
|  | \_PlaySound(4); |
|  | if (\_LAST\_POINT.c != 0) { //Xét |
|  | trường hợp Undo 2 lần trên 1 ô |
|  | //Hiệu ứng đổi lượt |
|  | if (\_TURN == true) { |
|  | SetColor(BRIGHT\_WHITE, |
|  | GRAY); |
|  | PrintX(TOP + 4, LEFT + |
|  | BOARD\_SIZE \* 4 + 12, 2); |
|  | SetColor(BRIGHT\_WHITE, |
|  | GREEN); |
|  | PrintO(TOP + ((BOARD\_SIZE |
|  | \* 2 - (4 \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | GotoXY(\_LAST\_POINT.x, |
|  | \_LAST\_POINT.y); |
|  | } |
|  | else if (\_TURN == false) { |
|  | SetColor(BRIGHT\_WHITE, |
|  | RED); |
|  | PrintX(TOP + 4, LEFT + |
|  | BOARD\_SIZE \* 4 + 12, 2); |
|  | SetColor(BRIGHT\_WHITE, |
|  | GRAY); |
|  | PrintO(TOP + ((BOARD\_SIZE |
|  | \* 2 - (4 \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | GotoXY(\_LAST\_POINT.x, |
|  | \_LAST\_POINT.y); |
|  | } |
|  |  |
|  | \_X = \_LAST\_POINT.x; |
|  | \_Y = \_LAST\_POINT.y; |
|  | GotoXY(\_X, \_Y); |
|  | cout << char(32); |
|  | GotoXY(\_X, \_Y); |
|  | XYinMatrix(\_X, \_Y, row, col); |
|  | \_A[row][col].c = 0; |
|  | \_TURN = !\_TURN; |
|  | \_LAST\_POINT.c = 0; |
|  | } |
|  | } |
|  | break; |
|  | } |
|  | case (0x4D): { |
|  | if (SOUND == 0) |
|  | SOUND = 1; |
|  | else |
|  | SOUND = 0; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | keyevent.bKeyDown = false; |
|  | validEnter = true; |
|  | if (flag == 2)break; |
|  | } |
|  | ShowLoadingPage(); |
|  | ShowMenu(); |
|  | if (\_EXIT) return; |
|  | } |
|  | } |

Hàm này mở ra trang chơi game. Đầu tiên sẽ khởi động bằng hàm StartGame(). Sau đó chạy vòng lặp để nhận phím từ người chơi.

Dòng 18 – 23 kiểm tra nếu là phím “ESC” hoặc “L” thì sẽ chạy hàm hỏi muốn thoát game hoặc lưu file.

Dòng 24 – 28 kiểm tra nếu là phím “H” thì sẽ hiện bảng hướng dẫn chơi.

Dòng 29 – 81 kiểm tra nếu là phím “Enter” thì sẽ kiểm tra bàn cờ để in dấu X O và xử lí khi có kết quả thắng hay hòa, sau đó sẽ hỏi có muốn chơi tiếp không.

Dòng 82 – 125 kiểm tra nếu là phím “U” – Undo, ghi đè “ “ lên dấu vừa đánh, gán giá trị .c = 0 tại ô đó và đánh lại.

Trong đó dòng 36 – 41, 52 – 59 và 88 – 108 để tạo hiệu ứng đổi lượt. Đến lượt nào thì in có màu dấu đó và in màu xám dấu còn lại.

## void DrawBoard()

void DrawBoard() {

system("color F1");

//In logo

SetColor(BRIGHT\_WHITE, BLACK);

PrintInMatch(3, CENTER\_X - 20);

PrintRectangle2lines(TOP, LEFT + BOARD\_SIZE \* 4 + 34, 62, BOARD\_SIZE - 1);

PrintRectangle2lines(TOP + BOARD\_SIZE + 1, LEFT + BOARD\_SIZE \* 4 + 34, 62, BOARD\_SIZE - 1);

SetColor(BRIGHT\_WHITE, RED);

PrintX(TOP + 2, LEFT + BOARD\_SIZE \* 4 + 47, 2);

SetColor(BRIGHT\_WHITE, GREEN);

PrintO(TOP + 2, LEFT + BOARD\_SIZE \* 4 + 77, 2);

DesignBoard();

RemoveScore(TOP + 8, LEFT + BOARD\_SIZE \* 4 + 46);

RemoveScore(TOP + 8, LEFT + BOARD\_SIZE \* 4 + 76);

DrawScore(SCORE\_X, TOP + 8, LEFT + BOARD\_SIZE \* 4 + 47);

DrawScore(SCORE\_O, TOP + 8, LEFT + BOARD\_SIZE \* 4 + 77);

Sleep(500);

GotoXY(LEFT, TOP); cout << char(201);

GotoXY(LEFT, TOP + BOARD\_SIZE \* 2); cout << char(200);

for (int j = 1; j < BOARD\_SIZE \* 4 + 30; j++) {

Sleep(20);

GotoXY(LEFT + j, TOP);

if (j % 4 == 0 && j < BOARD\_SIZE \* 4)cout << char(209);

else cout << char(205);

GotoXY(LEFT + j, TOP + BOARD\_SIZE \* 2);

if (j % 4 == 0 && j < BOARD\_SIZE \* 4)cout << char(207);

else cout << char(205);

if (j > BOARD\_SIZE \* 4) {

GotoXY(LEFT + j, TOP + ((BOARD\_SIZE \* 2) / 2));

cout << char(205);

}

}

GotoXY(LEFT + BOARD\_SIZE \* 4 + 30, TOP); cout << char(187);

GotoXY(LEFT + BOARD\_SIZE \* 4 + 30, TOP + BOARD\_SIZE \* 2); cout << char(188);

GotoXY(LEFT + BOARD\_SIZE \* 4, TOP); cout << char(203);

GotoXY(LEFT + BOARD\_SIZE \* 4, TOP + BOARD\_SIZE \* 2); cout << char(202);

//Vẽ đường viền trái phải

for (int j = 1; j < BOARD\_SIZE \* 2; j++) {

Sleep(20);

GotoXY(LEFT, j + TOP);

if (j % 2 == 0)cout << char(199);

else cout << char(186);

GotoXY(LEFT + BOARD\_SIZE \* 4, j + TOP);

if (j % 2 == 0)cout << char(182);

else cout << char(186);

GotoXY(LEFT + BOARD\_SIZE \* 4 + 30, j + TOP);

cout << char(186);

}

//Vẽ các cột, dòng

for (int i = 1; i < BOARD\_SIZE; i++)

for (int j = 1; j < BOARD\_SIZE \* 4; j++) {

if (j < BOARD\_SIZE \* 2 && j % 2 == 1) {

GotoXY(LEFT + i \* 4, j + TOP);

cout << char(179);

}

if (j % 4 != 0) {

GotoXY(LEFT + j, 2 \* i + TOP);

cout << char(196);

}

}

//Sleep(10);

for (int i = 1; i < BOARD\_SIZE; i++)

for (int j = 1; j < BOARD\_SIZE \* 4; j++) {

GotoXY(LEFT + j, 2 \* i + TOP);

if (j % 4 == 0)cout << char(197);

}

//Vẽ hiệu ứng đổi lượt X O

if (\_TURN == 1 || NEW\_GAME == 1) {

SetColor(BRIGHT\_WHITE, RED);

PrintX(TOP + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2);

SetColor(BRIGHT\_WHITE, GRAY);

PrintO(TOP + ((BOARD\_SIZE \* 2) / 2) + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2);

}

else if (\_TURN == 0) {

SetColor(BRIGHT\_WHITE, GRAY);

PrintX(TOP + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2);

SetColor(BRIGHT\_WHITE, GREEN);

PrintO(TOP + ((BOARD\_SIZE \* 2)/ 2) + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2);

}

SetColor(LIGHT\_AQUA, BLACK);

if (\_MODEPLAY == \_MENU[2].c) {

GotoXY(LEFT, TOP + BOARD\_SIZE \* 2 + 1);

cout << "UNAVAILABLE !!!";

}

GotoXY(LEFT + BOARD\_SIZE \* 4 + 10, TOP + 2); SetColor(BRIGHT\_WHITE, RED); cout << " PLAYER 1 : ";

int n = ceil((30 - PLAYER1.size() - 2) / 2 + 0.5);

GotoXY(LEFT + BOARD\_SIZE \* 4 + n, TOP + 3); cout << PLAYER1;

GotoXY(LEFT + BOARD\_SIZE \* 4 + 10, TOP + ((BOARD\_SIZE \* 2) / 2) + 2); SetColor(BRIGHT\_WHITE, GREEN);

cout << " PLAYER 2 : ";

if (\_MODEPLAY == \_MENU[0].c) {

n = ceil((30 - PLAYER2.size() - 2) / 2 + 0.5);

GotoXY(LEFT + BOARD\_SIZE \* 4 + n, TOP + ((BOARD\_SIZE \* 2) / 2) + 3); cout << PLAYER2;

}

else cout << "BOT CARO";

Button(TOP + BOARD\_SIZE \* 2 + 2, LEFT, 15, 2, "U:Undo");

Button(TOP + BOARD\_SIZE \* 2 + 2, LEFT + 15 + 2, 15, 2, "H:Help");

Button(TOP + BOARD\_SIZE \* 2 + 2, LEFT + 15 \* 2 + 4, 15, 2, "M:Sound");

Button(TOP + BOARD\_SIZE \* 2 + 2, LEFT + 15 \* 2 + 32, 15, 2, "L:Save");

Button(TOP + BOARD\_SIZE \* 2 + 2, LEFT + 15 \* 3 + 32 + 2, 15, 2, "ESC:Exit");

}

Hàm này sẽ vẽ khung bàn cờ, thông tin trận đấu và các nút chức năng: U – Undo, H – Help, M – Sound, L – Save, ESC – Exit.

# Xử lý màn hình chính

## void ShowMenu()

|  |  |
| --- | --- |
|  | void ShowMenu() { |
|  | if (\_EXIT) return; |
|  | \_PlaySound(4); |
|  | HANDLE word; |
|  | SOUND = 1; |
|  | MODE = 1; |
|  | NEW\_GAME = 1; |
|  | \_X = CENTER\_X - 14;\_Y = CENTER\_Y - 3; |
|  | int backgroundColor = BRIGHT\_WHITE, textColor = BLUE; |
|  | DrawMenu(); |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | PrintLeftCursor(CENTER\_Y - 3, CENTER\_X - 21); |
|  | PrintRightCursor(CENTER\_Y - 3, CENTER\_X + 47); |
|  | SetColor(backgroundColor, textColor); |
|  | int previousOption = 0; |
|  | while (1) { |
|  | ShowCursor(false); |
|  | GotoXY(\_X, \_Y); |
|  | \_COMMAND = toupper(\_getch()); |
|  | if (\_COMMAND == 'W') { |
|  | \_PlaySound(3); |
|  | MoveUp(); |
|  | //Xóa con trỏ ở vị trí cũ + unhighlight option trước |
|  | SetColor(backgroundColor, backgroundColor); |
|  | PrintLeftCursor(\_Y + 5, CENTER\_X - 21); |
|  | PrintRightCursor(\_Y + 5, CENTER\_X + 47); |
|  | ClearPreviousHightlight(previousOption); |
|  | //Di chuyển con trỏ và hightlight option hiện tại |
|  | SetColor(backgroundColor, BLACK); |
|  | PrintLeftCursor(\_Y, CENTER\_X - 21); |
|  | PrintRightCursor(\_Y, CENTER\_X + 47); |
|  | previousOption--; |
|  | PrintCurrentHighlight(previousOption); |
|  | } |
|  | else if (\_COMMAND == 'S') { |
|  | \_PlaySound(3); |
|  | MoveDown(); |
|  | //Xóa con trỏ ở vị trí cũ + unhighlight option trước |
|  | SetColor(backgroundColor, backgroundColor); |
|  | PrintLeftCursor(\_Y - 5, CENTER\_X - 21); |
|  | PrintRightCursor(\_Y - 5, CENTER\_X + 47); |
|  | ClearPreviousHightlight(previousOption); |
|  | SetColor(backgroundColor, BLACK); |
|  | //Di chuyển con trỏ và hightlight option hiện tại |
|  | PrintLeftCursor(\_Y, CENTER\_X - 21); |
|  | PrintRightCursor(\_Y, CENTER\_X + 47); |
|  | previousOption++; |
|  | PrintCurrentHighlight(previousOption); |
|  | } |
|  | else if (\_COMMAND == 13 && \_Y == \_MENU[4].y) { |
|  | \_EXIT = 1; |
|  | \_PlaySound(4); |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | system("cls"); |
|  | ByeBye(); |
|  | return; |
|  | } |
|  | else if (\_COMMAND == 13) { |
|  | \_PlaySound(3); |
|  | ShowLoadingPage(); |
|  | for (int i = 0;i < MENU\_SIZE;i++) { |
|  | if (\_EXIT) return; |
|  | if (\_Y == \_MENU[i].y) |
|  | ShowPage(\_MENU[i].c); |
|  | } |
|  | } |
|  | else if (\_COMMAND == 'M') { |
|  | if (SOUND == 0) |
|  | SOUND = 1; |
|  | else |
|  | SOUND = 0; |
|  | } |
|  | } |
|  | } |

Hàm này mở ra trang Menu và đặt biến toàn cục NEW\_GAME = 1, sau khi vẽ ra giao diện trang thì chạy vòng lặp nhận phím từ người chơi.

Dòng 20 – 49, xử lí hiệu ứng khi con trỏ di chuyển lên xuống.

Dòng 50 – 57, thoát trò chơi nếu người dùng chọn EXIT.

Dòng 58 - 66, khi người dùng nhấn Enter tại lựa chọn nào thì mở ra trang tương ứng.

Dòng 67 – 72, bật/mở âm thanh khi ấn M.

# Điều chỉnh các thuộc tính của màn hình console

## void FixConsoleWindows()

void FixConsoleWindows() {

HANDLE hdc = GetStdHandle(STD\_OUTPUT\_HANDLE);

HWND consoleWindow = GetConsoleWindow();

LONG style = GetWindowLong(consoleWindow, GWL\_STYLE);

style = style & ~(WS\_MAXIMIZEBOX) & ~(WS\_THICKFRAME);

SetWindowLong(consoleWindow, GWL\_STYLE, style);

SetFontInfo();

SetAndCenterWindow(); //Đặt màn hình ở giữa

SetWindowSize(WIDTH, HEIGHT); //Cài đặt kích thước màn hình

SetScreenBufferSize(WIDTH, HEIGHT);

DisableSelection();

DisableMaximize(); //Làm mờ nút phóng to thu nhỏ

SetConsoleTitle(L"Caro");

HideScrollBars(); //Ẩn thanh cuộn

ShowCursor(0);

DisableMouseInput(); //Vô hiệu hóa click chuột

}

Hàm này fix cứng kích thước màn hình để trong quá trình chơi thì người dùng không tinh chỉnh màn hình gây vỡ giao diện.

## void SetColor(int backgoundColor, int textColor)

void SetColor(int backgoundColor, int textColor) {

HANDLE hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

int colorCode = backgoundColor \* 16 + textColor;

SetConsoleTextAttribute(hStdout, colorCode);

}

Hàm này dùng để chỉnh màu của chữ so với nền mỗi lần in.

# Các hàm dùng để vẽ, in

## In hình ảnh, icon

void PrintRectangle2lines(int top, int left, int width, int height);

void PrintRectangle(int top, int left, int width, int height);

void PrintTree(int left, int top);

void PrintCloud(int left, int top, int type);

void PrintLine(int top, int left, int length);

void Button(int top, int left, int width, int height, string label);

void PrintPencil(int top, int left);

void PrintHelicopter(int x, int y);

void PrintCastle(int x, int y);

void PrintX(int top, int left, int size);

void PrintO(int top, int left, int size);

void DrawScore(int score, int top, int left);

void RemoveScore(int top, int left);

void Animal(int top, int left, int animal);

void ByeBye();

## In tiêu đề các lựa chọn ở trang menu

void PrintNEW();

void PrintCONTINUE();

void PrintABOUT();

void PrintEXIT();

void PrintPVC();

## In tiêu đề trang

void PrintContinueLogo(int top, int left);

void PrintInMatch(int top, int left);

void PrintAboutLogo(int top, int left);

void PrintMenuLogo(int top, int left);

void PrintLoadingLogo(int top, int left);

void PrintHelpLogo(int top, int left);

## Hiệu ứng

void ClearPreviousHightlight(int& a);

void PrintCurrentHighlight(int& a);

void PrintLeftCursor(int top, int left);

void PrintRightCursor(int top, int left);

## Vẽ giao diện các trang

void DrawMenu();

void DrawBoard();

void DrawExistedData();

void DrawMatchList();

void DrawPopUp(WORD wVirtualKeyCode);

void Draw\_AskContinue();

void DrawAbout();

void DrawLoadingPage();

void DrawHelp();

# Chức năng Undo

if (\_MODEPLAY == \_MENU[0].c) {

\_PlaySound(4);

if (\_LAST\_POINT.c != 0) { //Xét trường hợp Undo 2 lần trên 1 ô

//Hiệu ứng đổi lượt

if (\_TURN == true) {

SetColor(BRIGHT\_WHITE, GRAY);

PrintX(TOP + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2);

SetColor(BRIGHT\_WHITE, GREEN);

PrintO(TOP + ((BOARD\_SIZE \* 2 - (4 \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2);

GotoXY(\_LAST\_POINT.x, \_LAST\_POINT.y);

}

else if (\_TURN == false) {

SetColor(BRIGHT\_WHITE, RED);

PrintX(TOP + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2);

SetColor(BRIGHT\_WHITE, GRAY);

PrintO(TOP + ((BOARD\_SIZE \* 2 - (4 \* 2)) / 2) + 4 \* 2 + 4, LEFT + BOARD\_SIZE \* 4 + 12, 2);

GotoXY(\_LAST\_POINT.x, \_LAST\_POINT.y);

}

\_X = \_LAST\_POINT.x;

\_Y = \_LAST\_POINT.y;

GotoXY(\_X, \_Y);

cout << char(32);

GotoXY(\_X, \_Y);

XYinMatrix(\_X, \_Y, row, col);

\_A[row][col].c = 0;

\_TURN = !\_TURN;

\_LAST\_POINT.c = 0;

}

}

Chức năng này nằm trong hàm ShowGame(), dùng để gỡ dấu mà người chơi vừa đánh để đánh lại .

Mỗi lần đánh 1 dấu đều được lưu vào biến toàn cục \_LAST\_POINT. Khi nhấn “U” thì đi đến tọa độ của \_LAST\_POINT và in đè lên ô đó 1 khoảng trống, đồng thời tìm trong mảng bàn cờ điểm có chứa tọa độ đó để đặt giá trị .c về 0. Cuối cùng, vì mỗi lần đánh xuống thì sẽ đổi lượt ngay, vì thế nên thêm 1 lần đổi lượt trong hàm này để trở về lượt cũ. Đồng thời cũng chuyển đổi hiển thị lượt đánh.

# Chế độ đánh với máy

|  |  |
| --- | --- |
|  | void PVC(TURN\_BOT tb[]) { |
|  | srand((unsigned char)time(0)); |
|  | int max = 0, number = 0; |
|  | \_POINT turn\_b = { 0,0,2 }; |
|  | TURN\_BOT turn\_bot; |
|  | for (int i = 0; i < 4; i++) { |
|  | if (tb[i].led[0].c == 0 || tb[i].led[1].c == 0) { |
|  | if ((tb[i].led[0].c == 0 && tb[i].led[1].c == 0 && tb[i].sz >= max) || tb[i].sz > max) { |
|  | max = tb[i].sz; |
|  | turn\_bot = tb[i]; |
|  | } |
|  | } |
|  | } |
|  | number = rand() % 2; |
|  | turn\_b = turn\_bot.led[number]; |
|  | if (turn\_b.c != 0) turn\_b = turn\_bot.led[1-number]; |
|  | bool stop = 0; |
|  | if (turn\_b.x < 0 || turn\_b.y < 0) { |
|  | for (int i = 0; i < BOARD\_SIZE; i++) { |
|  | if (stop == 1) break; |
|  | for (int j = 0; j < BOARD\_SIZE; j++) |
|  | if (\_A[i][j].c == 0) { |
|  | \_X = \_A[0][0].x + j \* 4; |
|  | \_Y = \_A[0][0].y + i \* 2; |
|  | stop = 1; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | else { |
|  | \_X = turn\_b.x; |
|  | \_Y = turn\_b.y; |
|  | } |
|  | \_TURN = false; |
|  | Sleep(200); |
|  | SetColor(BRIGHT\_WHITE, GREEN); |
|  | GotoXY(\_X, \_Y); |
|  | \_PlaySound(1); |
|  | cout << "O"; |
|  | CheckBoard(\_X, \_Y); |
|  | switch (ProcessFinish(TestBoard())) { |
|  | case -1:case 1: { |
|  | \_PlaySound(4); |
|  | AskContinue(); |
|  | if (\_EXIT) return; |
|  | break; |
|  | } |
|  | case 0: { |
|  | AskContinue(); |
|  | if (\_EXIT) return; |
|  | break; |
|  | } |
|  | } |
|  | \_TURN = false; |
|  |  |
|  | SetColor(BRIGHT\_WHITE, RED); |
|  | PrintX(TOP + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | SetColor(BRIGHT\_WHITE, GRAY); |
|  | PrintO(TOP + ((BOARD\_SIZE \* 2) / 2) + 6, LEFT + BOARD\_SIZE \* 4 + 12, 2); |
|  | GotoXY(\_LAST\_POINT.x, \_LAST\_POINT.y); |
|  | \_LAST\_POINT.c = 0; |
|  |  |
|  | } |

Mảng lưu tọa độ các vị trí chặn xung quanh khi lượt X đánh xuống TURN\_BOT tb[].

Có 4 hướng cần xét nên TURN\_BOT tb[] có 4 phần tử sau khi kết thúc hàm TestBoard.

Xét 4 phần tử này:

if (tb[i].led[0].c == 0 || tb[i].led[1].c == 0) : Kiểm tra phần tử mảng tb[] tương ứng kiểm tra từng hướng, xét xem hướng đó có tồn tại ít nhất 1 đầu chưa bị chặn.

if ((tb[i].led[0].c == 0 && tb[i].led[1].c == 0 && tb[i].sz >= max) || tb[i].sz > max) :

+ (tb[i].sz > max) : Tìm hướng có nhiều lượt đánh của người chơi X nhất.

+ (tb[i].led[0].c == 0 && tb[i].led[1].c == 0 && tb[i].sz >= max) : nếu tồn tại hướng có số lượt đánh của người chơi X là >=max có hai đầu đều không bị chặn thì ưu tiên gán hướng đó cho max thay cho hướng ban đầu.

=>nhận hướng có lượt đánh của X nhiều nhất hoặc khi có hai hướng cùng có lượt đánh cao nhất thì sẽ ưu tiên hương chưa bị chặn cả hai đầu.

Dòng 14 – 16 : thực hiện random 1 trong 2 vị trí chặn của hướng sau khi xét điều kiện.

Dòng 17 – 33: nếu vị trí chặn tìm được có tọa độ .x, .y âm thì Bot sẽ đánh ngay tại ô còn trống đầu tiên của bàn cờ, ngược lại thì giữ nguyên tọa độ của vị trí chặn tìm được.

Dòng 37 – 39: gán tọa độ \_X,\_Y là vị trị đánh của Bot đi đến vị trí đó và cout kí hiệu “O”.

Dòng 40: Thực hiện gán lượt đánh của Bot vào bàn cờ thông qua CheckBoard(\_X,\_Y)

Dòng 41 – 54: Kiểm tra lượt đánh của Bot để xử lí chiến thắng hay đổi lượt tiếp tục trận đấu.

Dòng 56 – 61: In giao diện chuyển lượt trên khung phải bàn cờ.

# Âm thanh

void \_PlaySound(int i)

{

vector<const wchar\_t\*> soundFile{ L"moveO.wav", L"moveX.wav",

L"move.wav", L"enter.wav", L"error.wav", L"placed.wav", L"win.wav", L"lose.wav", L"background.wav", L"effect.wav" };

if (SOUND == 1) {

PlaySound(soundFile[i], NULL, SND\_FILENAME | SND\_ASYNC);

}

}

Hàm này dùng để phát âm thanh trò chơi. Có 1 biến toàn cục là SOUND, biến này dùng để bật/tắt âm thanh bằng phím “M”. Khi âm thanh được mở thì sẽ chạy file âm thanh trong mảng soundFile tương ứng với biến I được truyền vào hàm.

# Xóa file

|  |  |
| --- | --- |
|  | void RemoveMatchFile(string& matchName, int& pos) { |
|  | //Xoa ten file trong file luu ten cac file |
|  | ifstream input("game\_files.txt"); |
|  | if (!input) return; |
|  | ofstream output("new\_game\_files.txt"); |
|  | string line; |
|  | int i = 0; |
|  | while (getline(input, line)) { |
|  | if (line != matchName) { |
|  | if (i == 0)output << line; |
|  | else output << endl << line; |
|  | i++; |
|  | } |
|  | } |
|  | input.close(); |
|  | output.close(); |
|  | remove("game\_files.txt"); |
|  | bool file = rename("new\_game\_files.txt", "game\_files.txt"); |
|  | remove(matchName.c\_str());//Xóa file lưu dữ liệu |
|  |  |
|  | //xoa file trong mang cau truc & cap nhat lai toa do |
|  | for (int j = pos + 1; j < MATCH\_LIST\_SIZE; j++) { |
|  | \_MATCH\_LIST[j].y -= 4; |
|  | } |
|  | \_MATCH\_LIST.erase(\_MATCH\_LIST.begin() + pos); |
|  | MATCH\_LIST\_SIZE--; |
|  | } |

Dòng 3: Mở file lưu các tên trận đấu.

Dòng 5: Tạo 1 file tạm để lưu danh sách các tên.

Dòng 8 – 14: Copy các tên trận đấu trừ tên muốn xóa từ file cũ.

Dòng 17: Xóa file chứa tên cũ.

Dòng 18: Đổi tên file mới theo tên file cũ.

Dòng 19: Xóa file lưu dữ liệu.

Dòng 22 – 26: Xóa file trong mảng cấu trúc và cập nhật lại tọa độ.

# Tìm file

|  |  |
| --- | --- |
|  | void SearchFile() { |
|  | if (\_EXIT) return; |
|  | MODE = 4; |
|  | string subName = ""; |
|  | char c = '\0'; |
|  | SUB\_ML\_SIZE = 0; |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | PrintRectangle2lines(CENTER\_Y - 7, CENTER\_X, 30, 4); |
|  |  |
|  | ClearListFile(SUB\_ML\_SIZE); |
|  | CoutListFile(); |
|  |  |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | GotoXY(CENTER\_X + 8, CENTER\_Y - 5); |
|  | for (int i = 0; i < 15; i++) cout << " "; |
|  | GotoXY(CENTER\_X + 1, CENTER\_Y - 5); |
|  | cout << " Find: "; |
|  | do { //Nhập tên file |
|  | c = \_getch(); |
|  | if (c == 13) { |
|  | if (SUB\_ML\_SIZE == 0) continue; |
|  | else { |
|  | //Di chuyển con trỏ |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | PrintLeftCursor(\_Y, CENTER\_X - 6); |
|  | PrintRightCursor(\_Y, CENTER\_X + 32); |
|  | } |
|  | if (subName == "") MODE = 3; |
|  | return; |
|  | } |
|  | if (48 <= c && c <= 57 || 65 <= c && c <= 90 || 97 <= c && c <= 122 || c == '\_' || c == '-') { |
|  | subName += c; |
|  | cout << c; |
|  | } |
|  | if (c == 8) { |
|  | if (subName.length() > 0) { |
|  | subName.erase(subName.end() - 1); |
|  | cout << c << ' ' << c; |
|  | } |
|  | } |
|  | SUB\_ML\_SIZE = 0; |
|  | \_SUB\_ML.clear(); |
|  | if (subName != "") { |
|  | GotoXY(CENTER\_X + 8, CENTER\_Y); |
|  | cout << " "; |
|  | for (int i = 0; i < MATCH\_LIST\_SIZE; ++i) { |
|  | if (CheckSubStr(subName, \_MATCH\_LIST[i].item)) { |
|  | \_SUB\_ML.push\_back({ \_MATCH\_LIST[i].x,\_MATCH\_LIST[SUB\_ML\_SIZE].y,\_MATCH\_LIST[i].item }); |
|  | ++SUB\_ML\_SIZE; |
|  | } |
|  | } |
|  | ClearListFile(SUB\_ML\_SIZE); |
|  | } |
|  | if (subName == "" && c == 8) { |
|  | GotoXY(CENTER\_X + 8, CENTER\_Y); |
|  | cout << " "; |
|  | CoutListFile(); |
|  | } |
|  | if (c == 27) { |
|  | SUB\_ML\_SIZE = 0; |
|  | \_SUB\_ML.clear(); |
|  | ShowFileGame(); |
|  | } |
|  | if (SUB\_ML\_SIZE) { |
|  | for (int i = 0; i < SUB\_ML\_SIZE; i++) { |
|  | if (\_SUB\_ML[i].item == "\0")continue; |
|  | PrintRectangle(CENTER\_Y - 1 + i \* 4, CENTER\_X, 30, 4); |
|  | GotoXY(CENTER\_X + 10, CENTER\_Y - 1 + i \* 4 + 2); |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | cout << \_SUB\_ML[i].item; |
|  | SetColor(BRIGHT\_WHITE, LIGHT\_AQUA); |
|  | } |
|  | \_Y = \_SUB\_ML[0].y; |
|  | } |
|  | else { |
|  | if (subName != "") { |
|  | GotoXY(CENTER\_X + 8, CENTER\_Y); |
|  | cout << "No files found !"; |
|  | } |
|  | } |
|  | GotoXY(CENTER\_X + 8 + subName.length(), CENTER\_Y - 5); |
|  | SetColor(BRIGHT\_WHITE, BLACK); |
|  | } while (1); |
|  | } |

Hàm nhận kí tự mà người dùng nhập vào, nếu là kí tự hợp lệ làm tên file thì sẽ xử lí xuất ra file tương ứng, nếu là “ESC” thì hiện các file ban đầu, nếu là những nút khác thì không có gì xảy ra.

Khi người dùng nhập vào mỗi kí tự thì sẽ kiểm tra các file có tên chứa kí tự đó. Các file đó được đưa vào 1 mảng con \_SUB\_ML để xuất ra màn hình và cho người dùng lựa chọn.

Dòng 13 – 17: In khung nhập tên.

Dòng 20 – 30: Xử lí khi nhấn “Enter”, in con trỏ lựa chọn, nếu không nhập gì thì thoát khỏi chức năng tìm file.

Dòng 31 – 34: Xử lí khi nhập các kí tự trong tên.

Dòng 43 – 53: Khi tên được nhập thì thêm các file chứa chuỗi con đó vào mảng con.

Dòng 54 – 58: Xử lí khi nhấn nút BackSpace khi đã xóa hết các kí tự trong tên.

Dòng 59 – 63: Xử lí thoát hàm khi nhấn “ESC”.

Dòng 64 – 80: Xuất các file trong mảng con và đặt con trỏ vào phần tử đầu.

# Tài liệu tham khảo

1. Tài liệu hướng dẫn đồ án của thầy Trương Toàn Thịnh.
2. Hướng dẫn sử dụng thư viện Windows.h: [https://learn.microsoft.com/en-us/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbWZyNE5Ia3Buc21KbmlOanNsTzItSUFMQ1h6QXxBQ3Jtc0ttRGtGeWduZDRmd1pkSW55ZUt6WDlPcWM5ZlZOOUppRk1SblM5WkZoYnByNldkQ2paT1g3WUFmN2JBLTlsT3NHQml2TjE4djZYQ1BfRWUzTHRxVDUyTmU3R3I5TzBWWjhwMlpfLVFvSHo1b0I4enRJZw&q=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2F&v=gtT4cyaR_Hk)
3. Video demo của các anh chị.

# Video demo

Link tải: <https://mega.nz/file/SeYXyZBC#ol1skBQeE0sEFe7YyCKzj6PPuCmg2oNcrxSM-95-wtQ>

Link Youtube: <https://youtu.be/gtT4cyaR_Hk>