TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **CƠ SỞ LẬP TRÌNH**

BÁO CÁO ĐỒ ÁN CỜ CARO

LẦN 1

Môn học: **Cơ sở lập trình**

GV hướng dẫn: **TS. Trương Toàn Thịnh**

Lớp: ***24CTT3 (2024 – 2025)***

Nhóm 1:

Ngô Viết Thanh Bình - 24120269

Đặng Bùi Thế Bảo – 24120261

Dương Nhựt Duy – 24120293

Ngô Mẫn Nhi – 24120250

TP.HCM, ngày 22 tháng 10 năm 2024

MỤC LỤC

1 Giới thiệu

Sản phẩm đồ án cờ CARO của nhóm một được xây dựng dựa trên những kiến thức cơ bản như: vòng lặp, xử lý tập tin, xử lý hàm, thao tác nhập xuất, cấu trúc dữ liệu mảng một/hai chiều…

Sản phẩm vẫn đang trong giai đoạn hoàn thiện nên chỉ mới có một số chức năng cơ bản như vẽ bàn cờ, xử lý thắng thua, lưu tải trò chơi,…

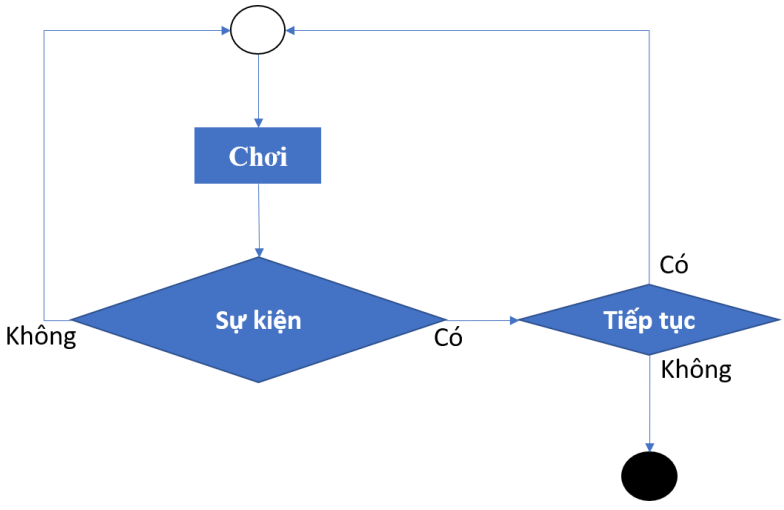
2 Kịch bản trò chơi

Ban đầu khi vào game sẽ xuất hiện một bảng menu, người chơi sẽ dùng các phím 'W', 'S' để chọn chức năng. Khi người chơi nhấn phím 'enter' thì chương trình sẽ thực hiện chức năng tương ứng.

Khi chọn "new game" sẽ xuất hiện bảng cờ caro, người chơi sẽ dùng các phím 'W', 'A', 'S', 'D' để điều chỉnh hướng di chuyển. Khi người chơi nhấn phím 'enter' thì sẽ xuất hiện dấu 'X' hoặc 'O' tùy vào lượt.

Khi một trong hai người chiến thắng theo luật caro thì màn hình xuất hiện dòng chữ chúc mừng người chiến thắng. Sau đó sẽ hỏi người dùng muốn tiếp tục chơi hay không, nếu chọn phím "y" thì chương trình khởi động lại dữ liệu từ đầu, còn nhấn phím bất kì thì thoát chương trình.

Trường hợp khi vị trí bàn cờ đã kín chỗ thì màn hình xuất hiện dòng chữ 'Hai ben hoa nhau'. Sau đó hỏi người dùng có muốn thoát hay chơi tiếp tương tự như trên.



Hình 1: Sơ đồ kịch bản trò chơi

3 Các bước xây dựng trò chơi

(Tham khảo từ tài liệu của Giảng viên hướng dẫn: Thầy Trương Toàn Thịnh)

Bước 1: Cố định màn hình với kích thước phù hợp. Mục đích hạn chế thay đổi kích thước màn hình trong quá trình xử lý game.

Dòng	// Hàm nhóm View

	void FixConsoleWindow() {
	HWND consoleWindow = GetConsoleWindow();
	LONG style = GetWindowLong(consoleWindow, GWL_STYLE);
	style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);
	SetWindowLong(consoleWindow, GWL_STYLE, style);
	}

Bước 2: Đưa con trỏ đến vị trí mong muốn trên màn hình console để thao tác.

Dòng	// Hàm nhóm View

1	void GotoXY(int x, int y) {
	COORD coord;
	coord.X = x;
	coord.Y = y;
	SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
	}

Tương ứng với x, y trong GotoXY(x,y) con trỏ sẽ đi đến vị trí có tọa độ x, y để bắt đầu thực hiện câu lệnh tương ứng.

Ví dụ:

GotoXY(3, 5);

cout<<"Xin chao";

Với câu lệnh trên chữ "Xin chao" sẽ được in từ vị trí có tọa độ x = 3, y = 5 trên màn hình console.

Bước 3: Một số khai báo hằng để hỗ trợ khởi tạo trò chơi

Dòng	

1	//Hằng số
	#define BOARD_SIZE 15// Kích thước ma trận bàn cờ
	#define LEFT 5 // Tọa độ trái màn hình bàn cờ //x
	#define TOP 5// Tọa độ trên màn hình bàn cờ //y
	// Khai báo kiểu dữ liệu
	struct _POINT { int x, y, c; }; // x: tọa độ dòng, y: tọa độ cột, c: đánh dấu
	_POINT _A[BOARD_SIZE][BOARD_SIZE]; //Ma trận bàn cờ
	bool _TURN; //true là lượt người thứ nhất và false là lượt người thứ hai
	//1--> Player_1
	//0--> Player_2
	int _COMMAND; // Biến nhận giá trị phím người dùng nhập
	int _X, _Y; //Tọa độ hiện hành trên màn hình bàn cờ

Bước 4: Khởi tạo dữ liệu lại trang thái ban đầu, sẵn sàng để bắt đầu trò chơi mới khi có yêu cầu.

Dòng	

1	//Hàm khởi tạo dữ liệu mặc định ban đầu cho ma trận bàn cờ
	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; // Trùng với hoành độ màn hình bàn cờ
	
	_A[i][j].y = 2 * i + TOP + 1; // Trùng với tung độ màn hình bàn cờ
	
	_A[i][j].c = 0; // 0 nghĩa là chưa ai đánh dấu, nếu đánh dấu phải theo quy
	//định như sau: -1 là lượt true đánh, 1 là lượt false đánh
	
	}
	}
	_TURN = true; _COMMAND = -1; // Gán lượt và phím mặc định
	_X = _A[0][0].x; _Y = _A[0][0].y; // Thiết lập lại tọa độ hiện hành ban đầu
	
	}

Bước 5: Vẽ bàn cờ, và khung trò chơi có kích thước là pSize tùy người dùng yêu cầu.

Dòng	**// Hàm nhóm View**

1	void DrawBoard(int pSize)
	{
	//ngang
	for (int ix = LEFT + 1; ix <= LEFT + pSize * 4 - 1; ix += 4)
	{
	for (int iy = TOP; iy <= TOP + pSize * 2; iy += 2)
	{
	GotoXY(ix, iy);
	cout << char(196);
	GotoXY(ix + 1, iy);
	cout << char(196);
	GotoXY(ix + 2, iy);
	cout << char(196);
	}
	}
	//doc
	for (int iy = TOP + 1; iy <= TOP + pSize * 2 - 1; iy += 2)
	{
	for (int ix = LEFT; ix <= LEFT + pSize * 4; ix += 4)
	{
	GotoXY(ix, iy);
	cout << char(179);
	}
	}
	//cong
	for (int ix = LEFT + 4; ix <= LEFT + pSize * 4; ix += 4)
	{

```
                for (int iy = TOP + 2; iy <= TOP + pSize * 2 - 2; iy += 2)
                {
                        GotoXY(ix, iy);
                        cout << char(197);
                }
        }
        //cong tren/duoi
        for (int ix = LEFT + 4; ix <= LEFT + pSize * 4; ix += 4)
        {
                GotoXY(ix, TOP);
                cout << char(194);
                GotoXY(ix, TOP + pSize * 2);
                cout << char(193);
        }
        //cong trai/phai
        for (int iy = TOP + 2; iy <= TOP + pSize * 2; iy += 2)
        {
                GotoXY(LEFT, iy);
                cout << char(195);
                GotoXY(LEFT + pSize * 4, iy);
                cout << char(180);
        }
        GotoXY(LEFT, TOP); cout << char(218);//goc tren tai
        GotoXY(pSize * 4 + LEFT, TOP); cout << char(191); //goc tren phai
        GotoXY(LEFT, TOP + pSize * 2); cout << char(192); //goc duoi trai
        GotoXY(LEFT + pSize * 4, TOP + pSize * 2); cout << char(217); //goc duoi phai
 }
void DrawBound(int x0, int y0, int h, int w)
{
        //ngang
        for (int ix = x0 + 1; ix < x0 + w; ix++)
        {
                GotoXY(ix, y0);
                cout << char(196);
                GotoXY(ix, y0 + h);
                cout << char(196);
        }
        //doc
        for (int iy = x0 + 1; iy < y0 + h; iy++)
        {
                GotoXY(x0, iy);
                cout << char(179);
                GotoXY(x0 + w, iy);
                cout << char(179);
        }

        GotoXY(x0, y0); cout << char(218);
        GotoXY(x0 + w, y0); cout << char(191);
        GotoXY(x0, y0 + h); cout << char(192);
        GotoXY(x0 + w, y0 + h); cout << char(217);
 }
```

Hàm DrawBound() vẽ khung viền của trò chơi.

Bước 6: Tiếp theo ta sẽ xây dựng hàm StartGame(), hàm này thực chất là tập các công việc cần làm trước khi vào trò chơi

Dòng	**// Hàm nhóm Control**

```
void StartGame() {
	system("cls");
	system("color F0");
	ResetData(); // Khởi tạo dữ liệu gốc
	DrawBoard(BOARD_SIZE); // Vẽ màn hình game
	DrawBound(1, 1, 40, 150);
 }
```

Dòng mã đầu tiên để xóa trắng màn hình.
Dòng mã thứ hai dùng để tô trắng màn hình console.
Dòng thứ ba đặt lại dữ liệu.
Dòng thứ tư, năm vẽ bàn cờ và khung.

Bước 7: Hai hàm ExitGame() và GabageCollect() thực hiện chức năng thoát và dọn dẹp tài nguyên khi dừng trò chơi.

Dòng 1

```
//Hàm dọn dẹp tài nguyên (hàm nhóm Model)
void GabageCollect()
{
	// chuc ban thanh cong
}
//Hàm thoát game (hàm nhóm Control)

void ExitGame() {
	system("cls");
	GabageCollect();
	//Có thể lưu game trước khi exit
 }
```

Bước 8: Tiếp theo ta xây dựng hàm xử lý khi người chơi thắng/thua. Khi người chơi thắng/thua/hòa, hàm trên đơn giản in ra dòng chữ báo hiệu. Ngoài ra xây dựng them hàm tiện ích AskContinue() để nhận phím quyết định có tiếp tục hay không của người dùng.

Dòng 1 **// Hàm nhóm View**

```
 //Hàm xử lý khi người chơi thua
int ProcessFinish(int pWhoWin) {
	GotoXY(85 + 2, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 8); // Nhảy tới vị trí thích hợp để in chuỗi thắng/thua/hòa

	switch (pWhoWin) {

	case -1:
		Box(75, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 10, 70, 10);
		GotoXY(85 + 10, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 5);
		cout << "Player_1: " << Player_1.Name << " THANG" << " Player_2: " << Player_2.Name << endl;
		break;
	case 1:
		Box(75, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 10, 70, 10);
```

```
                        GotoXY(85 + 10, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 5);
                        cout << "Player_2: " << Player_2.Name << " THANG" << " Player_1: " << Player_1.Name
<< endl;
                        break;
                case 0:
                        Box(75, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 10, 70, 10);
                        GotoXY(85 + 10, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 5);
                        cout << "Player_1: " << Player_1.Name << " HOA" << " Player_2: " << Player_2.Name <<
endl;
                        break;
                case 2:
                        _TURN = !_TURN; // Đổi lượt nếu không có gì xảy ra
                }
                GotoXY(_X, _Y); // Trả về vị trí hiện hành của con trỏ màn hình bàn cờ
                return pWhoWin;
}

// Hàm hỏi người chơi có tiếp tục hay không
int AskContinue() {
                Box(75, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 20, 70, 6);
                GotoXY(75 + 10, _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y - 18); // Nhảy đến vị trí để hỏi người
chơi
                printf("Nhan 'y/n' de tiep tuc/dung: ");
                return toupper(_getch()); // Đọc ký tự và trả về dạng chữ hoa
 }
```

Bước 9: Nhận biết thắng thua hòa

Dòng	**// Hàm nhóm Model**

1	//Hàm kiểm tra xem có người thắng/thua hay hòa
	//Hàm kiểm tra hoa hay khong
	
	bool isDraw() {
	// Kiểm tra nếu tất cả các ô trên bàn cờ đều đã được điền
	for (int i = 0; i < BOARD_SIZE; i++) {
	for (int j = 0; j < BOARD_SIZE; j++) {
	if (_A[i][j].c == 0) // Ô trống
	return false; //--> chua full
	}
	}
	return true; // Hòa khi tất cả các ô đều đầy
	}

	//Hàm kiểm tra co ai thang hay khong
	
	bool checkWin(int x, int y) {
	int player = _A[x][y].c; // Giá trị người chơi ở ô (x, y)
	
	int dem_1 = 0;
	int dem_2 = 0;
	int dem_3 = 0;
	int dem_4 = 0;
	
	//ngang
	for (int j = y; j < y + 5; j++)
	{

```
                if (_A[x][j].c == player) dem_1++;
        }

        //doc
        for (int i = x; i < x + 5; i++)
        {
                if (_A[i][y].c == player) dem_2++;
        }

        //cheo chinh
        for (int k = 0; k < 5; k++)
        {
                if(_A[k+x][k+y].c == player) dem_3++;
        }

        //cheo phu
        for (int k = 0; k < 5; k++)
        {
                if (_A[x - k][k + y].c == player) dem_4++;
        }

        // Nếu có đủ 5 quân liên tiếp
        if (dem_1 >= 5 || dem_2 >= 5 || dem_3 >= 5 || dem_4 >= 5) {
                return true;
        }

        return false;
}
```

```
//Hàm kiểm tra ai thang

int TestBoard() {
        // Kiểm tra điều kiện hòa
        if (isDraw()) {
                return 0;    // Hòa
        }

        // Kiểm tra điều kiện thắng
        for (int i = 0; i < BOARD_SIZE; i++) {
                for (int j = 0; j < BOARD_SIZE; j++) {
                        if (_A[i][j].c != 0 && checkWin(i, j)) {
                                if (_TURN == true) {
                                        Player_1.Wins++;
                                        return -1;
                                }
                                else {
                                        Player_2.Wins++;
                                        return 1;
                                }
                                //return (_TURN == true ? -1 : 1);    // -1 là lượt người chơi 1 thắng, 1 là
người chơi 2 thắng
                        }
                }
        }
        return 2;    // 2 nghĩa là chưa ai thắng, trò chơi tiếp tục
}
```

Hàm isDraw() trả ra 0 có nghĩa là hòa.

Hàm checkWin() kiểm tra xem có đủ điều kiện thắng hay chưa, nếu có trả về true, ngược

lại trả về false.
Hàm TestBoard() trả về -1 có nghĩa lượt 'true' thắng, 1 có nghĩa lượt 'false' thắng và 2 có nghĩa là chưa ai thắng.

Bước 10: Ta cần xây dựng hàm đánh dấu vào ma trận bàn cờ khi người chơi nhấn phím 'enter'.

Dòng	// Hàm nhóm Model

1	int CheckBoard(int pX, int pY) {
	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; // Nếu lượt hiện hành là true thì c = -1
	else _A[i][j].c = 1; // Nếu lượt hiện hành là false thì c = 1
	return _A[i][j].c;
	}
	}
	}
	return 0;
	}

Bước 11: Các hàm di chuyển trên bàn cờ

Dòng	// Hàm nhóm Control

	void MoveRight() {
	if (_X < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].x)
	{
	_X += 4;
	GotoXY(_X, _Y);
	}
	}
	void MoveLeft() {
	if (_X > _A[0][0].x) {
	_X -= 4;
	GotoXY(_X, _Y);
	}
	}
	void MoveDown() {
	if (_Y < _A[BOARD_SIZE - 1][BOARD_SIZE - 1].y)
	{
	_Y += 2;
	GotoXY(_X, _Y);
	}
	}
	void MoveUp() {
	if (_Y > _A[0][0].y) {
	_Y -= 2;
	GotoXY(_X, _Y);
	}
	}

Trong quá trình di chuyển trên màn hình bàn cờ, nếu vượt quá phạm vi thì ta không xử lý, ngược lại ta thực hiện cập nhật tọa độ vị trí mới trên màn hình bàn cờ

Bước 12: Khởi tạo thông số người chơi (Chế độ PvP)

Dòng	

1	struct Player {
	string Name;
	int Moves = 0;//so buoc di chuyen
	int Wins = 0;//so luoc thang
	
	void Thong_so_Players(int n) //n: chi so cua nguoi choi
	{
	cout << endl << "Nhap thong tin Players: " << endl;
	cout << "Name of Player_" << n << ": ";
	getline(cin, Name);
	}
	
	void In_thong_so_Player(int n)
	{
	cout << "Player_" << n << "'s Name: " << Name << endl;
	}
	void So_buoc_di_chuyen() {
	cout << "Moves: " << Moves << endl;
	}
	void So_lan_thang() {
	cout << "Wins: " << Wins << endl;
	}
	};
	
	Player Player_1;
	Player Player_2;

Sử dụng cấu trúc dữ liệu struct để khởi tạo cũng như xử lý nhập, xuất thông số người chơi.

Bước 13: Hàm nhập, xuất thông số người chơi

Dòng	

1	void Khoi_tao_Players(Player& Player_1, Player& Player_2, int n)
	{
	if (n == 1)
	{
	Player_1.Thong_so_Players(1);
	}
	else if (n == 2)
	{
	Player_1.Thong_so_Players(1);
	Player_2.Thong_so_Players(2);
	}
	}
	void Show_Player(int x, int y, int w, int h, int number_of_players)
	{
	
	{
	if (number_of_players == 1)

```
		{
			Box(x, y, w, h);
			GotoXY(x + 2, y + 1);
			Player_1.In_thong_so_Player(1);
			GotoXY(x + 2, y + 2);
			Player_1.So_buoc_di_chuyen();
			GotoXY(x + 2, y + 3);
			Player_1.So_lan_thang();

		}
		else if (number_of_players == 2)
		{
			Box(x, y, w, h);
			GotoXY(x + 2, y + 1);
			Player_1.In_thong_so_Player(1);
			GotoXY(x + 2, y + 2);
			Player_1.So_buoc_di_chuyen();
			GotoXY(x + 2, y + 3);
			Player_1.So_lan_thang();

			Box(x + 45, y, w, h);
			GotoXY(x + 45 + 2, y + 1);
			Player_2.In_thong_so_Player(2);
			GotoXY(x + 45 + 2, y + 2);
			Player_2.So_buoc_di_chuyen();
			GotoXY(x + 45 + 2, y + 3);
			Player_2.So_lan_thang();
		}
	}

}
```

Gọi hàm khởi tạo để nhập thông số người chơi khi chọn chức năng new game từ menu. Hiển thị thông số người chơi như: số lược đi moves, wins, trên màn hình chơi game.

Bước 14: Hàm main()

Dòng	

1	int main()
	{
	int choice = menu();
	ShowCur(true);
	switch (choice)
	{
	case 0:
	system("cls");
	Play_Game();
	
	case 1:
	break;
	case 2:
	break;
	case 3:
	system("cls");
	Help(10, 10, 80, 30);
	break;
	case 4:
	ExitGame();
	system("pause");

```
                    break;
            }
            system("pause");
            return 0;
}
```

4 CÁC TÍNH NĂNG BỔ SUNG

4.1 Xử lý lưu/tải trò chơi (save/load)

File Save_Load_Game.h

Dòng	
1	

```
#pragma once
#include <iostream>
#include <string>
#include <fstream>

using namespace std;


void Save(_POINT _A[BOARD_SIZE][BOARD_SIZE], const Player& Player_1, const Player& Player_2, bool _TURN, const string& FileName) // Doc va ghi File --> luu Game vao file
{
	//----------------------------------------------------SAVE GAME----------
	ofstream outputFile(FileName, ios::app);
	if (!outputFile)
	{
		cerr << "Khong the mo File de ghi!" << endl;
		return;
	}

	//Luu ban co
	for(int j = 0; j < BOARD_SIZE; j++)
	{
		for (int i = 0; i < BOARD_SIZE;i++)
		{
			outputFile << _A[i][j].c << " ";//outputFile << _A[i][j]. --> THIẾU .c là sai --> có thể sửa bằng một số cách khác nhau Như định nghĩa toán tử
		}
		outputFile << endl;
	}

	//Luu thong so nguoi choi
	outputFile << Player_1.Name << " " << Player_1.Moves << " " << Player_1.Wins << endl;
	outputFile << Player_2.Name << " " << Player_2.Moves << " " << Player_2.Wins << endl;
	outputFile << _TURN << endl;

	outputFile.close();
}

void Load(_POINT _A[BOARD_SIZE][BOARD_SIZE], Player Player_1, Player Player_2, bool _TURN, const string& FileName)
{
	// doc --> luu --> hien thi
```

```
	ifstream inputFile(FileName);
	if (!inputFile)
	{
		cerr << "Khong the mo File de doc!" << endl;
		return;
	}
	//Doc ban co
	for (int j = 0; j < BOARD_SIZE; j++)
	{
		for (int i = 0; i < BOARD_SIZE; i++)
		{
			inputFile >> _A[i][j].c;//outputFile << _A[i][j]. --> THIẾU .c là sai --> có thể sửa bằng một số cách khác nhau Như định nghĩa toán tử
		}
	}

	//Luu thong so nguoi choi
	inputFile >> Player_1.Name >>    Player_1.Moves    >> Player_1.Wins ;
	inputFile >> Player_2.Name >>    Player_2.Moves    >> Player_2.Wins ;
	inputFile >> _TURN;

	inputFile.close();
}
```

4.2 Xử lý hiệu ứng thắng/thua/hòa

File Win.h

Dòng	

1	`#pragma once`
	`#include <iostream>`
	`#include <cstdlib> // Thư viện để sử dụng hàm system("clear") hoặc system("cls")`
	`#include <thread>    // Thư viện để sử dụng hàm sleep_for`
	`#include <chrono>    // Thư viện để định nghĩa thời gian chờ`
	
	`using namespace std;`
	
	`// Hàm để thay đổi màu sắc bằng mã màu ANSI`
	`// Hàm để thay đổi màu sắc bằng mã màu ANSI`
	`void SettColor(int color) {`
	` cout << "\033[" << color << "m";`
	`}`
	
	`// Hàm để in chữ "WIN" với hiệu ứng lấp lánh`
	`void printWin() {`
	` cout << "W            W    IIIII    N        N" << endl;`
	` cout << "W            W      I      NN       N" << endl;`
	` cout << "W     W      W      I      N N      N" << endl;`
	` cout << "W    W W    W      I      N    N    N" << endl;`
	` cout << "W W      W W      I      N     N N N" << endl;`
	` cout << "WW         WW    IIIII    N        N" << endl;`
	`}`

```
void win() {
    while (true) {
        // Đổi màu đỏ
        setColor(31);
        system("cls"); // Dùng "cls" nếu chạy trên Windows
        printWin();
        std::this_thread::sleep_for(std::chrono::milliseconds(300));

        // Đổi màu xanh lá
        setColor(32);
        system("cls");
        printWin();
        std::this_thread::sleep_for(std::chrono::milliseconds(300));

        // Đổi màu vàng
        setColor(33);
        system("cls");
        printWin();
        std::this_thread::sleep_for(std::chrono::milliseconds(300));
    }

    // Reset lại màu về mặc định
    setColor(0);
     return;
```

Hàm hiển thị hiệu ứng khi có người thắng, chữ WIN đổi màu liên tục.

4.3 Xử lý giao diện màn hình khi chơi

Hàm Play_Game()

Dòng	

1	void Play_Game()
	{
	FixConsoleWindow();
	//StartGame();
	bool validEnter = true;
	Khoi_tao_Players(Player_1, Player_2, 2);
	_flushall();
	StartGame();
	Show_Player(75, 7, 25, 4, 2);
	while (1) {
	/*Check_new_game();*/
	_COMMAND = toupper(_getch());
	if (_COMMAND == 27) { // 27 là mã của phím ESC
	return_menu();
	return;
	}
	else {
	if (_COMMAND == 'A') MoveLeft();
	else if (_COMMAND == 'W') MoveUp();
	else if (_COMMAND == 'S') MoveDown();
	else if (_COMMAND == 'D') MoveRight();
	else if (_COMMAND == 13) { // Người dùng nhấn Enter

```
                                        switch (CheckBoard(_X, _Y)) {
                                        case -1:
                                                printf("X");
                                                Player_1.Moves++;
                                                Show_Player(75, 7, 25, 4, 2);
                                                break;
                                        case 1:
                                                printf("O");
                                                Player_2.Moves++;
                                                Show_Player(75, 7, 25, 4, 2);
                                                break;
                                        case 0:
                                                validEnter = false; // Ô đã được đánh dấu trước đó
                                                break;
                                        }
                                        if (validEnter == true) {
                                                switch (ProcessFinish(TestBoard())) {
                                                case -1:
                                                case 1:
                                                case 0:
                                                        if (AskContinue() != 'Y') {
                                                                system("cls");
                                                                return_menu();
                                                        }
                                                        else {
                                                                StartGame();
                                                        }
                                                }
                                        }
                                        validEnter = true; // Mở khóa cho lượt tiếp theo
                                }
                        }
                }
        }
```

Đầu tiên cần cố định màn hình console, sau đó khởi tạo thông số người chơi khi chọn new game, tiếp đến gọi hàm StarGame và bắt đầu trò chơi.

4.4 Xử lý màn hình chính

File Menu.h

Dòng	

1	#pragma once
	#include <iostream>
	#include <conio.h>
	#include <windows.h>
	#include <string>
	#include <thread> // Thư viện để sử dụng std::this_thread::sleep_for
	#include <chrono> // Thư viện cho thời gian
	using namespace std;
	void DrawBound(int x0, int y0, int h, int w);
	int x = 50;
	int y = 5;
	int box_spacing = 1; // Giảm khoảng cách giữa các khung
	
	// Hàm điều khiển con trỏ tới vị trí x, y trong console

```
void GotoXY(int x, int y);

// Hàm ẩn hoặc hiện con trỏ
void ShowCur(bool CursorVisibility) {
	CONSOLE_CURSOR_INFO cursorInfo;
	GetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursorInfo);
	cursorInfo.bVisible = CursorVisibility; // Ẩn hoặc hiện con trỏ
	SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cursorInfo);
}

// Hàm thay đổi màu chữ
void SetColor(int color) {
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), color);
}

// Hàm tạo hộp (box) có đường viền và tô màu nền
void box(int x, int y, int w, int h, int t_color, int b_color, string nd) {
    system("color F0");
	 // Tô màu nền lam cho tất cả các khung
	 SetColor(b_color);
	// system("color F0");
	 for (int iy = y; iy <= y + h ; iy++) {    // Bỏ "+1" và "-1"
		for (int ix = x; ix <= x + w; ix++) {
			GotoXY(ix, iy);
			cout << " "; // Tô đầy khung
		}
	 }


	 // Đặt màu chữ cho nội dung
	// SetColor(240); // Màu chữ trắng
	 GotoXY(x + (w - nd.length()) / 2, y + h / 2); // Căn giữa nội dung
	 cout << nd;

	 // Vẽ viền
	 SetColor(t_color);
	// SetColor(240);
	 if (h <= 1 || w <= 1) return;

	 for (int ix = x; ix <= x + w; ix++) {
		GotoXY(ix, y);
		cout << char(196);
		GotoXY(ix, y + h);
		cout << char(196);
	 }
	 for (int iy = y; iy <= y + h; iy++) {
		GotoXY(x, iy);
		cout << char(179);
		GotoXY(x + w, iy);
		cout << char(179);
	 }

	 GotoXY(x, y); cout << char(218);
	 GotoXY(x + w, y); cout << char(191);
	 GotoXY(x, y + h); cout << char(192);
	 GotoXY(x + w, y + h); cout << char(217);
}

// Hàm vẽ thanh sáng khi chọn mục
```

```
void thanh_sang(int x, int y, int w, int h, int b_color_sang, string nd) {
	SetColor(b_color_sang); // Màu đậm hơn cho mục được chọn
	for (int iy = y + 1; iy <= y + h - 1; iy++) {
		for (int ix = x + 1; ix <= x + w - 1; ix++) {
			GotoXY(ix, iy);
			cout << " "; // Tô đầy
		}
	}

    // SetColor(252); // Màu chữ trắng --> bỏ dòng này thì chữ sẽ trở về màu cũ sau khi hết chọn
	GotoXY(x + (w - nd.length()) / 2, y + h / 2); // Căn giữa nội dung
	cout << nd;
}


// Hàm tạo menu động
int menu() {
	//system("color F0");
	DrawBound(1, 1, 40, 150);

	ShowCur(false);

	// Cài đặt các thông số menu
	int w = 13; // Thu nhỏ chiều rộng của khung
	int h = 2;    // Chiều cao của khung giữ nguyên
	int t_color = 240; // Màu viền
	int b_color = 240; // Màu nền trắng, chữ đen
	int b_color_sang = 252; // Nền đỏ, chữ đen
	string items[] = { "New Game", "Load Game", "Setting", "Help", "Exit" };
	int sl = 5; // Số lượng mục menu

	// Vẽ menu ban đầu với khoảng cách nhỏ hơn giữa các khung
	for (int i = 0; i < sl; i++) {
		box(x, y + i * (h + box_spacing), w, h , t_color, b_color, items[i]);
	}

	int xp = x;
	int yp = y; // Tọa độ thanh sáng
	int xcu = xp;
	int ycu = yp;
	bool kt = true;

	while (true) {
		if (kt == true) {
			GotoXY(xcu, ycu);
			thanh_sang(xcu, ycu, w, h, b_color, items[(ycu - y) / (h + box_spacing)]);
			xcu = xp;
			ycu = yp;

			thanh_sang(xp, yp, w, h, b_color_sang, items[(yp - y) / (h + box_spacing)]);
			kt = false;
		}


		if (_kbhit()) {
			_COMMAND = toupper(_getch());
			kt = true;
			/* if (c == -32) {
				kt = true;
				c = _getch();*/
```

```
                if (_COMMAND == 'W') { // Lên
                    if (yp != y)
                        yp -= (h + box_spacing);
                    else
                        yp = y + (h + box_spacing) * (sl - 1);
                }
                else if (_COMMAND == 'S') { // Xuống
                    if (yp != y + (h + box_spacing) * (sl - 1))
                        yp += (h + box_spacing);
                    else
                        yp = y;
                }

                if (_COMMAND == 13) { // Enter
                    // Thực thi hành động tương ứng khi nhấn Enter
                    int index = (yp - y) / (h + box_spacing);
                    cout << "\nYou selected: " << items[index] << endl;
                    return index;

                    // Loại bỏ phần màu đỏ (12) và thay bằng hành động hoặc thông báo
                    std::this_thread::sleep_for(std::chrono::milliseconds(1000)); // Tạm dừng 1 giây
                    break; // Thoát vòng lặp

                }

            }
        }
    }
```

4.5 Chức năng help trên menu

File Help.h

Dòng	

1	#pragma once
	#pragma once
	#include <iostream>
	#include <Windows.h>
	#include <string>
	using namespace std;
	void ExitGame();
	void StartGame();
	void Play_Game();
	void GotoXY(int x, int y);
	void Box(int x, int y, int w, int h);
	void Box(int x, int y, int w, int h) {
	if (h <= 1 || w <= 1) return;
	for (int ix = x; ix <= x + w; ix++) {
	GotoXY(ix, y);
	cout << char(196);
	GotoXY(ix, y + h);
	cout << char(196);
	}
	for (int iy = y; iy <= y + h; iy++) {
	GotoXY(x, iy);
	cout << char(179);
	GotoXY(x + w, iy);

```
                cout << char(179);
        }
        GotoXY(x, y);
        cout << char(218);
        GotoXY(x + w, y);
        cout << char(191);
        GotoXY(x, h + y);
        cout << char(192);
        GotoXY(x + w, y + h);
        cout << char(217);
}

void Help(int x, int y, int w, int h)
{
        Box(x + 35, y, w, h);
        GotoXY(x + 35 + 2, y + 2);
        cout << "HUONG DAN THAO TAC VOI MENU:" << endl;
        GotoXY(x + 35 + 2, y + 3);
        cout << "+su dung phim W hoac S de lua chuc nang." << endl;
        GotoXY(x + 35 + 2, y + 4);
        cout << "+su dung phim enter de chon chuc nang.";
        GotoXY(x + 35 + 2, y + 8);
        cout << "HUONG DAN CHOI GAME:" << endl;
        GotoXY(x + 35 + 2, y + 9);
        cout << "+su dung cac phin A, W, S, D de lua vi tri dat quan." << endl;
        GotoXY(x + 35 + 2, y + 10);
        cout << "+su dung phim ENTER de dat quan." << endl;
        GotoXY(x + 35 + 2, y + 11);
        cout << "+ben nao cos duoc 5 quan lien tiep theo cac hang ngang, doc, cheo thi " << endl;
        GotoXY(x + 35 + 2, y + 12);
        cout << " se la nguoi chien thang." << endl;

        while (1) {
                _COMMAND = toupper(_getch());
                if (_COMMAND == 27) { // 27 là mã của phím ESC
                        ExitGame();
                }
        }
        system("pause");
 }
```