VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

DEPARTMENT OF INFORMATION TECHNOLOGY

SUBJECT: **OBJECT-ORIENTED PROGRAMMING**

PROJECT REPORT
CARO

Student made:

Nguyễn Trung Anh-19127333

Dương Thị Xuân Diệu-19127360

Lê Nguyễn Minh Tâm-1753097

INSTRUCTORS: Lecturer Trương Toàn Thịnh

HCM CITY,August 2nd 2021

CONTENTS

1.Introduction

CARO a popular board game of writing on a nine-square, 3x3 paper board. Two players, one using the symbol O, the other using the symbol X, take turns filling in their symbols in the boxes. The winner is the one who can create a sequence of their own symbols first, either horizontally, vertically or diagonally.

Vietnamese people often play a similar game, called Chess, the board is not limited to 9 cells, can draw more cells, to expand until whoever reaches a row of 5 wins.

In this part, we use some class techniques and basic data structure to build a simple caro. To complete this project, we need to know some basic knowledge: object-oriented design, file processing, two-dimensional array…

PROGRAMMING LANGUAGE: C++

2. Requirements and completed components

a. Requirement

1. Save/load – 2m In this guideline, we cannot save and load the game. We need two features. When the players hit ‘L’, we show the line requesting the players provide the filename to save. When the players hit ‘T’, we show the line requesting the players provide the file to load.

2. Recognize win/lose/draw – 2m Need to provide the caro rule to know win/lose/draw.

3. Provide animation of win/lose/draw – 2m In this guideline, we only print the simple line showing win/lose/draw. Provide the vivid animation for this event.

4. Creating playing interface – 1.5m When two players tick the board, we print some statistics of two players, for example, counting the number of ticking for each player … You yourself organize the interface such that it is clear and pretty.

5. Provide the main menu – 1.5m When coming to boardgame, we print the menu game, for example, “New Game”, “Load Game”, “Settings”, … So, this helps the players to easily choose actions they want

6. Playing with machine (Alpha – beta pruning) – 1m Allow the players to choose “play with machine”, and the players can choose the level, for example, easy-level if we allow machine to randomize the position, and hard-level if we apply the alpha – beta pruning

b. completed components

1.Save and load

2. Recognize win/lose/draw

3. Provide animation of win/lose/draw

4.Creating playing interface

5. Provide the main menu

6. Playing with machine (Alpha – beta pruning)

3. Structure components

3.1 Program structure: oop

header file: class declaration

cpp file: build properties

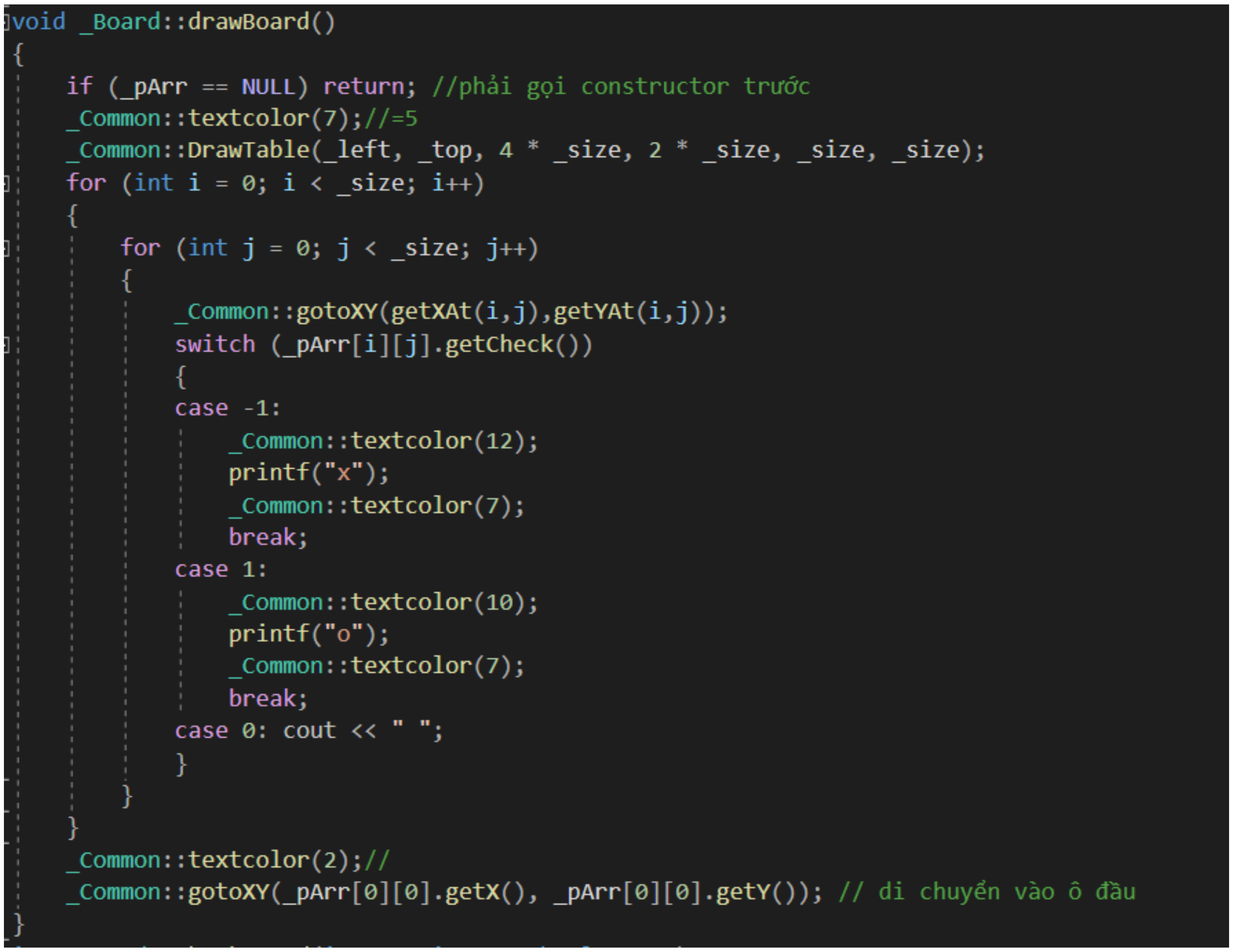
and main.cpp contains main function and program execution

3.2 Architecture and components

3.2.1 _Board.cpp

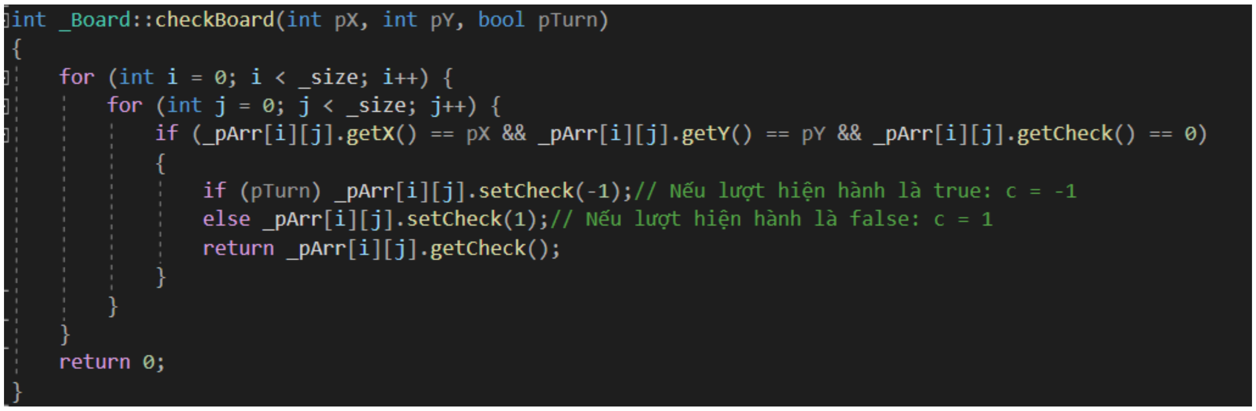
Includes Properties of Board Class

Drawboard(): draw board function



```
void _Board::drawBoard()
{
    if (_pArr == NULL) return; //phải gọi constructor trước
    _Common::textcolor(7);//=5
    _Common::DrawTable(_left, _top, 4 * _size, 2 * _size, _size, _size);
    for (int i = 0; i < _size; i++)
    {
        for (int j = 0; j < _size; j++)
        {
            _Common::gotoXY(getXAt(i,j),getYAt(i,j));
            switch (_pArr[i][j].getCheck())
            {
            case -1:
                _Common::textcolor(12);
                printf("x");
                _Common::textcolor(7);
                break;
            case 1:
                _Common::textcolor(10);
                printf("o");
                _Common::textcolor(7);
                break;
            case 0: cout << " ";
            }
        }
    }
    _Common::textcolor(2);//
    _Common::gotoXY(_pArr[0][0].getX(), _pArr[0][0].getY()); // di chuyển vào ô đầu
}
```

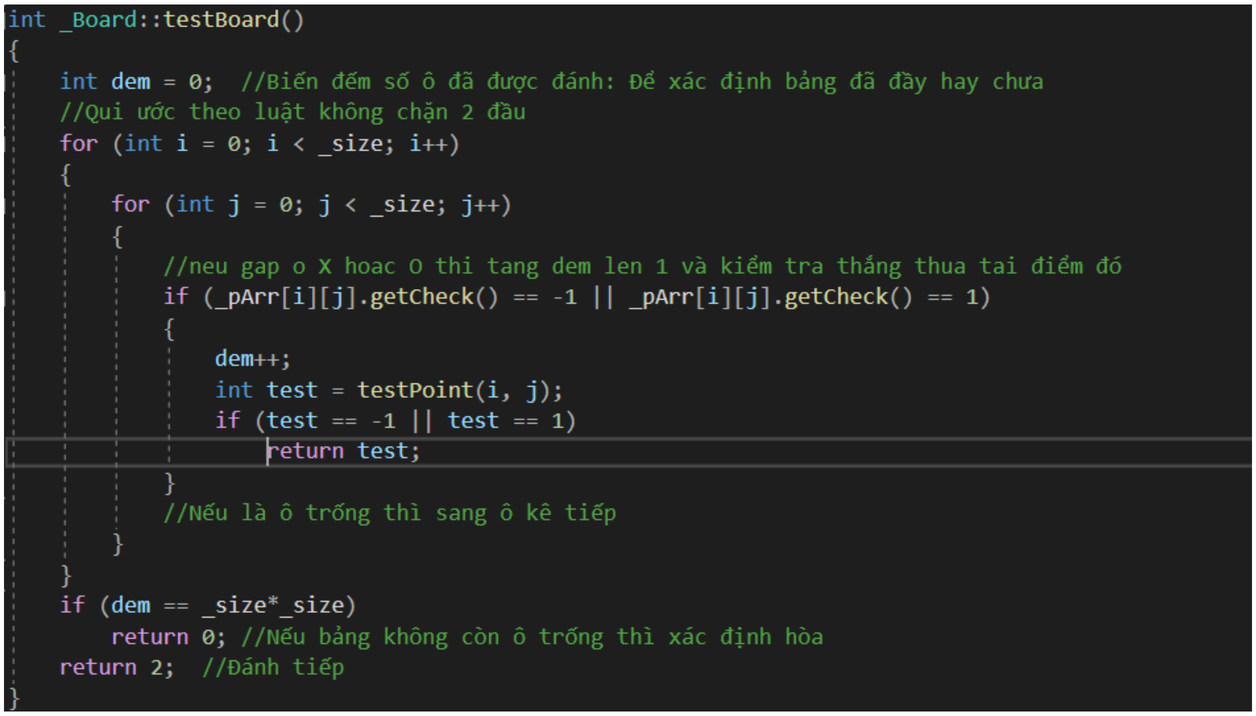
Checkboard(): Check board of function- to update the property _check at the position the player enters.



```
int _Board::checkBoard(int pX, int pY, bool pTurn)
{
    for (int i = 0; i < _size; i++) {
        for (int j = 0; j < _size; j++) {
            if (_pArr[i][j].getX() == pX && _pArr[i][j].getY() == pY && _pArr[i][j].getCheck() == 0)
            {
                if (pTurn) _pArr[i][j].setCheck(-1);// Nếu lượt hiện hành là true: c = -1
                else _pArr[i][j].setCheck(1);// Nếu lượt hiện hành là false: c = 1
                return _pArr[i][j].getCheck();
            }
        }
    }
    return 0;
}
```

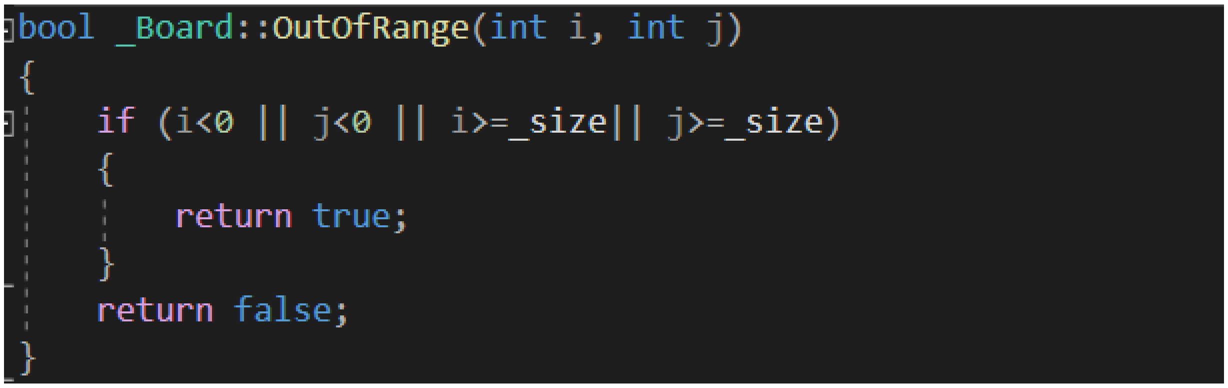
testBoard():

Depending on turn, the variable called _check is set to 1 or -1. Furthermore, we add function called "testBoard" to check the result (win, lose with caro rule). Below code is just a demo, you implement such that the function returns -1 (the 1st wins), 0 (draw), 1 (the 2nd wins) or 2 (no one win).



```
int _Board::testBoard()
{
    int dem = 0;  //Biến đếm số ô đã được đánh: Để xác định bảng đã đầy hay chưa
    //Qui ước theo luật không chặn 2 đầu
    for (int i = 0; i < _size; i++)
    {
        for (int j = 0; j < _size; j++)
        {
            //neu gap o X hoac O thi tang dem len 1 và kiểm tra thắng thua tai điểm đó
            if (_pArr[i][j].getCheck() == -1 || _pArr[i][j].getCheck() == 1)
            {
                dem++;
                int test = testPoint(i, j);
                if (test == -1 || test == 1)
                    return test;
            }
            //Nếu là ô trống thì sang ô kê tiếp
        }
    }
    if (dem == _size*_size)
        return 0; //Nếu bảng không còn ô trống thì xác định hòa
    return 2;  //Đánh tiếp
}
```

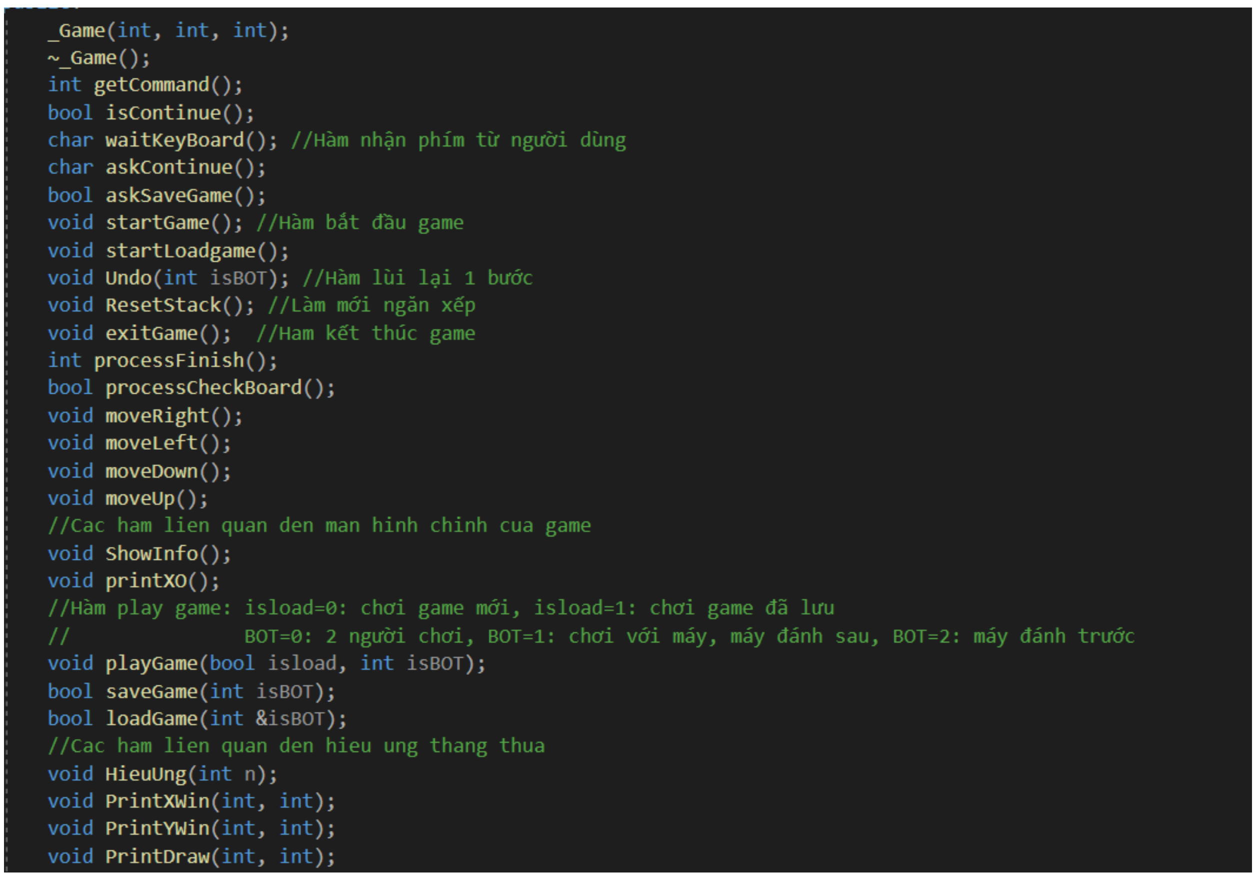
OutOfRange(): Return true if the number is just outside the actual range index


```
bool _Board::OutOfRange(int i, int j)
{
    if (i<0 || j<0 || i>=_size|| j>=_size)
    {
        return true;
    }
    return false;
}
```

3.2.2 _Game.cpp

Next, we create class _Game to execute the game. Class _Game includes a boardgame with _Board type, variable called _turn with bool type to represent two turns, current coordination (_x, _y) of cursor, variable called _command receiving the input-key from the players, and the variable called _loop to check the finish of game.

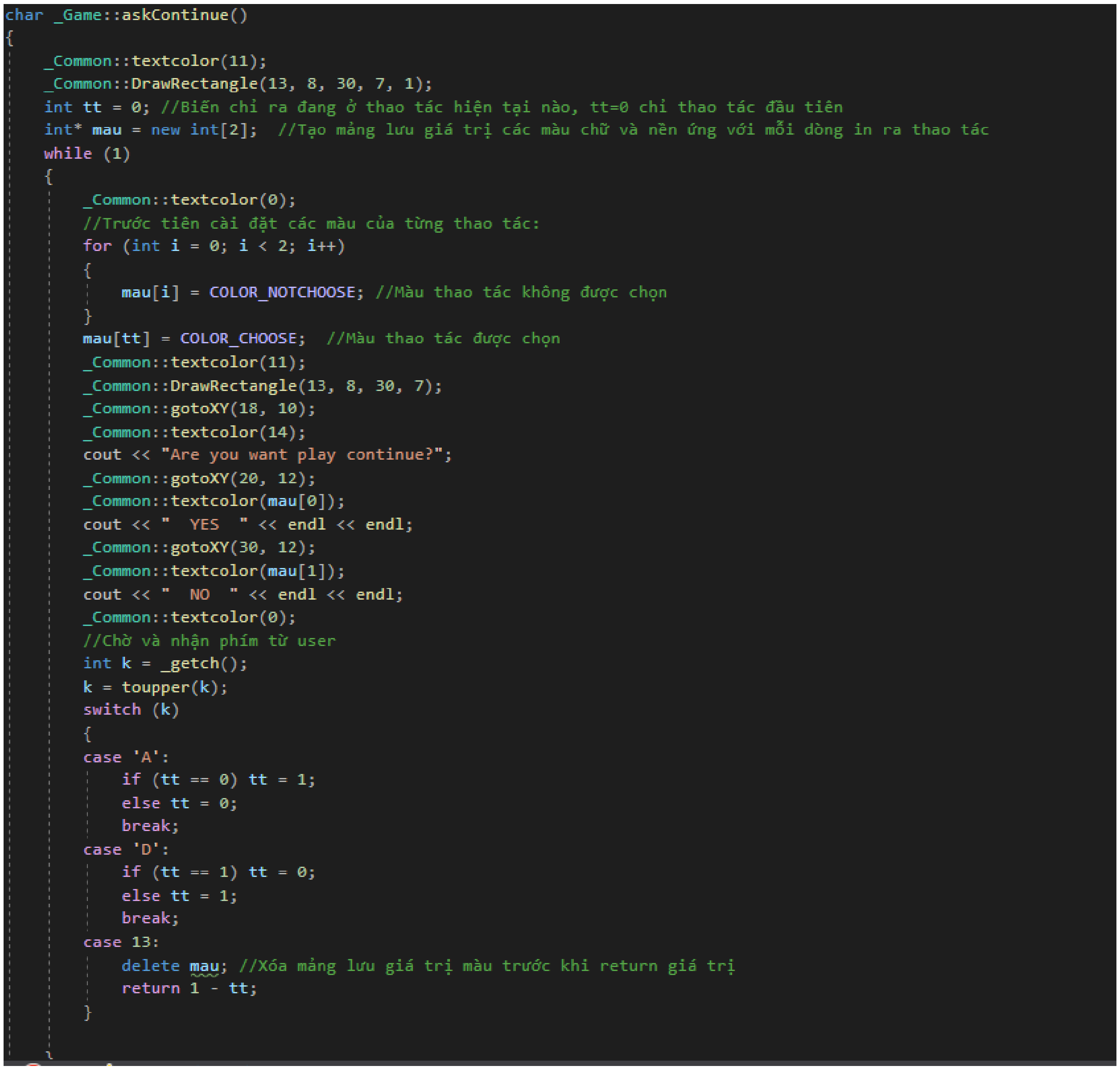
Implement properties of _Game class



```
    _Game(int, int, int);
    ~_Game();
    int getCommand();
    bool isContinue();
    char waitKeyBoard(); //Hàm nhận phím từ người dùng
    char askContinue();
    bool askSaveGame();
    void startGame(); //Hàm bắt đầu game
    void startLoadgame();
    void Undo(int isBOT); //Hàm lùi lại 1 bước
    void ResetStack(); //Làm mới ngăn xếp
    void exitGame();  //Ham kết thúc game
    int processFinish();
    bool processCheckBoard();
    void moveRight();
    void moveLeft();
    void moveDown();
    void moveUp();
    //Cac ham lien quan den man hinh chinh cua game
    void ShowInfo();
    void printXO();
    //Hàm play game: isload=0: chơi game mới, isload=1: chơi game đã lưu
    //                BOT=0: 2 người chơi, BOT=1: chơi với máy, máy đánh sau, BOT=2: máy đánh trước
    void playGame(bool isload, int isBOT);
    bool saveGame(int isBOT);
    bool loadGame(int &isBOT);
    //Cac ham lien quan den hieu ung thang thua
    void HieuUng(int n);
    void PrintXWin(int, int);
    void PrintYWin(int, int);
    void PrintDraw(int, int);
```

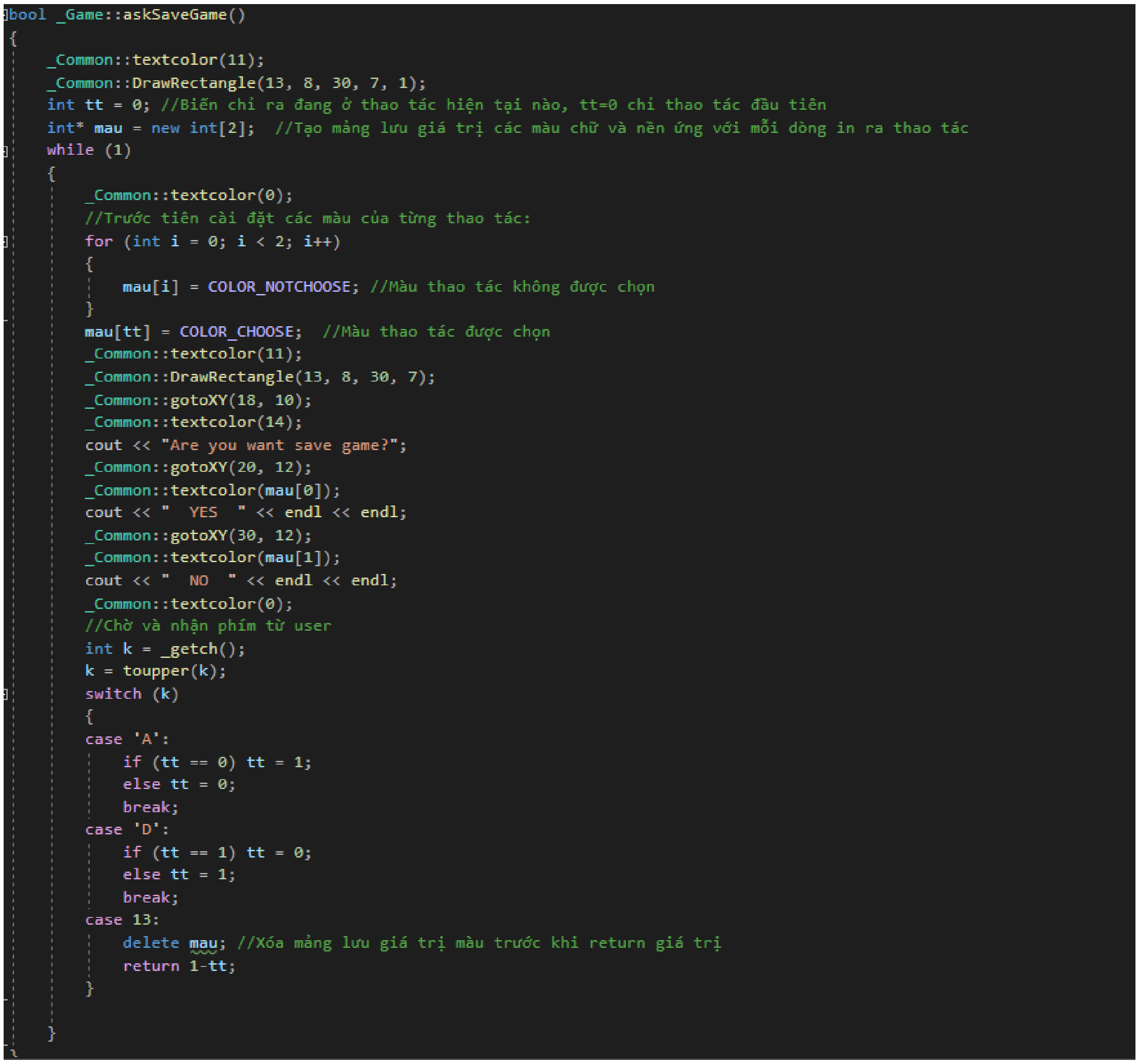
AskContinue():

The handler function asks if the player wants to continue playing



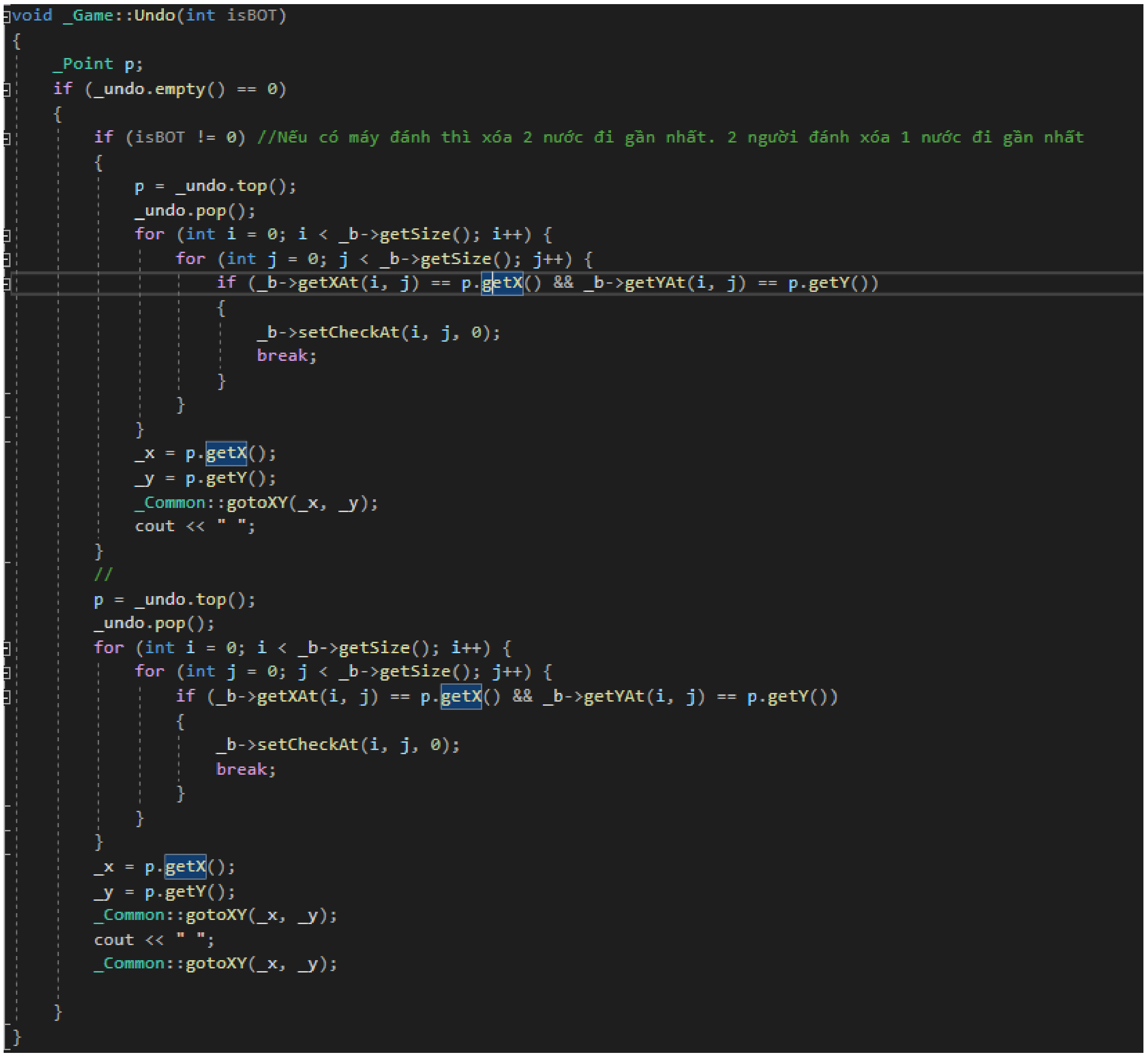
```
char _Game::askContinue()
{
    _Common::textcolor(11);
    _Common::DrawRectangle(13, 8, 30, 7, 1);
    int tt = 0; //Biến chỉ ra đang ở thao tác hiện tại nào, tt=0 chỉ thao tác đầu tiên
    int* mau = new int[2];  //Tạo mảng lưu giá trị các màu chữ và nền ứng với mỗi dòng in ra thao tác
    while (1)
    {
        _Common::textcolor(0);
        //Trước tiên cài đặt các màu của từng thao tác:
        for (int i = 0; i < 2; i++)
        {
            mau[i] = COLOR_NOTCHOOSE; //Màu thao tác không được chọn
        }
        mau[tt] = COLOR_CHOOSE;  //Màu thao tác được chọn
        _Common::textcolor(11);
        _Common::DrawRectangle(13, 8, 30, 7);
        _Common::gotoXY(18, 10);
        _Common::textcolor(14);
        cout << "Are you want play continue?";
        _Common::gotoXY(20, 12);
        _Common::textcolor(mau[0]);
        cout << "  YES  " << endl << endl;
        _Common::gotoXY(30, 12);
        _Common::textcolor(mau[1]);
        cout << "  NO  " << endl << endl;
        _Common::textcolor(0);
        //Chờ và nhận phím từ user
        int k = _getch();
        k = toupper(k);
        switch (k)
        {
        case 'A':
            if (tt == 0) tt = 1;
            else tt = 0;
            break;
        case 'D':
            if (tt == 1) tt = 0;
            else tt = 1;
            break;
        case 13:
            delete mau; //Xóa mảng lưu giá trị màu trước khi return giá trị
            return 1 - tt;
        }
```

askSaveGame(): the processing function asks the player if he wants to save



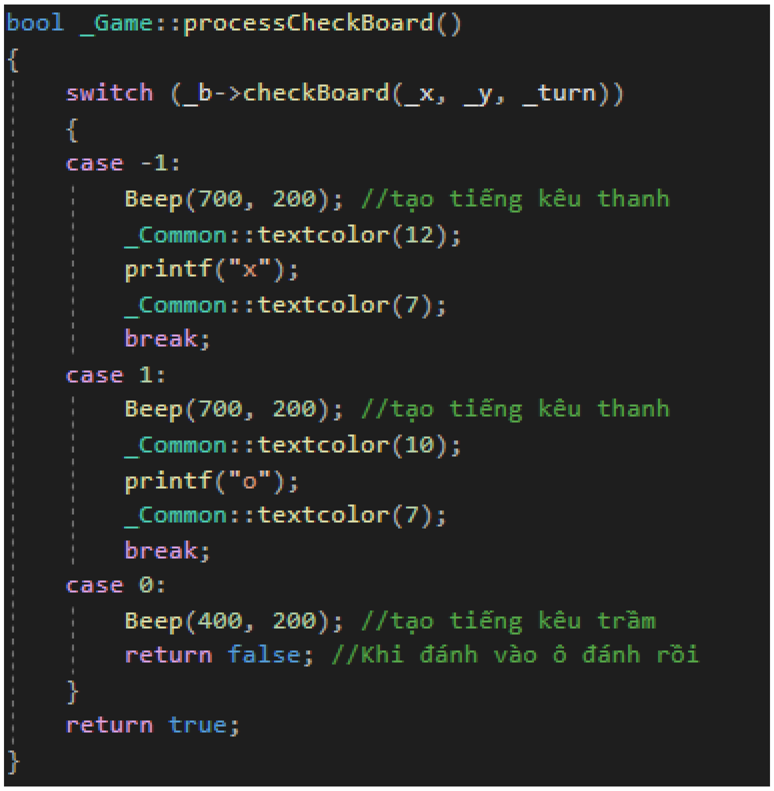
```
bool _Game::askSaveGame()
{
    _Common::textcolor(11);
    _Common::DrawRectangle(13, 8, 30, 7, 1);
    int tt = 0; //Biến chỉ ra đang ở thao tác hiện tại nào, tt=0 chỉ thao tác đầu tiên
    int* mau = new int[2];  //Tạo mảng lưu giá trị các màu chữ và nền ứng với mỗi dòng in ra thao tác
    while (1)
    {
        _Common::textcolor(0);
        //Trước tiên cài đặt các màu của từng thao tác:
        for (int i = 0; i < 2; i++)
        {
            mau[i] = COLOR_NOTCHOOSE; //Màu thao tác không được chọn
        }
        mau[tt] = COLOR_CHOOSE;  //Màu thao tác được chọn
        _Common::textcolor(11);
        _Common::DrawRectangle(13, 8, 30, 7);
        _Common::gotoXY(18, 10);
        _Common::textcolor(14);
        cout << "Are you want save game?";
        _Common::gotoXY(20, 12);
        _Common::textcolor(mau[0]);
        cout << "  YES  " << endl << endl;
        _Common::gotoXY(30, 12);
        _Common::textcolor(mau[1]);
        cout << "  NO  " << endl << endl;
        _Common::textcolor(0);
        //Chờ và nhận phím từ user
        int k = _getch();
        k = toupper(k);
        switch (k)
        {
        case 'A':
            if (tt == 0) tt = 1;
            else tt = 0;
            break;
        case 'D':
            if (tt == 1) tt = 0;
            else tt = 1;
            break;
        case 13:
            delete mau; //Xóa mảng lưu giá trị màu trước khi return giá trị
            return 1-tt;
        }

    }
}
```

Undo(): handle when the player hits back



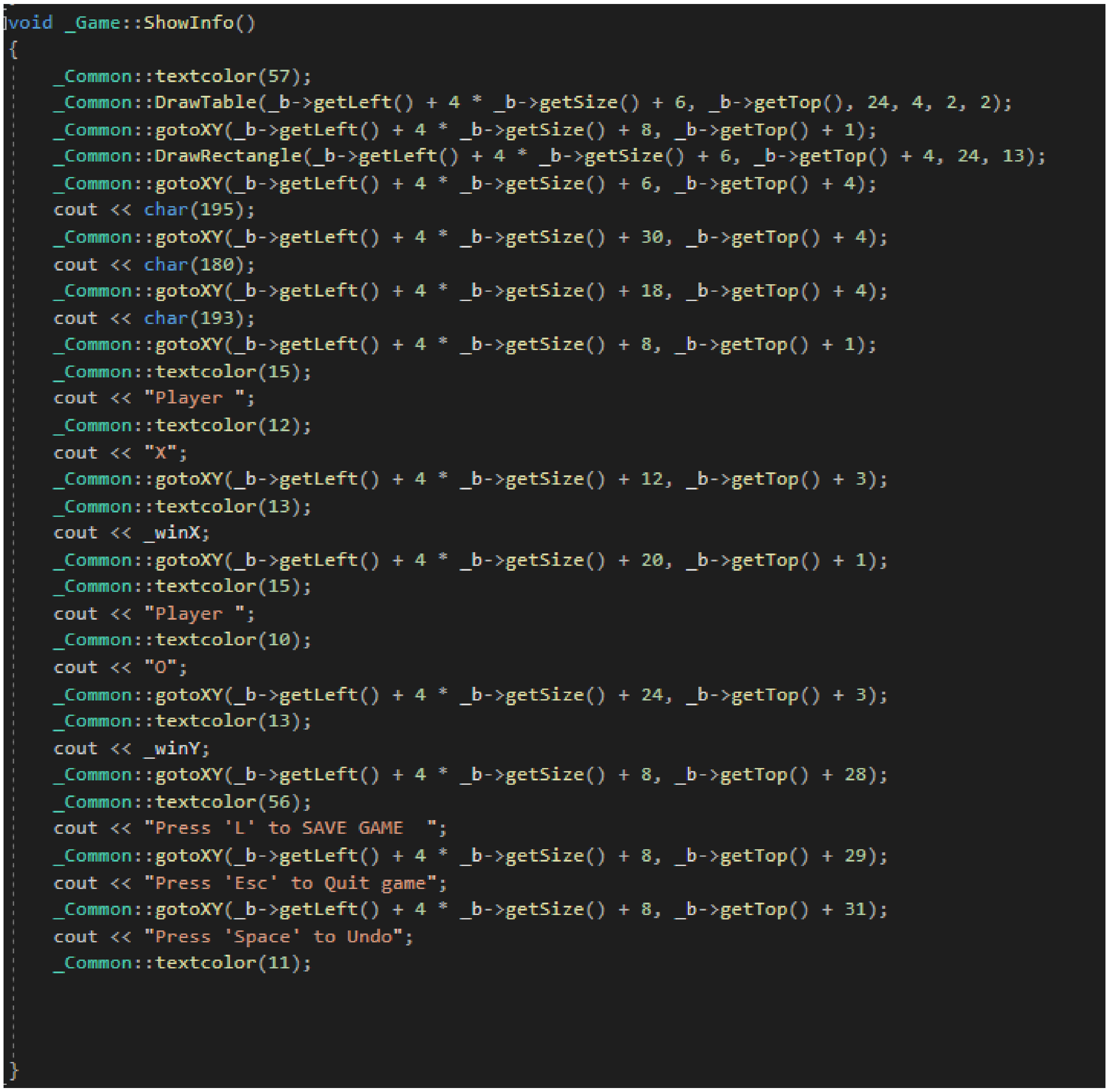
```
void _Game::Undo(int isBOT)
{
    _Point p;
    if (_undo.empty() == 0)
    {
        if (isBOT != 0) //Nếu có máy đánh thì xóa 2 nước đi gần nhất. 2 người đánh xóa 1 nước đi gần nhất
        {
            p = _undo.top();
            _undo.pop();
            for (int i = 0; i < _b->getSize(); i++) {
                for (int j = 0; j < _b->getSize(); j++) {
                    if (_b->getXAt(i, j) == p.getX() && _b->getYAt(i, j) == p.getY())
                    {
                        _b->setCheckAt(i, j, 0);
                        break;
                    }
                }
            }
            _x = p.getX();
            _y = p.getY();
            _Common::gotoXY(_x, _y);
            cout << " ";
        }
        //
        p = _undo.top();
        _undo.pop();
        for (int i = 0; i < _b->getSize(); i++) {
            for (int j = 0; j < _b->getSize(); j++) {
                if (_b->getXAt(i, j) == p.getX() && _b->getYAt(i, j) == p.getY())
                {
                    _b->setCheckAt(i, j, 0);
                    break;
                }
            }
        }
        _x = p.getX();
        _y = p.getY();
        _Common::gotoXY(_x, _y);
        cout << " ";
        _Common::gotoXY(_x, _y);

    }
}
```

processCheckBoard(): function that handles when the player hits, I print to the screen and the sound



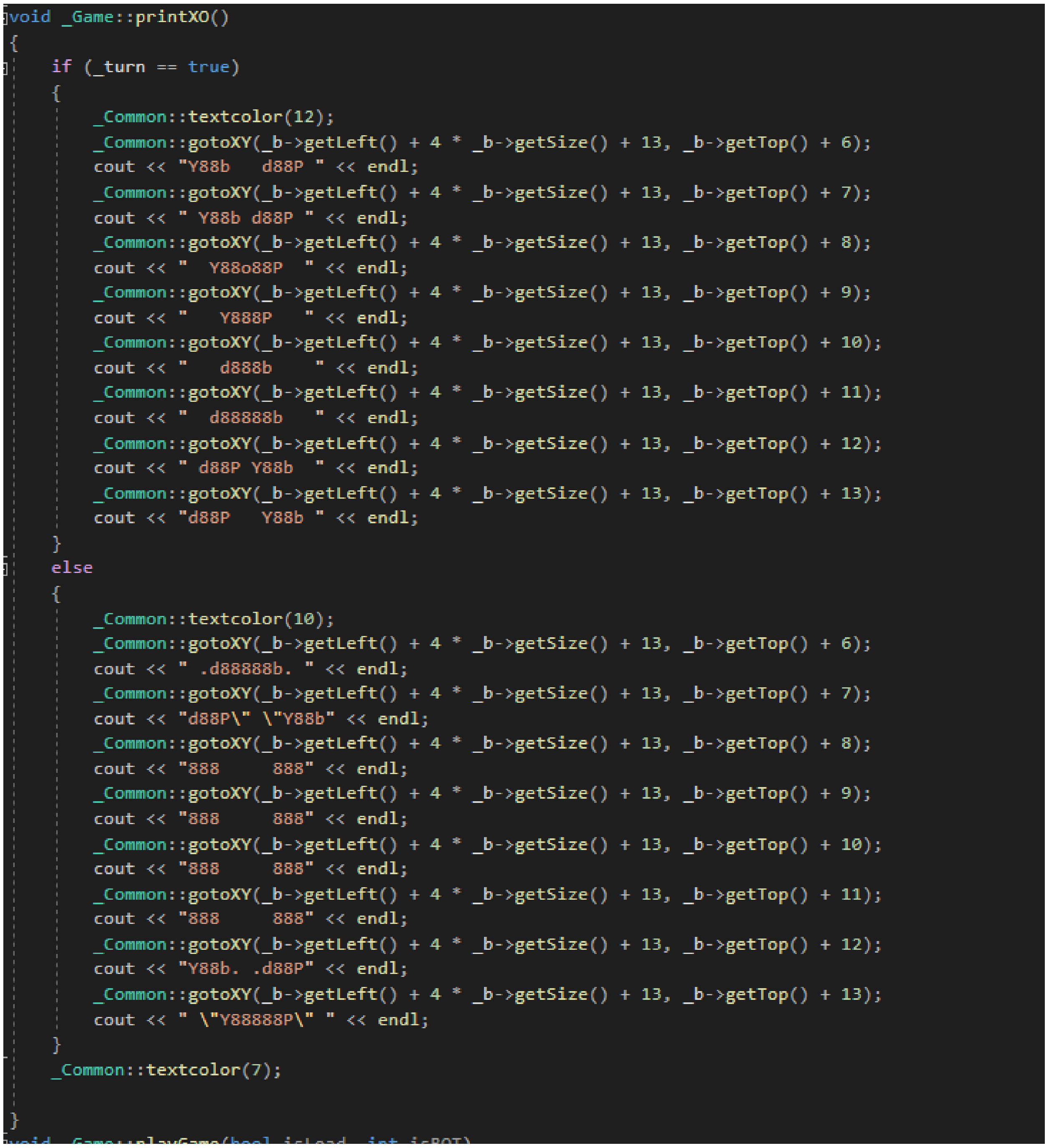
```
bool _Game::processCheckBoard()
{
    switch (_b->checkBoard(_x, _y, _turn))
    {
    case -1:
        Beep(700, 200); //tạo tiếng kêu thanh
        _Common::textcolor(12);
        printf("x");
        _Common::textcolor(7);
        break;
    case 1:
        Beep(700, 200); //tạo tiếng kêu thanh
        _Common::textcolor(10);
        printf("o");
        _Common::textcolor(7);
        break;
    case 0:
        Beep(400, 200); //tạo tiếng kêu trầm
        return false; //Khi đánh vào ô đánh rồi
    }
    return true;
}
```

ShowInfo(): show information during the match



```
void _Game::ShowInfo()
{
    _Common::textcolor(57);
    _Common::DrawTable(_b->getLeft() + 4 * _b->getSize() + 6, _b->getTop(), 24, 4, 2, 2);
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 8, _b->getTop() + 1);
    _Common::DrawRectangle(_b->getLeft() + 4 * _b->getSize() + 6, _b->getTop() + 4, 24, 13);
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 6, _b->getTop() + 4);
    cout << char(195);
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 30, _b->getTop() + 4);
    cout << char(180);
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 18, _b->getTop() + 4);
    cout << char(193);
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 8, _b->getTop() + 1);
    _Common::textcolor(15);
    cout << "Player ";
    _Common::textcolor(12);
    cout << "X";
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 12, _b->getTop() + 3);
    _Common::textcolor(13);
    cout << _winX;
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 20, _b->getTop() + 1);
    _Common::textcolor(15);
    cout << "Player ";
    _Common::textcolor(10);
    cout << "O";
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 24, _b->getTop() + 3);
    _Common::textcolor(13);
    cout << _winY;
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 8, _b->getTop() + 28);
    _Common::textcolor(56);
    cout << "Press 'L' to SAVE GAME  ";
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 8, _b->getTop() + 29);
    cout << "Press 'Esc' to Quit game";
    _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 8, _b->getTop() + 31);
    cout << "Press 'Space' to Undo";
    _Common::textcolor(11);



}
```

PrintXO():



```cpp
void _Game::printXO()
{
    if (_turn == true)
    {
        _Common::textcolor(12);
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 6);
        cout << "Y88b   d88P " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 7);
        cout << " Y88b d88P " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 8);
        cout << "  Y88o88P  " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 9);
        cout << "   Y888P   " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 10);
        cout << "   d888b    " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 11);
        cout << "  d88888b   " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 12);
        cout << " d88P Y88b  " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 13);
        cout << "d88P   Y88b " << endl;
    }
    else
    {
        _Common::textcolor(10);
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 6);
        cout << " .d88888b. " << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 7);
        cout << "d88P\" \"Y88b" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 8);
        cout << "888     888" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 9);
        cout << "888     888" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 10);
        cout << "888     888" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 11);
        cout << "888     888" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 12);
        cout << "Y88b. .d88P" << endl;
        _Common::gotoXY(_b->getLeft() + 4 * _b->getSize() + 13, _b->getTop() + 13);
        cout << " \"Y88888P\" " << endl;
    }
    _Common::textcolor(7);

}
```

And more…

Playgame(),Savegame():to save game ,loadgame():to load game,HieuUng(),printXWin(), printYWin(), printdraw(),…

3.2.3 _Common.cpp

Includes some general properties: `gotoXY(int pX, int pY)`

```
fixConsoleWindow()

SetWindow(int Width, int Height)

cleanscr()

textcolor(int x),
textcolorAt(int x, int y, char* a, int color)
DrawRectangle(int x, int y, int width, int height, bool insert)
DrawTable(int x, int y, int width, int height, int row, int col)
```

```
void _Common::gotoXY(int pX, int pY)
{
	COORD coord;
	coord.X = pX;
	coord.Y = pY;
	SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
void _Common::fixConsoleWindow()
{
	HWND consoleWindow = GetConsoleWindow();
	LONG style = GetWindowLong(consoleWindow, GWL_STYLE);
	style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME);
	SetWindowLong(consoleWindow, GWL_STYLE, style);
}
void _Common::SetWindow(int Width, int Height)
{
	_COORD coord;
	coord.X = Width;
	coord.Y = Height;

	_SMALL_RECT Rect;
	Rect.Top = 0;
	Rect.Left = 0;
	Rect.Bottom = Height - 1;
	Rect.Right = Width - 1;

	HANDLE Handle = GetStdHandle(STD_OUTPUT_HANDLE);      // Get Handle
	SetConsoleScreenBufferSize(Handle, coord);            // Set Buffer Size
	SetConsoleWindowInfo(Handle, TRUE, &Rect);            // Set Window Size
}

void _Common::cleanscr()
{
	HANDLE hOut;
	COORD Position;
	hOut = GetStdHandle(STD_OUTPUT_HANDLE);
	Position.X = 0;
	Position.Y = 0;
	SetConsoleCursorPosition(hOut, Position);
}
void _Common::textcolor(int x)
{
	HANDLE color;
	color = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(color, x);
}
void _Common::textcolorAt(int x, int y, char* a, int color)
{
	gotoXY(x, y);
	textcolor(color);
	cout << a;
	textcolor(7);
}
void _Common::DrawRectangle(int x, int y, int width, int height, bool insert)
{
	gotoXY(x, y);
	cout << char(218);
	gotoXY(x + width, y);
	cout << char(191);
	gotoXY(x, y + height);
	cout << char(192);
	gotoXY(x + width, y + height);
	cout << char(217);
	if (insert)
	{
		for (int i = x + 1; i <= x + width - 1; i++)
		{
			for (int j = y + 1; j <= y + height - 1; j++)
			{
				gotoXY(i, j);
				cout << " ";
			}
		}
	}
	gotoXY(x, y);
	for (int i = x + 1; i < x + width; i++)
	{
		gotoXY(i, y);
		cout << char(196);
	}
	gotoXY(x, y + height);
	for (int i = x + 1; i < x + width; i++)
	{
		gotoXY(i, y + height);
		cout << char(196);
	}
	gotoXY(x, y);
	for (int i = y + 1; i < y+height; i++)
	{
		gotoXY(x, i);
		cout << char(179);
	}
	gotoXY(x + width, y);
	for (int i = y + 1; i < y+height; i++)
	{
		gotoXY(x + width, i);
		cout << char(179);
	}
}
void _Common::DrawTable(int x, int y, int width, int height, int row, int col)
```

```
{
	int wid1 = width / col;
	int hei1 = height / row;
	_Common::gotoXY(x, y);
	cout << char(218);
	_Common::gotoXY(x + width, y);
	cout << char(191);
	_Common::gotoXY(x, y + height);
	cout << char(192);
	_Common::gotoXY(x + width, y + height);
	cout << char(217);
	for (int i = x + 1; i < x + width; i++)
	{
		_Common::gotoXY(i, y);
		if ((i - x) % wid1 == 0) cout << char(194);
		else cout << char(196);
		_Common::gotoXY(i, y + height);
		if ((i - x) % wid1 == 0) cout << char(193);
		else cout << char(196);
	}
	for (int i = y + 1; i < y + height; i++)
	{
		_Common::gotoXY(x, i);
		if ((i - y) % hei1 == 0) cout << char(195);
		else cout << char(179);
		_Common::gotoXY(x+width, i);
		if ((i - y) % hei1 == 0) cout << char(180);
		else cout << char(179);

	}
	for (int i = y + hei1; i < y + height; i += hei1)
	{
		for (int j = x + 1; j < x + width; j++)
		{
			_Common::gotoXY(j, i);
			if ((j - x) % wid1 == 0) cout << char(197);
			else cout << char(196);
		}
	}
	for (int i = x + wid1; i < x + width; i += wid1)
	{
		for (int j = y + 1; j < y + height; j++)
		{
			_Common::gotoXY(i, j);
			if ((j-y) % hei1 != 0) cout << char(179);
		}
	}
	}
```

3.2.4 _Point.cpp

_Point is array 2D, include (x,y) and bool check

```
#include "_Point.h"
```

```
_Point::_Point() { _x = _y = _check = 0; }
_Point::_Point(int pX, int pY)
{
	_x = pX;
	_y = pY;
	_check = 0;
}
_Point::_Point(int pX, int pY, int check)
{
	_x = pX;
	_y = pY;
	_check = check;
}
int _Point::getX() { return _x; }
int _Point::getY() { return _y; }
int _Point::getCheck() { return _check; }
void _Point::setX(int pX) { _x = pX; }
void _Point::setY(int pY) { _y = pY; }
bool _Point::setCheck(int pCheck) {
	if (pCheck == -1 || pCheck == 1 || pCheck == 0) {
		_check = pCheck;
		return true;
	}
	return false;
	}
```

3.2.5 _Bot.cpp

_Bot class is a class that contains attributes that help players fight against the computer, using the heurictis algorithm, the user can choose to play first or hit later.

Đánh giá bảng giá trị thông qua 4 hướng: `void _BOT::EvaluateBoard(int checkObject, int checkBOT, int checkPlayer)`

```
void _BOT::EvaluateBoard(int checkObject, int checkBOT, int checkPlayer)
{
	// Khởi tạo giá trị mặc định cho bảng
	eBoard.ResetValue();

	// Đánh giá bảng giá trị thông qua 4 hướng
	EvaluateBoard(checkObject, checkBOT, checkPlayer, 1, 0); //Đường ngang
	EvaluateBoard(checkObject, checkBOT, checkPlayer, 0, 1); //Đường dọc
	EvaluateBoard(checkObject, checkBOT, checkPlayer, 1, 1);  //Đường chéo chính
	EvaluateBoard(checkObject, checkBOT, checkPlayer, 1, -1);  //Đường chéo phụ
	}


void _BOT::EvaluateBoard(int checkObject, int checkBOT, int checkPlayer, int xvl, int
yvl)
{
	int iBOT;
	// Số quân đồng minh
	int iPlayer;
	// Số quân địch
	_Point lowerBound(0, 0);
	// Tọa độ mặc định bắt đầu vòng lặp
	_Point upperBound(Board->getSize(), Board->getSize()); // Tọa độ mặc định kết thúc
vòng lặp
	// Hiệu chỉnh lại tọa độ bắt đầu và kết thúc tùy theo hướng đang đánh giá
	if (xvl == -1)
		lowerBound.setX(lowerBound.getX() + 4);
	if (yvl == -1)
		lowerBound.setY(lowerBound.getY() + 4);
	if (xvl == 1)
		upperBound.setX(upperBound.getX() - 4);
	if (yvl == 1)
		upperBound.setY(upperBound.getY() - 4);
	for (int y = lowerBound.getY(); y < upperBound.getY(); y++)
	{
		for (int x = lowerBound.getX(); x < upperBound.getX(); x++)
		{
			iBOT = 0;
			iPlayer = 0;
			// Đếm theo hướng xem có bao nhiêu đồng minh, quân địch
			for (int i = 0; i < 5; i++)
			{
				if (Board->getCheckAt(y + yvl * i, x + xvl * i) == checkBOT)
					iBOT++;

				if (Board->getCheckAt(y + yvl * i, x + xvl * i) ==
checkPlayer)

					iPlayer++;
			}
			if (iBOT * iPlayer == 0 && iBOT != iPlayer)
			{
				for (int i = 0; i < 5; i++)
				{
					if (Board->getCheckAt(y + yvl * i, x + xvl * i) == 0)
//Nếu là ô trống

					{
						if (iBOT == 0)
						{
							// Dựa vào đối tượng đang xét là ai để
đánh giá điểm hợp lệ

							if (checkObject == checkBOT)
								eBoard._Value[y + yvl * i][x + xvl

* i] += defenceScore[iPlayer];

							else
								eBoard._Value[y + yvl * i][x + xvl

* i] += attackScore[iPlayer];

						}
						if (iPlayer == 0)
						{
							// Dựa vào đối tượng đang xét là ai để
đánh giá điểm hợp lệ

							if (checkObject == checkPlayer)
								eBoard._Value[y + yvl * i][x + xvl

* i] += defenceScore[iBOT];

							else
```

```
                                                        eBoard._Value[y + yvl * i][x + xvl
* i] += attackScore[iBOT];
                                                }
                                                // 4 quân trên một đường (có thể của địch hoặc
của ta) thì ưu tiên cho nước đi này
                                                if (!(Board->OutOfRange(x + xvl * (i - 1), y +
yvl * (i - 1)) || Board->OutOfRange(x + xvl * (i + 1), y + yvl * (i + 1))))
                                                {
                                                        if ((iBOT == 4 || iPlayer == 4) && Board-
>getCheckAt(y + yvl * (i - 1), x + xvl * (i - 1)) == 0 && Board->getCheckAt(y + yvl * (i
+ 1), x + xvl * (i + 1)) == 0)
                                                        eBoard._Value[y + yvl * i][x + xvl
* i] *= 3;
                                                }
                                        }
                                }
                        }

                }
        }
        }
```

3.2.6 _EvalutingBoard.cpp

Contain properties: `_EvaluatingBoard, ResetValue(),GetIndexMax()`

```
_EvaluatingBoard::_EvaluatingBoard()
{
        _size = 0;
}
_EvaluatingBoard::_EvaluatingBoard(int size)
{
        _size = size;
        _Value = new long *[_size];

        for (int y = 0; y < _size; y++)
                _Value[y] = new long[_size];
}
// Đưa tất cả các ô giá trị về lại 0
void _EvaluatingBoard::ResetValue()
{
        for (int y = 0; y < _size; y++)
                for (int x = 0; x < _size; x++)
                        _Value[y][x] = 0;
}
_Point _EvaluatingBoard::GetIndexMax()
{
        _Point index(0, 0);
        for (int y = 0; y < _size; y++)
                for (int x = 0; x < _size; x++)
                {
                        if (_Value[index.getY()][index.getX()] < _Value[y][x])
                                index = _Point(x, y);
                }
```

```
	return index;
}
```

3.2.7 _Menu.cpp

Menu class is the class that contains the properties related to the menu handling.

Includes : `Menu(),MenuBot(), Logo2() and getkey()`

```
int _Menu::Menu()
{
	int tt = 0; //Biến chỉ ra đang ở thao tác hiện tại nào, tt=0 chỉ thao tác đầu tiên
	int n = 4;//Số thao tác có trong menu
	//int* mau = new int[n];  //Tạo mảng lưu giá trị các màu chữ và nền ứng với mỗi
dòng in ra thao tác
	int mau[4];
	HANDLE Color;
	Color = GetStdHandle(STD_OUTPUT_HANDLE);
	int i = 0;
	while (1)
	{
		SetConsoleTextAttribute(Color, 0);
		//Trước tiên cài đặt các màu của từng thao tác:
		for (int i = 0; i < n; i++)
		{
			mau[i] = COLOR_NOTCHOOSE; //Màu thao tác không được chọn
		}
		mau[tt] = COLOR_CHOOSE;  //Màu thao tác được chọn
		_Common::cleanscr();
		cout << endl << endl << endl << endl << endl;
		LOGO2();
		cout << endl << endl << endl << endl << endl;
		//In MENU với list các thao tác
		cout << "                                         ";
		SetConsoleTextAttribute(Color, mau[0]);
		cout<<" 2 PLAYER MODE  " << endl << endl;
		SetConsoleTextAttribute(Color, 0);
		cout << "                                         ";
		SetConsoleTextAttribute(Color, mau[1]);
		cout <<" COMPUTER MODE  " << endl << endl;
		SetConsoleTextAttribute(Color, 0);
		cout << "                                         ";
		SetConsoleTextAttribute(Color, mau[2]);
		cout<<" LOAD GAME      " << endl << endl;
		SetConsoleTextAttribute(Color, 0);
		cout << "                                         ";
		//Chờ và nhận phím từ user
		SetConsoleTextAttribute(Color, mau[3]);
		cout << "     EXIT       " << endl;
		int k = _getch();
		CONTROL control = getKey(k);
		PlaySound(TEXT("jump.wav"), NULL, SND_FILENAME | SND_ASYNC);
		switch (control)
		{
		case UP:
			if (tt == 0) tt = n - 1;
```

```
				else tt--;
				break;
			case DOWN:
				if (tt == n - 1) tt = 0;
				else tt++;
				break;
			case ENTER:
				//delete mau; //Xóa mảng lưu giá trị màu trước khi return giá trị
				return tt;
			}

		}
		}

CONTROL _Menu::getKey(int k)
{

		switch (toupper(k))
		{
		case 'W': return UP;
		case 'S': return DOWN;
		case 'D': return RIGHT;
		case 'A': return LEFT;
		}
		if (k == 224)
		{
			char c;
			c = _getch();
			switch (c)
			{
			case 72: return UP;
			case 80: return DOWN;
			case 77: return RIGHT;
			case 75: return LEFT;
			}
		}
		else if (k == 13) return ENTER;
		return NOTHING;
		}
```

4. Basic features of the project

2 player

computer player

Save/load game

Undo move

Exit

Save/load game

use fstream object to store current data, save to file the basic properties: chessboard size, chessboard coordinates, current cursor coordinates, player score, status value per cell, variable that identifies the machine or the player.

when exiting the game, the player will be asked if he wants to save the game

To continue playing the old game, the player must enter the correct path

Undo move

Press 'space' to go back one move, for machines, it will go back 2 moves

Use a stack data structure to store previous moves