# BÁO CÁO

Mục lục

[BÁO CÁO 1](#_Toc184072560)

[A. Giải thích các file Header 2](#_Toc184072561)

[1. Console.h: 2](#_Toc184072562)

[1.1Hàm clearScreen() 2](#_Toc184072563)

[1.2 Hàm moveSound() 2](#_Toc184072564)

[1.3 Hàm choseSound() 2](#_Toc184072565)

[1.4 Hàm GotoXY(int column, int line) 2](#_Toc184072566)

[1.5 Hàm setColor(int color) 2](#_Toc184072567)

[1.6 Hàm ShowBlinkingCursor(bool isVisible) 3](#_Toc184072568)

[1.7 Hàm FixConsoleWindow() 3](#_Toc184072569)

[1.8 Hàm setConsoleWindow(int w, int h) 3](#_Toc184072570)

[2. Chessboard.h 3](#_Toc184072571)

[2.1. Xác định giá trị quân cờ: 3](#_Toc184072572)

[2.2. Biến toàn cục và cấu trúc dữ liệu: 4](#_Toc184072573)

[2.3. Các chức năng chính: 4](#_Toc184072574)

[2.4. Một số hàm phụ trợ khác: 5](#_Toc184072575)

[2.5. Các tính năng còn lại: 5](#_Toc184072576)

[3. Bot.h: 5](#_Toc184072577)

[3.1. Hàm BotMove 6](#_Toc184072578)

[3.2. Hàm PlayWithBot 6](#_Toc184072579)

# Giải thích các file Header (.h)

1. Console.h: Chứa các hàm hỗ trợ thao tác với console trên hệ điều hành Windows và có thể một số đoạn mã cũng hỗ trợ hệ điều hành khác (như Linux/Mac).

### 1.1Hàm clearScreen()

* **Chức năng**: Xóa màn hình console.
* **Cách hoạt động**:
  + Sử dụng lệnh hệ thống phù hợp dựa trên hệ điều hành.
  + Windows: Dùng lệnh cls.
  + Linux/Mac: Dùng lệnh clear.

### 1.2 Hàm moveSound()

* **Chức năng**: Phát âm thanh khi di chuyển (move).
* **Cách hoạt động**:
  + Sử dụng hàm PlaySound (Windows API) để phát tệp âm thanh move.wav ở chế độ bất đồng bộ (SND\_ASYNC).

### 1.3 Hàm choseSound()

* **Chức năng**: Phát âm thanh khi chọn (chose).
* **Cách hoạt động**:
  + Tương tự như moveSound() nhưng sử dụng tệp âm thanh chose.wav.

### 1.4 Hàm GotoXY(int column, int line)

* **Chức năng**: Di chuyển con trỏ đến vị trí chỉ định trên màn hình console.
* **Cách hoạt động**:
  + Sử dụng hàm SetConsoleCursorPosition từ Windows API.
  + Tọa độ (x, y) được truyền qua tham số column (cột) và line (dòng).

### 1.5 Hàm setColor(int color)

* **Chức năng**: Thay đổi màu chữ trong console.
* **Cách hoạt động**:
  + Sử dụng hàm SetConsoleTextAttribute từ Windows API để thay đổi thuộc tính màu.

### 1.6 Hàm ShowBlinkingCursor(bool isVisible)

* **Chức năng**: Hiển thị hoặc ẩn con trỏ nhấp nháy trên console.
* **Cách hoạt động**:
  + Lấy thông tin con trỏ hiện tại bằng GetConsoleCursorInfo.
  + Thay đổi thuộc tính hiển thị (bVisible) và kích thước con trỏ (dwSize) dựa trên tham số isVisible.

### 1.7 Hàm FixConsoleWindow()

* **Chức năng**: Vô hiệu hóa khả năng thay đổi kích thước cửa sổ console.
* **Cách hoạt động**:
  + Dùng Windows API để lấy và chỉnh sửa thuộc tính GWL\_STYLE của cửa sổ console.
  + Xóa các thuộc tính cho phép phóng to (WS\_MAXIMIZEBOX) và thay đổi kích thước (WS\_THICKFRAME).

### 1.8 Hàm setConsoleWindow(int w, int h)

* **Chức năng**: Thiết lập kích thước cửa sổ console.
* **Cách hoạt động**:
  + Sử dụng hàm MoveWindow từ Windows API.
  + Nhận thông tin tọa độ hiện tại của cửa sổ console, sau đó thay đổi chiều rộng (w) và chiều cao (h) theo tham số.

## Chessboard.h

### 2.1. Xác định giá trị quân cờ:

* X() và O():
  + Hai hàm này xác định ký tự được hiển thị trên bàn cờ cho hai người chơi (X và O) cùng với việc thay đổi màu của chúng bằng hàm setColor().

### 2.2. Biến toàn cục và cấu trúc dữ liệu:

* \_POINT \_A[BOARD\_SIZE][BOARD\_SIZE]: Ma trận lưu trạng thái của từng ô trên bàn cờ.
  + x, y: Tọa độ trên màn hình console.
  + c: Trạng thái ô (0 - chưa đánh, 1 - quân O, -1 - quân X).
* \_TURN: Biến cờ xác định lượt của người chơi.
* \_X, \_Y: Vị trí hiện tại của con trỏ trên bàn cờ.

### 2.3. Các chức năng chính:

**a. Lưu và tải game:**

* **SaveGame()**:
  + Lưu trạng thái hiện tại của bàn cờ và lượt chơi vào file.
  + Dùng vòng lặp để duyệt toàn bộ ma trận \_A và ghi giá trị vào file.
* **LoadGame()**:
  + Đọc trạng thái từ file, khôi phục bàn cờ và lượt chơi.
  + Dựa vào giá trị đọc được, tái tạo bàn cờ trên màn hình.

**b. Khởi tạo và vẽ bàn cờ:**

* **ResetData()**:
  + Đặt lại dữ liệu ban đầu cho ma trận \_A.
  + Xác định tọa độ trên console và đặt trạng thái mỗi ô là 0.
* **DrawBoard(int pSize)**:
  + Dùng các ký tự đặc biệt để vẽ bàn cờ trên console.
  + Gọi GotoXY() để di chuyển con trỏ và printf() để in ký tự.

**c. Quản lý lượt chơi và giao diện:**

* **vienPlayer()**:
  + Vẽ khung giao diện cho thông tin của hai người chơi (X và O).
* **turnPlayer()**:
  + Thay đổi màu sắc và trạng thái để hiển thị lượt chơi hiện tại.
* **StartGame()**:
  + Khởi tạo dữ liệu và vẽ bàn cờ mới để bắt đầu trò chơi.

**d. Xử lý kết thúc trận:**

* **ProcessFinish(int pWhoWin)**:
  + Xác định người thắng, người thua hoặc hòa dựa trên giá trị trả về.
  + Hiển thị thông báo và trả về kết quả.
* **AskContinue()**:
  + Hỏi người chơi có muốn tiếp tục chơi hay không.

**e. Kiểm tra chiến thắng:**

* **CheckWin(int player)**:
  + Kiểm tra xem người chơi player (X hoặc O) đã thắng hay chưa.
  + Duyệt qua các hàng, cột và đường chéo để tìm chuỗi 5 quân liên tiếp.

### 2.4. Một số hàm phụ trợ khác:

* **ExitGame()**:
  + Dọn dẹp dữ liệu trước khi thoát game.
* **GabageCollect()**:
  + Có thể dùng để giải phóng bộ nhớ (nếu cần).

### 2.5. Các tính năng còn lại:

* Hiển thị giao diện console sinh động (màu sắc, khung viền).
* Hỗ trợ lưu và tải game, giúp người chơi tiếp tục từ vị trí đã lưu.
* Hệ thống di chuyển, kiểm tra điều kiện thắng/thua/hòa.

1. Bot.h: được thiết kế để tạo chế độ chơi "đánh với máy" trong trò chơi cờ caro. Trong đó:

* **Người chơi** thực hiện thao tác bằng các phím mũi tên (di chuyển con trỏ) và phím Enter để đánh dấu ô.
* **Máy (Bot)** đánh ngẫu nhiên vào một ô trống trên bàn cờ.

### 3.1. Hàm BotMove

Hàm này thực hiện logic để máy đánh vào một ô ngẫu nhiên.

- **Mục đích**:

* Chọn ngẫu nhiên một ô trống.
* Đánh dấu nước đi của máy trên ô đó.

- **Cách hoạt động**:

* Dùng rand() để tạo tọa độ (x, y) ngẫu nhiên.
* Nếu ô (x, y) không trống (\_A[x][y].c != 0), tiếp tục random đến khi tìm được ô trống.
* Ghi giá trị 1 vào ô (máy đánh "O").
* Dùng GotoXY() để di chuyển con trỏ tới vị trí của ô trên console và in ra chữ O.

### 3.2. Hàm PlayWithBot

Hàm chính để điều khiển luồng chơi giữa người và máy.

1.1 **Khởi tạo**:

* Gọi StartGame() để thiết lập bàn cờ và các biến cần thiết.
* x, y: Tọa độ ban đầu của con trỏ (bắt đầu ở góc trên trái bàn cờ).
* isPlaying: Biến kiểm soát vòng lặp trò chơi.
* result: Lưu kết quả kiểm tra trạng thái bàn cờ (thắng/thua/hòa).

1.2 **Vòng lặp chính**:

* Chạy liên tục cho đến khi trò chơi kết thúc (isPlaying = false).

1.3 **Lượt của người chơi**:

* Người chơi dùng các phím:
  + **Mũi tên lên/xuống/trái/phải** để di chuyển con trỏ trên bàn cờ.
  + **Phím Enter** để chọn ô hiện tại. Chỉ cho phép đánh vào ô trống (\_A[x][y].c == 0).
* Nước đi hợp lệ sẽ được ghi vào bàn cờ và hiển thị.

1.4 **Lượt của máy**:

* Gọi hàm BotMove() để thực hiện nước đi ngẫu nhiên của máy.

1.5 **Kiểm tra trạng thái bàn cờ**:

* Gọi TestBoard() để kiểm tra kết quả sau mỗi lượt (thắng/thua/hòa).
* Nếu kết thúc trò chơi, vòng lặp dừng (isPlaying = false).

1.6 **Hiển thị kết quả**:

* In thông báo thắng/thua/hòa sau khi kết thúc trò chơi.

# Giải thích các file Source (.cpp)

## 3.Bot.cpp

### 3.1 Hàm BotMove

|  |  |
| --- | --- |
| Dòng |  |
| 1 | void BotMove() { |
| 2 | srand(static\_cast<unsigned>(time(0))); // Khởi tạo seed cho hàm random |
| 3 | int x, y; |
| 4 | do { |
| 5 | x = rand() % BOARD\_SIZE; // Random tọa độ x |
| 6 | y = rand() % BOARD\_SIZE; // Random tọa độ y |
| 7 | } while (\_A[x][y].c != 0); // Kiểm tra ô có trống hay không |
| 8 |  |
| 9 | \_A[x][y].c = 1; // Máy luôn đánh O (giá trị 1) |
| 10 | GotoXY(\_A[x][y].x, \_A[x][y].y); |
| 11 | cout << O(); // Hiển thị nước đi của máy thông qua hàm O() |
| 12 | } |

### Hàm PlayWithBot

|  |  |
| --- | --- |
| Dòng |  |
| 1 | void PlayWithBot() { |
| 2 | ShowBlinkingCursor(true); |
| 3 | system("color 70"); |
| 4 | fixConsoleWindow(); |
| 5 | StartGame(); // Khởi tạo bàn cờ |
| 6 | int x = 0, y = 0; // Tọa độ ban đầu của con trỏ |
| 7 | bool isPlaying = true; |
| 8 | int result; |
| 9 |  |
| 10 | while (isPlaying) { |
| 11 | if (\_TURN) { |
| 12 | // Lượt người chơi |
| 13 | bool validMove = false; |
| 14 | while (!validMove) { |
| 15 | GotoXY(\_A[x][y].x, \_A[x][y].y); // Di chuyển con trỏ đến vị trí hiện tại |
| 16 | char key = \_getch(); // Đợi người chơi nhập phím |
| 17 |  |
| 18 | switch (key) { |
| 19 | case 72: // Mũi tên lên |
| 20 | if (x > 0) {x--; moveSound();} |
| 21 | break; |
| 22 | case 80: // Mũi tên xuống |
| 23 | if (x < BOARD\_SIZE - 1) {x++; moveSound()} |
| 24 | break; |
| 25 | case 75: // Mũi tên trái |
| 26 | if (y > 0) {y--; moveSound();} |
| 27 | break; |
| 28 | case 77: // Mũi tên phải |
| 29 | if (y < BOARD\_SIZE - 1) {y++; moveSound()} |
| 30 | break; |
| 31 | case 13: // Phím Enter |
| 32 | if (\_A[x][y].c == 0) { // Nếu ô trống |
| 33 | \_A[x][y].c = -1; // Người chơi đánh X (giá trị -1) |
| 34 | GotoXY(\_A[x][y].x, \_A[x][y].y); |
| 35 | cout << X(); // Hiển thị nước đi thông qua hàm X() |
| 36 | choseSound(); |
| 37 | validMove = true; // Thoát khỏi vòng lặp |
| 38 | } |
| 39 | break; |
| 40 | default: |
| 41 | break; |
| 42 | } |
| 43 | } |
| 44 | \_TURN = false; |
| 45 | } else { |
| 46 | // Lượt máy |
| 47 | BotMove(); |
| 48 | \_TURN = true; |
| 49 | } |
| 50 |  |
| 51 | // Kết thúc trò chơi |
| 52 | GotoXY(0, BOARD\_SIZE \* 2 + 3); |
| 53 | if (result == -1) { |
| 54 | cout << "You win!"; |
| 55 | } else if (result == 1) { |
| 56 | cout << "Bot wins!"; |
| 57 | } else { |
| 58 | cout << "It's a draw!"; |
| 59 | } |
| 60 | cout << endl; |
| 61 | system("pause"); |
| 62 | } |