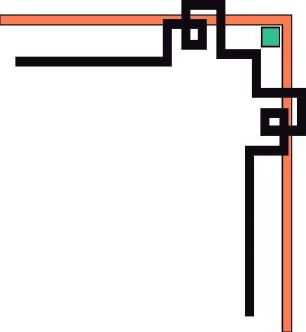
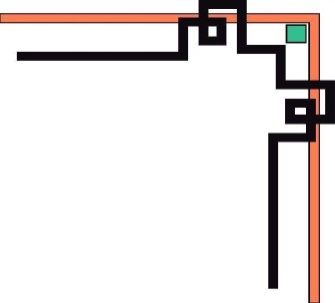
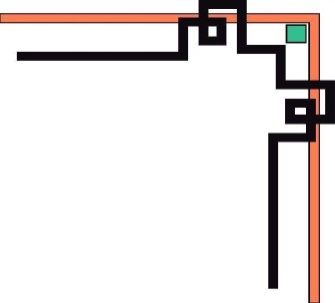
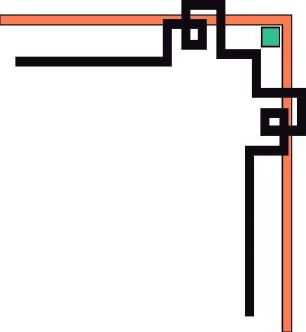
**BỘ GIÁO DỤC VÀ ĐÀO TẠO**



**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**🕯✡🕮🕮✡🕯**

****

**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**VIẾT CHƯƠNG TRÌNH GAME CARO TRÊN**

**MÀN HÌNH CONSOLE BẰNG NGÔN NGỮ C++**

**GVHD: THS. TRẦN CÔNG TÚ**

**SVTH: MSSV**

**NGUYỄN NGỌC PHƯƠNG MAI 16110150**

**NGÔ ĐÌNH HẢI 16110064**

**LỚP THỨ 6 – TIẾT 12345**

**DASA240179\_17\_1\_01CLC**

**TP.Hồ Chí Minh, tháng 12 năm 2017**

**NHẬN XÉT CỦA GIẢNG VIÊN**

*TP.Hồ Chí Minh, ngày tháng 12 năm 2017*

**Giáo viên hướng dẫn**

(Ký và ghi rõ họ tên)

**TS.Trần Công Tú**

**I/ MÔ TẢ ĐỀ TÀI**

## Mục đích

Game cờ Caro dùng cho mục đích giải trí, thư giãn tinh thần. Đây là một trò chơi thú vị, bổ ích, dễ chơi và giúp con người nâng cao việc tư duy tính toán cũng như là khả năng logic.

## Tình huống sử dụng

Nó được sử dụng để giải trí mọi lúc, mọi nơi do đặc thù đơn giản là chơi trên máy tính,... Đặc biệt đây là trò chơi được các bạn trẻ rất yêu thích.

## Yêu cầu sử dụng

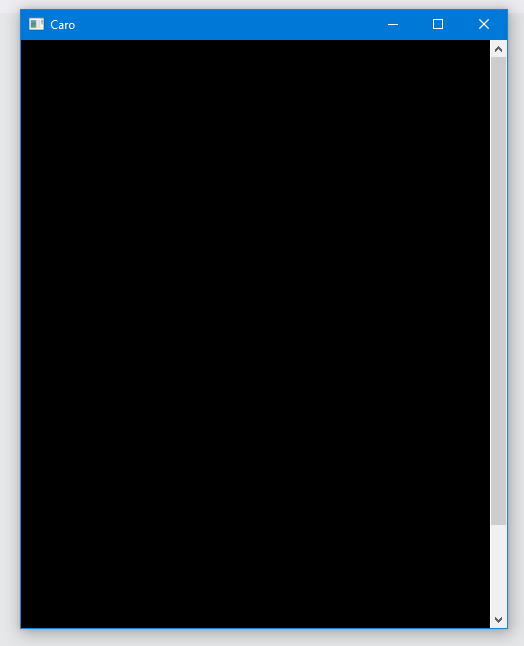
- Hệ điều hành tương thích: Win 7 trở lên

- Phần mềm cài đặt: Visual Studio bản 2010 trở lên

- Ngôn ngữ sử dụng: ngôn ngữ C++

# II/ QUÁ TRÌNH TIẾN HÀNH

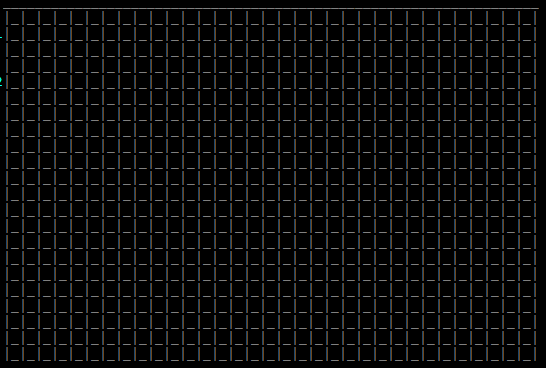
1. ***Thiết kế giao diện***

- Sử dụng giao diện Console, viết bằng ngôn ngữ C++ (phần mềm lập trình Visual Studio)

***Hình 1:***

Giao diện Console

- Thiết kế 1 bàn cờ để chơi cờ Caro



***Hình 2:***

Giao diện bàn cờ

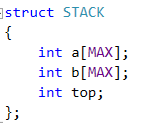
1. ***Mô tả Code***: Nhóm chia code ra 1 phần

***2.1. Tạo stack***

- Cài đặt Stack bằng mảng 2 chiều.

- Trong phần khai báo cấu trúc Stack gồm có:

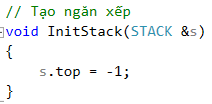
* Khai báo mảng a[1000] chứa hoành độ của điểm kiểu int.
* Khai báo mảng b[1000] chứa tung độ của điểm kiểu số int.
* Khai báo top kiểu int dùng để chỉ số phần tử của đỉnh đầu ngăn xếp.



***Hình 3:***

Khai báo cấu trúc Stack

- Hàm khởi tạo ngăn xếp **InitSack(STACK &s)**: Giá trị ban đầu của top = – 1

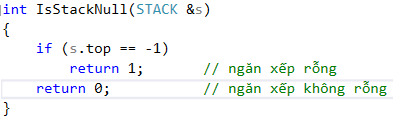


***Hình 4:***

Khởi tạo ngăn xếp

- Hàm kiểm tra ngăn xếp rỗng **IsStackNull(STACK &s)**:

* Nếu top= – 1 có nghĩa là Stack rỗng.
* Còn khác – 1 có nghĩa là Stack không rỗng.

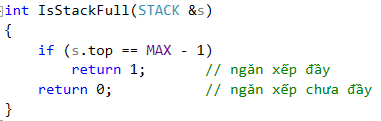


***Hình 5:***

Kiểm tra ngăn xếp rỗng

- Hàm kiểm tra ngăn xếp đầy hay không **IsStackFull(STACK &s)**:

* Nếu giá trị top= MAX -1= 1000 -1= 999 thì có nghĩa ngăn xếp đầy.
* Ngược lại, có nghĩa là ngăn xếp chưa đầy.

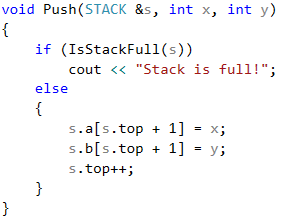


***Hình 6:***

Kiểm tra ngăn xếp đầy

- Hàm thêm một phần tử vào ngăn xếp **Push(STACK &s, int x, int y)**

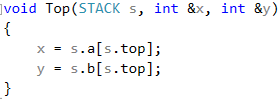
* Biến x, biến y là biến được truyền vào khi gọi hàm Push().
* Biến x,y sẽ được lưu vào vị trí top trong Stack.
* Top++ sẽ tạo ra một vị trí mới.



***Hình 7:***

Thêm một phần tử vào ngăn xếp

- Lấy thông tin ở đỉnh phần tử của ngăn xếp **Top(STACK &s, int &x, int &y)**

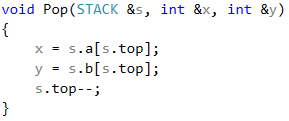
****

***Hình 8:***

Lấy thông tin ở đỉnh phần tử ngăn xếp

- Hàm trích hủy phần tử ở đỉnh ngăn xếp **Pop(STACK &s, int &x, int &y)**

* Nếu stack không rỗng thì trả về giá trị ở top.
* Phần tử đó bị xóa khỏi Stack (top--).

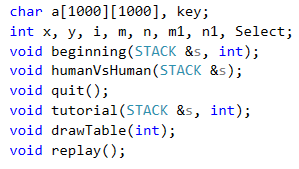


***Hình 9:***

Trích huy phần tử ở đỉnh ngăn xếp

***2.2. Tạo các hàm xử lý***

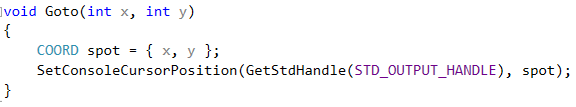
- Khai báo các hàm



***Hình 10:***

Khai báo các hàm

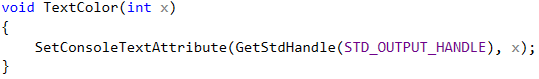
- Hàm vẽ tọa độ **Goto** **(int x, int y)**



***Hình 11:***

Vẽ tọa độ

- Tô màu chữ **TextColor(int x)**

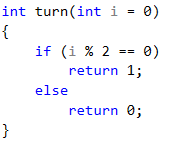
****

***Hình 12:***

Tô màu chữ

- Hàm tính lượt đi **turn (int i=0)**

* Nếu phần dư của i chia cho 2= 0 thì trả về giá trị 1.
* Còn khác 0 thì trả về giá trị 0.

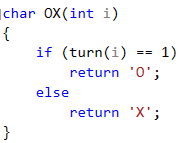
****

***Hình 13:***

Tính lượt đi

- Hàm thứ tự lượt đi **OX(int i)**

* Dựa vào i để biết đến lượt thứ mấy từ đó xuất ra O hoặc X.
* Nếu i chẵn tức lượt đi là số chẵn là O, số lẻ là X.

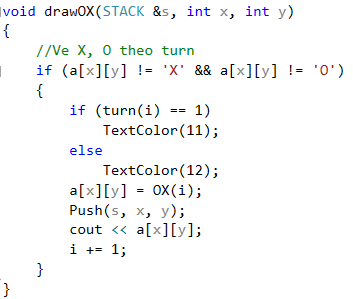


***Hình 14:***

Thứ tự lượt đi OX

- Hàm kiểm tra xem ô đó đã được đánh chưa **drawOX(STACK &s, int x, int y)**

* Nếu chưa thì kiểm tra đến lượt X hay O đi, từ đó in ra C hoặc O.
* Đồng thời lưu vị trí vừa đánh vào Stack để lưu, nếu có Undo thì dùng tới.



***Hình 15:***

Kiểm tra đánh dấu OX

- Các hàm left, right, up, down để chặn các biên của bàn cờ

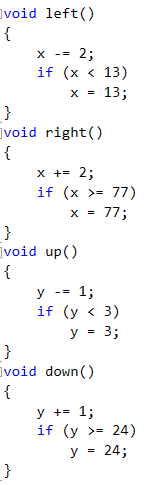
**left()**

**right()**

**up()**

**down()**

* Các hàm giúp không để con trỏ ra ngoài bàn cờ
* Đảm bảo đánh X và O luôn trong bàn cờ

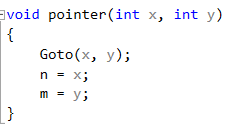


***Hình 16:***

Các hàm dùng để chặn biên

- Hàm **pointer(int x, int y)**

* Khi được dùng trong hàm **humanVsHuman** dùng để đưa con trở về đầu bàn cờ
* Nếu không như vậy thù sẽ không có cách đưa con trỏ vào trong bàn cờ



***Hình 17:***

Hàm pointer

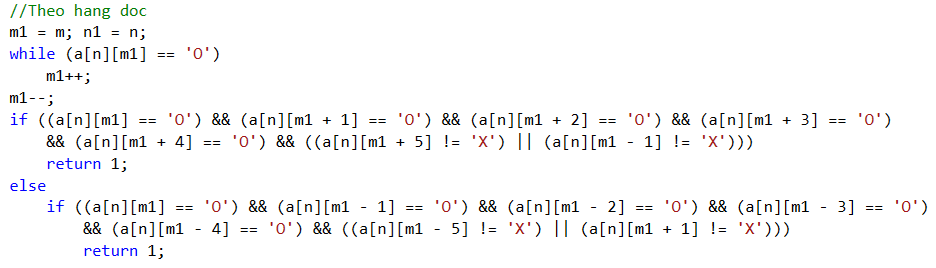
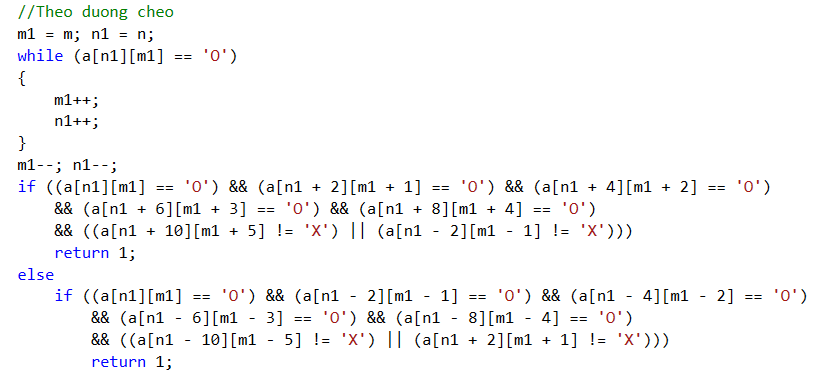
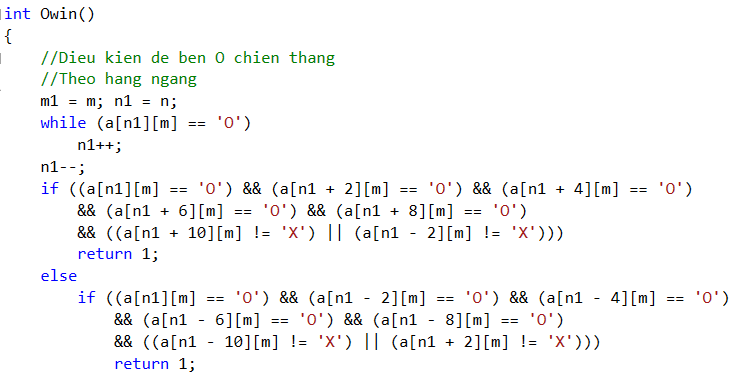
- Hàm **GameInfo()**: đưa thông tin (Có bao gồm hướng dẫn chơi) về Game trong khi trận đấu đang diễn ra

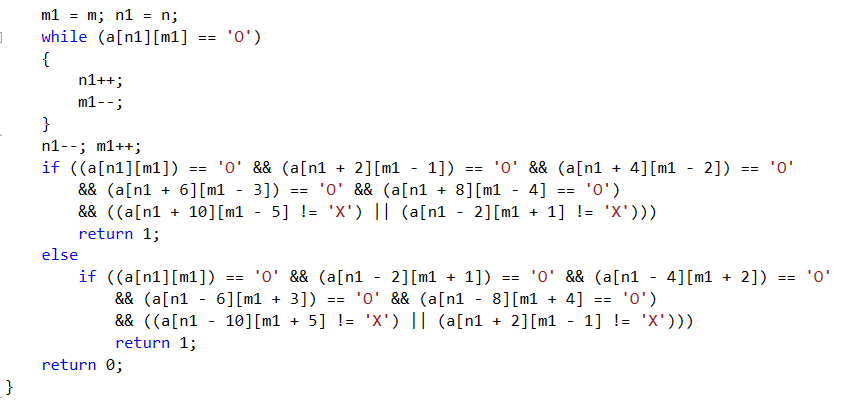


**Hình 18:**

Hàm gameInfo

- Hàm **Owin()**: đưa ra các điều kiện để xét người chơi O thắng

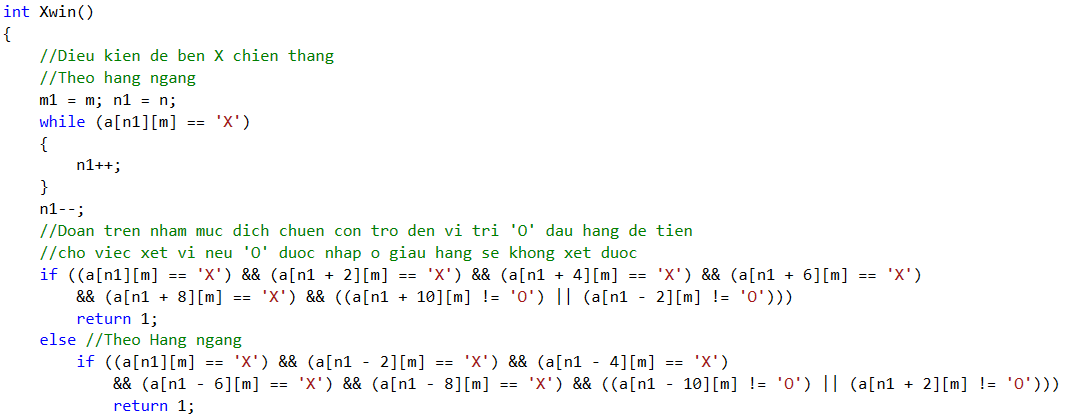
****

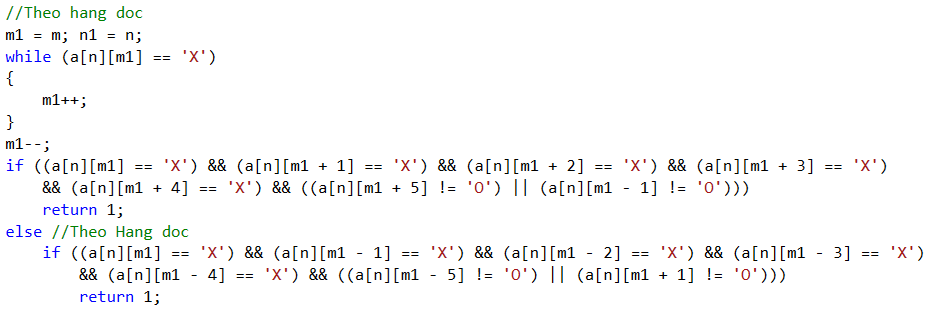
****

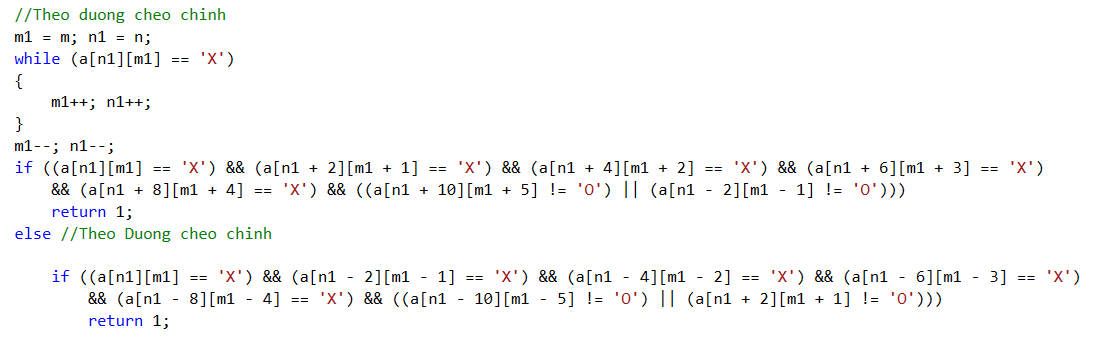
***Hình 19:***

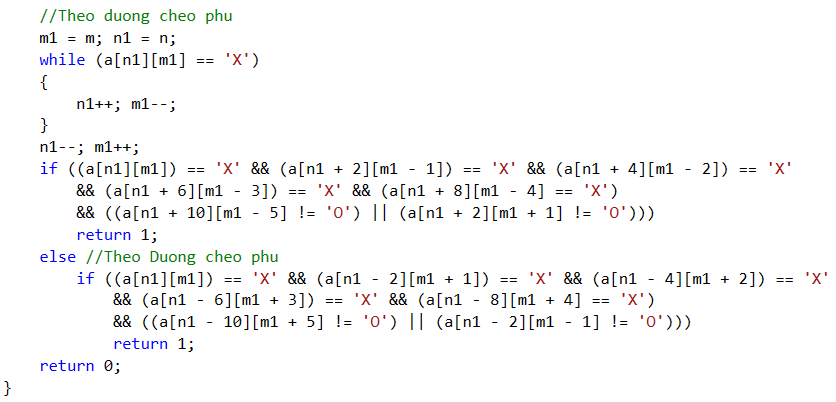
Hàm Owin

- Hàm **Xwin()**: hàm đưa ra điều kiện để xét người chơi X thắng





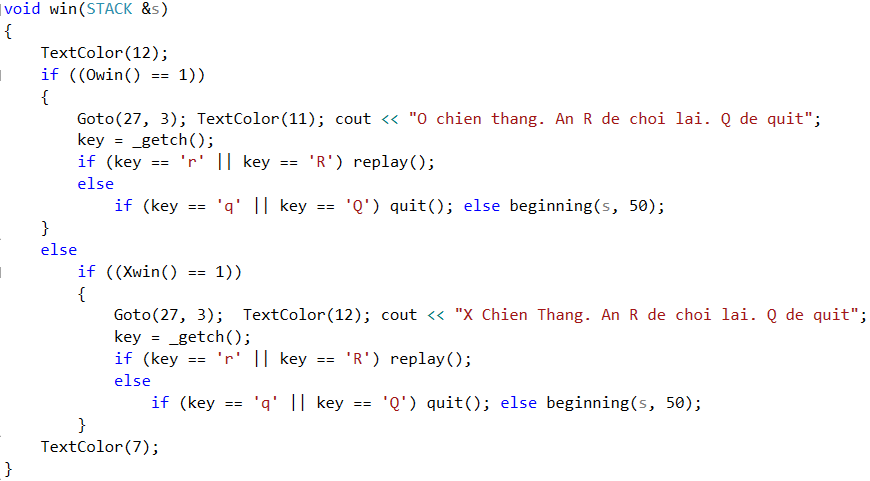




***Hình 20:***

Hàm Xwin

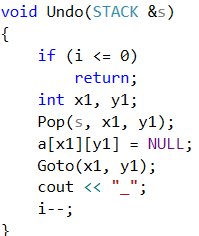
- Hàm **win(STACK &s)**: dùng khi có người chiến thắng thì sẽ hiện thông tin người thắng và đưa ra lựa chọn cho người chơi sau khi trận đấu kết thúc.



***Hình 21:***

Hàm win

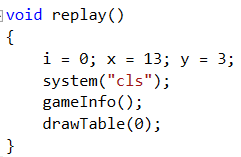
- Hàm **Undo(STACK &s)**: có thể sử dụng liên tục cho đến khi không còn quân cờ X hay O trên bàn cờ.



***Hình 22:***

Hàm Undo

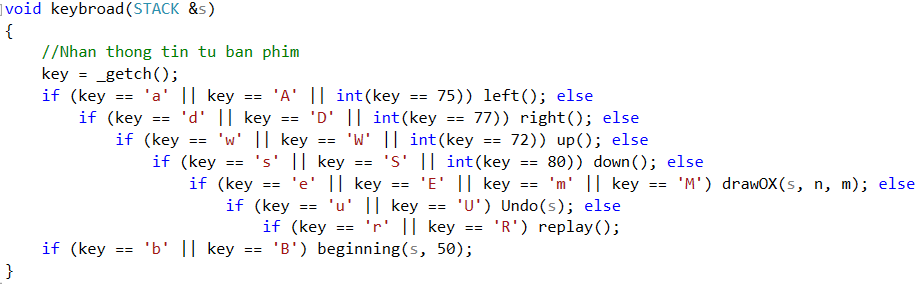
- Hàm **replay()** là để cấp phép cho người chơi chơi lại 1 trận mới khi 1 trận vừa kết thúc hoặc 2 người chơi đồng ý chơi 1 trận mới thay vì tiếp tục trận đấu đó.



***Hình 23:***

Hàm Replay

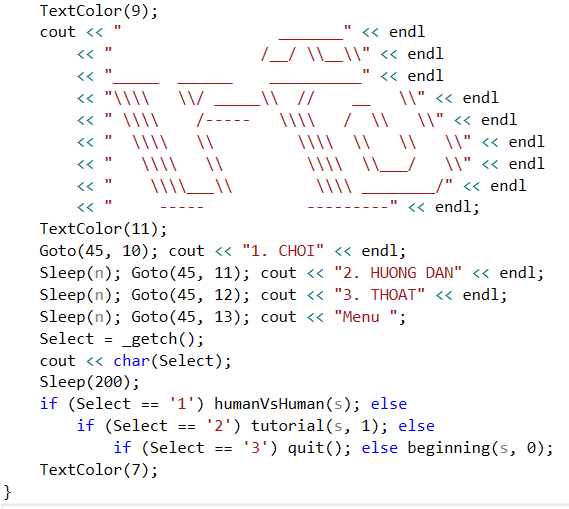
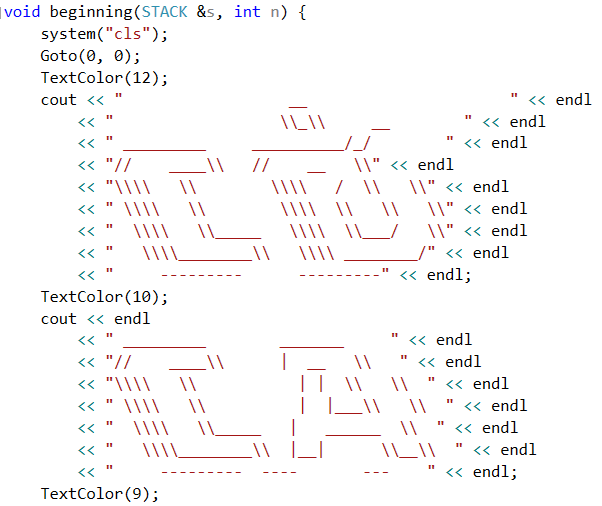
- Hàm **keyboard(STACK &s)**: là để nhận thông tin nhập từ bàn phím, từ đó xử lý cho phù hợp.



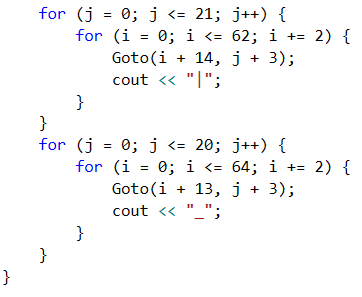
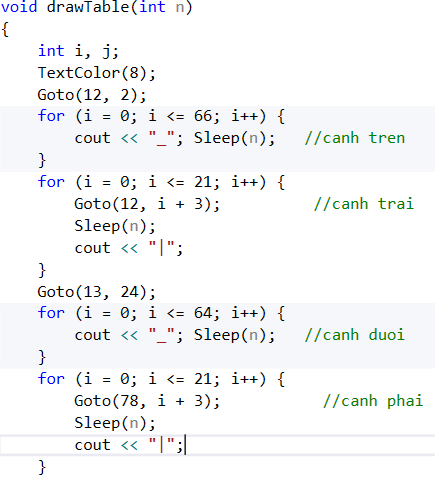
***Hình 24:***

Hàm keyboard

- Hàm **beginning(STACK &s, int n)** là khởi tạo trò chơi.



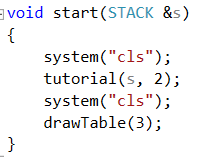
- Hàm **drawTable(int n)** dùng để vẽ bàn cờ.



***Hình 26:***

Hàm drawTable

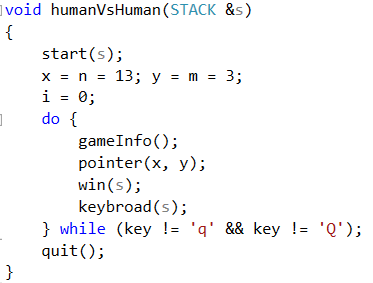
- Hàm **start(STACKS &s)** dùng để bắt đầu trò chơi.



***Hình 27:***

Hàm start

- Hàm **humanvsHuman(STACK &s)** đưa ra các công việc cần làm khi ở chế độ PVP.



***Hình 28:***

Hàm humanVsHuman

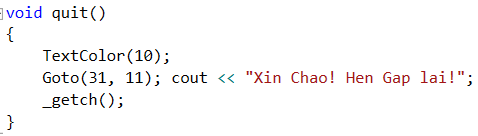
- Hàm **tutorial(STACK &s, int k)** là hướng dẫn người chơi.



***Hình 29:***

Hàm tutorial

- Hàm **quit()** dùng để thoát game.



***Hình 30:***

Hàm thoát

# III/ BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung công việc** | **Nguyễn Ngọc Phương Mai** | **Ngô Đình Hải** | **Ghi chú** |
| 1 | Phụ trách viết chương trình |  | ✓ |  |
| 2 | Phụ trách làm báo cáo đồ án | ✓ |  |  |
| 3 | Phác họa ý tưởng | ✓ | ✓ | 5/5 |
| 4 | Mô tả thuật toán | ✓ | ✓ | 4/6 |

**IV/ KẾT LUẬN**

- Đánh giá mức độ hoàn thành mục tiêu: 75% và chương trình có khả năng dự đoán tối đa được 21 nước đi.

- Khó khăn gặp phải: Trong quá trình thực hiện còn có nhiều lỗi phát sinh về thuật toán cũng như code, đặc biệt là chức năng Undo.

- Khắc phục: Nghiên cứu các bài hướng dẫn trên Youtube, Google, có học nhóm để cùng nhau thống nhất đưa ra phương hướng khắc phục hiệu quả.

- Ý tưởng phát triển đồ án: Dựa vào trò chơi cờ Caro truyền thống và có thêm chức năng Undo để hỗ trợ việc quay trở lại nước đi trước.

## Khuyết điểm

- Chưa thực hiện được tất cả nước đi.s

- Giao diện chưa được bắt mắt.

1. ***Ưu điểm***

- Chơi game khá ổn.

- Có chức năng Undo giúp người chơi trở lại nước đi trước tiện lợi, dễ dàng và đây cũng là một điểm khác biệt so với những game caro thông thường.