

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA KHOA HỌC MÁY TÍNH



BÁO CÁO ĐỒ ÁN  
MÔN TRÍ TUỆ NHÂN TẠO

TRÒ CHƠI CỜ TƯỚNG  
(ĐÃ CHỈNH SỬA THEO NHẬN XÉT GIẢNG VIÊN)



LỚP: KHTN2015  
GVLT: ĐỖ VĂN NHƠN  
GVTH: HUỲNH THỊ THANH THƯƠNG

NHÓM THỰC HIỆN

1. Đinh Duy Phương 15520659
2. Nguyễn Văn Minh 15520488
3. Huỳnh Ngọc Thiên Trang 15520917
4. Phạm Duy Thành 15520804

# MỤC LỤC

CHƯƠNG	NỘI DUNG	TRANG
	MỞ ĐẦU	3
	HỌP ĐỒNG NHÓM	4
	KẾ HOẠCH LÀM VIỆC CỦA NHÓM FANTASTIC	9
	ĐÁNH GIÁ NỘI BỘ NHÓM FANTASTIC	12
Chương 1	GIỚI THIỆU CỜ TƯỚNG	14
	I. NGUỒN GỐC TRÒ CHƠI	14
	II. MÔ TẢ TRÒ CHƠI	15
	III. ỨNG DỤNG – LỢI ÍCH	24
Chương 2	CƠ SỞ LÝ THUYẾT	27
	I. NGÔN NGỮ LẬP TRÌNH C#	27
	II. LÝ THUYẾT ÁP DỤNG	28
	III. KẾT LUẬN	37
Chương 3	PHÂN TÍCH VÀ THIẾT KẾ	38
	I. PHÂN TÍCH BÀI TOÁN	38
	II. THIẾT KẾ CÁU TRÚC DỮ LIỆU VÀ TRẠNG THÁI	40
	III. CÁC VẤN ĐỀ VÀ GIẢI THUẬT	64
Chương 4	ỨNG DỤNG	92
	I. MÔ TẢ CHƯƠNG TRÌNH	92
	II. CÁI ĐẶT VÀ MỘT SỐ HÀM XỬ LÝ	102
Chương 5	KẾT LUẬN	112
	I. KẾT QUẢ ĐẠT ĐƯỢC	112
	II. HẠN CHÉ	112
	III. HƯỚNG PHÁT TRIỂN	113
	TÀI LIỆU THAM KHẢO	114

## MỞ ĐẦU

Để có thể áp dụng môn học trí tuệ nhân tạo vào thực tiễn, đồng thời có thể hoàn thành môn học này, nhóm FANTASTIC thực hiện đồ án “Trò chơi Cờ Tướng”, áp dụng hết tất cả các kiến thức được học trên lớp kết hợp với sự hướng dẫn của giảng viên để có thể hoàn thành đồ án một cách tốt nhất.

Trò chơi Cờ Tướng (cờ Trung Hoa), là loại cờ phổ biến nhất thế giới cùng với cờ vua, là một trò chơi trí tuệ dành cho hai người. Trong ván cờ, mục đích của mỗi người là tìm ra cách đi cho các quân cờ theo đúng luật để có thể chiêu bí hoặc bắt được Tướng của đối phương. Trong đồ án này, ván cờ được lập trình gồm hai đối tượng là người chơi và máy tính, để máy tính có được những nước cờ tốt cần có những kĩ thuật và giải thuật tốt cho máy. Đối với học kì này, đồ án được thực hiện theo yêu cầu của giảng viên là áp dụng “chiến lược Minimax và cắt tỉa alpha – beta” để hoàn thành và không yêu cầu nước đi thông minh nhất!

*Thành phố Hồ Chí Minh, ngày 30 tháng 07 năm 2017*

*Nhóm FANTASTIC*



# HỢP ĐỒNG NHÓM

## I. THÔNG TIN NHÓM:

- Tên nhóm: FANTASTIC TEAM
- Đồ án: Cờ tướng.

STT	Họ và tên	MSSV	Chức vụ	Chữ ký
1	Đinh Duy Phương	15520659	Trưởng nhóm	
2	Nguyễn Văn Minh	15520488	Thành viên	
3	Huỳnh Ngọc Thiên Trang	15520917	Thành viên	
4	Phạm Duy Thanh	15520804	Thành viên	

## II. MỤC TIÊU THÀNH LẬP NHÓM:

- Xây dựng mối quan hệ đoàn kết, giúp đỡ lẫn nhau, sự kết hợp giữa các thành viên trong nhóm để hoàn thành các công việc đề ra.
- Có kiến thức về các kỹ năng phục vụ cho việc học và việc làm trong tương lai.
- Chủ động thực hiện quá trình học và tự học với phẩm chất đạo đức tốt, có kiến thức và kỹ năng chuyên môn đáp ứng yêu cầu làm việc của xã hội.

## III. NỘI QUY NHÓM:

Những điều một thành viên thuộc về nhóm **PHẢI** làm theo

- Luôn đúng giờ, không vắng mặt.
- Có trách nhiệm với nhóm, hoàn thành tốt công việc được giao.
- Tránh xảy ra mâu thuẫn lớn trong quá trình làm việc.

- Tôn trọng ý kiến và duy trì tình cảm với các thành viên trong nhóm.

Những điều một thành viên thuộc về nhóm **NÊN** làm theo (không bắt buộc)

- Cố gắng đoàn kết nhóm để đạt đến mục tiêu chung.
- Có ý thức tự giác, có trách nhiệm đối với công việc của nhóm.
- Vui vẻ, hòa đồng, không tự kỷ.
- Biết quan tâm, lắng nghe và chia sẻ.

#### IV. TIÊU CHÍ ĐÁNH GIÁ:

Tiêu chí	Tuyệt vời	Tốt	Tạm được	Kém
<b>Tinh thần trách nhiệm</b>	Hoàn thành tốt công việc được phân công trong thời gian đề ra, giúp đỡ các thành viên khác trong nhóm khi cần thiết.	Hoàn thành tốt công việc được phân công trong thời gian đề ra.	Hoàn thành công việc.	Không hoàn thành công việc.

<b>Tinh thần hợp tác</b>	Tôn trọng ý kiến của các thành viên trong nhóm. Tích cực đóng góp ý kiến về thuật toán, giao diện... cho đồ án.	Đạt kết quả tốt trong công việc mình được giao, không có đóng góp gì thêm.	Tinh thần hợp tác chưa tốt, còn hoạt động riêng lẻ.	Không tôn trọng ý kiến người khác, gây chia rẽ nội bộ nhóm.
<b>Quản lý xung đột</b>	Hợp tác với nhau tìm ra giải pháp tốt nhất cho đồ án, giúp nâng cao chất lượng đồ án.	Cùng nhau tìm những giải pháp trung hòa để có thể hoàn thành đồ án.	Góp ý tương đối khi có xung đột về thuật giải, ý tưởng.	Né tránh sự va chạm, sờ đổi đầu với mâu thuẫn, không quan tâm đến nhu cầu, không làm cưng không sao.
<b>Họp nhóm</b>	Đi họp 1 lần/tuần, có tham gia đóng góp ý kiến trong buổi họp	Đi họp 1 lần/tuần, không tham gia đóng góp ý kiến	Đi họp trễ.	Vắng mặt trong buổi họp.

<b>Giải quyết vấn đề đồ án</b>	Tích cực cùng thành viên tìm ra vấn đề và nhanh chóng giải quyết.	Đóng góp tương đối.	Thỉnh thoảng đóng góp.	Không quan tâm.
--------------------------------	---	---------------------	------------------------	-----------------

## V. CÁCH TÍNH ĐIỂM:

Các thành viên sẽ chấm điểm lẫn nhau: thành viên này sẽ chấm điểm cho 3 thành viên còn lại.

Quy ước:

- Kém: 5 điểm.
- Tạm được: 6 đến 7.5 điểm.
- Tốt: 8 đến 9.
- Tuyệt vời: 9 trở lên.

## VI. PHÂN CÔNG THỰC HIỆN

### 1) Quy định chung

- Mỗi cá nhân trong nhóm thực hiện công việc cụ thể theo sự phân công của Trưởng nhóm.
- Các thành viên giúp đỡ lẫn nhau để hoàn thành đồ án tốt nhất.

### 2) Phân công nhiệm vụ cụ thể

TT	Họ và tên	Chức vụ	Công việc cụ thể được phân công
1	Đinh Duy Phương	Trưởng nhóm	<ul style="list-style-type: none"> <li>- Thông báo tiến độ công việc và lên lịch thực hiện công việc trên group nhóm.</li> <li>- Liên lạc với các thành viên trong nhóm, thúc đẩy tiến độ công việc.</li> <li>- Tìm hiểu và thực hiện giải thuật minimax cắt tỉa alpha.</li> <li>- Thực hiện hàm heuristic</li> </ul>

			<ul style="list-style-type: none"> <li>- Nhạc nền cho game.</li> <li>- Tổng hợp code.</li> </ul>
2	Nguyễn Văn Minh	Thành viên	<ul style="list-style-type: none"> <li>- Cùng nhóm trưởng thực hiện thuật toán.</li> <li>- Tìm hiểu và thực hiện giao diện.</li> <li>- Sinh nước đi cho cờ.</li> <li>- Kiểm tra điều kiện kết thúc</li> <li>- Nhạc nền cho trò chơi</li> </ul>
3	Huỳnh Ngọc Thiên Trang	Thành viên	<ul style="list-style-type: none"> <li>- Cùng nhóm trưởng thực hiện thuật toán.</li> <li>- Tìm hiểu và thực hiện giao diện.</li> <li>- Sinh nước đi cho cờ.</li> <li>- Kiểm tra điều kiện kết thúc</li> </ul>
4	Phạm Duy Thành	Thành viên	<ul style="list-style-type: none"> <li>- Tìm hiểu, tham khảo tài liệu về cờ tướng.</li> <li>- Viết báo cáo đồ án.</li> <li>- Thiết kế giao diện.</li> <li>- Hỗ trợ các thành viên (nếu cần thiết).</li> </ul>

## VII. CAM KẾT CÁC THÀNH VIÊN:

Sau khi đọc kỹ nội dung của hợp đồng nhóm, các thành viên trong nhóm nhất trí và cam kết hoàn thành tốt những mục tiêu đã đề ra.

## VIII. KÝ TÊN

# KẾ HOẠCH LÀM VIỆC CỦA NHÓM FANTASTIC

## I. MỤC TIÊU

- Tạo giao diện cho trò chơi.
- Tạo được nước đi cho quân cờ.
- Lập trình được nước đi tốt tiếp theo của quân cờ áp dụng “Chiến lược minimax – cắt tỉa alpha-beta”.

## II. NỘI DUNG VÀ CÁC CÔNG VIỆC YÊU CẦU THỰC HIỆN

TT	Nội dung	Yêu cầu
1	Viết kế hoạch làm việc cụ thể	<ul style="list-style-type: none"> <li>- Các thành viên cho ý kiến về kế hoạch và thống nhất kế hoạch nhóm thực hiện để hoàn thành đồ án môn học tốt nhất.</li> </ul>
2	Giao diện trò chơi	<ul style="list-style-type: none"> <li>- Giao diện đẹp mắt, gây thích thú cho người chơi.</li> <li>- Đề cao tính phong phú trong các chức năng hỗ trợ người chơi.</li> <li>- Tiện ích.</li> </ul>
3	Nhạc nền trò chơi	<ul style="list-style-type: none"> <li>- Mang phong cách Trung Hoa.</li> <li>- Nhẹ nhàng, giúp người chơi tập trung.</li> </ul>
4	Nước đi cho từng quân cờ	<ul style="list-style-type: none"> <li>- Đúng luật.</li> <li>- Hiện vị trí quân cờ có thể di chuyển đến trên bàn cờ.</li> </ul>

5	Minimax – cắt tỉa alpha-beta	<ul style="list-style-type: none"> <li>- Đảm bảo đúng với kiến thức đã học trên lớp.</li> <li>- Giúp nước đi tiếp theo của máy tính là nước đi tốt trên trạng thái hiện tại của ván cờ.</li> </ul>
6	Báo cáo đồ án	<ul style="list-style-type: none"> <li>- Hình thức: đảm bảo đúng về font, size theo yêu cầu của giảng viên.</li> <li>- Nội dung: đảm bảo đầy đủ các phần như phân tích bài toán, cấu trúc dữ liệu, nêu các vấn đề gặp phải trong quá trình thực hiện, trình bày thuật giải, demo...</li> <li>- Mạch lạc, rõ ràng.</li> </ul>

### III. PHÂN CÔNG CÔNG VIỆC VÀ THỜI GIAN HOÀN THÀNH

#### 1) Quy định chung

- Mỗi cá nhân trong nhóm phải hoàn thành công việc theo sự phân công đã thống nhất.
- Các thành viên giúp đỡ lẫn nhau trong quá trình thực hiện đồ án.

#### 2) Thời gian hoàn thành

STT	Nội Dung	Thời gian hoàn thành	Người thực hiện
1	Tham khảo và hiện thực giao diện	Từ 10/06 đến 13/06	Trang, Minh
2	Thiết kế hình ảnh giao diện: background, flash screen, các nút cần thiết, icon ...	Từ 16/06 đến 18/06	Duy Thanh
3	Tạo nước đi cho cờ	26/06/2017	Minh, Trang
4	Hàm heuristic – minimax – cắt tỉa alpha	28/06/2017	Duy Phương
5	Báo cáo đồ án	29/06/2017	Duy Thanh
6	Tổng kết	Sáng 30/06/2017	Cả Nhóm

**Nơi nhận:**  
Kính gửi ThS Huỳnh Thị  
Thanh Thương – giảng viên  
hướng dẫn môn Trí Tuệ Nhân  
Tạo (CS106.H21.KHTN – VN)

Tp. Hồ Chí Minh, ngày 10 tháng 06 năm 2017  
**Trưởng nhóm**

*Dinh Duy Phuong*

# ĐÁNH GIÁ NỘI BỘ NHÓM FANTASTIC

## I. YÊU CẦU

- Các thành viên trong nhóm đánh giá lẫn nhau qua những ngày cùng nhau thực hiện đồ án.
- Các thành viên dựa trên những tiêu chí trong họp đồng nhóm để đánh giá trên thang điểm từ 5 đến 10.
- Đề cao sự trung thực, đây sẽ là yếu tố giúp các thành viên hoàn thiện hơn học tập lẫn trách nhiệm bản thân.

## II. KẾT QUẢ ĐÁNH GIÁ

Tiêu chí Thành viên	Tinh thần trách nhiệm	Tinh thần hợp tác	Quản lý xung đột	Hợp nhóm	Giải quyết vấn đề đồ án	Trung bình
Đinh Duy Phương	100%	100%	100%	100%	90%	98%
Nguyễn Văn Minh	100%	100%	100%	100%	90%	98%
Huỳnh Ngọc Thiên Trang	100%	100%	100%	100%	90%	98%
Phạm Duy Thanh	100%	100%	100%	100%	80%	96%

### III. CAM KẾT VÀ KÝ TÊN

Các thành viên thống nhất với bảng đánh giá trên.

*Thành phố Hồ Chí Minh, ngày 30 tháng 06 năm 2017*

**KÝ TÊN**

# Chương 1

## GIỚI THIỆU CỜ TƯỚNG

### I. NGUỒN GỐC TRÒ CHƠI

#### 1) Nguồn gốc

Cờ tướng là loại cờ có từ khoảng thế kỷ 7. Cờ này được bắt nguồn từ Saturanga, một loại cờ cổ được phát minh ở Ấn Độ từ thế kỷ 5 đến thế kỷ 6 (trước cờ tướng khoảng 200 năm). Chính Saturanga được phát minh từ Ấn Độ, sau đó đi về phía tây, trở thành cờ vua và đi về phía Đông trở thành cờ tướng. Người Trung Quốc cũng đã thừa nhận điều này.

#### 2) Cải tiến

Cờ tướng cổ đại không có quân Pháo. Các nhà nghiên cứu đều thống nhất là quân Pháo được bổ sung từ thời nhà Đường (sau năm 618), là quân cờ ra đời muộn nhất trong bàn cờ tướng, bởi cho tới thời đó, con người mới tìm ra vũ khí pháo để sử dụng trong chiến tranh.

Tuy nhiên, người Trung Hoa đã cải tiến bàn cờ Saturanga như sau:

- Họ không dùng "ô", không dùng hai màu để phân biệt ô, mà họ chuyển sang dùng "đường" để đặt quân và đi quân. Chỉ với động tác này, họ đã tăng thêm số điểm đi quân từ 64 của Saturanga lên 81.
- Đã là hai quốc gia đối kháng thì phải có biên giới rõ ràng, từ đó, họ đặt ra "hà", tức là sông. Khi "hà" xuất hiện trên bàn cờ, 18 điểm đặt quân nữa được tăng thêm. Như vậy, bàn cờ tướng bây giờ đã là 90 điểm so với 64, đó là một sự mở rộng đáng kể. Tuy nhiên, diện tích chung của bàn cờ hầu như không tăng mấy (chỉ tăng thêm 8 ô) so với số điểm tăng lên tới một phần 3.
- Đã là quốc gia thì phải có cung cấm (宮) và không thể đi khắp bàn cờ như kiểu trò chơi Saturanga được. Thé là "Cửu cung" đã được tạo ra. Điều này thể hiện tư duy phuong Đông hết sức rõ ràng.
- Bàn cờ Saturanga có hình dáng quân cờ là những hình khối, nhưng cờ Tướng thì quân nào trông cũng giống quân nào, chỉ có mỗi tên là khác nhau, lại được

viết bằng chữ Hán. Đây có thể là lý do khiến cờ tướng không được phổ biến bằng cờ vua, chỉ cần liếc qua là có thể nhận ra đâu là Vua, đâu là Hoàng hậu, kỵ sĩ, v.v. Tuy nhiên, đối với người Trung Hoa thì việc thuộc mực cờ này là không có vấn đề gì khó khăn. Có lẽ việc cải tiến này cũng một phần là do điều kiện kinh tế bấy giờ chưa sản xuất được bộ cờ có hình khối phức tạp như cờ vua. Cờ tướng không phải là một trò chơi sang trọng, muốn tạo ra một bàn cờ tướng cực kỳ đơn giản, chỉ cần lấy que vạch xuống nền đất cũng xong, còn cờ vua thì mất công hơn nhiều khi phải tạo ra các ô đen/trắng xen kẽ nhau. Gần đây ngày càng có nhiều ý kiến đề nghị cải cách hình dáng các quân cờ tướng và trên thực tế người ta đã đưa những phác thảo của những bộ quân mới bằng hình tượng thay cho chữ viết, nhất là khi cờ tướng được chơi ở những nước không sử dụng tiếng Trung Quốc.

- Với sự thay đổi bố cục bàn cờ, người Trung Hoa đã phải có những điều chỉnh để lấy lại sự cân bằng cho bàn cờ. Đó chính là những ngoại lệ mà người chơi phải tự nhớ.

## II. MÔ TẢ TRÒ CHƠI

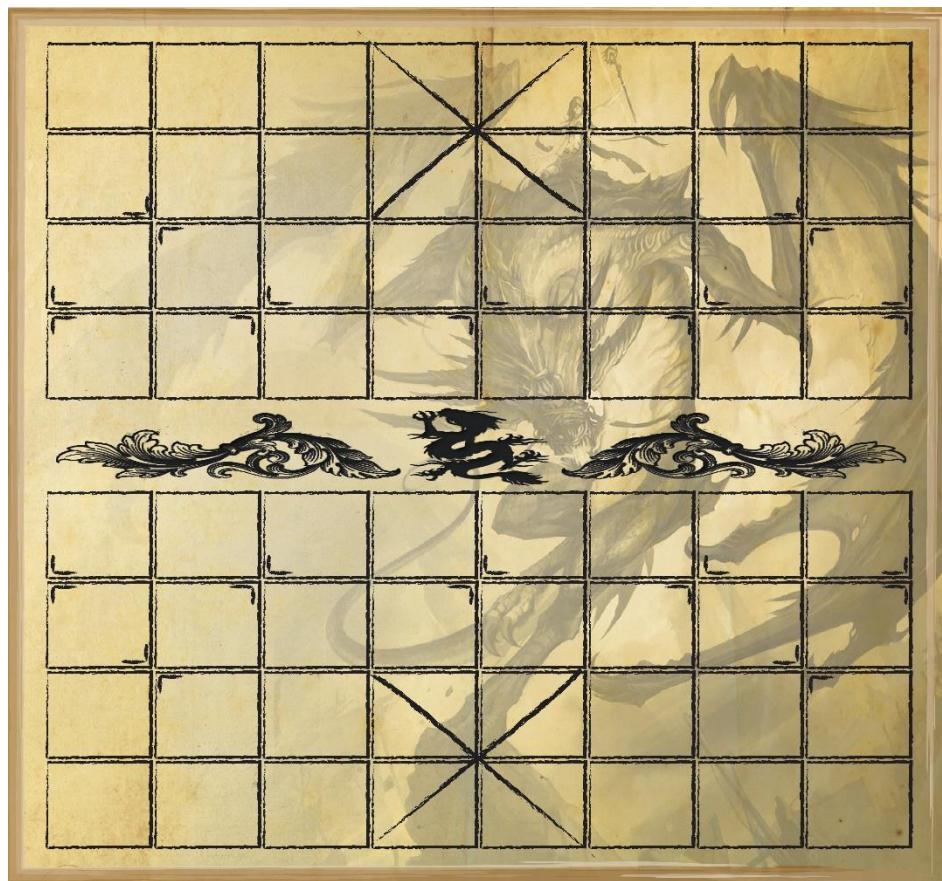
### 1) Mục đích của ván cờ

Ván cờ được tiến hành giữa hai người, một người cầm quân Trắng (hay Đỏ), một người cầm quân Đen (hay Xanh lục). Mục đích của mỗi người là tìm mọi cách di quân trên bàn cờ theo đúng luật để chiếu bí hay bắt Tướng (hay Soái, hoặc Suý) của đối phương và giành thắng lợi.

### 2) Bàn cờ

Bàn cờ là một hình chữ nhật do 9 đường dọc và 10 đường ngang cắt nhau vuông góc tại 90 điểm hợp thành. Một khoảng trống gọi là sông (hay hà) nằm ngang giữa bàn cờ, chia bàn cờ thành hai phần đối xứng bằng nhau. Mỗi bên có một cung Tướng hình vuông (Cửu cung) do 4 ô hợp thành tại các đường dọc 4, 5, 6 kể từ đường ngang cuối của mỗi bên, trong 4 ô này có vẽ hai đường chéo xuyên qua.

Theo quy ước, khi bàn cờ được quan sát chính diện, phía dưới sẽ là quân Trắng (hoặc Đỏ), phía trên sẽ là quân Đen (hoặc Xanh lục). Các đường dọc bên Trắng (Đỏ) được đánh số từ 1 đến 9 từ phải qua trái. Các đường dọc bên Đen (Xanh lục) được đánh số từ 9 tới 1 từ phải qua trái.



(Hình được thiết kế: Bàn cờ tướng)

### 3) Quân cờ và cách đi

Mỗi ván cờ lúc bắt đầu phải có đủ 32 quân, chia đều cho mỗi bên gồm 16 quân Trắng (Đỏ) và 16 quân Đen (Xanh lục), gồm bảy loại quân. Tuy tên quân cờ của mỗi bên có thể viết khác nhau (ký hiệu theo chữ Hán) nhưng giá trị và cách đi quân của chúng lại giống nhau hoàn toàn. Bảy loại quân có ký hiệu và số lượng cho mỗi bên như sau:

Quân	Ký hiệu	Số lượng

Tướng	 	1
Sĩ	 	2
Tượng	 	2
Xe	 	2
Pháo	 	2
Mã	 	2
Tốt	 	5

a) Tướng (Soái)



(Hình thiết kế - Nước đi quân Tướng)

Ở Trung Hoa, vua là thiên tử (con trời), do vậy, nếu nhắc tới vua thì phải tôn kính, sùng bái. Quân cờ quan trọng nhất, vốn tên là vua nhưng các nhà cải cách đã cải tên từ "vua" thành "tướng" hay "soái" cho quân này để phù hợp với việc kiêng húy của chế độ phong kiến. Tuy nhiên, đó chỉ là cách thay đổi tên, thay đổi bì ngoài, hình thức mà thôi, chứ quân cờ này thực chất vẫn là vua, vì quân này chỉ ở trong cung, có hai Tượng và Sĩ kè kề bên cạnh bảo vệ mọi lúc mọi nơi, và mất quân này đồng nghĩa với việc thua ván cờ. Vì vậy, chữ Tướng ở đây chỉ là danh nghĩa.

Tướng chỉ được đi ngang hay đi dọc từng bước một trong phạm vi cung tướng. Tính theo khả năng chiến đấu thì Tướng là quân yếu nhất do chỉ đi nước một và bị giới hạn trong cung. Tuy nhiên trong nhiều tình huống, đặc biệt khi cờ tàn đòn "lộ mặt tướng" (xem ở dưới) lại tỏ ra rất mạnh. Lúc này Tướng mạnh ngang với Xe.

Tướng được chốt chặt trong cung và có tới 2 Sĩ và Tượng canh gác hai bên. Chính điều này làm cho ván cờ trở nên khó phân thắng bại, cơ may hoà cờ là rất lớn. Từ một thực tế như vậy, luật "lộ mặt Tướng" được thiết lập: một bên Tướng đã chiếm được một lộ rồi mà Tướng bên kia có mặt ra lộ ấy mà không có ít nhất một quân khác che chắn ở giữa là bị thua ngay lập tức. Việc này

làm cờ tướng có nhiều biến hóa thắng bại, với Tướng có giá trị như Xe khi chiếm hàn một cột.

b) Sỹ



(Hình thiết kế - Nước đi quân Sỹ)

Trong cờ vua, quân cỗ vẫn được đổi thành quân Hậu. Trong cờ tướng, quân Sĩ có vai trò "hộ giá" cho Tướng. Chúng đứng ngay sát cạnh Tướng, chỉ đi từng bước một và đi theo đường chéo trong Cửu cung. Như vậy, chúng chỉ di chuyển và đứng tại 5 điểm và được coi là quân cờ yếu nhất. Sĩ có chức năng trong việc bảo vệ Tướng, mất Sĩ được cho là nguy hiểm khi đối phương còn đủ 2 Xe hoặc dùng Xe Mã Tốt tấn công. Bỏ Pháo ăn Sĩ rồi dùng 2 Xe tấn công là đòn chiến thuật thường thấy. Khi còn Pháo thì phải chú ý giữ Sĩ để làm ngòi cho Pháo tấn công.

### c) Tượng



(Hình thiết kế - Nước đi quân Sĩ)

Quân Tượng đứng bên cạnh quân Sĩ và tương đương với Tượng trong cờ vua.

Quân này đi theo đường chéo của hình vuông gồm 2 ô cờ. Chúng không được qua sông, chúng có nhiệm vụ ở lại bên này sông để bảo vệ vua. Chỉ có 7 điểm mà Tượng có thể di chuyển tới và đứng ở đó

Tượng sẽ không di chuyển được đến vị trí đã nêu nếu có 1 quân đặt tại vị trí giữa của hình vuông 2 ô. Khi đó ta gọi là Tượng bị cản và vị trí cản được gọi là "mắt Tượng".

Tượng được tính là mạnh hơn Sĩ một chút. Một Tốt qua hà được đổi lấy 1 Sĩ hay Tượng. Tuy nhiên khả năng phòng thủ của Tượng được tính nhỉnh hơn nên nếu Sĩ là 2 thì Tượng là 2,5. Nói chung mắt Tượng cờ dễ nguy khi đối phương có Pháo.

d) Xe



(Hình thiết kế - Nước đi quân Xe)

Quân Xe đi và ăn theo một đường thẳng đứng hoặc ngang giống quân Xe trong cờ vua. Xe được coi là quân cờ mạnh nhất, kiêm cả công lẫn thủ (đòn xe sát vạn). Giá trị của Xe thường tính là bằng 2 pháo hoặc pháo mã.

Khai cuộc thường tranh đưa các quân Xe ra các đường dọc thông thoáng, dễ phòng thủ và tấn công.

e) Pháo



(Hình thiết kế - Nước đi quân Pháo)

Quân Pháo đi giống quân Xe, theo chiều thẳng đứng hoặc ngang, nhưng nếu ăn quân thì phải có 1 quân đứng làm "ngòi" (kể cả của mình hay của đối phương).

Cờ tướng cổ đại không có quân Pháo. Các nhà nghiên cứu đều nhất trí là quân Pháo được bổ sung từ thời nhà Đường. Đây là quân cờ ra đời muộn nhất trên bàn cờ tướng vì tới thời đó, pháo được sử dụng trong chiến tranh với hình thức là một loại máy dùng để bắn những viên đá to. Bấy giờ, từ Pháo trong chữ Hán được viết với bộ "thạch", nghĩa là đá. Cho đến đời nhà Tống, khi loại pháo mới mang thuốc nổ được phát minh thì quân Pháo đã được viết lại với bộ "hỏa".

Do đặc điểm phải có ngòi, Pháo thường dùng Tốt của quân mình trong khai cuộc, hoặc dùng chính Sĩ Tượng của mình làm ngòi để chiếu hết tướng đối phương trong cờ tàn. Quân Pháo có quyền lực mạnh ở lúc khai cuộc, lúc bàn cờ còn nhiều quân, nhưng quyền lực của Pháo giảm dần về sau khi số quân làm ngòi giảm bớt.

Trên thực tế thì có tới 70% khai cuộc là dùng Pháo đưa vào giữa dọa bắt tốt đầu của đối phương, gọi là thế Pháo đầu. Đối phương có thể dùng Pháo đổi lại cũng vào giữa. Kéo Pháo cùng bên gọi là trận Thuận Pháo, kéo Pháo vào ngược bên nhau gọi là trận Nghịch Pháo (hay Liệt Pháo).

#### f) Mã



(Hình thiết kế - Nước đi quân Mã)

Quân Mã trong cờ tướng đi theo hình chữ L 2x1, với bước đi thẳng sau đó đi một bước chéo. Nếu khi đi thẳng có 1 ô ở vị trí đó thì Mã không đi được, gọi là Mã bị cản.

Nước đi và nước cản của Mã

Mã do không đi thẳng, lại có thể bị cản nên mức độ cơ động của quân này kém hơn Xe và Pháo. Khi khai cuộc, Mã kém hơn Pháo do bị cản nhiều. Khi tàn cuộc, Mã trở nên mạnh hơn Pháo do dễ đi hơn và Pháo thì ít ngòi hơn.

### g) Tốt (Binh)



(Hình thiết kế - Nước đi quân Tốt)

Quân Tốt có 5 quân giống nhau, chúng đi thẳng và ăn thẳng theo chiều dọc khi ở bên phần đất của bên mình. Khi Tốt qua được sông, chúng có thể đi và ăn theo chiều ngang. Hai tốt qua sông đứng cạnh nhau gọi là tốt huynh đệ và có sức mạnh cỡ một quân pháo hoặc mã.

Khi đi đến hết bàn cờ, lúc này, chúng được gọi là Tốt lụt.

Trong khai cuộc, việc thí tốt là chuyện tương đối phổ biến. Ngoại trừ việc phải bảo vệ tốt ở giữa (còn gọi là tốt đầu), các quân tốt khác thường xuyên bị xe pháo mã ăn mất. Việc mất mát một vài Tốt ngay từ đầu cũng được xem như việc thí quân trong cờ tướng chứ không là "thảm họa" như trong cờ vua.

Đến cờ tàn, giá trị của quân tốt tăng nhanh và số lượng tốt khi đó có thể đem lại thắng lợi hoặc chỉ hòa cờ. Khi đó việc đưa được tốt qua sông và tới gần cung tướng của đối phương trở nên rất quan trọng. Tốt khi đến hàng sát cuối, ép sát cung tướng được coi là mạnh như Xe.

### **III. ÚNG DỤNG - LỢI ÍCH**

#### **1) Học chơi cờ tướng giúp thông minh hơn**

Các nhà khoa học đã làm cuộc khảo sát, sau 4 tháng học cờ tướng, chỉ số IQ của người chơi tăng lên đáng kể. Chính vì vậy, chơi cờ tướng sẽ giúp thông minh hơn.

#### **2) Chơi cờ tướng giúp phát triển hai bán cầu não**

Khi chơi cờ tướng, ta phải tính toán theo logic và cả phán đoán cảm tính. Vì vậy, để ra những quyết định nhanh chóng và chính xác đòi hỏi phải có sự tham gia của cả 2 bán cầu não.

#### **3) Học chơi cờ tướng làm tăng khả năng sáng tạo**

Một nghiên cứu kéo dài 4 năm với các học sinh từ lớp 7 tới lớp 9, một bên chơi cờ vua dùng máy tính, bên kia tham gia các hoạt động khác một tuần một lần. Kết quả, nhóm chơi cờ vua ghi điểm cao hơn trong tất cả các phép đo lường về khả năng sáng tạo. Điều này có lẽ cũng không có gì ngạc nhiên bởi vì chơi cờ tướng giúp tăng cường hoạt động của não phải.

#### **4) Học chơi cờ tướng giúp tăng kỹ năng giải quyết vấn đề**

Một trận cờ tướng giống như 1 câu đố lớn cần được giải quyết, và việc giải quyết này diễn ra liên tục, bởi đối thủ luôn thay đổi các thông số.

#### **5) Học chơi cờ tướng làm cải thiện khả năng tập trung**

Việc chiến thắng trong một trận đấu giữa các kiện tướng cờ tướng thường là kết quả của sự tập trung cao độ vào bàn cờ. Đó là tố chất một kiện tướng cờ tướng cần phải có. Nhìn đi chỗ khác hoặc suy nghĩ về điều gì đó khác ở 1 thời điểm cũng có thể dẫn đến mất mát cả trận đấu. Nhiều nghiên cứu của các sinh viên ở Mỹ, Nga và Trung Quốc chỉ ra rằng khả năng của con người về sự tập trung được phát triển qua việc chơi cờ tướng (hoặc cờ vua).

#### **6) Chơi cờ tướng làm phát triển các tế bào thần kinh**

Việc chơi cờ tướng làm tăng các tín hiệu liên kết giữa các nơron thần kinh trên não bộ, qua đó giúp các tế bào thần kinh phát triển. Việc học một kỹ năng mới như chơi cờ tướng, hay việc tương tác với trong những hoạt động thử thách giúp cho tế bào thần kinh được nạp đầy nhiên liệu.

#### **7) Học chơi cờ tướng giúp ta lập kế hoạch và phán đoán tương lai**

Vỏ não trước trán là khu vực não chịu trách nhiệm lập kế hoạch, đánh giá và tự kiểm soát. Do đó, về mặt khoa học thì các thanh thiếu niên được gọi là chưa trưởng thành cho tới khi vùng não này phát triển. Những game chiến thuật như cờ tướng có thể thúc đẩy sự phát triển của vỏ não trước trán và giúp chúng ta đưa ra các quyết định tốt hơn trong mọi khía cạnh của cuộc sống.

#### **8) Học chơi cờ tướng giúp phòng ngừa hội chứng đãng trí**

Đây là căn bệnh rất nguy hiểm, các phương pháp điều trị hiện tại chỉ giúp giảm một phần nhỏ triệu chứng bệnh. Chưa có phương pháp trị liệu nào có thể ngăn chặn hoặc làm chậm tiến triển của bệnh. Một nghiên cứu gần đây của tờ báo The New England chỉ ra rằng những người trên 75 tuổi tham gia vào các hoạt động trí óc kéo dài như chơi cờ vua sẽ có ít khả năng mắc bệnh này hơn là những người không chơi.

#### **9) Việc chơi cờ tướng trong thời gian dài cũng giúp mọi người rèn luyện được những nhân phẩm tốt**

Cờ tướng tạo cho người ta tư cách của một nhà thể thao, một tính cách cao thượng, một con người chân chính: biết thắng một cách trung thực và biết chấp nhận thua một cách đường hoàng. Không thèm "ăn gian" hay dùng các mánh lới xấu.

Cờ tướng là sự giao lưu, tiếp xúc với nhiều người. Từ tạo cho bạn cách cư xử lịch sự nhã nhặn. Cũng từ đó bạn sẽ kết thân được với nhiều bạn bè ở mọi lứa tuổi, mọi nơi. Bạn bè luôn là một phần quan trọng của cuộc sống chúng ta. Nếu được tham dự các cuộc thi đấu trong nước và quốc tế bạn sẽ được đi tới nhiều miền đất lạ, thăm được nhiều di tích thắng cảnh, mở mang tầm hiểu biết của mình.

#### **10) Việc chơi cờ tướng cũng rèn luyện người chơi có được tính kiên nhẫn, sự tập trung và bình tĩnh để phán đoán tình huống**

Cờ tướng giúp người ta có tầm nhìn xa vì phải tính trước nhiều nước. Chơi cờ đúng với ý nghĩa đích thực của nó sẽ khiến cho người ta khôn ngoan hơn, uyển chuyển hơn, hài hòa hơn, tinh thần hơn trong cuộc sống.

Cờ tướng hỗ trợ không nhỏ cho nghề nghiệp của bạn hay con bạn sau này, nhất là khi nghề nghiệp đó liên quan tới các hoạt động trí tuệ.

Theo kết quả thống kê từ phần lớn người chơi cờ, chiến thắng sẽ không bao giờ đến với người có tính nóng vội, thiếu bình tĩnh. Chỉ có kiên nhẫn và tập trung đó là chìa khóa vàng dẫn đến thành công.

~~~~~

## Chương 2

# CƠ SỞ LÝ THUYẾT

### I. NGÔN NGỮ LẬP TRÌNH C#

Ngôn ngữ C# là một ngôn ngữ được dẫn xuất từ C và C++, nhưng nó được tạo từ nền tảng phát triển hơn. Microsoft bắt đầu với công việc trong C và C++ và thêm vào những đặc tính mới để làm cho ngôn ngữ này dễ sử dụng hơn. Nhiều trong số những đặc tính này khá giống với những đặc tính có trong ngôn ngữ Java. Không dừng lại ở đó, Microsoft đưa ra một số mục đích khi xây dựng ngôn ngữ này, và đó cũng là lý do nhóm thực hiện đã lựa chọn ngôn ngữ này.

#### 1) C# là ngôn ngữ đơn giản

C# loại bỏ một vài sự phức tạp và rắc rối của những ngôn ngữ như Java và C++, bao gồm việc loại bỏ những macro, những template, đa kế thừa, và lớp cơ sở ảo (virtual base class).

Ngôn ngữ C# đơn giản vì nó dựa trên nền tảng C và C++. Nếu chúng ta thân thiện với C và C++ hoặc thậm chí là Java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn.

#### 2) C# là ngôn ngữ hiện đại

Những đặc tính như là xử lý ngoại lệ, thu gom bộ nhớ tự động, những kiểu dữ liệu mở rộng, và bảo mật mã nguồn là những đặc tính được mong đợi trong một ngôn ngữ hiện đại. C# chứa tất cả những đặc tính trên.

Con trỏ được tích hợp vào ngôn ngữ C++. Chúng cũng là nguyên nhân gây ra những rắc rối của ngôn ngữ này. C# loại bỏ những phức tạp và rắc rối phát sinh bởi con trỏ. Trong C#, bộ thu gom bộ nhớ tự động và kiểu dữ liệu an toàn được tích hợp vào ngôn ngữ, sẽ loại bỏ những vấn đề rắc rối của C++.

#### 3) C# là ngôn ngữ hướng đối tượng

Những đặc điểm chính của ngôn ngữ hướng đối tượng (Object-oriented language) là sự đóng gói (encapsulation), sự kế thừa (inheritance), và đa hình (polymorphism). C# hỗ trợ tất cả những đặc tính trên.

#### 4) C# là ngôn ngữ mạnh mẽ và cũng mềm dẽo

Với ngôn ngữ C# chúng ta chỉ bị giới hạn ở chính bởi bản thân hay là trí tưởng tượng của chúng ta. Ngôn ngữ này không đặt những ràng buộc lên những việc có thể làm. C# được sử dụng cho nhiều các dự án khác nhau như là tạo ra ứng dụng xử lý văn bản, ứng dụng đồ họa, bản tính, hay thậm chí những trình biên dịch cho các ngôn ngữ khác.

### **5) C# là ngôn ngữ ít từ khóa**

C# là ngôn ngữ sử dụng giới hạn những từ khóa. Phần lớn các từ khóa được sử dụng để mô tả thông tin. Chúng ta có thể nghĩ rằng một ngôn ngữ có nhiều từ khóa thì sẽ mạnh hơn. Điều này không phải sự thật, ít nhất là trong trường hợp ngôn ngữ C#, chúng ta có thể tìm thấy rằng ngôn ngữ này có thể được sử dụng để làm bất cứ nhiệm vụ nào.

### **6) C# là ngôn ngữ hướng module**

Mã nguồn C# có thể được viết trong những phần được gọi là những lớp, những lớp này chứa các phương thức thành viên của nó. Những lớp và những phương thức có thể được sử dụng lại trong ứng dụng hay các chương trình khác. Bằng cách truyền các mẫu thông tin đến những lớp hay phương thức chúng ta có thể tạo ra những mã nguồn dùng lại có hiệu quả.

## **II. LÝ THUYẾT ÁP DỤNG**

### **1) Lập trình hướng đối tượng**

Cờ tướng là môn thể thao trí tuệ do con người sáng tạo, chính vì vậy, để tạo ra trò chơi này thì lập trình hướng đối tượng là định hướng tốt cho quá trình thực hiện đồ án, do lập trình hướng đối tượng có được được những yếu tố:

- Dựa trên suy nghĩ của con người về thực thể, các thuộc tính và hành vi của chúng.
- Lập trình hướng đối tượng cho phép kết hợp các thực thể, hành vi trong cuộc sống thực với các hàm, dữ liệu chương trình.
- Lập trình hướng đối tượng cho phép phân tích và thiết kế ứng dụng dưới dạng các thực thể hoặc đối tượng.
- Mô phỏng theo cách suy nghĩ của con người.

- Mã lệnh và dữ liệu được gắn kết với nhau thành một thực thể duy nhất là đối tượng.
- Sự tương đồng giữa các đối tượng trong cuộc sống thực với các đối tượng trong chương trình.

Ngoài ra, với những đặc điểm của lập trình hướng đối tượng tạo nên những thuận lợi cho quá trình lập trình ứng dụng:

- Kế thừa: Xây dựng các lớp mới từ các lớp cũ thông qua sự kế thừa. Các lớp dẫn xuất (lớp con), có thể thừa hưởng dữ liệu và các phương thức của lớp cơ sở (lớp cha) ban đầu, đồng thời, có thể bổ sung các thành phần dữ liệu và các phương thức mới vào những thành phần dữ liệu và các phương thức mà nó thừa kế từ lớp cơ sở.
- Đóng gói: cơ chế ràng buộc dữ liệu và thao tác trên dữ liệu đó thành một thể thống nhất, tránh được các động tác bất ngờ tuu72 bên ngoài. Đây là phương thức che giấu thông tin so với các ngôn ngữ lập trình cấu trúc.
- Đa hình: Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở của nó, tức là có cùng chức năng nhưng các lớp khác nhau sẽ hành xử khác nhau.

Cụ thể hóa áp dụng lập trình hướng đối tượng vào trò chơi Cờ Tướng (phân tích chi tiết ở chương 3):

- Đối tượng là Cờ Tướng.
- Cờ Tướng bao gồm những yếu tố: giao diện, ván cờ (trạng thái của cờ), bàn cờ, quân cờ và người chơi. Mỗi yếu tố là một lớp hoặc từ nhiều lớp hình thành.
- Giao diện gồm các hình ảnh, âm thanh và các chức năng.
- Các quân Tướng, Sỹ, Tượng, Xe, Pháo, Mã, Tốt được kế thừa lớp Quân Cờ.

## 2) Trí tuệ nhân tạo

### a) Thuật giải heuristic

Một số đặc tính áp dụng vào trò chơi:

- Thường tìm được lời giải tốt, mặc dù không phải là tốt nhất.

- Thực hiện dễ dàng nhanh chóng so với thuật giải tối ưu.
- Khá tự nhiên gần gũi với cách giải của con người.

Cụ thể trong trò chơi cờ tướng, hàm heuristic dùng để đánh giá độ tốt nước đi tiếp theo của quân cờ cũng như hạn chế giá trị đồ tốt của người chơi.

### b) Cây trò chơi – minimax – cắt tỉa alpha-beta

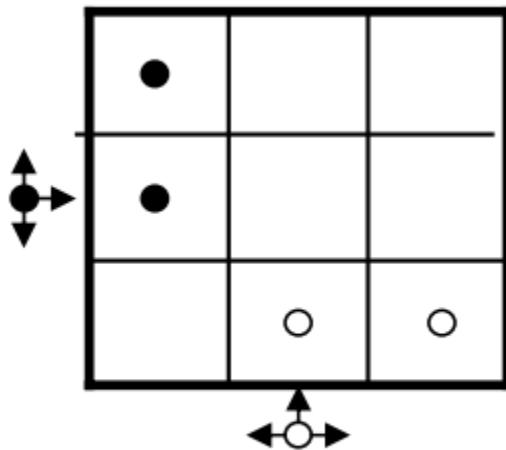
- [1] Cây trò chơi
    - Năm 1950, Claude Shannon đã viết chương trình chơi cờ đầu tiên.
    - Nghiên cứu các chiến lược chơi cho máy tính với các trò chơi có đối thủ (có hai người tham gia)
    - Việc giải quyết bài toán này có thể đưa về bài toán tìm kiếm trong không gian trạng thái, tức là tìm một chiến lược chọn các nước đi hợp lệ cho máy tính.
    - Tuy nhiên, vấn đề tìm kiếm ở đây phức tạp hơn so với vấn đề tìm kiếm trước, vì người chơi không biết trước đối thủ sẽ chọn nước đi nào tiếp theo.
    - Các trò chơi có đối thủ có các đặc điểm: hai người thay phiên nhau đưa ra các nước đi tuân theo các luật của trò chơi (các nước đi hợp lệ), các luật này là như nhau đối với cả hai người chơi, và cụ thể trong đồ án này là cờ tướng.
    - Cả hai người chơi đều biết đầy đủ các thông tin về tình thế cuộc chơi.
    - Thực hiện trò chơi là người chơi tìm kiếm nước đi tốt nhất trong số rất nhiều nước đi hợp lệ, tại mỗi lượt chơi của mình, sao cho sau một dãy nước đi đã thực hiện người chơi phải thắng cuộc.
    - Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm trong không gian trạng thái : Trạng thái ban đầu là sự sắp xếp các quân cờ của hai bên lúc bắt đầu cuộc chơi, các toán tử là các nước đi hợp lệ, các trạng thái kết thúc là các tình thế mà cuộc chơi dừng, một hàm kết cuộc ứng mỗi trạng thái kết thúc với một giá trị nào đó.
    - Cây trò chơi được xây dựng như sau:
- Gốc của cây ứng với trạng thái ban đầu.

Gọi đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen). Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó.

Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hặc đều là Đen, các lá của cây ứng với các trnag thái kết thúc.

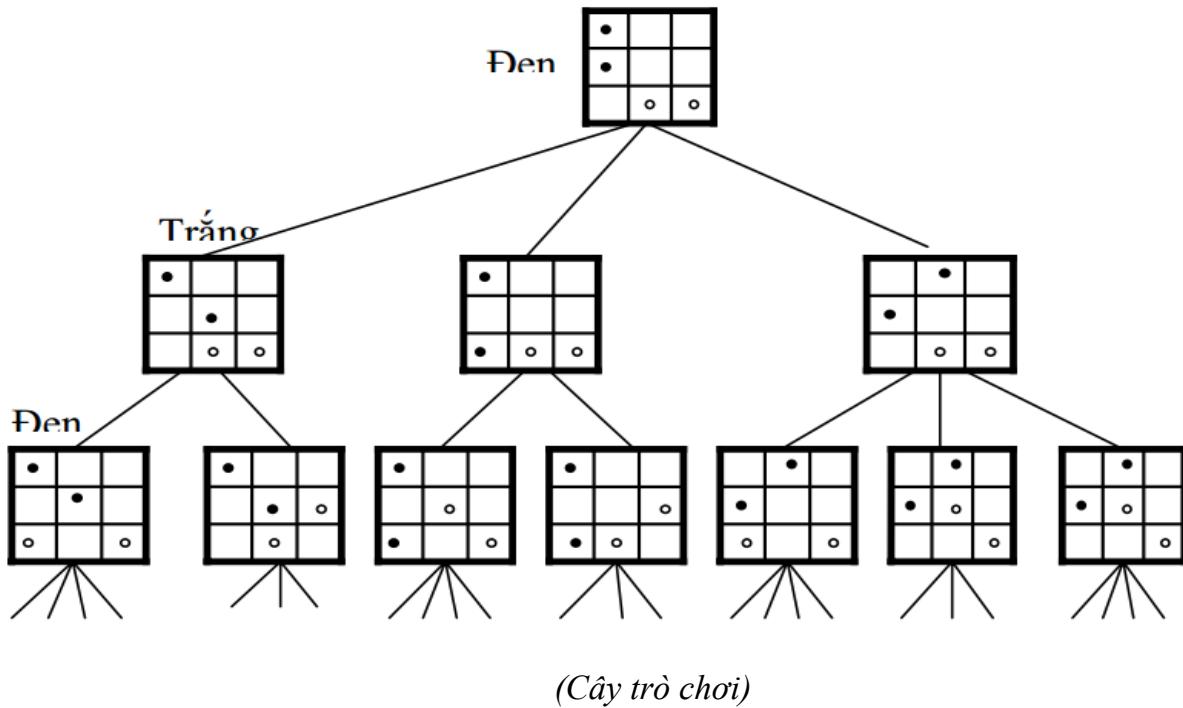
Ví dụ:

- Có hai quân Trắng và hai quân Đen, ban đầu được xếp vào bàn cờ  $3 \times 3$  (Hình vẽ).
- Quân Đen có thể đi tới ô trống ở bên phải, ở trên hoặc ở dưới. Quân Trắng có thể đi tới trống ở bên trái, bên phải, ở trên.
- Quân Đen nếu ở cột ngoài cùng bên phải có thể đi ra khỏi bàn cờ, quân Trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ.
- Ai đưa hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình thế bắt đói phuong không đi được cũng sẽ thắng.



## Trò chơi Dodgem

Giả sử đen đi trước



- [2] Thủ tục minimax

Khi chơi các trò chơi có thể triển khai hết không gian trạng thái, khó khăn chủ yếu là phải tính toán phản ứng của đối thủ. Một cách xử lý đơn giản nhất là giả sử đối thủ của bạn cũng sử dụng kiến thức về không gian trạng thái giống như bạn và áp dụng kiến thức đó kiên định để thắng cuộc. Mặc dù giả thiết này có những hạn chế của nó nhưng nó cũng cho chúng ta một cơ sở hợp lý để dự đoán hành vi của đối thủ. Minimax sẽ tìm kiếm không gian của trò chơi này theo giả thiết đó.

Hai đối thủ trong một trò chơi được gọi là MIN và MAX. MAX đại diện cho đối thủ quyết giành thắng lợi hay cố gắng tối đa hóa ưu thế của mình. Ngược lại MIN là đối thủ cố gắng tối thiểu hóa điểm số của MAX. Ta giả thiết MIN cũng dùng cùng những thông tin như MAX.

Khi áp dụng thủ tục Minimax, chúng ta đánh dấu luân phiên từng mức trong không gian tìm kiếm phù hợp với đối thủ có nước đi ở mức đó. Trong ví dụ trên, MIN được quyền đi trước, từng nút lá được gán giá trị 1 hay 0 tùy theo kết quả đó là thắng cuộc đối với MAX hay MIN. Minimax sẽ truyền các giá trị này lên cao dần trên đồ thị qua các nút cha mẹ kế tiếp nhau theo luật sau:

Nếu trạng thái cha mẹ là nút MAX, gán cho nó giá trị tối đa của các con cháu của nó.

Nếu trạng thái cha mẹ là nút MIN, gán cho nó giá trị tối thiểu của các con cháu của nó.

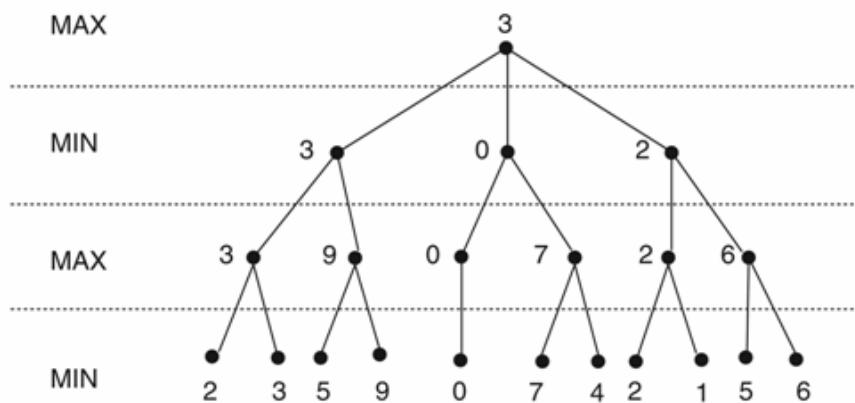
Giá trị được gán cho từng trạng thái bằng cách đó sẽ chỉ rõ giá trị của trạng thái tốt nhất mà đối thủ này có thể hy vọng đạt được. Các giá trị này sẽ được dùng để lựa chọn các nước đi có thể có.

- Áp dụng minimax đến độ sâu lớp cố định

Khi áp dụng Minimax cho các trò chơi phức tạp, hiếm khi có khả năng mở rộng đồ thị không gian trạng thái đến các nút lá. Thay vào đó không gian trạng thái này chỉ có thể được triển khai đến một số mức xác định phụ thuộc tiềm năng về thời gian và bộ nhớ chặng hạn. Chiến lược này được gọi là tính trước n nước đi ( $n$  – move lookahead). Vì giá trị các nút trong đồ thị con này không phải là trạng thái kết thúc của trò chơi nên chúng không phản ánh giá trị thắng cuộc hay thua cuộc. Chúng chỉ có thể được gán một giá trị phù hợp với một hàm đánh giá heuristic nào đó. Giá trị được truyền ngược về nút gốc không cung cấp thông tin thắng cuộc hay thua cuộc mà chỉ là giá trị heuristic của trạng thái tốt nhất có thể tiếp cận sau n nước đi kể từ nút xuất phát. Việc tính trước này sẽ làm tăng hiệu quả của heuristic vì nó được áp dụng vào một phạm vi lớn hơn trong không gian trạng thái. Minimax sẽ hợp nhất tất cả các giá trị của các nút con cháu của một trạng thái thành một giá trị duy nhất cho trạng thái đó.

Trong các đồ thị trò chơi được tìm kiếm bằng mức hay lớp, MAX và MIN luân phiên nhau chọn các nước đi. Mỗi nước đi của một đối thủ sẽ xác định một lớp mới trên đồ thị. Các chương trình trò chơi nói chung đều dự tính trước một độ sâu lớp cố định (thường được xác định bằng các giới hạn về không gian hoặc thời gian của máy tính). Các trạng thái trên mức đó được đánh giá theo các heuristic và các giá trị này sẽ được truyền ngược lên bằng thủ tục Minimax, sau đó thuật toán tìm

kiếm sẽ dùng các giá trị vừa nhận được để chọn lựa một nước trong số các nước đi kế tiếp. Bằng cách tối đa hóa cho các cha mẹ MAX và tối thiểu hóa cho các cha mẹ MIN, những giá trị này đi lùi theo đồ thị đến con của trạng thái hiện hành. Sau đó trạng thái hiện hành dùng chúng để tiến hành lựa chọn trong các con của nó. Hình sau trình bày quá trình Minimax trên một không gian trạng thái giả thuyết tính trước bốn lớp.



(Minimax trong không gian trạng thái giả định)

- Thủ tục cắt tỉa alpha beta

Minimax yêu cầu phải có sự phân tích qua hai bước đối với không gian tìm kiếm: Bước đầu truyền xuống đến độ sâu của lớp áp dụng heuristic và bước sau để truyền ngược các giá trị trên cây. Minimax lần theo tất cả các nhánh trong không gian bao gồm cả những nhánh mà một thuật toán thông minh hơn có thể bỏ qua hay tóm tắt. Các nhà nghiên cứu trong lĩnh vực chơi game đã xây dựng một kỹ thuật tìm kiếm gọi là cắt tỉa alpha –beta nhằm nâng cao hiệu quả tìm kiếm trong các bài toán trò chơi hai đối thủ.

Ý tưởng của tìm kiếm alpha – beta rất đơn giản: Thay vì nếu như tìm kiếm toàn bộ không gian đến một độ sâu lớp cố định, tìm kiếm alpha – beta thực hiện theo kiểu tìm kiếm sâu. Có hai giá trị, gọi là alpha và beta được tạo ra trong quá trình tìm kiếm. Giá trị alpha liên quan với các nút MAX và có khuynh hướng không

bao giờ giảm. Ngược lại giá trị beta liên quan đến các nút MIN và có khuynh hướng không bao giờ tăng. Giả sử có giá trị alpha của một nút MAX là 6, MAX không cần phải xem xét giá trị truyền ngược nào nhỏ hơn hoặc bằng 6 có liên quan với một nút MIN nào đó bên dưới. Alpha là giá trị thấp nhất mà MAX có thể nhận được sau khi cho rằng MIN cũng sẽ nhận giá trị tốt nhất của nó. Tương tự nếu MIN có giá trị beta là 6 nó cũng không cần xem xét các nút nằm dưới nó có giá trị lớn hơn hoặc bằng 6.

Để bắt đầu thuật toán tìm kiếm alpha – beta, ta đi xuống hết độ sâu lớp theo kiểu tìm kiếm sâu, đồng thời áp dụng đánh giá heuristic cho một trạng thái và tất cả các trạng thái anh em của nó. Giả thuyết tất cả đều là nút MIN. Giá trị tối đa của các nút MIN này sẽ được truyền ngược lên cho nút cha mẹ (là một nút MAX). Sau đó giá trị này được gán cho ông bà của các nút MIN như là một giá trị beta kết thúc tốt nhất. Tiếp theo thuật toán này sẽ đi xuống các nút cháu khác và kết thúc việc tìm kiếm đối với nút cha mẹ của chúng nếu gặp bất kỳ một giá trị nào lớn hơn hoặc bằng giá trị beta này. Quá trình này gọi là cắt tỉa beta ( $\beta$  cut). Cách làm tương tự cũng được thực hiện cho việc cắt tỉa alpha ( $\alpha$  cut) đối với các nút cháu của một nút MAX.

Hai luật cắt tỉa dựa trên các giá trị alpha và beta là:

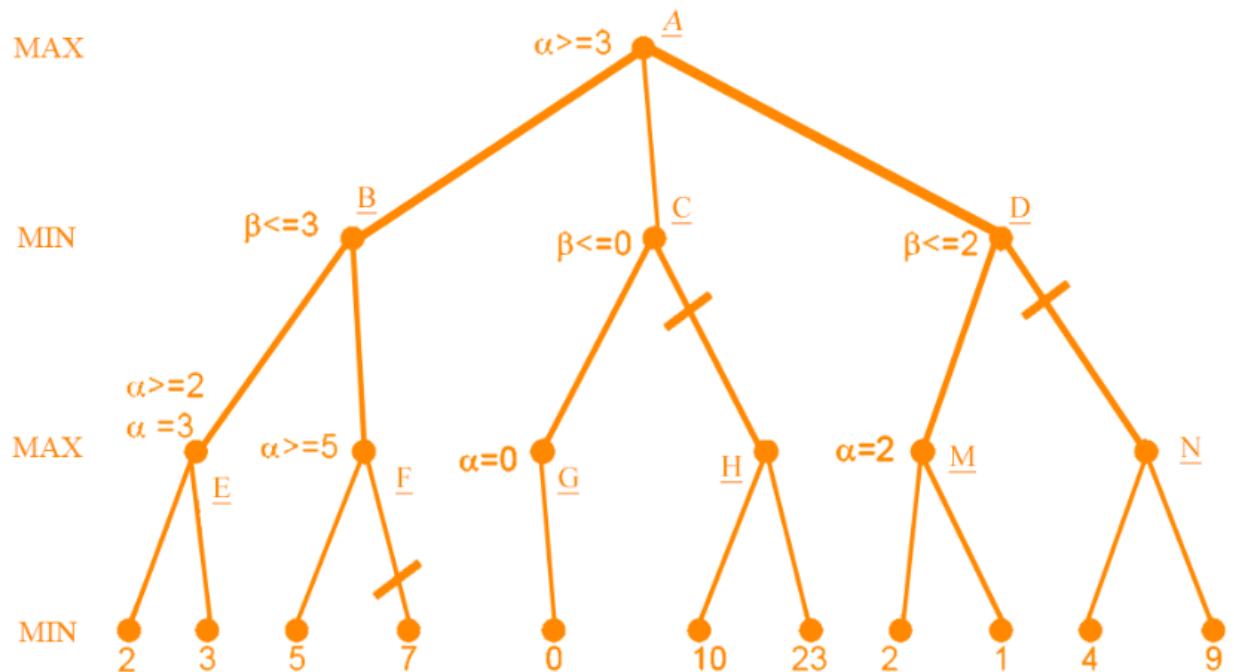
Quá trình tìm kiếm có thể kết thúc bên dưới một nút MIN nào có giá trị beta nhỏ hơn hoặc bằng giá trị alpha của một nút cha MAX bất kỳ của nó.

Quá trình tìm kiếm có thể kết thúc bên dưới một nút MAX nào có giá trị alpha lớn hơn hoặc bằng giá trị beta của một nút cha MIN bất kỳ của nó.

Việc cắt tỉa alpha – beta như vậy thể hiện quan hệ giữa các nút ở lớp n và các nút ở lớp n+2 và do quan hệ đó toàn bộ các cây con bắt nguồn ở lớp n+1 đều có thể loại khỏi việc xem xét. Chú ý rằng giá trị truyền ngược thu được hoàn toàn giống

như kết quả Minimax, đồng thời tiết kiệm được các bước tìm kiếm một cách đáng kể.

[3] Ví dụ:



Ở đây chúng ta cũng xét từ trái qua phải bắt đầu từ nút gốc và nút con bên trái sẽ được ưu tiên duyệt trước. Duyệt nguyên cây này sẽ khá dài dòng nhưng để bạn hiểu tôi sẽ viết ra các bước sau.

Xét duyệt từ trên gốc xuống sâu (vì ban đầu chưa hề tồn tại giá trị alpha hay beta của các nút). Nút đầu tiên ta duyệt là E sẽ gặp giá trị 2 (alpha  $\geq$  2), khi đó ở trên chưa có giá trị beta để ta có thể so sánh nên sẽ bắt đầu duyệt con tiếp theo của nút E đó và ở đây ta sẽ chọn cho alpha = 3 (Max).

Lưu ý là chúng ta luôn luôn duyệt từ trái sang phải và phải lần lượt từng nhánh một, sau đó sang nhánh tiếp theo cùng gốc. Vậy nên tiếp theo chúng ta sẽ đưa giá trị alpha này lên nút B (Min) và nút B – beta  $\leq$  3, sau đó nút F sẽ được duyệt, và ta phải tìm alpha của F. Khi duyệt con đầu tiên mang giá trị 5 vậy alpha của F – alpha  $\geq$  5.

Tại B – beta  $\leq$  3 và tại F – alpha  $\geq$  5. Như vậy chúng ta không cần xem xét các nút con còn lại của F vì cái ta cần ở đây chỉ là khoảng  $\leq 3$  nên ta cắt toàn bộ các con còn lại.

Sau khi duyệt toàn bộ các con của B thì tại  $B - \text{beta} = 3$ , và tại nút A- $\text{alpha} \geq 3$ .

### III. KẾT LUẬN

- Lập trình hướng đối tượng giúp tổ chức chương trình dễ dàng, rõ ràng và dễ chỉnh sửa.
- Giải thuật Hueristic đánh giá tốt nước đi của quân cờ, giúp máy thực hiện các nước đi tốt nhất.
- Thủ tục minimax, cắt tỉa alpha-beta giúp tối ưu và lựa chọn được các nước đi tốt nhất.

QUESTION

## Chương 3

# PHÂN TÍCH VÀ THIẾT KẾ

## I. PHÂN TÍCH BÀI TOÁN

### 1) Thông tin dữ liệu

#### a) Các đối tượng

- Bàn cờ: được định nghĩa là class BanCo.
- Người chơi: được định nghĩa là class Player.
- Quân cờ: là lớp cơ sở, được định nghĩa là class QuanCo, và các lớp dẫn xuất như: Tướng (class Tuong), Sỹ (class Sy), Tịnh (class Tinh), Xe (class Xe), Pháo (class Phao), Mã (class Ma), Chốt (class Chot) kế thừa từ lớp cơ sở.
- Ván cờ: được định nghĩa là class VanCo, đây là lớp quản lý tất cả các trường hợp trong từng trạng thái của ván cờ đang chơi.
- Giao diện: là nơi thể hiện hình ảnh và tương tác với người chơi.

#### b) Dữ liệu bàn cờ

- Ta sẽ dùng một mảng static hai chiều [10x9] với mỗi phần tử là một ô cờ struct tương ứng với 1 vị trí trên bàn cờ, trên mỗi phần tử sẽ có các thuộc tính: Hàng, Cột, Trống, Tên, Phe, ThứTự.
- Nếu vị trí đó có quân cờ thì Trống=false, Tên=Tên quân cờ, Phe=Phe của quân cờ đó(0 hoặc 1), ThứTự=Thứ tự của quân cờ. Nếu vị trí đó không có quân cờ thì Trống = true, Tên=" ", Phe=2, ThứTự=" ".
- PictureBox CanMove: thể hiện hình ảnh quân cờ có thể đi được hay không trên ô cờ đó.

#### c) Dữ liệu quân cờ

- Ta sẽ tạo lớp Player để quản lý các quân cờ, trong lớp Player sẽ tạo các thể hiện của các lớp quân cờ(Tuong, Sy, Tinh, Xe, Phao, Ma, Chot), tức là khai tạo vị trí ban đầu của từng quân cờ.
- Trong Ván Cờ sẽ có 2 thể hiện của lớp Player tương trưng cho 2 người chơi.

#### d) **Dữ liệu ván cờ**

Ta sẽ xây dựng lớp VanCo bao gồm các thuộc tính và phương thức quản lý ván cờ đó. Gồm có:

- 2 thể hiện của lớp Player
- Nước đi: được định nghĩa là class NuocDi, bao gồm các quân cờ, vị trí có thể đến.
- các options trong ván cờ như tên 2 người chơi, âm thanh, thời gian còn lại của từng người chơi, các quân cờ bị ăn của người chơi...
- Các phương thức để kiểm tra và thay đổi trạng thái của ván cờ như Ăn quân cờ, kiểm tra có tướng nào đang bị chiếu không, nước đi nào là chiếu bí... Và các phương thức hỗ trợ cho người chơi như thoát.

### 2) Xử lý tính toán

#### a) **Bắt đầu chương trình**

Khi người chơi nhấn new game thì chương trình bắt đầu chạy vào hàm New Game ở class VanCo để load các con cờ lên, khởi tạo phe đầu tiên đánh là người chơi.

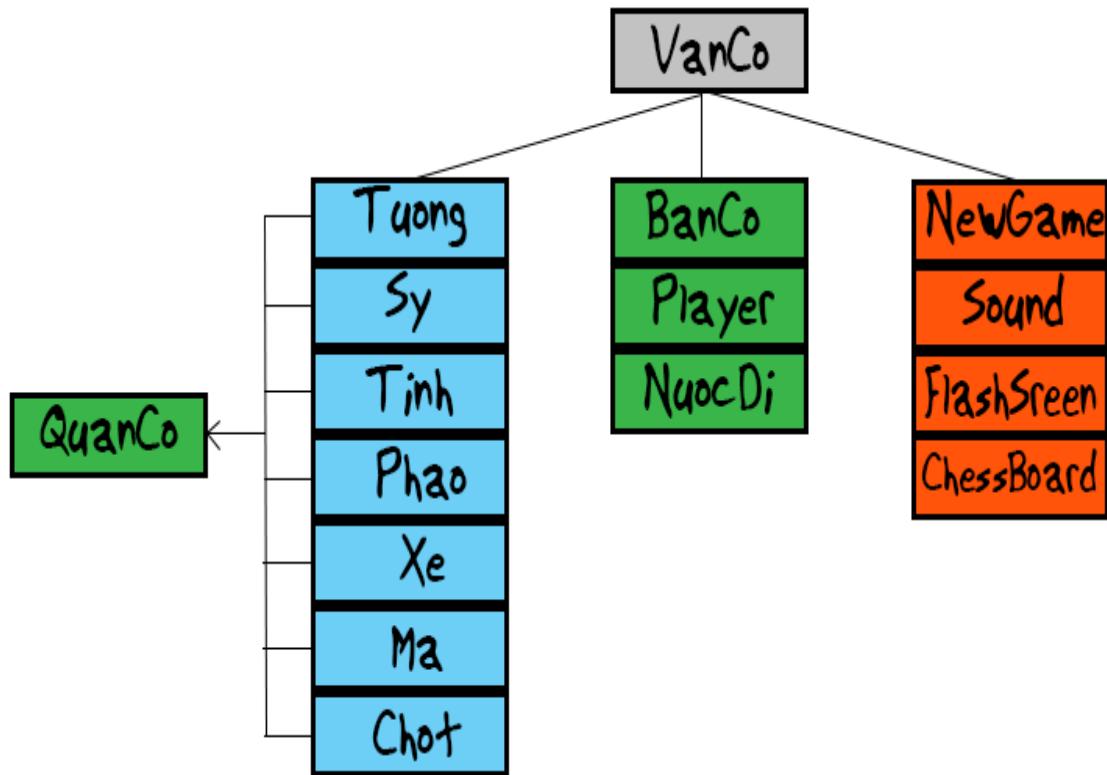
#### b) **Các xử lý khi chương trình chạy**

- Mô tả cách sinh nước cờ: Từ trạng thái cho trước chương trình sẽ tìm con cờ hiện tại có trên bàn cờ và nó có thể di chuyển (có đường đi và không có đang bảo vệ tướng) bỏ vào 1 mảng quân cờ, sau đó tạo mảng các nước đi khoảng 200 phần tử, chạy qua mảng quân cờ, mỗi quân cờ di chuyển đến 1 vị trí mới thì ta bỏ vào mảng nước đi, chạy qua tất cả quân cờ tìm các nước đi của nó.
- Cách lượng giá độ tốt của một trạng thái: Căn cứ vào một thể cờ máy sẽ gán cho nó một điểm số (lượng giá tĩnh) để đánh giá độ tốt - xấu. Nhờ điểm này máy mới có thể so sánh các thể cờ với nhau và biết chọn nước đi tốt nhất.
- Cách lựa chọn nước đi tốt: Có nhiều cách chọn nước đi tốt, trong đó thuật toán Minimax (cắt tỉa Alpha Beta) là một cách. Thuật toán giúp việc tìm kiếm nước đi tối ưu nhanh hơn

#### c) **Điều kiện dừng**

Điều kiện kết thúc của trò chơi là tướng của 1 trong 2 bị bí đường đi, hoặc là 1 trong 2 phe tướng không còn trên bàn cờ.

### 3) Sơ đồ cấu trúc chương trình



## II. THIẾT KẾ CẤU TRÚC DỮ LIỆU VÀ TRẠNG THÁI

### 1) Bàn cờ

Các thuộc tính của lớp **BanCo** được khai báo static để tiện xử lý trong các lớp khác. Bàn cờ là mảng hai chiều, với các phần tử là ô cờ.

#### a) Thuộc tính

- **struct Oco**

Chứa các thông tin của 1 ô cờ, gồm có:

**public int Hang:** chứa giá trị hàng của ô cờ.

**public int Cot:** chứa giá trị cột của ô cờ.

public string ThuTu: thứ tự của ô cờ.

public bool Trong: kiểm tra ô cờ có trống hay không, nếu trống thì giá trị là true và ngược lại.

public string Ten: tên của quân cờ, khi ô cờ có quân cờ.

public int Phe: phe của quân cờ, gồm phe 1 và phe 2.

public PictureBox CanMove: hiển thị hình ảnh những nơi có thể đi của quân cờ.

- static OCo[,] ViTri = new OCo[10, 9]

Mảng lưu 90 vị trí, tương ứng với [10x9] vị trí trên bàn cờ, mỗi phần tử của mảng là một Oco.

### b) Phương thức

- static BanCo()

Constructor khởi tạo bàn cờ trống.

Cách khởi tạo: Tạo vòng lặp for để quét mảng 2 chiều, với từng phần tử ta khởi tạo giá trị thích hợp như sau:

```

ViTri[i, j].Hang = i;
ViTri[i, j].Cot = j;
ViTri[i, j].Trong = true;
ViTri[i, j].Ten = "";
ViTri[i, j].ThuTu = "";
ViTri[i, j].Phe = 0;
ViTri[i, j].CanMove = new PictureBox();
ViTri[i, j].CanMove.Image =
UIT_ChineseChess.Properties.Resources.CanMove;
ViTri[i, j].CanMove.Width = 28;
ViTri[i, j].CanMove.Height = 28;
ViTri[i, j].CanMove.BackColor = Color.Transparent;
ViTri[i, j].CanMove.Top = i * 46 + 225;
```

```

ViTri[i, j].CanMove.Left = j * 52 + 180;
ViTri[i, j].CanMove.Cursor = Cursors.Hand;
ViTri[i, j].CanMove.Visible = false;
• public static void ResetCanMove()

```

Đây là hàm tắt thông báo những vị trí mà quân cờ có thể di chuyển đến.

Cách thực hiện: tạo vòng lặp for, với mỗi phần tử ta gán như sau:

```

BanCo.ViTri[i, j].CanMove.Visible = false;
• public static void ResetBanCo()

```

Đây là hàm thiết lập lại bàn cờ

Cách thực hiện: Tạo vòng lặp for, với mỗi phần tử, ta gán lại giá trị thích hợp như sau:

```

ViTri[i, j].Hang = i;
ViTri[i, j].Cot = j;
ViTri[i, j].Trong = true;
ViTri[i, j].Ten = "";
ViTri[i, j].ThuTu = "";
ViTri[i, j].Phe = 0;
ViTri[i, j].CanMove.Visible = false;

```

## 2) Quân cờ

Trong source code, lớp cơ sở quân cờ được định nghĩa là class QuanCo

### a) Thuộc tính

- public int Hang

Giá trị Hàng của quân cờ

- public int Cot

Giá trị Cột của quân cờ

- public string Ten

Tên quân cờ

- public int Phe

Phe 0 hoặc 1

- **public string** ThuTu

Thứ tự Trái hoặc Phải(vd: Pháo trái, Pháo phải), đối với chốt thì Thứ Tự sẽ là 0 đến 4

- **public int** TrangThai

Trạng thái của quân cờ (còn sống => TrangThai=1, đã bị ăn => TrangThai=0)

- **public PictureBox** picQuanCo = **new PictureBox()**

PictureBox chứa hình ảnh của quân cờ

- **public bool** Khoa

Thuộc tính Khóa, không cho người dùng tương tác vào quân cờ để di chuyển  
(Khoa=false tức là quân cờ có thể di chuyển được)

### b) Phương thức

- **public QuanCo()**

Constructor thiết lập các giá trị mặc định khi quân cờ được tạo ra: Hang=10, Cot=10 (vị trí [10x10] không có trên bàn cờ), Ten="" "", Phe=2 (ngoài 2 giá trị 0,1), ThuTu="" "", TrangThai=0, Khoa=True

- **public void** KhoiTao(**int** phe, **string** name, **string** thutu, **int** stt, **bool** khoa, **int** x, **int** y)

Khởi tạo quân cờ với các giá trị tương ứng

Các khởi tạo:

Hang = x;

Cot = y;

Ten = name;

TrangThai = stt;

Phe = phe;

ThuTu = thutu;

Khoa = khoa;

BanCo.ViTri[x, y].Hang = x;

BanCo.ViTri[x, y].Cot = y;

BanCo.ViTri[x, y].Ten = name;

```

BanCo.ViTri[x, y].ThuTu = thutu;
BanCo.ViTri[x, y].Phe = phe;
BanCo.ViTri[x, y].Trong = false;
picQuanCo.MouseClick += new MouseEventHandler(picQuanCo_MouseClick);
DoTot = dotot;

```

- **public void draw()**

Phương thức vẽ quân cờ vào vị trí [Hang,Cot]

Với mỗi phe ta tải lên hình ảnh tương ứng cho từng phe.

- **public virtual int KiemTra(int i,int j)**

Kiểm tra quân cờ có thể di chuyển đến vị trí [i,j] theo đúng luật mà 2 tướng không đối mặt nhau hay không, trả về 0 nếu không đi được, trả về 1 nếu có thể đi, phương thức này sẽ được Override lại ở các lớp quân cờ dẫn xuất cụ thể.

- **public virtual int TuongAnToan(int i, int j)**

Kiểm tra nếu quân cờ di chuyển đến vị trí [i,j] thì tướng phe mình có an toàn không (không bị chiếu).

- **private void picQuanCo\_MouseClick(Object sender, MouseEventArgs e)**

Thao tác click chuột chọn quân cờ để di.

### 3) Các lớp dẫn xuất quân cờ

#### a) Mã

Kế thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm TuongAnToan(int i, int j)

- **public override int KiemTra(int i, int j)**

Hàm được dùng để kiểm tra tại vị trí hàng i, cột j quân Mã có thể di chuyển ở vị trí đó, kiểu trả về int nếu trả về 1 là có thể di chuyển, trả về 0 không thể di chuyển.

➤ Input: Vị trí [i,j] trên bàn cờ với i là cột, j là hàng.

➤ Cài đặt:

B1: tao biến bool turn = false;

Tìm các vị trí quân xe có thể đi

B2: Nếu i cách hàng hiện tại là 2 và j cách cột hiện tại là 1

B2.1: Đảm bảo vị trí này nằm trong bàn cờ

B2.1.1 Nếu vị trí cách hàng hiện tại là 1 trống

B2.1.1.1 Nếu tại vị trí hàng i, cột j trống) turn =

true;

Else

Nếu tại vị trí hàng i, cột j có phe khác với phe của quân mã

turn = true;

B3: Nếu i cách hàng hiện tại là 1 và j cách cột hiện tại là 2

B3.1: Đảm bảo vị trí này nằm trên bàn cờ

B3.1.1 Nếu vị trí cách cột hiện tại là 1 trống

B3.1.1.1 nếu tại vị trí hàng i, cột j trống) turn true;

Else

Nếu tại vị trí hàng i cột j có phe khác với phe

quân mã

Turn = true;

B4: Nếu quân tướng ở 2 phe ở cùng 1 cột và quân mã đang đứng trên cột này

B4.1: Tạo biến int ct = 0;

B4.2: Nếu cột j khác cột hiện tại

B4.2.1 Thiết lập ô cờ hiện tại của quân mã thành rỗng

B4.2.2 Tạo biến t chạy qua các hàng trên cột của quân tướng

B4.2.2.1 Nếu vị trí hàng t , cột của quân tướng trống ct++

B4.2.3: Nếu ct == 0 turn = false;

B4.2.4: Thiết lập lại các thuộc tính trên bảng cờ

B5: nếu turn return 1;

Else return 0;

➤ Ví dụ minh họa:

Mã kiemtra(2,3) nếu đi đến được vị trí (2,2) Hàm kiemtra trả về 1 và ngược lại.

- public override int TuongAnToan(int i, int j)

Hàm được dùng để tại hàng i, cột j tướng có an toàn không, hàm có kiểu trả về là kiểu int, trả về 1 nếu tướng an toàn, trả về 0 tướng không an toàn.

➤ Input: Hàng i, cột j.

➤ Cách cài đặt:

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

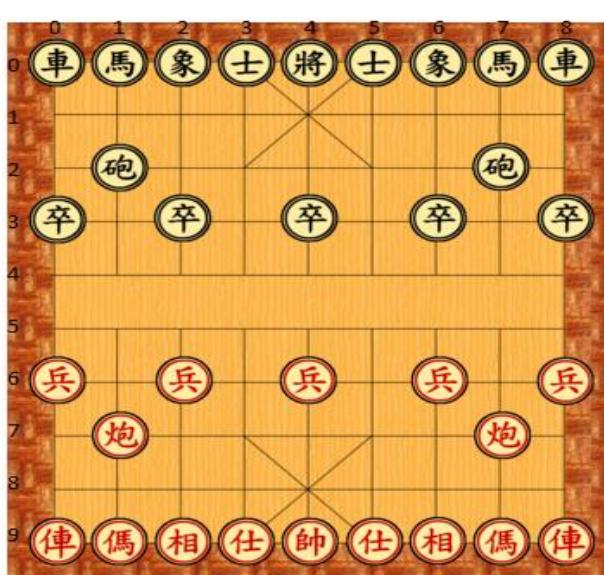
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

➤ Ví dụ:



(Hình ảnh mô phỏng vị trí quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm                                            | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|-------------------------------------------------------------------------|-----------------------|
| 9             | 1            | 7                  | 2                 | Thỏa điều kiện dòng if thứ 2, turn = true                               | Có thể di chuyển      |
| 9             | 1            | 8                  | 3                 | Không thỏa điều kiện vì vị trí (9, 2) không trống nên biến turn = false | Không thể di chuyển   |

### b) Pháo

Kế thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm TuongAn-Toan(int i, int j), các input và ý nghĩa hàm tương tự lớp đẫn xuất trên.

- public override int KiemTra(int i, int j)

B1: tạo biến bool turn = true;

B2: tạo biến kiểu int count = 0;

B3: nếu hàng i và cột j nằm ngoài bàn cờ turn = false;

B4: hàng i khác hàng hiện tại và cột j khác cột hiện tại turn = false;

B5: hàng i và cột j nằm trong bàn cờ

B5.1 Nếu vị trí hàng i, cột j trống

B5.1.1 i == Hang && j >= 0 &&& j <= 8

B5.1.1.1 if(j > Cot){

B5.1.1.1.1:for( int k =Cot + 1; k<=j;k++)

nếu vị trí hàng i, cột k không trống

Turn = false;

break;

B5.1.1.2 if(j < Cot)

B5.1.2.1 for( int k =j; k<= Cot -1;k++)

Nếu vị trí hàng i, cột k không trống

Turn = false;

break'

B5.3 If(j == Cot && i>=0 && i<=9){

B5.3.1 if(i > Hang)

B5.2.1.1 for( int k = Hang+ 1; k<=j;k++)

Nếu vị trí hàng k, cột j không trống

Turn = false;

break;

B5.3.2 if(i<Hang){

B5.2.2.1 for( int k =j; k<=Hang -1;k++)

nếu vị trí hàng k, cột j không trống

Turn = false;

Else{

B5.4: if( vị trí i, j có phe khác với quân pháo)

B5.4.1 if( i == Hang && j>= 0 &&& j<=8)

B5.4.1.1: Nếu j > Cot

B5.3.1.1.1 for( int k =Cot + 1; k<=j;k++)

Nếu vị trí hàng i, cột k không trống

count ++

B5.4.1.2: j < Cot

for( int k =j; k<= Cot -1;k++){

Nếu vị trí hàng i, cột k không trống

count++;

B5.4.1: if(j == Cot && i>=0 && i<=9)

B5.4.1.1: if(i > Hang){

B5.4.1.2: for( int k =Hang+ 1; k<=j;k++){

Nếu vị trí hàng k, cột j không trống

count ++;

B5.4.1.2: if(i<Hang){

B5.4.1.3: for( int k =j; k<=Hang -1;k++){

Nếu vị trí hàng k, cột j không trống

count++;

B5.4.2: If(count != 1) turn == false;

Else turn = false;

B5.5 quân tướng 2 phe trên cùng một cột và quân pháo ở cùng cột

B5.5.1 khởi tạo biến int ct = 0;

B5.2.2: if( cột j khác cột hiện tại

B5.2.2.1: Thiết lập ô cờ hiện tại của quân pháo thành rỗng

B5.2.2.1: Tạo biến t chạy qua các hàng trên cột của quân tướng

Nếu vị trí hàng t , cột của quân tướng trống)ct++;

Nếu ct == 0 turn = false;

B5.2.2.3 Thiết lập lại các thuộc tính trên bàn cờ

B5.6 Cho j == Cot && (i < hàng của quân tướng bên đỏ || i> hang của quân tướng đen

B5.6.1: Thiết lập ô cờ hiện tại của quân pháo thành rỗng

B5.6.2: Tạo biến t chạy qua các hàng trên cột của quân tướng{

Nếu vị trí hàng t , cột của quân tướng trống) ct++;

Nếu ct == 0 turn = false;

B5.6.3: Thiết lập lại các thuộc tính trên bàn cờ

B7:(turn) return 1;

Else return 0;

}

- public override int TuongAnToan(int i, int j)

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

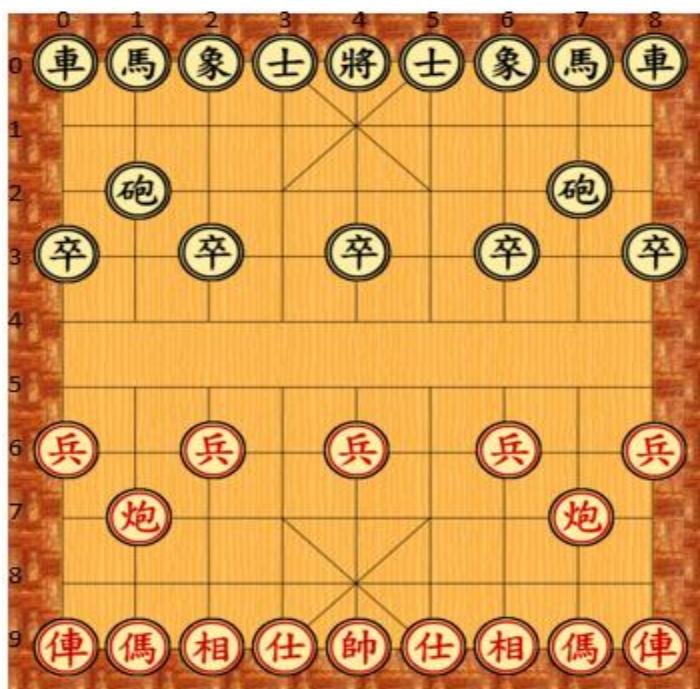
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

- Ví dụ minh họa:



(Hình ảnh minh họa vị trí quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm | Biến đếm khi Pháo bắn ngòi | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|------------------------------|----------------------------|-----------------------|
| 7             | 1            | 0                  | 1                 | Thỏa tất cả điều kiện        | Count == 1                 | Có thể di             |

|   |   |   |   |                                                                                                    |                           |
|---|---|---|---|----------------------------------------------------------------------------------------------------|---------------------------|
|   |   |   |   | Vì vị trí<br>(2, 1)<br>không<br>trống                                                              |                           |
| 7 | 1 | 1 | 1 | Sai điều kiện khi vị trí<br>(1, 1) trống nhưng ở<br>vị trí (2, 1) không<br>trống biến turn = false | Không<br>thể di<br>chuyển |

### c) Tịnh

Ké thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm TuongAn-Toan(int i, int j), các input và ý nghĩa hàm tương tự lớp dẫn xuất trên.

- public override int KiemTra(int i, int j)

B1: Tạo biến bool turn = false;

B2: Nếu Phe == 0

B2.1: Nếu i nằm nửa trên bàn cờ

B2.1.1: Nếu i, j cách hàng, cột hiện tại là 2 và ở các vị trí cách  
hàng, cột là 1 trống){

B2.1.1.1: Nếu vị trí i,j trống turn = true;

B2.1.1.2: Nếu vị trí i, j không trống và có phe khác với quân tịnh) turn =  
true;

B3: Phe == 1

B3.1: If( i nằm nửa dưới bàn cờ

B3.1.1: If vị trí i, j cách hàng, cột hiện tại là 2 và ở các vị trí cách  
hàng, cột là 1 trống

B3.1.1.1: Nếu vị trí i,j trống turn = true;

B3.1.1.2: Nếu vị trí i, j không trống và có phe khác với quân tịnh turn =  
true;

B4: quân tướng ở 2 phe ở cùng 1 cột và quân mã đang đứng trên cột này){

B4.1 tạo biến int ct = 0;

B4.1: Nếu cột j khác cột hiện tại

B4.1.1 Thiết lập ô cờ hiện tại của quân tịnh thành rỗng

B4.1.2: Tạo biến t chạy qua các hàng trên cột của quân tướng

Nếu vị trí hàng t , cột của quân tướng trống)ct++;

Nếu ct == 0turn = false;

Thiết lập lại các thuộc tính trên bảng cờ

B5 if(turn) return 1;

Else return 0;

- public override int TuongAnToan(int i, int j)

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

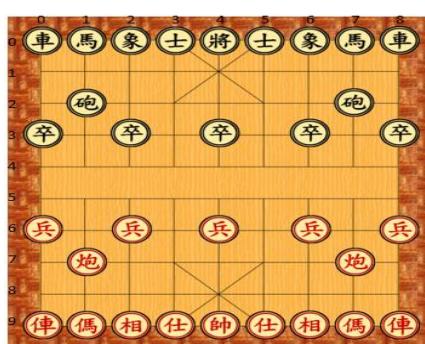
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

- Ví dụ minh họa



(hình ảnh minh họa vị trí của quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm                                                          | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|---------------------------------------------------------------------------------------|-----------------------|
| 9             | 2            | 7                  | 0                 | Thỏa các điều kiện và vị trí(8, 1) trống nên turn = true;                             | Có thể di chuyển      |
| 9             | 2            | 7                  | 1                 | Không thỏa điều kiện vì cột muốn đến không thỏa đường đi của quân tịnh, turn = false; | Không thể di chuyển   |

#### d) Xe

Được kế thừa từ class quân cờ và các thuộc tính khác giành riêng cho quân xe  
Function Kiemtra(int I,int j) để kiểm tra xem quân cờ có được đi ở vị trí tạo độ (I,j) trên bàn cờ hay không, nếu đi được sẽ trả về giá trị == 1. Nếu con cờ không đi được sẽ trả về là 0 ;

Function TuongAnToan(int I,int j) để kiểm tra xem quân cờ nếu đi ở vị trí tạo độ (I,j) trên bàn cờ thì nó con tướng có an toàn hay không.

- public override int KiemTra(int i, int j)

B1:Tạo biến turn== true kiểu bool để kiểm tra quân có thể đi được ở vị trí I,j hay không nếu biến turn== true quân cờ có thể đi được và hàm trả về giá trị =1, ngược lại turn == false quân cờ không đi được và trả về giá trị bằng 0.

B2: if( i và j nằm ngoài bàn cờ) turn =false ;

B3: if( i khác hàng và j khác cột) turn =false ;

B4: if(i == hằng và j = 0 đến bằng 8 )

B4.1: nếu j> Cot)

B4.1.1 for( k = cột +1 k < j ;k++)

Nếu ô cờ ở vị trí i,k trống ) thì turn = false ; thoát ;

B4.2: if ( j<Cot)

B4.2.1: For(k = j+1 k< hơn cột )

Nếu ô cờ ở vị trí i,k trống thì turn = false => thoát ;

B4.3: Bàn cờ ở vị trí i,j == false

nếu phe == phe thì turn == flase ;

B5 Nếu j bằng cột và i nằm trong ô cờ

B5.1: Nếu j>Hang

B5.1.1: for(k=Hang+1 k< i )

nếu bàn cờ ở vị trí (k,j) có quân cờ => turn = false => break;

B5.1.2 if(vị trí(i,j) có quân cờ

Nếu phe== phe thì turn == false;

B6 Vị trí cột của quân tướng của player0 == vị trí cột của quân tướng player 1  
và Cột = ván cờ phe 1

B6.1: tạo biến ct =0 ;

B6.2: Nếu j!=cột

B6.2.1: for(t == hàng của quan tướng player 0 +1; t<hàng cuản quân  
tướng player 1 )

if( banco.Vitri[t,Cot].Trong == false ) ct ++;

B6.2.2: if( ct==0) turn == false ;

}

B7 Nếu turn == true return 1;

Else return 0;

}

- public override int TuongAnToan(int i, int j)

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

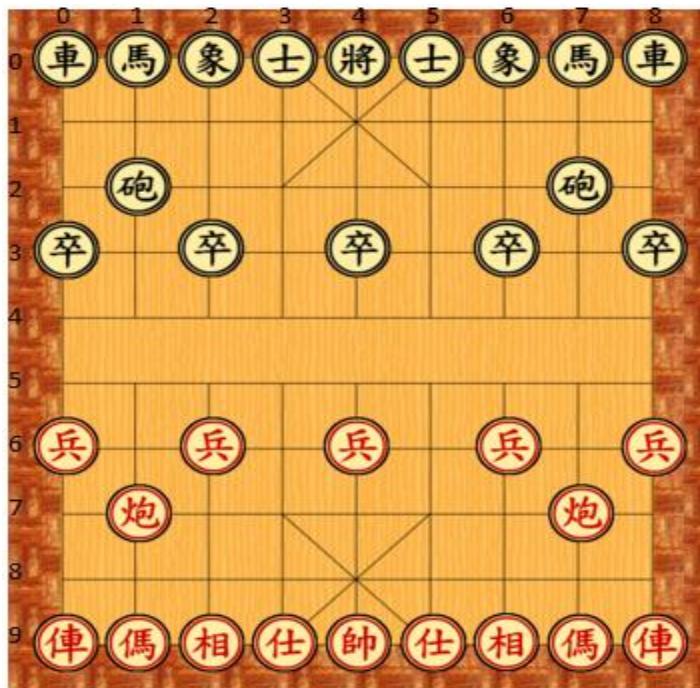
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

- Ví dụ minh họa:



| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm                                                                                       | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|--------------------------------------------------------------------------------------------------------------------|-----------------------|
| 9             | 0            | 7                  | 0                 | Thỏa các điều kiện nên biến turn = true;                                                                           | Có thể di chuyển      |
| 9             | 0            | 5                  | 0                 | Không thỏa điều kiện vì tại vị trí (6, 0) có quân tốt cùng phe chặn đường nên đến đó biến turn = false và dừng lại | Không thể di chuyển   |

### e) Chốt

Kế thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm TuongAn-Toan(int i, int j), các input và ý nghĩa hàm tương tự lớp dẫn xuất trên.

- public override int KiemTra(int i, int j)

B1: tạo biến turn == false

B2: Nếu phe ==0

B2.1: Nếu  $i >= 0$  và  $i <= 4$

B2.1.1: if( bàn cờ trống bằng true ) turn == true ;

B2.1.1: if( bàn cờ trống == false ) {

Nếu Phe!=Phe turn == true ;

B2.1: Nếu  $i > 4$  và  $i <= 9$

{

B2.1.1 (i==hang+1 và j==Cot) || i==Hang &&j==cột -1 || j == cot +1 )

{

B2.1.1.1if( i>=0 và i<=9 && j>=0 và j<=8)

Nếu bàn cờ trống == true turn == true;

Nếu bàn cờ trng == flase )

if( phe != phe ) turn == true ;

B3:if( phe ==1)

B3.1:if( i<=9 và i>=5)

B3.1.1: Nếu ((i==hang -1 )&& j== cột)){

B3.1.2: Nếu bàn cờ trống bằng true turn == true ;

B3.1.2: Nếu bàn cờ trống == false

if( Phe!=Phe ) turn == true ;

}

B3.2: nếu i<5 và i>=5

B3.2.1: if( (i== hang-1 và j==Cot) || i== Hang &&j==cột -1 || j == cot +1

)

B3.2.1.1if( i>=0 và i<=9 && j>=0 và j<=8)

Nếu bàn cờ trống == true turn == true;

Nếu bàn cờ trng == flase

Nếu phe != phe ) turn == true ;

B4: Nếu turn == true => return 1;

Else return 0;

}

- public override int TuongAnToan(int i, int j)

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

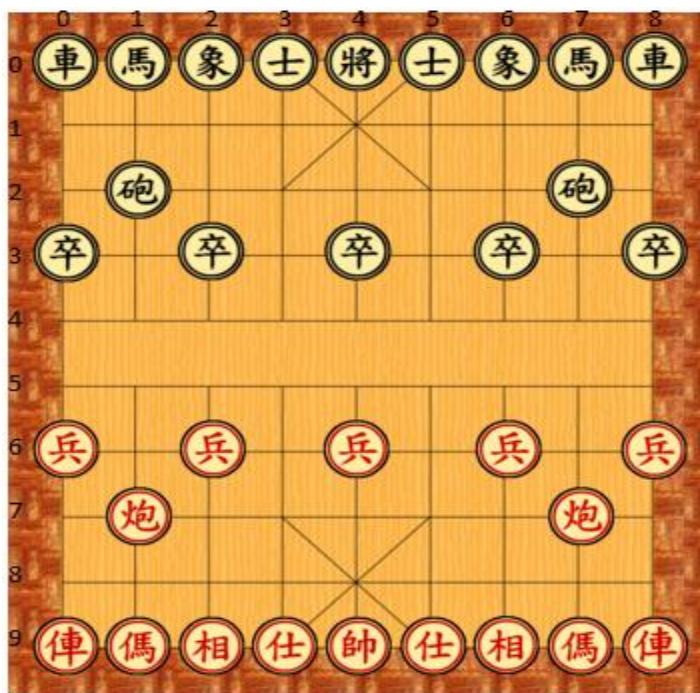
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

- Ví dụ minh họa:



(Hình ảnh minh họa vị trí quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm                                                                                                                                | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 6             | 0            | 5                  | 0                 | Thỏa các điều kiện ở phe == 1; turn = true;                                                                                                                 | Có thể di chuyển      |
| 6             | 0            | 6                  | 1                 | Không thỏa các điều kiện ở phe == 1 vì quân cờ muốn di chuyển đến vị trí (6, 1) mà quân cờ thuộc phe 1 nên không vào được vào if ở bước 3.1.2, turn = false | Không thể di chuyển   |

### f) Sỹ

Kế thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm

TuongAnToan(int i, int j), các input và ý nghĩa hàm tương tự lớp dũng xuất trên.

- public override int KiemTra(int i, int j)

B1: Tạo biến bool turn = false;

B2: Nếu i nằm trong khoảng từ 0 đến 2 và j nằm trong khoảng từ 3 đến 5

B2.1: Nếu i cách hàng hiện tại là 1 và j cách cột hiện tại là 1

Nếu vị trí i, j trống turn = true;

Nếu vị trí i, j không trống và có phe khác với quân sỹ) turn true;

B3: quân tướng ở 2 phe ở cùng 1 cột và quân mã đang đứng trên cột này

B3.1: Tạo biến Int ct = 0;

B3.2: cột j khác cột hiện tại

B3.2.1: Thiết lập ô cờ hiện tại của quân sỹ thành rỗng

B3.2.1: Tạo biến t chạy qua các hàng trên cột của quân tướng

Nếu vị trí hàng t , cột của quân tướng trống ct++;

B3.2.2: Nếu ct == 0 turn = false;

Thiết lập lại các thuộc tính trên bảng cờ

B4 if( turn) return 1;

Else return 0;

- public override int TuongAnToan(int i, int j)

B1: Tạo biến bool turn = true;

B2: nếu turn == true

B2.1: Tạo 2 biến để lưu hàng và cột hiện tại của quân mã;

B2.2: Tạo 1 quân cờ tạm;

B2.3: If(tại vị trí hàng i, cột j trống

Con cờ đang ở vị trí hàng i, cột j được lưu lại tại quân cờ tạm

B2.4: Giả sử con cờ đi được đi đến vị trí hàng i, cột j;

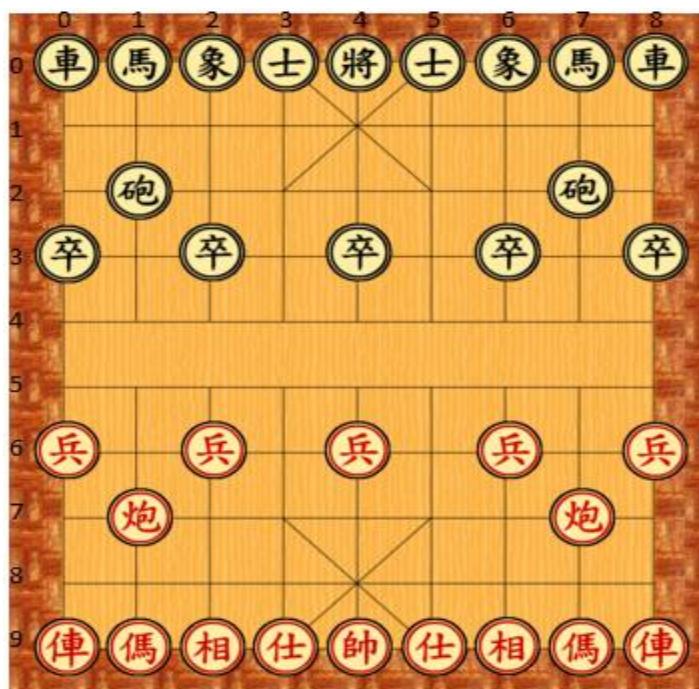
B2.5: If(tướng bị chiếu) turn = false;

Trả con cờ về vị trí cũ;

B3: nếu turn return 1;

Else return 0;

- Ví dụ minh họa:



(Hình ảnh minh họa vị trí quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm        | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|-------------------------------------|-----------------------|
| 9             | 3            | 8                  | 4                 | Thỏa các điều kiện nên turn = true; | Có thể di chuyển      |

|   |   |   |   |                                                                                                                   |                     |
|---|---|---|---|-------------------------------------------------------------------------------------------------------------------|---------------------|
| 9 | 3 | 8 | 2 | Không thỏa các điều kiện vì cột muốn di chuyển đến nằm ngoài vị trí cho phép di chuyển của quân Sy, turn = false; | Không thể di chuyển |
|---|---|---|---|-------------------------------------------------------------------------------------------------------------------|---------------------|

### g) Tướng

Kế thừa từ class Quân cờ gồm các hàm: KiemTra( int i, int j) và Hàm TuongAnToan(int i, int j), các input và ý nghĩa hàm tương tự lớp dân xuất trên.

- public override int KiemTra(int i, int j)

B1: tạo biến turn = false , ho,co;

B2: tạo 1 quân cờ tạm = temp;

B3: Khởi tạo quân cờ tạm

B4: Nếu ( $i \geq 0 \&& i \leq 2 \&& j \geq 3 \&& j \leq 5$ ) || ( $i \geq 7 \&& i \leq 9 \&& j \geq 3 \&& j \leq 5$ )

B4.1: Nếu  $i == \text{hang} + 1 \&& j == \text{Cot}$  Hoặc  $i == \text{hang} - 1 \&& j == \text{cot}$  || ( $i == \text{hang} \&& j == \text{Cot} + 1$ ) ||  $i == \text{hang} \&& j == \text{Cot} - 1$

B4.1.1 Nếu vị trí bàn cờ trống== true turn == true;

B4.1.2:Nếu vị trí bàn cờ trống== false

Nếu (phe!= Phe) turn = true;

B5:Phe ==0

B5.1: khởi tạo Tạo biến ct =0;

B5.2: if( j == Cột của phe==1){

B5.3: Nếu bàn cờ Vitri[i,j].Trong== true

for(int t= hang +1 t< Hàng của tướng phe 1

Nếu banco.Vitri[t,j].trong== false ) ct++

Nếu ct!=0 turn = false;

else

{

B5.4:temp = con cờ ứng với vị trí[i,j] trên bàn cờ

ho = hang của con tuong

co = cot của con tuong

Làm Ô cờ trống(hang,cot);

khởi tạo ô cờ sẽ đi đến với tên, phe , trống = false ;

hang = i ;

Cột = j

Temp.trangthai =0;

B5.5: for(int t= hang +1 t< Hàng của tướng phe 1){

if( banco.Vitri[t,j].trong== false ) ct++

if( ct!=0) ct++;

}

B5.6: Nếu( ct ==0 ) turn = false ;

trả về con tướng ở vị trí cũ

Khởi tạo Oco vs con cờ = tampt

B6: Nếu phe ==1

B6.1Tạo biến1 ct =0;

B6.2: Nếu  $j ==$  Cột của phe==0

B6.2.1 Nếu bàn cờ Vitri[i,j].Trong== true

B6.2.1.1: for(int t= hang -1 t< Hàng của tướng phe 1;t--

Nếu banco.Vitri[t,j].trong== false ct++

nếu ct!=0 turn = false

else

{

:

B6.3:temp = con cờ ứng với vị trí[i,j] trên bàn cờ

ho = hang của con tuong

co = cot của con tuong

Làm Ô cờ trống(hang,cot);

khởi tạo ô cờ sẽ đi đến với tên, phe , trống = false ;

hang = i ;

Cột = j

Temp.trangthai =0;

B6.4: for(int t= hang +1 t< Hàng của tướng phe 1){

if( banco.Vitri[t,j].trong== false ) ct++

if( ct!=0) ct++;

}

B6.5: Nếu ct ==0 turn = false ;

B6.6: trả về con tướng ở vị trí cũ.

B6.7: Khởi tạo Oco vs con cờ = tam

B7: Nếu turn == true return 1;

else return 0;

- public override int TuongAnToan(int i, int j)

B1: Khởi tạo biến turn = true ;

B2:int ho,int co; // Bien tam de luu tre hang cot

B3: Khởi tạo quân cờ

B4:ho= hang;

B5: co = Cot;

B6: Nếu vị trí bàn cờ không trống)

{

B6.1: biến temp = Quân cờ trong ô cờ vị trí [i,j];

B6.2: thiết lập trạng thái chết cho quân cờ

B6.3: Nếu VanCo.Chietuong[Phe].qTuong== true ) turn = else

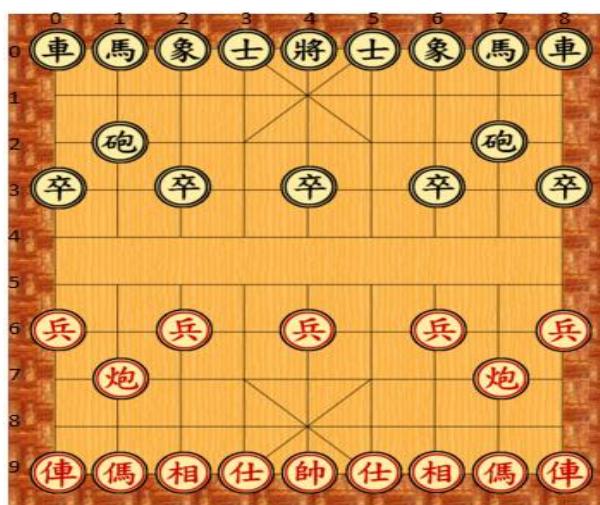
B6.4: gán giá trị ở vị trí [i,j] cho quân tướng

B6.5: Nếu con cờ temp là con cờ còn sống){

B6.6: Trả lại con cờ vào bàn cờ

B7: Nếu turn == true return 1 else return 0;

- Ví dụ minh họa:



(Hình biểu diễn vị trí quân cờ)

| Hàng hiện tại | Cột hiện tại | Hàng di chuyển đến | Cột di chuyển đến | Kiểm tra điều kiện trong hàm                                                    | Có thể di chuyển đến? |
|---------------|--------------|--------------------|-------------------|---------------------------------------------------------------------------------|-----------------------|
| 9             | 4            | 8                  | 4                 | Thỏa các điều kiện nên turn = true                                              | Có thể di chuyển      |
| 9             | 4            | 9                  | 3                 | Không thỏa điều kiện vì ở vị trí (9,3) không trống và cùng phe nên turn = false | Không thể di chuyển   |

### III. CÁC VẤN ĐỀ VÀ GIẢI THUẬT

#### 1) Ván cờ

Để quản lý được cả chương trình khi đang chạy, lớp Ván Cờ sẽ thực hiện việc quản lý này

##### a) Thuộc tính

- public static Player[]
- Player=new Player[2]

Tạo ra 2 người chơi: Player[0] và Player[1]

- public static string TenPlayer1

Tên Player[0]

- public static string TenPlayer2

Tên Player[1]

- public static bool DangChoi

Trạng thái của ván cờ: Ván cờ đang được chơi => DangChoi=True

- public static PictureBox ThongBaoChieuTuong = new PictureBox()

PictureBox chứa picture thông báo tướng đang bị chiếu

- public static Panel ChieuBi = new Panel()

Panel thông báo nước chiếu bí, bao gồm 2 picturebox con có nhiệm vụ như 2 button là DiLai(xin đi lại) và ChiuThua(chịu thua)

- public static Panel KetQua = new Panel()

Panel thông báo kết quả của ván cờ, bao gồm Label NguoiThang ghi tên của người thắng, 2 picturebox con có nhiệm vụ như 2 button là ChoiLai(Chơi lại) và Thoat(thoát)

- **public static Panel** ChapCo = **new Panel()**

Panel hiển thị thông báo chọn quân cờ muốn chấp (trong trường hợp người chơi chọn chấp cờ), gồm có 1 picturebox con có nhiệm vụ như 1 button là ChapXong(bắt đầu chơi khi người chơi chấp xong)

- **public static bool** Marked

Xác định đã có quân cờ nào được click chọn để đi chưa(quân cờ được click => Marked=True)

- **public static QuanCo** DanhDau

Quân cờ DanhDau tham chiếu đến quân cờ được chọn để đi

- **public static int** LuotDi

Kiểm tra đến lượt đi của Player[0] hay Player[1](lượt đi của Player[0] => LuotDi=0)

- **public static int** winner

Xác định người thắng ván cờ (Player[0] thắng => winner=0, Player[1] thắng => winner=1), nếu chưa có người nào thắng thì winner=2

- **public static NuocDi[]** GameLog

Mảng GameLog, nhật ký các nước đi, phục vụ cho chức năng Undo, với mỗi phần tử là một nước đi (struct NuocDi gồm có vị trí đầu, vị trí cuối, và các quân cờ tương ứng với vị trí đầu, cuối)

- **public static QuanBiAn[]** QuanDoBiAn

- **public static QuanBiAn[]** QuanDenBiAn

Mảng lưu các quân cờ đã bị ăn của 2 người chơi (struct QuanBiAn bao gồm hàng, cột, tên của quân bị ăn)

- **public static bool** Chap

Xác định có mở chức năng chấp cờ hay không(chấp => Chap=True)

- **public static bool** AmThanh

Trạng thái của tùy chọn tiếng động khi di chuyển quân cờ

- **public static bool NhacNen**

Bật, tắt nhạc nền (bật => NhacNen=true)

- **public static bool TinhThoiGian**

Thiết lập chức năng tính thời gian cho ván cờ (TinhThoiGian=True)

### b) Phương thức

- **static VanCo()**

Constructor khởi tạo 2 người chơi và các giá trị ban đầu

- **public static void NewGame()**

Tạo ván cờ mới, vẽ các quân cờ lên bàn cờ trong lần chơi đầu tiên, đưa các quân cờ về vị trí ban đầu trong những lần chơi sau...

- **public static void DoiLuotDi()**

Đổi lượt đi của người chơi bằng cách Khóa tất cả các quân cờ của phe vừa đi, mở Khóa tất cả các quân cờ của phe chuẩn bị đi

- **public static void OCoTrong(int i, int j)**

Trả về ô cờ trống cho vị trí [i,j] bằng cách reset các giá trị của struct Oco

- **public static void DatQuanCo(Object sender, QuanCo q, int i, int j)**

Đặt quân cờ q vào vị trí [i,j]

- **public static bool ChieuTuong(QuanCo tuong)**

Kiểm tra quân Tướng được đưa vào làm tham số có bị chiếu không bằng cách kiểm tra từng quân cờ của đối phương có di chuyển đến vị trí của quân tướng được không, trả về True nếu bị chiếu, trả về False nếu không bị chiếu

- **public static void KiemTraChieuTuong()**

Kiểm tra và thông báo quân tướng phe nào bị chiếu nếu là nước đi chiếu tướng

- **public static void AnQuanCo(QuanCo q)**

Thiết lập quân cờ q thành quân cờ đã bị ăn(TrangThai=0) và đưa vào mảng các quân cờ bị ăn

- **public static void KiemTraChieuBi()**

Kiểm tra và thông báo nếu là nước đi chiếu bí bằng cách duyệt lần lượt các quân cờ của phe bị chiếu xem có quân cờ nào còn nước đi hợp lệ không (đúng luật đi của loại quân đó, không chống tướng, nước đi không dâng tướng cho đối phương)

- **public static void ClickSound(string s)**

Phát ra tiếng động khi quân cờ di chuyển hoặc ăn quân đối phương

- **public static void PlayNhacNen(bool nhacnen)**

Kiểm tra giá trị của nhacnen, nhacnen=True => play nhạc nền, nhacnen=False => stop nhạc nền.

- **public static void Computer()**

Sau khi thực hiện cắt tỉa alpha-beta, và lựa chọn được nước đi tốt nhất.

## 2) Bài toán tổng quát

Từ khi bắt đầu trò chơi cho đến khi kết thúc chương trình đã trải qua các hàm để thực hiện:

### a) Hàm new game

Nhấn newgame và bắt đầu trò chơi

Cài đặt hàm new game:

B1: if(trạng thái là DangChoi == true)

B1.1: Xóa các quân cờ trên bàn cờ bằng cách bậc các picQuanCo.Visible = false của các quân cờ đó;

B1.2: Xóa 2 người chơi;

B1.3: Khởi tạo 2 người chơi mới;

B1.4: Reset lại bàn cờ mới bằng hàm ResetBanCo bên class BanCo;

B1.5: winner=2; // chưa có người chơi thắng

B1.6: LuotDi = 1; // Lượt đi của người chơi

B1.7: Count\_den = 0; // chưa có quân cờ đen nào bị ăn

B1.8: Count\_do = 0; // chưa có quân cờ đỏ nào bị ăn

B1.9: Các thông báo trong trò chơi chưa được phép bậc;

B1.10: Hình ảnh thông báo lượt đi của người chơi được bậc;

B1.11: Hình ảnh thông báo lượt đi của máy không được bậc;

B1.12: Đặt quân cờ của 2 người chơi;

B1.13: break;

B2: if(trạng thái là DangChoi == false)

B2.1: Khởi tạo 2 người chơi mới;

B2.2: DangChoi = true;

B2.3: Các thông báo trong trò chơi chưa được phép bậc;

B2.4: Hình ảnh thông báo lượt đi của người chơi được bậc;

B2.5: Hình ảnh thông báo lượt đi của máy không được bậc;

B2.6: Đặt quân cờ của 2 người chơi;

B2.7: break;

### b) Hàm ô cờ trống

Input: i là hàng, j là cột trên bàn cờ

Ví dụ: Hàm được gọi với tham số i,j Bàn cờ ở vị trí i,j sẽ dc làm trống.

a) Đưa ô cờ thành rỗng

trong = false ;

Ten= " ";

ThuTu="";

Phe = 2;

### c) Hàm đặt quân cờ

Input: Quân cờ được đặt, vị trí i là hàng trên bàn cờ, j là cột trên bàn cờ.

VD: q= Quân cờ Xe và i = 2, j= 1, quân cờ Xe sẽ được đặt ở vị trí (2,1) trên bàn cờ.

```
public static void DatQuanCo(QuanCo q, int i, int j)
```

```
{
```

```
//thiết lập các thuộc tính cho ô cờ của bàn cờ
```

```
BanCo.ViTri[i, j].Trong = false;
```

```
BanCo.ViTri[i, j].Phe = q.Phe;
```

```
BanCo.ViTri[i, j].Ten = q.Ten;
```

```
BanCo.ViTri[i, j].ThuTu = q.ThuTu;
```

```

q.Hang = i;
q.Cot = j;
q.picQuanCo.Top = i * 46 + 220;
q.picQuanCo.Left = j * 51 + 180;
}

```

#### **d) Hàm kiểm tra chiếu tướng**

B1: Khởi tạo

```
Int t = 0;
```

B2: Xác định tướng phe nào bị chiếu

```

if(tướng bị chiếu thuộc phe máy) t = 0;
if(tướng bị chiếu thuộc phe người) t = 1;
if(tướng không bị chiếu) t = 2;

```

B3: Hiển thị thông báo tướng bị chiếu và âm thanh

#### **e) Hàm kiểm tra chiếu bí**

Input: sử dụng biến toàn cục winner. Nếu không bị chiếu bị winner = 2.

VD Khi hàm được gọi nếu phe 1 Bị chiếu winner = 0 nếu phe 0 bị chiếu winner = 1

B1: Khởi tạo

```
bool cuu = false ;// tao 1 bien quan co co the cuu duoc
```

```
int tuong = 0 ,sy0 = 0, sy1=0.., khởi tao các biến nếu có thể cuu đc tướng
```

B2: if(Lượt đi == 0 )

B2.1: if( chiếu tướng Player[0] == true )

```
B2.1.1: for(int i = 0 i-> 9 ) {
```

```
    for(int j=0 j->8){
```

```
        if(Trang thai các quan cờ của player 0 vẫn còn sống ){
```

```
            if (Quan tướng đó có thể di chuyển đc tới vị trí [i,j]
```

và quan cờ có thể di chuyển đc đến vị trí [i,j]) tuong = 1;

```
}
```

```
        if (Player[0].qXe[0].TrangThai == 1)
```

```

        if (Player[0].qXe[0].KiemTra(i, j) == 1
&& Player[0].qXe[0].TuongAnToan(i, j) == 1) xe0 = 1;

B2.1.2: if(Tuong =1 || sy== 1 ||...) cuu = true;
        else cuu = true ;
        if(!cuu) winner=1;

B3: if (lượt đi ==1)
B3.1: if( chiếu tướng Player[0] == true )
    fB3.1.1: for(int i =0 i-> 9 ) {
        for(int j=0 j->8){
            if(Trang thai các quan cờ của player 0 vẫn còn sống ){
                if (Quan tướng đó có thể di chuyển được tới vị trí [i,j]
và quan cờ có thể di chuyển được đến vị trí [i,j]) tuong = 1;
            }
            if (Player[1].qXe[1].TrangThai == 1)
                if (Player[1].qXe[1].KiemTra(i, j) == 1
&& Player[1].qXe[1].TuongAnToan(i, j) == 1) xe0 = 1;

        B3.1.2: if(Tuong =1 || sy== 1 ||...) cuu = true;
        else cuu = true ;
        if(!cuu) winner=1;

f) Hàm kiểm tra tướng
Input: sử dụng biến toàn cục winner.
Ví dụ: không tìm thấy tướng phe 1 => winner =0 và ngược lại.
Function KiemTraTuong(){

B1: Khởi tạo
        Tạo biến tạm;

B2: Tìm tướng
B2.1: Tìm tướng ở phe máy
for(int i = 0; i < 3; i++)
{
    for(int j = 3; j <= 5; j++)

```

```

{
    if(BanCo.ViTri[i, j].Ten == "tuong")
    {
        Biến tạm được thay đổi;
        break;
    }
}

if (k == 0)
{
    Người chơi thắng;
    return;
}

```

### B2.2: Tìm tướng ở phe người

```

for(int i = 7; i <= 9; i++)
{
    for(int j = 3; j <= 5; j++)
    {
        if(BanCo.ViTri[i,j].Ten == "tuong")
        {
            Thay đổi biến tạm;
            break;
        }
    }

    if(k == 0)
    {
        Người chơi thua;
        return;
    }
}

```

}

### **g) Hàm đổi lượt đi**

Input: Biến toàn cục lượt đi

Output: Nếu Biến lượt đi trước là 1 thì đổi thành 0 và ngược lại.

VD Hàm đổi lượt đi được gọi. Biến lượt đi sẽ được đổi, đồng thời khóa các con cờ trên phe kia không được đi.

```
if(lượt đi ==1 ) Luotdi=0;
```

```
else Luotdi =1;
```

```
if(Ván cờ có lượt đi ==0)
```

```
{
```

Khóa các con cờ do người chơi Player[1] không thể di chuyển;

Và các Player[0] có thể di chuyển;

VD VanCo.Player[1].qSy[i].Khoa = true;

VanCo.Player[0].qMa[i].Khoa = false;

```
}
```

```
else {
```

Khóa các con cờ do người chơi Player[0] không thể di chuyển;

Và các Player[1] có thể di chuyển;

VD VanCo.Player[0].qSy[i].Khoa = true;

VanCo.Player[1].qSy[i].Khoa = false;

```
}
```

### **h) Hàm Computer**

Input: Sử dụng biến toàn cục bestMove

Output: gán quân cờ để đi tiếp theo vào bestMove. Gọi các hàm OCoTrong(), Datquanco(), amthanh(), KiemtraChieuTuong(), Doiluotdi(), KiemTrachieuBi()

Function Computer(){

B1: Khởi tạo

Tạo biến i ,j;

Khởi tạo quân cờ đánh dấu;

### B2: Cắt tỉa

Thực hiện cắt tỉa chọn nước đi tốt nhất;

B3: Gán quân cờ di chuyển tốt nhất vào quân cờ đánh dấu

Gán i = hàng của quân cờ di chuyển tốt nhất;

Giá j = cột của quân cờ di chuyển tốt nhất;

Quân cờ DanhDau = quân cờ di chuyển tốt nhất;

B4: Xác định vị trí sắp di chuyển có trạng thái ra sao để đi cho phù hợp

if(vị trí i, j trống){

B4.1: Bỏ chọn quân cờ;

B4.2: Thiết lập ô cờ trống tại hàng và cột của quân cờ đánh dấu;

B4.3: Đặt quân cờ đánh dấu vào vị trí i, j;

B4.4: Thiết lập âm thanh;

B4.5: Kiểm tra chiếu tướng;

B4.6: Kiểm tra chiếu bí;

B4.7: Kiểm tra tướng;

B4.8: If(tướng không chạy được nữa hoặc bị ăn){

Khóa bàn cờ;

if(bị chiếu bí){

Bật thông báo chiếu bí;

Hiện kết quả;

Dừng lại;

}

Else{

Hiện kết quả;

Dừng lại;

}

}

B4.9: Đổi lượt đi;

B4.10: Rest lại CanMove;

Dừng lại;

}

Else{

B4.1: Tạo biến phekia;

B4.2: if(DanhDau.Phe == 0) phekia = 1;

Else phekia = 0;

B4.3: Tạo quân cờ tạm;

B4.4: Gán quân cờ tạm = quân cờ đang có tại vị trí i ,j;

B4.5: Bỏ chọn quân cờ;

B4.6: Thiết lập ô cờ trống tại hàng và cột của quân cờ đánh dấu;

B4.7: Đặt quân cờ đánh dấu vào vị trí i, j;

B4.8: Thiết lập âm thanh;

B4.9: Kiểm tra chiêu tướng;

B4.10: Kiểm tra chiêu bí;

B4.11: Kiểm tra tướng;

B4.12: If(tướng không chạy được nữa hoặc bị ăn){

Khóa bàn cờ;

if(bị chiêu bí){

Bật thông báo chiêu bí;

Hiện kết quả;

Dừng lại;

}

Else{

Hiện kết quả;

Dừng lại;

}

}

B4.13: Đổi lượt đi;

B4.14: Rest lại CanMove;

Dừng lại; }

## 4) Các vấn đề

### a) Vấn đề 1: Phát sinh các nước đi hợp lệ

Phát biểu bài toán: Một trong những việc quan trọng nhất để máy tính có thể chơi được cờ là phải chỉ cho nó biết mọi nước đi có thể đi được từ một thế cờ. Máy sẽ tính toán để chọn nước đi có lợi nhất cho nó. Các yêu cầu chính đối với một thủ tục sinh nước đi là:

- Chính xác (rất quan trọng). Một nước đi sai sẽ làm hỏng mọi tính toán.
- Đầy đủ (rất quan trọng). Sinh được mọi nước đi có thể có từ một thế cờ.
- Nhanh. Do chức năng này phải sinh được hàng triệu nước đi mỗi khi máy đến lượt nên giảm thời gian sinh nước đi có ý nghĩa rất lớn.

Vậy làm cách nào để đáp ứng đầy đủ cả 3 yêu cầu trên ?

Ý tưởng giải quyết: Sinh nước đi là một thuật toán vét cạn. Máy sẽ tìm mọi nước đi hợp lệ có thể có. Máy phải sinh được từ những nước đi rất hay cho đến những nước đi rất ngớ ngẩn (như đẩy Tướng vào vị trí khống chế của đối phương). Đây là một trong những thủ tục phức tạp và dễ sai nhất.

Thuật giải:

Input: thế cờ ban đầu

Output: danh sách các thế cờ mới.

Mã giả

Bước 1: int dem = 0; //đếm tất cả các thế cờ mới

QuanCo[ ] q = new QuanCo[40]; //lưu các quân cờ trên bàn cờ

Bước 2:

Bước 2.1. Đến lượt bên nào đi thì đếm tất cả các quân cờ có trên bàn cờ của bên đó.

//biến TrangThai kiểm tra quân cờ có trên bàn cờ hay không

// biến Khoa kiểm tra xem lượt bên nào thì chỉ đếm quân cờ bên đó

```

//do việc đếm các quân cờ được lặp đi lặp lại nên em chỉ liệt kê một vài
quân cờ.

if (Player[1].qChot[0].TrangThai == 1 && Player[1].qChot[0].Khoa ==
false)
{
    q[dem] = Player[1].qChot[0];
    dem++;
}

if (Player[1].qMa[1].TrangThai == 1 && Player[1].qMa[1].Khoa == false)
{
    q[dem] = Player[1].qMa[1];
    dem++;
}

.....
if (Player[0].qChot[0].TrangThai == 1 && Player[0].qChot[0].Khoa ==
false)
{
    q[dem] = Player[0].qChot[0];
    dem++;
}

if (Player[0].qXe[0].TrangThai == 1 && Player[0].qXe[0].Khoa == false)
{
    q[dem] = Player[0].qXe[0];
    dem++;
}

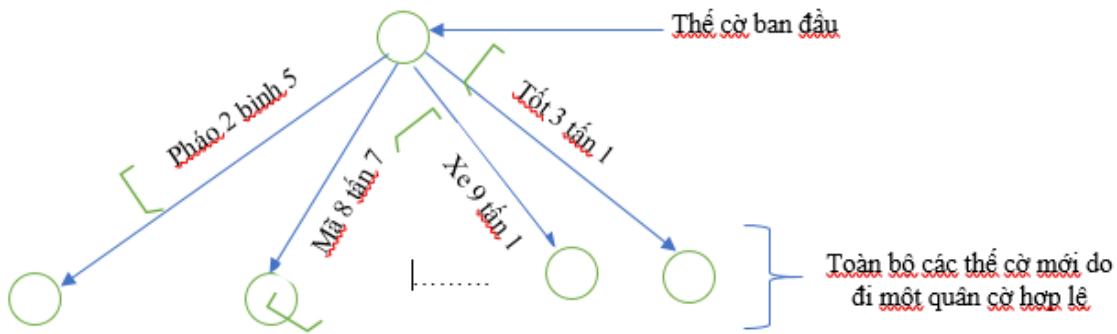
.....

```

Bước 2.2. Sau khi đếm các quân cờ trên bàn cờ, ta sinh tất cả nước đi hợp lệ của từng quân cờ đó

```
int ndd = 0;  
NuocDi [ ] nd = new NuocDi[200];  
  
for (int k = 0; k <= dem; k++)  
{  
    for (int i = 0; i <= 9; i++)  
    {  
        for (int j = 0; j <= 8; j++)  
        {  
            if (q[k].KiemTra(i, j) == 1 && q[k].TuongAnToan(i, j) == 1)  
            {  
                nd[ndd] = new NuocDi();  
                nd[ndd].Q = q[k];  
                nd[ndd].x = i;  
                nd[ndd].y = j;  
                ndd++;  
            }  
        }  
    }  
}  
temp = --ndd;// so nuoc di sinh ra
```

Bước 3: return nd;



### b) Vấn đề 2: Lượng giá độ tốt của một trạng thái (Đánh giá độ tốt của một thế cờ)

Phát biểu bài toán: Đánh giá một thế cờ là một trong những nhiệm vụ quyết định chương trình chơi cờ của bạn có là "cao thủ" hay không. Căn cứ vào một thế cờ máy sẽ gán cho nó một điểm số (lượng giá tĩnh) để đánh giá độ tốt - xấu. Nhờ điểm này máy mới có thể so sánh các thế cờ với nhau và biết chọn nước đi tốt nhất. Đây là một nhiệm vụ rất khó khăn và phức tạp do không tồn tại một thuật toán tổng quát và thống nhất nào để tính điểm cả. Điểm của một thế cờ dựa trên rất nhiều yếu tố mà khó có thể số hoá hết được như phụ thuộc vào số lượng và giá trị các quân cờ hiện tại, phụ thuộc vào tính hầm, tính biến, thế công, thế thủ của từng quân cờ cũng như cả cục diện trận đấu. Vậy làm cách nào để số hóa tất cả các yếu tố trên ?

Ý tưởng giải quyết: Số hoá độ tốt của một thế cờ qua hàm lượng giá quả là một điều hết sức khó khăn. Tuy nhiên ta cũng có thể giản lược bớt một số tiêu chí nhỏ trong đó nhưng vẫn đảm bảo độ tốt - xấu của một thế cờ. Chúng ta bắt đầu với việc công nhận các giả thuyết sau:

- Ta có thể biểu diễn chất lượng một thế cờ bằng một con số. Ví dụ, con số đó có thể là đánh giá của ta về xác suất chiến thắng, nhưng đối với đa số chương

trình thì con số đó không có gì đặc biệt, nó chỉ là con số mà mục đích chính là so sánh được với nhau.

- Chúng ta nên đo chất lượng của một thế cờ tương tự như phép đo của đối phương (do đó, nếu ta coi là đạt được một thế tốt thì đối phương của ta phải thấy đó là thế xấu đối với họ và ngược lại). Điều này tuy không thật đúng lầm, nhưng nó giúp cho thuật toán tìm kiếm của ta làm việc tốt và trong thực tế cũng khá gần với sự thật.

Thuật giải: Chúng ta chỉ cài đặt phương pháp đơn giản nhưng cơ bản nhất: lượng giá dựa trên cơ sở giá trị của từng quân cờ và độ tốt nước đi của quân cờ đó. Cách tính này sẽ lấy tổng giá trị các quân cờ hiện có của bên mình trừ đi tổng giá trị các quân cờ hiện có của đối phương. Do đó, một thế cờ này hơn thế cờ kia ở chỗ nó còn nhiều quân bên mình hơn, nhiều quân giá trị cao hơn cũng như có bắt được nhiều quân và quân giá trị cao của đối phương hơn không.

Điểm các quân cờ được đánh giá theo kinh nghiệm và cho biết sự tương quan giữa các quân cờ. Sau đây là điểm từng quân mà mọi người thường chấp nhận:

| Tên quân cờ  | Kí hiệu | Độ tốt |
|--------------|---------|--------|
| Tướng        | King    | 10000  |
| Sỹ           | Advisor | 40     |
| Tịnh (tượng) | Bishop  | 40     |
| Xe           | Rook    | 180    |
| Pháo         | Cannon  | 90     |
| Mã           | Knight  | 80     |
| Chốt         | Pawn    | 30     |

Và ứng với mỗi quân cờ có độ tốt nước đi riêng. Do có nhiều quân cờ nên em chỉ liệt kê độ tốt một vài quân cờ.

```
public static int[,] BlackKnightPosition = new int[,]{
    { 70, 80, 90, 80, 70, 80, 90, 80, 70 },
    { 80, 110, 125, 90, 70, 90, 125, 110, 80 },
    { 90, 100, 120, 125, 120, 125, 120, 100, 90 },
    { 90, 100, 120, 130, 110, 130, 120, 100, 90 },
    { 90, 110, 110, 120, 100, 120, 110, 110, 90 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 70, 75, 75, 70, 50, 70, 75, 75, 70 },
    { 60, 70, 75, 70, 60, 70, 75, 70, 60 }
};
```

```
public static int[,] RedKnightPosition = new int[,]{
    { 60, 70, 75, 70, 60, 70, 75, 70, 60 },
    { 70, 75, 75, 70, 50, 70, 75, 75, 70 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 80, 80, 90, 90, 80, 90, 90, 80, 80 },
    { 90, 110, 110, 120, 100, 120, 110, 110, 90 },
    { 90, 100, 120, 130, 110, 130, 120, 100, 90 },
    { 90, 100, 120, 125, 120, 125, 120, 100, 90 },
    { 80, 110, 125, 90, 70, 90, 125, 110, 80 },
```

```
{ 70, 80, 90, 80, 70, 80, 90, 80, 70},  
};
```

Input: Các quân cờ hiện có trên bàn cờ.

Output: Độ tốt của thế cờ.

Mã giả:

Bước 1: int s=0; //biến cho ra kết quả cuối cùng của 1 thế cờ. Nếu s > 0  
thì

//bên mình chiếm ưu, ngược lại máy tính chiếm ưu.

Bước 2: tiến hành đánh giá thế cờ

//Do phải cộng các giá trị từng quân cờ nên nếu viết đầy đủ thì sẽ khá dài  
nên em chỉ viết vài dòng code tượng trưng.

```
for (int i = 0; i < 5; i++)  
    if (Player[1].qChot[i].TrangThai == 1)  
    {  
        s -= (2*30 + RedPawnPosition[Player[1].qChot[i].Hang,  
Player[1].qChot[i].Cot]);  
    }
```

```
for (int i = 0; i < 5; i++)  
    if (Player[0].qChot[i].TrangThai == 1)  
    {  
        s += (2*30 + BlackPawnPosition[Player[0].qChot[i].Hang,  
Player[0].qChot[i].Cot]);  
    }
```

```
for (int i = 0; i < 2; i++)  
    if (Player[0].qPhao[i].TrangThai == 1)  
    {
```

```

s += (2*90 + BlackCannonPosition[Player[0].qPhao[i].Hang,
Player[0].qPhao[i].Cot]);
}

```

```

for (int i = 0; i < 2; i++)
{
    if (Player[1].qPhao[i].TrangThai == 1)
    {
        s -= (2*90 + RedCannonPosition[Player[1].qPhao[i].Hang,
Player[1].qPhao[i].Cot]);
    }
}

```

Bước 3: return s;

### c) Vấn đề 3: Xử lý một nước đi “thứ”

Phát biểu bài toán: Trong quá trình tính toán tìm nước đi tốt nhất cho mình, máy thường phải đi “thứ” một nước rồi lại tính tiếp. Sau đó nó sẽ phục hồi lại nước đi này. Để làm sao cho máy tính có thể hiểu được những gì mình muốn làm ?

Ý tưởng giải quyết: Do quá trình thử - phục hồi diễn ra rất nhanh và không hiện ra màn hình nên máy có thể thử hàng chục triệu lần mỗi khi đến lượt. Việc đi “thứ” được thực hiện bằng cách lưu vị trí hiện tại, sau đó gọi hàm OCoTrong để tạo ô trống ở vị trí hiện tại và gọi hàm DatThuQuanCo để đặt “thứ” quân cờ vào vị trí mới. Việc khôi phục nước đi này được thực hiện bằng cách gọi hàm OCoTrong để tạo ô trống ở vị trí đặt thử, sau đó gọi hàm DatThuQuanCo để đặt quân cờ vào vị trí hiện tại. Quá trình này được thực hiện bên trong hàm Alpha\_Beta\_pruning

### d) Vấn đề 4: Lựa chọn nước đi tốt nhất (Chiến lược cắt tỉa Alpha-Beta)

Phát biểu bài toán: Giải thuật cắt tỉa Alpha-beta cực kỳ quan trọng khi lập trình các trò chơi như cờ vua hay cờ tướng, khi các khung gian trạng thái của những trò chơi này có độ phức tạp cao. Cắt tỉa Alpha-beta sẽ giúp loại bỏ những khung gian

trạng thái không cần thiết và hỗ trợ tối ưu hóa thuật toán tìm kiếm Minimax. Vậy giải thuật này có những ưu điểm nào ?

Ý tưởng giải quyết: Thay vì tìm kiếm toàn bộ không gian đến một độ sâu lớp cố định, tìm kiếm Alpha-Beta thực hiện theo kiểu tìm kiếm sâu. Có hai giá trị, gọi là alpha và beta được tạo ra trong quá trình tìm kiếm, trong đó:

- Giá trị alpha liên quan với các nút Max và có khuynh hướng không bao giờ giảm.
  - Giá trị beta liên quan đến các nút Min và có khuynh hướng không bao giờ tăng.
- Để bắt đầu thuật toán tìm kiếm Alpha-Beta, ta đi xuống hết độ sâu lớp theo kiểu tìm kiếm sâu, đồng thời áp dụng đánh giá heuristic (đánh giá độ tốt của thế cờ) cho một trạng thái và tất cả các trạng thái anh em của nó. Giả thuyết tất cả đều là nút Max. Giá trị tối đa của các nút Max này sẽ được truyền ngược lên cho nút cha mẹ (là một nút Min). Sau đó giá trị này được gán cho ông bà của nút Max như là một giá trị beta kết thúc tốt nhất. Tiếp theo thuật toán này sẽ đi xuống các nút cháu khác và kết thúc việc tìm kiếm đối với nút cha mẹ của chúng nếu gặp bất kỳ một giá trị nào lớn hơn hoặc bằng giá trị beta này. Quá trình này gọi là cắt tỉa Beta.

Thuật giải:

Input: một nước đi do người dùng tạo nên (nôm na là phiên người chơi)

Output: một nước đi được lựa chọn qua thuật giải minimax với độ sâu giới hạn (nôm na đến phiên máy chơi)

Bước 1: Khởi tạo các giá trị

```
QuanCo temp_c = new QuanCo(); //lưu quân cờ bị ăn
int phekia; //dùng để phân biệt phe này với phe kia
```

```
NuocDi move = new NuocDi();
int temp = 0; //biến này đếm tất cả nước đi
NuocDi[] nd = new NuocDi[100]; //khởi tạo mảng những nước đi
```

```
nd = TimNuocDi(ref temp);
```

int best = -1000000; //độ tốt của thέ cờ

Bước 2:

```
//thực hiện việc cắt tỉa
//trả về điểm ưu thế. Nếu dương thì người chiếm ưu, ngược lại máy chiếm ưu
if (depth == 0)
    return Heuristic();
int k = 0;           //biến này dùng để duyệt các nước đi đã sinh ra
while (k <= temp && best < beta)
{
    if (best > alpha)
        alpha = best;//biến alpha này sẽ được gán vào beta để thực hiện việc cắt
        tỉa
    if (nd[k].Q.Phe == 0)
        phekia = 1;
    else
        phekia = 0;
    //Tạo ô cờ trống ở vị trí hiện tại
    OCoTrong(nd[k].Q.Hang, nd[k].Q.Cot);
    if (BanCo.ViTri[nd[k].x, nd[k].y].Trong)      //nếu vị trí mới trống
    {
        //lưu vị trí ban đầu của quân cờ
        int i = nd[k].Q.Hang;
        int j = nd[k].Q.Cot;
        DatThuQuanCo(nd[k].Q, nd[k].x, nd[k].y); //đặt thử quân cờ tại vị trí
        mới
        //lưu lại nước đi cũ sau khi đặt thử quân cờ vào vị trí mới
        nd[k].x = i;
        nd[k].y = j;
```

```

}

Else//nếu vị trí mới có quân cờ của đối phương
{
    QuanCoBiAn(nd[k].x, nd[k].y, phekia, ref nd[k].Bian);//lưu quân cờ bị
    ăn vào nước đi này
    //lưu vị trí ban đầu của quân cờ
    int i = nd[k].Q.Hang;
    int j = nd[k].Q.Cot;
    DatThuQuanCo(nd[k].Q, nd[k].x, nd[k].y);//đặt thử quân cờ vào vị trí
    mới
    //lưu lại nước đi cũ sau khi đặt thử quân cờ vào vị trí mới
    nd[k].x = i;
    nd[k].y = j;
    nd[k].Bian.TrangThai = 0;//giả định quân cờ bị ăn
}

DoiThuLuotDi();
//nếu giá trị Alpha_Beta_pruning dương, tức người chiếm ưu thế thì máy
sẽ đổi thành giá trị âm
//để máy hạn chế người chiếm ưu thế và ngược lại

int value = -Alpha_Beta_pruning(-beta, -alpha, depth - 1);
    //trả về vị trí ban đầu
if (nd[k].Bian.Phe != 2)
{
    int i = nd[k].Q.Hang;
    int j = nd[k].Q.Cot;
    OCoTrong(nd[k].Q.Hang, nd[k].Q.Cot);
    DatThuQuanCo(nd[k].Q, nd[k].x, nd[k].y);
    DatThuQuanCo(nd[k].Bian, nd[k].Bian.Hang, nd[k].Bian.Cot);
    nd[k].Bian.TrangThai = 1;
}

```

```

nd[k].x = i;
nd[k].y = j;
}
else
{
    int i = nd[k].Q.Hang;
    int j = nd[k].Q.Cot;
    OCoTrong(nd[k].Q.Hang, nd[k].Q.Cot);
    DatThuQuanCo(nd[k].Q, nd[k].x, nd[k].y);
    nd[k].x = i;
    nd[k].y = j;
}
DoiThuLuotDi();

if (value > best)
{
    best = value;
    move = nd[k];
}

k++;
}

if (depth == MaxDepth); //sau khi đã duyệt hết các nước đi thì gán nước đi
tốt nhất cho bestmove
{
    bestmove = move;
}

Bước 3: return best;

```

### 3.3.6. Vấn đề 5: Máy tính chọn nước đi

Phát biểu bài toán: Làm cách nào để máy tính có thể chọn nước đi phù hợp ?

Ý tưởng giải quyết: Gọi hàm Alpha\_Beta\_pruning để tìm ra nước đi phù hợp, sau đó vẽ nước đi này trên bàn cờ

Thuật giải:

Input: Các nước đi của máy

Output: Nước đi phù hợp cho máy

Mã giả:

Bước 1: int i, j; //2 biến này dùng để lưu vị trí nước đi tốt nhất để vẽ lên bàn cờ

```
DanhDau = new QuanCo(); //biến này dùng để đánh dấu quân cờ này sẽ
được          //chọn để đi (nôm na là nước đi phù hợp cho máy)
```

```
Alpha_Beta_pruning(-500000, 500000, MaxDepth);
```

```
i = bestmove.x;
```

```
j = bestmove.y;
```

```
DanhDau = bestmove.Q;
```

Bước 2: Nếu vị trí mới trống thì:

```
//Bỏ chọn quân cờ
```

```
Marked = false;
```

```
//Tạo ô cờ trống tại ví trí ban đầu
```

```
OCoTrong(DanhDau.Hang, DanhDau.Cot);
```

```
//Đặt quân cờ đã chọn vào vị trí mới [i,j]
```

```
DatQuanCo(DanhDau, i, j);
```

```
//Tiếng động
```

```
if (AmThanh) ClickSound("0");
```

```

//Kiểm tra chiếu tướng
KiemTraChieuTuong();

//Thay đổi lượt đi
DoiLuotDi();

//Kiểm tra chiếu bí
KiemTraChieuBi();
if (winner != 2)
{
    ThongBaoChieuTuong.Visible = false;
    ChieuBi.Visible = true;
}
else ChieuBi.Visible = false;

BanCo.ResetCanMove();

```

Nếu không trống thì:

```

int phekia = 2;
if (DanhDau.Phe == 0) phekia = 1;
else phekia = 0;
QuanCo temp;
temp = new QuanCo();

if (BanCo.ViTri[i, j].Ten == "tuong") temp = Player[phekia].qTuong;
if (BanCo.ViTri[i, j].Ten == "sy")
{
    if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qSy[0];
    if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phekia].qSy[1];
}

```

```

    }

    if (BanCo.ViTri[i, j].Ten == "tinh")
    {
        if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qTinh[0];
        if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phekia].qTinh[1];
    }

    if (BanCo.ViTri[i, j].Ten == "xe")
    {
        if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qXe[0];
        if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phekia].qXe[1];
    }

    if (BanCo.ViTri[i, j].Ten == "phao")
    {
        if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qPhao[0];
        if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phekia].qPhao[1];
    }

    if (BanCo.ViTri[i, j].Ten == "ma")
    {
        if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qMa[0];
        if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phekia].qMa[1];
    }

    if (BanCo.ViTri[i, j].Ten == "chot")
    {
        if (BanCo.ViTri[i, j].ThuTu == "0") temp = Player[phekia].qChot[0];
    }
}

```

```

        if (BanCo.ViTri[i, j].ThuTu == "1") temp = Player[phe-
kia].qChot[1];
        if (BanCo.ViTri[i, j].ThuTu == "2") temp = Player[phe-
kia].qChot[2];
        if (BanCo.ViTri[i, j].ThuTu == "3") temp = Player[phe-
kia].qChot[3];
        if (BanCo.ViTri[i, j].ThuTu == "4") temp = Player[phe-
kia].qChot[4];
    }

    //Bỏ chọn quân cờ
    Marked = false;

    //Ăn quân cờ của đối phương
    AnQuanCo(temp);

    //Trả lại ô cờ trống
    OCoTrong(DanhDau.Hang, DanhDau.Cot);

    //Thiết lập quân cờ đã chọn vào bàn cờ
    DatQuanCo(DanhDau, i, j);
    //Tiếng động
    if (AmThanh) ClickSound(DanhDau.Ten);

    //Kiểm tra chiếu tướng
    KiemTraChieuTuong();

    //Thay đổi lượt đi
    DoiLuotDi();

```

```
//Kiểm tra chiêu bí
KiemTraChieuBi();
if (winner != 2)
{
    ThongBaoChieuTuong.Visible = false;
    ChieuBi.Visible = true;
}
else ChieuBi.Visible = false;
```

**କେବଳ**

## Chương 4

# CÀI ĐẶT ỦNG DỤNG

### I. MÔ TẢ CHƯƠNG TRÌNH

#### 1) Giao diện tương tác

##### a) Hình ảnh

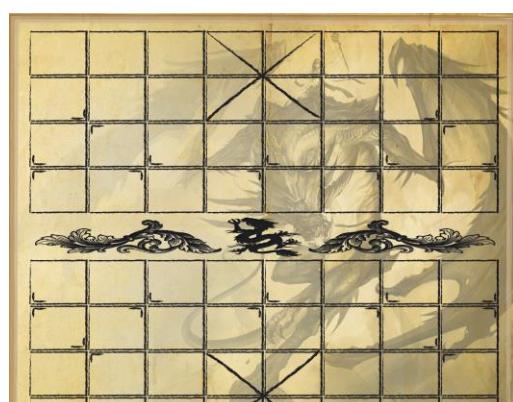
Hình ảnh trò chơi được thiết kế theo bối cảnh các cuộc chiến tranh cổ trang Trung Hoa, các hình ảnh chủ yếu được tham khảo từ google, sau đó được thiết kế lại, tạo thành hình ảnh riêng cho trò chơi của nhóm FANTASTIC thực hiện.

Các hình ảnh bao gồm:

- Hình flash screen



- Hình bàn cờ



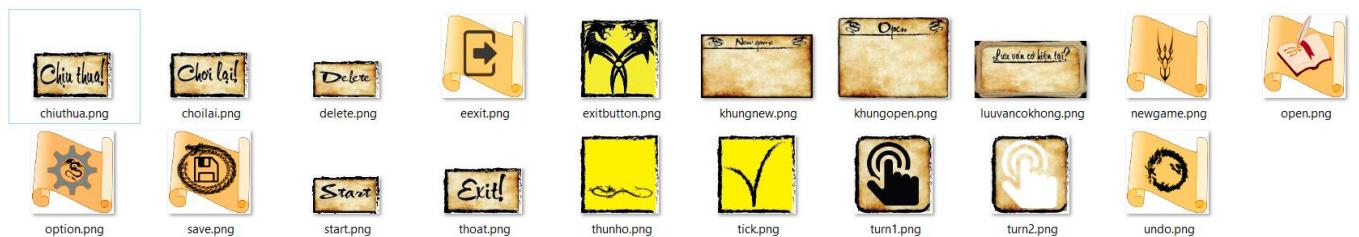
- Hình quân cờ



- Hình chessboard



- Hình các nút cần thiết (new game, thoát, thu nhỏ, ...)



- Hình thông báo (chiếu tướng, chiếu bí, ...)



### b) Chessboard

Form chính, dùng cho 2 người chơi tương tác với game như các menu để thực hiện các chức năng NewGame, Exit, thu nhỏ, hiển thị tên người chơi, thời gian còn lại của từng người chơi, các quân cờ bị ăn...



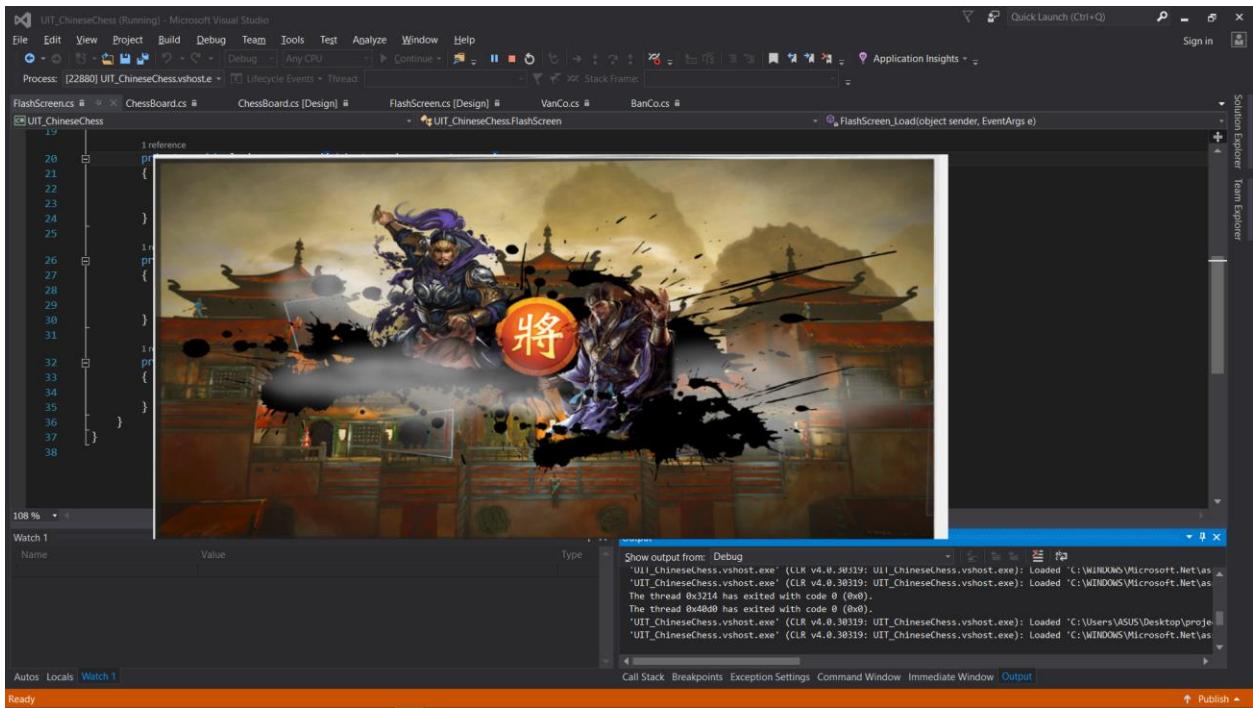
### c) Newgame

Form lấy thông tin cho một ván cờ mới, gồm có tên 2 người chơi, tùy chọn tính thời gian, chấp cờ. Khi form được submit thì các giá trị tương ứng với thông tin sẽ truyền cho các thuộc tính static tương ứng trong lớp VanCo để quản lý ván cờ đó.



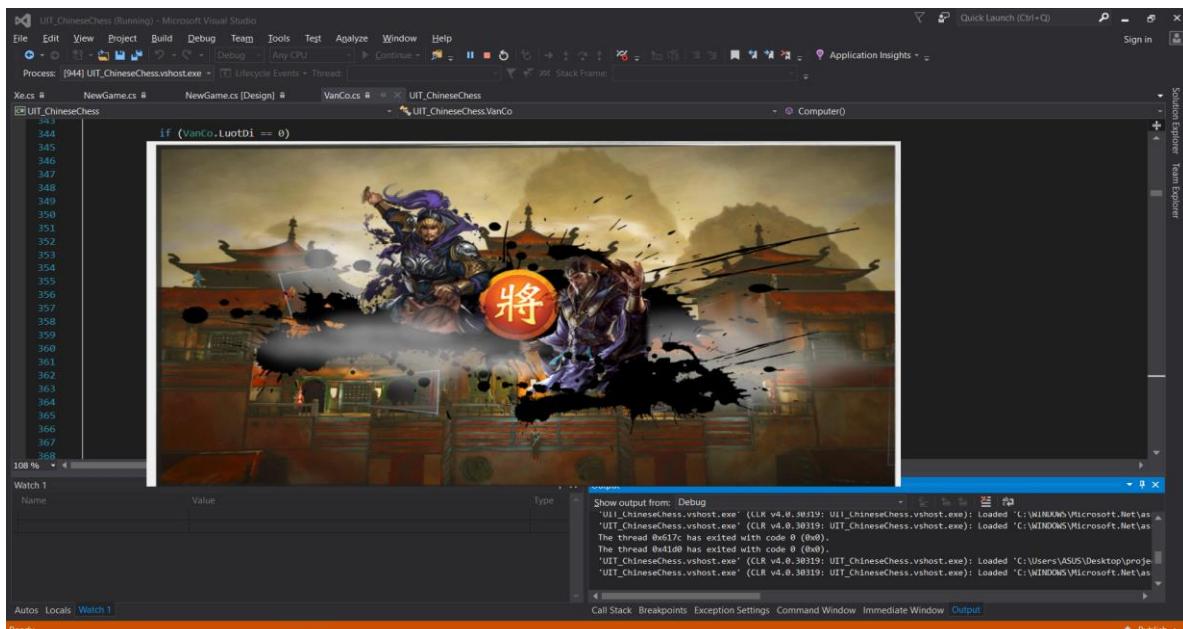
#### d) Flash screen

Khi khởi động chương trình, sẽ xuất hiện hình ảnh chờ trước khi vào trò chơi.



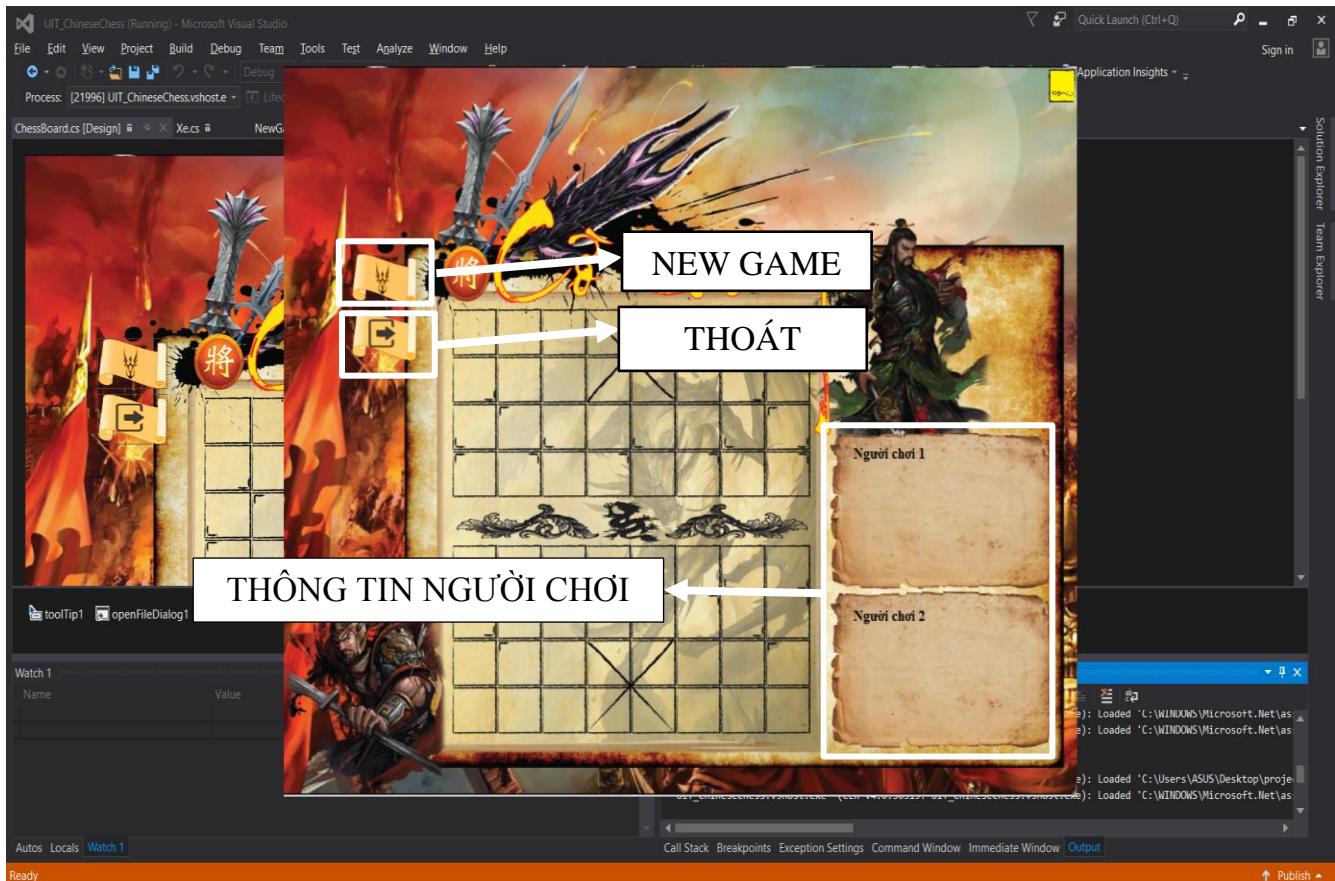
#### 2) Hướng dẫn sử dụng

- Khởi động: khi khởi động ứng dụng sẽ có màn hình xuất hiện, đồng thời nhạc sẽ được bật lên, thông báo trò chơi bắt đầu.



(Hình ảnh khi khởi động trò chơi)

- Khi khởi động xong, trò chơi bắt đầu với giao diện bao gồm: New game, thoát, bàn cờ, khung chứa thông tin người chơi 1 và 2.



*(Giao diện trò chơi khi mới bắt đầu)*

- Để bắt đầu trò chơi: người chơi cần click nút new game để khởi tạo thông tin.  
Sau khi khởi tạo thông tin, người chơi click nút “start” để bắt đầu trò chơi.

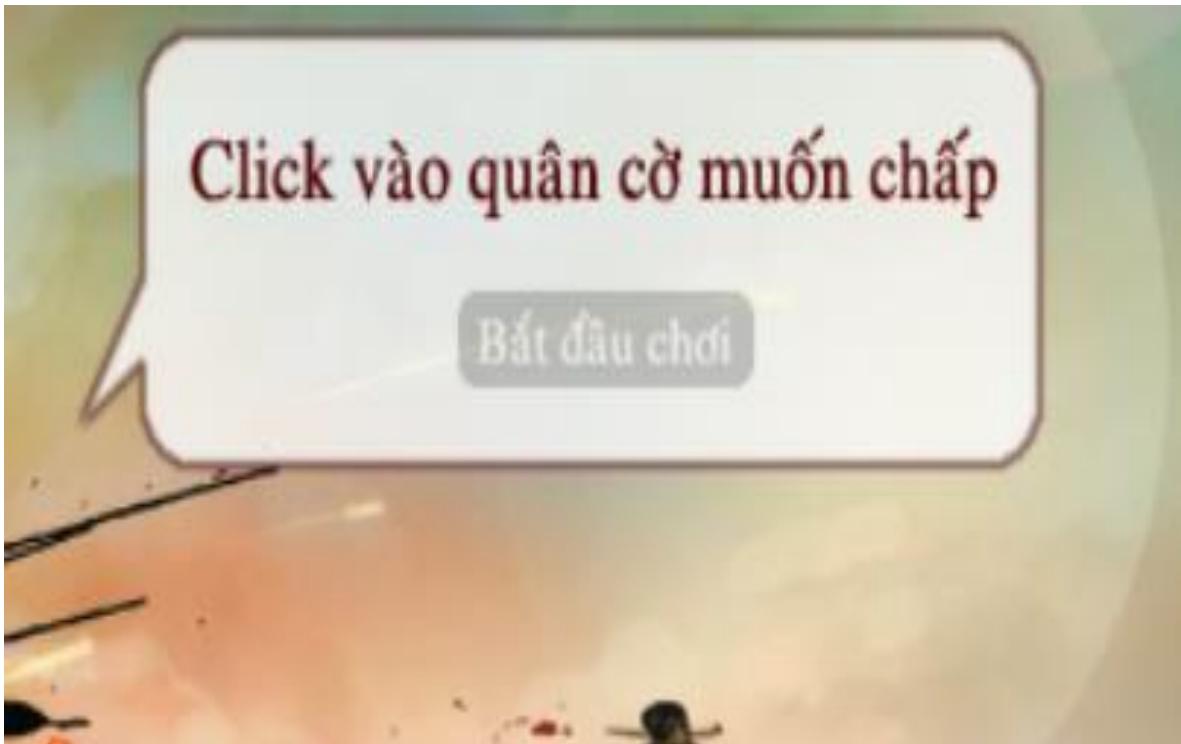


(Khởi tạo trò chơi)



(Lựa chọn chấp cờ)

- Khi hoàn thành bước lựa chọn chấp cờ, trò chơi sẽ khởi động với thông báo: “Lựa chọn quân cờ muốn chấp?”, người chơi lựa chọn quân muốn chấp, sau đó ấn nút “bắt đầu” để bắt đầu chơi.



(Thông báo lựa chọn quân cờ muốn chép)



(click chọn quân cờ muốn chép)

- Sau khi chép cờ, ta tiếp tục với ván cờ đang diễn ra

### 3) Chạy thử chương trình

#### a) Khởi động và khởi tạo thông tin

- Khởi động trò chơi với trường hợp không chấp cờ.
- Với quân đỏ là Máy Tính (MayTinh) và quân đen là người chơi (NguoiChoi).
- Có tính thời gian là 15 phút.

#### b) Chơi thử

- Theo luật chơi, bất kỳ nước đi nào làm cho Tướng của đối phương bị ăn ở nước tiếp theo thì sẽ có thông báo “Chiếu Tướng” (Hình 7.8).
- Nếu một bên chiếu (bắt Tướng) và đối phương không còn khả năng đỡ. Bên chiếu tướng thắng, đây là trường hợp “Chiếu Bí” (Hình 7.9).
- Lưu ý: Nếu người chơi muốn tạo lại một ván cờ mới, thực hiện lại thao tác nút new game.



(Khởi động và khởi tạo thông tin)



(Hình 7.8 – Chiếu tướng)



(Hình 7.9 – Chiếu Bí)

#### 4) Nhận xét

##### a) Giao diện

Giao diện đẹp mắt, đưa người chơi vào bối cảnh chiến tranh cổ trang Trung Hoa.

##### b) Âm thanh

Âm thanh êm tai, nhẹ nhàng, giúp người chơi tập trung. Đặc biệt, sinh động đỗi với từng quân cờ như tiếng ngựa hý khi quân Mã chiếm được quân địch, tiếng đi quân cờ...

##### c) Trí tuệ nhân tạo

- Ưu điểm:

- Có chức năng chấp cờ, giúp trò chơi thêm hấp dẫn.
- Quân cờ máy tính thực hiện nước đi nhanh do áp dụng lý thuyết cắt tỉa alpha-beta.
- Các nước đi thông minh khi áp dụng hàm heuristic để tính toán độ tốt.
- Quân cờ áp dụng đúng luật chơi, và luôn cố gắng đưa người chơi vào tình thế thua bại.

- Khuyết điểm:

- Thực hiện nước đi của người chơi còn chậm.

Chưa có nhiều chức năng hỗ trợ người chơi trong trò chơi.

##### d) Thời gian

Với độ sâu là 2, khi người chơi lựa chọn nước đi, chương trình phải mất 4s đến 5s để xử lý và tiếp tục ván cờ.

Nếu tăng độ sâu thời gian xử lý sẽ tăng.

## II. CÀI ĐẶT VÀ MỘT SỐ HÀM XỬ LÝ

### 1) Cài đặt

- Ngôn ngữ sử dụng: ngôn ngữ C#
- Công cụ thực hiện: Visual Studio 2015

### 2) Một số hàm xử lý chính

##### a) Hàm tìm nước đi

1. `public static NuocDi[] TimNuocDi(ref int temp) //Hàm Tìm Tất Cả Các nuoc di`
2. {

```
3.         int dem = 0;
4.         QuanCo[] q = new QuanCo[40];
5.         if (Player[1].qChot[0].TrangThai == 1 &&
Player[1].qChot[0].Khoa == false)
6.             {
7.                 q[dem] = Player[1].qChot[0];
8.                 dem++;
9.             }
10.            if (Player[1].qChot[1].TrangThai == 1 &&
Player[1].qChot[1].Khoa == false)
11.                {
12.                    q[dem] = Player[1].qChot[1];
13.                    dem++;
14.                }
15.                if (Player[1].qChot[2].TrangThai == 1 &&
Player[1].qChot[2].Khoa == false)
16.                    {
17.                        q[dem] = Player[1].qChot[2];
18.                        dem++;
19.                    }
20.                    if (Player[1].qChot[3].TrangThai == 1 &&
Player[1].qChot[3].Khoa == false)
21.                        {
22.                            q[dem] = Player[1].qChot[3];
23.                            dem++;
24.                        }
25.                        if (Player[1].qChot[4].TrangThai == 1 &&
Player[1].qChot[4].Khoa == false)
26.                            {
27.                                q[dem] = Player[1].qChot[4];
28.                                dem++;
29.                            }
30.                            if (Player[1].qSy[0].TrangThai == 1 && Player[1].qSy[0].Khoa
== false)
31.                                {
32.                                    q[dem] = Player[1].qSy[0];
33.                                    dem++;
34.                                }
35.                                if (Player[1].qSy[1].TrangThai == 1 && Player[1].qSy[1].Khoa
== false)
36.                                    {
37.                                        q[dem] = Player[1].qSy[1];
38.                                        dem++;
39.                                    }
40.                                    if (Player[1].qTinh[1].TrangThai == 1 &&
Player[1].qTinh[1].Khoa == false)
41.                                        {
42.                                            q[dem] = Player[1].qTinh[1];
43.                                            dem++;
44.                                        }
45.                                        if (Player[1].qTinh[0].TrangThai == 1 &&
Player[1].qTinh[0].Khoa == false)
46.                                            {
47.                                                q[dem] = Player[1].qTinh[0];
48.                                                dem++;
49.                                            }
50.
51.        if (Player[1].qXe[1].TrangThai == 1 && Player[1].qXe[1].Khoa
== false)
```

```
52.          {
53.              q[dem] = Player[1].qXe[1];
54.              dem++;
55.          }
56.          if (Player[1].qXe[0].TrangThai == 1 && Player[1].qXe[0].Khoa
== false)
57.          {
58.              q[dem] = Player[1].qXe[0];
59.              dem++;
60.          }
61.          if (Player[1].qMa[1].TrangThai == 1 && Player[1].qMa[1].Khoa
== false)
62.          {
63.              q[dem] = Player[1].qMa[1];
64.              dem++;
65.          }
66.          if (Player[1].qMa[0].TrangThai == 1 && Player[1].qMa[0].Khoa
== false)
67.          {
68.              q[dem] = Player[1].qMa[0];
69.              dem++;
70.          }
71.          if (Player[1].qPhao[0].TrangThai == 1 &&
Player[1].qPhao[0].Khoa == false)
72.          {
73.              q[dem] = Player[1].qPhao[0];
74.              dem++;
75.          }
76.          if (Player[1].qPhao[1].TrangThai == 1 &&
Player[1].qPhao[1].Khoa == false)
77.          {
78.              q[dem] = Player[1].qPhao[1];
79.              dem++;
80.          }
81.          if (Player[1].qTuong.TrangThai == 1 && Player[1].qTuong.Khoa
== false)
82.          {
83.              q[dem] = Player[1].qTuong;
84.          }
85.          if (Player[0].qChot[0].TrangThai == 1 &&
Player[0].qChot[0].Khoa == false)
86.          {
87.              q[dem] = Player[0].qChot[0];
88.              dem++;
89.          }
90.          if (Player[0].qChot[1].TrangThai == 1 &&
Player[0].qChot[1].Khoa == false)
91.          {
92.              q[dem] = Player[0].qChot[1];
93.              dem++;
94.          }
95.          if (Player[0].qChot[2].TrangThai == 1 &&
Player[0].qChot[1].Khoa == false)
96.          {
97.              q[dem] = Player[0].qChot[2];
98.              dem++;
99.          }
100.         if (Player[0].qChot[3].TrangThai == 1 &&
Player[0].qChot[3].Khoa == false)
```

```
101.          {
102.              q[dem] = Player[0].qChot[3];
103.              dem++;
104.          }
105.          if (Player[0].qChot[4].TrangThai == 1 &&
    Player[0].qChot[4].Khoa == false)
106.          {
107.              q[dem] = Player[0].qChot[4];
108.              dem++;
109.          }
110.          if (Player[0].qSy[0].TrangThai == 1 &&
    Player[0].qSy[0].Khoa == false)
111.          {
112.              q[dem] = Player[0].qSy[0];
113.              dem++;
114.          }
115.          if (Player[0].qSy[1].TrangThai == 1 &&
    Player[0].qSy[1].Khoa == false)
116.          {
117.              q[dem] = Player[0].qSy[1];
118.              dem++;
119.          }
120.          if (Player[0].qTinh[1].TrangThai == 1 &&
    Player[0].qTinh[1].Khoa == false)
121.          {
122.              q[dem] = Player[0].qTinh[1];
123.              dem++;
124.          }
125.          if (Player[0].qTinh[0].TrangThai == 1 &&
    Player[0].qTinh[0].Khoa == false)
126.          {
127.              q[dem] = Player[0].qTinh[0];
128.              dem++;
129.          }
130.          if (Player[0].qPhao[1].TrangThai == 1 &&
    Player[0].qPhao[1].Khoa == false)
131.          {
132.              q[dem] = Player[0].qPhao[1];
133.              dem++;
134.          }
135.          if (Player[0].qPhao[0].TrangThai == 1 &&
    Player[0].qPhao[0].Khoa == false)
136.          {
137.              q[dem] = Player[0].qPhao[0];
138.              dem++;
139.          }
140.          if (Player[0].qXe[1].TrangThai == 1 &&
    Player[0].qXe[1].Khoa == false)
141.          {
142.              q[dem] = Player[0].qXe[1];
143.              dem++;
144.          }
145.          if (Player[0].qXe[0].TrangThai == 1 &&
    Player[0].qXe[0].Khoa == false)
146.          {
147.              q[dem] = Player[0].qXe[0];
148.              dem++;
149.          }
```

```

150.           if (Player[0].qMa[1].TrangThai == 1 &&
    Player[0].qMa[1].Khoa == false)
151.           {
152.               q[dem] = Player[0].qMa[1];
153.               dem++;
154.           }
155.           if (Player[0].qMa[0].TrangThai == 1 &&
    Player[0].qMa[0].Khoa == false)
156.           {
157.               q[dem] = Player[0].qMa[0];
158.               dem++;
159.           }
160.           if (Player[0].qTuong.TrangThai == 1 &&
    Player[0].qTuong.Khoa == false)
161.           {
162.               q[dem] = Player[0].qTuong;
163.           }
164.           int i, j;
165.           int ndd = 0;
166.           NuocDi[] nd = new NuocDi[200];
167.
168.           for (int k = 0; k <= dem; k++)
169.           {
170.               for (i = 0; i <= 9; i++)
171.               {
172.                   for (j = 0; j <= 8; j++)
173.                   {
174.                       if (q[k].KiemTra(i, j) == 1 &&
    q[k].TuongAnToan(i, j) == 1)
175.                       {
176.                           nd[ndd] = new NuocDi();
177.                           nd[ndd].Q = q[k];
178.                           nd[ndd].x = i;
179.                           nd[ndd].y = j;
180.                           ndd++;
181.                       }
182.                   }
183.               }
184.           }
185.           temp = --ndd;// so nuoc di sinh ra
186.           return nd;
187.       }

```

## b) Hàm computer

```

1. public static void Computer()
2. {
3.     //State root = new Board.State();
4.     //root.self = new List<QuanCo>();
5.     //root.enemy = new List<QuanCo>();
6.
7.     int i, j;
8.     DanhDau = new QuanCo();
9.     Alpha_Beta_pruning(-1000000, 1000000, MaxDepth);
10.    i = bestmove.x;
11.    j = bestmove.y;
12.    DanhDau = bestmove.Q;
13.
14.    switch (BanCo.ViTrix[i, j].Trong)
15.    {

```

```

16.          case true:
17.              if (DanhDau.Phe == 0)
18.              {
19.                  if (DanhDau.Ten == "tuong")
DanhDau.picQuanCo.Image = UIT_ChineseChess.Properties.Resources._1tuong;
20.                  if (DanhDau.Ten == "sy") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._1sy;
21.                  if (DanhDau.Ten == "tinh") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._1tinh;
22.                  if (DanhDau.Ten == "xe") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._1xe;
23.                  if (DanhDau.Ten == "phao") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._1phao;
24.                  if (DanhDau.Ten == "ma") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._1ma;
25.                  if (DanhDau.Ten == "chot") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._1chot;
26.              }
27.              if (DanhDau.Phe == 1)
28.              {
29.                  if (DanhDau.Ten == "tuong")
DanhDau.picQuanCo.Image = UIT_ChineseChess.Properties.Resources._2tuong;
30.                  if (DanhDau.Ten == "sy") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._2sy;
31.                  if (DanhDau.Ten == "tinh") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._2tinh;
32.                  if (DanhDau.Ten == "xe") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._2xe;
33.                  if (DanhDau.Ten == "phao") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._2phao;
34.                  if (DanhDau.Ten == "ma") DanhDau.picQuanCo.Image =
UIT_ChineseChess.Properties.Resources._2ma;
35.                  if (DanhDau.Ten == "chot") DanhDau.picQuanCo.Image
= UIT_ChineseChess.Properties.Resources._2chot;
36.              }
37.              //Bỏ chọn quân cờ
38.              Marked = false;
39.
40.              //Ô cờ trống tại vị trí ban đầu
OCoTrong(DanhDau.Hang, DanhDau.Cot);
42.
43.              //Đặt quân cờ đã chọn vào vị trí mới [i,j]
DatQuanCo(DanhDau, i, j);
45.              //Tiếng động
46.              if (AmThanh) ClickSound("0");
47.
48.              //Kiểm tra chiếu tướng
KiemTraChieuTuong();
50.
51.              //Thay đổi lượt đi
DoiLuotDi();
53.
54.              //Kiểm tra chiếu bí
KiemTraChieuBi();
56.              if (winner != 2)
{
58.                  ThongBaoChieuTuong.Visible = false;
59.                  ChieuBi.Visible = true;
60.              }

```

```

61.                     else ChieuBi.Visible = false;
62.
63.                     BanCo.ResetCanMove();
64.                     break;
65.
66.                 case false:
67.                     if (DanhDau.Phe == 0)
68.                     {
69.                         if (DanhDau.Ten == "tuong")
70.                             DanhDau.picQuanCo.Image = UIT_ChineseChess.Properties.Resources._1tuong;
71.                             if (DanhDau.Ten == "sy") DanhDau.picQuanCo.Image =
72.                                 UIT_ChineseChess.Properties.Resources._1sy;
73.                                 if (DanhDau.Ten == "tinh") DanhDau.picQuanCo.Image =
74.                                     UIT_ChineseChess.Properties.Resources._1tinh;
75.                                     if (DanhDau.Ten == "xe") DanhDau.picQuanCo.Image =
76.                                         UIT_ChineseChess.Properties.Resources._1xe;
77.                                         if (DanhDau.Ten == "phao") DanhDau.picQuanCo.Image =
78.                                             UIT_ChineseChess.Properties.Resources._1phao;
79.                                             if (DanhDau.Ten == "ma") DanhDau.picQuanCo.Image =
80.                                                 UIT_ChineseChess.Properties.Resources._1ma;
81.                                                 if (DanhDau.Ten == "chot") DanhDau.picQuanCo.Image =
82.                                                     UIT_ChineseChess.Properties.Resources._1chot;
83.                                                     }
84.                                                     if (DanhDau.Phe == 1)
85.                                                     {
86.                                                         if (DanhDau.Ten == "tuong")
87.                                                             DanhDau.picQuanCo.Image = UIT_ChineseChess.Properties.Resources._2tuong;
88.                                                             if (DanhDau.Ten == "sy") DanhDau.picQuanCo.Image =
89.                                                                 UIT_ChineseChess.Properties.Resources._2sy;
90.                                                                 if (DanhDau.Ten == "tinh") DanhDau.picQuanCo.Image =
91.                                                                     UIT_ChineseChess.Properties.Resources._2tinh;
92.                                                                     if (DanhDau.Ten == "xe") DanhDau.picQuanCo.Image =
93.                                                                         UIT_ChineseChess.Properties.Resources._2xe;
94.                                                                         if (DanhDau.Ten == "phao") DanhDau.picQuanCo.Image =
95.                                                                             UIT_ChineseChess.Properties.Resources._2phao;
96.                                                                             if (DanhDau.Ten == "ma") DanhDau.picQuanCo.Image =
97.                                                                                 UIT_ChineseChess.Properties.Resources._2ma;
98.                                                                                 if (DanhDau.Ten == "chot") DanhDau.picQuanCo.Image =
99.                                                                 UIT_ChineseChess.Properties.Resources._2chot;
100.                                                                 }
101.                                                                
int phekia = 2;
if (DanhDau.Phe == 0) phekia = 1;
else phekia = 0;
QuanCo temp;
temp = new QuanCo();

if (BanCo.ViTrix[i, j].Ten == "tuong") temp =
    Player[phekia].qTuong;
if (BanCo.ViTrix[i, j].Ten == "sy")
{
    if (BanCo.ViTrix[i, j].ThuTu == "0") temp =
        Player[phekia].qSy[0];
    if (BanCo.ViTrix[i, j].ThuTu == "1") temp =
        Player[phekia].qSy[1];
}
if (BanCo.ViTrix[i, j].Ten == "tinh")
{

```

```

102.           if (BanCo.ViTri[i, j].ThuTu == "0") temp =
103.           Player[phekia].qTinh[0];
104.           if (BanCo.ViTri[i, j].ThuTu == "1") temp =
105.           Player[phekia].qTinh[1];
106.       }
107.       if (BanCo.ViTri[i, j].Ten == "xe")
108.       {
109.           if (BanCo.ViTri[i, j].ThuTu == "0") temp =
110.           Player[phekia].qXe[0];
111.           if (BanCo.ViTri[i, j].ThuTu == "1") temp =
112.       }
113.       if (BanCo.ViTri[i, j].Ten == "phao")
114.       {
115.           if (BanCo.ViTri[i, j].ThuTu == "0") temp =
116.           Player[phekia].qPhao[0];
117.           if (BanCo.ViTri[i, j].ThuTu == "1") temp =
118.       }
119.       if (BanCo.ViTri[i, j].Ten == "ma")
120.       {
121.           if (BanCo.ViTri[i, j].ThuTu == "0") temp =
122.           Player[phekia].qMa[0];
123.           if (BanCo.ViTri[i, j].ThuTu == "1") temp =
124.           Player[phekia].qMa[1];
125.       }
126.       if (BanCo.ViTri[i, j].Ten == "chot")
127.       {
128.           if (BanCo.ViTri[i, j].ThuTu == "0") temp =
129.           if (BanCo.ViTri[i, j].ThuTu == "1") temp =
130.           if (BanCo.ViTri[i, j].ThuTu == "2") temp =
131.           if (BanCo.ViTri[i, j].ThuTu == "3") temp =
132.           if (BanCo.ViTri[i, j].ThuTu == "4") temp =
133.       }
134.       //Bỏ chọn quân cờ
135.       Marked = false;
136.       //Ăn quân cờ của đối phương
137.       AnQuanCo(temp);
138.       //Trả lại ô cờ trống
139.       OCoTrong(DanhDau.Hang, DanhDau.Cot);
140.       //Thiết lập quân cờ đã chọn vào bàn cờ
141.       DatQuanCo(DanhDau, i, j);
142.       //Tiếng động
143.       if (AmThanh) ClickSound(DanhDau.Ten);
144.       //Kiểm tra chiếu tướng
145.       KiemTraChieuTuong();
146.       //Thay đổi lượt đi
147.       DoiLuotDi();

```

```

148.
149.                                //Kiểm tra chiếu bí
150.                                KiemTraChieuBi();
151.                                if (winner != 2)
152.                                {
153.                                    ThongBaoChieuTuong.Visible = false;
154.                                    ChieuBi.Visible = true;
155.                                }
156.                                else ChieuBi.Visible = false;
157.
158.                                break;
159.                            }
160.
161.                        }

```

### c) Hàm heuristic

```

1. public static int Heuristic()
2. {
3.     int s = 0;
4.
5.     for (int i = 0; i < 5; i++)
6.         if (Player[1].qChot[i].TrangThai == 1)
7.         {
8.             s -= (2*30 + RedPawnPosition[Player[1].qChot[i].Hang,
    Player[1].qChot[i].Cot]);
9.
10.            }
11.            for (int i = 0; i < 5; i++)
12.                if (Player[0].qChot[i].TrangThai == 1)
13.                {
14.
15.                    s += (2*30 +
    BlackPawnPosition[Player[0].qChot[i].Hang, Player[0].qChot[i].Cot]);
16.                }
17.                for (int i = 0; i < 2; i++)
18.                    if (Player[0].qSy[i].TrangThai == 1)
19.                    {
20.
21.                        s += (2*40 +
    BlackAdvisorPosition[Player[0].qSy[i].Hang, Player[0].qSy[i].Cot]);
22.                    }
23.                    for (int i = 0; i < 2; i++)
24.                        if (Player[1].qSy[i].TrangThai == 1)
25.                        {
26.
27.                            s -= (2*40 + RedAdvisorPosition[Player[1].qSy[i].Hang,
    Player[1].qSy[i].Cot]);
28.                        }
29.                        for (int i = 0; i < 2; i++)
30.                            if (Player[1].qTinh[i].TrangThai == 1)
31.                            {
32.
33.                                s -= (2*40 +
    RedBishopPosition[Player[1].qTinh[i].Hang, Player[1].qTinh[i].Cot]);
34.                            }
35.                            for (int i = 0; i < 2; i++)
36.                                if (Player[0].qTinh[i].TrangThai == 1)
37.                                {
38.

```

```

39.           s += (2*40 +
    BlackBishopPosition[Player[0].qTinh[i].Hang, Player[0].qTinh[i].Cot]);
40.       }
41.       for (int i = 0; i < 2; i++)
42.           if (Player[0].qPhao[i].TrangThai == 1)
43.           {
44.
45.           s += (2*90 +
    BlackCannonPosition[Player[0].qPhao[i].Hang, Player[0].qPhao[i].Cot]);
46.       }
47.       for (int i = 0; i < 2; i++)
48.           if (Player[1].qPhao[i].TrangThai == 1)
49.           {
50.               s -= (2*90 +
    RedCannonPosition[Player[1].qPhao[i].Hang, Player[1].qPhao[i].Cot]);
51.           }
52.           for (int i = 0; i < 2; i++)
53.               if (Player[1].qXe[i].TrangThai == 1)
54.               {
55.                   s -= (2*180 + RedRookPosition[Player[1].qXe[i].Hang,
    Player[1].qXe[i].Cot]);
56.               }
57.               for (int i = 0; i < 2; i++)
58.                   if (Player[0].qXe[i].TrangThai == 1)
59.                   {
60.                       s += (2*180 + BlackRookPosition[Player[0].qXe[i].Hang,
    Player[0].qXe[i].Cot]);
61.                   }
62.                   for (int i = 0; i < 2; i++)
63.                       if (Player[1].qMa[i].TrangThai == 1)
64.                       {
65.                           s -= (2*80 + RedKnightPosition[Player[1].qMa[i].Hang,
    Player[1].qMa[i].Cot]);
66.                       }
67.                       for (int i = 0; i < 2; i++)
68.                           if (Player[0].qMa[i].TrangThai == 1)
69.                           {
70.                               s += (2*80 +
    BlackKnightPosition[Player[0].qMa[i].Hang, Player[0].qMa[i].Cot]);
71.                           }
72.                           if (Player[0].qTuong.TrangThai == 1)
73.                           {
74.                               s += (2*10000 + BlackKingPosition[Player[0].qTuong.Hang,
    Player[0].qTuong.Cot]);
75.                           }
76.                           if (Player[1].qTuong.TrangThai == 1)
77.                           {
78.
79.                               s -= (2*10000 + RedKingPosition[Player[1].qTuong.Hang,
    Player[1].qTuong.Cot]);
80.                           }
81.           return s;
82.       }

```

ରେଣ୍ଡିଶନ୍

## Chương 5

# KẾT LUẬN

### I. KẾT QUẢ ĐẠT ĐƯỢC

- Hoàn thành đồ án “Trò chơi Cờ Tướng” theo yêu cầu môn học.
- Áp dụng được lý thuyết môn học trí tuệ nhân tạo vào thực tiễn, cụ thể là áp dụng thủ tục minimax – cắt tỉa alpha-beta và thuật giải heuristic để giúp nước đi quân cờ nhanh và thông minh.
- Tăng khả năng làm việc nhóm, biết phân chia công việc, cũng như trách nhiệm phải thực hiện của từng thành viên. Hỗ trợ được kỹ năng mềm cho nghề nghiệp tương lai.
- Nâng cao khả năng kỹ thuật lập trình, giúp hoàn thành các môn học các kì học tiếp theo.
- Tăng khả năng tư duy, suy nghĩ khi đối mặt với một đồ án khó. Đòi hỏi ý chí, tinh thần cao, đồng thời, kiến thức là điều không thể thiếu.
- Giúp cải thiện kỹ năng tìm kiếm, lựa chọn, sàng lọc và kế thừa những tinh hoa của người đi trước.
- Nâng cao kỹ năng viết báo cáo, tính kỹ luật về hình thức lẫn nội dung đáp ứng yêu cầu giảng viên.
- Luyện tập kỹ năng vấn đáp, rèn luyện sự tự tin khi phỏng vấn.

### II. HẠN CHẾ

#### 1) Quá trình thực hiện

- Xung đột ý tưởng trong cách tổ chức và lưu trữ dữ liệu.
- Chưa sắp xếp được thời gian cá nhân hợp lý.
- Không giải quyết được vấn đề khó khăn một cách nhanh chóng.

#### 2) Ứng dụng

- Khi thực hiện độ sâu của cây trò chơi càng cao, các quân cờ thực hiện nước đi càng lâu.
- Các chức năng tương tác người chơi còn hạn chế.

### III. HƯỚNG PHÁT TRIỂN

- Áp dụng thủ tục minimax – cắt tỉa alpha-beta vào các trò chơi có tính đối kháng khác.
- Tìm ra các nước đi quân cờ tối ưu hơn, đồng thời thực hiện hàm heuristic tốt hơn để giúp các nước cho quân cờ thông minh hơn.

❖❖❖❖❖

# TÀI LIỆU THAM KHẢO

- Tìm hiểu về cờ tướng trên trang wikipedia

Link: [https://vi.wikipedia.org/wiki/C%C3%A1%BB%9D\\_t%C3%A1%BB%9Bng](https://vi.wikipedia.org/wiki/C%C3%A1%BB%9D_t%C3%A1%BB%9Bng)

Chương 1: Mục I, II.

- Lợi ích của cờ tướng

Link: <http://hoicotuongvn.blogspot.com/2016/12/5-loi-ich-khi-choi-co-tuong.html>

Chương 1: Mục III.

- Tài liệu ngôn ngữ C#

Chương 2: Mục I

- [1] Lý thuyết cây trò chơi: slide trong chương trình học

- [2] Lý thuyết thủ tục minimax, cắt tỉa alpha-beta

Link: <https://voer.edu.vn/c/su-dung-heuristic-trongcac-tro-choi/764b3239/a66239c3>

- [3] Ví dụ cắt tỉa alpha-beta

Link: <https://www.stdio.vn/articles/read/564/giai-thuat-cat-tia-alpha-beta>

- Tham khảo source code: sử dụng cấu trúc dữ liệu đã có( các class, code giao diện), sau đó tự xây dựng và áp dụng yêu cầu giảng viên để thực hiện đồ án.

Về mặt hình ảnh: được thiết kế lại hoàn toàn.

[https://www.youtube.com/watch?v=PwYEQoyty\\_w](https://www.youtube.com/watch?v=PwYEQoyty_w)

- Nội dung đã chỉnh sửa: cả chương 3 và chương 4.