**ĐẠI HỌC KINH TẾ - TÀI CHÍNH TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYỄN MINH HUY**

175050002

**TRÒ CHƠI CARO**

BÁO CÁO DỰ ÁN

**TRÍ TUỆ NHÂN TẠO**

GIẢNG VIÊN

**TS. NGUYỄN HÀ GIANG**

**TP. HỒ CHÍ MINH, 08.04.2019**

Khoa Công nghệ Thông tin

Course:

**KẾ HOẠCH**

|  |
| --- |
| **Tên Dự án:** Trò chơi Caro |
| **Thời gian:** 18.02.2019 - 08.04.2019 |
| **Nhóm:**   * Học sinh:   + Họ & Tên: Nguyễn Minh Huy   + Mã số sinh viên: 175050002   + Lớp: 17D1TH01 |
| **Thông tin dự án:**  Trò chơi Caro sử dụng kỹ thuật lượng giá Heuristic, thuật toán Minimax và giải thuật cắt tỉa Alpha – Beta. Với giao diện thiết kế đơn giản, nhiều chức năng, nhiều độ khó.  Trò chơi xây dựng bằng Winform với ngôn ngữ lập trình C#, trong trương trình có rất nhiều kỹ thuật lập trình từ cơ bản đến nâng cao. |
| **Lịch trình:**   * 18.02 - 11.03: * Tìm hiểu về Trí tuệ nhân tạo * Lên ý tưởng thiết kế phần mềm, giao diện, chức năng,... * 11.03 - 17.03: * Tìm hiểu về C# và Winform trên nền tảng IDE Visual Studio * 18.03 - 21.03: * Thiết kế giao diện bàn cờ, các nút chức năng * Xây dựng chức năng của trò chơi, đánh quân cờ, đổi lượt người chơi * Chức năng xét thắng thua trên đường ngang, dọc và chéo * 21.03 - 01.04: * Xây dựng hàm lượng giá Heuristic cho các ô trống * Xây dựng cây trò chơi và áp dụng thuật toán Minimax * Xây dựng giải thuật Alpha - Beta để giảm bớt không gian tìm kiếm * 01.04 - 08.04: * Sửa lỗi trương trình và thêm các chức năng cho trò chơi * Hoàn thiện trò chơi và viết báo cáo |

**NỘI DUNG**

[CHƯƠNG 1 MỞ ĐẦU 4](#_Toc4957626)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 5](#_Toc4957627)

[2.1 Trí tuệ nhân tạo 5](#_Toc4957628)

[2.2 Thuật giải Heuristic 6](#_Toc4957629)

[2.3 Thuật toán Minimax 7](#_Toc4957630)

[2.4 Giải thuật Alpha - Beta 7](#_Toc4957631)

[CHƯƠNG 3 PHÂN TÍCH ĐỀ TÀI 8](#_Toc4957632)

[3.1 Yêu cầu bài toán 8](#_Toc4957633)

[3.2 Phân tích yêu cầu 8](#_Toc4957634)

[3.3 Phương pháp giải quyết 8](#_Toc4957635)

[CHƯƠNG 4 XÂY DỰNG ỨNG DỤNG 10](#_Toc4957636)

[4.1 Giao diện và chức năng 10](#_Toc4957637)

[4.2 Cài đặt thuật toán 12](#_Toc4957638)

[4.2.1 Các thuộc tính và chức năng của trò chơi 12](#_Toc4957639)

[4.2.2 Thuật toán AI 14](#_Toc4957640)

[CHƯƠNG 5 KẾT LUẬN 16](#_Toc4957641)

[5.1 Đánh giá 16](#_Toc4957642)

[5.2 Nhược điểm 16](#_Toc4957643)

[5.3 Hướng phát triển 16](#_Toc4957644)

[CHƯƠNG 6 TÀI LIỆU THAM KHẢO 17](#_Toc4957645)

[6.1 Tham khảo lý thuyết 17](#_Toc4957646)

[6.2 Tham khảo code 17](#_Toc4957647)

# MỞ ĐẦU

Cờ Caro là một trò chơi rất phổ biến, có cấu trúc đơn giản chỉ gồm 2 quân cờ “X” và “O”, là một loại cờ cổ xưa của Trung Quốc, là một trò chơi bàn cờ theo chiến thuật trừu tượng. Cờ ca rô trong tiếng Hàn Quốc là omok, tiếng Trung là wǔzǐqí, trong tiếng Nhật là gomoku narabe và trong tiếng Anh là Connect 5,... Hai bên lần lượt đi các nước cờ. Người thắng là người có được 5 quân liên tục theo hàng ngang, dọc hoặc chéo.

Hiện nay, việc ứng dụng trí tuệ nhân tạo vào việc phát triển trò chơi đã trở nên vô cùng phố biến, đặc biệt là những trò chơi mang tính trí tuệ cao. Chính sự đơn giản vốn có nhưng lại có các đặc tính của một trò chơi trí tuệ, em quyết định lựa chọn đề tài làm trò chơi cờ Caro cho môn Trí tuệ nhân tạo.

Với môn học Trí tuệ nhân tạo, dự án được áp dụng thuật toán Minimax, giải thuật Alpha - Beta và các kiến thức cơ bản về ngôn ngữ C# như danh sách (List), hướng đối tượng (OOP), giao diện (Interface),... cùng với sự hỗ trợ của Winform .NET trên nền tảng IDE Visual Studio 2017.

Em thực hiện đề tài này với mục đích chính là hiểu rõ hơn về Trí tuệ nhân tạo, qua đó xây dựng được trò chơi Caro có tính nhân tạo cao. Tuy nhiên do thời gian hạn chế nên em vẫn chưa tối ưu hóa được các thuật toán sử dụng, trong quá trình thực hiện vẫn còn tồn đọng một vài lỗi và sai sót, mong nhận được góp ý và đánh giá của thầy.

# CƠ SỞ LÝ THUYẾT

## Trí tuệ nhân tạo

* **Định nghĩa**

Trí tuệ nhân tạo hay trí thông minh nhân tạo (Artificial, AI) là một ngành thuộc lĩnh vực khoa học máy tính (Computer science). Là trí tuệ do con người lập trình tạo nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi thông minh như con người. Trí tuệ nhân tạo khác với việc lập trình logic trong các ngôn ngữ lập trình là ở việc ứng dụng các hệ thống học máy (machine learning) để mô phỏng trí tuệ của con người trong các xử lý mà con người làm tốt hơn máy tính. Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ, tiếng nói, biết học và tự thích nghi,...

Trí tuệ nhân tạo bao gồm các cơ sở lý thuyết và việc lập trình xây dựng của các hệ thống máy tính có thể thực hiện các nhiệm vụ thường đòi hỏi trí thông minh của con người như nhận thức thị giác, nhận dạng giọng nói, ra quyết định và dịch giữa các ngôn ngữ.

* Trí tuệ nhân tạo giúp tạo ra máy tính có khả năng suy nghĩ, máy tính có trí tuệ theo đầy đủ nghĩa của từ này (Haugeland, 1985).
* Trí tuệ nhân tạo là khoa học nghiên cứu xem thế nào để máy tính có thể thực hiện được những công việc mà con người làm tốt hơn máy tính (Rich và Knight,1991).
* Trí tuệ nhân tạo là khoa học nghiên cứu các hoạt động trí não thông qua các mô hình tính toán (Chaniaka và McDemott, 1985).
* Trí tuệ nhận tạo nghiên cứu các mô hình máy tính có thể nhận thức, lập luận và hành động (Winston, 1992)
* Trí tuệ nhân tạo nghiên cứu các hành vi thông minh mô phỏng các vật thể nhân tạo (Nilsson, 1998)
* Trí tuệ nhân tạo là khoa học nghiên cứu các hành vi thông minh nhằm giải quyết các vấn đề được đặt ra đối với các chương trình máy tính (Học viện Kỹ thuật Quân sự).
* **Mục đích**

Các nhà nghiên cứu đầu tiên đã phát triển các thuật toán bắt chước theo lý luận từng bước mà con người sử dụng khi giải quyết các câu đố hoặc đưa ra các phương pháp loại trừ logic. Vào cuối những năm 1980 và 1990, nghiên cứu về AI đã phát triển các phương pháp xử lý thông tin không chắc chắn hoặc không đầy đủ, sử dụng các khái niệm từ xác suất và kinh tế.

Đối với những vấn đề khó, các thuật toán bắt buộc phải có phần cứng đủ mạnh để thực hiện phép tính toán khổng lồ - để trải qua "vụ nổ tổ hợp": lượng bộ nhớ và thời gian tính toán có thể trở nên vô tận nếu giải quyết một vấn đề khó. Mức độ ưu tiên cao nhất là tìm kiếm các thuật toán giải quyết vấn đề.

Con người thường sử dụng các phán đoán nhanh và trực quan chứ không phải là phép khấu trừ từng bước mà các nghiên cứu AI ban đầu có thể mô phỏng. AI đã tiến triển bằng cách sử dụng cách giải quyết vấn đề "biểu tượng phụ": cách tiếp cận tác nhân được thể hiện nhấn mạnh tầm quan trọng của các kỹ năng cảm biến động đến lý luận cao hơn; nghiên cứu mạng thần kinh cố gắng để mô phỏng các cấu trúc bên trong não làm phát sinh kỹ năng này. Các phương pháp tiếp cận thống kê đối với AI bắt chước khả năng của con người.

## Thuật giải Heuristic

Heuristic ("tìm kiếm" hoặc "khám phá") là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hay khám phá nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu. Với việc nghiên cứu khảo sát không có tính thực tế, các phương pháp heuristic được dùng nhằm tăng nhanh quá trình tìm kiếm với các giải pháp hợp lý thông qua các suy nghĩ rút gọn để giảm bớt việc nhận thức vấn đề khi đưa ra quyết định. Ví dụ của phương pháp này bao gồm sử dụng một luật ngón tay cái, giả thuyết, phán đoán trực giác, khuôn mẫu hay nhận thức thông thường.

Thuật giải Heuristic là một sự mở rộng khái niệm thuật toán. Nó thể hiện cách giải bài toán với các đặc tính sau:

* Thường tìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất)
* Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.
* Thuật giải Heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động của con người.

Có nhiều phương pháp để xây dựng một thuật giải Heuristic, trong đó người ta thường dựa vào một số nguyên lý cơ sở như sau:

**Nguyên lý vét cạn thông minh:** Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

**Nguyên lý tham lam (Greedy):** Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

**Nguyên lý thứ tự:** Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

**Hàm Heuristic:** Trong việc xây dựng các thuật giải Heuristic, người ta thường dùng các hàm Heuristic. Ðó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

## Thuật toán Minimax

Minimax (minmax) là một phương pháp trong lý thuyết quyết định có mục đích là tối thiểu hóa (minimize) tổn thất vốn được dự tính có thể là "tối đa" (maximize). Có thể hiểu ngược lại là, nó nhằm tối đa hóa lợi ích vốn được dự tính là tối thiểu (maximin). Nó bắt nguồn từ trò chơi có tổng bằng không. Nó cũng được mở rộng cho nhiều trò chơi phức tạp hơn và giúp đưa ra các quyết định chung khi có sự hiện diện của sự không chắc chắn.

Một phiên bản của giải thuật áp dụng cho các trò chơi như tic-tac-toe, khi mà mỗi người chơi có thể thắng, thua, hoặc hòa. Nếu người chơi A có thể thắng trong 1 nước đi, thì "nước đi tốt nhất" chính là nước đi để dẫn đến kết quả thắng đó. Nếu người B biết rằng có một nước đi mà dẫn đến tình huống người A có thể thắng ngay ở nước đi tiếp theo, trong khi nước đi khác thì sẽ dẫn đến tình huống mà người chơi A chỉ có thể, tốt nhất, là hòa thì nước đi tốt nhất của người B chính là nước đi sau.

Ta sẽ nắm rõ, thế nào là một nước đi "tốt nhất". Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi trở về đầu. Tại mỗi bước, nó sẽ ước định rằng người A đang cố gắng tối đa hóa cơ hội thắng của A khi đến phiên anh ta, còn ở nước đi kế tiếp thì người chơi B cố gắng để tối thiểu hóa cơ hội thắng của người A (nghĩa là tối đa hóa cơ hội thắng của B).

## Giải thuật Alpha - Beta

Giải thuật cắt tỉa Alpha-beta từng được nhiều nhà khoa học máy tính đề xuất ý tưởng và không ngừng được cải tiến cho đến ngày nay. Giải thuật này thường sử dụng chung với thuật toán tìm kiếm Minimax nhằm hỗ trợ giảm bớt các không gian trạng thái trong cây trò chơi, giúp thuật toán Minimax có thể tìm kiếm sâu và nhanh hơn. Giải thuật cắt tỉa Alpha-beta có nguyên tắc đơn giản "Nếu biết là trường hợp xấu thì không cần phải xét thêm".

Nút Max có một giá trị alpha (lớn hơn hoặc bằng alpha - luôn tăng), nút min có một giá trị beta (nhỏ hơn hoặc bằng beta - luôn giảm). Khi chưa có alpha và beta xác định thì thực hiện tìm kiếm sâu (depth-first) để xác định được alpha, beta, và truyền ngược lên các nút cha, nếu không tìm kiếm được giá trị phù hợp thì loại bỏ nhánh đi đó.

# PHÂN TÍCH ĐỀ TÀI

## Yêu cầu bài toán

Xây dựng một bàn cờ có kẻ các ô vuông với kích thước n x n (n > 5). Có 2 quân cờ là “X” và “O”. Người chơi có thể đánh với máy. Người thắng là người đi được 5 quân cờ cùng kiểu trên hàng ngang, dọc và hai đường chéo mà không bị quân đối phương chặn ở hai đầu. Hai bên hòa nhau khi bàn cơ hết chỗ đánh mà vẫn chưa phân được thắng bại.

## Phân tích yêu cầu

Bàn cờ (Board) bao gồm các ô cờ được đặt trong một ma trận với kích thước mặc định là 20x20. Trong mỗi phần từ của ma trận có thể xác định được:

* Vị trí của phần tử, hàng và cột
* Trạng thái của phần tử, ô đang trống hoặc đã được đánh

Giống như thực tế, người chơi thường đánh giá một số nước cờ là nguy hiểm, bình thường hoặc ít nguy hiểm. Máy tính cũng đánh giá như vậy, nhưng cụ thể hơn bằng các con số thể hiện.

## Phương pháp giải quyết

* **Tìm kiếm nước đi**

Trong trò chơi Caro, cứ sau một nước cờ, mỗi bên sẽ chọn ra từ những ô còn trống để đi, số ô trống còn lại sẽ giảm. Như vậy, việc tìm nước đi tiếp theo cho trạng thái có sẵn chỉ là việc tìm kiếm những ô trống còn lại, đồng thời, không gian tìm kiếm sẽ thu hẹp theo số nước đi đã tạo.

Không gian chọn nước đi từ mỗi trạng thái ban đầu là hữu hạn, nhưng không gian tìm kiếm một nước đi dẫn đến chiến thắng là rất lớn. Do đó ta không thể vét cạn không gian tìm kiếm nước đi này mà ta phải giới hạn không gian tìm kiếm. Một không gian tìm kiếm có thể hiện theo một cây đa phân và được gọi là cây tìm kiếm hoặc cây trò chơi.

Việc tìm kiếm nước đi là chọn một nút trên cây (độ sâu 1) sao cho nước đó là tốt. Nước đi tốt được xác định dựa trên khả năng dành chiến thắng là cao hay thấp sau khi nước đó được đi. Vì vậy muốn chọn một nước đi tốt thì nếu chỉ dựa vào thế cờ hiện tại là chưa đủ, mà phải biết thông tin của những thế cờ sau khi chọn nước này để đi.

* **Thuật toán Minimax**

Chiến lược được xác định bằng cách xét giá trị min, max đối với mỗi nút trong cây trò chơi. Max chọn nước đi ứng với giá trị lớn nhất, đem lại lợi thế tốt nhất cho mình. Ngược lại, Min chọn nước đi ứng với giá trị nhỏ nhất, tức là hạn chế tối đa lợi thế của đối thủ.

Vấn đề ở đây là các khả năng nước đi cần phải xét quá lớn, tốn thời gian và bộ nhớ để có thể duyệt hết được tất cả các trường hợp trong cây trò chơi với độ sâu cao. Vậy sẽ ra sao nếu ta có thể bỏ không xét đến một số nhánh tìm kiếm trong cây trò chơi?

* **Giải thuật Alpha – Beta**

Nếu một nhánh tìm kiếm nào đó không thể cải thiện đối với giá trị mà ta đã có, thì không cần phải xét đến nhánh tìm kiếm đó nữa.

Việc bỏ đi các nhánh tìm kiếm không ảnh hưởng đến kết quả cuối cùng là giá trị nước đi tốt nhất đối với Max tính đến hiện tại đối với nhánh tìm kiếm. Nếu tồn tại một giá trị thấp hơn giá trị đang có, Max sẽ bỏ qua nước đi đó. Và tương tự như thế đối với người chơi Min, bỏ qua các nước có giá trị cao hơn giá trị của nhánh đang tìm kiếm.

Đối với các trò chơi có không gian trạng thái lớn, thì phương pháp Alpha - Beta vẫn không phù hợp do không gian tìm kiếm vẫn quá lớn. Vậy liệu có thể hạn chế không gian tìm kiếm bằng cách sử dụng các tri thức cụ thể sẵn có áp dụng với bài toán cờ Caro?

* **Hàm lượng giá Heuristic**

Kỹ thuật lượng giá là một kỹ thuật quan trọng trong việc xây dựng trò chơi cờ Caro, lượng giá cho điểm trạng thái của bàn cờ để xây dựng cây trò chơi. Việc xây dựng hàm lượng giá hợp lý, chính xác sẽ giúp cho hệ thống có đánh giá chính xác về trạng thái bàn cờ để đưa ra nước đi có hiệu quả cao hơn.

Đối với bài toán cờ Caro, ta có thể dùng một hàm lượng giá để đánh giá tính tốt, xấu tại một thời điểm. Những ô nào ở gần các quân đã đánh trước sẽ được điểm cao hơn. Những ô càng xa thì được càng ít điểm. Tuy nhiên nếu vùng có 2, 3,... quân liên tiếp thì số điểm được cộng thêm sẽ nhiều hơn dựa vào số quân, tức càng nhiều quân thì điểm cộng càng nhiều hơn. Tồn tại hai bảng điểm tương ứng với tấn công và phòng thủ, hàm lượng giá sẽ xác định nước đi tiếp theo nên tấn công hay chặn nước đi của đối phương.

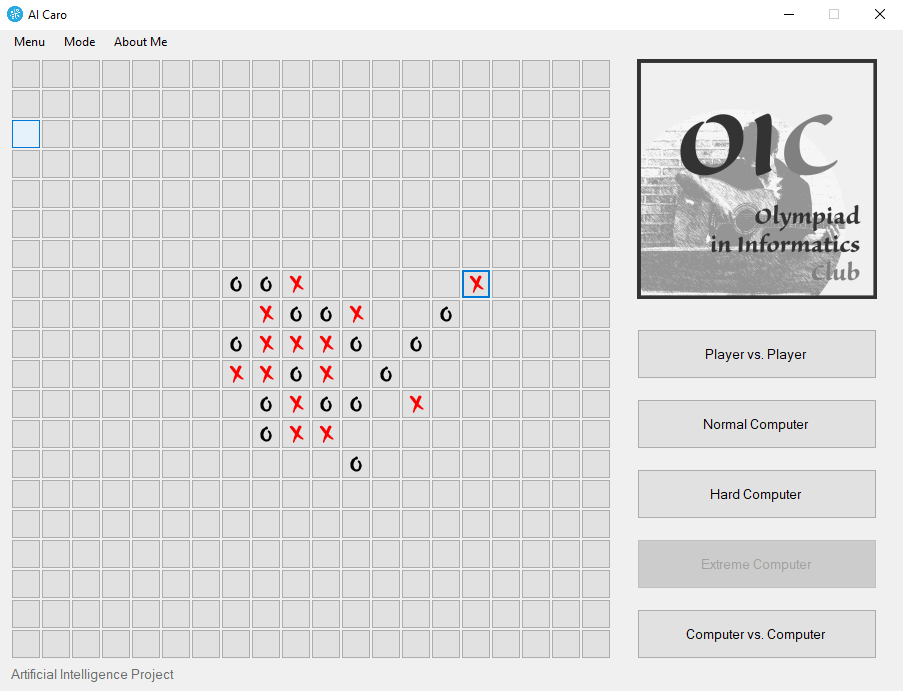
Sau mỗi nước đi, hệ thống sẽ kiểm tra bàn cờ tìm các thế cờ đó rồi tùy vào độ lợi thế đã định trước để tính ra điểm ứng với mỗi ô trống. Dựa vào số lượng quân ở mỗi điểm đang xét, hàm lượng giá sẽ đưa ra số điểm ứng với thuật toán lượng giá được cài đặt.

# XÂY DỰNG ỨNG DỤNG

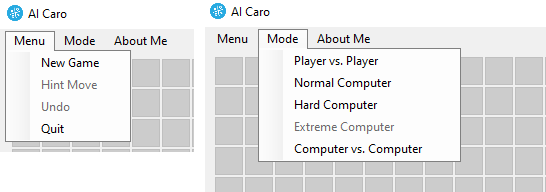
## Giao diện và chức năng

* **Giao diện**

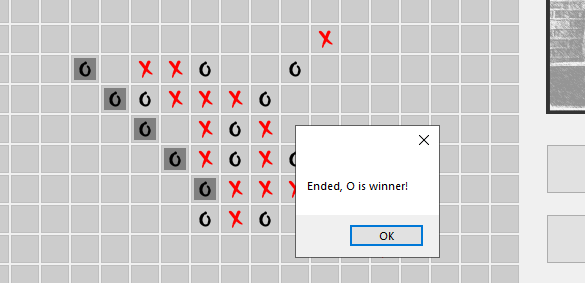
Giao diện tổng quát của trò chơi



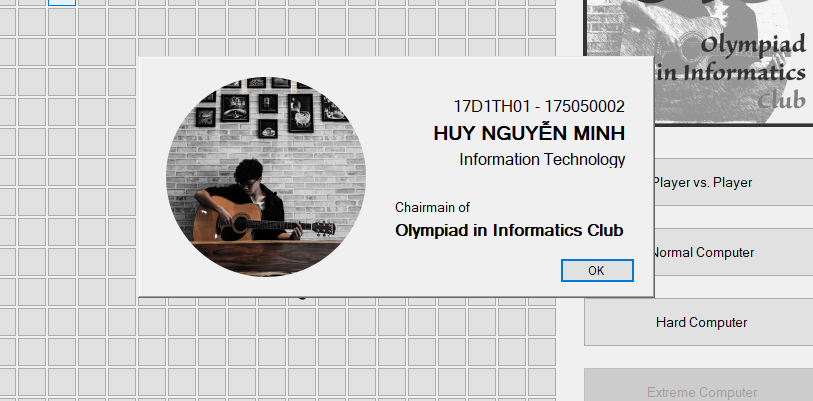
Menu Strip chứa các lựa chọn chức năng trong trò chơi



Xử lý thắng thua, kết quả trò chơi



Thông tin của người lập trình

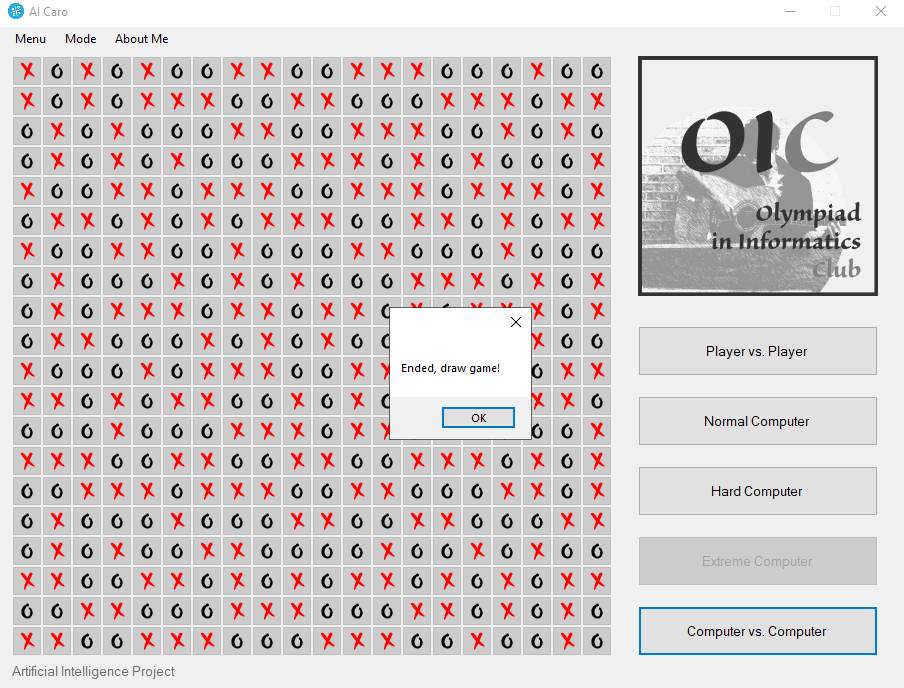


* **Chức năng**

Trò chơi có các chức năng:

* New Game: Xóa và tạo lại bàn cờ, lựa chọn chức năng để bắt đầu chơi.
* Hint Move: Gợi ý nước đi tốt cho người chơi, sử dụng thuật toán của máy.
* Undo: Hồi lại một nước đã đi, nếu chơi giữa người với người chỉ hồi một nước đi, còn nếu đánh với máy sẽ hồi cả nước đi của người chơi và của máy.
* Exit: Thoát khỏi trò chơi.

Kết quả hòa khi không còn nước đi trống



Bàn cờ được tạo bằng danh sách các Button trên một Panel của Winform. Hình ảnh của quân cờ được load từ file png.

Chương trình gồm các Class:

* Value.cs: Chứa các biến về thông số của bàn cờ, phương thức phục vụ cho việc xử lí các thao tác đánh cờ, lượng giá, xử lý AI,...
* Algorithm.cs: Chứa tất cả các thuật toán liên quan đến trò chơi như tạo bàn cờ, xử lý lượt đánh của người chơi, xử lý thắng thua, thuật toán AI,...
* Board.cs: Chứa Panel bàn cờ, các sự kiện (Event) của các Button ứng với mỗi chức năng trong trò chơi.
* About.cs: Form về thông tin của người lập trình trò chơi.
* Program.cs: Chứa hàm Main(), thực thi chương trình.

## Cài đặt thuật toán

### Các thuộc tính, chức năng của trò chơi

* **public class Value**

Lưu trữ các giá trị hằng số của trò chơi.

* public static int chessWidth = 30;
* public static int chessHeight = 30;

Chiều dài và chiều rộng của một button trên bàn cờ.

* public static int chessBoardWidth = 20;
* public static int chessBoardHeight = 20;

Số lượng quân cờ theo hàng ngang và hàng dọc, mặc định là 20x20.

* public static int maxDepth = 9;
* public static int maxMove = 3;

Mỗi lượt đánh chọn ra 3 ô trống có giá trị điểm cao nhất, tiến hành đánh thử và tiếp tục chọn và lưu 3 ô trống tiếp theo có giá trị điểm cao nhất với độ sâu là 9.

* **public class Player**

Lưu tên và kí hiệu đánh (“X”, “O”) của mỗi người chơi.

* private string name;
* private Image mark;
* **public class Info**

Lưu vị trí và lượt người chơi hiện tại.

* private Point point;
* private int curPlayer;

Biến curPlayer lưu giá trị “0” là lượt người chơi hiện tại, “1” là người chơi tiếp theo.

* **public class Score : IComparer<Score>**

Lưu vị trí và số điểm của ô trống được xét.

* private int empScore;
* private int xPos, yPos;

Vị trí ô trống đang xét, “x” là hàng ngang, “y” là hàng dọc.

* public int Compare(Score a, Score b);

Hàm so sánh các class Score dựa trên biến empScore.

* **public class Eval**

Xây dựng một danh sách lưu điểm số và vị trí của các ô trống được xét.

* private int xPos, yPos;
* private List<Score> arrMatrix;

Tạo danh sách class Score lưu vị trí và điểm số.

* public Eval();

Hàm tạo danh sách arrMatrix.

* public Point MaxPosition();

Hàm tìm kiếm vị trí có điểm số cao nhất trong danh sách arrMatrix. Dùng thuật toán sắp xếp (Sort) từ lớn đến bé theo hàm so sánh đã viết ở trên, lấy vị trí của phần tử đầu tiên.

* **public class Board**
* private Panel chessBoard;
* private List<Player> player;

Danh sách người chơi

* private int curPlayer;

Biến curPlayer lưu giá trị “0” là lượt người chơi hiện tại, “1” là người chơi tiếp theo.

* private List<List<Button>> matrix;

Danh sách lồng danh sách với phần tử là Button tạo thành bàn cờ trò chơi Caro.

* private Stack<Info> curMove;

Stack (LIFO) dùng để lưu các nước đã đi, sử dụng cho chức năng Undo.

* private Stack<Point> isEndButton;

Stack để lưu các Button khi trò chơi kết thúc, đổi màu nền Button để thông báo cho người chơi biết thắng hay thua tại vị trí nào.

* private int modeGame;

Biến lưu giá trị kiểu chơi, giá trị “0” là người chơi với người, “1” là máy bình thường, “2” là máy khó và “3” là chức năng xem máy đánh với máy.

* private List<int> comAttack = new List<int> { 0, 2, 89, 768, 2607, 12867 };
* private List<int> comDefend = new List<int> { 0, 4, 49, 515, 1771, 8769 };

Hai danh sách điểm tấn công và phòng thủ, số điểm cộng tương ứng với số quân bản thân hoặc đối phương theo hướng ngang, dọc và chéo của ô trống đang xét.

Mặc định của trò chơi được thiết lập người chơi 1 là “O”, người chơi 2 là “X”. Không cố định người chơi nào sẽ là máy mà phụ thuộc vào chế độ chơi của trò chơi.

* public void boardDraw();

Hàm khởi tạo bàn cờ, danh sách bảng điểm, Stack lưu các nước đã đi,...

* public void btn\_Click();

Hàm xử lý sự kiện click vào Button trên bàn cờ.

* public Point getChessPoint(Button btn);

Hàm lấy vị trí của Button được xét.

* public bool isEnd();
* private bool isEndHor(Button btn);
* private bool isEndVer(Button btn);
* private bool isEndPri(Button btn);
* private bool isEndSub(Button btn);

Hàm xử lý kết quả của trò chơi, nếu đã kết thúc trả về giá trị “True”, nếu chưa kết thúc trả về giá trị “False”.

* public bool Undo();

Hàm xử lý chức năng hồi bước đi của người chơi.

### Thuật toán AI

* **Hàm lượng giá Heuristic**
* public Point comBestMove(int CurPlayer);

Vét cạn các ô trống chưa được đánh trên bàn cờ, lưu giá trị điểm và vị trí của ô trống đó vào danh sách điểm, trả về vị trí có điểm số cao nhất.

* private int comAttHor(Button btn);
* private int comAttVer(Button btn);
* private int comAttPri(Button btn);
* private int comAttSub(Button btn);
* private int comDefHor(Button btn);
* private int comDefVer(Button btn);
* private int comDefPri(Button btn);
* private int comDefSub(Button btn);

Với mỗi Button trống được xét theo 4 hướng là ngang, dọc và hai đường chéo, đếm số quân cờ mỗi bên và tiến hành tính điểm dựa trên những ô xung quanh là phe mình hay phe đối phương theo danh sách điểm tấn công hoặc phòng thủ.

Tiến hành quét các ô xung quanh ô trống đang được xét và đếm số lượng quân mình và quân đối phương theo một hướng. Nếu số quân đối phương là 2, tức là đã bị chặn hai đầu, trả giá trị của ô trống đó bằng 0. Nếu số quân của ta bằng 4, tức chỉ cần 1 nước đi nữa sẽ chiến thắng, điểm số của ô trống đó sẽ được \*5. Trả về kết quả điểm số bằng số điểm tấn công trừ số điểm phòng thủ dựa trên số quân theo bảng điểm ở trên. Tương tự như với hàm tính điểm phòng thủ, tổng điểm sẽ bằng số điểm phòng thủ trừ số điểm tấn công.

Tổng số điểm của ô trống được xác định bằng cách tính tổng số điểm tấn công hoặc phòng thủ của cả 4 hướng và so sánh. Nếu số điểm tấn công lớn hơn điểm phòng thủ thì sẽ lựa chọn nước đi tấn công, và ngược lại.

* **Thuật toán Minimax, giải thuật Alpha - Beta**

Đầu tiên đánh giá bàn cờ và chọn ra 3 ô trống có điểm cao nhất, tiến hành đánh thử. Trong mỗi lượt đánh thử cũng tiến hành đánh giá bàn cờ và chọn ra 3 ô trống cao điểm nhất của người để đánh trả lại. Độ sâu tối đa là 11. Nếu tìm thấy nước đi dẫn tới chiến thắng thì đánh theo nước đó, không thì đánh vào ô có điểm cao nhất.

* private int depth = 0;

Độ sâu hiện tại đang xét, qua mỗi lượt xét thì độ sâu tăng 1.

* private Point[] ourMoves = new Point[Value.maxMove + 2];
* private Point[] eneMoves = new Point[Value.maxMove + 2];

Mảng lưu vị trí các nước đi của quân mình và quân đối phương.

* private Point[] winMove = new Point[Value.maxDepth + 2];
* private Point[] loseMove = new Point[Value.maxDepth + 2];

Mảng lưu vị trí các nước đi dẫn đến kết quả thắng hoặc thua của nước đi đang xét.

* public void findMove();

Hàm tìm kiếm nước đi và tiến hành đánh thử các ô trống có số điểm cao được lưu.

* public void comDepthTurn();

Hàm đánh nước đi của máy, đánh vào vị trí ô trống dẫn đến kết quả thắng nếu có, hoặc đánh vào ô có điểm số cao nhất theo hàm lượng giá Heuristic.

Trò chơi có 2 mức độ khó ứng với mỗi thuật toán khác nhau. Độ khó bình thường chỉ đánh những ô có điểm số cao nhất. Độ khó khó hơn máy sử dụng thuật toán Minimax và Alpha - Beta để tìm nước đi tối ưu hơn, và hàm lượng giá có một chút cải tiến hơn để đánh giá ô trống đang xét có tốt hay không.

# KẾT LUẬN

## Đánh giá

Trò chơi Caro đơn giản với những cải tiến như chức năng xem máy đánh với máy, cải thiện thuật toán nếu bị chặn 2 đầu sẽ không kết thúc trò chơi, kết hợp các thuật toán AI xử lý tính toán nước đi phù hợp cho máy theo nhiều mức độ.

Hiện nay những trò chơi mang tính đối kháng như Caro đã có rất nhiều lập trình viên tạo nên và có những thuật toán tối ưu hơn, trò chơi của em chỉ mới đạt được một số điều kiện cơ bản nhất định và chắc chắn tồn tại đâu đó một vài lỗi mà em chưa tìm ra, hoặc chưa thực hiện được.

Qua môn học Trí tuệ nhân tạo, và đặc biệt là qua đồ án trò chơi Caro này, em học được rất nhiều kiến thức mới. Không những cải thiện được kỹ năng lập trình ngôn ngữ C#, mà em còn tìm hiểu được rất nhiều kỹ thuật mới. Làm việc với Visual Studio tốt hơn, học thêm về Winform *(mặc dù môn này em chưa được học)*, hiểu rõ hơn về các thuật toán Trí tuệ nhân tạo,...

## Nhược điểm

* Để có thể tìm kiếm được ô trống có điểm số cao nhất, vẫn phải sử dụng phương pháp vét cạn từng ô trống.
* Thuật toán Heuristic vẫn chưa xét được những trường hợp có lợi thế hơn.
* Thời gian xử lý tạo bàn cờ bằng danh sách các nút (Button) còn chậm.
* Vẫn còn một số lỗi tồn đọng trong quá trình chơi.

## Hướng phát triển

* Hoàn thiện các chức năng cơ bản còn thiếu.
* Kiểm và rà soát, sửa các lỗi còn tồn đọng trong quá trình chơi.
* Phát triển thuật toán tính điểm đường đi tối ưu hơn nữa.
* Hoàn thiện chức năng “Máy siêu khó”.
* Thiết kế giao diện trò chơi thân thiện với người dùng hơn.
* Tích hợp các công nghệ mới nhất cho trò chơi.

# TÀI LIỆU THAM KHẢO

## Tham khảo lý thuyết

* https://vi.wikipedia.org/wiki/Trí\_tuệ\_nhân\_tạo
* https://vi.wikipedia.org/wiki/Heuristic
* https://vi.wikipedia.org/wiki/Minimax
* https://www.stdio.vn/articles/giai-thuat-cat-tia-alpha-beta-564

## Tham khảo code

* https://stackoverflow.com/questions/tagged/gomoku
* https://www.codeproject.com
* https://www.howkteam.vn/course/lap-trinh-game-caro-voi-c-winform-14
* http://diendan.congdongcviet.com/forum.php
* https://codervietnam.com/2017/09/28/tai-lieu-tri-tue-nhan-tao-thuat-toan-min-max-game-caro-c/
* https://codervietnam.com/2017/09/28/game-caro-c-co-xu-ly-ai-nang-cao/
* https://www.howkteam.vn/learn/lap-trinh/lap-trinh-c-net-7-5
* https://myclass.vn/lap-trinh-c-nang-cao-qua-lap-trinh-game-caro-va-ung-dung-paint-10172.html