[Chương 1. MỞ ĐẦU 3](#_Toc27051193)

[1.1. Giới thiệu đề tài 3](#_Toc27051194)

[1.2. Mục tiêu cần đạt được 3](#_Toc27051195)

[1.3. Đối tượng và phạm vi nghiên cứu 4](#_Toc27051196)

[1.4. Phương pháp nghiên cứu 4](#_Toc27051197)

[Chương 2. STACK VÀ SỬ DỤNG TRONG GAME CARO 5](#_Toc27051198)

[2.1. Tìm hiểu về Stack 5](#_Toc27051199)

[2.1.1. Biểu diễn cấu trúc dữ liệu ngăn xếp Stack 5](#_Toc27051200)

[2.1.2. Ví dụ minh họa: 5](#_Toc27051201)

[2.2. Các hoạt động của Stack 6](#_Toc27051202)

[2.2.1. Hoạt động cơ bản 7](#_Toc27051203)

[2.2.2. Các hàm hỗ trợ khác 10](#_Toc27051204)

[2.3. Sử dụng Stack trong việc cài đặt game CARO 12](#_Toc27051205)

[2.3.1. Vấn đề: 12](#_Toc27051206)

[2.3.2. Giải pháp: 12](#_Toc27051207)

[2.3.3. Phương pháp cài đặt chức năng Undo bằng Stack: 12](#_Toc27051208)

[Chương 3. XÂY DỰNG GAME CARO HAI NGƯỜI CHƠI 13](#_Toc27051209)

[3.1. Giới thiệu chương trình 13](#_Toc27051210)

[3.2. Chạy chương trình 13](#_Toc27051211)

[3.3. Những vấn đề gặp phải và cách giải quyết: 16](#_Toc27051212)

[3.4. Một số đoạn code quan trọng trong chương trình 17](#_Toc27051213)

[3.5. Tổng kết 21](#_Toc27051214)

[LỜI KẾT 23](#_Toc27051215)

[TÀI LIỆU THAM KHẢO 23](#_Toc27051216)

**DANH MỤC HÌNH ẢNH**

[*Hình 2. 1 Cấu trúc dữ liệu ngăn xếp Stack 5*](#_Toc26885894)

[*Hình 2. 2 Hoạt động PUSH của cấu trúc dữ liệu ngăn xếp 8*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885895)

[*Hình 2. 3 Giải thuật cho hoạt động PUSH 8*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885896)

[*Hình 2. 4 Hoạt động POP của cấu trúc dữ liệu ngăn xếp 9*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885897)

[*Hình 2. 5 Giải thuật cho hoạt động POP 10*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885898)

[*Hình 2. 6 Giải thuật của hàm PEEK 10*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885899)

[*Hình 2. 7 Giải thuật của hàm isFULL 11*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885900)

[*Hình 2. 8 Giải thuật của hàm isEMPTY 11*](file:///C:\Users\Admin\Desktop\MỞ%20ĐẦU.docx#_Toc26885901)

[Hình 3. 1 Dao diện chính của chương trình 13](file:///C:\Users\Admin\Desktop\New%20folder\MỞ%20ĐẦU.docx#_Toc27045900)

[Hình 3. 2 Lượt đánh của Player X 14](file:///C:\Users\Admin\Desktop\New%20folder\MỞ%20ĐẦU.docx#_Toc27045901)

[Hình 3. 3 Lượt đánh của Player O 14](file:///C:\Users\Admin\Desktop\New%20folder\MỞ%20ĐẦU.docx#_Toc27045902)

[Hình 3. 4 Thông báo chiến thắng và kết thúc trò chơi 15](file:///C:\Users\Admin\Desktop\New%20folder\MỞ%20ĐẦU.docx#_Toc27045903)

[Hình 3. 5 Khóa bàn cờ 15](file:///C:\Users\Admin\Desktop\New%20folder\MỞ%20ĐẦU.docx#_Toc27045904)

# MỞ ĐẦU

## Giới thiệu đề tài

Tên đề tài: Cài đặt game CARO 2 người chơi có sử dụng Stack

CARO là một trò chơi không quá xa lạ với mọi người trong cuộc sống. CARO bao gồm một bàn cờ hình chữ nhật, kích thước có thể giới hạn hoặc không. Nhiệm vụ của mỗi người chơi là đánh dấu vào các ô cờ theo lượt. Người chiến thắng là người đầu tiên có 5 hặc hơn 5 quân thuộc sở hữu của mình xếp thành 1 hàng thẳng liên tục (ngang, dọc hoặc chéo).

Stack (ngăn xếp) là một cấu trúc dữ liệu trừu tượng (Abstract Data Type – viết tắt là ADT), hầu như được sử dụng trong hầu hết mọi ngôn ngữ lập trình. Đặt tên là ngăn xếp bởi vì nó hoạt động như một ngăn xếp trong đời sống thực, ví dụ như một cỗ bài hay một chồng đĩa, …

Trong đời sống thực, ngăn xếp chỉ cho phép các hoạt động tại vị trí trên cùng của ngăn xếp. Ví dụ, chúng ta chỉ có thể đặt hoặc thêm một lá bài hay một cái đĩa vào trên cùng của ngăn xếp. Do đó, cấu trúc dữ liệu trừu tượng ngăn xếp chỉ cho phép các thao tác dữ liệu tại vị trí trên cùng. Tại bất cứ thời điểm nào, chúng ta chỉ có thể truy cập phần tử trên cùng của ngăn xếp.

Đặc điểm này làm cho ngăn xếp trở thành cấu trúc dữ liệu dạng LIFO (Last-In-First-Out). Ở đây, phần tử được đặt vào (được chèn, được thêm vào) cuối cùng sẽ được truy cập đầu tiên. Trong thuật ngữ ngăn xếp, hoạt động chèn được gọi là hoạt động PUSH và hoạt động xóa được gọi là hoạt động POP.

Chính nhờ đặc điểm này mà stack có khá nhiều ứng dụng đa dạng như đảo ngược xâu, đổi cơ số, tính toán các biểu thức đại số, thao tác Undo hay Redo thường gặp, …. Đề tài “Cài đặt game CARO 1 người chơi có sử dụng stack” nhằm giúp tìm hiểu sâu hơn về cấu trúc dữ liệu Stack (ngăn xếp) và tìm hiểu vai trò của nó trong việc cài đặt game CARO hai người chơi cụ thể là chức năng đi lại (undo) trong game.

## Mục tiêu cần đạt được

* Cài đặt game cờ CARO 2 người chơi hoàn chỉnh với giao diện đơn giản dễ sử dụng.
* Xây dựng dược cấu trúc dữ liệu Stack gồn các thao tác cần thiết.
* Sử dụng Stack vừa tạo để làm chứac năng Undo trong game CARO.
* Tạo được các chức năng cần thiết khác cho chương trình.

## Đối tượng và phạm vi nghiên cứu

* Ngôn ngữ C# và Winform của ngôn ngữ C#
* Stack và hàm của Stack

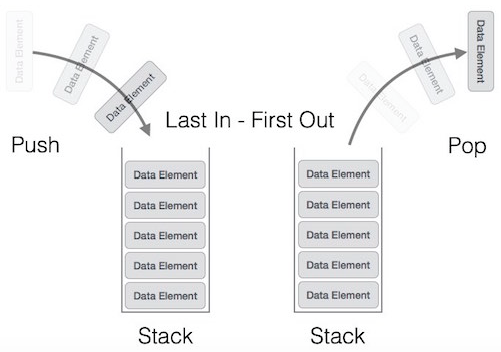
## Phương pháp nghiên cứu

* Chọn đề tài nghiên cứu vì đề tài gần gũi với mọi người và là một ví dụ điển hình để tìm hiểu về cấu trúc dữ liệu Stack.
* Kế hoạch: Phân chia phù hợp với từng thành viên và tự nguyện. Tiến độ luôn ưu tiên theo kế hoạch đề ra. Các thành viên luôn sẵn sàng giúp đỡ lần nhau trong việc tìm kiếm và thực hiện.
* Tài liệu được lấy từ các giáo trình có sẵn và lấy từ các trang mạng, bài giảng trên internet. Tổng hợp tài liệu là việc quan trọng giúp chọn lọc tài kiệu cần thiết và lược bỏ phần dư thừa.
* Có sự giúp đỡ từ bạn bè và tham khảo ý kiến của giảng viên.

# STACK VÀ SỬ DỤNG TRONG GAME CARO

## Tìm hiểu về Stack

### Biểu diễn cấu trúc dữ liệu ngăn xếp Stack



Hình 2. 1 Cấu trúc dữ liệu ngăn xếp Stack

Một ngăn xếp (Stack) là một cấu trúc dữ liệu dạng thùng chứa của các phần tử và có hai phép toán cơ bản Push and Pop. Push bổ sung một phần tử vào đỉnh của ngăn xếp, nghĩa là đứng sau các phần tử đã có trong ngăn xếp. Pop giải phóng phần tử đang đứng ở đỉnh của ngăn xếp. Trong Stack, các đối tượng có thể được thêm vào Stack bất kỳ lúc nào nhưng chỉ có đối tượng thêm vào sau cùng mới được phép lấy ra khỏi Stack.

### Ví dụ minh họa:

Chúng ta lấy ví dụ đơn giản về ứng dụng của Stack đó là tính giá trị của biểu thức đại số.

Ví dụ: Tính biểu thức

((1 + 2) \* 4) + 3

Trước hết viết biểu thức theo [hậu thứ tự](https://vi.wikipedia.org/wiki/K%C3%BD_ph%C3%A1p_Ba_Lan) : 1 2 + 4 \* 3 +

Khi tính một biểu thức ta dùng một ngăn xếp để cất dữ liệu trung gian như sau: Đọc từ trái sang phải mỗi lần một phần tử:

* Nếu gặp một toán hạng (biến hoặc hằng), đặt (push)toán hạng đó vào ngăn xếp
* Nếu gặp một toán tử, lấy (pop) hai phần tử trên đỉnh ngăn xếp ra, thực hiện phép toán và đặt (push) lại kết quả vào ngăn xếp.

|  |  |  |
| --- | --- | --- |
| **Input** | **Phép toán** | **Ngăn xếp** |
| 1 | Push(1) | 1 |
| 2 | Push(2) | 1, 2 |
| + | Pop(2), Pop(1), 1+2=3; Push(3) | 3 |
| 4 | Push(4) | 3, 4 |
| \* | Pop(4), Pop(3), 3\*4=12; Push(12) | 12 |
| 3 | Push(3) | 12, 3 |
| + | Pop(3), Pop(12), 12+3=15; Push(15) | 15 |

Kết quả 15 đặt ở đỉnh của ngăn xếp khi tính xong.

## Các hoạt động của Stack

Một ngăn xếp có thể được triển khai theo phương thức của Mảng (Array), Cấu trúc (Struct), Con trỏ (Pointer) và Danh sách liên kết (Linked List)**.** Ngăn xếp có thể là ở dạng kích cỡ cố định hoặc ngăn xếp có thể thay đổi kích cỡ. Phần dưới chúng ta sẽ triển khai ngăn xếp bởi sử dụng các mảng với việc triển khai các ngăn xếp cố định.

Các hoạt động cơ bản trên ngăn xếp có thể liên quan tới việc khởi tạo ngăn xếp, sử dụng nó và sau đó xóa nó. Ngoài các hoạt động cơ bản này, một ngăn xếp có hai hoạt động nguyên sơ liên quan tới khái niệm, đó là:

* Hoạt động Push(): lưu giữ một phần tử trên ngăn xếp.
* Hoạt động Pop(): xóa một phần tử từ ngăn xếp.

Khi dữ liệu đã được PUSH lên trên ngăn xếp:

Để sử dụng ngăn xếp một cách hiệu quả, chúng ta cũng cần kiểm tra trạng thái của ngăn xếp. Để phục vụ cho mục đích này, dưới đây là một số tính năng hỗ trợ khác của ngăn xếp:

* Hoạt động Peek(): lấy phần tử dữ liệu ở trên cùng của ngăn xếp, mà không xóa phần tử này.
* Hoạt động isFull(): kiểm tra xem ngăn xếp đã đầy hay chưa.
* Hoạt động isEmpty(): kiểm tra xem ngăn xếp là trống hay không.

Tại mọi thời điểm, chúng ta duy trì một con trỏ tới phần tử dữ liệu vừa được PUSH cuối cùng vào trên ngăn xếp. Vì con trỏ này luôn biểu diễn vị trí trên cùng của ngăn xếp vì thế được đặt tên là top. Con trỏ top cung cấp cho chúng ta giá trị của phần tử trên cùng của ngăn xếp mà không cần phải thực hiện hoạt động xóa ở trên (hoạt động pop).

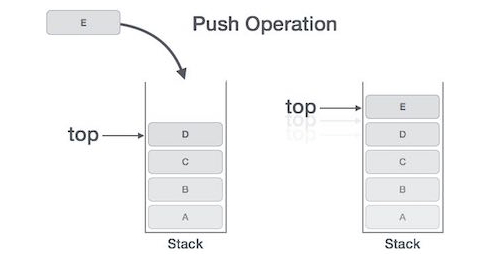
Sau dây chúng ta sẽ tìm hiểu sâu về các hoạt động trên (giải thuật dữa trên ngôn ngữ C# và xây dựng Stack bằng Array[]):

### Hoạt động cơ bản

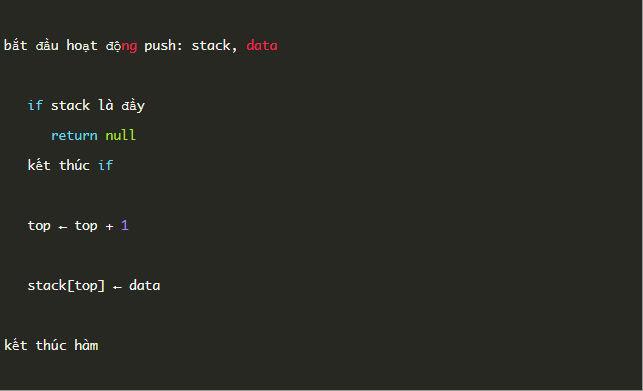
Hoạt động PUSH của cấu trúc dữ liệu ngăn xếp

Tiến trình đặt (thêm) một phần tử dữ liệu mới vào trên ngăn xếp còn được biết đến với tên Hoạt động PUSH. Hoạt động PUSH bao gồm các bước sau:

* Bước 1: kiểm tra xem ngăn xếp đã đầy hay chưa.
* Bước 2: nếu ngăn xếp là đầy, tiến trình bị lỗi và thoát ra.
* Bước 3: nếu ngăn xếp chưa đầy, tăng top để trỏ tới phần bộ nhớ trống tiếp theo.
* Bước 4: thêm phần tử dữ liệu vào vị trí nơi mà top đang trỏ đến trên ngăn xếp.
* Bước 5: trả về success.

Nếu Danh sách liên kết được sử dụng để triển khai ngăn xếp, thì ở bước 3 chúng ta cần cấp phát một không gian động.

Hình 2. 2 Hoạt động PUSH của cấu trúc dữ liệu ngăn xếp

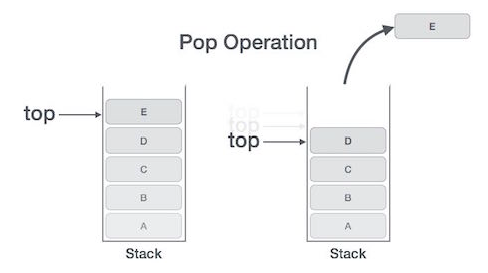
Giải thuật cho hoạt động PUSH:

Hình 2. 3 Giải thuật cho hoạt động PUSH

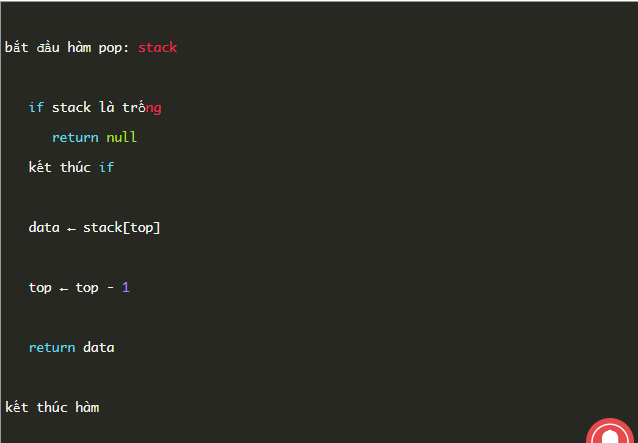
Hoạt động POP của cấu trúc dữ liệu ngăn xếp

Việc truy cập nội dung phần tử trong khi xóa nó từ ngăn xếp còn được gọi là Hoạt động POP. Trong sự triển khai Mảng của hoạt động Pop(), phần tử dữ liệu không thực sự bị xóa, thay vào đó top sẽ bị giảm về vị trí thấp hơn trong ngăn xếp để trỏ tới giá trị tiếp theo. Nhưng trong sự triển khai Danh sách liên kết, hoạt động Pop() thực sự xóa phần tử xữ liệu và xóa nó khỏi không gian bộ nhớ.

Hoạt động POP có thể bao gồm các bước sau:

* Bước 1: kiểm tra xem ngăn xếp là trống hay không.
* Bước 2: nếu ngăn xếp là trống, tiến trình bị lỗi và thoát ra.
* Bước 3: nếu ngăn xếp là không trống, truy cập phần tử dữ liệu tại top đang trỏ tới.
* Bước 4: giảm giá trị của top đi 1.
* Bước 5: trả về success.

Hình 2. 4 Hoạt động POP của cấu trúc dữ liệu ngăn xếp

Giải thuật cho hoạt động POP

Hình 2. 5 Giải thuật cho hoạt động POP

### Các hàm hỗ trợ khác

Giải thuật của hàm peek(): 

Hình 2. 6 Giải thuật của hàm PEEK

Giải thuật của hàm isFull():

Hình 2. 7 Giải thuật của hàm isFULL

Giải thuật của hàm isEmpty():

Hình 2. 8 Giải thuật của hàm isEMPTY

## Sử dụng Stack trong việc cài đặt game CARO

### Vấn đề:

Nếu một người chơi đánh dấu nhầm vị trí của quân cờ của mình hay người chơi đó muốn đánh lại một nước cờ tại thời điểm bất kì nào đó thì việc đi lại sẽ được thực hiện.

### Giải pháp:

Để thực hiện điều đó thì ghi nhớ lại thứ tự các nước đã đánh và vị trí của chúng quả là một việc khó khăn nếu có quá nhiều nước đi đã được thực hiện. Việc đó là một minh chứng cho chức năng Undo có vai trò quan trọng trong đề tài này.

Yêu cầu thông tin vị trí ô cờ cho chức năng Undo phải chính xác theo thứ tự gần nhất đến xa thời điểm hiện tại nhất phù hợp với dạng Last-In-First-Out và thông tin phải được lấy ra một cách thuận tiện nhất.

Stack đáp ứng được nhu cầu của chức năng Undo là Last-In-First-Out và thông tin của phần tử cuối cùng luôn được lấy ra đầu tiên một cách thuận lợi nhất. Như vậy dây là giải pháp phù hợp cho vấn đề này.

### Phương pháp cài đặt chức năng Undo bằng Stack:

* Tổ chức bàn cờ dưới dạng ma trận, mỗi ô cờ sẽ có một tọa độ riêng. Ô cờ sẽ lưu thông tin người chơi sở hữu quân cờ được đánh vào nó.
* Tạo một Stack rỗng (đề xuất sử dụng List< >) để lưu lại vị trí ô cờ mỗi khi chúng được đánh. Ưu điểm của List là không giới hạn số phần tử trong Stack.
* Mỗi khi có một quân cờ được đánh vào, chương trình sẽ bỏ thông tin vị trí của nó vào phần tử top của Stack => Push(data) và tiến hành kiểm tra chiến thắng nếu có thì kết thúc trò chơi.
* Mỗi lần chức năng Undo được sử dụng:
  + Thông tin của phần tử top sẽ được sử dụng để tìm vị trí quân cờ được đánh gần thời điểm hiện tại nhất và xóa thông tin nước đi cuối cùng => Pop(top).
  + Thực hiện thao tác xóa thông tin của quân cờ tại vị trí ô cờ đó.
  + Đổi thông tin người chơi hiện tại.

# XÂY DỰNG GAME CARO HAI NGƯỜI CHƠI

## Giới thiệu chương trình

Hình 3. 1 Dao diện chính của chương trình

- Tên chương trình: CARO

- Thông tin người chơi và quân cờ của họ.

- Bàn cờ có kích thước 20x20

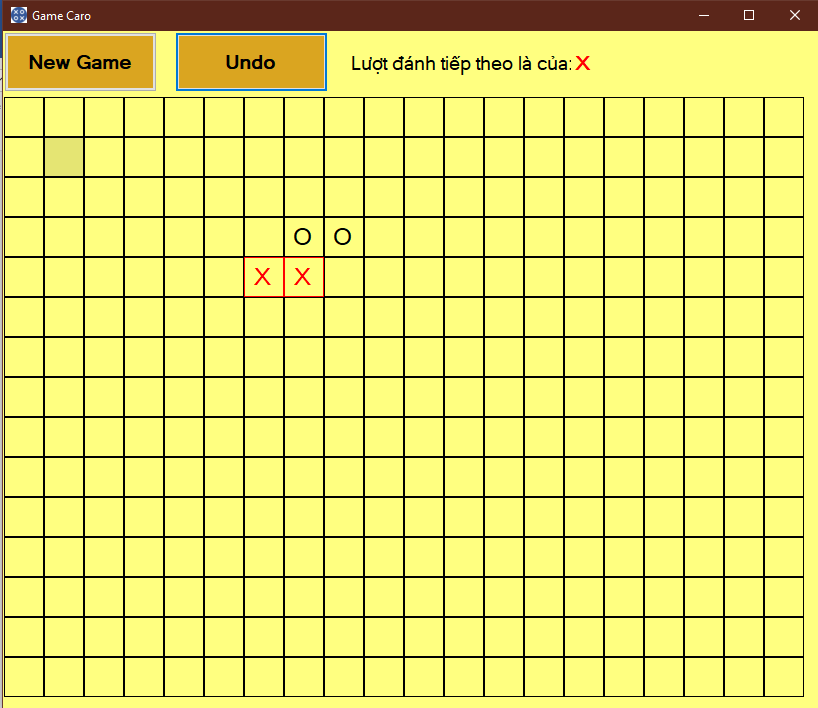
- Các chức năng:

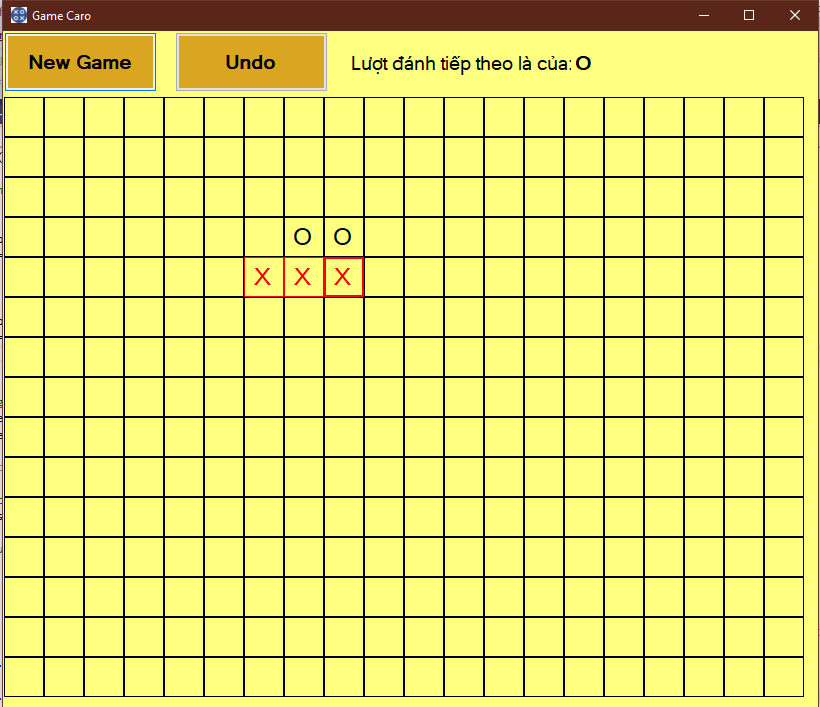
* New Game : Tạo mới trò chơi (Ctrl + N)
* Undo : Đi lại (Ctrl + Z)
* Quit : Thoát khỏi chương trình (Ctrl + Q)

## Chạy chương trình

- Người chơi thứ nhất Player X sẽ có quân cờ là X sẽ đánh trước.

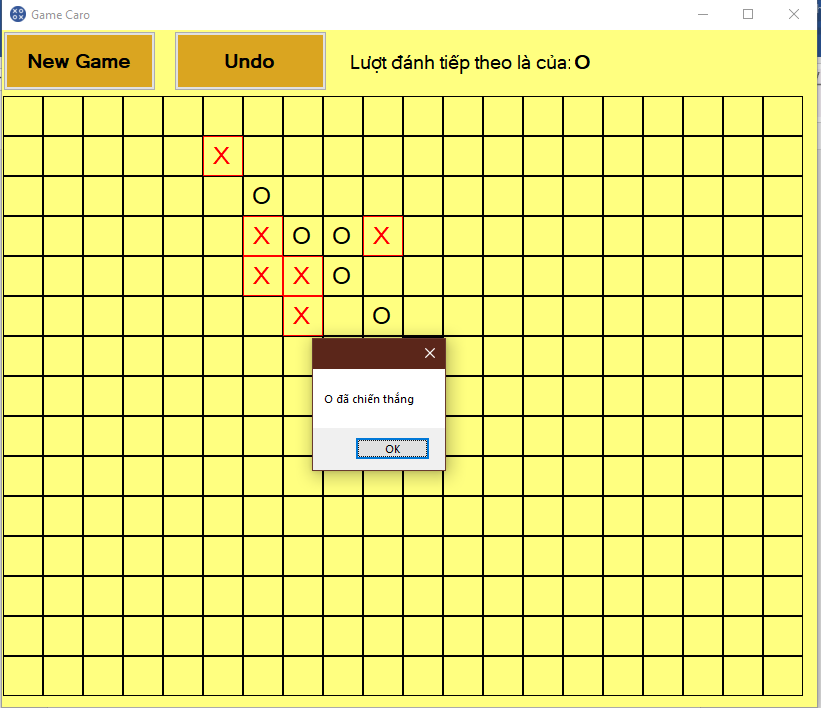
- Người chơi thứ hai Player O sẽ có quân cờ là O sẽ đánh sau.

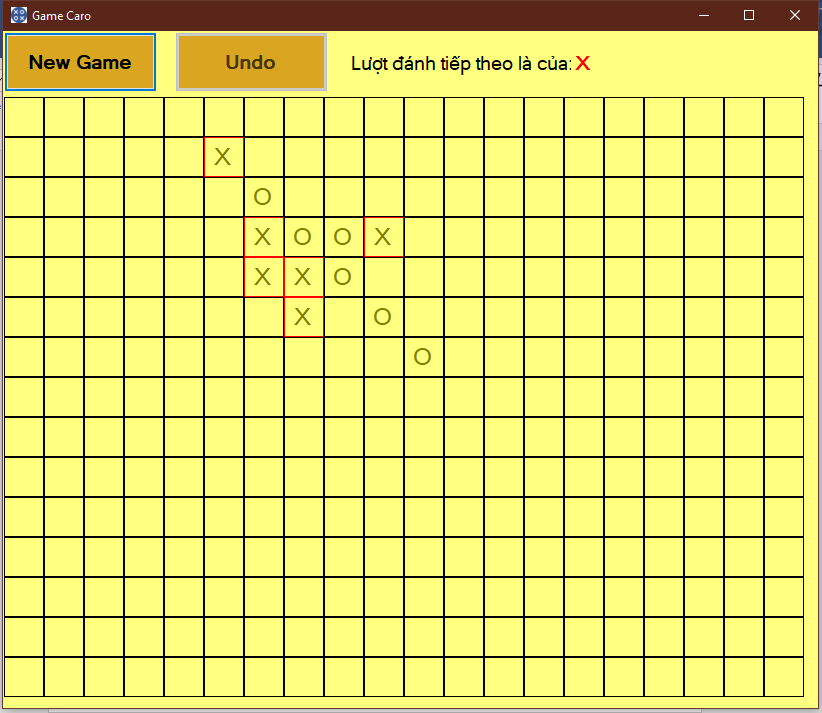
- Lượt đánh của người chơi nào thì màn hình sẽ hiển thị tên và quân cờ của người chơi đó, thông tin sẽ được đổi lại mỗi khi có người chơi đánh xong nước cờ.

**- Chương trình sẽ tự động xét chiến thắng bằng thuật toán có sẵn.

Hình 3. 2 Lượt đánh của Player X

Hình 3. 3 Lượt đánh của Player O

- Khi có người chơi chiến thắng chương trình sẽ tự động thông báo người chơi chiến thắng, kết thúc trò chơi và khóa bàn cờ.



Hình 3. 4 Thông báo chiến thắng và kết thúc trò chơi

Hình 3. 5 Khóa bàn cờ

## Những vấn đề gặp phải và cách giải quyết:

Tổ chức bàn cờ:

* Sử dụng dao diện Winfrom ngôn ngữ C# trên Visual Studio
* Bàn cờ là một ma trận gồm các nút (button).
* Các button sẽ là nơi đánh cờ và sẽ lưu thông tin về quân cờ trên nó bằng cách tạo event ấn cho button.

Vị trí các button:

* Được xác định như các phần tử trong ma trận bởi tọa độ riêng mỗi button.
* Stack sẽ lưu tọa độ của các button được đánh vào.

Xét chiến thắng:

* Mỗi một nước đi được thực hiên xong thì sẽ kiểm tra chiến thắng cho tất cả các nước đi đã có trên bàn cờ.
* Xét chiến thắng trên hàng ngang, dọc, cả đường chéo chính và đường chéo phụ.
* Khóa bàn cờ khi đã chiến thắng và kết thúc trò chơi

Cách xét chiến thắng:

Với hàng ngang lần lượt xét bên trái và bên phải của quân cờ

B1: Kiểm tra sở hữu quân cờ, đặt biến dem =1 tính cả quân vừa đánh

B2: Kiểm tra ô bên trái có sở hữu quân cờ đó hay không. Nếu có tăng dem lên 1 đơn nếu dem == 5thì qua B4 nếu ko tiếp tục xét về bên trái cho đến khi hết tính liên tục hay đến ô ngoài cùng của bàn cờ

B3: Kiểm tra ô bên phải có sở hữu quân cờ đó hay không. Nếu có tăng dem lên 1 đơn nếu dem == 5thì qua B4 nếu ko tiếp tục xét về bên phải cho đến khi hết tính liên tục hay đến ô ngoài cùng của bàn cờ

B4: Thông báo người chơi chiến thắng và kết thúc trò chơi.

Tượng tự với hàng dọc, đường chéo chính và đường chéo phụ.

Lưu ý:

* Lưu ý khi xét phải có điều kiện để giới hạn ô ngoài cùng của bản cờ.

Hòa:

* Ván cờ có thể kết thúc với kết quả hòa khi chưa có người chơi nào đạt đủ điều kiện chiến thắng và bàn cờ đã kín.
* Thực hiện thông báo kết quả và khóa bàn cờ.

Undo:

* Không thể Undo khi chưa đánh cờ và khi hết nước cờ trong Stack cho nên phải kiểm tra số phần tử có trong Stack trước khi Undo.
* Nên tạo Stack để Undo bằng List<> vì không giới hạn phần tử có trong Stack.

## Một số đoạn code quan trọng trong chương trình

|  |
| --- |
| // vitridong: Là vị trí dòng của quân vừa đánh xuống  // vitricot: Là vị trí cột của quân vừa đánh xuống  bool KiemTraHangNgang(Button[,] a, int socot, int vitridong, int vitricot)  {  int dem = 1; // Tính luôn quân vừa đánh xuống  // Xét qua bên trái trước (ngang trái)  for (int j = vitricot - 1; j >= 0; --j)  {  if (a[vitridong, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5)  {  return true; // đã thắng  }  }  else break; // chỉ cần phát hiện không còn tính liên tục => dừng quá trình lặp lại  }  // Xét qua bên phải (ngang phải)  for (int j = vitricot + 1; j < socot; ++j)  {  if (a[vitridong, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true; // đã thắng  }  else break;  }  return false;  }  bool KiemTraHangDoc(Button[,] a, int sodong, int vitridong, int vitricot)  {  int dem = 1;  for(int i = vitridong - 1; i >= 0; --i)  {  if(a[i, vitricot].Text == a[vitridong, vitricot].Text)  {  dem++;  if(dem == 5) return true;  }  else break;  }  // Xét xuống dưới (dọc dưới)  for (int i = vitridong + 1; i < sodong; ++i)  {  if (a[i, vitricot].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true;  }  else break;  }  return false;  }  bool KiemTraCheoChinh(Button[,] a, int sodong, int socot, int vitridong, int vitricot)  {  int dem = 1;  // Kiểm tra chéo chính trên: Các phần tử dòng - cột luôn giảm 1 đơn vị và giảm đến khi nào 1 trong 2 thằng xuất hiện 0  int i = vitridong - 1;  int j = vitricot - 1;  while(true)  {  if(i < 0 || j < 0) break; // vị trí xét không hợp lệ  // Nếu còn chạy xuống đây được thì có nghĩa vị trí i, j hợp lệ  if (a[i, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true;  }  else break;  // Xét qua lần lặp mới  i--;  j--;  }  // Kiểm tra chéo chính dưới: Các phần tử dòng - cột đều tăng lên 1 đơn vị và tăng đến khi nào 1 trong 2 thằng bằng n - 1 thì dừng lại  i = vitridong + 1;  j = vitricot + 1;  while(true)  {  if (i == sodong || j == socot) break;  // Nếu còn chạy xuống đây được thì có nghĩa vị trí i, j hợp lệ  if (a[i, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true;  }  else break;  i++;  j++;  }  return false;  }  bool KiemTraCheoPhu(Button[,] a, int sodong, int socot, int vitridong, int vitricot)  {  int dem = 1;  // Kiểm tra chéo phụ trên: Dòng giảm - cột tăng. Dòng giảm tối đa tới 0, Cột tăng tối đa nới n - 1  int i = vitridong - 1;  int j = vitricot + 1;  while(true)  {  if (i < 0 || j == socot) break;  // Nếu còn chạy xuống đây được thì có nghĩa vị trí i, j hợp lệ  if (a[i, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true;  }  else break;  i--;  j++;  }  // Kiểm tra chéo phụ dưới: Dòng tăng - cột giảm. Dòng tăng tới tối đa là n - 1 và cột giảm tới tối đa là 0  i = vitridong + 1;  j = vitricot - 1;  while(true)  {  if (i == sodong || j < 0) break;  // Nếu còn chạy xuống đây được thì có nghĩa vị trí i, j hợp lệ  if (a[i, j].Text == a[vitridong, vitricot].Text)  {  dem++;  if (dem == 5) return true;  }  else break;  i++;  j--;  }  return false; // chưa thắng  } |

Hai đối thủ hòa nhau

|  |
| --- |
| // Kiểm tra xem là nước đánh đó có phải là nước đánh cuối cùng hay không?\  if (stack.COUNT() == sodong \* socot)  {  MessageBox.Show("Bàn cờ đã hết chỗ để đánh" + (banco[0, 0].Enabled == true ? ". 2 đấu thủ hòa nhau" : ""));  } |

Class Satck tự tạo để sử dụng (sử dụng List<data>)

|  |
| --- |
| class CustomStack<T>  {  public int top;  private List<T> stack;  public CustomStack()  {  stack = new List<T>();  top = -1;  }  public int COUNT() => stack.Count;  public bool ISEMPTY() => stack.Count == 0;  public void PUSH(T item)  {  stack.Add(item);  top++;  }  public T POP()  {  T item = stack[top];  stack.RemoveAt(top);  top--;  return item;  }  public T PEEK()  {  return stack[top];  }  public void CLEAR()  {  stack.Clear();  top = -1;  }  } |

Chức năng Undo sử dụng Stack:

|  |
| --- |
| private void btnUndo\_Click(object sender, EventArgs e)  {  //Kiểm tra số phần tử có trong Stack nếu rỗng thì không thể Undo  if (stack.COUNT() > 0)  {  ToaDo toadoquancovuadanh = (ToaDo)stack.POP(); // Lấy đối tượng đầu stack ra  banco[toadoquancovuadanh.toado\_dong, toadoquancovuadanh.toado\_cot].Text = ""; // Xóa quân cờ vừa đánh gần nhất  banco[toadoquancovuadanh.toado\_dong, toadoquancovuadanh.toado\_cot].ForeColor = Color.Black; // Reset lại về màu đen  luotdi = !luotdi; // Xác định lại lượt đi.    if (luotdi == true)  {  lblquanco.ForeColor = Color.Red; // Tô màu đỏ cho quân cờ được đánh ra  lblquanco.Text = "X";  }  else  {  lblquanco.ForeColor = Color.Black; // Tô màu đen cho quân cờ được đánh ra  lblquanco.Text = "O";  }  }  } |

## Tổng kết

Khó khăn:

* Winform trên C# là một điều khá mới mẻ đối với nhóm cho nên đây là trở ngại lớn khi phải vừ tìm hiểu cái mới và vừa tìm hiêu cách thực hiện. Thời gian tìm hiểu từ cơ bản của Winform tốn phần lớn thời gian của kế hoạch.
* Tìm kiếm tài liệu khá khó khăn vì phải chọn lọc lài liệu phù hợp và cách làm phù hợp nhất cho đề tài này. Phạm vi tìm kiếm trên internet là chủ yếu nên có rất nhiều sự lựa chon cần phải cân nhắc.
* Thời gian làm việc nhóm còn hạn chế, bất đồng ý kiếm ban đầu về việc chọn cách thực hiện đề tài.

Cách giải quyết:

* Dựa vào từng khả năng, điểm mạnh và sở thích của từng thành viên để phân công nhiệm vụ thích hợp cho họ.
* Mỗi thành viên sẽ tìm kiếm tài liệu cho từng phần đảm trách rồi chọn lọc ra những tài liệu cần thiết nhất.
* Tổng hợp lại thông tin rồi thống nhất để chọn tài liệu và hướng đi phù hợp cho đề tài.
* Truyền đạt lại kiến thức phần mình đảm trách cho thành viên còn lại để cả hai đề có thể hiểu được toàn bộ đề tài.
* Mỗi luần đề có trao đổi trên mạng và gặp mặt để trao đổi tiến độ hoành thành công việc.

Tự đánh giá kết quả:

* Công việc tìm kiếm tài liệu tốn thời gian vì có nhiều cách thực hiện đề tài khác nhau. Nhóm phải tìm hiểu từng cách và tìm ra cách tối ưu nhất và phù hợp nhất với đề tài và điều kiện hiện tại.
* Chương trình chỉ hạn chế ở một vài chức năng cơ bản và khá đơn giản.
* Giao diện đơn giản dễ sử dụng.
* Chưa có chức năng đánh dấu nước đi chiến thắng.
* Đánh giá bài làm : 7/10.

# LỜI KẾT

Đề tài “Cài dặt game CARO 2 người chơi sử dung Stack” là một ví dụ điển hình về cấu trúc dữ liệu Stack và ứng dụng của nó. Đề tài cung cấp một khía cạnh thú vị cho chúng ta về cấu trúc dữ liệu và cả lập trình dao diện bằng Winform. Bên cạnh đó còn cung cấp lối suy nghĩ mới mẻ về phong cách lập trình cho người thực hiện đề tài về cách làm bàn cờ, xét chiến thắng, ….

*Kết thúc đề tài, nhóm xin được chân thành cảm ơn sự hướng dẫn của giảng viên, sự giúp đỡ và tài liệu đến từ các nhóm khác đã giúp nhóm thúc hiện đề tài này.*

Nhóm thực hiện đề tài.

# TÀI LIỆU THAM KHẢO

1. **Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh – ĐH Sư Phạm Kỹ Thuật TP.HCM**
2. <https://www.howkteam.vn/course/lap-trinh-game-caro-voi-c-winform/demo-game-caro-lap-trinh-c-winform-111> - HowKteam.
3. [**https://vi.wikipedia.org/wiki/Ng%C4%83n\_x%E1%BA%BFp**](https://vi.wikipedia.org/wiki/Ng%C4%83n_x%E1%BA%BFp) **– Wikipedia.**
4. [**https://tuhocict.com/ngan-xep-stack-cau-truc-cai-dat-ung-dung-lop-stack-trong-c/**](https://tuhocict.com/ngan-xep-stack-cau-truc-cai-dat-ung-dung-lop-stack-trong-c/) **- Tự học ICT.**
5. [**https://sites.google.com/site/lophocproptit/dslk/5-xay-dung-stack-queue-bang-danh-sach-lien-ket**](https://sites.google.com/site/lophocproptit/dslk/5-xay-dung-stack-queue-bang-danh-sach-lien-ket) **- Lớp học ProPTIT.**
6. [**https://nguyenvanhieu.vn/ngan-xep-stack/**](https://nguyenvanhieu.vn/ngan-xep-stack/) **.**