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

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**ĐỀ TÀI:**

**LÀM GAME THÁP HÀ NỘI BẰNG STACK**

**Lớp: Sáng T6**

**Nguyễn Thành Công MSSV:18110086**

**Nguyễn Tấn Tài MSSV:18110193**

**GVHD: Thầy Trần Công Tú**

**TP. Hồ Chí Minh, tháng 11 năm 2019**

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

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



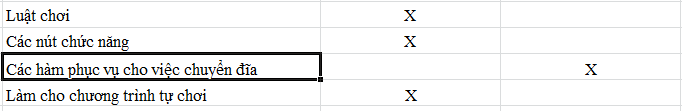
**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**ĐỀ TÀI:**

**LÀM GAME THÁP HÀ NỘI**

**Tài Công**

****

**GVHD: Thầy Trần Công Tú**

**TP. Hồ Chí Minh, tháng 11 năm 2019**

MỤC LỤC

[PHẦN 1: MỞ ĐẦU 5](#_Toc25778398)

[***1.*** ***Lý do chọn đề tài*** 5](#_Toc25778399)

[***2.*** ***Mục tiêu nhiệm vụ*** 6](#_Toc25778400)

[***3.*** ***Đối tượng nghiên cứu*** 6](#_Toc25778401)

[***4.*** ***Phương pháp nghiên cứu*** 6](#_Toc25778402)

[**5.** ***Cấu trúc báo cáo*** 6](#_Toc25778403)

[PHẦN 2: NỘI DUNG 7](#_Toc25778404)

[***1.*** ***Mô tả sản phẩm*** 7](#_Toc25778405)

[**1.1.** **Giới thiệu trò chơi** 7](#_Toc25778406)

[**1.2.** **Giới thiệu cấu trúc dữ liệu và giải thuật đã dùng** 8](#_Toc25778407)

[***2.*** ***Mô tả quá trình làm*** 9](#_Toc25778408)

[**2.1.** **Quá trình thiết kế form** 9](#_Toc25778409)

[**2.2.** **Quá trình cài đặt mã cho các nút** 10](#_Toc25778410)

[**2.2.1.** **Nút luật chơi** 10](#_Toc25778411)

[**2.2.2.** **Nút chịu thua** 10](#_Toc25778412)

[**2.3.** **Quá trình xử lý di chuyển** 11](#_Toc25778413)

[**2.3.1.** **Click vào cột muốn di chuyển** 11](#_Toc25778414)

[**2.3.2.** **Xử lý di chuyển đĩa giữa hai cột đã được nhấn** 12](#_Toc25778415)

[**2.3.3.** **Xử lý trong hàm chuyển đĩa( moveDisk)** 13](#_Toc25778416)

[**2.4.** **Quá trình khai báo các biến** 14](#_Toc25778417)

[**2.5.** **Quá trình hình thành các đĩa khi bắt đầu chơi** 15](#_Toc25778418)

[***3.*** ***Luồng chạy của chương trình*** 16](#_Toc25778419)

[***4.*** ***Bảng phân công công việc*** 17](#_Toc25778420)

[PHẦN 3: KẾT LUẬN 18](#_Toc25778421)

[Danh mục tài liệu tham khảo 20](#_Toc25778422)

[Hình 1: Sơ đồ tháp Hà Nội 7](#_Toc25785770)

[Hình 2: Cấu trúc dữ liệu stack 8](#_Toc25785771)

[Hình 3: Giao diện 10](#_Toc25785772)

[Hình 4; Mã của luật chơi 10](#_Toc25785773)

[Hình 5: Mã của nút chịu thua 11](#_Toc25785774)

[Hình 6: Mã khi nhấn vào cột 12](#_Toc25785775)

[Hình 7: Mã xử lý di chuyển 13](#_Toc25785776)

[Hình 8: Mã di chuyển 14](#_Toc25785777)

[Hình 9: Phần khai báo đầu tiên 14](#_Toc25785778)

[Hình 10: Cấp phát vùng nhớ 15](#_Toc25785779)

[Hình 11: Mã quá trình tạo đĩa 15](#_Toc25785780)

[Hình 12: Bảng phân công theo kế hoạch 17](#_Toc25785781)

# PHẦN 1: MỞ ĐẦU

1. ***Lý do chọn đề tài***

Hiện nay chúng ta đang sống trong thời đại mà công nghệ phát triển nhanh chóng, đó là cuộc cách mạng 4.0. Vì thế công nghệ thông tin theo đó cũng phát triển với tốc độ chóng mặt như vũ bão, kéo theo đó là sự ra đời của các thuật toán mới để đáp ứng yêu cầu của thế giới hiện nay. Tuy nhiên không vì vậy mà thế giới quên lãng những cấu trúc dữ liệu cơ bản như stack, queue… cùng với giải thuật đệ quy, nó chiếm 1 vị trí rất quan trọng trong ngành máy tính. Bởi vì những thứ cơ bản ấy cần cho việc giải quyết những bài toán phức tạp. Cũng như stack, queue và đệ quy, dù có tuổi thọ khá cao trong lĩnh vực này nhưng vị trí, vai trò của nó vẫn rất quan trọng . Nhờ có đệ quy mà nhiều bài toán phức tạp được giải quyết một cách dễ dàng hơn.

Chính vì thế mà ở môn cấu trúc dữ liệu và giải thuật của trường Sư phạm kỹ thuật, cấu trúc dữ liệu stack và giải thuật đệ quy chiếm vai trò rất quan trọng. Việc học nó chính là nền tảng cho các sinh viên hiểu rõ cách vận hành để tiếp xúc với các thuật toán khác. Một tòa nhà muốn xây cao thì ngoại trừ vật liệu dồi dào còn cần phải có 1 nền móng vững chắc.

Và để cho các sinh viên có thể tiếp thu những kiến thức đó một cách dễ dàng, hiệu quả, trành rơi vào trạng thái mơ hồ, khó hiểu (khi tiếp xúc với những kiến thức mới khó hình dung của sinh viên) thì nhóm chúng em quyết định chọn cách làm 1 ứng dụng của stack đó là tháp Hà Nội. Đây là 1 trò chơi đã lâu rồi, có ứng dụng nguyên lý cấu trúc dữ liệu stack vào hoạt động của nó.

Dù việc nghiên cứu và học tập về stack là một đề tài không còn mới mẻ nữa, thậm chí lạc hậu. Tuy nhiên stack là một mảng kiến thức không thể thiếu cho những lập trình viên tương lai. Do đó, việc học tập và nghiên cứu luôn cần thiết và việc làm một ứng dụng của stack làm cho việc tiếp cần trở nên dễ dàng hơn.

1. ***Mục tiêu nhiệm vụ***

* Hoàn thành tốt sản phẩm có ứng dụng cấu trúc dữ liệu stack đó là tháp Hà Nội.
* Nắm rõ nguyên lý hoạt động và ứng dụng của stack.

1. ***Đối tượng nghiên cứu***

* Lý thuyết về cấu trúc dữ liệu stack.
* Một ứng dụng của stack: game tháp Hà Nội.

1. ***Phương pháp nghiên cứu***

Nghiên cứu, học tập chủ yếu thông qua các bài giảng trên mạng youtube về cấu trúc dữ liệu và giải thuật, các toolbox winform, ngoài ra còn các bài giảng của giảng viên, sách tham khảo.

1. ***Cấu trúc báo cáo***

Báo cáo gồm 3 phần:

Phần 1- Mở đầu: nêu lý do chọn đề tài, mục đích nghiên cứu đề tài, đối tượng và phương pháp nghiên cứu.

Phần 2- Nội dung:

* Mô tả về sản phẩm.
* Mô tả quá trình làm:làm rõ thiết kế cấu trúc dữ liệu giải thuật và giải thích.
* Mô tả phân công công việc.

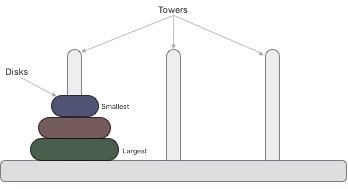
Phần 3- Kết luận: tự đánh giá mức độ hoàn thành mục tiêu.

# PHẦN 2: NỘI DUNG

1. ***Mô tả sản phẩm***
   1. **Giới thiệu trò chơi**

Game tháp Hà Nội của nhóm em lập trình có thể thực hiện các bước di chuyển như khi chơi ở ngoài đời, có đồng hồ tính giờ, có chỗ lưu lại số bước di chuyển để dễ dàng đánh giá sự tiến bộ của mình. Có nút thua để có thể chơi lại. Có thể chọn số đĩa từ 1->8. Ngoài ra còn có thể tự động chơi nếu người chơi muốn xem.

* Ta hãy đi tìm hiểu tháp Hà Nội là gì?
* “Tháp Hà Nội (Tower of Hanoi) là một trò chơi toán học bao gồm 3 cột và với số đĩa nhiều hơn 1. Các đĩa có kích cỡ khác nhau và xếp theo tự tự tăng dần về kích cỡ từ trên xuống: đĩa nhỏ hơn ở trên đĩa lớn hơn. Với số đĩa khác nhau thì ta có các bài toán Tháp Hà Nội khác nhau, tuy nhiên lời giải cho các bài toán này là tương tự nhau. Lời giải tối ưu cho bài toán Tháp Hà Nội là khi trò chơi chỉ có 3 cọc. Với số cọc lớn hơn thì lời giải bài toán vẫn chưa được khẳng định.”(1)



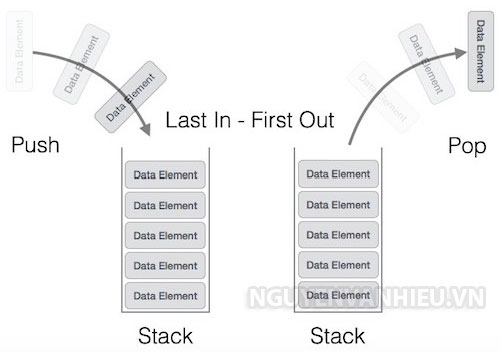
Hình 1: Sơ đồ tháp Hà Nội

* “Quy tắc của trò chơi: Chỉ được di chuyển đĩa nằm trên cùng (không được di chuyển các đĩa nằm giữa). Đĩa có kích thước lớn hơn không thể được đặt trên đĩa có kích thước nhỏ hơn. Mỗi lần chỉ có thể di chuyển 1 đĩa từ cột này sang cột khác. Kết quả cuối là cột ở giữa có tất cả các đĩa theo thứ tự từ nhỏ đến lớn.”(1)
  1. **Giới thiệu cấu trúc dữ liệu và giải thuật đã dùng**

“Trong khoa học máy tính, một ngăn xếp (còn gọi là bộ xếp chồng, stack) là một cấu trúc dữ liệu trù tượng hoạt động theo nguyên lý “vào sau ra trước” (Last In First Out) hay (LIFO). Tức là, phần tử cuối cùng được chèn vào ngăn xếp sẽ là phần tử đầu tiên được lấy ra khỏi ngăn xếp.”(2)

Một ví dụ trực quan, bạn có 1 chồng đĩa được xếp vào 1 cái hộp như hình dưới. Giả sử hộp này là vừa khít với các đĩa. Khi đó, bạn có các thao tác là:

* Thêm 1 cái đĩa vào hộp(push).
* Lấy một cái đĩa ra khỏi hộp, chỉ có thể lấy được đĩa trên cùng(pop)



Hình 2: Cấu trúc dữ liệu stack

Cấu trúc dữ liệu ngăn xếp bị giới hạn theo cách như trên. Như vậy, việc thao tác với ngăn xếp của chúng ta chỉ bao gồm các hành động sau: “

* Push: thêm 1 phần tử vào đỉnh ngăn xếp, số phần tử của ngăn xếp tăng thêm một.
* Pop: xóa bỏ phần tử đầu tiên ở đỉnh ngăn xếp, số phần tử của ngăn xếp giảm đi một.
* Top: lấy giá trị của phần tử đầu tiên ở đỉnh của ngăn xếp, số phần tử của ngăn xếp không thay đổi.
* IsEmpty: kiểm tra ngăn xếp trống hay không. Ngăn xếp trống là ngăn xếp không có phần tử nào.
* IsFull: kiểm tra ngăn xếp đã đầy hay chưa. Thao tác này không phải lúc nào cũng có.
* Size: lấy số lượng phần tử stack đang có.”(2)

1. ***Mô tả quá trình làm***
   1. **Quá trình thiết kế form**

Để thiết kế giao diện cho game nhóm chúng em sử dụng chương trình Microsoft Visual Studio.

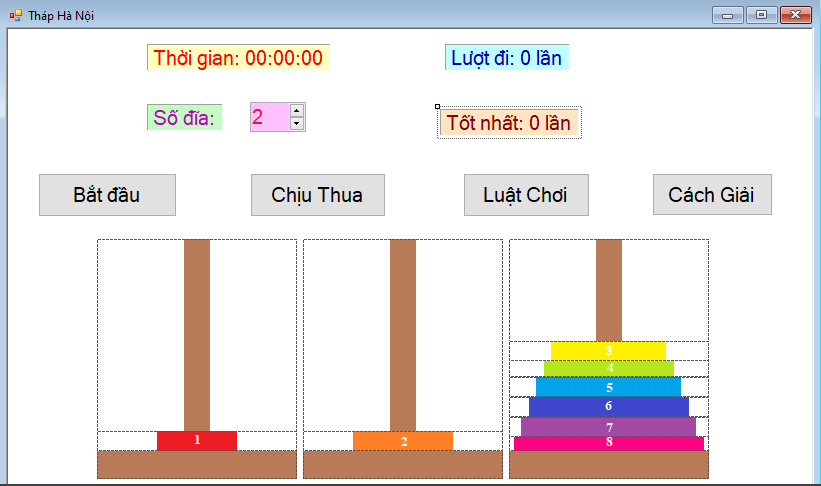
Ở đây ta thấy chương trình gồm 4 nút( button) đó là :bắt đầu, chịu thua, luật chơi và cách giải.

Có ba thanh cho biết các thông tin như: thời gian, lượt đi, tốt nhất. Trong đó:

* Thời gian là thời gian tính từ lúc bạn bắt đầu đến hiện tại ( hoặc khi bạn chịu thua),.
* Lượt đi là số lượt bạn đã di chuyển,.
* Tốt nhất là số lượt đi ngắn nhất để chiến thắng.

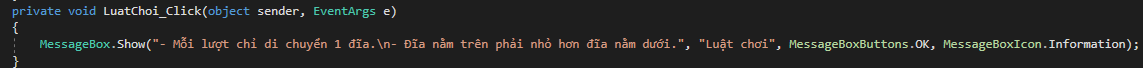
Giao diện có một ô chọn để bạn lựa chọn số đĩa khi chơi, số đĩa sẽ quyết định số lượt đi ngắn nhất. Số đĩa có thể chọn là từ 1->8.

Ở dưới là ba cây cột để người chơi di chuyển các đĩa qua lại.



Hình 3: Giao diện

* 1. **Quá trình cài đặt mã cho các nút**
     1. **Nút luật chơi**



Hình 4; Mã của luật chơi

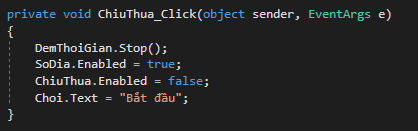
Ở đây ta sử dụng cấu trúc MessageBox.Show() để tạo 1 hộp thoại hiện lên khi người chơi nhấn vào để thông báo cho họ biết về luật chơi đã được đặt ra.

MessageBoxButtons.OK nghĩa là khi hộp thoại hiện lên thì nó cũng kèm 1 nút nhấn “OK” để tắt hộp thoại.

MessageBoxIcon.Infomation là lệnh dùng để làm xuất hiện 1 hình ảnh dấu chấm hỏi ở trong hộp thoại, làm nó sinh động hơn.

* + 1. **Nút chịu thua**

Khi chịu thua thì nghĩa là thời gian đang đếm sẽ ngừng lại. Nút chịu thua sẽ không bấm được. Nút bắt đầu sẽ sáng lên và ta có thể chỉnh lại số đĩa.



Hình 5: Mã của nút chịu thua

Ở đây em sử dụng lệnh Stop() để ngừng thời gian đang đếm trên thanh thời gian.

Enable là khả năng thực hiện lệnh của 1 nút,.. ở đây đặt enable của sodia thành true hàm ý ta có thể chọn số đĩa được. Enable của chịu thua là false cho thấy ta không thể nhấn chịu thua liên tục được vì nó vô nghĩa.

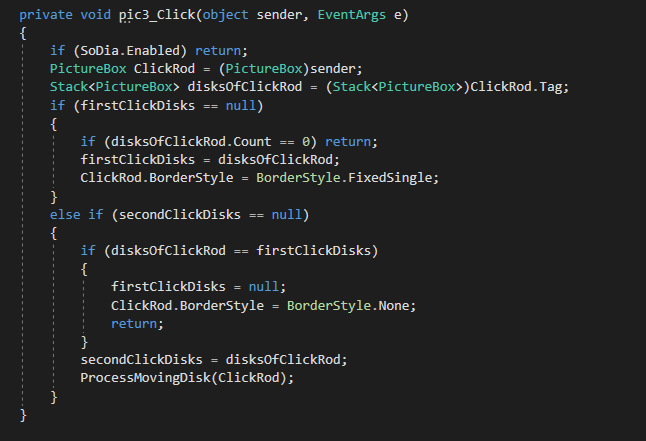
* 1. **Quá trình xử lý di chuyển**
     1. **Click vào cột muốn di chuyển**

Đầu tiên ta kiểm tra xem trò chơi đã được bắt đầu hay chưa bằng if. Nếu chưa thì sẽ không có gì xảy ra khi nhấn.

Tiếp theo ta chuyển cây cột được nhấn từ object sang pictureBox. Từ đó tạo 1 stack các pictureBox chứa các đĩa ở cột đã nhấn.

Sau đó ta kiểm tra xem tập firstClickDisks có rỗng hay không, nếu rỗng nghĩa là ta chưa nhấn lần nào. Và hàm sẽ tiếp tục xử lý bằng cách kiểm tra xem cột được nhấn có chứa đĩa nào hay không, nếu không chứa sẽ kết thúc. Nếu không thì tiếp theo ta sẽ chứa số đĩa ở cột đã nhấn vào firstClickDisks. Tiếp theo đó viền xung quanh cột sẽ bật lên cho biết mình đã nhấn vào.

Tiếp đến, nếu đã nhấn lần một thì chương trình sẽ kiểm tra xem ta đã nhấn lần hai chưa. Nếu chưa hàm sẽ xử lý theo hai trường hợp: một là nếu ta nhấn lần hai trùng với lần nhấn một thì chương trình xem như là người chơi bỏ chọn cột đó, làm firstClickDisks trở thành null và tắt borderStyle sau đó thoát xử lý.



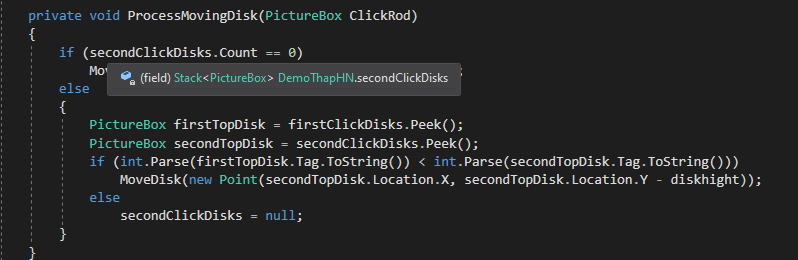
Hình 6: Mã khi nhấn vào cột

Nếu là trường hợp còn lại thì tập đĩa được nhấn lần hai sẽ chứa stack các đĩa trong cột được nhấn và vào hàm xử lý di chuyển tiếp theo.

* + 1. **Xử lý di chuyển đĩa giữa hai cột đã được nhấn**

Đầu tiên, ta kiểm tra xem tập các đĩa trong cột được nhấn lần hai (secondClickDisks) xem nếu không có đĩa thì ta gọi hàm chuyển đĩa (moveDisk) . Ta truyền vào hàm chuyển đĩa 1 point cho biết tọa độ của vị trí chuyển đến.

Ở trường hợp còn lại, trước tiên ta lấy đĩa ở trên cùng của tập đĩa nhấn lần 1 (firstClickDisks) và tập đĩa nhấn lần hai bằng hàm Peek(). Sau đó ta so sánh chúng, nếu đĩa nhấn lần 1 nhỏ hơn đĩa nhấn lần hai thì ta gọi hàm chuyển đĩa và truyền vào 1 point gồm tọa độ của vị trí đích.



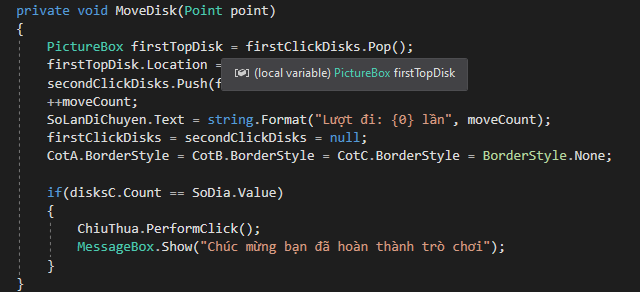
Hình 7: Mã xử lý di chuyển

Nếu đĩa trên cùng của lần nhấn một lớn hơn đĩa trên cùng của lần hai thì ta cho secondClickDisks bằng null, xem như là chưa chọn lần hai.

* + 1. **Xử lý trong hàm chuyển đĩa( moveDisk)**

Ở đây ta tạo 1 biến firstTopDisks để giữ đĩa trên cùng của tập đĩa được nhấn lần đầu bằng cách pop từ stack. Sau đó ta dùng Location để lấy tọa độ nơi đĩa đó đang ở. Tiếp đến ta dùng hàm push() để bỏ đĩa fisrtTopDisks lên trên đầu tập đĩa nhấn lần hai.

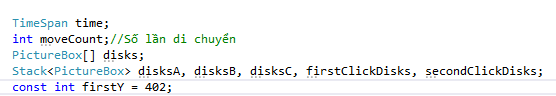
Tiếp theo ta cập nhập số bước di chuyển bằng cách tăng biến đếm lên thêm một đơn vị. Vì đã di chuyển xong nên ta cho tập nhấn lần một và hai trở về giá trị null. Các borderStyle ở các cột được nhấn cũng tắt đi báo hiệu việc chuyển đổi đã hoàn tất.



Hình 8: Mã di chuyển

Cuối cùng ta kiểm tra xem ở cột thứ 3 đã chứa đủ số đĩa chưa, nếu đủ nghĩa là trò chơi đã hoàn thành, 1 thông báo sẽ hiện lên chúc mừng kẻ chiến thắng.

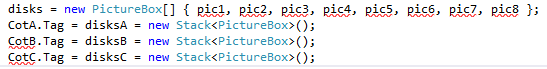
* 1. **Quá trình khai báo các biến**



Hình 9: Phần khai báo đầu tiên

Ở đây chúng em khai báo biến time với kiểu dữ liệu TimeSpan để lưu thời gian mà người chơi sử dụng khi giải.

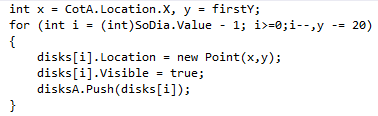
Tiếp theo, để lưu trữ tập đĩa trong 1 cột hay tập đĩa trong cột mà ta đã nhấn em sử dụng cấu trúc dữ liệu stack. Stack<PictureBox> ở đây nghĩa là tạo ra một ngăn xếp chứa các phần tử là các PictureBox. Ở đây các biến sử dụng kiểu dữ liệu này là disksA (lưu các đĩa đang được bỏ vô cột A) , disksB, disksC, firstClickDisks (lưu các đĩa ở trong cột mà người chơi nhấn lần đầu), secondClickDisk.



Hình 10: Cấp phát vùng nhớ

Ở phần này các ngăn xếp được khai báo bên trên, cụ thể là ba ngăn xếp chứa đĩa disksA, disksB, disksC được cấp vùng nhớ. Phải dùng cấu trúc dữ liệu stack là vì đặc tính vào trước ra sau (FIFO) của nó, ở trò chơi này những đĩa được di chuyển sẽ là những đĩa trên cùng- phần tử vào sau cùng của ngăn xếp và cũng là phần tử ra đầu tiên khi được chọn.

* 1. **Quá trình hình thành các đĩa khi bắt đầu chơi**



Hình 11: Mã quá trình tạo đĩa

Khi nhấn bắt đầu thì ngoài việc thời gian bắt đầu đếm, số bước di chuyển sẽ hiển thị thì 1 phần quan trọng cũng sẽ được thức hiện đó là quá trình tạo ra các đĩa.

Ở đây ta thấy rằng thuật toán tạo đĩa trên cột được thực hiện bằng việc sử dụng vòng lặp for. Điều kiện của vòng lặp này chính là nó sẽ chạy từ đĩa thứ tự sau đến trước vì luật của trò chơi (đĩa nhỏ ở trên, đĩa cao ở dưới) .

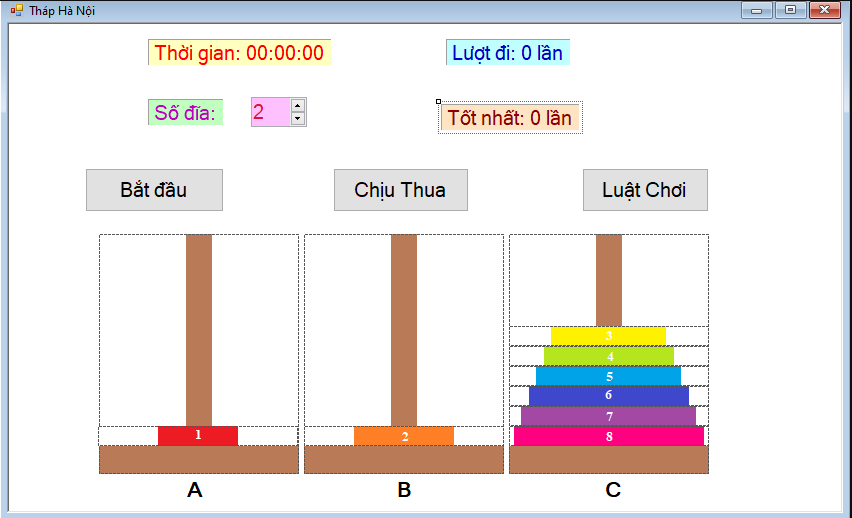
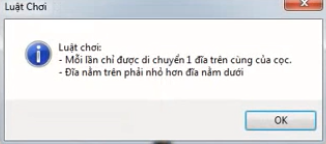
Đĩa thứ cao nhất (người chơi lựa chọn) sẽ được gọi tên đầu tiên, sau đó nó sẽ được thiết lập vị trí bằng Point(x,y) với x là tọa độ trục hoành, y là tọa độ trục tung. Sau khi thiết lập vị trí xong thì ta sẽ cho nó hiện hình lên, bỏ đĩa đó vào stack disksA bằng phép Push. Qua mỗi vòng lặp thì giá trị của i và y giảm dần cho đến khi i<0 thì vòng lặp sẽ ngừng.

1. ***Luồng chạy của chương trình***

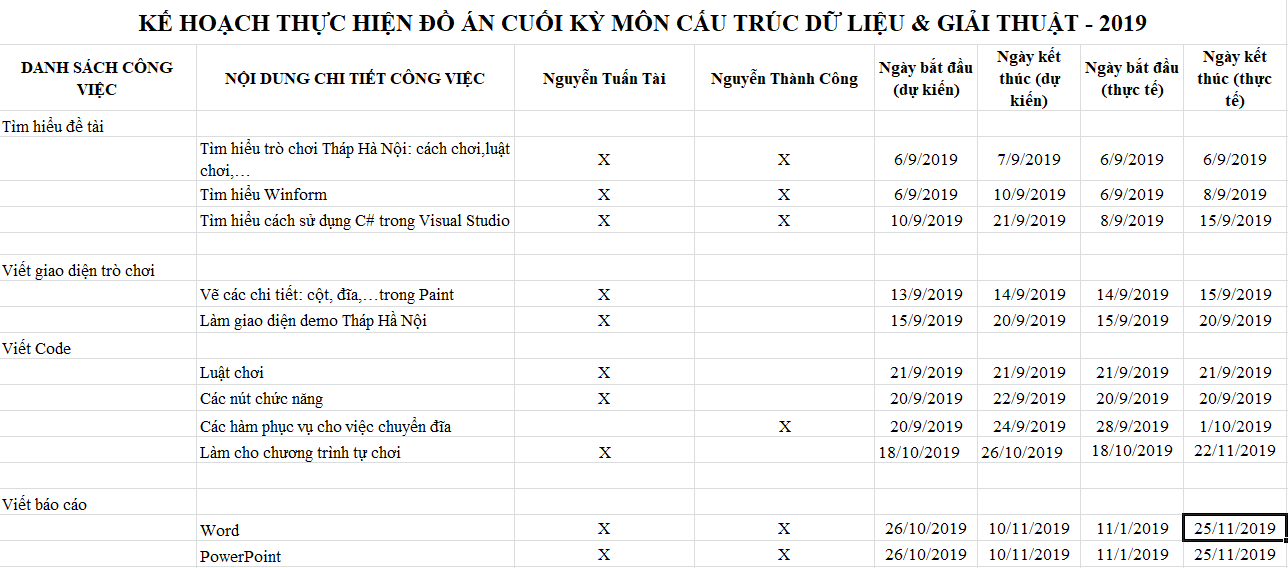
Khi bắt đầu, người chơi sẽ chọn số đĩa muốn chơi sau đó nhấn chơi. Sau đó các đĩa sẽ hiện ra ở cột bên trái theo thứ tự từ trên xuống dưới là 1,2… Bạn là người sẽ bắt đầu di chuyển các đĩa theo ý thích của mình. Với lần nhấn đầu tiên,bạn phải nhấn vào 1 cột có đĩa nếu không thì sẽ không có gì xảy ra cả. Khi nhấn vào, viền của cột được bật lên báo hiệu rằng bạn đã chọn đĩa ở cột này, nếu bạn muốn bỏ chọn chỉ cần nhấn vào lại một lần nữa. Nếu bạn muốn di chuyển đĩa qua 1 cột khác, chỉ cần nhấn vào cột bạn muốn. Có một lưu ý ở đây là khi di chuyển thì phải đáp ứng được luật chơi đã đề ra nếu không thì sẽ chẳng có gì xảy ra.

Ở lần nhấn đầu tiên, một ngăn xếp sẽ được tạo để chứa tập đĩa ở cột bạn đã nhấn. Sau đó khi bạn nhấn vào cột muốn chuyển thì nếu cột muốn chuyển của bạn trùng với cột nhấn lần 1 thì lần nhấn 1 sẽ trở về Null, nếu nhấn vào cột khác thì 1 stack sẽ tạo ra để chứa tập đĩa ở cột đích. Tiếp đến chương trình sẽ so sánh số thứ tự của 2 đĩa ở trên cùng, nếu đáp ứng được điều kiện đã đề ra thì đĩa sẽ được chuyển sang cột đích.

Cứ như vậy cho đến khi tất cả các đĩa ở cột trái chuyển hết sang cột phải là trò chơi đã hoàn thành.

Một số hình ảnh trong trò chơi:

1. ***Bảng phân công công việc***



Hình 12: Bảng phân công theo kế hoạch

# PHẦN 3: KẾT LUẬN

Như vậy, thông qua chương trình trên, tuy còn chưa hoàn hảo, nhưng đã có thể thấy rõ được tác dụng, vai trò của stack trong các bước di chuyển ở trên. Theo cách đánh giá của nhóm, sản phẩm đã hoàn thiện mục tiêu ở mức độ khoảng 80% về khả năng hoạt động, độ tối ưu và hình thức giao diện.

Về khả năng hoạt động: dù đã thử nghiệm nhiều lần khá mượt nhưng đây chỉ là sản phẩm lần đầu thực hiện, có thể còn một số lỗi tiềm tàng mà sinh viên chưa chạm đến được mong thầy và các bạn đóng góp ý kiến để sản phẩm thêm hoàn hảo.

Về độ tối ưu: chính là những dòng mã dưới chương trình, thành phần thầm lặng không thể thiếu. Ở đây chúng em sử dụng cấu trúc dữ liệu stack là giải pháp cho việc di chuyển các đĩa. Hiện nay, công nghệ phát triển nên việc cài đặt chương trình này bằng một cách đơn giản hơn là có thể thực hiện được.

Về hình thức giao diện: giao diện này là lần đầu nhóm em làm nên có thể chưa bắt mắt, cách di chuyển chưa được mượt mà, hiệu ứng di chuyển các đĩa không có kéo thả mà chỉ có nhấn và chọn thôi. Không có trang chính và trang chơi, không có âm thanh sống động.

Về các khó khăn khi gặp phải đó là:

* Cách áp dụng stack vào trong chương trình.
* Điều gì sẽ xảy ra khi nhấn các nút chức năng.
* Nhấn vào vùng xung quanh cột (không có đĩa) thì được nhưng nhấn vào đĩa thì không được.
* Khó khăn trong việc làm cho chương trình tự chơi.

Cách khắc phục các khó khăn ở trên:

* Bàn bạc với nhau tìm cách giải quyết.
* Tham khảo những người đi trước.
* Hỏi thăm những người xung quanh.
* Rút ra giải pháp sau nhiều lần thử nghiệm.

Những giới hạn của chương trình:

* Chương trình chỉ mới chơi được với số đĩa là 8.
* Khi người chơi cố tình di chuyển sai, không có thông báo hiện lên.
* Các lượt đi ngắn nhất chỉ là trên lý thuyết.

Phương hướng phát triển của báo cáo là không chỉ dừng ở độ hoàn thiện 80% mà phải hướng tới 85-95% độ hoàn thiện. Tiếp theo đó là cải thiện các dòng mã để chương trình dễ hiểu hơn, giúp cho các sinh viên tiếp cận với môn học cấu trúc dữ liệu và giải thuật có thể hiểu được một cách nhanh nhất. Tìm hiểu các thuật toán mới và thử nghiệm xem có thể áp dụng vào để cải tiến chương trình tháp Hà Nội này hay không. Sửa chữa các lỗi mà các bạn, giáo viên báo cáo cho nhóm mình. Cuối cùng là cải thiện giao diện để thu hút hơn, có hình ảnh động, âm thanh, màn hình chờ…

# Danh mục tài liệu tham khảo

Lê Văn Vinh (2013). *Cấu trúc dữ liệu và giải thuật.* Nhà xuất bảnđại học quốc gia TP. Hồ Chí Minh, thành phố Hồ Chí Minh.

Nguyễn Thị Kim Oanh (2008). *Cấu trúc dữ liệu stack và ứng dụng của stack trong các giải thuật đệ quy*. Luận văn (sinh viên), khoa công nghệ thông tin, trường đại học sư phạm Hà Nội.

(1): Tỷ Phú (2018). Bài toán tháp Hà Nội (Tower of Hanoi)[online], truy cập lần cuối 16/11/2019, từ:< <https://quantrimang.com/bai-toan-thap-ha-noi-tower-of-hanoi-156528>>.

(2): Nguyễn Văn Hiếu , Ngăn xếp- Stack[online], truy cập lần cuối 16/11/2019, từ:< <https://nguyenvanhieu.vn/ngan-xep-stack/>>.

Lưu Thế Phương (2012). Lập Trình C# \_ Tháp Hà Nội \_ Sử dụng Stack để xây dựng[online],truy cập lần cuối vào ngày 24/11/2019, từ: < https://www.youtube. com/watch?v=vk1tf0EZAgY&t=2774s >.