TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

KHOA ĐÀO TẠO
CHẤT LƯỢNG CAO

MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

BÁO CÁO ĐỒ ÁN CUỐI KÌ
MÔ PHỎNG BÀI TOÁN THÁP HÀ NỘI SỬ DỤNG NGĂN XẾP (STACK)

Nhóm sinh viên thực hiện:

Trần Ái Hải Sơn	**18110192**
Lương Anh Tuấn	**18110227**

Giảng viên hướng dẫn: ThS. Trần Công Tú

Tp. Hồ Chí Minh, tháng 12 - 2019

MỤC LỤC

MỤC LỤC HÌNH VÀ BẢNG

NỘI DUNG

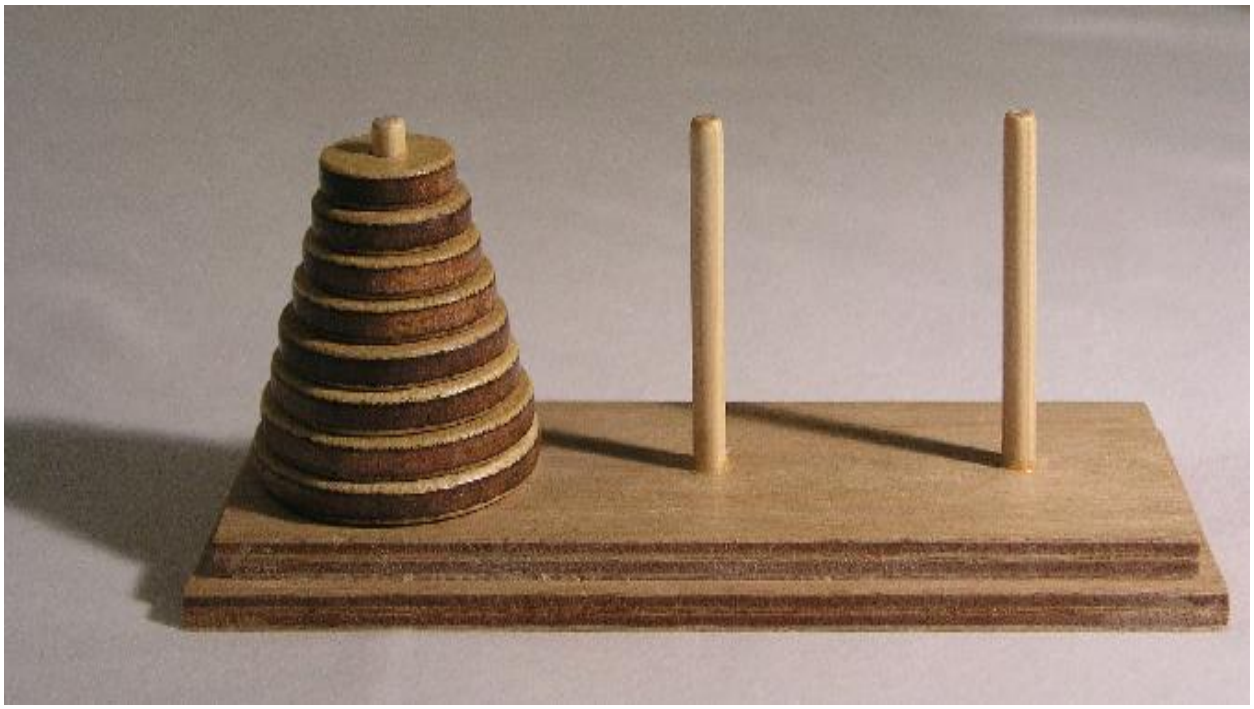
1. Tổng quan đồ án

1.1. *Lí do, mục đích chọn Đồ án mô phỏng Tháp Hà Nội*

Để thực hiện một chương trình ta phải xây dựng cấu trúc dữ liệu và giải thuật, đó cũng là tên môn mà chúng ta đã được học. Nhằm hiện thực những kiến thucwc đã được học, nhóm sẽ nghiên cứu về Ngăn xếp, muốn tìm hiểu thật chắc về Ngăn xếp thì phải hiểu được cách nó chạy, cách nó thực thi như thế nào,thông qua mô phỏng mà việc học một ngôn ngữ hay một thuật toán sẽ dễ dàng hơn. Chính vì vậy nhóm quyết định đi xây dựng thuật toán cho chương trình, cụ thể là mô phỏng đề tài Demo Tháp Hà Nội sử dụng thuật toán Ngăn xếp (Stack).

1.2. *Giới thiệu tổng quan về bài toán Tháp Hà Nội*

1.2.1. Lịch sử hình thành



Hình 1.1: Tháp Hà Nội.

[Tháp Hà Nội – Wikipedia tiếng Việt.]

Tháp Hà Nội là một trò chơi được biết đã xuất hiện ở Đông Á từ thế kỷ 19. Trò chơi này được đưa sang phương Tây lần đầu bởi nhà toán học người Pháp Edouard Lucas và được các nhà toán học nghiên cứu sau đó trở nên nổi tiếng vì độ phức tạp của nó. Lời giải tối ưu cho trò chơi có thể tìm thấy chính xác cho trường hợp 3 cọc. Nhưng khi mở rộng cho 4 cọc hoặc nhiều hơn, lời giải chính xác cho đến nay vẫn chưa được khẳng định**.**

1.2.2. Cách giải trong thực tế

Ba cột nằm ngay thẳng hàng. Đĩa được sắp xếp từ lớn đến nhỏ từ thấp lên cao, tạo nên một Tòa tháp. Trò chơi đòi hỏi di chuyển các đĩa, bằng cách đặt chúng vào cột bên cạnh, một đĩa trong một cột di chuyển, theo luật sau:

I. Sau mỗi di chuyển, các đĩa đều nằm trên một, hai, hoặc ba cột, theo thứ tự từ lớn đến nhỏ từ thấp đến cao.

II. Đĩa trên cùng của một trong ba cột đĩa được đặt vào cột rỗng.

III. Đĩa trên cùng của một trong ba cột đĩa được đặt lên một cột đĩa khác, nếu đĩa này nhỏ hơn các đĩa của cột này.

Trò chơi được phân theo cấp độ tuỳ vào việc chọn số đĩa tăng-giảm đồng nghĩa với việc thời gian chơi cũng được tăng lên.

1.2.3. Cách giải khác với cách nhìn của khoa học máy tính

Bài toán Tháp Hà Nội là một ví dụ về phương pháp giải đệ. Sau đây là dạng rút gọn của giải thuật đệ quy được áp dụng:

Bước 1: Chuyển đĩa 1 sang cọc C

Bước 2: Chuyển đĩa 2 sang cọc B

Bước 3: Chuyển đĩa 1 từ C sang B sao cho nó nằm lên 2.

Vậy ta hiện có 2 đĩa đã nằm trên cọc B, cọc C hiện thời trống

Bước 4: Chuyển đĩa 3 sang cọc C

Bước 5: Lặp lại 3 bước trên để chuyển 1 & 2 cho nằm lên 3

Mỗi lần dựng xong tháp từ đĩa i đến 1, chuyển đĩa i + 1 từ cọc A là cọc xuất phát, rồi lại di chuyển tháp đã dựng lên đĩa i + 1.

Nhưng trong đồ án này, chúng ta sẽ không sử dụng giải thuật đệ quy để giải quyết nó mà thay vào đó là dùng cấu trúc dữ liệu ngăn xếp (stack) để khử đệ quy, giải quyết bài toán.

1.2.4. Ứng dụng

- Là một bài toán thường được dùng để dạy về lập trình cơ bản.
- Dùng trong nghiên cứu tâm lý về cách giải quyết vấn đề.
- Dùng trong chẩn đoán và điều trị thần kinh tâm lý đối với các chức năng thực hành.

1.3. *Mục tiêu đồ án Mô phỏng bài toán Tháp Hà Nội*

- Nắm bắt được nội dung bài học và áp dụng vào đề tài.
- Tìm hiểu cách làm, xây dựng bố cục đồ án.
- Áp dụng ngôn ngữ trình, bài học ứng dụng vào giải quyết bài toán.

1.4. *Về đồ án Mô phỏng bài toán Tháp Hà Nội*

1.4.1. Khái niệm về mô phỏng bài toán

Mô phỏng bài toán là quá trình tách dữ liệu, thao tác, ngữ nghĩa và mô phỏng đồ họa cho quá trình trên.

1.4.2. Tác dụng mô phỏng bài toán

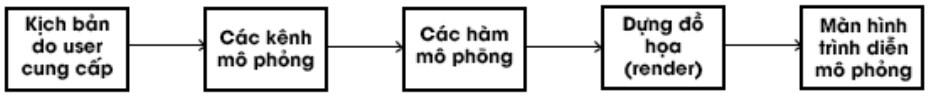
Người dùng có thể xem từng bước chương trình được thực thi, nhờ thế có thể hiểu chi tiết và đánh giá bài.

1.4.3. Kiến trúc một hệ thống mô phỏng thuật toán

Đa số các hệ thống, chương trình mô phỏng thuật toán đều có những thư viện hỗ trợ thủ tục mô phỏng và giao diện mô phỏng. Nhìn chung, mỗi hệ thống, chương trình mô phỏng đều gồm 2 phần chính:

1. Phần xử lí: Xứ lí dữ liệu đầu vào của thuật toán, chạy thuật toán cần mô phỏng đã được sửa lại với mục đích gửi dữ liệu đã tính toán tới phần thứ 2.

2. Phần khung nhìn (cảnh quan): Là nơi người dùng nhìn những đối tượng mô phỏng. Thông điệp bao gồm thông tin đồ họa của đối tượng cần mô phỏng. Sau khi khung nhìn nhận dữ liệu, nó tính toán lại và kéo (render) đối tượng mức đồ họa.



Hình 1.2: Sơ đồ khối hệ thống mô phỏng thuật toán.

1.4.4. Mô phỏng bài toán tháp Hà Nội

Cũng tuân theo kiến trúc trên, chương trình sản phẩm của đồ án mô tháp Hà Nội sử dụng stack của nhóm cũng tuân theo hai phần chính. Chương trình nhận dữ liệu đầu vào là số đĩa và tốc độ mô phỏng, qua quá trình tính toán, chương trình vẽ ra trên giao diện người dùng từng bước đi cụ thể của từng chiếc đĩa, đồng thời, ghi lại nhật kí (log) cho từng bước di chuyển dưới dạng kí tự (text display).

1.4.5. Lựa chọn ngôn ngữ cài đặt mô phỏng

Vì chương trình mô phỏng của nhóm biểu diễn thuật toán dưới dạng các đối tượng có thuộc tính, hành vi rõ ràng nên nhóm nhóm quyết định sử dụng ngôn ngữ Microsoft C#.NET với phiên bản .NET Framework 4.5.2; trình biên soạn (IDE) và trình biên dịch (Compiler) mặc định trong bộ Microsoft Visual Studio 2017 để cài đặt thuật toán và viết chương trình mô phỏng bài toán Tháp Hà Nội này này.

2. Sử dụng Stack để mô phỏng đề tài

2.1. *Khái quát bài toán tháp Hà Nội bằng ngăn xếp*

2.1.1. Ngăn xếp (stack) là gì ?

Trong khoa học máy tính, ngăn xếp là một cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý "vào sau ra trước" (Last In First Out (LIFO).

2.1.2. Cấu trúc và đặc điểm của ngăn xếp

Ngăn xếp là một cấu trúc dữ liệu dạng thùng chứa (container) của các phần tử và có hai phép toán cơ bản: push and pop. Push bổ sung một phần tử vào đỉnh (top) của ngăn xếp. Pop giải phóng và trả về phần tử đang đứng ở đỉnh của ngăn xếp. 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. Ngoài ra, stack cũng hỗ trợ một số thao tác khác:

- isEmpty(): Kiểm tra xem stack có rỗng không.
- Top() (Hay peek()): Trả về giá trị của phần tử nằm ở đầu stack mà không hủy nó khỏi stack. Nếu stack rỗng thì lỗi sẽ xảy ra.
- Push(): Đẩy một ngăn mới vào một "xếp".

2.1.3. Ứng dụng của ngăn xếp

Ngăn xếp có nhiều ứng dụng trong khoa học máy tính. Mà quan trọng nhất là sử dụng để khử đệ quy và quản lí bộ nhớ khi thi hành chương trình bằng cách biểu diễn các thanh ghi (register) dưới dạng ngăn xếp.Đối với chương trình mô phỏng Tháp Hà Nội này nhóm ứng dụng để xử lí thuật toán sắp xếp đĩa.

2.1.4. Ứng dụng của ngăn xếp vào bài toán Tháp Hà Nội

2.1.4.1. Nguyên lí áp dụng vào thuật toán

Muốn đưa n đĩa từ cột A (cột nguồn) sang cột C (cột đích) thông qua cột B (cột trung gian) thì chỉ cần đưa (n - 1) đĩa từ A qua B, rồi đưa 1 đĩa từ A qua C và cuối cùng là đưa (n - 1) đĩa từ B qua C, bài toán được giải quyết.

Để (n - 1) đưa đĩa từ A sang B thì khi đó xem A là cột nguồn, B là cột đích và C là cột trung gian. Việc tiến hành tương tự, đưa (n - 2) khối từ cột nguồn qua cột trung gian, 1 khối từ cột nguồn sang cột đích và cuối cùng là (n - 2) khối từ cột trung gian sang cột đích.

2.1.4.2. Source thuật toán Tháp Hà Nội bằng ngăn xếp bằng C#

```
public void ChuyenDia(int soDia, Stack<PictureBox> A,
Stack<PictureBox> B, Stack<PictureBox> C)
{
    PicRod picrod = new PicRod();
    picrod.numberDisk = soDia;
    picrod.cocA = A;
    picrod.cocB = B;
    picrod.cocC = C;

    Stack<PicRod> myStack = new Stack<PicRod>();
    myStack.Push(picrod);
    while (myStack.Count > 0)
    {
        if (bstop == true)
            break;
        PicRod picrod0 = myStack.Pop();
        if (picrod0.numberDisk == 1)
        {
            MoveDemo(picrod0.cocA, picrod0.cocC);
        }
        else
        {
            PicRod picrod1 = new PicRod()
            {
                numberDisk = picrod0.numberDisk - 1,
                cocA = picrod0.cocB,
                cocB = picrod0.cocA,
                cocC = picrod0.cocC
            };
            myStack.Push(picrod1);
            PicRod picrod2 = new PicRod()
            {
                numberDisk = 1,
                cocA = picrod0.cocA,
                cocB = picrod0.cocB,
                cocC = picrod0.cocC
            };
            myStack.Push(picrod2);
            PicRod picrod3 = new PicRod()
            {
                numberDisk = picrod0.numberDisk - 1,
                cocA = picrod0.cocA,
```

```
                    cocB = picrod0.cocC,
                    cocC = picrod0.cocB
                };
                myStack.Push(picrod3);
            }
        }

    }
```

2.1.4.3. Input và output của thuật toán.

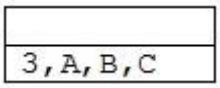
Input: Số đĩa: soDia, 3 Cọc: DisksA, DisksB, DisksC;

Output: Xuất ra tuần tự các bước di chuyển N đĩa từ cột A sang cột C, lấy cột B làm cột đệm (buffer).

2.1.4.4. Minh họa cho lời gọi ChuyenDia(3,A,B,C),

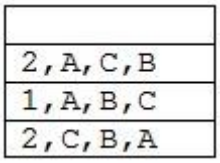
_ Tức là soDia = 3.

Ngăn xếp khởi đầu:



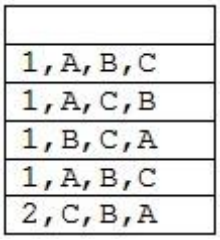
Hình 2.1.

Ngăn xếp sau lần lặp thứ nhất:



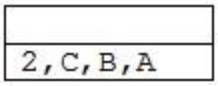
Hình 2.2.

Ngăn xếp sau lần lặp thứ hai:



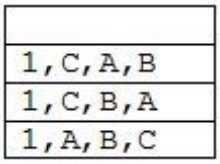
Hình 2.3.

Các lần lặp 3,4,5,6, Hàm vẽ (Movement(temp.A, temp.B)) lấy thông tin và bắt đầu sinh lệnh đồ họa di chuyển các đĩa trên màn hình người dùng, vì vậy không có ngăn dữ liệu nào được thêm vào ngăn xếp. Mỗi lần xử lý, phần tử đầu ngăn xếp bị xoá. Ta sẽ có ngăn xếp như sau:



Hình 2.4.

Tiếp tục lần lặp bước 7, ta được:



Hình 2.5.

Các lần lặp tiếp tục chỉ xử lý việc chuyển 1 đĩa. Chương trình con in ra các phép chuyển và dẫn đến ngăn xếp rỗng.[5]

2.1.4.5. Vai trò và tác dụng của ngăn xếp trong thuật toán.

Trong toàn bộ chương trình mô phỏng thuật toán mà nhóm xây dựng, tổng cộng có 3 ngăn xếp được sửa dụng(stack *DisksA*, stack *DisksB* và stack *DisksC*), đại diện cho 3 cột A, B, C của tháp Hà Nội với A là cột nguồn, B là cột trung gian, C là cột đích. Số đĩa trong cột sẽ ứng với số ngăn hiện có của ngăn xếp đó.

Khi người dùng truyền kịch bản vào, chương trình sẽ nạp cho ngăn xếp số ngăn (Đĩa được mô hình hóa thành các picturebox trong chương trình) đúng theo kịch bản vào stack *DisksA (cột A hay cột gốc).*

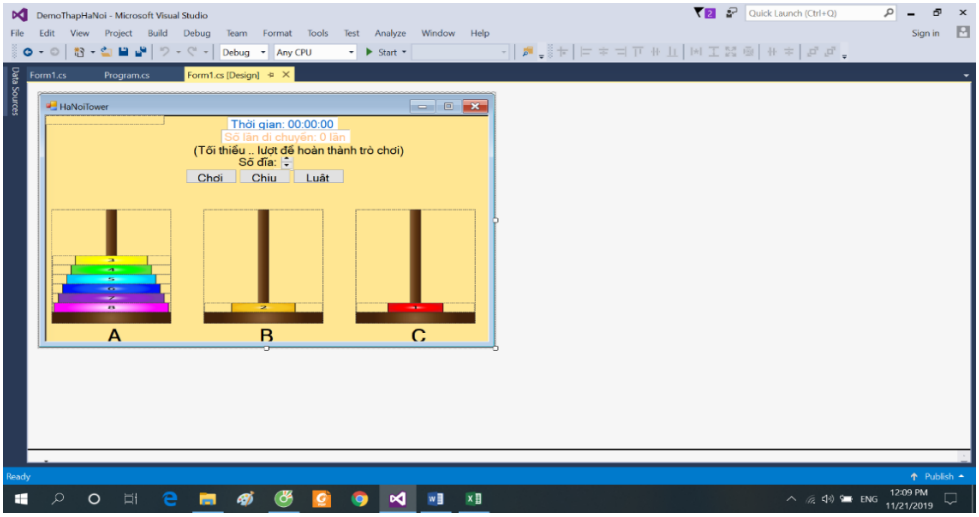
Sau đó, một biến n chứa số đĩa lấy từ kịch bản và cả ba stack này theo thứ tự 1, 2, 3 sẽ được đóng gói thành 1 struct và truyền vào thủ tục ChuyenDia để xử lí.

Tại đây, thủ tục sẽ đẩy struct đó vào một stack khác, tạm gọi là stack *myStack*. Chương trình kiểm rút ngăn trên cùng của *myStack* và kiểm tra xem biến n của ngăn đó có bằng 1 hay không. Nếu có thì sẽ gọi thủ tục di chuyển đĩa (*MoveDemo*) và truyền chỉ dẫn (di chuyển 1 đĩa ở cột tại vị trí số 1 về cột tại vị trí thứ 2. Với 1 và 2 theo ngăn xếp đang bị lấy ra xem xét) vào để nó di chuyển đĩa. Nếu n khác 1:

- Giảm n xuống 1 đơn vị, tráo hai cột ở vị trí số 1 và vị trí số 3 trong ngăn cho nhau rồi đẩy ngăn đã được xử lí vào stack *myStack*.
- Giữ nguyên n, tráo hai cột ở vị trí số 1 và vị trí số 2 trong ngăn cho nhau rồi đẩy ngăn đã xử lí vào stack *myStack.*
- Lại giảm n xuống 1 đơn vị, tráo hai cột ở vị trí số 2 và vị trí số 3 trong ngăn cho nhau rồi đẩy ngăn đã xử lí vào stack *myStack.*[(2)]

Quá trình này lặp lại từ (1) đến (2) cho tới khi stack *myStack* không còn chứa bất kì ngăn xếp nào nữa thì bài toán giải xong!

2.2. Thiết kế giao diện, backdrop.



Hình 2.6: Form chính của chương trình.

- Gồm những PictureBox về cọc và đĩa trong game.
- Các button về các chức năng như: Chơi, Chịu Thua, Luật chơi và Demo.
- Các label chú thích mở rộng tính năng để người chơi có thể cố gắng tìm ra những cách đi hay nhất.

2.3. Cài đặt thuật toán và viết chương trình.

- Cấu dữ liệu ngăn xếp (Stack): Sử dụng lớp (class) stack của C#, thuộc Namespace: System.Collections, được xây dựng sẵn trong bộ Visual Studio 2017.
- Cài đặt thuật toán Tháp Hà Nội sử dụng ngăn xếp gồm 2 phần:

    + Phần 1: struct PicRod.

    + Phần 2: Hàm void ChuyenDia(int soDia, Stack<PictureBox> A, Stack<PictureBox> B, Stack<PictureBox> C)

    Hàm void ChuyenDia(int soDia, Stack<PictureBox> A, Stack<PictureBox> B, Stack<PictureBox C) nhận đầu vào số nguyên x là số đĩa của tháp cần giải. Sau đó, nó xếp các ngăn (Các struct ThuTuc đã được truyền dữ liệu phù hợp) và bắt đầu giả tháp.

Đồng thời, nó truyền dữ liệu của từng bước giải ra, chỉ dẫn cho hàm vẽ thực hiện các lệnh đồ họa.

- Cài đặt hàm vẽ void MoveDemo(Stack<PictureBox> A, Stack<PictureBox> C):
    + Nhận dữ liệu đầu vào và cột gốc và cột đích.
    + Di chuyển đĩa từ cột gốc về cột đích.
    + Tốc độ di chuyển tùy vào người dùng.

2.4. Kiểm lỗi và debug.

Trong quá trình kiểm lỗi cho sản phẩm, nhóm đã phát hiện ra 3 lỗi:

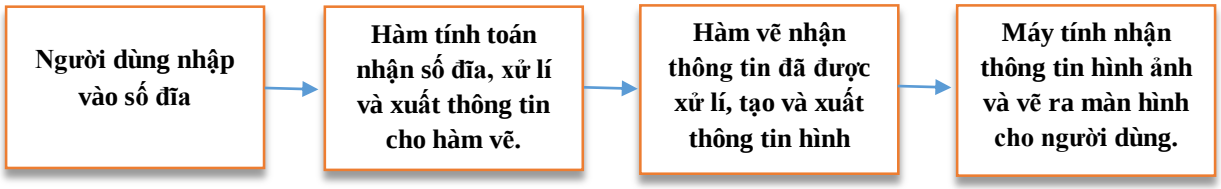
Mô tả lỗi	Nguyên nhân	Người phát hiện	Đã khắc phục
Người dùng có thể phóng to cửa sổ chương trình, gây sai lệnh tỉ lệ các thành phần trong cửa sổ.	Chưa set lại thuộc tính MaximizeBox của form về false.	Tuấn	Rồi
Đồng hồ đếm thời gian chạy sai.	Bị ảnh hưởng bởi lệnh delay trong hàm vẽ.	Tuấn	Rồi
Nút Demo bị lỗi khi bắt đầu di chuyển các đĩa	Do code chưa đúng	Sơn	Rồi

Bảng 1. Kiểm lỗi và debug

2.5. Chức năng nổi bật của chương trình

- Có thể chọn số đĩa cho bài toán tháp Hà Nội cần mô phỏng.
- Có thể dễ dàng quan sát mô phỏng.
- Có thể tự chơi và tìm ra cách nhanh nhất để giải.

2.6. Cách thức hoạt động của chương trình



Hình 2.7: Sơ đồ khối cách hoạt động của chương trình.

3. Kết luận và hướng phát triển

3.1. Ưu điểm

- Giúp người sử dụng dễ hiểu, dễ hình dung thuật toán.
- Giao diện thiết kế đơn giản, dễ sử dụng.
- Có thể chơi và tìm ra cách nhanh nhất để có thể giải bài toán.

3.2. Nhược điểm

- Bị giới hạn ở mức giải được Tháp Hà Nội với tối đa 8 đĩa.
- Đĩa di chuyển chưa được mượt mà.

3.3. Phương án phát triển

- Xây dựng các đĩa của tháp Hà Nội dưới dạng các đối tượng (Object) với các thuộc tính và hành vi rõ ràng để giải quyết mô phỏng tháp Hà Nội nhiều hơn 8 đĩa.
- Cải tiến thuật toán để nó chạy trơn tru và ổn định nhất.
- Dành nhiều thời gian hơn để test và debug, fix các lỗi chưa phát hiện được và làm chương trình hoàn thiện hơn.
- Dùng các thư viện đồ họa tiên tiến như OpenGL, DirectX, Vulkan, … để làm màn hình mô phỏng trở nên sinh động và đẹp mắt hơn.
- Có thể thêm chức năng Undo khi chơi và khi demo.
- Tạo 1 bảng lịch sử đã hoàn thành game để so sánh với những người chơi khác.

TÀI LIỆU THAM KHẢO

[1]. Tháp Hà Nội – Wikipedia tiếng Việt.
https://www.google.com.vn/search?tbm=isch&q=Demo%20Th%C3%A1p%20H%C3%A0%20N%E1%BB%99i#imgrc=Nhzzjie7TDOvGM

[2]. Ngăn xếp – Wikipedia tiếng Việt.
https://vi.wikipedia.org/wiki/Ng%C4%83n_x%E1%BA%BFp

[3]. Cấu trúc dữ liệu phần ngăn xếp.
https://voer.edu.vn/c/ngan-xep-stack/c5e6dc01/5fdc2109

[4]. Ngăn xếp (STACK) - voer.edu.vn

[5]. Stack Class (System.Collections) | Microsoft Docs.

LỜI CẢM ƠN

Để hoàn thành đồ án này, trong quá trình thực hiện nhóm em đã nhận được sự giúp đỡ của thầy và các bạn. Nhân đây, cho phép nhóm em gửi lời cảm ơn tới thầy và các bạn. Đặc biệt là thầy ThS. Trần Công Tú , thầy đã tận tâm hướng dẫn giúp đỡ để nhóm có thể hoàn thành đề tài một cách tốt nhất.

Vì đây cũng là lần đầu tiên thực hiện đồ án nên chắc chắn có nhiều thiếu xót, vì thế nhóm rất mong được sự đóng góp của các thầy cô cũng như bạn đọc. Những ý kiến đóng góp sẽ giúp nhóm nhận ra mặt hạn chế và đó cũng là nguồn tư liệu bổ ích để nhóm học tập đi đến hoàn thiện kiến thức và xa hơn có thể xây dựng chương trình tốt hơn cả trong học tập và công việc sau này.

Em xin chân thành cảm ơn!

PHỤ LỤC

***Link github*:** https://github.com/SonTran2019/Demo-HaNoi-Tower

DANH SÁCH CÔNG VIỆC	NỘI DUNG CHI TIẾT CÔNG VIỆC	Tuấn	Sơn
Tìm hiểu về kiến thức			
	Tìm hiểu về GitHub		X
	Tìm hiểu về Stack	X	X
	Tìm hiểu về nguyên lí trò chơi tháp Hà Nội		X
	Tìm hiểu cách sử dụng WinDows Forms	X	
	Tìm hiểu về cách trình bày đồ án cuối kì.		X
Phác thảo ý tưởng			
	Cách thức áp dụng Stack vào bài toán	X	
	Sử dụng winfoms để thiết kế khung và code	X	
	Xây dựng giao diện của trò chơi :	X	
	Hình thành trò chơi	X	
	Xây dựng trò chơi: +Tạo các nút lệnh +Đưa code vào nút	X	
	Lên nội dung báo cáo		X
Triển khai ý tưởng			
	Tham khảo và áp dụng các bài Demo đơn giản	X	X
	Thiết kế chi tiết theo đề tài	X	
	Hoàn thành và tối giản thuật toán	X	
	Hoàn chỉnh nội dung và trình chiếu.		X

Bảng 2 : Kế hoạch chi tiết thực hiện đồ án.