**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 : DEMO CÁC THAO TÁC TRÊN CÂY TÌM KIẾM NHỊ PHÂN**

**GVHD: Ths TRẦN CÔNG TÚ**

**SVTH : NGUYỄN THỊ YẾN NHI**

**MSSV : 18110170**

**SVTH : BÙI ĐỨC HUY**

**MSSV : 18110117**

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

# 

# MỤC LỤC

[MỤC LỤC 1](#_Toc26000257)

[DANH MỤC CÁC HÌNH 3](#_Toc26000258)

[NỘI DUNG 4](#_Toc26000259)

[*I.Giới thiệu đề tài:* 4](#_Toc26000260)

[1.Mô tả phần mềm 4](#_Toc26000261)

[2.Cấu trúc dữ liệu và giải thuật được dùng trong đề tài 4](#_Toc26000262)

[*II.Quá trình thực hiện:* 5](#_Toc26000263)

[1.Thiết kế giao diện 5](#_Toc26000264)

[2.Cấu trúc dữ liệu và giải thuật 5](#_Toc26000265)

[2.1. Cấu trúc dữ liệu chính trong chương trình 5](#_Toc26000266)

[*2.1.1. Tổng quan về cấu trúc dữ liệu cây* 5](#_Toc26000267)

[*2.1.2. Cây nhị phân tìm kiếm* 6](#_Toc26000268)

[2.2. Các thao tác chính trên cây nhị phân tìm kiếm 6](#_Toc26000269)

[*2.2.1.Tạo nút* 6](#_Toc26000270)

[*2.2.2.Tạo cây ban đầu* 7](#_Toc26000271)

[*2.2.3.Thao tác duyệt qua các nút trên cây nhị phân* 8](#_Toc26000272)

[*2.2.4.Các thao tác trên cây tìm kiếm nhị phân* 9](#_Toc26000273)

[2.3.Ứng dụng của cây nhị phân vào chương trình demo 13](#_Toc26000274)

[*2.3.1. Tạo hiệu ứng vẽ cây* 13](#_Toc26000275)

[*2.3.2.* *Tạo hiệu ứng tìm kiếm* 15](#_Toc26000276)

[*2.3.3.Tạo hiệu ứng thêm nút* 16](#_Toc26000277)

[*2.3.4.Tạo hiệu ứng xóa nút* 17](#_Toc26000278)

[*2.3.5.Tạo hiệu ứng duyệt* 20](#_Toc26000279)

[3.Cài đặt và kiểm thử 20](#_Toc26000280)

[3.1.Các trường hợp kiểm thử 20](#_Toc26000281)

[*3.1.1.Random tạo cây ban đầu* 21](#_Toc26000282)

[*3.1.2.Thực hiện Preoder* 21](#_Toc26000283)

[*3.1.3.Thực hiện Inorder* 21](#_Toc26000284)

[*3.1.4.Thực hiện Postorder* 21](#_Toc26000285)

[*3.1.5.Thực hiện Search* 21](#_Toc26000286)

[*3.1.6.Thực hiện Delete* 21](#_Toc26000287)

[*3.1.7.Lỗi chưa giải quyết* 21](#_Toc26000288)

[*III. Phân công công việc:* 22](#_Toc26000289)

[*IV. Đánh giá:* 22](#_Toc26000290)

[4.1.Đánh giá mức độ hoàn thành 22](#_Toc26000291)

[4.2.Các khó khăn 22](#_Toc26000292)

[*V.Kết luận:* 23](#_Toc26000293)

[5.1.Ưu điểm 23](#_Toc26000294)

[5.2.Nhược điểm 23](#_Toc26000295)

[5.3.Hướng phát triển 23](#_Toc26000296)

[TÀI LIỆU THAM KHẢO 24](#_Toc26000297)

# DANH MỤC CÁC HÌNH

[Hình 1: Giao diện 5](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133331)

[Hình 2: Tạo cây ban đầu 8](#_Toc26133332)

[Hình 3: Duyệt Preoder 8](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133333)

[Hình 4: Duyệt Inorder 9](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133334)

[Hình 5: Duyệt Postorder 9](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133335)

[Hình 6: Tìm kiếm trên cây 9](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133336)

[Hình 7: Thêm nút trên cây 10](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133337)

[Hình 8: Hủy nút lá 12](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133338)

[Hình 9: Hủy nút trung gian 12](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133339)

[Hình 10: Hủy nút có hai cây con 13](file:///D:\laptrinh\do%20an%20CTDLGT\bao_cao_do_an_CTDLGT.docx#_Toc26133340)

# NỘI DUNG

## *I.Giới thiệu đề tài:*

### 1.Mô tả phần mềm

Với sự phát triển của khoa học kỹ thuật ngày nay con người đã đạt được những thành tựu mới trong tất cả các lĩnh vực. Trong đó có ngành công nghệ thông tin với việc tạo ra các phần mềm hữu ích cho cuộc sống con người. Và môn cấu trúc dữ liệu và giải thuật ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực. Với một lượng thông tin ngày càng nhiều việc giải quyết vấn đề về tìm kiếm dữ liệu có hiệu quả và nhanh chóng đang là mục tiêu quan tâm hàng đầu của các nhà phát triển phần mềm. Thông thường thì dữ liệu được biểu diễn dưới dạng danh sách liên kết. Việc tìm kiếm dữ liệu, xử lý thông tin và truy xuất chưa được hiệu quả cao. Việc sử dụng cấu trúc dữ liệu cây là một giải pháp nhằm tăng hiệu suất trong các thao tác xử lý. Vấn đề với việc sử dụng cây là chúng ta cần phải dùng giải thuật nào với từng dạng dữ liệu để đạt hiệu quả cao nhất. Để giải quyết vấn đề trên chúng ta cùng tìm hiểu một số thao tác trên cây nhị phân tìm kiếm.

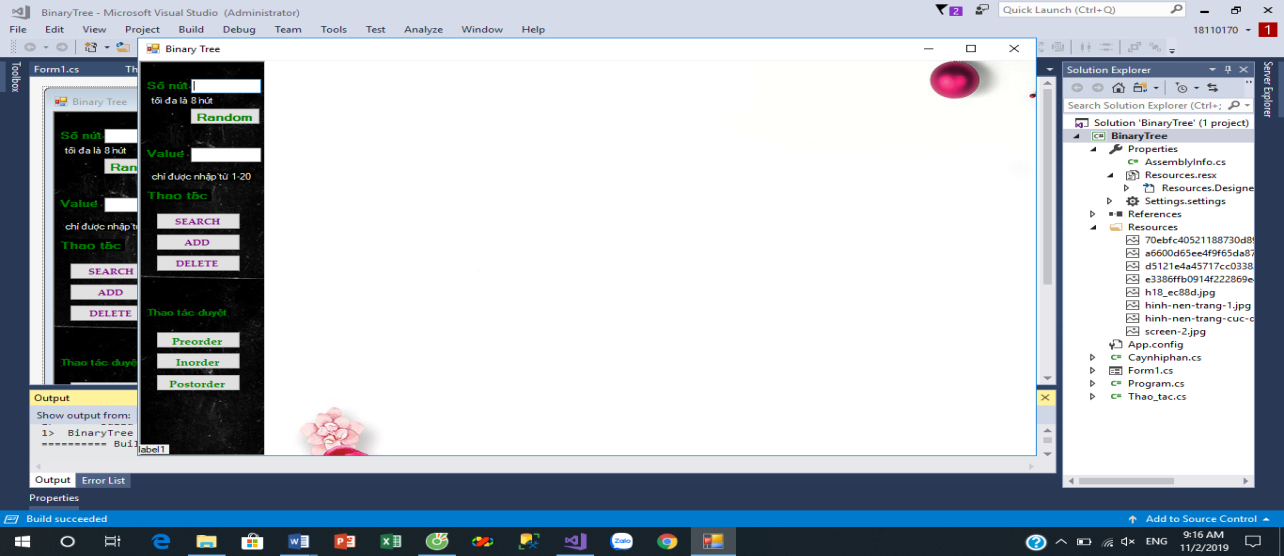
Chương trình dùng để hỗ trợ việc giảng dạy được thuận tiện hơn. Giúp cho học sinh sinh viên và người tìm hiểu có thể hiểu rõ hơn về nguyên tắc hoạt động của cây nhị phân tìm kiếm.

### 2.**Cấu trúc dữ liệu và giải thuật được dùng trong đề tài**

Sử dụng cây nhị phân, các hàm của cây nhị phân tìm kiếm để cài đặt chương trình demo. Các thao tác chính trên cây nhị phân. Bên cạnh đó, kết hợp một số thao tác Graphics để tạo thiết kế về giao diện. Dữ liệu đầu vào bao gồm phần nút gốc với giá trị được random trong khoảng từ 9 đến 12, những nút còn lại có giá trị random từ khoảng 1 đến 20, nhằm mục đích dễ tạo sự cân bằng hơn cho cây. Chương trình sử dụng giao diện Winform, số nút được tạo được nhập từ bàn phím với giá trị từ 1 đến 8.

## *II.Quá trình thực hiện:*

### 1.Thiết kế giao diện

**Form1** là giao diện chính của bài dùng để nhập số nút cần để tạo cây, từ đó thực hiện các thao tác trên cây đã tạo.

Hình 1: Giao diện

### 2.Cấu trúc dữ liệu và giải thuật

#### 2.1. Cấu trúc dữ liệu chính trong chương trình

##### *2.1.1. Tổng quan về cấu trúc dữ liệu cây*

Cây là một tập hợp (C) gồm các phần tử gọi là nút (hay node) của cây. Trong đó có một nút đặt biệt gọi là nút gốc (root). Các nút còn lại được chia thành các tập rời nhau C1, C2, …, Cn. Trong đó, Ci cũng gọi là một cây. Mỗi nút ở cấp I sẽ quản lý một nút ở cấp i+1. Mối quan hệ này gọi là mối quan hệ cha – con. Ngược lại, các cây ở cấp i+1 đó gọi là cây con của cây ở cấp i.

*Một số khái niệm cơ bản của cấu trúc dữ liệu cây:*

* **Bậc của nút:** là số cây con của nút đó.
* **Bậc của một cây:** là bậc lớn nhất của các nút trong cây.
* **Nút gốc (Root):** nút trên cùng của cây được gọi là nút gốc. Một cây sẽ chỉ có một nút gốc và một đường xuất phát từ nút gốc tới bất kỳ nút nào khác. Nút gốc là nút duy nhất không có bất kỳ nút cha nào.
* **Nút lá:** là nút có bậc bằng 0, hay là nút không có con.
* **Nút nhánh:** hay nút trung gian là nút không phải nút lá, không phải nút con.
* **Đường:** là một dãy các nút cùng với các cạnh của một cây.
* **Truy cập:** kiểm tra giá trị của một nút khi điều khiển là đang trên một nút đó.
* **Duyệt:** duyệt qua các nút theo một thứ tự nào đó.
* **Khóa (Key):** biểu diễn một giá trị của một nút dựa trên những gì mà một thao tác tìm kiếm thực hiện trên nút.

Các loại cấu trúc dữ liệu trên cây bao gồm: cấu trúc cây tổng quát cây nhiều nhánh, cây nhị phân, cây nhị phân tìm kiếm, cây AVL.

##### *2.1.2. Cây nhị phân tìm kiếm*

Một cây tìm kiếm nhị phân (Binary Search Tree – viết tắt là BST) là một cây mà trong đó tất cả các nút đều có các đặc điểm sau:

* Cây con bên trái của một nút có khóa (key) nhỏ hơn hoặc bằng giá trị khóa của nút cha (của cây con này).
* Cây con bên phải của một nút có khóa lớn hơn hoặc bằng giá trị khóa của nút cha (của cây con này).

Một cây rỗng có thể coi là cây nhị phân tìm kiếm. Dựa vào định nghĩa chúng ta có một số nhận xét như sau:

* Trên cây nhị phân tìm kiếm, không có các nút cùng khóa.
* Các cây con trái, phải của một cây nhị phân tìm kiếm cũng là một cây nhị phân tìm kiếm.

2.2. Các thao tác chính trên cây nhị phân tìm kiếm

Các thao tác chính trên cây nhị phân tìm kiếm bao gồm:

Tạo cây, tìm kiếm trên cây, thêm một nút vào cây, xóa một nút, duyệt trên cây.

##### *2.2.1.Tạo nút*

Gồm 3 thành phần:

Data : giá trị của node

Left: vị trí của cây con trái

Right: vị trí của cây con phải

Giả sử dữ liệu giá trị truyền vào là data, ta xây dựng node như sau:

* + Thuật toán:

B1: Data=data

B2: Left=NULL

B3: Right=NULL

\*Tạo cây rỗng:

##### *2.2.2.Tạo cây ban đầu*

Từ mảng dữ liệu data được random, ta có thể tạo cây tìm kiếm nhị phân ban đầu như sau:

Ta sẽ thêm nút vào lần lượt theo thứ tự của mảng dữ liệu.

Kiểm tra Data có bé hơn key của nút tại vị trí đang xét hay không, nếu bé thì node mới sẽ nằm bên trái node đang xét, ngược lại nằm bên phải node đang xét.

Thuật toán:

B1: Tạo mảng dữ liệu.

B2: Kiểm tra đã thêm hết dữ liệu chưa.

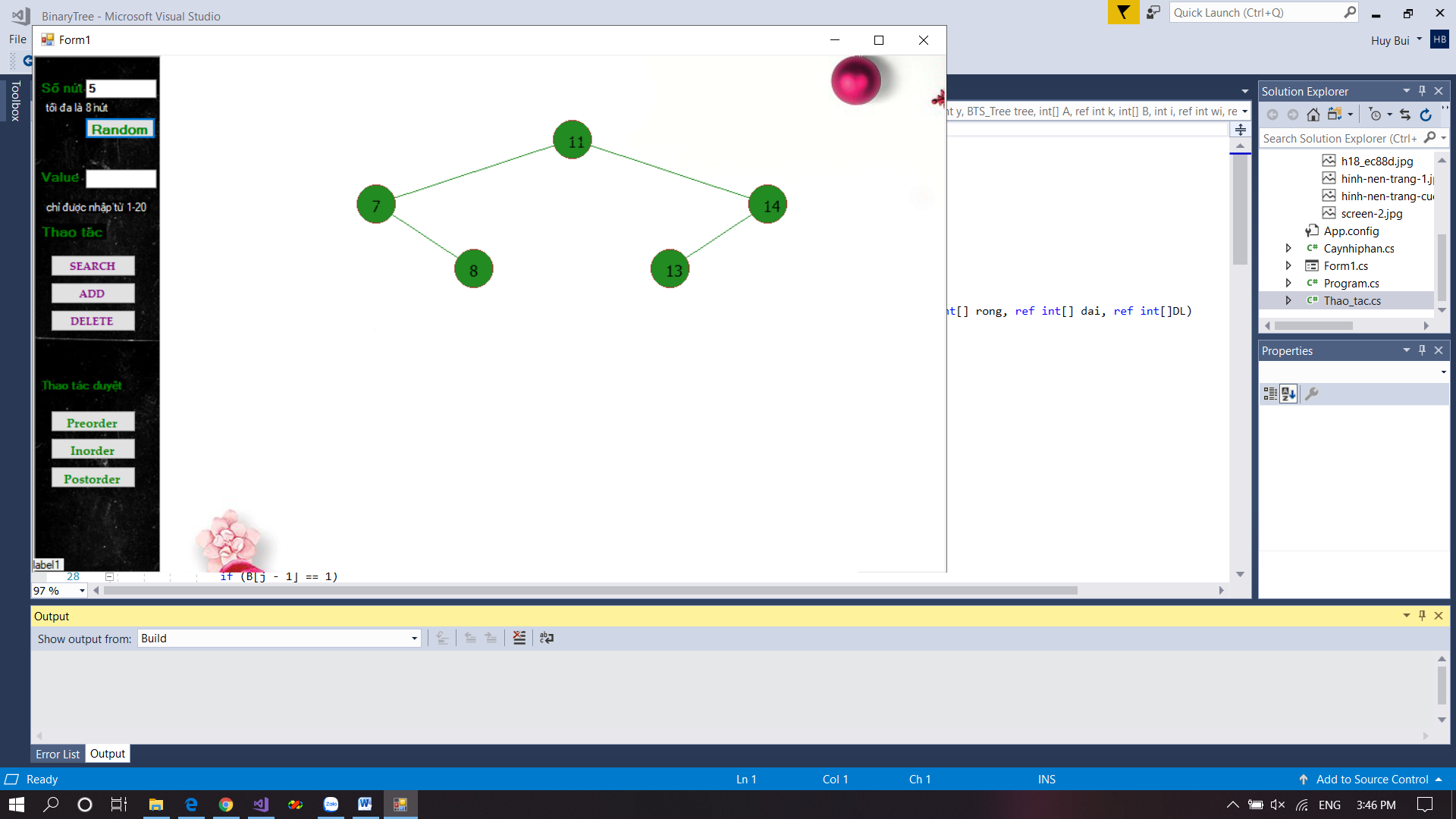
B3: Nếu chưa hết thực hiện B4. Nếu hết rồi thực hiện B6.

B4: Kiểm tra cây có rỗng không

B5: Nếu cây rỗng thì tạo cây mới và gán bằng nút ban đầu

Nếu cây không rỗng thì thêm nút vào cây theo quy tắc: data nút thêm vào bé hơn data nút đang xét thì thuộc cây con trái của cây mà nút đang xét là nút gốc, ngược lại thì thuộc cây con phải.

B6: Kết thúc.



Hình 2: Tạo cây ban đầu

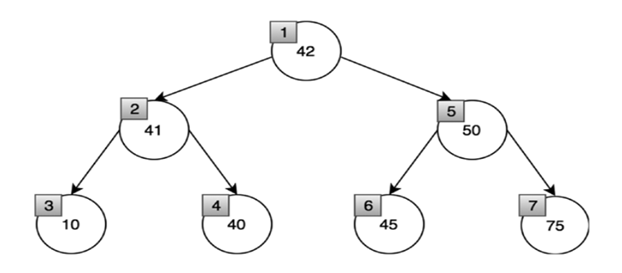
##### *2.2.3.Thao tác duyệt qua các nút trên cây nhị phân*

###### \*Duyệt theo thứ tự từ nút gốc trước ( Preoder )

Theo cách duyệt này, ta duyệt nút gốc trước, sau đó mới lần lượt duyệt đến hai cây con. Vì thế ta có 2 cách duyệt theo thứ tự duyệt gốc trước:

Duyệt gốc, duyệt cây con trái, duyệt cây con phải (Root – Left – Right)

Duyệt gốc, duyệt cây con phải, duyệt cây con trái (Root – Right – Left)

Ở đây ta thực hiện Root – Left Right :

Thuật toán:

Ta thực hiện Preoder bằng đệ quy như sau:

B1: Kiểm tra Node = NULL ?

B2: Xử lý xuất

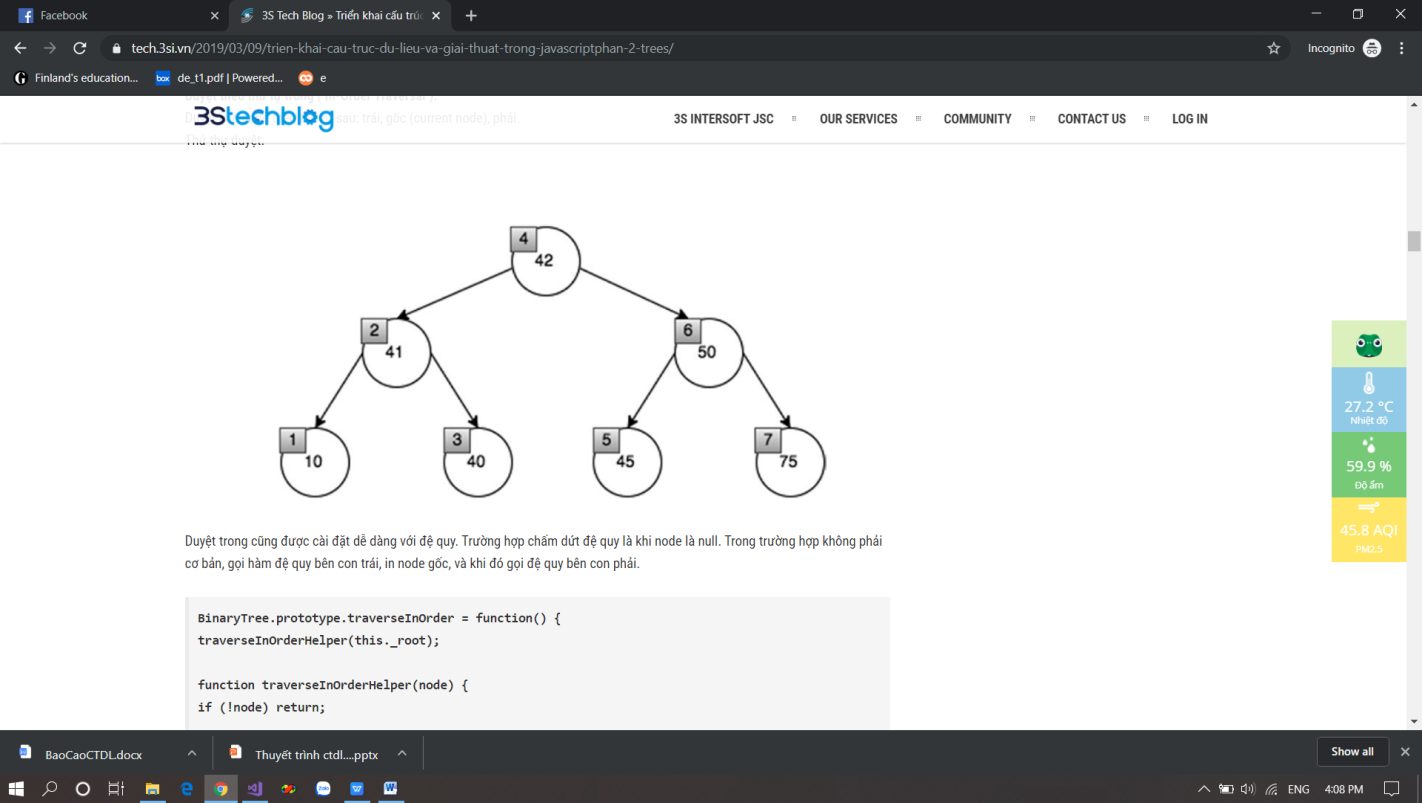
B3: Nếu Node = NULL => thực hiện B6

Hình 3: Duyệt Preoder

Nếu Node != NULL => thực hiện B4

B4: Thực hiện Preoder với cây con trái

B5: Thực hiện Preoder với cây con phải

B6: Kết thúc.

###### \*Duyệt theo thứ tự nút gốc giữa (Inorder)

Theo cách duyệt này ta sẽ duyệt một trong hai con trước rồi duyệt đến nút gốc sau đó mới duyệt đến cây con còn lại. Do đó ta cũng có 2 cách duyệt cho thứ tự nút gốc duyệt giữa.

Hình 4: Duyệt Inorder

Hình 5: Duyệt Postorder

###### \*Duyệt theo thứ tự nút gốc sau (Postorder)

Tương tự như Preoder, Postorder nút gốc sẽ được duyệt sau cùng so với 2 nút gốc của cây con.

##### *2.2.4.Các thao tác trên cây tìm kiếm nhị phân*

###### \*Tìm kiếm 1 nút trên cây (Search Node)

Giả sử chúng ta tìm kiếm trên cây có tồn tại nút có khóa key là Data hay không?

Để giải quyết vấn đề trên, ta áp dụng thuật toán tìm kiếm nhị phân vì các node trên cây nhị phân được sắp xếp theo quy tắc những node của cây con trái luôn luôn bé hơn những node ở cây con phải.

Ta sẽ dùng đệ quy để thực hiện yêu cầu trên:

Hình 6: Tìm kiếm trên cây

Thuật toán:

B1: Kiểm tra xem Node = NULL?

Nếu Node =NULL => return 0 (Không tìm được node có khóa key).

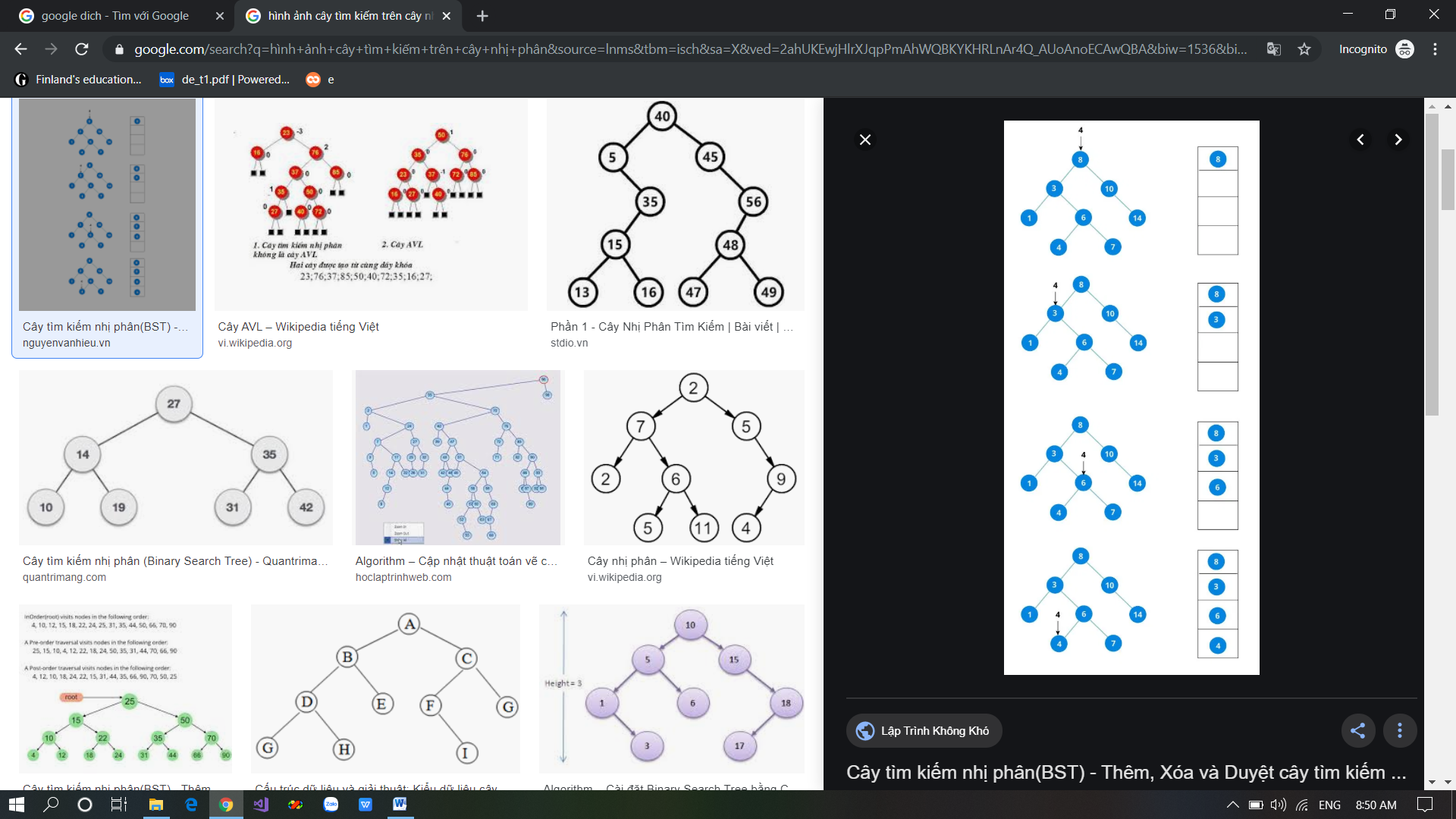
Nếu Node != NULL => thực hiện B2

B2: So sánh key với data node đang xét

Nếu key == data => return 1 (Có nghĩa là có node với khóa key trong cây).

Nếu key > data => thực hiện search trên cây con phải.

Nếu key < data => thực hiện search trên cây con trái.



Hình 7: Thêm nút trên cây

###### \*Thêm nút trên cây (Add Node)

Giả sử chúng ta cần thêm một nút có thành phần dữ liệu key là Data vào trong cây

nhị phân tìm kiếm, sao cho sau khi thêm cây vẫn là cây nhị phân tìm kiếm. Trong thao tác này ta phải áp dụng 2 giải thuật. Đầu tiên ta phải tìm kiếm vị trí cần thêm sau đó mới tiến hành thêm nút vào cây. Quá trình tìm kiếm vị trí phải tuân thủ các quy tắc trong phần tìm kiếm đã trình bày ở trên.

Ở đây, ta chỉ thực hiện thêm khi node chưa xuất hiện ở trong cây. Vì thế nếu thêm nút có trong cây ta sẽ random giá trị data khác cho phù hợp.

Thuật toán:

B1: Kiểm tra cây có rỗng không

B5: Nếu cây rỗng thì tạo cây mới và gán bằng nút ban đầu

Nếu cây không rỗng thì thêm nút vào cây theo quy tắc: data nút thêm vào bé hơn data nút đang xét thì thuộc cây con trái của cây mà nút đang xét là nút gốc, ngược lại thì thuộc cây con phải.

B6: Kết thúc.

###### \*Hủy một nút trên cây (Delete Node)

Cũng như thao tác thêm một nút vào trong cây nhị phân tìm kiếm, thao tác hủy một nút trên cây nhị phân tìm kiếm cũng phải bảo đảm cho cây sau khi hủy vẫn là cây nhị phân tìm kiếm. Đây là một thao tác không đơn giản vì nếu chúng ta không cẩn thận sẽ biến cây thành 1 rừng.

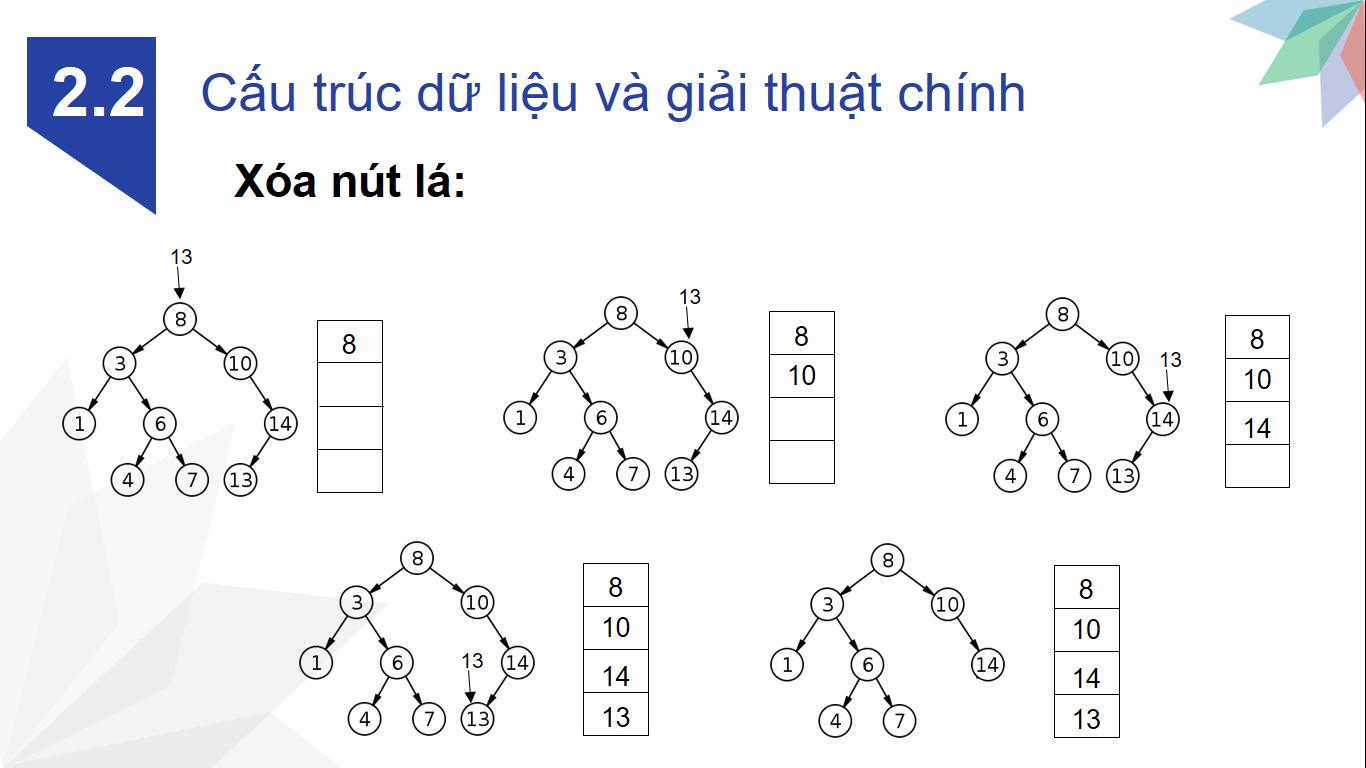
Giả sử chúng ta cần hủy nút có thành phần dữ liệu key là Data ra khỏi cây nhị phân tìm kiếm. Điều đầu tiên trong thao tác này là chúng ta phải tìm kiếm địa chỉ của nút cần hủy là DelNode, sau đó mới tiến hành hủy nút có địa chỉ là DelNode này nếu tìm thấy. Do đó thuật toán này còn được gọi là thuật toán tìm kiếm và loại bỏ trên cây. Quá trình tìm kiếm đã trình bày ở trên, ở đây chúng ta chỉ trình bày thao tác hủy khi tìm thấy nút có địa chỉ DelNode mà giá trị của nó bằng Data. Bên cạnh đó ta phải giữ lại vị trí của node cha là PredelNode.

Có 3 trường hợp hủy:

* + Hủy nút lá.
  + Hủy nút trung gian (nút có 1 cây con).
  + Hủy nút gốc (nút có 2 cây con ).

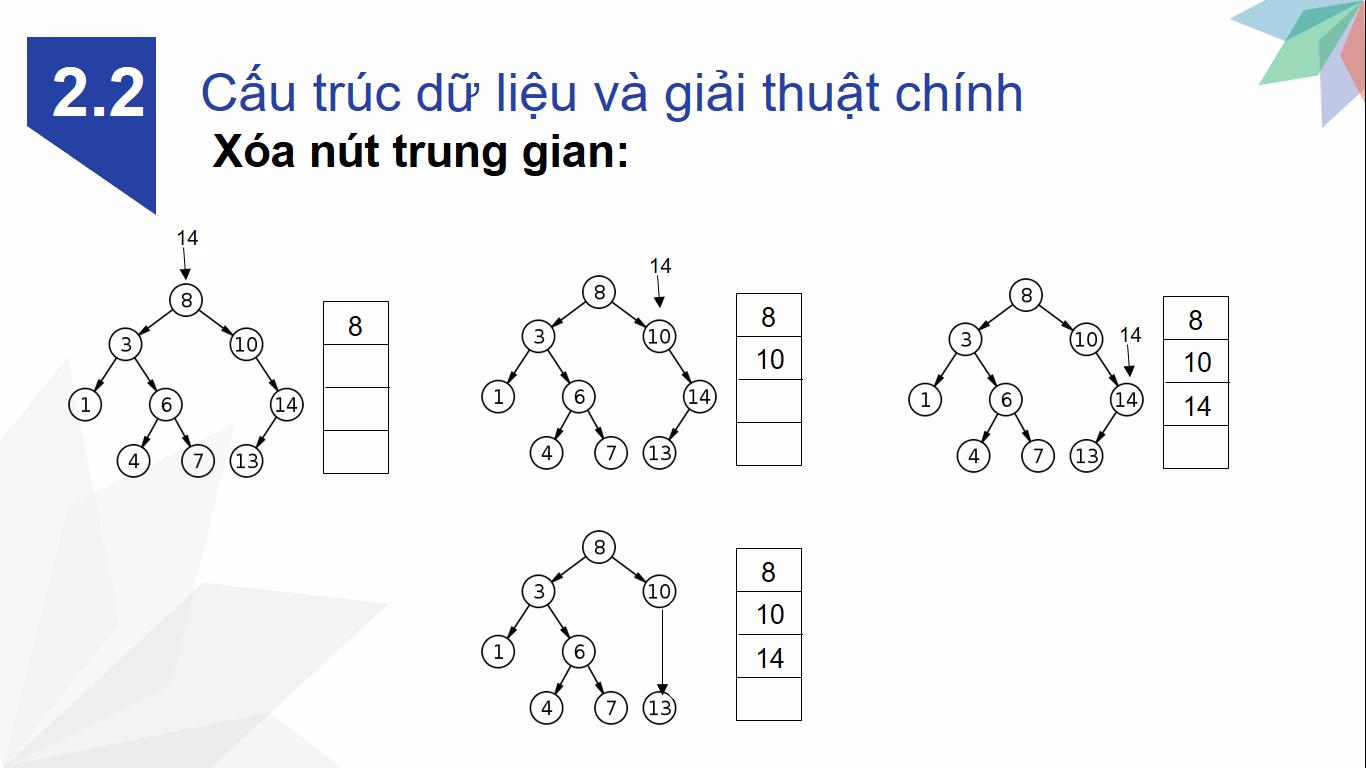
+Hủy nút lá

Đây là trường hợp đơn giản nhất trong 3 trường hợp. Chúng ta chỉ cần cắt bỏ mối quan hệ cha con giữa PredelNode và DelNode bằng cách cho DelNode.Left (nếu DelNode là nút con bên trái PredelNode) or DelNode.Right (Nếu DelNode là nút con nằm bên phải PredelNode) bằng NULL và tiến hành delete nút có địa chỉ DelNode này.



Hình 8: Hủy nút lá

+Hủy nút trung gian

Trong trường hợp này cũng khá đơn giản vì chúng ta chỉ cần chuyển mối quan hệ cha con giữa PreDelNode và DelNode thành mối quan hệ cha con giữa PredelNode và nút gốc cây con của DelNode rồi tiến hành cắt bỏ mối quan hệ cha con giữa DelNode và 1 nút gốc của cây con của nó và tiến hành hủy nút có địa chỉ DelNode.

Hình 9: Hủy nút trung gian

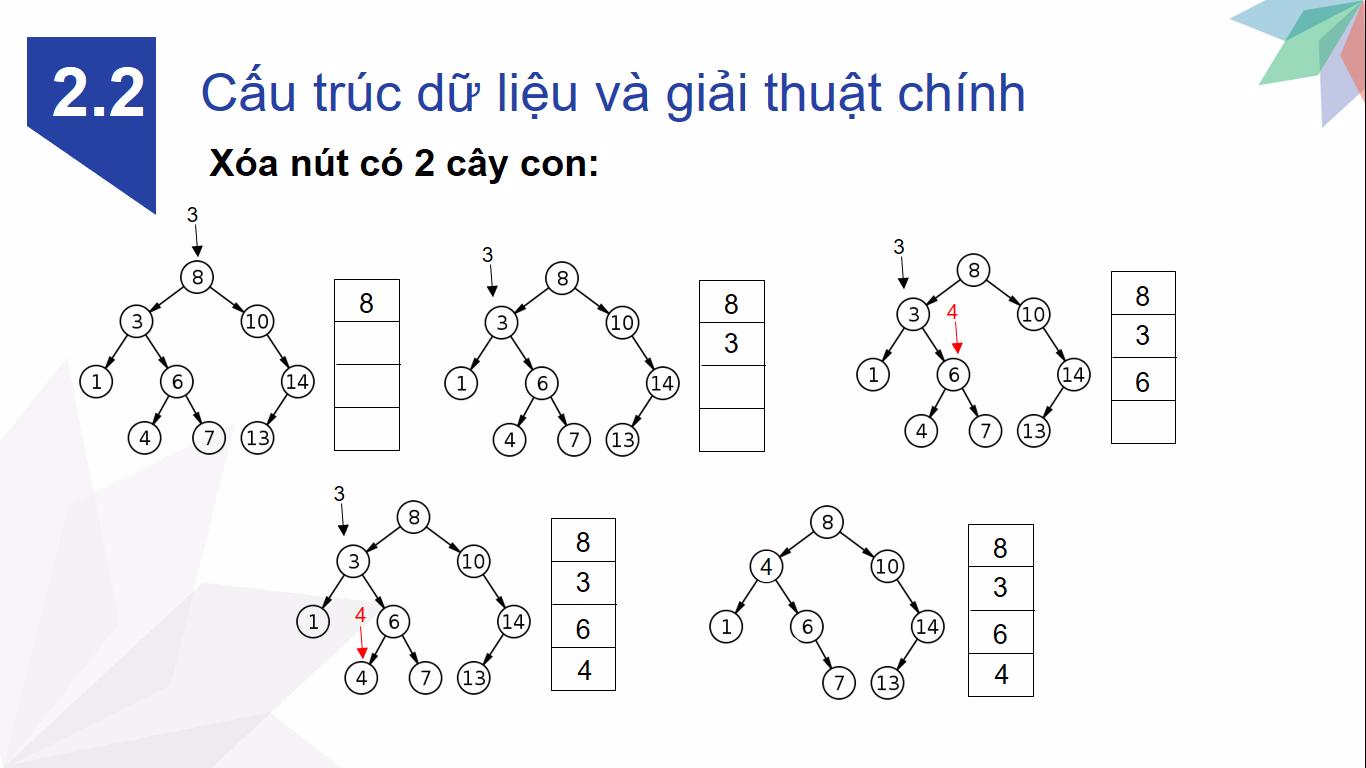
+Hủy nút có 2 cây con

Trường hợp này khá phức tạp, việc hủy được tiến hành theo 1 trong 2 cách sau:

Chuyển 2 cây con của DelNode về thành 1 cây con:

Theo phương pháp này, chúng ta sẽ chuyển cây con phải của DelNode về thành cây con phải cả cây con có nút gốc là nút phải nhất trong cây con trái của DelNode về thành cây con trái của cây con có nút gốc là nút trái nhất trong cây con phải của DelNode. Sau khi chuyển thì DelNode trở thành nút lá hoặc nút chỉ có 1 cây con và chúng ta hủy DelNode như đối với 2 trường hợp trên.

Sử dụng phần tử thế mạng:

Theo phương pháp này chúng ta sẽ không hủy nút có vị trí DelNode mà chúng ta sẽ hủy nút có địa chỉ của phần tử thế mạng của nút phải nhất trong cây con trái hay phần tử trái nhất trong cây con phải của DelNode (MLNode). Sau khi chuyển toàn bộ nội dung dữ liệu của nút thế mạng cho DelNode thì chúng ta sẽ hủy nút thế mạng như trường hợp hủy nút lá hoặc hủy nút trung gian ở trên.

Hình 10: Hủy nút có hai cây con

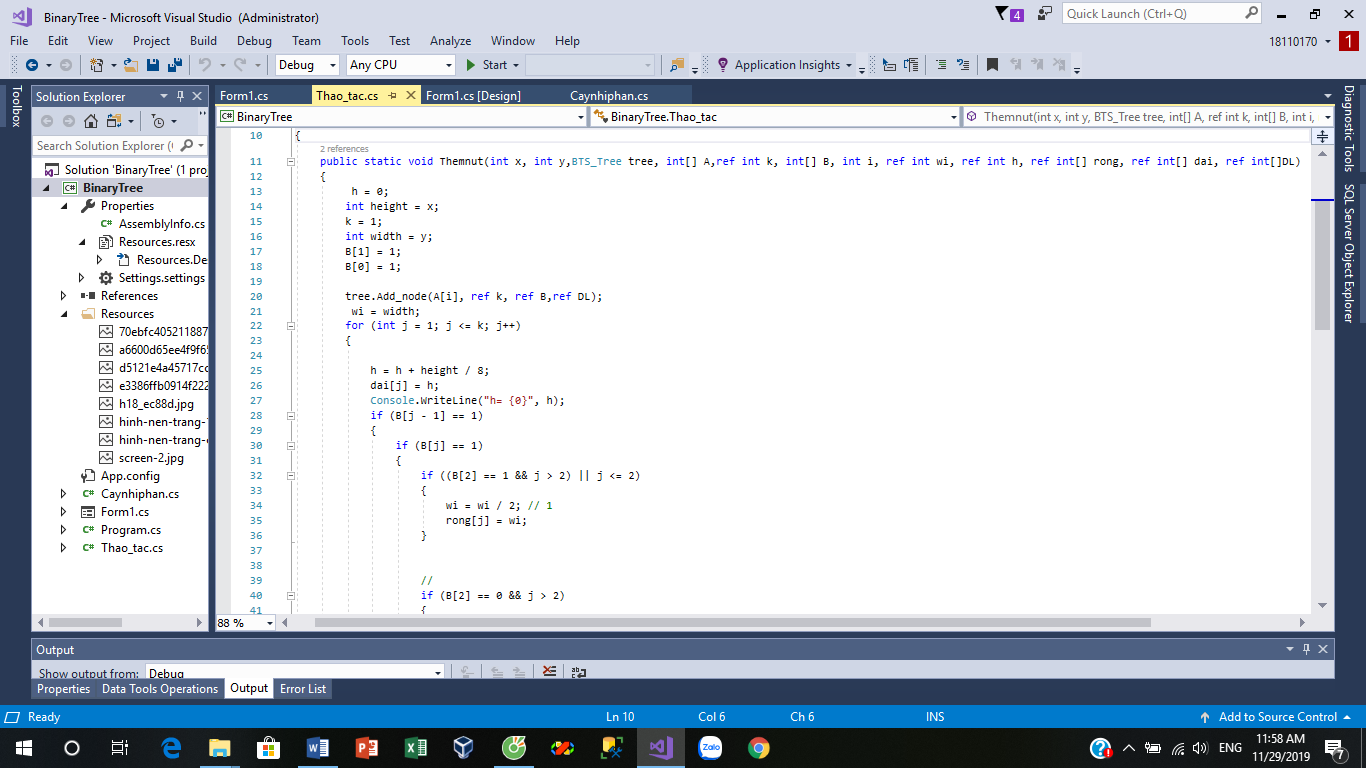
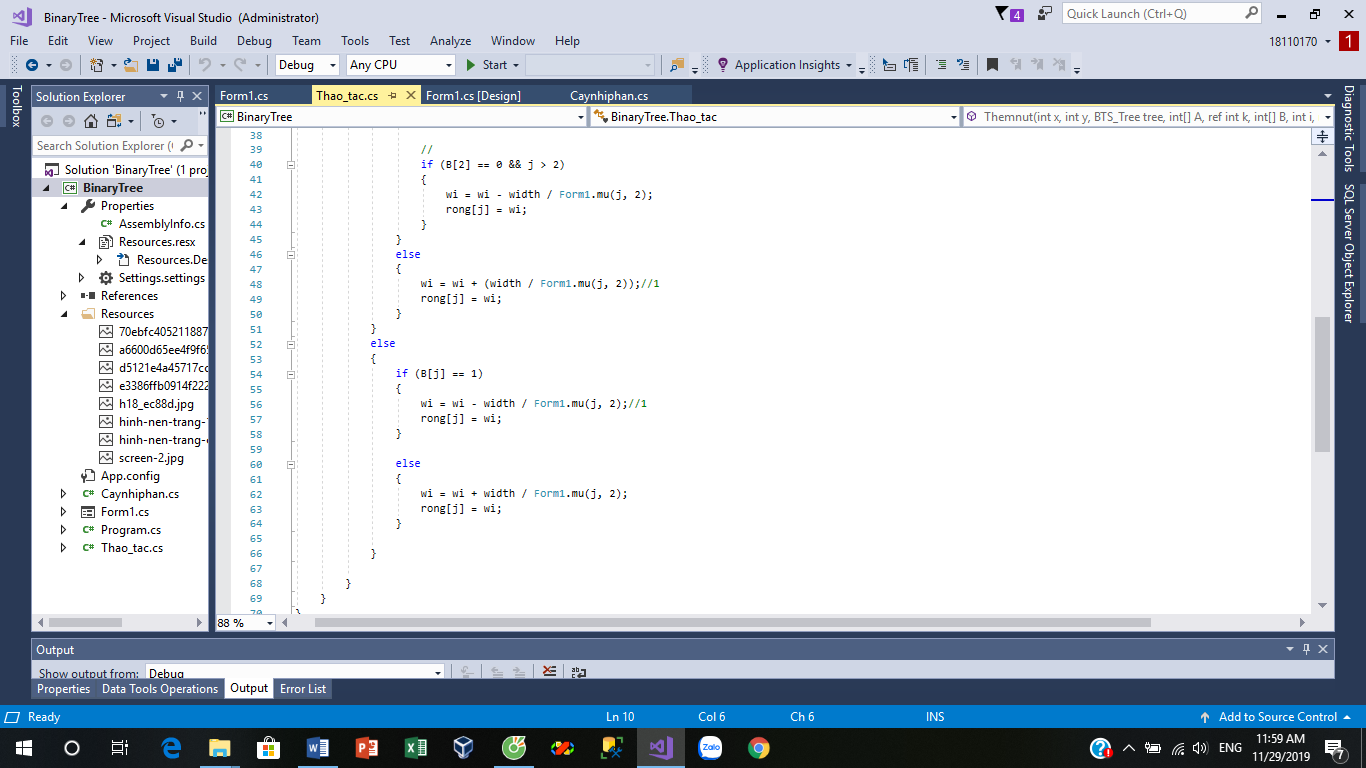
2.3.Ứng dụng của cây nhị phân vào chương trình demo

Dựa vào dữ liệu đầu ra của từng thao tác kết hợp với việc thiết kế để vẽ cây và điều chỉnh hiệu ứng động phù hợp với từng thao tác. Mang đến cảm giác dễ hiểu dễ tiếp thu.

##### *2.3.1. Tạo hiệu ứng vẽ cây*

###### **\***Xác định tọa độ của nút

Ý tưởng: Cho cây ban đầu tối đa là 8 nút. Do đó, nếu mỗi nút chỉ có 1 nút con, thì ta có cây dài 8 bậc. Từ đó ta hình dung mỗi bậc sẽ có chiều dài là h/8 (h là chiều dài khung pictureBox). Bên cạnh đó, mỗi nút cách nhau về chiều ngang theo hình thứ wi=wi/2k . Từ đó, ta có vị trí tọa độ mà nút sẽ xuất hiện.

Thực hiện:

###### \*Vẽ nút và đường nối

Ý tưởng: Dựa vào tọa độ trái phải đã tìm, sử dụng lớp Graphics của thư viện System.Drawing để tạo đối tượng nút và đối tượng đường nối.

Thực hiện:

public void eclip(int trai, int doc, int dai, int rong, int x)

{

Graphics g = pt\_cay.CreateGraphics();

g.DrawEllipse(Pens.Brown, new Rectangle(trai, doc, dai, rong));// x, y, with, heigh// quan tâm tới rộng, và dài.tham số đươc truyền vào

Brush brush = new SolidBrush(Color.ForestGreen);

g.FillEllipse(brush, trai, doc, dai, rong);

Brush br = new SolidBrush(Color.Black);

g.DrawString(Convert.ToString(x), new Font("Tahoma", 12), br, new PointF(trai + dai / 3, doc + rong / 3));

}

public void duongthang(int trai, int doc, int dai, int rong)

{

Graphics g = pt\_cay.CreateGraphics();

g.DrawLine(Pens.ForestGreen, trai, doc, dai, rong);// sử dụng drawLine để vẽ đường

}

##### *2.3.2. Tạo hiệu ứng tìm kiếm*

Vận dụng những kiến thức Search BST đã học, từ kiến thức đó, vận dụng giải thuật vào việc tìm kiếm và xác định vị trí tọa độ của nút cần tìm.

Ý tưởng:

Ở phần tạo cây, thực hiện hành động lưu lại mảng vị trí tọa độ của các nút.

Áp dụng kỹ thuật tìm kiếm để kiểm tra xem nút cần tìm có tồn tại hay không. Bên cạnh đó trả về mảng giá trị là mảng các nút theo thứ tự duyệt để tìm kiếm, nhằm mục đích dựa vào mảng đã được lưu ta có thể đưa ra vị trí của chúng nhằm bật màu đỏ lên các nút đã được duyệt trong quá trình tìm kiếm, góp phần miêu tả thêm sinh động và dễ hiểu.

Khi kết thúc quá trình tìm kiếm, nếu hàm tìm kiếm trả về giá trị 0, ta sẽ Show ra MesseageBox “Không có giá trị Value trong cây!”. Ngược lại, ta sẽ bật màu vàng lên nút cần tìm để kết thúc quá trình.

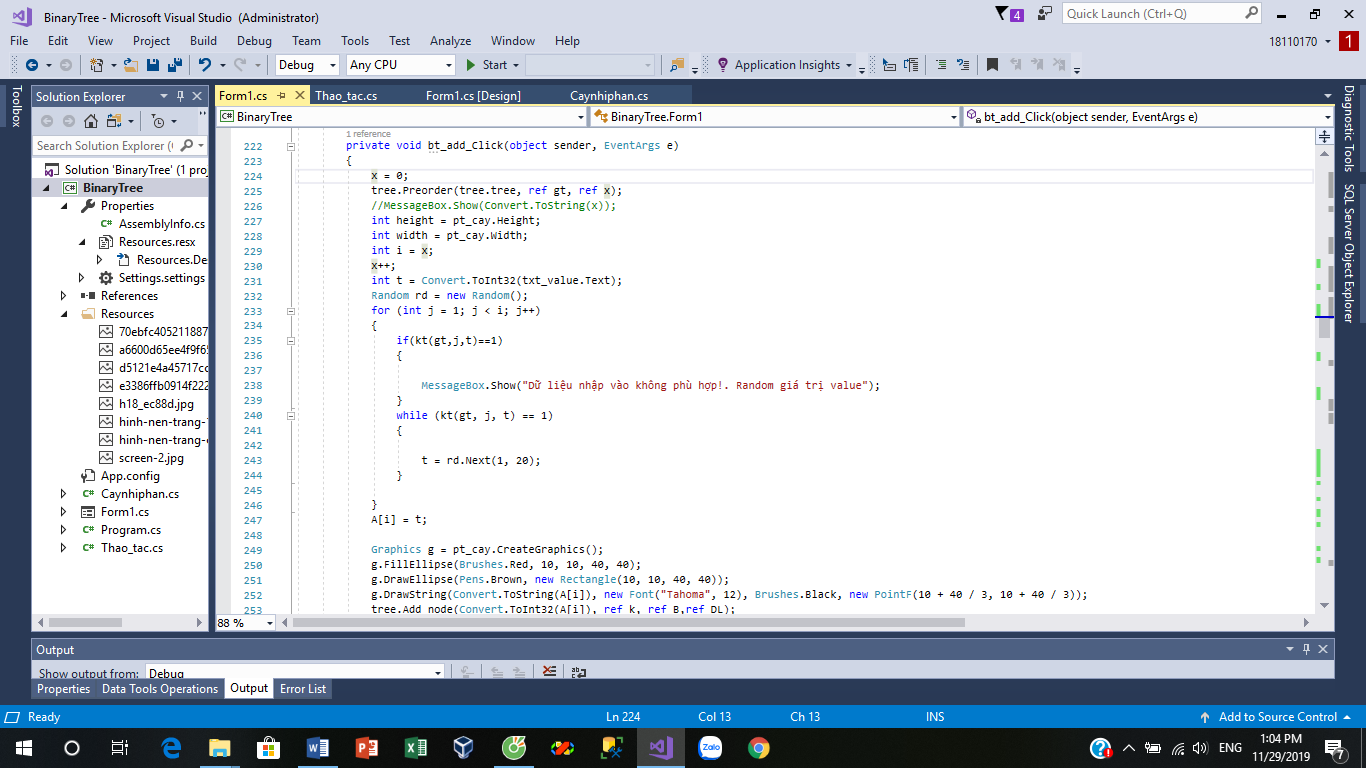
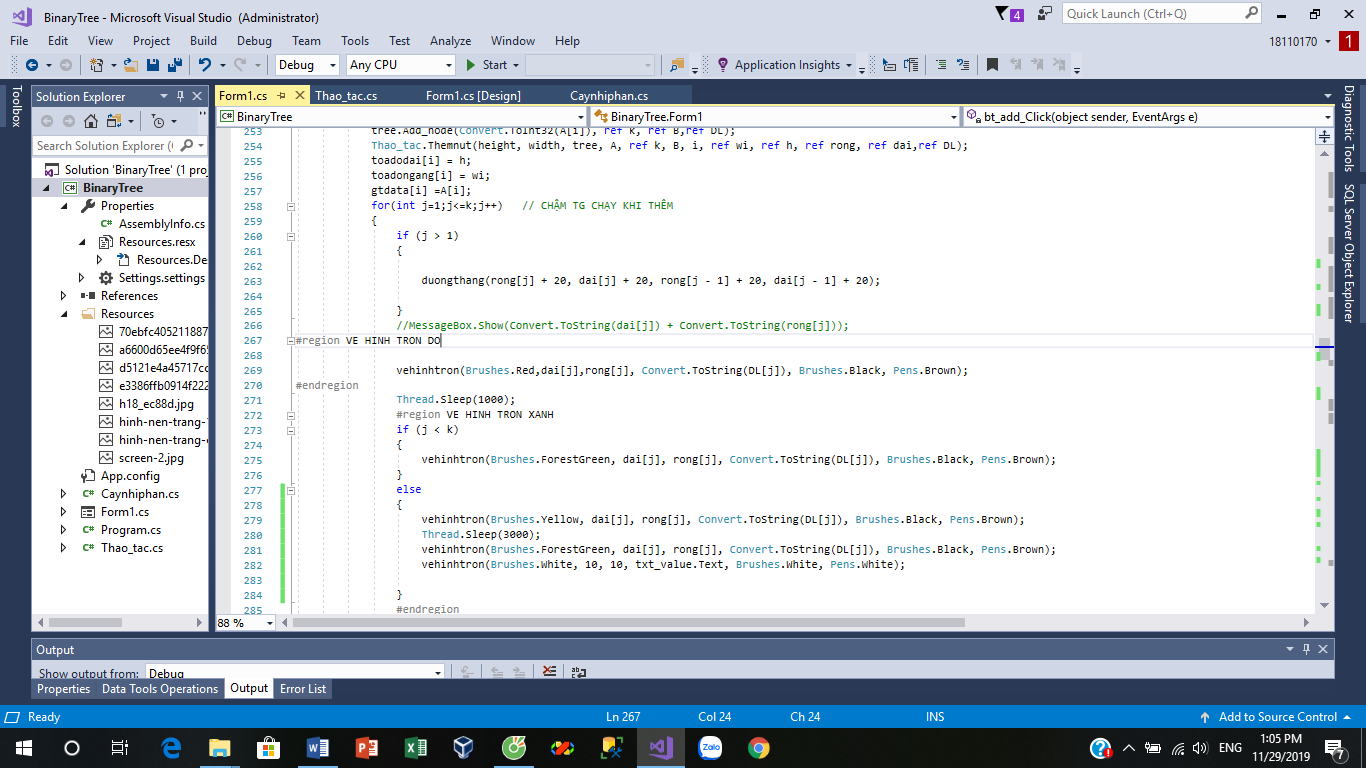
Sau khi show màu vàng sau 3s thì reset màu lại để thực hiện các hành động kế.

 Thực hiện:

##### *2.3.3.Tạo hiệu ứng thêm nút*

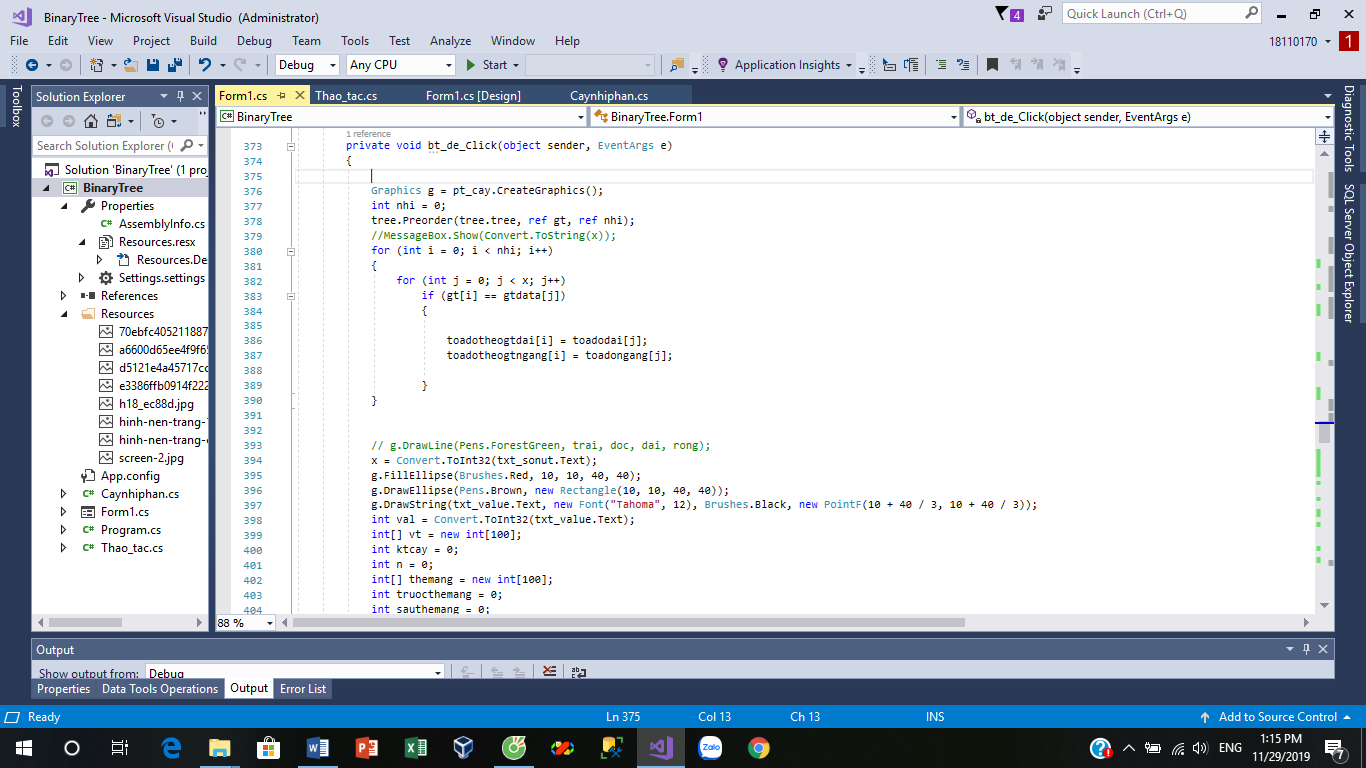
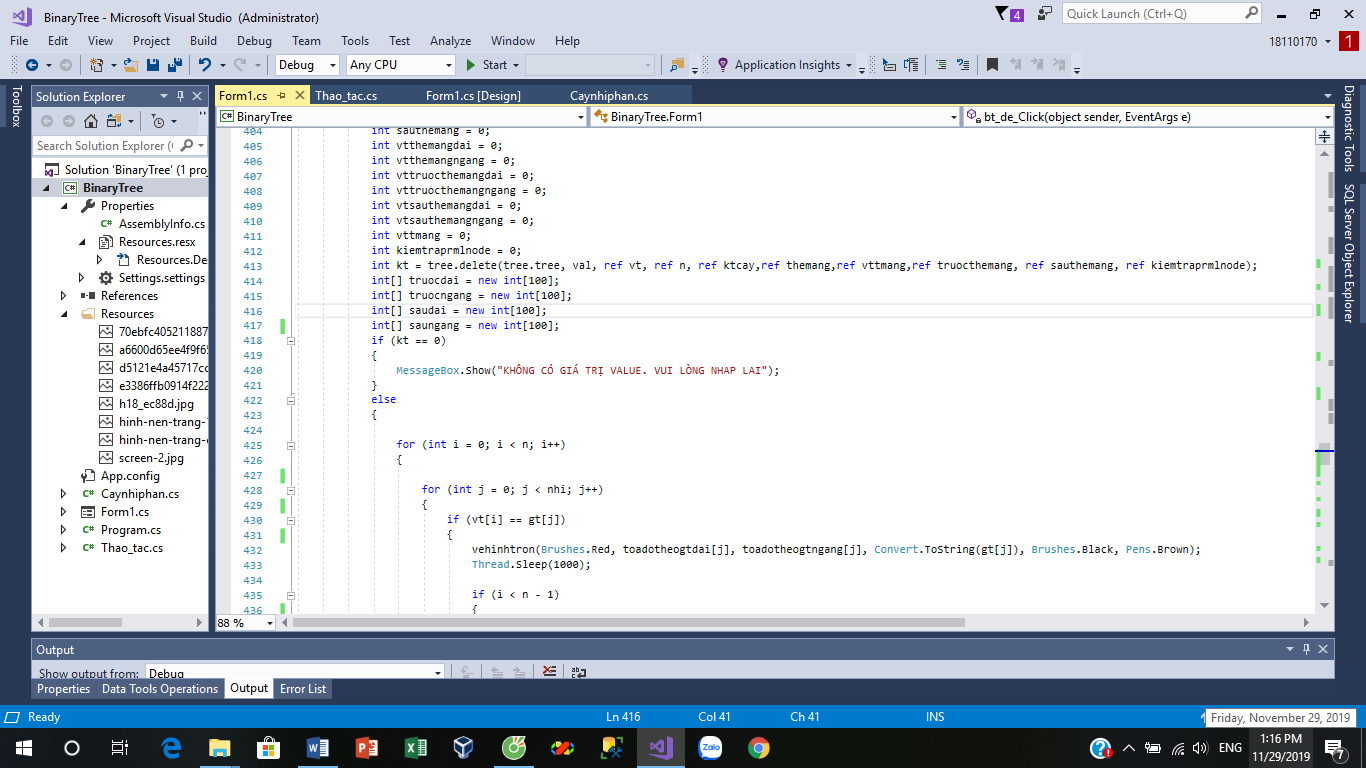
Ý tưởng: Cũng như hành động search nút. Ta áp dụng thuật toán thêm nút để tìm ra vị trí cần thêm. Sau khi thêm nút cần bổ xung nút cần thêm vào mảng lưu trữ nút ban đầu.

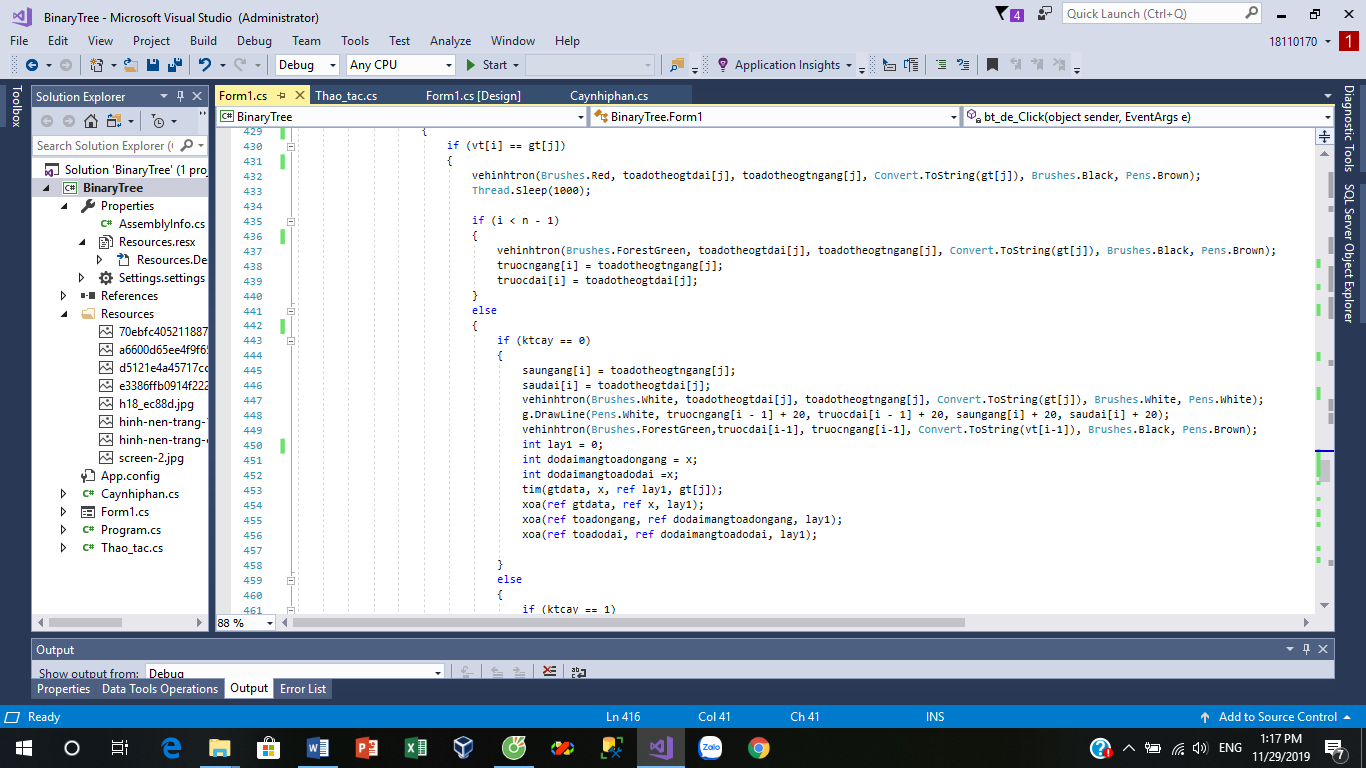
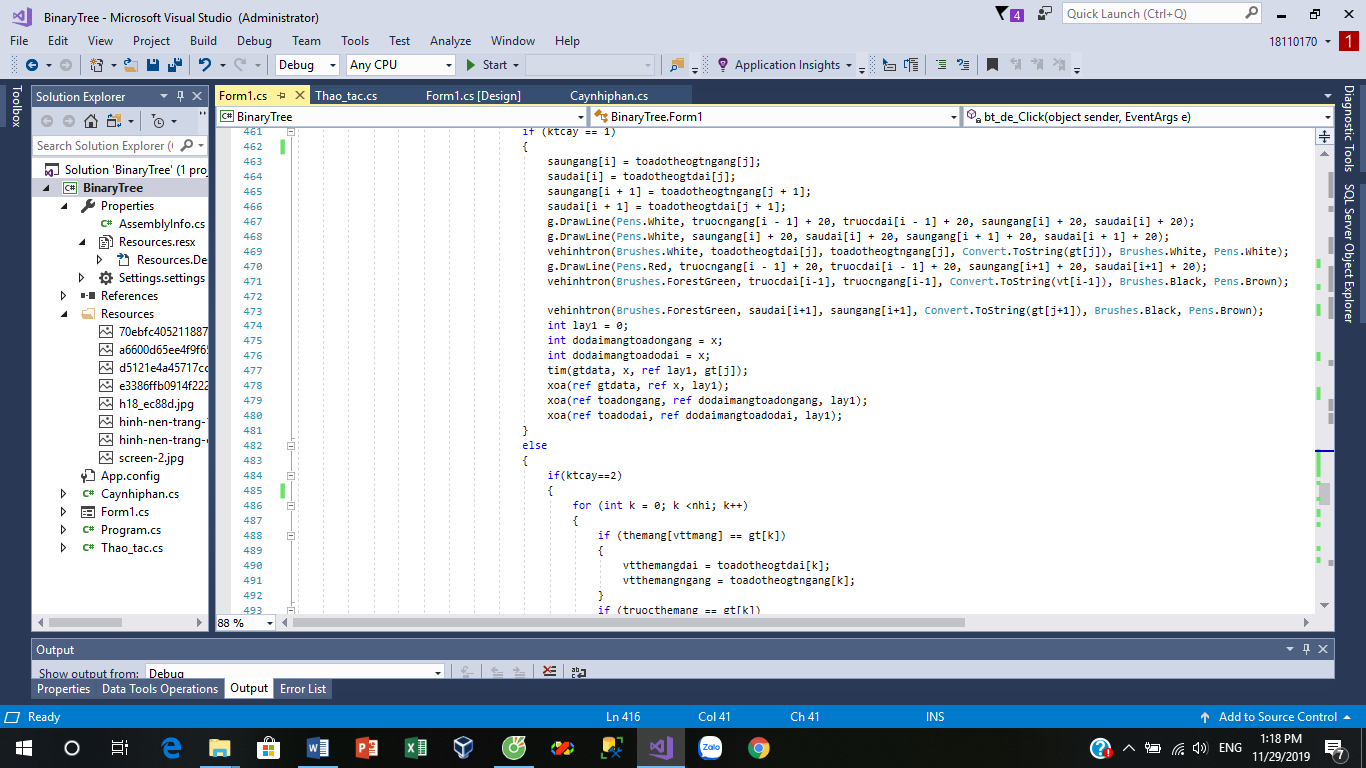
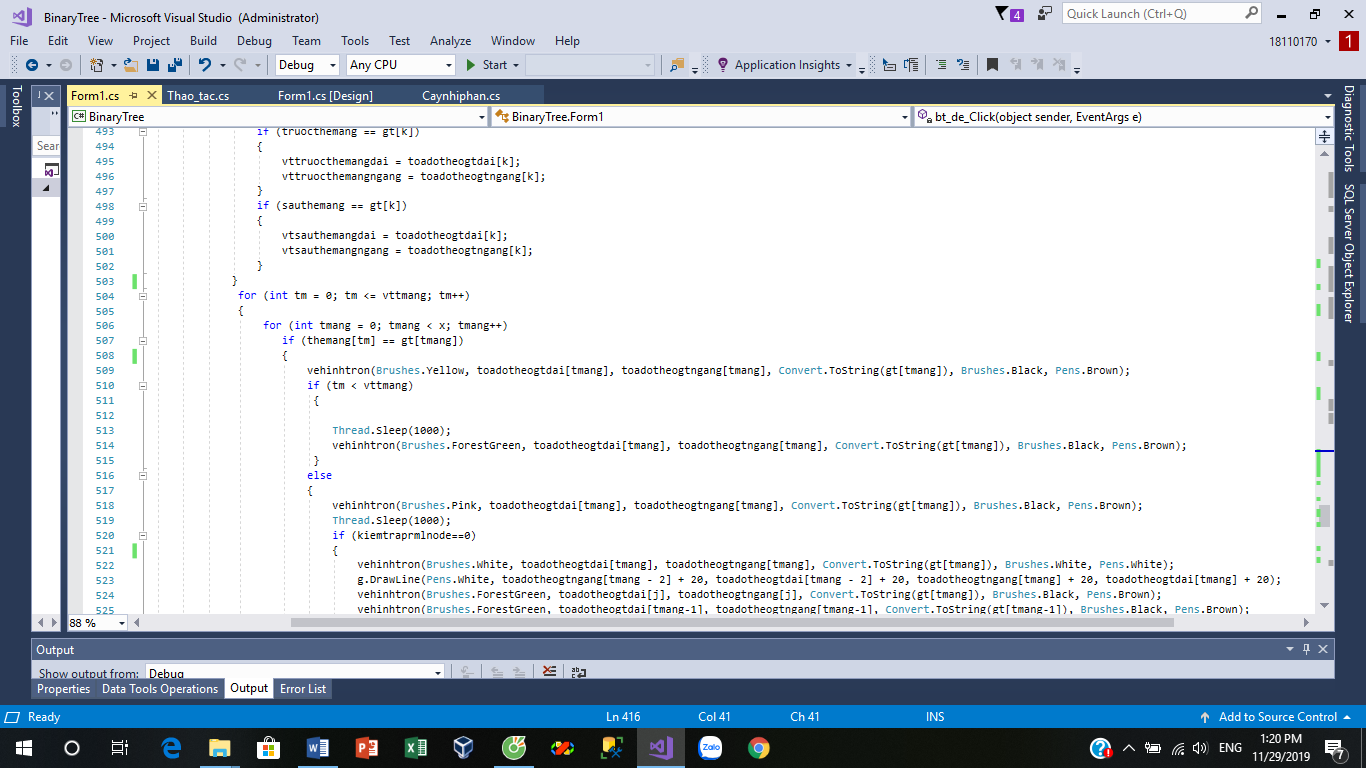
Thực hiện:



##### *2.3.4.Tạo hiệu ứng xóa nút*

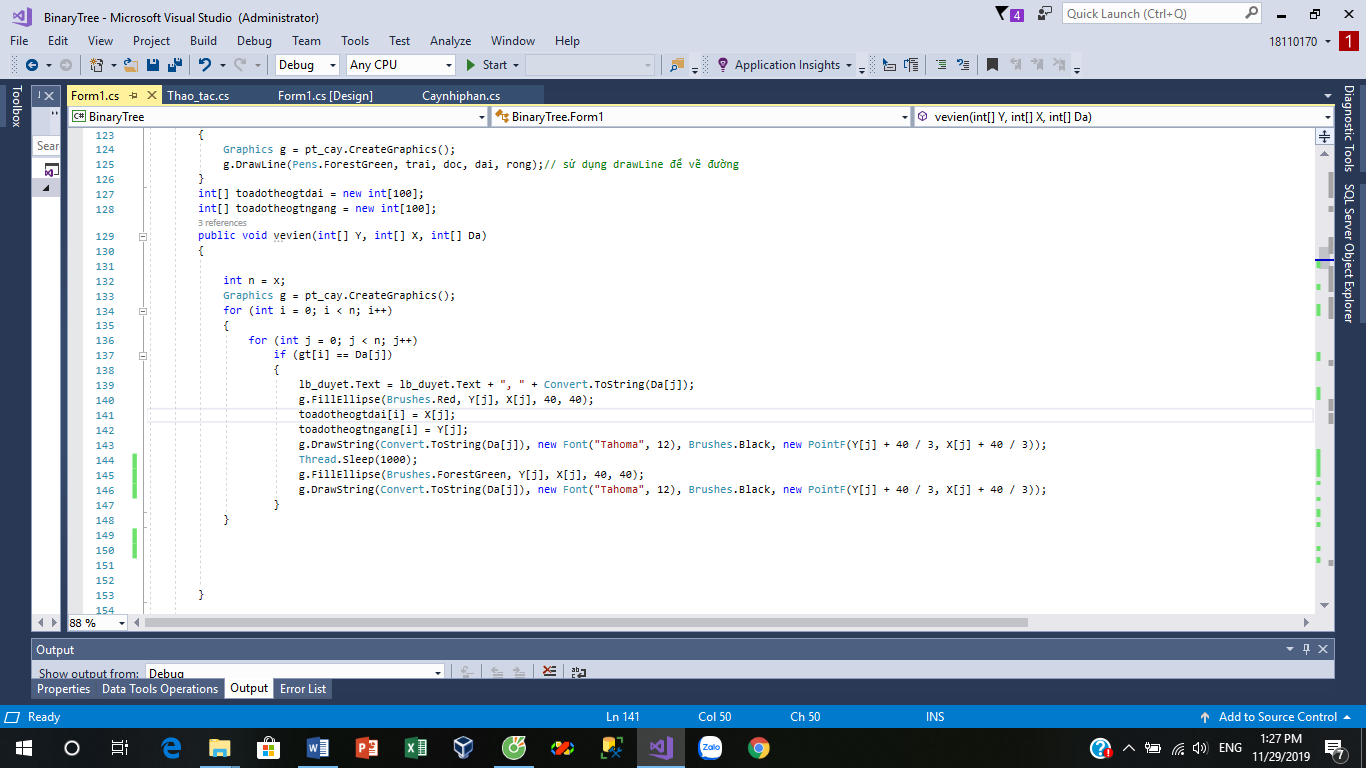
Ý tưởng: Tương tự hành động tìm kiếm, áp dụng thuật toán xóa nút để tìm ra tọa độ nút cần xóa, tọa độ nút trước và sau nút xóa. Từ đó, chuyển màu nút cần xóa sang màu trắng ẩn dưới màu nền, dùng tọa độ nút trước và sau tạo thành đường nối. Trường hợp nút có 2 con. Cần lưu thêm tọa độ nút thế mạng, và bên cạnh đó bật màu khác cho hành động tìm nút thế mạng để phân biệt rõ đây là quá trình tìm nút thế mạng. Sau khi tìm nút thế mạng, dựa vào vị trí của nút xóa ban đầu, chỉnh sửa nội dung trong nút, cùng với việc thực hiện lại hành động xóa như xóa 1 nút đã nêu. Sau khi kết thúc hành động, cần phải reset lại mảng lưu trữ data ban đầu.

Thực hiện:



##### *2.3.5.Tạo hiệu ứng duyệt*

Ý tưởng: Tương tự như trên, trong quá trình duyệt tìm ra được mạng nút theo thứ tự duyệt, từ đó, so sánh với mảng phần tử ban đầu để đưa ra vị trí xóa. Bật màu nút xóa khi chúng được duyệt đến, sau đó 1s thì bật lại màu xanh ban đầu. Sau khi kết thúc quá trình duyệt, đưa ra label\_duyệt là thứ tự các data được duyệt.

Thực hiện:

### 3.Cài đặt và kiểm thử

3.1.Các trường hợp kiểm thử

##### *3.1.1.Random tạo cây ban đầu*

Đầu tiên ta phải nhập vào số nút cần tạo cây vào ô textbox txt\_sonut trong điều kiện 0 < x <9.

Nếu chúng ta nhập ngoài phạm vi điều kiện chương trình sẽ yêu cầu nhập lại.

Sau khi chọn nút Random cây sẽ được tạo ra ở phía trên màn hình.

##### *3.1.2.Thực hiện Preoder*

Click vào ô Preoder sau đó từng bước của thao tác duyệt Preoder sẽ thực hiện trên Form, tại ví trí đang duyệt thì nút sẽ hiện đỏ sau khi duyệt xong sẽ quay về màu xanh.

##### *3.1.3.Thực hiện Inorder*

Click vào ô Inorder sau đó từng bước của thao tác duyệt Preoder sẽ thực hiện trên Form, tại ví trí đang duyệt thì nút sẽ hiện đỏ sau khi duyệt xong sẽ quay về màu xanh.

##### *3.1.4.Thực hiện Postorder*

Click vào ô Postorder sau đó từng bước của thao tác duyệt Preoder sẽ thực hiện trên Form, tại ví trí đang duyệt thì nút sẽ hiện đỏ sau khi duyệt xong sẽ quay về màu xanh.

##### *3.1.5.Thực hiện Search*

Nhập giá trị cần tìm vào textbox txt\_value

Sau khi chọn nút Add thì từng bước của thao tác duyệt sẽ được hiển thị trên Form.

Khi thêm thành công nút sẽ chuyển sang màu vàng.

##### *3.1.6.Thực hiện Delete*

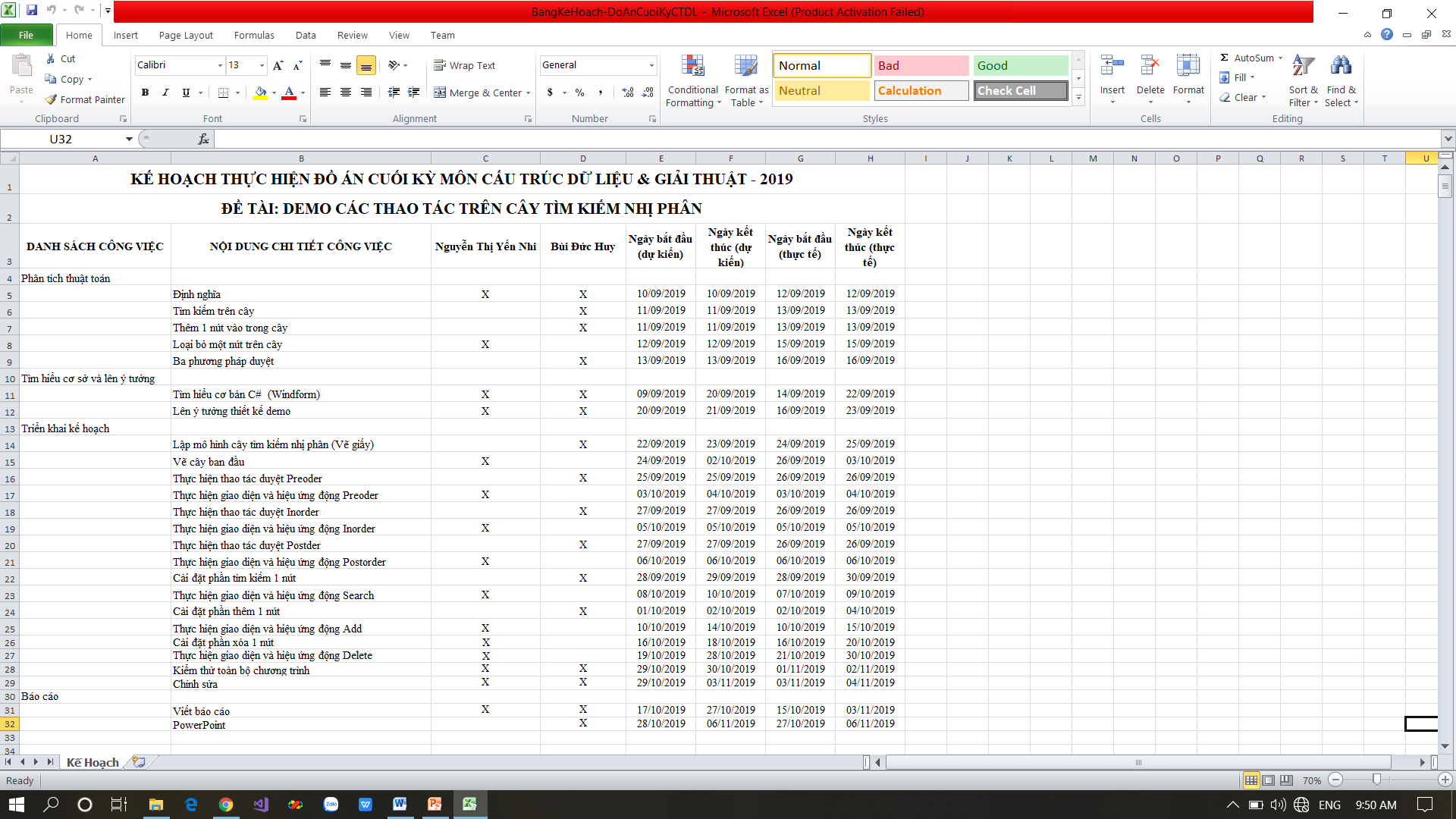
Tương tự như bước thêm …..

##### *3.1.7.Lỗi chưa giải quyết*

Sau khi xóa 1 nút thì việc thêm nút gặp khó khăn. Vị trí của nút bị cập nhật lại khiến cây bị rối.

Nhận dạng vị trí còn sai, khi xóa 2 nút còn gặp 1 số trường hợp chưa thê khắc phục.

## *III. Phân công công việc:*



Bảng 1: Phân công công việc

## *IV. Đánh giá:*

4.1.Đánh giá mức độ hoàn thành

Trong đồ án này, nhóm thực hiện đã tìm hiểu được các kiến thức về cây nhị phân tìm kiếm, cách tổ chức dữ liệu trên cây và các thao tác trên cây như tìm kiếm, thêm và xóa nút trên cây. Áp dụng vào phần demo các thao tác trên cây tìm kiếm nhị phân. Nhìn chung các thuật toán được áp dụng một cách linh hoạt để có thể hoàn thành bản demo

Tuy nhiên, sản phẩm vẫn chỉ là một ứng dụng mô phỏng các thuật toán trên cây nhị phân. Để có thể đem sản phẩm vào thực tiễn để sử dụng, chúng em cần một quá trình để cải tiến và phát triển thêm.

4.2.Các khó khăn

Trong quá trình thực hiện đồ án, nhóm đã gặp không ít những khó khăn. Bởi vì đây là lần đầu tiên lập trình một ứng dụng, nhóm phải bắt đầu tìm hiểu tất cả từ con số 0. Từ tìm kiếm các tài liệu về cây nhị phân tìm kiếm đến việc áp dụng cấu trúc này trên một ngôn ngữ lập trình mới là C#. Bên cạnh đó, áp lực về thời gian và lượng công việc là những khó khăn mà nhóm đã gặp phải. Tuy nhiên nhóm chúng em đã vượt qua tất cả khó khăn để hoàn thành sản phẩm bằng tất cả khả năng của mình.

## *V.Kết luận:*

5.1.Ưu điểm

Chương trình dùng để hỗ trợ việc giảng dạy được thuận tiện hơn. Giúp cho học sinh sinh viên và người tìm hiểu có thể hiểu rõ hơn về nguyên tắc hoạt động của cây nhị phân tìm kiếm. Các phương pháp duyệt trên cây nhị phân tìm kiếm.

5.2.Nhược điểm

Còn một vài chỗ chưa xử lí được như khi xóa một nút thì còn đường vẽ tạo cây, chưa thể thêm nút sau khi xóa nút và chưa thực hiện được xóa toàn bộ cây.

5.3.Hướng phát triển

Thời gian tiếp theo, nhóm chúng em sẽ tiếp tục cải thiện và phát triển một số tính năng mới cho bản demo được hoàn hảo hơn

Nhóm sẽ khắc phục các lỗi còn hạn chế như: khắc phục exception thêm sau khi xóa, thực hiện detele all, hình thức bắt mắt hơn, bổ sung thêm demo cây AVL.

# TÀI LIỆU THAM KHẢO

https://yinyangit.wordpress.com/2011/01/16/algorithm-cai-d%E1%BA%B7t-binary-search-tree-b%E1%BA%B1ng-c/