**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---🙠**🕮**🙢---**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI:**

**VIẾT CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN DUYỆT CÂY NHỊ PHÂN TRÊN MÔI TRƯỜNG ĐỒ HOẠ**

***Giảng Viên Hướng Dẫn : TS. Nguyễn Đình Hưng***

***Người Thực Hiện : Nguyễn Quốc Khánh***

***Mã Số Sinh Viên : 62130853***

***Lớp : 62.CNTT-2***

**MỤC LỤC**

[**LỜI CẢM ƠN 3**](#_Toc123412358)

[**CHƯƠNG I: GIỚI THIỆU 3**](#_Toc123412359)

[**1. Tổng Quan Về Cây (Tree) 3**](#_Toc123412360)

[**2. Cây Nhị Phân (Binary Tree) 3**](#_Toc123412361)

[**3. Duyệt Cây 3**](#_Toc123412362)

[**4. Yêu Cầu Đề Tài 4**](#_Toc123412363)

[**3.1 Ngôn Ngữ Lập Trình C# 4**](#_Toc123412367)

[**3.2 Phần Mềm Microsoft Visual Studio 5**](#_Toc123412368)

[**3.3 Một Số Thư Viện Sử Dụng 5**](#_Toc123412369)

[**CHƯƠNG II: CƠ SỞ LÝ THUYẾT 6**](#_Toc123412370)

[**1. Tìm hiều ngôn ngữ lập trình C# 6**](#_Toc123412371)

[**1.1 Cơ bản về ngôn ngữ lập trình C# 6**](#_Toc123412372)

[**1.2 Kiểu dữ liệu 6**](#_Toc123412373)

[**1.3 Câu lệnh If và Else, 6**](#_Toc123412374)

[**1.4 Tính đóng gói trong C# 6**](#_Toc123412375)

[**1.5 Phương thức (Hàm) 7**](#_Toc123412376)

[**1.6 Lớp (Class) 7**](#_Toc123412377)

[**1.7 Kế thừa. 7**](#_Toc123412378)

[**2. Tìm hiểu Winform 8**](#_Toc123412379)

[**2.1 Tổng quan. 8**](#_Toc123412380)

[**2.2 Cơ bản về Winform 8**](#_Toc123412381)

[**3. Giải Thuật Duyệt Cây 8**](#_Toc123412382)

[**3.1 Duyệt Trước 8**](#_Toc123412383)

[**3.2 Duyệt Giữa 9**](#_Toc123412384)

[**3.3 Duyệt Sau 9**](#_Toc123412385)

[**CHƯƠNG III: CÀI ĐẶT CHƯƠNG TRÌNH 11**](#_Toc123412386)

[**1. Giao Diện 11**](#_Toc123412387)

[**2. Các Lớp và một số hàm bên trong 11**](#_Toc123412388)

[**2.1 Form1.cs 11**](#_Toc123412389)

[**2.2 Node\_Tree.cs 16**](#_Toc123412390)

[**2.3 B\_Tree.cs 22**](#_Toc123412391)

[**TÀI LIỆU THAM KHẢO 26**](#_Toc123412392)

**LỜI CẢM ƠN**

Trước hết, em muốn gửi lời cảm ơn chân thành và sâu sắc đến thầy Nguyễn Đình Hưng người đã trực tiếp quan tâm và hướng dẫn trong thời gian qua, cũng như các thầy cô Khoa Công Nghệ Thông Tin của Trường Đại Học Nha Trang đã tạo điều kiện, hỗ trợ và giúp đỡ chúng em trong quá trình học tập và nghiên cứu đề tài này.

Do kiến thức còn hạn chế và thời gian thực hiện ngắn nên bài báo cáo của em còn nhiều thiếu sót. Kính mong sự góp ý của quý thầy cô.

**CHƯƠNG I: GIỚI THIỆU**

## **Tổng Quan Về Cây (Tree)**

Cây là một cấu trúc dữ liệu gồm một tập hữu hạn các nút, giữa các nút có một quan hệ phân cấp gọi là quan hệ “cha – con”. Có một nút đặc biệt gọi là gốc (root).

Không giống như Mảng (Array) hay Danh Sách Liên Kết (Linked List), là các cấu trúc dữ liệu tuyến tính, Cây (Tree) là cấu trúc dữ liệu phi tuyến tính (dạng phân cấp).

Cấu trúc Cây (Tree) thường được dụng để lưu trữ các thông tin dưới dạng một hệ thống phân cấp tự nhiên.

Ví dụ: Cấu trúc thư mục trên đĩa cũng có cấu trúc cây, thư mục gốc có thể coi là gốc của cây đó với các cây con là các thư mục con và tệp nằm trên thư mục gốc.

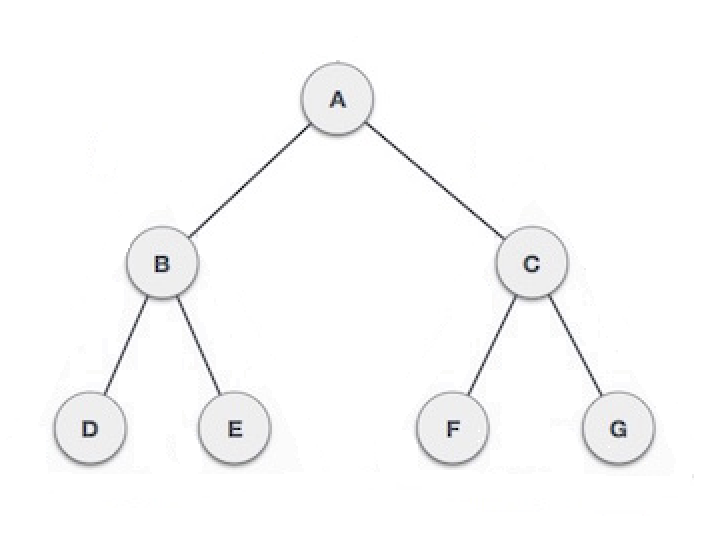
## **Cây Nhị Phân (Binary Tree)**

Cây nhị phân là một dạng quan trọng của cấu trúc cây. Nó có đặc điểm là mọi nút trên cây chỉ có tối đa hai nhánh con. Với một nút thì người ta cũng phân biệt cây con trái và cây con phải của nút đó. Cây nhị phân là cây có tính đến thứ tự của các nhánh con.

## **Duyệt Cây**

Duyệt cây là một tiến trình để truy cập tất cả các nút của một cây và cũng có thể in các giá trị của các nút này. Bởi vì tất cả các nút được kết nối thông qua các cạnh (hoặc các liên kết), nên chúng ta luôn luôn bắt đầu truy cập từ nút gốc. Do đó, chúng ta không thể truy cập ngẫu nhiên bất kỳ nút nào trong cây.

Có ba phương thức có thể sử dụng để duyệt một cây:

* **Duyệt Trước (Pre-order Traversal)** : nút gốc được duyệt đầu tiên, sau đó sẽ duyệt cây con bên trái và cuối cùng sẽ duyệt cây con bên phải:
  + Bước 1: Truy cập nút gốc và duyệt nút gốc
  + Bước 2: Duyệt các cây con bên trái một cách đệ quy.
  + Bước 3: Duyệt các cây con bên phải một cách đệ quy.

Ví dụ: A là nút gốc được duyệt đầu tiên, tiếp đến cây con bên trái A là B, duyệt B sau đó tiếp tục duyệt tiền thứ tự ta được kết quả sau khi duyệt: A B D E C F G.

* **Duyệt Giữa (In-order Traversal)**: cây con bên trái được truy cập đầu tiên, sau đó là nút gốc và sau đó là cây con bên phải:
  + Bước 1: Duyệt các cây con bên trái một cách đệ quy.
  + Bước 2: Truy cập nút gốc và duyệt nút gốc
  + Bước 3: Duyệt các cây con bên phải một cách đệ quy.

Ví dụ: A là nút gốc di chuyển đến cây con bên trái là B, đến B tiếp tục duyệt trung thứ tự đi đến D. Kết quả sau khi duyệt là: D B E A F C G

* **Duyệt Sau (Post-order Traversal)**: đầu tiên duyệt cây con bên trái, sau đó sẽ duyệt cây con bên phải và cuối cùng là duyệt nút gốc:
  + Bước 1: Duyệt các cây con bên trái một cách đệ quy.
  + Bước 2: Duyệt các cây con bên phải một cách đệ quy.
  + Bước 3: Truy cập nút gốc và duyệt nút gốc

Ví dụ: A là nút gốc di chuyển đến cây con bên trái là B, đến B tiếp tục duyệt hậu thứ tự đi đến D. Kết quả sau khi duyệt là: D E B F G C A

## **Yêu Cầu Đề Tài**



### Ngôn Ngữ Lập Trình C#

Ngôn ngữ C# (hay C sharp) là một được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000. C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và kết thừa một số khả năng của C++ và Java.

C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), phát triển game, ứng dụng Web, ứng dụng Mobile trở nên rất dễ dàng.

C# hiện là một trong những công cụ lập trình phổ biến nhất thế giới, có cộng đồng phát triển lớn, hướng đối tượng (cho phép tái sử dụng mã nguồn, giảm chi phí phát triển).

### Phần Mềm Microsoft Visual Studio

Phần mềm Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Microsoft Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới ", được dùng để lập trình C++ và C# là chính.

Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

### Một Số Thư Viện Sử Dụng

#### **System.Windows.Forms**

Windows Forms (hay WinForms) là thư viện đồ họa (GUI) mã nguồn mở và miễn phí được bao gồm như một phần của Microsoft.NET Framework hoặc Mono Framework, cung cấp nền tảng để viết các ứng dụng khách phong phú cho máy tính để bàn, máy tính xách tay và máy tính bảng.

#### **System.Drawing**

Lớp đồ họa cung cấp các phương pháp để vẽ đến thiết bị hiển thị. Các lớp như Hình chữ nhật và điểm đóng gói các nguyên thủy GDI +. Lớp Pen được sử dụng để vẽ các đường thẳng, trong khi các lớp có nguồn gốc từ lớp trừu tượng Brush được sử dụng để lấp đầy nội thất của các hình dạng.

#### **System.Threading**

System.Threading trong .NET Framework cung cấp các lớp và các đối tượng cho việc tạo và quản lý các luồng (thread) trong ứng dụng của bạn. Luồng là một đoạn mã chạy song song với các đoạn mã khác trong chương trình của bạn.

**CHƯƠNG II: CƠ SỞ LÝ THUYẾT**

## **Tìm hiều ngôn ngữ lập trình C#**

### Cơ bản về ngôn ngữ lập trình C#

C# là ngôn ngữ hiện đại

C# là một ngôn ngữ lập trình thuần hướng đối tượng

C# là một ngôn ngữ ít từ khóa

C# kế thừa một số ưu điểm của C++ và Java

### Kiểu dữ liệu

Dữ liệu được lưu trữ trong bộ nhớ có nhiều kiểu khác nhau trong C# như integer, float và string. Phần này sẽ đề cập về các kiểu dữ liệu đặc biệt được sử dụng trong chương trình:

**Integers (Int, Kiểu Số Nguyên):** Là các số không có phần thập phân, chẳng hạn như: 8,2, 5,9….

**String (Kiểu Chuỗi):** Là đến dạng dữ liệu văn bản. Ví dụ điển hình là như ‘Hello World’ như chương trình đầu tiên.

**List (Danh Sách):** Khi đề cập đến một tập dữ liệu có liên hệ với nhau. Thay vì lưu trữ các dữ liệu này dưới dạng các biến riêng biệt, thì ta lưu trữ các dữ liệu này dưới dạng danh sách.

### Câu lệnh If và Else,

**Lệnh If – Else:** Trong C# lệnh If đánh giá lệnh nếu điều kiện là đúng, nếu điều kiện là sai sẽ có câu lệnh khác. Lệnh này thông báo biết phải làm gì khi điều kiện If là sai.

**Cú pháp:**

if(điều kiện)

{// lệnh nếu điều kiện là đúng}

else

{// lệnh nếu điều kiện là sai}

### Tính đóng gói trong C#

Tính đóng gói (Encapsulation) trong C# được định nghĩa là "tiến trình đóng gói một hoặc nhiều mục bên trong một gói logic hoặc vật lý". Tính đóng gói, trong phương pháp lập trình hướng đối tượng, ngăn cản việc truy cập tới chi tiết của trình trình triển khai

### Phương thức (Hàm)

Một phương thức là một nhóm lệnh cùng nhau thực hiện một tác vụ. Mỗi chương trình C# có ít nhất một lớp với một phương thức là Main.

Để sử dụng một phương thức trong C#, bạn cần:

Định nghĩa phương thức

Gọi phương thức

**Định nghĩa phương thức**: Trong C# khi bạn định nghĩa một phương thức, về cơ bản, bạn khai báo các phần tử của cấu trúc của nó. Cú pháp để định nghĩa một phương thức trong C# là như sau:

<Access Specifier> <Kiểu\_trả\_về> <tên\_phương\_thức>(danh\_sách\_tham\_số)

{//phần thân phương thức}

### Lớp (Class)

Class trong C# là một loại kiểu dữ liệu đặc biệt chứa định nghĩa những thuộc tính (thông tin) và phương thức (hành vi), dùng để mô tả chung cho một nhóm những thực thể cùng loại.

Trong C#, mỗi class có thể chứa:

Biến thành viên (field): Lưu trữ các thông tin mô tả về đối tượng hay trạng thái của đối tượng;

Thuộc tính (property): có vai trò lưu trữ thông tin tương tự như biến thành viên nhưng có khả năng kiểm soát dữ liệu xuất nhập;

Phương thức (method): Dùng để cập nhật, tính toán, cung cấp và xử lý thông tin;

Sự kiện (delegate/event): Gửi thông báo về sự thay đổi trạng thái của đối tượng ra bên ngoài.

Ngoài ra, trong class còn có thể chứa định nghĩa của kiểu dữ liệu khác, gọi là kiểu thành viên. Class có thể chứa định nghĩa của bất kỳ nhóm kiểu nào mà bạn đã biết.

### Kế thừa.

Tính kế thừa (Inheritance) trong C# là một trong những khái niệm quan trọng nhất trong lập trình hướng đối tượng C#.

Tính kế thừa cho phép chúng ta định nghĩa một lớp trong điều kiện một lớp khác, mà làm cho nó dễ dàng hơn để tạo và duy trì một ứng dụng. Điều này cũng cung cấp một cơ hội để tái sử dụng tính năng code, tiện hơn và thời gian thực thi nhanh hơn.

## **Tìm hiểu Winform**

### 2.1 Tổng quan.

Winform là thuật ngữ mô tả một ứng dụng được viết dùng .NET FrameWorrk và có giao diện người dùng Windows Forms.

Mỗi màn hình windows cung cấp một giao diện giúp người dùng giao tiếp với ứng dụng. Giao diện này được gọi là giao diện đồ họa (GUI) của ứng dụng.

Là các ứng dụng windows chạy trên máy tính – mã lệnh thực thi ngay trên máy tính: Microsoft, Word, Excel, Access, Calculator, yahoo, Mail… là các ứng dụng Windows Forms.

### Cơ bản về Winform

Form: Dùng để thiết kế giao diện cho người sử dụng.

Button: Dùng để người dùng tương tác với giao diện khi nhấn sẽ kích hoạt sự kiện thiết lập trong ứng dụng.

Label: Hiển thị văn bản cố định trên form.

Textbox: Dùng để hiển thị dữ liệu hoặc để người dùng nhập dữ liệu vào ứng dụng.

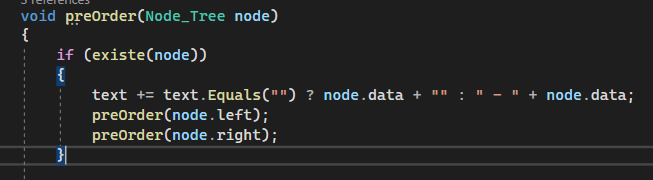
MessageBox: Hiển thị thông báo khi các sự kiện trong ứng dụng được kích hoạt.

Panel: Là thành phần hiển thị một khối nội dung bố trí trên một form. Nó có thể chứa các điều khiển khác như các nút, hộp văn bản, và các điều khiển khác, và có thể được sử dụng để tạo các khu vực cho phép người dùng thao tác với các điều khiển bên trong nó.

## **Giải Thuật Duyệt Cây**

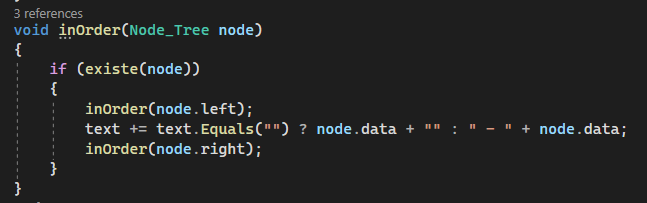
### 3.1 Duyệt Trước

Hàm PreOrder là hàm duyệt trước:

* Đầu tiên thì hàm sẽ xử lý và in ra giá trị nút hiện tại trước.
* Tiếp theo hàm sẽ gọi đệ quy hàm preOrder để duyệt nút cho nhánh bên trái.
* Cuối cùng, hàm sẽ gọi đệ quy hàm preOrder để duyệt nút cho nhánh bên phải.

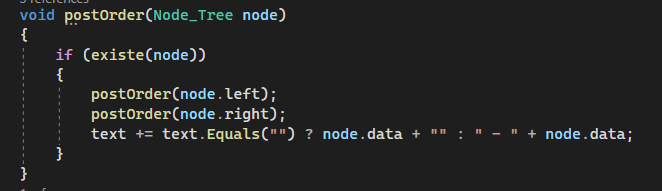
### Duyệt Giữa

Hàm inOrder là hàm duyệt giữa:

* Đầu tiên thì hàm sẽ gọi đệ quy hàm inOrder để duyệt nút cho nhánh bên trái.
* Tiếp theo hàm sẽ xử lý và in ra giá trị của nút hiện tại.
* Cuối cùng, hàm sẽ gọi đệ quy hàm inOrder để duyệt nút cho nhánh bên phải.

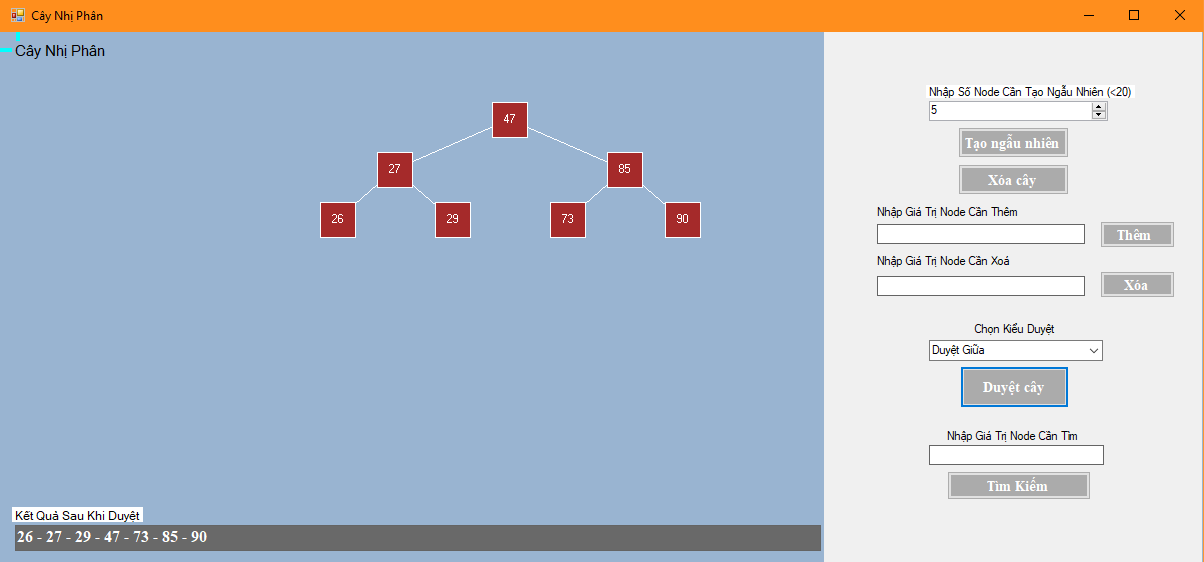
### Duyệt Sau

Hàm postOder là hàm duyệt sau:

* Đầu tiên thì hàm sẽ gọi đệ quy hàm postOrder để duyệt nút cho nhánh bên trái.
* Tiếp theo hàm sẽ gọi đệ quy hàm postOrder để duyệt nút cho nhánh bên phải.
* Cuối cùng, hàm sẽ xử lý và in giá trị của nút hiện tại.

**CHƯƠNG III: CÀI ĐẶT CHƯƠNG TRÌNH**

## **Giao Diện**



## **Các Lớp và một số hàm bên trong**

### 2.1 Form1.cs

#### **2.1.1. Chức Năng**

Quản lý giao diện đồ họa (GUI) trên Windows Form. Chương trình này dùng để tạo ra một cây nhị phân và có thể thực hiện các thao tác như thêm, xóa, tìm kiếm, in kết quả duyệt trước, giữa, sau và vẽ cây ra màn hình. Ngoài ra, chương trình còn có thể kiểm tra lỗi khi người dùng nhập dữ liệu và hiển thị thông báo lỗi tương ứng.

#### **2.1.2. Mã Lệnh**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CayNhiPhan\_NQK

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.StartPosition = FormStartPosition.CenterScreen;

}

int Dato = 0;

int cont = 0;

int n = 0;

B\_Tree bTree = new B\_Tree(null);

Node\_Tree tree\_Node = new Node\_Tree();

Graphics g;

string text = "";

Boolean existe(Node\_Tree Node)

{

try { int x = Node.data; return true; }

catch { return false; }

}

void preOrder(Node\_Tree node)

{

if (existe(node))

{

text += text.Equals("") ? node.data + "" : " - " + node.data;

preOrder(node.left);

preOrder(node.right);

}

}

void inOrder(Node\_Tree node)

{

if (existe(node))

{

inOrder(node.left);

text += text.Equals("") ? node.data + "" : " - " + node.data;

inOrder(node.right);

}

}

void postOrder(Node\_Tree node)

{

if (existe(node))

{

postOrder(node.left);

postOrder(node.right);

text += text.Equals("") ? node.data + "" : " - " + node.data;

}

}

private void Form1\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

}

private void btInsert\_Click(object sender, EventArgs e)

{

txtPrintf.Text = "";

if (txtInsert.Text == "")

{

MessageBox.Show("Vui lòng nhập node cần thêm");

}

else

{

Dato = int.Parse(txtInsert.Text);

vs[n++] = Dato;

if (Dato <= 0 || Dato >= 100)

{

MessageBox.Show("Dữ liệu phải nằm trong khoảng từ 0->99", "Thông báo lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

bTree.Insert(Dato);

txtInsert.Clear();

txtInsert.Focus();

cont++;

Refresh();

Refresh();

}

}

}

int[] vs = new int[30];

private const int Radio = 35;

private void Form1\_Paint(object sender, PaintEventArgs e)

{

this.CenterToScreen();

e.Graphics.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAliasGridFit;

g = e.Graphics;

Pen pen = new Pen(Color.White, 1);

bTree.VeCay(g, this.Font, Brushes.Brown, Brushes.White, pen, Brushes.Aqua);

Rectangle rect = new Rectangle((int)(bTree.tda - Radio / 2), (int)(bTree.tdb - Radio / 2), Radio, Radio);

Pen pencil = new Pen(Brushes.Aqua, 4);

g.DrawRectangle(pencil, rect);

}

int TimPhanTu(int x)

{

for (int i = 0; i < n; i++)

if (vs[i] == x)

return i;

return -1;

}

void XoaPhanTu(int x)

{

int vt = TimPhanTu(x);

for (int i = vt; i <= n - 2; i++)

vs[i] = vs[i + 1];

n--;

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void btdelete\_Click(object sender, EventArgs e)

{

txtPrintf.Text = "";

if (txtDelete.Text == "")

{

MessageBox.Show("Vui lòng nhập node cần xóa");

}

else

{

Dato = int.Parse(txtDelete.Text);

XoaPhanTu(Dato);

bTree.Delete(Dato);

txtDelete.Clear();

txtDelete.Focus();

bTree.tda = 0;

bTree.tdb = 0;

cont++;

Refresh();

Refresh();

}

}

private void btFind\_Click(object sender, EventArgs e)

{

if (txtFind.Text == "")

{

MessageBox.Show("Vui lòng nhập vào node cần tìm");

}

else

{

Dato = int.Parse(txtFind.Text);

bTree.Find(Dato);

Refresh();

}

}

private void btPrintf\_Click(object sender, EventArgs e)

{

if (this.comboBox2.SelectedItem == "Duyệt Trước")

{

preOrder(bTree.root);

txtPrintf.Text = text.ToString();

text = "";

}

if (this.comboBox2.SelectedItem == "Duyệt Giữa")

{

inOrder(bTree.root);

txtPrintf.Text = text.ToString();

text = "";

}

else

if (this.comboBox2.SelectedItem == "Duyệt Sau")

{

postOrder(bTree.root);

txtPrintf.Text = text.ToString();

text = "";

}

}

private void btClear\_Click(object sender, EventArgs e)

{

for (int i = n - 1; i >= 0; i--)

{

bTree.Delete(vs[i]);

bTree.tda = 0;

bTree.tdb = 0;

Refresh();

Refresh();

}

n = 0;

cont = 0;

txtPrintf.Text = string.Empty;

}

private void btRandom\_Click(object sender, EventArgs e)

{

for (int i = n - 1; i >= 0; i--)

{

bTree.Delete(vs[i]);

Refresh();

Refresh();

}

n = 0;

cont = 0;

txtPrintf.Text = string.Empty;

int k = Convert.ToInt32(numericUpDown1.Value);

int number;

bool kt;

Random rand = new Random();

do

{

kt = true;

number = rand.Next(0, 99);

for (int i = 0; i < n; i++)

{

if (vs[i] == number)

{

kt = false;

}

}

if (kt == true)

{

vs[n++] = number;

}

} while (n < k);

for (int i = 0; i < n; i++)

{

bTree.Insert(vs[i]);

Thread.Sleep(500);

Refresh();

Refresh();

}

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

}

}

}

### 2.2 Node\_Tree.cs

#### **2.2.1 Chức Năng**

Dùng để lưu trữ dữ liệu của nút, các liên kết với các nút con và các liên kết với nút cha của nó, và một số thuộc tính khác để hỗ trợ việc vẽ cây và thao tác.

#### **2.2.2 Mã Lệnh**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.TaskbarClock;

namespace CayNhiPhan\_NQK

{

class Node\_Tree

{

public int data;

public Node\_Tree left;

public Node\_Tree right;

public Node\_Tree goc;

public int chieuCao;

public int bac;

public int timX;

public int timY;

public bool truyCapL = false;

public bool truyCapR = false;

public bool daXet = false;

public B\_Tree tree

{

get { return tree; }

set { tree = value; }

}

public Node\_Tree()

{

}

public Node\_Tree(int new\_data, Node\_Tree new\_left, Node\_Tree new\_right, Node\_Tree new\_goc)

{

data = new\_data;

left = new\_left;

right = new\_right;

goc = new\_goc;

chieuCao = 0;

}

public Node\_Tree Insert(int x, Node\_Tree t, int cap, Node\_Tree new\_goc = null)

{

if (t==null)

{

t=new Node\_Tree(x,null,null,new\_goc);

t.bac = cap;

}

else if (x < t.data)

{

cap++;

t.left = Insert(x, t.left, cap, t);

}

else if (x > t.data)

{

cap++;

t.right = Insert(x, t.right, cap, t);

}

else

{

MessageBox.Show("Node đã tồn tại", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return t;

}

public static int Cao(Node\_Tree t)

{

return t==null ? -1 : t.bac;

}

public void Delete(int x, ref Node\_Tree t)

{

if (t!=null)

{

if (x < t.data)

{

Delete(x, ref t.left);

}

else

{

if (x > t.data)

{

Delete(x, ref t.right);

}

else

{

Node\_Tree Delete\_Node = t;

if (Delete\_Node.right == null)

{

t = Delete\_Node.left;

}

else

{

if (Delete\_Node.left == null)

{

t = Delete\_Node.right;

}

else

{

if (Cao(t.left) - Cao(t.right)>0)

{

Node\_Tree PNode = null;

Node\_Tree P = t.left;

bool co = false;

while (P.right != null)

{

PNode = P;

P = P.right;

co = true;

}

t.data=P.data;

Delete\_Node = P;

if (co == true)

{

PNode.right = P.right;

}

else

{

t.left = P.left;

}

}

else

{

if (Cao(t.right)-Cao(t.left)>0)

{

Node\_Tree PNode = null;

Node\_Tree P = t.right;

bool co = false;

while (P.left != null)

{

PNode = P;

P = P.left;

co = true;

}

t.data = P.data;

Delete\_Node = P;

if (co == true)

{

PNode.left = P.right;

}

else

{

t.right = P.right;

}

}

else

{

if (Cao(t.right) - Cao(t.left) == 0)

{

Node\_Tree PNode = null;

Node\_Tree P = t.left;

bool co = false;

while (P.right != null)

{

PNode = P;

P = P.left;

co = false;

}

t.data = P.data;

Delete\_Node = P;

if (co == true)

{

PNode.right = P.left;

}

else

{

t.left = P.left;

}

}

}

}

}

}

}

}

}

else

{

MessageBox.Show("Node không tồn tại", "Thông báo lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void Find(int x, Node\_Tree t)

{

if (t != null)

{

if (x < t.data)

{

Find(x, t.left);

}

else if (x > t.data)

{

Find(x, t.right);

}

else

{

if (x == t.data)

{

timX = t.TDX;

timY = t.TDY;

}

}

}

else

{

MessageBox.Show("Node không tồn tại", "Thông báo lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

#region "ViTri"

private const int Radio = 35;

private const int DistanciaH = 80;

private const int DistanciaV = 15;

private int TDX;

private int TDY;

public void ViTriNode(ref int xmin, int ymin)

{

int aux1, aux2;

TDY = (int)(ymin + Radio / 2);

if (left != null)

{

left.ViTriNode(ref xmin, ymin + Radio + DistanciaV);

}

if ((left != null) && (right != null))

{

xmin += DistanciaH;

}

if (right != null)

{

right.ViTriNode(ref xmin, ymin + Radio + DistanciaV);

}

if (left != null && right != null)

{

TDX = (int)((left.TDX + right.TDX) / 2);

}

else if (left != null)

{

aux1 = left.TDX;

left.TDX = TDX - 80;

TDX = aux1;

}

else if (right != null)

{

aux2 = right.TDX;

right.TDX = TDX + 80;

TDX = aux2;

}

else

{

TDX = (int)(xmin + Radio / 2);

xmin += Radio;

}

}

public void VeNhanhNode(Graphics grap, Pen pen)

{

if (left != null)

{

grap.DrawLine(pen, TDX, TDY, left.TDX, left.TDY);

left.VeNhanhNode(grap, pen);

}

if (right != null)

{

grap.DrawLine(pen, TDX, TDY, right.TDX, right.TDY);

right.VeNhanhNode(grap, pen);

}

}

public void VeNode(Graphics grap, Font source, Brush Filling, Brush FillingSoure, Pen pen, Brush encuentro)

{

Rectangle rect = new Rectangle((int)(TDX - Radio / 2), (int)(TDY - Radio / 2), Radio, Radio);

grap.FillRectangle(encuentro, rect);

grap.FillRectangle(Filling, rect);

grap.DrawRectangle(pen, rect);

StringFormat formato = new StringFormat();

formato.Alignment = StringAlignment.Center;

formato.LineAlignment = StringAlignment.Center;

grap.DrawString(data.ToString(), source, FillingSoure, TDX, TDY, formato);

if (left != null)

{

left.VeNode(grap, source, Filling, FillingSoure, pen, encuentro);

}

if (right != null)

{

right.VeNode(grap, source, Filling, FillingSoure, pen, encuentro);

}

}

public void color(Graphics grap, Font source, Brush Filling, Brush FillingSoure, Pen pen)

{

Rectangle rect = new Rectangle((int)(TDX - Radio / 2), (int)(TDY - Radio / 2), Radio, Radio);

grap.FillRectangle(Filling, rect);

grap.DrawRectangle(pen, rect);

StringFormat Format = new StringFormat();

Format.Alignment = StringAlignment.Center;

grap.DrawString(data.ToString(), source, FillingSoure, TDX, TDY, Format);

}

#endregion

}

}

### 2.3 B\_Tree.cs

#### **2.3.1 Chức Năng**

Sử dụng để tạo ra một cây nhị phân và thực hiện các thao tác trên cây như: thêm, xóa, tìm, vẽ cây, và duyệt cây.

#### **2.3.2 Mã Lệnh**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CayNhiPhan\_NQK

{

class B\_Tree

{

public Node\_Tree root;

public Node\_Tree aux;

public B\_Tree()

{

aux = new Node\_Tree();

}

public B\_Tree (Node\_Tree new\_root)

{

root = new\_root;

}

public void Insert(int x)

{

if (root == null)

{

root=new Node\_Tree(x, null, null, null);

root.bac = 0;

}

else

{

root = root.Insert(x, root, root.bac);

}

}

public void Delete(int x)

{

if (root == null)

{

root = new Node\_Tree(x, null, null, null);

root.bac = 0;

}

else

{

root.Delete(x, ref root);

}

}

public int tda;

public int tdb;

public void Find(int x)

{

root.Find(x, root);

tda = root.timX;

tdb = root.timY;

}

public List<int> re\_edit = new List<int>();

public bool end = false;

public void inOrder(Node\_Tree State\_Node)

{

if (State\_Node == null || end)

{

State\_Node = null;

end = true;

return;

}

else

{

if (State\_Node.truyCapR && State\_Node.truyCapL && State\_Node.daXet && !end)

{

State\_Node.truyCapL = false;

State\_Node.truyCapR = false;

State\_Node.daXet = false;

inOrder(State\_Node.goc);

}

if (State\_Node.left != null && State\_Node.truyCapL == false && !end)

{

State\_Node.truyCapL = true;

inOrder(State\_Node.left);

}

if ((State\_Node.left == null || State\_Node.truyCapL) && !end)

{

if (State\_Node.left == null && !end)

{

State\_Node.truyCapL = true;

}

re\_edit.Add(State\_Node.data);

State\_Node.daXet = true;

if (State\_Node.right != null && !State\_Node.truyCapL && !end)

{

State\_Node.truyCapR = true;

inOrder(State\_Node.right);

}

if (State\_Node.goc != null && !end)

{

if (State\_Node.right == null)

{

State\_Node.truyCapR = true;

}

inOrder(State\_Node.goc);

}

else

{

inOrder(State\_Node.goc);

}

}

}

}

public void VeCay(Graphics grap, Font source, Brush Filling, Brush Fillingsource, Pen pen, Brush encuentro)

{

int x = 320;

int y = 70;

if (root == null)

{

return;

}

root.ViTriNode(ref x, y);

root.VeNhanhNode(grap, pen);

root.VeNode(grap, source, Filling, Fillingsource, pen, encuentro);

}

public int x1 = 320;

public int y2 = 70;

public void color(Graphics grap, Font source, Brush Filling, Brush Fillingsource, Pen pen, Node\_Tree root, bool post, bool inor, bool preor)

{

Brush entorno = Brushes.Gray;

if (inor == true)

{

if (root != null)

{

color(grap, source, Filling, Fillingsource, pen, root.left, post, inor, preor);

root.color(grap, source, entorno, Fillingsource, pen);

Thread.Sleep(1000);

root.color(grap, source, Filling, Fillingsource, pen);

color(grap, source, Filling, Fillingsource, pen, root.right, post, inor, preor);

}

}

else if (preor)

{

root.color(grap, source, entorno, Fillingsource, pen);

Thread.Sleep(1000);

root.color(grap, source, Filling, Fillingsource, pen);

color(grap, source, Filling, Fillingsource, pen, root.left, post, inor, preor);

color(grap, source, Filling, Fillingsource, pen, root.right, post, inor, preor);

}

else if (post)

{

if (root != null)

{

color(grap, source, Filling, Fillingsource, pen, root.left, post, inor, preor);

color(grap, source, Filling, Fillingsource, pen, root.right, post, inor, preor);

root.color(grap, source, entorno, Fillingsource, pen);

Thread.Sleep(1000);

root.color(grap, source, Filling, Fillingsource, pen);

}

}

}

}

}

CHƯƠNG 4: KẾT LUẬN

1. Đánh giá chung về đề tài
   1. Kết quả đạt được
   * Tạo được cây nhị phân.
   * Duyệt cây theo 3 cách: Duyệt trước, duyệt giữa và duyệt sau.
   * Có các chức năng thêm, xoá, tìm kiếm, tạo cây ngẫu nhiên và xoá cây.
   * Thông báo khi người dùng nhập sai hoặc thao tác sai.
   1. Hạn chế
   * Mô phỏng khi duyệt cây còn hạn chế.
   * Giao diện chưa tối ưu và bắt mắt
2. Hướng phát triển của đề tài
   * Thêm các chuyển động để mô phỏng rõ hơn khi duyệt cây.
   * Cải thiện giao diện

**TÀI LIỆU THAM KHẢO**

[1] Microsoft Learn : [C# Documentation](https://learn.microsoft.com/en-us/dotnet/csharp/)

[2] Microsoft Learn : [Winform](https://learn.microsoft.com/vi-vn/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8)