

báo cáo thực tập cơ sở

Giảng viên hướng dẫn:Nguyễn Thủy Đoan Trang



Họ và tên: Võ Văn Thắng

MSSV: 59132246

Lớp: 59CNTT3

# ĐỀ TÀI 3: CÀI ĐẶT CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

**NHẬN XÉT KẾT QUẢ THỰC TẬP**

Họ và tên sinh viên: .

Mã số sinh viên: …………………… Lớp ……………… Hệ:

Khoa Công nghệ Thông tin, Trường Đại học Nha Trang.

Nội dung thực tập:

Đánh giá kết quả thực tập:

*………………… ngày …. tháng … năm ……* Giáo viên hướng dẫn

GIỚI THIÊU

Ngày nay, khi ngành công nghệ thông tin ngày càng phát triển, khoa học máy tính không ngừng vươn tới những tìm tòi mới mẻ hơn, mọi người chủ yếu làm việc dựa trên máy móc và thiết bị điện tử thì các phần mềm ứng dụng lại càng trở nên quan trọng và hữu ích hơn bao giờ hết.

Tất cả các thông tin muốn biết, muốn tìm hiểu bạn đều có thể tìm được trên mạng Internet thông qua các công cụ tìm kiếm. Các công cụ tìm kiếm đó được xây dựng từ các phần mềm tìm kiếm khác nhau., việc đưa ra một phương pháp nhằm giải quyết vấn đề tìm kiếm dữ liệu có hiệu quả và nhanh chóng nhất luôn được sự quan tâm của các nhà phát triển phần mềm.

Thông thường dữ liệu được biểu diễn dưới dạng danh sách liên kết. Việc truy suất dữ liệu chưa đạt hiệu quả cao. Sử dụng cấu trúc dữ liệu cây là một giải pháp làm tăng hiệu suất trong các thao tác xử lý. Vấn đề đặt ra : với việc sử dụng cấu trúc dạng cây, chúng ta cần dùng giải thuật nào với từng dạng dữ liệu để đạt hiệu quả cao nhất. Để giải quyết vấn đề trên ta cùng tìm hiểu một số phương pháp duyệt cây

**MỤC LỤC**

[ĐỀ TÀI 3: CÀI ĐẶT CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN 1](#_Toc60781721)

[GIỚI THIÊU 3](#_Toc60781722)

[**1)cơ sở lý thuyết** 4](#_Toc60781723)

[1.1CÂY 5](#_Toc60781724)

[1.2)một số khái niệm 5](#_Toc60781725)

[1.2.1)các nút 5](#_Toc60781726)

[1.2.2)nút gốc 6](#_Toc60781727)

[1.2.3)các nút lá 6](#_Toc60781728)

[1.2.4)các nút trong 6](#_Toc60781729)

[1.2.5)cây con 6](#_Toc60781730)

[1.2.6)cây trong lý thuyết đồ thị 6](#_Toc60781731)

[1.2.7)cây sắp thứ tự 6](#_Toc60781732)

[1.2.8)cây tổng quát và cây nhị phân 6](#_Toc60781733)

[1.3.biểu diễn cây 7](#_Toc60781734)

[1.3.1)biểu diễn bằng các nút và các con trỏ 7](#_Toc60781735)

[1.3.2)biểu diễn cây nhị phân bằng mảng 7](#_Toc60781736)

[1.3.3)Duyệt trung thứ tự trong cây nhị phân 8](#_Toc60781737)

[1.3.4Duyệt tiền thứ tự trong cây nhị phân 9](#_Toc60781738)

[1.3.5Duyệt hậu thứ tự trong cây nhị phân 10](#_Toc60781739)

[**2)cài đặt các thao tác thêm ,xóa nút,tìm kiếm trên cây nhị phân** 11](#_Toc60781740)

[2.1)định nghĩa dữ liệu cho node 11](#_Toc60781741)

[2.2 khởi tạo cây 11](#_Toc60781742)

[2.3Sau đó code hàm thêm phần tử vào cây 11](#_Toc60781743)

[2.4Tạo cây 12](#_Toc60781744)

[2.5Duyệt cây (theo NLR) 12](#_Toc60781745)

[2.6Tìm node 13](#_Toc60781746)

[2.7 xóa node 14](#_Toc60781747)

[2.8Thêm node 14](#_Toc60781748)

[2.9)Chạy chương trình 15](#_Toc60781749)

[**3)Code vẽ cây nhị phân trên dev C** 18](#_Toc60781750)

[KẾT LUẬN 20](#_Toc60781751)

[TÀI LIÊU THAM KHẢO 20](#_Toc60781752)

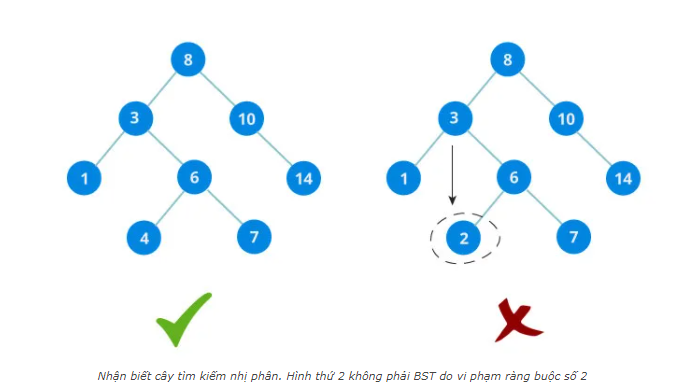
**1)cơ sở lý thuyết**

1.1CÂY

Cây (Trees) là một tập hợp hữu hạn các phần tử gọi là nút cây (Node), trong đó có một nút đặc biệt gọi là nút gốc (Root). Trên tập hợp các nút này có một quan hệ phân cấp gọi là quan hệ "cha - con".

Cây nhị phân tìm kiếm (binary search tree – BST) là cây nhị phân trong đó tại mỗi nút, khóa của nút đang xét lớn hơn nút khóa của tất cả các nút thuộc cây con trái và nhỏ hơn tất cả nút khóa thuộc cây con phải.

Ví dụ:



1.2)một số khái niệm

1.2.1)các nút

Một nút có thể chứa một giá trị, một điều kiện, một cấu trúc dữ liệu riêng biệt hoặc chính một cây. Mỗi nút trong một cây có thể không có hoặc có một số nút con, các nút con có mức cao hơn nó (theo quy ước khác với cây tự nhiên, cây trong cấu trúc dữ liệu phát triển từ trên xuống). Một nút có con được gọi là nút cha của các nút con. Một nút có nhiều nhất một nút cha.

1.2.2)nút gốc

Trong mỗi cây có một nút đặc biệt được gọi là nút gốc (hay nói đơn giản là g*ốc).* Nút gốc là nút duy nhất không có nút cha. Nút gốc là nơi khởi đầu của nhiều giải thuật trên cây. Tất cả các nút khác được nối về nút gốc bằng một đường đi qua các cạnh hay các liên kết.

1.2.3)các nút lá

Các nút không có nút con được gọi là nút lá hay đơn giản là lá

1.2.4)các nút trong

Nút trong của một cây là nút trên cây có ít nhất một con, nghĩa là các nút không phải là lá. Các khái niệm về mức của mỗi nút, chiều cao của cây được định nghĩa giống như cây trong lý thuyết đồ thị.

1.2.5)cây con

Một cây con là một bộ phận của cấu trúc dữ liệu cây mà tự nó cũng là một cây. Một nút bất kỳ trong câ*y T,* cùng với các nút dưới nó tạo thành một cây con của *T.*

1.2.6)cây trong lý thuyết đồ thị

Trong lý thuyết đồ thị, một cây là một đồ thị liên thông và không có chu trình. Cây như vậy còn được gọi là cây tự do. Một cây có gốc là một cây tự do, trong đó có một đỉnh được chọn làm gốc và các cạnh được định hướng là hướng của các đường đi đơn ra khỏi gốc tới các đỉnh khác. Trong trường hợp này, hai đỉnh bất kỳ dược nối với nhau bao hàm chúng có qua hệ cha-con. Một đồ thị không chu trình với nhiều thành phần liên thông được gọi là một rừng.

1.2.7)cây sắp thứ tự

Có hai dạng cấu trúc cơ sở của cây là không không thứ tự và cây có thứ tự. Một cây không thứ tự là cây có cấu trúc cây, trong đó giữa các con của một nút, không có thứ tự nào. Một cây, trong đó các con của một nút tuân theo một thứ tự xác định được gọi là cây có thứ tự. Các cây có thứ tự có nhiều ứng dụng sâu sắc trong cấu trúc của cây. Cây tìm kiếm nhị phân là một cây sắp thứ tự điển hình.

1.2.8)cây tổng quát và cây nhị phân

Các cây trong đó mỗi nút có thể có nhiều hơn hai con được gọi là cây tổng quát, các cây trong đó mỗi nút có không quá hai con được gọi là cây nhị phân.

1.3.biểu diễn cây

Có nhiều phương pháp biểu diễn cây. Cách thường dùng nhất là biểu diễn mỗi nút như một dữ liệu kiểu bản ghi, mỗi nút chứa các con trỏ tới các con hoặc cha của nó, hoặc cả hai. Cây cũng có thể biểu diễn bằng các mảng cùng với quan hệ giữa các vị trí trong mảng

1.3.1)biểu diễn bằng các nút và các con trỏ

Mỗi nút là một dữ liệu kiểu bản ghi với ba trường. Một trường thường gọi là INFOR, chứa thông tin lưu trữ tại nút đó. Thông tin này có thể chỉ là một số, một ký tự, cũng có thể là một tập hợp dữ liệu rất phức tạp. Hai trường LLINK và RLINK chứa các liên kết trái và phải. Nếu cây là cây nhị phân, LLINK trỏ tới con trai của nút, RLINK trỏ tới con phải của nút. Nếu cây là cây tổng quát, LLINK trỏ tới con cực trái và RLINK trỏ tới em kế cận phải của nút đó. Do đó danh sách các nút biểu diễn một cây tổng quát, khi được xem là biểu diễn của cây nhị phân sẽ cho một cây nhị phân. Cây nhị phân này được gọi là cây nhị phân tương đương với cây tổng quát ban đầu.

1.3.2)biểu diễn cây nhị phân bằng mảng

1- Cây nhị phân đầy đủ là cây nhị phân, trong đó mỗi nút trong chỉ có hai con. Cây nhị phân hoàn chỉnh là cây nhị phân đầy đủ, trong đó tất cả các lá đều ở mức cao nhất. Một cây nhị phân hoàn chỉnh chiều cao h chỉ có *2*\*- 1 nút.

2- Do đó người ta có thể dùng một mảng gồm *2*^ +1 – 1 phần tử để biểu diễn cây hoàn chỉnh, bằng cách lần lượt lưu trữ thông tin của mỗi nút vào mảng theo thứ tự từ trên xuống dưới, từ trái sang phải. Khi đó con trai của nút thứ *i* là phần tử thứ *2\*i*, con phải là phần tử thứ 2*\*i+1.* Cha của phần tử thứ i là phần tử thứ int(*1/2*)*.*

3- Nếu cây là không hoàn chỉnh, ta gán giá trị Null cho các vị trí còn thiếu so với cây nhị phân hoàn chỉnh.

4- Một cách khác, dùng một mảng hai chiều trong dòng thứ nhất ghi các thông tin của nút, dòng thứ hai ghi chỉ số của nút cha của nút đó với dấu + nếu nút hiện tại là con trái, với dấu - nếu nút hiện tại là con phải của nút cha.

3)các phương pháp 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 link), 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 mà chúng ta có thể sử dụng để duyệt một cây:

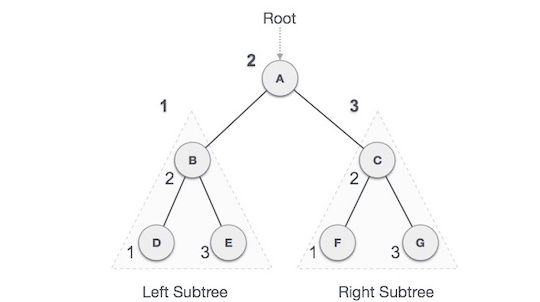
* Duyệt tiền thứ tự (Pre-order Traversal)
* Duyệt trung thứ tự (In-order Traversal)
* Duyệt hậu thứ tự (Post-order Traversal)

Nói chung, chúng ta duyệt một cây để tìm kiếm hay là để xác định vị trí phần tử hoặc khóa đã cho trong cây hoặc là để in tất cả giá trị mà cây đó chứa.

1.3.3)Duyệt trung thứ tự trong cây nhị phân

Trong cách duyệt này, cây con bên trái được truy cập đầu tiên, sau đó là nút gốc và sau đó là cây con bên phải. Bạn nên luôn luôn ghi nhớ rằng mỗi nút đều có thể biểu diễn một cây con.

Nếu một cây nhị phân được duyệt trung thứ tự, kết quả tạo ra sẽ là các giá trị khóa được sắp xếp theo thứ tự tăng dần.



Ở hình ví dụ minh họa trên, **A** là nút gốc. Với phương thức duyệt trung thứ tự, chúng ta bắt đầu từ nút gốc **A**, di chuyển tới cây con bên trái **B** của nút gốc. Tại đây, **B** cũng được duyệt theo cách thức duyệt trung thứ tự. Và tiến trình tiếp tục cho đến khi tất cả các nút đã được truy cập. Kết quả của cách thức duyệt trung thứ tự cho cây trên sẽ là:

**D -> B -> E -> A -> F -> C-> G**

Duyệt cho tới khi tất cả các nút đều được duyệt:

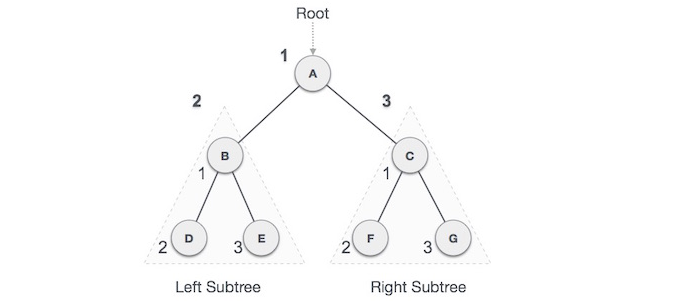
**Bước 1**: Duyệt các cây con bên trái một cách đệ qui

**Bước 2**: Truy cập nút gốc

**Bước 3**: Duyệt các cây con bên phải một cách đệ qui

1.3.4Duyệt tiền thứ tự trong cây nhị phân

Trong cách thức duyệt tiền thứ tự trong cây nhị phân, nút gốc được duyệt đầu tiên, sau đó sẽ duyệt cây con bên trái và cuối cùng sẽ duyệt cây con bên phải.



Ở hình ví dụ minh họa trên, **A** là nút gốc. Chúng ta bắt đầu từ **A**, và theo cách thức duyệt tiền thứ tự, đầu tiên chúng ta truy cập chính nút gốc **A** này và sau đó di chuyển tới nút con bên trái **B** của nó. **B** cũng được duyệt theo cách thức duyệt tiền thứ tự. Và tiến trình tiếp tục cho tới khi tất cả các nút đều đã được truy cập. Kết quả của cách thức duyệt tiền thứ tự cây này sẽ là:

**A -> B -> D -> E -> C -> F -> G**

Duyệt cho tới khi tất cả các nút đều được duyệt:

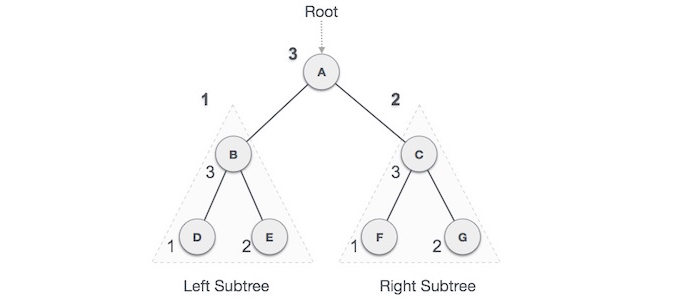
**Bước 1**: Truy cập nút gốc

**Bước 2**: Duyệt các cây con bên trái một cách đệ qui

**Bước 3**: Duyệt các cây con bên phải một cách đệ qui

1.3.5Duyệt hậu thứ tự trong cây nhị phân

Trong cách thức duyệt hậu thứ tự trong cây nhị phân, nút gốc của cây sẽ được truy cập cuối cùng, do đó bạn cần chú ý. Đầu tiên, chúng ta duyệt cây con bên trái, sau đó sẽ duyệt cây con bên phải và cuối cùng là duyệt nút gốc.



Ở hình ví dụ minh họa trên, **A** là nút gốc. Chúng ta bắt đầu từ **A**, và theo cách duyệt hậu thứ tự, đầu tiên chúng ta truy cập cây con bên trái **B**. **B** cũng được duyệt theo cách thứ duyệt hậu thứ tự. Và tiến trình sẽ tiếp tục tới khi tất cả các nút đã được truy cập. Kết quả của cách thức duyệt hậu thứ tự của cây con trên sẽ là:

**D -> E -> B -> F -> G -> C -> A**

Duyệt cho tới khi tất cả các nút đều được duyệt:

**Bước 1**: Duyệt các cây con bên trái một cách đệ qui

**Bước 2**: Duyệt các cây con bên phải một cách đệ qui

**Bước 3**: Truy cập nút gốc.

**2)cài đặt các thao tác thêm ,xóa nút,tìm kiếm trên cây nhị phân**

2.1)định nghĩa dữ liệu cho node

Chúng ta sẽ có struct lưu trữ một node như sau – ở đây để đơn giản mình sử dụng kiểu dữ liệu int cho thành phần dữ liệu của node:



2.2 khởi tạo cây

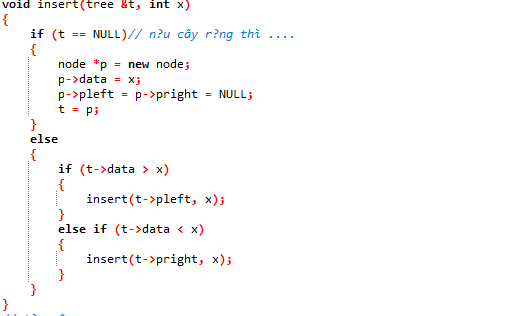
Vì nó là con trỏ nên các bạn gán nó bằng NULL để tránh lỗi, nhưng để mọi thứ rõ ràng hơn, mình sẽ dùng hàm tạo cây đơn giản gán nó bằng NULL.



2.3Sau đó code hàm thêm phần tử vào cây

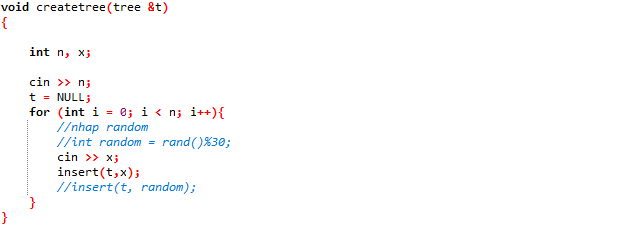
Việc thêm 1 phần tử vào cây nhị phân tìm kiếm vẫn phải đảm bảo được các ràng buộc của một BST đã trình bày ở trên. Như vậy, bạn cần phải tìm kiếm vị trí thích hợp trong BST để lưu giữ nó.

Nếu bạn để ý, bạn sẽ nhận ra vị trí của các Node được thêm vào sẽ luôn Node lá(không có Child nào hết). Như vậy, tại vị trí đó trước khi các Node mới tới ở thì nó là NULL. Ta có quy trình như sau

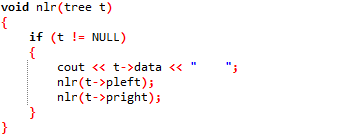


2.4Tạo cây

Khi tạo một nút node mới, chúng ta cần phải gán lại các thành phần của node để nó không nhận giá trị rác, tránh lỗi không mong muốn. Chúng ta sẽ tạo một biến động cho node và trả về địa chỉ của node đó, mình sẽ có đoạn code tạo node như sau:

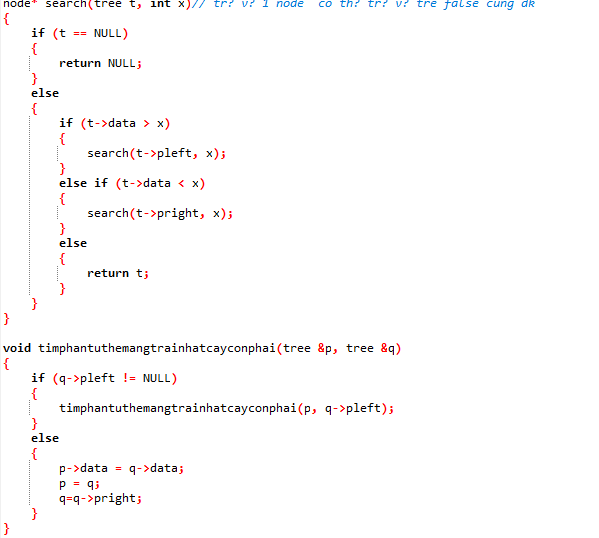
****

2.5Duyệt cây (theo NLR)

****

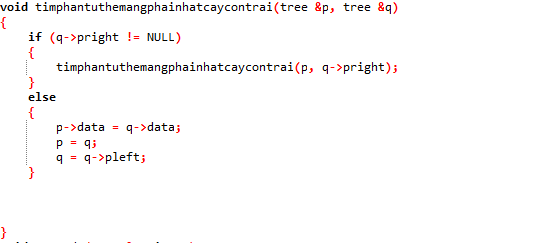
2.6Tìm node

Như đã giới thiệu ở trên, để tìm một phần tử trong cây nhị phân tìm kiếm, chúng ta sẽ thực hiện tương tự việc tìm kiếm nhị phân. Nếu như nút cần tìm nhỏ hơn nút đang xét, chúng ta sẽ tìm cây con bên trái, ngược lại chúng ta sẽ tìm trong cây con bên phải, nếu đúng nút cần tìm thì mình sẽ trả về địa chỉ của nút đó. Mình sẽ có thuật toán sau:

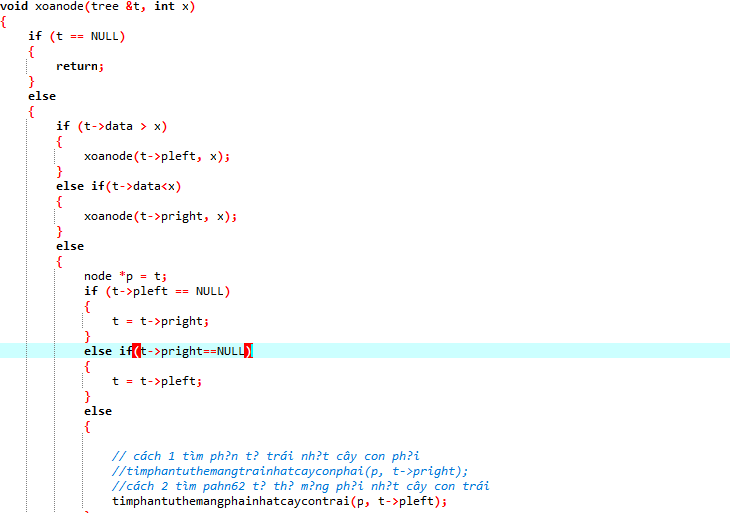
****

2.7 xóa node

Xóa node là lá

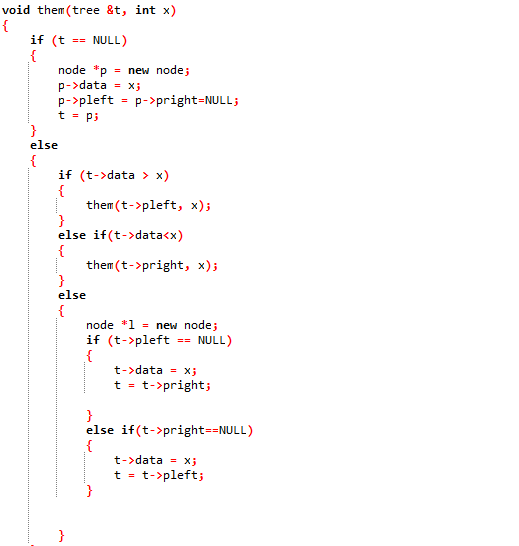
****

Xóa node

****

2.8Thêm node

Để thêm phần tử vào cây nhị phân tìm kiếm, ta phải thêm vào cây nhưng vẫn đảm bảo được cây đó vẫn là cây nhị phân tìm kiếmHàm duyệt tìm vị trí thích hợp và chèn của mình như sau:

****

2.9)Chạy chương trình

**int main()**

**{**

**tree t;**

**createtree(t);**

**//// nhap du lieu tu file**

**// t = NULL;**

**// ifstream filein;**

**// filein.open("INPUT.TXT", ios\_base::in);**

**// int n;**

**// filein >> n;**

**// for(int i=1;i<=n;i++){**

**// int x;**

**// filein >> x;**

**// them(t, x);**

**//// }**

**cout << "\nDuyet theo nlr: \n";**

**nlr(t);**

**int x;**

**cout << "\nNhap node can tim : "; cin >> x;**

**node \*u = search(t, x);**

**if (u == NULL)**

**{**

**cout << "\nKhong tim thay node xin kiem tra lai!";**

**}**

**else**

**{**

**cout << "\nNo de ban can tim co trong cay cam on!";**

**}**

**int g;**

**cout << "\nNhap phan tu ban muon xoa!"; cin >> g;**

**xoanode(t, g);**

**cout << "\nCay sau khi xoa!\n";**

**nlr(t);**

**d: int h;**

**cout << "\nNhap phan tu ban them!"; cin >> h;**

**them(t, h);**

**cout << "\nCay sau khi them!\n";**

**nlr(t);**

**goto d;**

**system("pause");**

**return 0;**

**}**

**3)Code vẽ cây nhị phân trên dev C**

#include <stdio.h>

#include <math.h>

#define PARENT(i) ((i-1) / 2)

#include<stdio.h>

#include<iostream>

using namespace std;

#define LINE\_WIDTH 70

int main() {

int tree[1000];

int n;

cin >> n;

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

cin >> tree[i];

}

int print\_pos[n];

int i, j, k, pos, x=1, level=0;

print\_pos[0] = 0;

for(i=0,j=1; i<n; i++,j++) {

pos = print\_pos[PARENT(i)] + (i%2?-1:1)\*(LINE\_WIDTH/(pow(2,level+1))+1);

for (k=0; k<pos-x; k++) printf("%c",i==0||i%2?' ':'-');

printf("%d",tree[i]);

print\_pos[i] = x = pos+1;

if (j==pow(2,level)) {

printf("\n");

level++;

x = 1;

j = 0;

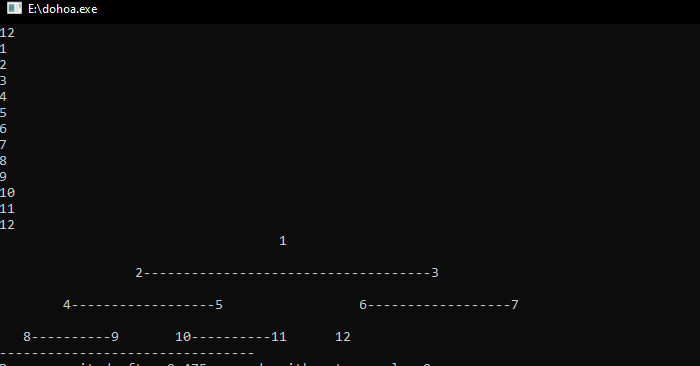
}

}

return 0;

}

**Ví dụ:**

****

KẾT LUẬN

Bằng những kiến thức được dạy trên lớp cùng sự hướng dẫn nhiệt tình của cô, chúng em đã xây dựng chương trình cây tìm kiếm nhị phân bằng ngôn ngữ CH.  Chương trình bao gồm những phương thức cơ bản để xây dựng một cây nhị phân tìm kiếm. Do đó có thể còn nhiều thiếu sót và chưa được mở rộng nhiều về mặt ứng dụng thực tế. Chúng em rất mong các thầy đóng góp ý kiến để chúng em rút kinh nghiệm và bổ sung thêm kiến thức cho những chương trình được xây dựng tiếp sau này.

TÀI LIÊU THAM KHẢO

em xin chân thành cảm ơn.