Họ tên sinh viên: Nguyễn Công Bình.

Mã số sinh viên: 19964.

Lớp: 64IT5.

***Bài thực hành số 2***

**Câu 1:** Giải thích các khái niệm :

* Các cấp của cây
* Đường đi và Độ dài của đường đi của nút trong cây nhị phân
* Chiều cao của cây
* Các loại cây nhị phân
* cho ví dụ để giải thích các khái niệm trên

*Giải*

***Cây nhị phân*** là cây trong đó mỗi nút có nhiều nhất 2 nút con đi ra.

***Ví dụ:***

Cấp 1

Cấp 2

Cấp 3

Cấp 4

Hình 1.

*Các cấp của cây*

Cây được biểu diễn theo dạng phân cấp, có các mức được đánh số từ 1,2…theo hướng xuống dưới, cấp độ của một nút là cơ sở để tính *độ dài của đường đi* từ nút gốc tới nút đó.

*Đường đi của nút* trong cây của nút trong cây là (quãng đường) đường đi từ nút gốc đến nút đó.

*Độ dài của đường đi* của một nút được tính bằng số cạnh đi từ nút gốc tới nút đó. Trong hình vẽ trên, các nút ở cấp 2 có độ dài bằng 1, ở cấp 3 có độ dài bằng 2, ở cấp 4 có độ dài bằng 3.

*Chiều cao của cây*

Chiều cao của cây là *đường đi dài nhất* đi từ nút gốc tới nút lá xa nhất của cây.

Trong cây trên thì chiều cao của cây là 3, là đường đi dài nhất, từ nút gốc (số 1) tới các nút lá số 5, 6.

*Các loại cây nhị phân:*

*Cây nhị phân cân bằng:*

Một cây nhị phân được coi là cân bằng khi :

* Tất cả các lá của cây đều ở trên cùng một cấp hoặc trong cùng một cấp của nhau.
* Chiều cao của cây <= log2N với N là số nút trong cây nhị phân.

*Cây nhị phân hoàn thiện:*

Khái niệm cây hoàn thiện liên quan đến cây cân bằng.

Một cây được coi là hoàn thiện nếu nó cân bằng và tất cả các lá ở cấp dưới cùng đều ở phía bên trái của cây.

Cây nhị phân hoàn thiện là cây mà mỗi nút của nó có đúng 2 con (trừ nút lá), và các nút lá có cùng mức.

*Cây nhị phân đầy đủ:*

Cây nhị phân đầy đủ là cây nhị phân mà mỗi nút trong cây có chính xác 0 hoặc 2 con.

**Câu 2:** Hãy trình bày các cách cài đặt cây nhị phân

*Giải*

Hiện tại có một vài cách cài đặt cây.

Cách cài đặt phổ biến nhất là dùng cấu trúc liên kết. Mỗi nút được coi là một lớp class TreeNode tương tự như lớp Node trong danh sách liên kết. Giống như danh sách liên kết đôi, mỗi nút có con trỏ trỏ tới nút con trái, nút con phải; khi với tư cách là nút con thì nó đang được nút cha trỏ tới.

Một cách cài đặt khác là kiểu *cây đệ quy* có sử dụng các liên kết. Cách này sẽ liên quan đến việc xác định mỗi nút là một cây với các thuộc tính cho các nút con của nó cũng là cây. Vì vậy, mỗi nút, và tất cả các nút con của nó, đại diện cho một cây đối với chính nó

Tuy cây là một cấu trúc phi tuyến, nhưng có thể dùng mảng để lưu trữ dữ liệu các nút của cây.

Khi sử dụng mảng để lưu trữ dữ liệu các nút của cây nhị phân, có 2 cách tiếp cận :

a)Cách 1 : coi cây là cây hoàn thiện, bất kỳ một nút cha nào cũng có 2 nút con, trừ các nút lá. Như vậy, đối với bất kỳ nút nào được lưu trữ ở vị trí thứ i của mảng thì nút con trái được lưu ở vị trí 2\*i, nút con phải ở vị trí 2\*i +1.

Cách tiếp cận này có hiệu quả trong việc quản lý dung lượng bộ nhớ, tuy nhiên đối với cây không hoàn thiện sẽ lãng phí bộ nhớ vì có những nút không tồn tại, ô nhớ bị bỏ trống.

Ví dụ cho cây :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| T[i] | A | B | C | D |  |  | E |

Lập mảng :

b)Cách tiếp cận thứ 2 là cài đặt cây theo cách quản lý của Hệ điều hành, chỉ những nút nào có mặt, xuất hiện trước được lưu trước, liền kề, không bỏ trống ô nhớ, mỗi nút được lưu trữ theo chỉ số mảng, theo thứ tự nút được nhập lúc tạo cây.

Với cây trên thì mảng T[] có dạng :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 |
| T[i] | A | B | C | D | E |

Rõ ràng cách này tiết kiệm ô nhớ hơn nhiều. Tuy nhiên khi thêm bớt nút trong cây, cách này phiền toái thêm việc phải dịch chuyển vị trí các ô nhớ để duy trì sự liền kế.

**Câu 3:** Cho khai báo cấu trúc nút cây nhj phân và khai báo cây nhị phân như sau :

struct TNode

{ int du\_lieu ;

TNode \*Lchild, \*Rchild ;

};

typedef TNode \*Tree ;

Hãy viết hàm *Tạo nút cây nhị phân*, hàm *Tạo cây nhị phân,*  giải thích các câu lệnh

*Giải:*

**Hàm tạo nút:** Cần cho khi nhập dữ liệu hay thực hiện các thao tác trước khi thêm nút, phải tạo một nút để nhận dữ liệu mới.

TNode\* TaoNode(TNode\* node, int du\_lieu) // Chèn 1 Node vào cây

{ node=new TNode; // Khai báo con trỏ node kiểu TNode bằng cấp bộ nhớ động

// Cho \*node theo kiểu cấu trúc TNode

Node ->du\_lieu=du\_lieu; // node trỏ đến du\_lieu được gán bằng du\_lieu

Node ->Lchild=NULL; // Gán node con trái bằng NULL

Node ->Rchild=NULL; //Gán node con phải bằng NULL

return node;

}

***Hàm tạo cây:*** Việc nhập dữ liệu để tạo cây ban đầu phải nhập từ gốc, và dữ liệu gốc phải khác 0, sau đó gọi đệ quy nhập cây con bên trái; gọi đệ quy nhập cây con bên phải.

TNode\* TaoTree(TNode \*node, int du\_lieu) // nhập node tạo cây

{ cout<<"\n Nhap du lieu cho nut : "; cin≫du\_lieu;

if (du\_lieu == 0) return NULL; // Nếu du\_lieu == 0 thì thoát

node = TaoNode(node,du\_lieu); // Khởi tạo giá trị cho node mới

cout<<" Nhap Du lieu node con TRAI cua nut "<<du\_lieu<<" : ";

node->Lchild = TaoTree(node->Lchild,du\_lieu); // Tạo nút con trái

cout<<" Nhap Du lieu node con PHAI cua nut "<<du\_lieu<<" : ";

node->Rchild = TaoTree(node->Rchild,du\_lieu); // Tạo nút con phải

return node;

}

**Câu 4**: Sử dụng khai báo nút, khai báo cây nhị phân ở **Câu 3**), viết hàm tìm nút có dữ liệu bằng dữ liêu nhập từ bàn phím*,* giải thích các câu lệnh.

*Giải*

TNode\* TimNode(Tree GOC, int du\_lieu) // tim nut theo du\_lieu, theo kiểu duyệt InOrder

{ if(GOC!=NULL) // Nếu GOC khác NULL thì thực hiện lệnh bên dưới

{ if (GOC->du\_lieu == du\_lieu) return GOC; // Nếu GOC trỏ đến du\_lieu == du\_lieu thì trả về GOC

TimNode(GOC->Lchild,du\_lieu);// Gọi đệ quy tìm theo nút con trái

TimNode(GOC->Rchild,du\_lieu); // Gọi đệ quy tìm theo nút con phải

}

}

***Nhập dữ liệu từ hàm main:***

int main(){

Tree GOC;

GOC = NULL; //Tạo cây rỗng

TNode \*node=NULL;

int du\_lieu;

cout<<"\n Nhap so 0 de chuyen sang nhap node khac hoac thoat";

GOC = TaoTree(node,du\_lieu); //Nhập node tạo cây

cout<<"Nhap du lieu nut can tim: "; cin>>du\_lieu;

P = TimNode(GOC, du\_lieu);// Gọi hàm tìm nút

if (P != NULL) cout<<"\n Da Tim thay nut co du lieu la "<<du\_lieu;

else cout<<" Khong co nut nao co du lieu la "<<du\_lieu<<" trong cay \n";

return 0;

}

**Câu 5:** Vẽ cây nhị phân biểu diễn biểu thức (B^2-4\*A\*C)/(2\*A), rồi dùng kiểu duyệt trung tự LDR , tiền tự DLR , hậu tự LRD để hiện ra các biểu thức mới tương ứng.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| ( | B | ^ | 2 | - | 4 | \* | A | \* | C | ) | / | ( | 2 | \* | A | ) |

`

1

2

3

5

4

6

7

8

9

10

13

15

12

14

11

16

17

Cấp 1

Cấp 2

Cấp 3

Cấp 4

Cấp 5

* Duyệt trung tự LDR: AA)2\*BC-)(/4(^2\*\*
* Duyệt tiền tự DLR: (B2AA)\*-C)^4/(\*2\*
* Duyệt hậu tự LRD: A)A\*2C)-B/(42\*\*^(

**Câu 6:** Sử dụng khai báo nút, khai báo cây nhị phân đã cho ở **Câu 3)**, viết hàm duyệt đệ quy cây nhị phân theo hậu tự LRD, theo trung tự LDR, theo tiền tự DLR.

// Hàm duyệt đệ quy cây nhị phân theo trung tự LDR

void InOrder(Tree GOC)

{ if(GOC!=NULL)

{InOrder(GOC->Lchild);

cout<<" "<<GOC->du\_lieu;//duyet goc

InOrder(GOC->Rchild);

}

}

// Hàm duyệt đệ quy cây nhị phân theo tiền tự DLR.

void PreOrder(Tree GOC)

{ if(GOC!=NULL)

{cout<<" "<<GOC->du\_lieu;

PreOrder(GOC->Lchild);

PreOrder(GOC->Rchild);

}

}

// Hàm duyệt đệ quy cây nhị phân theo hậu tự LRD.

void PostOrder(Tree GOC)

{ if(GOC!=NULL)

{PostOrder(GOC->Lchild);

PostOrder(GOC->Rchild);

cout<<" "<<GOC->du\_lieu;

}

}