Câu:

Cây cây tìm kiếm nhị phân mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Viết hàm kiểm tra cây có gốc là T có rỗng hay không?

**Nguyên mẫu (Prototype)**

* Tên hàm: **isEmpty**()
* Tham số: **T - Tree**
* Kiểu trả về: **int**

**Thân hàm (Body)**

* Nếu cây T rỗng, kết quả trả về là 1, ngược lại kết quả trả về là 0

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  if (isEmpty(T)){  printf("Cay rong");  }  else{  printf("Khong rong");  } | Cay rong |

Answer:

int isEmpty(Tree T){

return T==NULL;

}

**Câu:**

Cây cây tìm kiếm nhị phân mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Cho trước các phép toán:

- void insertNode(int x, Tree \*pT): thêm khóa x vào cây tìm kiếm nhị phân có gốc chỉ bởi con trỏ pT

Viết hàm liệt kê (in) các giá trị khóa trên đường đi của việc tìm kiếm một khóa x trong cây tìm kiếm nhị phân

**Nguyên mẫu (Prototype)**

* Tên hàm: **printPath**()
* Tham số:

        - x - int

        - T **- Tree**

* Kiểu trả về: **không**

**Thân hàm (Body)**

* Áp dụng giải thuật tìm kiếm trên cây tìm kiếm nhị phân, mỗi khi đi tới một nút trên theo giải thuật tìm, in giá trị khóa tại nút đó lên màn hình

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  Tree p;  int x;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  x=12;  printPath(x,T); | 27 12 -> Tim thay |
| Tree T=NULL;  Tree p;  int x;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  x=45;  printPath(x,T); | 27 40 50 -> Khong thay |

Answer:

/\*=== Xác định con trái của nút ===\*/

Tree LeftChild(Tree T)

{

if(T != NULL)

return T->Left;

else return NULL;

}

/\*=== Xác định con phải của nút ===\*/

Tree RightChild(Tree T)

{

if(T != NULL)

return T->Right;

else return NULL;

}

void printPath(KeyType x, Tree T)

{

KeyType temp;

if(T!=NULL){ //Kiem tra cay rong

temp = T->Key;

printf("%d ", temp);

if(x == temp) //tim thay khoa

//return T;

printf("-> Tim thay");

else if(x<temp) // Hy vong K nam ben trai

printPath(x,LeftChild(T));

else // Hy vong K nam ben phai

printPath(x,RightChild(T));

}else //return NULL;

printf("-> Khong thay");

}

Chieu cao cay:

<https://stackoverflow.com/questions/2597637/finding-height-in-binary-search-tree>

Cây [cây tìm kiếm nhị phân](https://else.ctu.edu.vn/mod/quiz/view.php?id=1976) mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Cho trước các phép toán:

- void insertNode(int x, Tree \*pT): thêm khóa x vào [cây tìm kiếm nhị phân](https://else.ctu.edu.vn/mod/quiz/view.php?id=1976) có gốc chỉ bởi con trỏ pT

Viết hàm tính chiều cao cây T

**Nguyên mẫu (Prototype)**

* Tên hàm: **getHeight**()
* Tham số:

        - T **-  Tree**

* Kiểu trả về: **int**

**Thân hàm (Body)**

* Nếu T rỗng kết quả trả về là -1, ngược lại kết quả trả về là chiều cao của nút gốc của cây

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**Gợi ý**: Chiều cao cây là độ dài đường đi từ nút gốc đến nút lá xa nhất

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  printf("Chieu cao cay %d",getHeight(T)); | Chieu cao cay 3 |
| Tree T=NULL;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(50,&T);  printf("Chieu cao cay %d",getHeight(T)); | Chieu cao cay 2 |

Answer:

/\*=== Xác định con trái của nút ===\*/

Tree LeftChild(Tree T)

{

if(T != NULL)

return T->Left;

else return NULL;

}

/\*=== Xác định con phải của nút ===\*/

Tree RightChild(Tree T)

{

if(T != NULL)

return T->Right;

else return NULL;

}

// Ham tim kiem nhi phan

Tree Search (int K, Tree T){

if(T!=NULL) //Kiem tra cay rong

if(K == T->Key) //tim thay khoa

return T;

else

if(K<T->Key) // Hy vong K nam ben trai

return Search(K,LeftChild(T));

else // Hy vong K nam ben phai

return Search(K,RightChild(T));

else return NULL;

}

int Max(int a, int b){

return (a>b)?a:b;

}

int getHeight(Tree T){

if(T==NULL) return -1;

else return 1 + Max(getHeight(T->Left), getHeight(T->Right));

}

int hNode(int x, Tree T){

Tree pT=Search(x, T);

if(pT==NULL) return -1;

else return getHeight(pT);

}

Cây cây tìm kiếm nhị phân mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Cho trước các phép toán:

- void insertNode(int x, Tree \*pT): thêm khóa x vào cây tìm kiếm nhị phân có gốc chỉ bởi con trỏ pT

Viết hàm duyệt hậu tự cây tìm kiếm nhị phân T

**Nguyên mẫu (Prototype)**

* Tên hàm: **posOrder**()
* Tham số: T **- Tree**
* Kiểu trả về: **không**

**Thân hàm (Body)**

* Liệt kê các khóa trên cây T theo biểu thức duyệt hậu tự

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  posOrder(T); | 30 34 20 4 50 40 |
| Tree T=NULL;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  posOrder(T); | 4 20 12 30 34 50 40 27 |

Answer:(penalty regime: 33.3, 66.7, ... %)

Tree LeftChild ( Tree T) {

if (T!=NULL) return T->Left; else return NULL;

}

Tree RightChild ( Tree T) {

if (T!=NULL) return T->Right ; else return NULL;

}

void posOrder( Tree T){

if (LeftChild(T) !=NULL) posOrder(LeftChild(T));

if (RightChild(T)!=NULL) posOrder(RightChild(T));

printf("%d ",T->Key);

}

Cây cây tìm kiếm nhị phân mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Cho trước các phép toán:

- void insertNode(int x, Tree \*pT): thêm khóa x vào cây tìm kiếm nhị phân có gốc chỉ bởi con trỏ pT

- void inOrder (Tree T): duyệt trung tự cây tìm kiếm nhị phân T

Viết hàm xóa một nút có khóa là x trong cây tìm kiếm nhị phân

**Nguyên mẫu (Prototype)**

* Tên hàm: **deleteNode**()
* Tham số:

        - x - int

        - pT **- con trỏ Tree**

* Kiểu trả về: **không**

**Thân hàm (Body)**

* Tiến hành xóa nút có khóa là x trong cây chỉ bởi con trỏ pT

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  int x;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  x=12;  deleteNode(x, &T);  inOrder(T); | 4 20 27 30 34 40 50 |
| Tree T=NULL;  int x;  insertNode(27,&T);  insertNode(12,&T);  insertNode(40,&T);  insertNode(4,&T);  insertNode(20,&T);  insertNode(34,&T);  insertNode(30,&T);  insertNode(50,&T);  x=50;  deleteNode(x, &T);  inOrder(T); | 4 12 20 27 30 34 |

int deleteMin (Tree \*pT) {

if ((\*pT)->Left==NULL) {

int K = (\*pT)->Key;

(\*pT) = (\*pT) ->Right;

return K;

}

else return deleteMin(&(\*pT)->Left);

}

void deleteNode(int x, Tree \*pT) {

if ((\*pT)!=NULL) {

if (x<(\*pT)->Key) deleteNode(x,&(\*pT)->Left);

else

if (x>(\*pT)->Key) deleteNode (x,&(\*pT)->Right);

else

if (((\*pT)->Left==NULL) && ((\*pT)->Right==NULL) )

(\*pT) =NULL;

else

if ((\*pT)->Left==NULL)

(\*pT) = (\*pT) ->Right;

else

if ((\*pT)->Right ==NULL ) (\*pT) = (\*pT) ->Left;

else

(\*pT) ->Key= deleteMin(&(\*pT)->Right);

}

}

ây cây tìm kiếm nhị phân mỗi khóa là 1 số nguyên được khai báo như sau:

struct Node{

      int Key;

      struct Node \*Left, \*Right;

};

Cho trước các phép toán:

- void preOrder(Tree T): duyệt tiền tự cây T

- void inOrder(Tree T): duyệt trung tự cây T

- void postOrder(Tree T): duyệt hậu tự cây T

Viết hàm thêm khóa X vào cây tìm kiếm nhị phân

**Nguyên mẫu (Prototype)**

* Tên hàm: **insertNode**()
* Tham số:

          - X - **int**

**-**pT -con trỏ **Tree**

* Kiểu trả về: **không**

**Thân hàm (Body)**

* Thêm khóa X vào cây tìm kiếm nhị phân chỉ bởi con trỏ pT (xem lại giải thuật trong phần lý thuyết)

**Chú ý**

* Chỉ viết hàm, KHÔNG VIẾT TOÀN BỘ CHƯƠNG TRÌNH

**For example:**

| **Test** | **Result** |
| --- | --- |
| Tree T=NULL;  insertNode(10,&T);  insertNode(-1,&T);  insertNode(15,&T);  insertNode(12,&T);  inOrder(T); | -1 10 12 15 |

Answer:(penalty regime: 33.3, 66.7, ... %)

void insertNode( int x, Tree \*pT) {

if ((\*pT)==NULL) {

(\*pT) = (struct Node\*) malloc(sizeof(struct Node));

(\*pT)->Key = x;

(\*pT)->Left= NULL;

(\*pT) ->Right= NULL;

}

else

if ((\*pT)->Key!=x) {

if ((\*pT)->Key>x) {

insertNode(x,&(\*pT)->Left);

}

else insertNode(x,&(\*pT)->Right);

}

}