BÀI TẬP VỀ CÂY NHỊ PHÂN

Bài 1:

#include <iostream>

using namespace std;

struct node

{

int info;

struct node \*pLeft;

struct node \*pRight;

};

typedef struct node NODE;

typedef NODE \*TREE;

void KhoiTaoCay(TREE &Root)

{

Root = NULL;

}

void ThemNode(TREE &Root, int x)

{

if (Root == NULL)

{

NODE \*p = new NODE;

p->info = x;

p->pLeft = NULL;

p->pRight = NULL;

Root = p;

}

else

{

if (x < Root->info)

ThemNode(Root->pLeft, x);

else

ThemNode(Root->pRight, x);

}

}

int DemNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemNode(Root->pLeft);

int b = DemNode(Root->pRight);

return (a + b + 1);

}

int TongNode(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongNode(Root->pLeft);

int b = TongNode(Root->pRight);

return (a + b + Root->info);

}

float TrungBinhCong(TREE Root)

{

int s = TongNode(Root);

int dem = DemNode(Root);

if (dem == 0)

return 0;

return (float)s / dem;

}

int DemDuong(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemDuong(Root->pLeft);

int b = DemDuong(Root->pRight);

if (Root->info > 0)

return (a + b + 1);

return (a + b);

}

int TongDuong(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongDuong(Root->pLeft);

int b = TongDuong(Root->pRight);

if (Root->info > 0)

return (a + b + Root->info);

return (a + b);

}

float TrungBinhDuong(TREE Root)

{

int s = TongDuong(Root);

int dem = DemDuong(Root);

if (dem == 0)

return 0;

return (float)s / dem;

}

int DemAm(TREE Root)

{

if (Root == NULL)

return 0;

int a = DemAm(Root->pLeft);

int b = DemAm(Root->pRight);

if (Root->info < 0)

return (a + b + 1);

return (a + b);

}

int TongAm(TREE Root)

{

if (Root == NULL)

return 0;

int a = TongAm(Root->pLeft);

int b = TongAm(Root->pRight);

if (Root->info < 0)

return (a + b + Root->info);

return (a + b);

}

float TrungBinhCongAm(TREE Root)

{

int s = TongAm(Root);

int dem = DemAm(Root);

if (dem == 0)

return 0;

return (float)s / dem;

}

float TinhTySo(TREE Root)

{

int a = TongDuong(Root);

int b = TongAm(Root);

if (b == 0)

return 0;

return (float)a / b;

}

void LNR(TREE Root)

{

if (Root != NULL)

{

LNR(Root->pLeft);

cout << Root->info << " ";

LNR(Root->pRight);

}

}

int main()

{

TREE Root;

KhoiTaoCay(Root);

int arr[] = {10, -5, 15, -10, 7, 20, -3, 8, 12};

int n = sizeof(arr) / sizeof(arr[0]);

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

{

ThemNode(Root, arr[i]);

}

cout << "Cay duyet theo LNR: ";

LNR(Root);

cout << endl;

cout << "Tong cac nut trong cay: " << TongNode(Root) << endl;

cout << "So luong nut trong cay: " << DemNode(Root) << endl;

cout << "Trung binh cong cac nut trong cay: " << TrungBinhCong(Root) << endl;

cout << "Tong cac nut duong: " << TongDuong(Root) << endl;

cout << "So luong nut duong: " << DemDuong(Root) << endl;

cout << "Trung binh cong cac nut duong: " << TrungBinhDuong(Root) << endl;

cout << "Tong cac nut am: " << TongAm(Root) << endl;

cout << "So luong nut am: " << DemAm(Root) << endl;

cout << "Trung binh cong cac nut am: " << TrungBinhCongAm(Root) << endl;

cout << "Ty so tong duong / tong am: " << TinhTySo(Root) << endl;

return 0;

}

Bài 2:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void LNR(TREE root) {

if (root != NULL) {

LNR(root->pLeft);

cout << root->info << " ";

LNR(root->pRight);

}

}

void SwapTree(TREE &p) {

if (p == NULL) return;

NODE\* temp = p->pLeft;

p->pLeft = p->pRight;

p->pRight = temp;

SwapTree(p->pLeft);

SwapTree(p->pRight);

}

int main() {

TREE root = NULL;

int arr[] = {10, 5, 15, 3, 7, 12, 18, 20,9};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(root, arr[i]);

}

cout << "Cay ban dau (LNR): ";

LNR(root);

cout << endl;

SwapTree(root);

cout << "Cay sau khi hoan doi (LNR): ";

LNR(root);

cout << endl;

return 0;

}

Bài 3:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void LRN(TREE root) {

if (root != NULL) {

LRN(root->pLeft);

LRN(root->pRight);

cout << root->info << " ";

}

}

int main() {

TREE root = NULL;

int arr[] = {10, 5, 15, 3, 9, 12, 18, 7, 20};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(root, arr[i]);

}

cout << "Ket qua duyet hau tu (LRN): ";

LRN(root);

cout << endl;

return 0;

}

Bài 4:

#include<iostream>

using namespace std;

struct NODE{

int info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(int x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,int x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

NODE\*NhoNhat(TREE Root){

if(Root==NULL) return NULL;

NODE\*lc=Root;

while(lc->pLeft) lc=lc->pLeft;

return lc;

}

NODE\*LonNhat(TREE Root){

if(Root==NULL) return NULL;

NODE\*lc=Root;

while(lc->pRight) lc=lc->pRight;

return lc;

}

void LNR(TREE root){

if(root!=NULL){

LNR(root->pLeft);

cout<<root->info<<" ";

LNR(root->pRight);

}

}

int main(){

TREE root=NULL;

int arr[]={10,5,15,3,9,12,18,7,20};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo LNR: ";

LNR(root);

cout<<endl;

NODE\*minNode=NhoNhat(root);

NODE\*maxNode=LonNhat(root);

if(minNode) cout<<"Phan tu nho nhat: "<<minNode->info<<endl;

if(maxNode) cout<<"Phan tu lon nhat: "<<maxNode->info<<endl;

return 0;

}

Bài 5:

#include<iostream>

using namespace std;

struct NODE{

int info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(int x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,int x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

int DemMotCon(TREE t){

if(t==NULL) return 0;

if((t->pLeft&&!t->pRight)||(!t->pLeft&&t->pRight))

return 1+DemMotCon(t->pLeft)+DemMotCon(t->pRight);

return DemMotCon(t->pLeft)+DemMotCon(t->pRight);

}

void NLR(TREE root){

if(root!=NULL){

cout<<root->info<<" ";

NLR(root->pLeft);

NLR(root->pRight);

}

}

int main(){

TREE root=NULL;

int arr[]={10,5,15,3,9,12,18,7,20};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo NLR: ";

NLR(root);

cout<<endl;

cout<<"So nut mot con: "<<DemMotCon(root)<<endl;

return 0;

}

Bài 6:

#include<iostream>

using namespace std;

struct NODE{

int info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(int x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,int x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

int DemNode(TREE t){

if(t==NULL) return 0;

return 1+DemNode(t->pLeft)+DemNode(t->pRight);

}

int TongNode(TREE t){

if(t==NULL) return 0;

return t->info+TongNode(t->pLeft)+TongNode(t->pRight);

}

void NLR(TREE root){

if(root!=NULL){

cout<<root->info<<" ";

NLR(root->pLeft);

NLR(root->pRight);

}

}

int main(){

TREE root=NULL;

int arr[]={10,5,15,3,9,12,18,7,20};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo NLR: ";

NLR(root);

cout<<endl;

cout<<"So nut trong cay: "<<DemNode(root)<<endl;

cout<<"Tong gia tri cac nut: "<<TongNode(root)<<endl;

return 0;

}

Bài 7:

• Giống nhau:

• CTDL động.

• Các thao tác cơ bản Thêm, Xóa, Cập Nhật được thực hiện một

cách linh hoạt.

• Khác nhau

• Dữ liệu trên cây NPTK được tổ chức và dslk đơn thì không.

• Chi phí tìm kiếm, thêm trên cây nhanh hơn trên dslk đơn.

Bài 8:

#include<iostream>

#include<fstream>

using namespace std;

struct NODE{

float info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(float x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,float x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

void LNR(TREE t,FILE\*fp){

if(t==NULL) return;

LNR(t->pLeft,fp);

fwrite(&t->info,sizeof(float),1,fp);

LNR(t->pRight,fp);

}

int Xuat(const char\*filename,TREE t){

FILE\*fp=fopen(filename,"wb");

if(fp==NULL) return 0;

LNR(t,fp);

fclose(fp);

return 1;

}

void LNR\_Display(TREE t){

if(t==NULL) return;

LNR\_Display(t->pLeft);

cout<<t->info<<" ";

LNR\_Display(t->pRight);

}

int main(){

TREE root=NULL;

float arr[]={10.5,5.2,15.7,3.1,9.8,12.4,18.6,7.3,20.9};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo LNR: ";

LNR\_Display(root);

cout<<endl;

if(Xuat("data.out",root)) cout<<"Xuat file thanh cong!"<<endl;

else cout<<"Loi khi mo file!"<<endl;

return 0;

}

Bài 9:

#include<iostream>

#include<fstream>

using namespace std;

struct NODE{

float info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(float x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,float x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

void NLR(TREE t,FILE\*fp){

if(t==NULL) return;

fwrite(&t->info,sizeof(float),1,fp);

NLR(t->pLeft,fp);

NLR(t->pRight,fp);

}

int Xuat(const char\*filename,TREE t){

FILE\*fp=fopen(filename,"wb");

if(fp==NULL) return 0;

NLR(t,fp);

fclose(fp);

return 1;

}

void NLR\_Display(TREE t){

if(t==NULL) return;

cout<<t->info<<" ";

NLR\_Display(t->pLeft);

NLR\_Display(t->pRight);

}

int main(){

TREE root=NULL;

float arr[]={10.5,5.2,15.7,3.1,9.8,12.4,18.6,7.3,20.9};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo NLR: ";

NLR\_Display(root);

cout<<endl;

if(Xuat("data.out",root)) cout<<"Xuat file thanh cong!"<<endl;

else cout<<"Loi khi mo file!"<<endl;

return 0;

}

Bài 10:

#include<iostream>

#include<fstream>

using namespace std;

struct NODE{

float info;

NODE\*pLeft;

NODE\*pRight;

};

typedef NODE\*TREE;

NODE\*CreateNode(float x){

NODE\*p=new NODE;

p->info=x;

p->pLeft=p->pRight=NULL;

return p;

}

void InsertNode(TREE &root,float x){

if(root==NULL){

root=CreateNode(x);

return;

}

if(x<root->info) InsertNode(root->pLeft,x);

else if(x>root->info) InsertNode(root->pRight,x);

}

void LRN(TREE t,FILE\*fp){

if(t==NULL) return;

LRN(t->pLeft,fp);

LRN(t->pRight,fp);

fwrite(&t->info,sizeof(float),1,fp);

}

int Xuat(const char\*filename,TREE t){

FILE\*fp=fopen(filename,"wb");

if(fp==NULL) return 0;

LRN(t,fp);

fclose(fp);

return 1;

}

void LRN\_Display(TREE t){

if(t==NULL) return;

LRN\_Display(t->pLeft);

LRN\_Display(t->pRight);

cout<<t->info<<" ";

}

int main(){

TREE root=NULL;

float arr[]={10.5,5.2,15.7,3.1,9.8,12.4,18.6,7.3,20.9};

int n=sizeof(arr)/sizeof(arr[0]);

for(int i=0;i<n;i++) InsertNode(root,arr[i]);

cout<<"Cay BST theo LRN: ";

LRN\_Display(root);

cout<<endl;

if(Xuat("data.out",root)) cout<<"Xuat file thanh cong!"<<endl;

else cout<<"Loi khi mo file!"<<endl;

return 0;

}

Bài 11:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void LNR(TREE root) {

if (root != NULL) {

LNR(root->pLeft);

cout << root->info << " ";

LNR(root->pRight);

}

}

int main() {

TREE root = NULL;

int arr[] = {10, 5, 15, 3, 9, 7, 12, 18, 20};

for (int i = 0; i < 9; i++) {

InsertNode(root, arr[i]);

}

LNR(root);

return 0;

}

Bài 12:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void RNL(TREE root) {

if (root != NULL) {

RNL(root->pRight);

cout << root->info << " ";

RNL(root->pLeft);

}

}

int main() {

TREE root = NULL;

int arr[] = {10, 5, 15, 3, 9, 7, 12, 18, 20};

for (int i = 0; i < 9; i++) {

InsertNode(root, arr[i]);

}

RNL(root);

return 0;

}

Bài 13:

#include <iostream>

#include <cstdio> // S? d?ng cstdio thay vì fstream d? dùng FILE\*

using namespace std;

struct NODE {

float info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(float x) {

NODE\* p = new NODE;

if (p) {

p->info = x;

p->pLeft = p->pRight = NULL;

}

return p;

}

void InsertNode(TREE &root, float x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void NLR(TREE t, FILE\* fp) {

if (t == NULL) return;

fwrite(&t->info, sizeof(float), 1, fp);

NLR(t->pLeft, fp);

NLR(t->pRight, fp);

}

int Xuat(const char \*filename, TREE t) {

FILE\* fp = fopen(filename, "wb");

if (fp == NULL) {

cerr << "Khong mo duoc file!" << endl;

return 0;

}

NLR(t, fp);

fclose(fp);

return 1;

}

int main() {

TREE root = NULL;

float arr[] = {10.5, 5.2, 15.8, 3.1, 9.7, 7.6, 12.4, 18.9, 20.3};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(root, arr[i]);

}

if (Xuat("data.out", root)) {

cout << "Da luu cay vao file data.out" << endl;

}

return 0;

}

Bài 14:

#include <iostream>

using namespace std;

struct NODETREE {

int info;

NODETREE\* pLeft;

NODETREE\* pRight;

};

typedef NODETREE\* TREE;

struct NODELIST {

int info;

NODELIST\* pNext;

};

typedef struct NODELIST NODELIST;

struct LIST {

NODELIST\* pHead;

NODELIST\* pTail;

};

typedef struct LIST LIST;

void Init(LIST &l) {

l.pHead = l.pTail = NULL;

}

NODELIST\* GetNode(int x) {

NODELIST\* p = new NODELIST;

if (p == NULL) return NULL;

p->info = x;

p->pNext = NULL;

return p;

}

void AddTail(LIST &l, NODELIST\* p) {

if (l.pHead == NULL)

l.pHead = l.pTail = p;

else {

l.pTail->pNext = p;

l.pTail = p;

}

}

void RNL(TREE Root, LIST &l) {

if (Root == NULL) return;

RNL(Root->pRight, l);

NODELIST\* p = GetNode(Root->info);

if (p != NULL) AddTail(l, p);

RNL(Root->pLeft, l);

}

void BuildList(TREE Root, LIST &l) {

Init(l);

RNL(Root, l);

}

NODETREE\* CreateNode(int x) {

NODETREE\* p = new NODETREE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void PrintList(LIST l) {

NODELIST\* p = l.pHead;

while (p != NULL) {

cout << p->info << " ";

p = p->pNext;

}

cout << endl;

}

int main() {

TREE root = NULL;

int arr[] = {10, 5, 15, 3, 9, 7, 12, 18, 20};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(root, arr[i]);

}

LIST l;

BuildList(root, l);

PrintList(l);

return 0;

}

Bài 15:

• Giống nhau

• Dữ liệu được tổ chức.

• Chi phí tìm kiếm một phần tử trên cả hai ctdl là như nhau.

• Khác nhau

• Chi phí thêm và xoá phần tử vào mảng lớn hơn chi phí cây nhị

phân tìm kiếm.

Bài 16:

#include <iostream>

using namespace std;

struct BST\_NODE {

int Key;

int So\_lan;

BST\_NODE\* Left;

BST\_NODE\* Right;

};

struct BST\_TREE {

BST\_NODE\* pRoot;

};

BST\_NODE\* CreateNode(int x) {

BST\_NODE\* p = new BST\_NODE;

p->Key = x;

p->So\_lan = 1;

p->Left = p->Right = NULL;

return p;

}

void InsertNode(BST\_NODE\*& Root, int x) {

if (Root == NULL) {

Root = CreateNode(x);

return;

}

if (x < Root->Key)

InsertNode(Root->Left, x);

else if (x > Root->Key)

InsertNode(Root->Right, x);

else

Root->So\_lan++;

}

int DeleteNode(BST\_NODE\*& Root, int x) {

if (Root == NULL) return 0;

if (x < Root->Key) return DeleteNode(Root->Left, x);

if (x > Root->Key) return DeleteNode(Root->Right, x);

if (Root->So\_lan > 1) {

Root->So\_lan--;

return 1;

}

if (Root->Left == NULL || Root->Right == NULL) {

BST\_NODE\* temp = (Root->Left != NULL) ? Root->Left : Root->Right;

delete Root;

Root = temp;

} else {

BST\_NODE\* temp = Root->Right;

while (temp->Left != NULL) temp = temp->Left;

Root->Key = temp->Key;

Root->So\_lan = temp->So\_lan;

temp->So\_lan = 1;

DeleteNode(Root->Right, temp->Key);

}

return 1;

}

void XoaGiaTri(BST\_TREE& t, int x) {

int kq = DeleteNode(t.pRoot, x);

if (kq == 0)

cout << "Khong ton tai " << x << endl;

else

cout << "Xoa thanh cong " << x << endl;

}

void NLR(BST\_NODE\* Root) {

if (Root == NULL) return;

if (Root->So\_lan > 0)

cout << Root->Key << " (" << Root->So\_lan << ") ";

NLR(Root->Left);

NLR(Root->Right);

}

void LietKe(BST\_TREE t) {

NLR(t.pRoot);

cout << endl;

}

int main() {

BST\_TREE t;

t.pRoot = NULL;

int arr[] = {10, 5, 15, 3, 9, 7, 12, 18, 20, 10, 9, 15};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(t.pRoot, arr[i]);

}

cout << "Cay BST sau khi them: ";

LietKe(t);

XoaGiaTri(t, 10);

cout << "Cay BST sau khi xoa 10: ";

LietKe(t);

XoaGiaTri(t, 15);

cout << "Cay BST sau khi xoa 15: ";

LietKe(t);

XoaGiaTri(t, 100);

return 0;

}

Bài 17:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

NODE\* CreateNode(int x) {

NODE\* p = new NODE;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

void InsertNode(TREE &root, int x) {

if (root == NULL) {

root = CreateNode(x);

return;

}

if (x < root->info)

InsertNode(root->pLeft, x);

else if (x > root->info)

InsertNode(root->pRight, x);

}

void SearchStandFor(TREE &p, TREE &q) {

if (q->pLeft) {

SearchStandFor(p, q->pLeft);

} else {

p->info = q->info;

p = q;

q = q->pRight;

}

}

void DeleteNode(TREE &root, int x) {

if (root == NULL) return;

if (x < root->info) {

DeleteNode(root->pLeft, x);

} else if (x > root->info) {

DeleteNode(root->pRight, x);

} else {

NODE\* p = root;

if (root->pLeft == NULL) {

root = root->pRight;

} else if (root->pRight == NULL) {

root = root->pLeft;

} else {

SearchStandFor(p, root->pRight);

}

delete p;

}

}

void NLR(TREE root) {

if (root == NULL) return;

cout << root->info << " ";

NLR(root->pLeft);

NLR(root->pRight);

}

int main() {

TREE root = NULL;

int arr[] = {65, 35, 80, 21, 44, 77, 92, 10, 70, 5, 17, 28, 55, 48, 53, 60, 59, 97, 94, 93, 95, 98, 90, 68, 66, 69, 72, 74};

int n = sizeof(arr) / sizeof(arr[0]);

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

InsertNode(root, arr[i]);

}

cout << "Cay truoc khi xoa: ";

NLR(root);

cout << endl;

int x = 92;

DeleteNode(root, x);

cout << "Cay sau khi xoa " << x << ": ";

NLR(root);

cout << endl;

return 0;

}

Bài 18:

#include <iostream>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

void Init(TREE &t) {

t = NULL;

}

NODE\* GetNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

int InsertNode(TREE &t, int x) {

if (t) {

if (t->info == x) return 0;

if (t->info < x) return InsertNode(t->pRight, x);

return InsertNode(t->pLeft, x);

}

t = GetNode(x);

if (t == NULL) return -1;

return 1;

}

int TaoCay(TREE &t, int a[], int n) {

Init(t);

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

if (InsertNode(t, a[i]) == -1) return 0;

}

return 1;

}

void LNR(TREE t, int a[], int &n) {

if (t == NULL) return;

LNR(t->pLeft, a, n);

a[n++] = t->info;

LNR(t->pRight, a, n);

}

void SortArrayUsingBST(int a[], int n) {

TREE t;

TaoCay(t, a, n);

int index = 0;

LNR(t, a, index);

}

int main() {

int a[] = {64, 37, 78, 10, 57, 69, 82, 6, 67, 93};

int n = sizeof(a) / sizeof(a[0]);

SortArrayUsingBST(a, n);

cout << "Mang sau khi sap xep: ";

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

cout << a[i] << " ";

}

cout << endl;

return 0;

}

Bài 33:

#include <iostream>

#include <stack>

#include <queue>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

void Init(TREE &t) {

t = NULL;

}

NODE\* GetNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

int InsertNode(TREE &t, int x) {

if (t) {

if (t->info == x) return 0;

if (t->info < x) return InsertNode(t->pRight, x);

return InsertNode(t->pLeft, x);

}

t = GetNode(x);

if (t == NULL) return -1;

return 1;

}

void NLR\_Stack(TREE t) {

if (t == NULL) return;

stack<TREE> s;

s.push(t);

while (!s.empty()) {

TREE p = s.top();

s.pop();

cout << p->info << " ";

if (p->pRight) s.push(p->pRight);

if (p->pLeft) s.push(p->pLeft);

}

}

void LevelOrder\_Queue(TREE t) {

if (t == NULL) return;

queue<TREE> q;

q.push(t);

while (!q.empty()) {

TREE p = q.front();

q.pop();

cout << p->info << " ";

if (p->pLeft) q.push(p->pLeft);

if (p->pRight) q.push(p->pRight);

}

}

int main() {

TREE t;

Init(t);

int a[] = {50, 30, 70, 20, 40, 60, 80};

int n = sizeof(a) / sizeof(a[0]);

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

InsertNode(t, a[i]);

}

cout << "Duyet NLR (dung stack): ";

NLR\_Stack(t);

cout << endl;

cout << "Duyet theo muc (dung queue): ";

LevelOrder\_Queue(t);

cout << endl;

return 0;

}

Bài 37:

#include <iostream>

#include <cmath>

using namespace std;

struct NODE {

int info;

NODE\* pLeft;

NODE\* pRight;

};

typedef NODE\* TREE;

void Init(TREE &t) {

t = NULL;

}

NODE\* GetNode(int x) {

NODE\* p = new NODE;

if (p == NULL) return NULL;

p->info = x;

p->pLeft = p->pRight = NULL;

return p;

}

int InsertNode(TREE &t, int x) {

if (t) {

if (t->info == x) return 0;

if (t->info < x) return InsertNode(t->pRight, x);

return InsertNode(t->pLeft, x);

}

t = GetNode(x);

if (t == NULL) return -1;

return 1;

}

int ChieuCao(TREE t) {

if (t == NULL) return 0;

int a = ChieuCao(t->pLeft);

int b = ChieuCao(t->pRight);

return max(a, b) + 1;

}

bool ktCanBang(TREE Root) {

if (Root == NULL) return true;

int x = ChieuCao(Root->pLeft);

int y = ChieuCao(Root->pRight);

if (abs(x - y) > 1) return false;

return ktCanBang(Root->pLeft) && ktCanBang(Root->pRight);

}

int main() {

TREE t;

Init(t);

int a[] = {50, 30, 70, 20, 40, 60, 80};

int n = sizeof(a) / sizeof(a[0]);

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

InsertNode(t, a[i]);

}

if (ktCanBang(t))

cout << "Cay can bang\n";

else

cout << "Cay khong can bang\n";

return 0;

}