Báo Cáo Tiến độ tuần các công việc thực hiện trong tuần 3

Tên: Lê Lâm Trường

Lớp: 60.CNTT-2

MSSV: 60131927

Đề Tài: Cài đặt và thao tác trên cây nhị phân theo hướng đối tượng.

**Tiến Hành cài đặt thuật toán:**

+ Cài đặt và chạy trên Dev C với cách xuất 2D lên màn hình console.

+ Đọc và tìm hiểu cách Cài Trên C# để tiến hành viết winform.

+ Code còn thiếu nhập từ file và ramdom(em vẫn còn đang tìm hiểu vì cách đó, em chạy nhưng bị sai ạ)

+Em xin gửi cô Code C++ cây nhị phân, code C# và winform em vẫn đang tìm hiểu, tuần sau em sẽ hoàn thành đầy đủ ạ.

**Code:**

#include<iostream>

#define SPACE 10

using namespace std;

//Tao class cay nhi phan

class TreeNode {

// khai bao cac thuoc tinh cho cay

public:

int value;

TreeNode \* left;

TreeNode \* right;

// khai bao cau truc cua 1 cay

TreeNode() {

value = 0;

left = NULL;

right = NULL;

}

//khai bao cau truc cua 1 cay gia tri v

TreeNode(int v) {

value = v;

left = NULL;

right = NULL;

}

};

class BST {

public:

TreeNode \* root;

BST() {

root = NULL;

}

bool isTreeEmpty() {

if (root == NULL) {

return true;

} else {

return false;

}

}

void insertNode(TreeNode \* new\_node) {

if (root == NULL)

// neu nut goc == NULLL

{

root = new\_node;

cout << "Da chen gia tri vao nut goc" << endl;

} else {

TreeNode \* temp = root; // khai bao bien con tro temp(dia chi) tro vao nut goc

while (temp != NULL) {

// neu gia tri nut can chen bang voi gia tri cua nut trong cay

if (new\_node -> value == temp -> value) {

cout << "Gia tri da ton tai" <<

"Hay chen gia tri khac" << endl;

return;

} else if ((new\_node -> value < temp -> value) && (temp -> left == NULL))

// neu gia tri nut can chen be hon gia tri cua nut dang tro va cay con trai nut do bang NULL

{

temp -> left = new\_node;

cout << "Da chen gia tri vao ben trai!" << endl;

break;

} else if (new\_node -> value < temp -> value)

// neu gia tri nut can chen be hon gia tri cua nut dang tro va cay con trai nut do khac NULL

{

temp = temp -> left;

} else if ((new\_node -> value > temp -> value) && (temp -> right == NULL))

// neu gia tri nut can chen lon hon gia tri cua nut dang tro va cay con phai nut do bang NULL

{

temp -> right = new\_node;

cout << "Da chen gia tri vao ben phai!" << endl;

break;

} else

// neu gia tri nut can chen lon hon gia tri cua nut dang tro va cay con phai nut do khac NULL

{

temp = temp -> right;

}

}

}

}

void print2D(TreeNode \* r, int space) {

if (r == NULL) // Base case 1

return;

space += SPACE; // Increase distance between levels 2

print2D(r -> right, space); // Process right child first 3

cout << endl;

for (int i = SPACE; i < space; i++) // 5

cout << " "; // 5.1

cout << r -> value << "\n"; // 6

print2D(r -> left, space); // Process left child 7

}

void printPreorder(TreeNode \* r) //(current node, Left, Right)

{

if (r == NULL)

return;

/\* first print data of node \*/

cout << r -> value << " ";

/\* then recur on left sutree \*/

printPreorder(r -> left);

/\* now recur on right subtree \*/

printPreorder(r -> right);

}

void printInorder(TreeNode \* r) // (Left, current node, Right)

{

if (r == NULL)

return;

/\* first recur on left child \*/

printInorder(r -> left);

/\* then print the data of node \*/

cout << r -> value << " ";

/\* now recur on right child \*/

printInorder(r -> right);

}

void printPostorder(TreeNode \* r) //(Left, Right, Root)

{

if (r == NULL)

return;

// first recur on left subtree

printPostorder(r -> left);

// then recur on right subtree

printPostorder(r -> right);

// now deal with the node

cout << r -> value << " ";

}

TreeNode \* iterativeSearch(int v) {

if (root == NULL) {

return root;

} else {

TreeNode \* temp = root;

while (temp != NULL) {

if (v == temp -> value) {

return temp;

} else if (v < temp -> value) {

temp = temp -> left;

} else {

temp = temp -> right;

}

}

return NULL;

}

}

TreeNode \* recursiveSearch(TreeNode \* r, int val) {

if (r == NULL || r -> value == val)

return r;

else if (val < r -> value)

return recursiveSearch(r -> left, val);

else

return recursiveSearch(r -> right, val);

}

int height(TreeNode \* r) {

if (r == NULL)

return -1;

else {

/\* compute the height of each subtree \*/

int lheight = height(r -> left);

int rheight = height(r -> right);

/\* use the larger one \*/

if (lheight > rheight)

return (lheight + 1);

else return (rheight + 1);

}

}

/\* Print nodes at a given level \*/

void printGivenLevel(TreeNode \* r, int level) {

if (r == NULL)

return;

else if (level == 0)

cout << r -> value << " ";

else // level > 0

{

printGivenLevel(r -> left, level - 1);

printGivenLevel(r -> right, level - 1);

}

}

void printLevelOrderBFS(TreeNode \* r) {

int h = height(r);

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

printGivenLevel(r, i);

}

// tim nut trai nhat cua cay con phai//

TreeNode \* minValueNode(TreeNode \* node) {

TreeNode \* current = node;// current : hien hanh

// vòng xuong tìm nut la ngoai cung ben trai

while (current -> left != NULL) {

current = current -> left;

}

return current;

}

TreeNode \* deleteNode(TreeNode \* r, int v) {

// kiem tra xem nut goc co NULL khong ?

if (r == NULL) {

return NULL;

}

// Neu nut can xoa be hon nut goc

// thi nut can xoa la cay con trai cua nut goc

else if (v < r -> value) {

r -> left = deleteNode(r -> left, v);

}

// Neu nut can xoa lon hon nut goc

// thi nut can xoa la cay con phai cua nut goc

else if (v > r -> value) {

r -> right = deleteNode(r -> right, v);

}

// neu nut can xoa bang nut goc

else {

// neu nut co 1 hoac 0 co cay con

if (r -> left == NULL) {

TreeNode \* temp = r -> right;// gan dia chi nut goc chinh la cay con phai, de khi xoa thi nut nay se nhay len thay the vi tri cho nut goc do

delete r;

return temp;

} else if (r -> right == NULL) {

TreeNode \* temp = r -> left;// gan dia chi nut goc chinh la cay con trai, de khi xoa thi nut nay se nhay len thay the vi tri cho nut goc do

delete r;

return temp;

} else {

//neu nut can xoa co 2 cay con

//Ta thuc hien tim nut trai nhat cua cay con phai

TreeNode \* temp = minValueNode(r -> right);

//lay nut nay lam nut thay the cho nut can xoa

r -> value = temp -> value;

// xoa nut thay the

r -> right = deleteNode(r -> right, temp -> value);

}

}

return r;

}

};

int main() {

BST obj;

int option, val;

do {

cout << "Ban muon lua chon nao ? " <<

" Hay chon 1 so trong menu. Nhan 0 de thoat" << endl;

cout << "1. Insert Node" << endl;

cout << "2. Search Node" << endl;

cout << "3. Delete Node" << endl;

cout << "4. Print/Traversal BST values" << endl;

cout << "5. Height of Tree" << endl;

cout << "6. Clear Screen" << endl;

cout << "7. Insert file" << endl;

cout << "0. Exit Program" << endl;

cin >> option;

//Node n1;

TreeNode \* new\_node = new TreeNode();

switch (option) {

case 0:

break;

case 1:

cout << "INSERT" << endl;

cout << "Nhap gia tri muon chen vao cay: ";

cin >> val;

new\_node -> value = val;

obj.insertNode(new\_node);

cout << endl;

break;

case 2:

cout << "SEARC" << endl;

cout << "Nhap gia tri can tim: ";

cin >> val;

//new\_node = obj.iterativeSearch(val);

new\_node = obj.recursiveSearch(obj.root, val);

if (new\_node != NULL) {

cout << "Nut co gia tri can tim duoc tim thay" << endl;

} else {

cout << "Khong tim thay nut co gia tri can tim" << endl;

}

break;

case 3:

cout << "DELETE" << endl;

cout << "Nhap gia tri muon xoa: ";

cin >> val;

new\_node = obj.iterativeSearch(val);

if (new\_node != NULL) {

obj.deleteNode(obj.root, val);

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

} else {

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

}

break;

case 4:

cout << "PRINT 2D: " << endl;

obj.print2D(obj.root, 5);

cout << endl;

cout << "Print Level Order BFS: \n";

obj.printLevelOrderBFS(obj.root);

cout << endl;

// cout <<"PRE-ORDER: ";

// obj.printPreorder(obj.root);

// cout<<endl;

// cout <<"IN-ORDER: ";

// obj.printInorder(obj.root);

// cout<<endl;

// cout <<"POST-ORDER: ";

// obj.printPostorder(obj.root);

break;

case 5:

cout << "TREE HEIGHT" << endl;

cout << "Height : " << obj.height(obj.root) << endl;

break;

case 6:

system("cls");

break;

default:

cout << "Nhap vao con so tren menu lua chon " << endl;

}

} while (option != 0);

return 0;

}