Trường Đại học Sài Gòn

**Khoa công nghệ thông tin**

A blue circle with purple letters

Description automatically generated

**CTDLGT25-[Thuc hanh 8]-Cai dat cay AVL chua cac so nguyen**

Sinh viên thực hiện: Bùi Viết Quang Vinh

MSSV:3124411347

#include <iostream>

using namespace std;

int COUNT = 10;

// Định nghĩa kiểu dữ liệu node

struct *Node* {

    int data;

*Node* \*left, \*right;

    int height;

};

// Hàm lấy chiều cao của node

int GetHeight(*Node* \**root*) {

    if (!*root*) return 0;

    return *root*->height;

}

// Hàm quay phải

*Node*\* RightRotate(*Node* \**root*) {

*Node* \*x = *root*->left;

*root*->left = x->right;

    x->right = *root*;

*root*->height = 1 + max(GetHeight(*root*->left), GetHeight(*root*->right));

    x->height = 1 + max(GetHeight(x->left), GetHeight(x->right));

    return x;

}

// Hàm quay trái

*Node*\* LeftRotate(*Node* \**root*) {

*Node* \*y = *root*->right;

*root*->right = y->left;

    y->left = *root*;

*root*->height = 1 + max(GetHeight(*root*->left), GetHeight(*root*->right));

    y->height = 1 + max(GetHeight(y->left), GetHeight(y->right));

    return y;

}

// Hàm cân bằng cây

int GetBalance(*Node* \**root*) {

    if (!*root*) return 0;

    return GetHeight(*root*->left) - GetHeight(*root*->right);

}

// Hàm chèn node vào cây AVL

*Node*\* Insert(*Node* \**root*, int *value*) {

    if (!*root*) return **new** *Node*{*value*, nullptr, nullptr, 1};

    if (*value* < *root*->data) *root*->left = Insert(*root*->left, *value*);

    else if (*value* > *root*->data) *root*->right = Insert(*root*->right, *value*);

    else return *root*;

*root*->height = 1 + max(GetHeight(*root*->left), GetHeight(*root*->right));

    int balance = GetBalance(*root*);

    // Các trường hợp mất cân bằng

    if (balance > 1 && *value* < *root*->left->data) return RightRotate(*root*);

    if (balance < -1 && *value* > *root*->right->data) return LeftRotate(*root*);

    if (balance > 1 && *value* > *root*->left->data) {

*root*->left = LeftRotate(*root*->left);

        return RightRotate(*root*);

    }

    if (balance < -1 && *value* < *root*->right->data) {

*root*->right = RightRotate(*root*->right);

        return LeftRotate(*root*);

    }

    return *root*;

}

// Hàm tìm node có giá trị nhỏ nhất

*Node*\* MinValueNode(*Node*\* *root*) {

    while (*root*->left) *root* = *root*->left;

    return *root*;

}

// Hàm xóa node trong AVL

*Node*\* Delete(*Node*\* *root*, int *value*) {

    if (!*root*) return *root*;

    if (*value* < *root*->data) *root*->left = Delete(*root*->left, *value*);

    else if (*value* > *root*->data) *root*->right = Delete(*root*->right, *value*);

    else {

        if (!*root*->left || !*root*->right) {

*Node* \*temp = *root*->left ? *root*->left : *root*->right;

**delete** *root*;

            return temp;

        } else {

*Node* \*temp = MinValueNode(*root*->right);

*root*->data = temp->data;

*root*->right = Delete(*root*->right, temp->data);

        }

    }

*root*->height = 1 + max(GetHeight(*root*->left), GetHeight(*root*->right));

    int balance = GetBalance(*root*);

    if (balance > 1 && GetBalance(*root*->left) >= 0) return RightRotate(*root*);

    if (balance > 1 && GetBalance(*root*->left) < 0) {

*root*->left = LeftRotate(*root*->left);

        return RightRotate(*root*);

    }

    if (balance < -1 && GetBalance(*root*->right) <= 0) return LeftRotate(*root*);

    if (balance < -1 && GetBalance(*root*->right) > 0) {

*root*->right = RightRotate(*root*->right);

        return LeftRotate(*root*);

    }

    return *root*;

}

// Hàm in cây

void Print(*Node* \**root*, int *space*) {

    if (!*root*) return;

*space* += COUNT;

    Print(*root*->right, *space*);

    cout << "\n";

    for (int i = COUNT; i < *space*; i++) cout << " ";

    cout << *root*->data << "(" << *root*->height << ")" << "\n";

    Print(*root*->left, *space*);

}

int main() {

*Node*\* tree = nullptr;

    tree = Insert(tree, 4);

    tree = Insert(tree, 2);

    tree = Insert(tree, 34);

    tree = Insert(tree, 30);

    tree = Insert(tree, 25);

    tree = Insert(tree, 69);

    tree = Insert(tree, 23);

    cout << "Cay AVL sau khi chen:\n";

    Print(tree, 0);

    tree = Delete(tree, 34);

    cout << "\nCay AVL sau khi xoa 34:\n";

    Print(tree, 0);

    return 0;

}