Aim-types of order

#include <iostream>

using namespace std;

// Node structure for tree

struct Node {

int data;

Node\* left;

Node\* right;

Node(int value) : data(value), left(nullptr), right(nullptr) {}

};

class BinaryTree {

private:

Node\* root;

void inorderTraversal(Node\* node) {

if (node == nullptr) return;

inorderTraversal(node->left);

cout << node->data << " ";

inorderTraversal(node->right);

}

void preorderTraversal(Node\* node) {

if (node == nullptr) return;

cout << node->data << " ";

preorderTraversal(node->left);

preorderTraversal(node->right);

}

void postorderTraversal(Node\* node) {

if (node == nullptr) return;

postorderTraversal(node->left);

postorderTraversal(node->right);

cout << node->data << " ";

}

public:

BinaryTree() : root(nullptr) {}

void insert(int value) {

if (root == nullptr) {

root = new Node(value);

cout << value << " inserted as root.\n";

return;

}

Node\* current = root;

while (true) {

if (value < current->data) {

if (current->left == nullptr) {

current->left = new Node(value);

cout << value << " inserted to the left of " << current->data << "\n";

break;

} else {

current = current->left;

}

} else {

if (current->right == nullptr) {

current->right = new Node(value);

cout << value << " inserted to the right of " << current->data << "\n";

break;

} else {

current = current->right;

}

}

}

}

void inorder() {

cout << "Inorder traversal: ";

inorderTraversal(root);

cout << "\n";

}

void preorder() {

cout << "Preorder traversal: ";

preorderTraversal(root);

cout << "\n";

}

void postorder() {

cout << "Postorder traversal: ";

postorderTraversal(root);

cout << "\n";

}

};

int main() {

BinaryTree tree;

int choice, value;

do {

cout << "\n1. Insert\n2. Inorder Traversal\n3. Preorder Traversal\n4. Postorder Traversal\n5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to insert: ";

cin >> value;

tree.insert(value);

break;

case 2:

tree.inorder();

break;

case 3:

tree.preorder();

break;

case 4:

tree.postorder();

break;

case 5:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid choice. Try again.\n";

}

} while (choice != 5);

return 0;

}