FAHAD M. AL-ALAWI

#include <iostream>

#include <vector>

#include <sstream>

#include <algorithm>

using namespace std;

struct TreeNode {

int value;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) : value(val), left(nullptr), right(nullptr) {}

};

class BinaryTree {

public:

TreeNode\* root;

BinaryTree() : root(nullptr) {}

void insertValue(int val) {

root = insertNode(root, val);

}

void showTree() const {

int treeHeight = calculateHeight(root);

int treeWidth = (1 << treeHeight) - 1;

vector<vector<string>> treeDiagram(treeHeight, vector<string>(treeWidth, " "));

buildTreeRepresentation(root, treeDiagram, 0, 0, treeWidth - 1);

printTreeRepresentation(treeDiagram);

}

private:

TreeNode\* insertNode(TreeNode\* node, int val) {

if (!node) {

return new TreeNode(val);

}

if (val == node->value) {

return node;

}

if (val < node->value) {

node->left = insertNode(node->left, val);

} else {

node->right = insertNode(node->right, val);

}

return node;

}

int calculateHeight(TreeNode\* node) const {

if (!node) {

return 0;

}

return 1 + max(calculateHeight(node->left), calculateHeight(node->right));

}

void buildTreeRepresentation(TreeNode\* node, vector<vector<string>>& diagram, int level, int left, int right) const {

if (!node) return;

int mid = left + (right - left) / 2;

diagram[level][mid] = to\_string(node->value);

buildTreeRepresentation(node->left, diagram, level + 1, left, mid - 1);

buildTreeRepresentation(node->right, diagram, level + 1, mid + 1, right);

}

void printTreeRepresentation(const vector<vector<string>>& diagram) const {

for (const auto& row : diagram) {

for (const auto& cell : row) {

cout << cell;

}

cout << endl;

}

}

};

int main() {

BinaryTree tree;

string input;

int num;

cout << "Enter tree values (e.g., 10 9 11 12 8 7): ";

getline(cin, input);

stringstream ss(input);

while (ss >> num) {

tree.insertValue(num);

}

cout << "\nBinary Tree Diagram:" << endl;

tree.showTree();

return 0;

}

