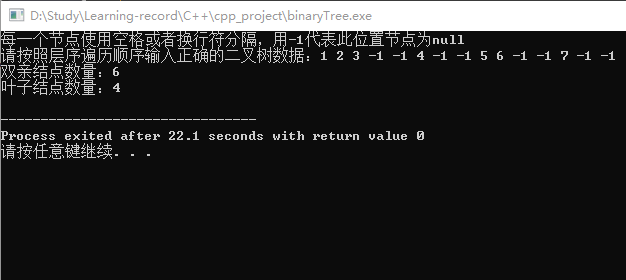
1. 已知一个二叉树（从键盘输入），分别求它的双亲结点个数和叶子结点的个数。



#include <iostream>

using namespace std;

//定义二叉树结构

struct Node {

int val;

Node\* left;

Node\* right;

// 初始化节点的构造函数

Node(int v) : val(v), left(nullptr), right(nullptr) {}

};

// 计算双亲结点数量

int countParentNodes(Node\* root) {

// 如果二叉树为空，则返回0

if (!root) {

return 0;

}

int count = 0;

// 如果二叉树的左子树不为空，则左子树的根节点是一个双亲节点

if (root->left) {

count++;

}

// 如果二叉树的右子树不为空，则右子树的根节点是一个双亲节点

if (root->right) {

count++;

}

// 递归计算左子树和右子树的双亲节点数量，并将它们加到当前节点的双亲节点数量上

return count + countParentNodes(root->left) + countParentNodes(root->right);

}

// 计算叶子结点数量

int countLeafNodes(Node\* root) {

// 如果二叉树为空，则返回0

if (!root) {

return 0;

}

// 如果当前节点没有左子树和右子树，则它是一个叶子节点，返回1

if (!root->left && !root->right) {

return 1;

}

// 递归计算左子树和右子树的叶子节点数量，并将它们加到当前节点的叶子节点数量上

return countLeafNodes(root->left) + countLeafNodes(root->right);

}

// 从键盘读取二叉树

Node\* readBinaryTree() {

int val;

cin >> val;

// 如果当前节点的值为-1，则该节点为空节点，返回nullptr

if (val == -1) {

return nullptr;

}

// 创建一个新的二叉树节点，并递归读取左子树和右子树

Node\* root = new Node(val);

root->left = readBinaryTree();

root->right = readBinaryTree();

return root;

}

int main() {

// 从键盘读取二叉树

cout << "每一个节点使用空格或者换行符分隔，用-1代表此位置节点为null\n请按照层序遍历顺序输入正确的二叉树数据：";

Node\* root = readBinaryTree();

// 计算双亲结点数量和叶子结点数量，并输出结果

int parentNodes = countParentNodes(root);

int leafNodes = countLeafNodes(root);

cout << "双亲结点数量：" << parentNodes << endl;

cout << "叶子结点数量：" << leafNodes << endl;

return 0;

}