#include <iostream>

#include <vector>

using namespace std;

//链表

struct ListNode {

int val;

ListNode \*next;

ListNode(int x) : val(x), next(NULL) {}

};

ListNode\* getList() {

int num = 0;

cout << "链表长度：";

cin >> num;

vector<int> v(num);

for (int i = 0; i < num; i++) {

cin >> v[i];

}

ListNode dummy(-1);

ListNode\* p = &dummy;

for (int i = 0; i < v.size(); i++) {

p->next = new ListNode(v[i]);

p = p->next;

}

p->next = NULL;

return dummy.next;

}

//二叉树

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(NULL), right(NULL) {}

};

//通过前序遍历和后序遍历构造二叉树

TreeNode\* ConstructCore(vector<int> preOrder, vector<int> inOrder,

int startPreorder, int endPreorder,

int startInorder, int endInorder) {

if (startPreorder > endPreorder || startInorder > endInorder) {

return nullptr;

}

TreeNode\* root = new TreeNode(preOrder[startPreorder]);

int divider = 0;

while (divider < endInorder && inOrder[divider] != root->val) {

divider++;

}

int offSet = divider - startInorder - 1;

root->left = ConstructCore(preOrder, inOrder,

startPreorder + 1, startPreorder + 1 + offSet,

startInorder, divider - 1);

root->right = ConstructCore(preOrder, inOrder,

startPreorder + offSet + 2, endPreorder,

divider + 1, endInorder);

return root;

}

1. //通过中序遍历和后序遍历构造二叉树
2. **public** **static** TreeNode5 ConstructCore2(**int**[] postOrder, **int**[] inOrder,
3. **int** startPostorder, **int** endPostorder,
4. **int** startInorder, **int** endInorder){
6. **if** (startPostorder > endPostorder || startInorder > endInorder) {
7. **return** **null**;
8. }
10. TreeNode5 root = **new** TreeNode5(postOrder[endPostorder]);
11. **int** divider = 0;
13. **while**(divider < endInorder && inOrder[divider] != root.val){
14. divider++;
15. }
17. **int** offSet = divider - startInorder - 1;
19. root.left = ConstructCore2(postOrder, inOrder,
20. startPostorder, startPostorder + offSet,
21. startInorder, divider - 1);

24. root.right = ConstructCore2(postOrder, inOrder,
25. startPostorder + offSet + 1, endPostorder - 1,
26. divider + 1, endInorder);
28. **return** root;
29. }

//此方法用的前序、中序构造二叉树

TreeNode\* getBinary\_Tree() {

int num = 0;

cout << "节点数：";

cin >> num;

cout << "前序遍历数据：";

vector<int> preoder(num);

for (int i = 0; i < num; i++) {

cin >> preoder[i];

}

cout << "中序遍历数据：";

vector<int> inoder(num);

for (int i = 0; i < num; i++) {

cin >> inoder[i];

}

TreeNode\* root = ConstructCore(preoder, inoder,0, num-1,0,num-1);

return root;

}

void main() {

TreeNode\* root = getBinary\_Tree();

}