二叉树前序后序中序遍历

https://blog.csdn.net/m0_50617544/article/details/128749340

九是否随机的称呼　于 2023-05-12 11:30:00 发布

```cpp
#include<cstdio>
#include<iostream>
#include<vector>
#include<string>
using namespace std;
struct nod {
    int val = -9;
    nod *l=NULL, *r=NULL;
};
int pre[] = {1, 2, 3, 4, 5, 6};
int in[] = {3, 2, 4, 1, 6, 5};

//不能产生树的情况
// int pre[] = {1, 2, 3, 4, 6, 7, 5};
// int in[] = {2, 3, 1, 7, 4, 5, 6};
void printvec(vector<int> a, string t){
    cout<<t<<"\t";
    for(int i=0; i<a.size();i++){
        cout<<a[i];
        if(i!=(a.size()-1))
            cout<<" ";
    }
    cout<<endl;
}
/*
       1
   2         5
3    4    6
*/

vector<int> postarr;
```

```
void poster(int root, int start, int end) {
    if(start > end) return;
    int k = start;
    while(k <=end && pre[root]!=in[k]) k++;
    if(k > end) {
        printf("//不能产生树的\n"); //来判断是否可以产生树
        return;
    }
    poster(root+1, start, k-1);
    poster(root + k - start + 1, k+1, end);
    postarr.push_back(pre[root]);
}
nod* gentree(nod *root, int preroot, int start, int end) {
    if(start > end) return NULL;
    if(root==NULL) {
        root = new(nod);
        root->val = pre[preroot];
    }
    int k = start;
    while(k <=end && pre[preroot]!=in[k]) k++;
    if(k > end) {
        printf("//不能产生树的\n"); //来判断是否可以产生树
        return NULL;
    }
    root->l = gentree(root->l, preroot+1, start, k-1);
    root->r = gentree(root->r, preroot + k - start + 1, k+1, end);
    return root;
}
void preorder(nod *root) {
    if(root==NULL) return;
    postarr.push_back(root->val);
    preorder(root->l);
    preorder(root->r);
}
void inorder(nod *root) {
    if(root==NULL) return;
    inorder(root->l);
    postarr.push_back(root->val);
    inorder(root->r);
```

```
}
void postorder(nod *root) {
    if(root==NULL) return;
    postorder(root->l);
    postorder(root->r);
    postarr.push_back(root->val);
}
int main() {
    poster(0, 0, 5); //0pre 0inl 5inr
    printvec(postarr, "generate postarr:");

    nod *root = NULL;
    root = gentree(root, 0, 0, 5);
    postarr.clear();
    preorder(root);
    printvec(postarr, "preorder:");

    postarr.clear();
    inorder(root);
    printvec(postarr, "inorder:");

    postarr.clear();
    postorder(root);
    printvec(postarr, "postorder:");
    return 0;
}
```

PAT/binarytree at master · ZouJiu1/PAT (github.com)

https://github.com/ZouJiu1/PAT/tree/master/binarytree