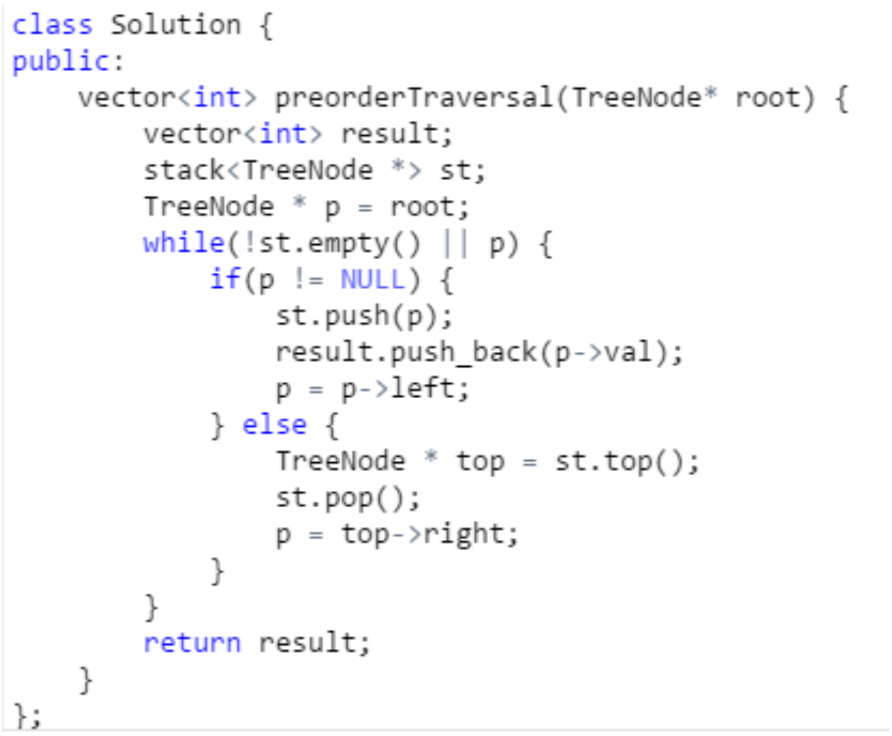
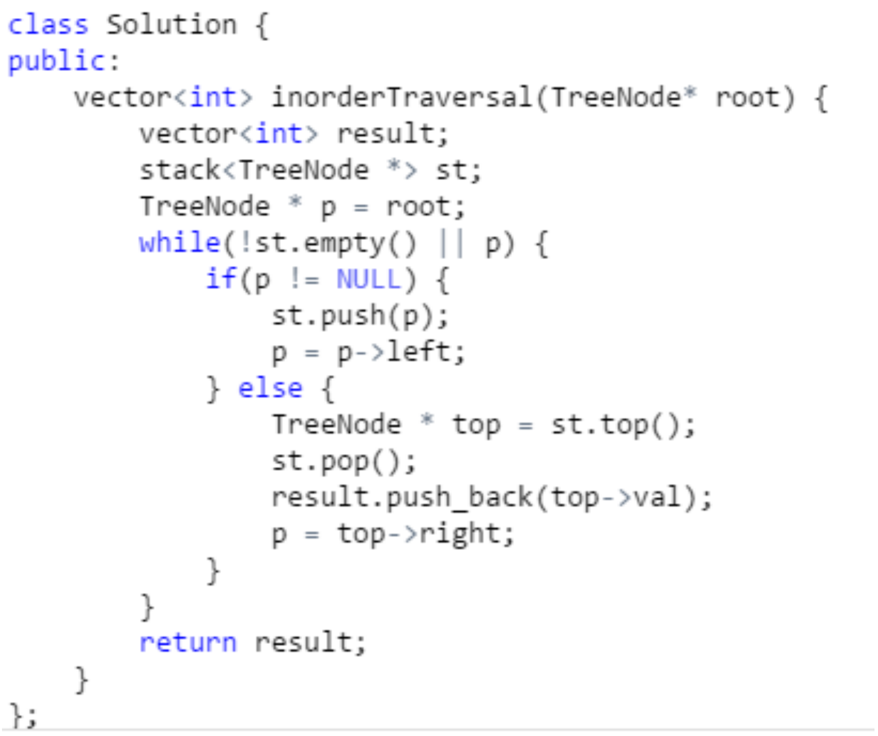
树的遍历总结

PreOrder



```
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        vector<int> result;
        stack<TreeNode *> st;
        TreeNode * p = root;
        while(!st.empty() || p) {
            if(p != NULL) {
                st.push(p);
                result.push_back(p->val);
                p = p->left;
            } else {
                TreeNode * top = st.top();
                st.pop();
                p = top->right;
            }
        }
        return result;
    }
};
```

InOrder



```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int> result;
        stack<TreeNode *> st;
        TreeNode * p = root;
        while(!st.empty() || p) {
            if(p != NULL) {
                st.push(p);
                p = p->left;
            } else {
                TreeNode * top = st.top();
                st.pop();
                result.push_back(top->val);
                p = top->right;
            }
        }
        return result;
    }
};
```

PostOrder

```
class Solution {
public:
    vector<int> postorderTraversal(TreeNode* root) {
        vector<int> result;
        stack<TreeNode *> st;
        TreeNode * p = root;
        TreeNode * pre = NULL;
        while(!st.empty() || p) {
            if(p != NULL) {
                st.push(p);
                p = p->left;
            } else {
                TreeNode * top = st.top();
                if(top->right == NULL || top->right == pre) {
                    st.pop();
                    pre = top;
                    result.push_back(top->val);
                } else {
                    p = top->right;
                }
            }
        }
        return result;
    }
};
```