**二叉树非递归遍历-先序遍历**

一、

·延续我们上节的代码构造情况哈

·那么我们知道，其实递归是调用系统栈来进行操作，那么现在我们不用递归，也就是不利用系统栈来进行，所以我们只能够自己构造栈来模拟

·思想：

1. 我们先序就是从左孩子开始，一直下去，遇到就打印出来，那么我们还用栈的后进先出的特点来先入栈右孩子再入栈左孩子
2. 那么，只要我们的栈不为空的话，那么我们的右孩子就会被压到所有的左孩子访问完才能轮到他
3. 所以，下面我们来coding吧
4. 代码

//先序遍历

void pre\_tree(BiTree\* T) {

if (!T) return; //如果为空就返回

//不然就进行遍历,先进行头节点的访问输出，再遍历递归左孩子，再右孩子

cout << T->data << " ";

pre\_tree(T->left\_child);

pre\_tree(T->right\_child);

}

#include <stack>

//非递归中序遍历

void noRec\_pre\_tree(BiTree\* T) {

stack<BiTree\*> S;

BiTree\* p = T;

S.push(p);

while (!S.empty()) {

//我们访问栈顶的元素

BiTree\* cur\_p = S.top();

S.pop();

std::cout << cur\_p->data << " ";

//如果有右孩子，则有孩子先入栈

if (cur\_p->right\_child) S.push(cur\_p->right\_child);

//左孩子

if (cur\_p->left\_child) S.push(cur\_p->left\_child);

}

}

运行截图

