**二叉树非递归遍历-后序遍历**

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. 其实我们可以延续上节的先序遍历的想法，但是有点不同的就是，我们先入栈左孩子再入栈右孩子，那么我们遍历出来的就是逆后序遍历
2. 那么，我们把输出的结果逆置一下就可以了，是不是很nice
3. 那么输出是不是也可以利用一个栈的先进后出来进行操作就可以了呀
4. 有没有get到了呢
5. 代码

//后序遍历，其实就是在先序遍历的基础上，把输出的函数放在后面而已

void post\_tree(BiTree\* T) {

if (!T) return; //如果为空就返回

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

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

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

}

//非递归后序遍历

#include <stack>

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

stack<BiTree\*> res; //在输出的时候我们不进行输出了，我们改成存进去栈中去

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();

res.push(cur\_p);

//左孩子

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

//如果有右孩子，则有孩子先入栈

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

}

//现在进行打印出来

while (!res.empty()) {

BiTree\* p = res.top();

res.pop();

std::cout << p->data << " ";

}

}

运行截图

