**二叉树的链式存储**

一、

1. 我们可以用结构体进行构造树的链式存储结构
   * 1. 第一就是保存树的数据data
     2. 左孩子：left\_child
     3. 右孩子：right\_child
2. 直接上代码。不懂的可以找别的视频的详解来进行体会一下即可啦
   1. 用的是递归的方法。非递归在后面进行讲解

#include <iostream>

using namespace std;

using T = char; //T 就是代表的就是char

struct BiTree {

T data;

BiTree\* left\_child{ 0 }; //定义左右孩子并且初始化出来

BiTree\* right\_child{ 0 };

};

//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);

}

//中序遍历，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 << " ";

}

//中序遍历，其实就是在先序遍历的基础上，把输出的函数放在中间而已

void ind\_tree(BiTree\* T) {

if (!T) return; //如果为空就返回

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

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

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

}

int main()

{

//那么我们现在构造出一颗简单的数出来吧

BiTree\* T = new BiTree(); T->data = 'A';

T->left\_child = new BiTree(); T->left\_child->data = 'B';

T->right\_child = new BiTree(); T->right\_child->data = 'C';

BiTree\* p = T->left\_child;

p->left\_child = new BiTree(); p->left\_child->data = 'D';

p->right\_child = new BiTree(); p->right\_child->data = 'E';

//现在我们写一下遍历的函数来遍历一下吧

cout << "这是先序遍历：" << endl;

pre\_tree(T); //A B D E C ok，证明我们的先序遍历的代码是正确的呀

cout << endl;

cout << "这是中序遍历：" << endl;

ind\_tree(T);

cout << endl;

cout << "这是后序遍历：" << endl;

post\_tree(T);

}