#include&lt;stdio.h&gt; // EOF(=^Z或F6),NULL

#include&lt;stdlib.h&gt; // srand( ) ,rand( ),exit(n)

#include&lt;string.h&gt;

#include&lt;iostream&gt;

using namespace std;

#define OK 1

#define ERROR 0

#define OVERFLOW -2

typedef int Status;

typedef char TElemType;

typedef void\* QElemType;

//---队列的顺序存储---

#define MAXQSIZE 100 //队列可能达到的最大长度

typedef struct

{

QElemType \*base;//存储空间的基地址

int front; //头指针（下标）

int rear; //尾指针（下标）

}SeQueue;

//---二叉树的二叉链表存储表达---

typedef struct BiTNode // 结点结构

{

TElemType data;//节点数据域

struct BiTNode \*lchild, \*rchild; // 左右孩子指针

} BiTNode, \*BiTree;

//创建二叉树链表（用的先序遍历创建的）

Status CreateBiTree (BiTree &amp;T)

{

char ch;

scanf("%c",&amp;ch);

if(ch=='#') T=NULL;

else

{

T = new BiTNode;

if (!T) exit(OVERFLOW);

T-&gt;data = ch; // 生成根结点并赋值

CreateBiTree(T-&gt;lchild); // 创建左子树

CreateBiTree(T-&gt;rchild); // 创建右子树

}

return OK;

}// CreateBiTre

// 先序遍历二叉树（递归）

void PreOrderTraverse (BiTree &amp;T)

{

if (T)

{

cout&lt;&lt;T-&gt;data; // 访问根结点

PreOrderTraverse(T-&gt;lchild); //先序遍历左子树

PreOrderTraverse(T-&gt;rchild);//先序遍历右子树

}

}

// 中序遍历二叉树（递归）

void InOrderTraverse (BiTree &amp;T)

{

if (T)

{

InOrderTraverse(T-&gt;lchild); //中序遍历左子树

cout&lt;&lt;T-&gt;data; // 访问根结点

InOrderTraverse(T-&gt;rchild);//中序遍历右子树

}

}