**实验题目：**二叉树的简单操作

**实验目的：** 掌握二叉树的定义、性质及存储方式，各种遍历算法。

**实验要求：**采用二叉树链表作为存储结构，完成二叉树的建立，先序、中序和后序，求所有叶子及结点总数和树的高度的操作。

**实验步骤：**

1. 通过输入建立一个ABC###DE###的树
2. 通过栈非递归的中序遍历二叉树，递归的后序遍历二叉树
3. 求节点的个数
4. 求叶子节点的个数
5. 求树的高度

**二叉树如下：**

**A**

**B D**

**C E**

**实验代码：**

#include <stdio.h>

#include <stdlib.h>

#define MaxSize 20

#define Add 5

typedef struct Node{//节点的类型

char data;

struct Node\* Lchild;

struct Node\* Rchild;

}\*ElemType,Elem;

typedef struct stk{//栈的定义

ElemType \*data;

int top;

int maxsize;

}Stack,\*Pstack;

Pstack Init(){//栈的初始化

Pstack s;

s = (Pstack)malloc(sizeof(Stack));

s->maxsize = MaxSize;

s->top = 0;

return s;

}

int IsFull(Pstack s){//判断栈是否已满

if(s->top==s->maxsize)

return 1;

else

return 0;

}

int IsEmpty(Pstack s){//判断栈是否为空

if(s->top==0)

return 1;

else

return 0;

}

void Push(Pstack s,ElemType e){//将元素入栈

if(IsFull(s)){

Pstack tmp;

tmp = realloc(s->data,s->maxsize+Add\*sizeof(ElemType));

s = tmp;

}

s->data[s->top++] = e;

}

ElemType Pop(Pstack s){//将元素出栈

if(IsEmpty(s)){

printf("栈已空！\n");

}

else{

return s->data[--s->top];

}

}

ElemType CreateTree(ElemType T){//建树

char c;

scanf("%c",&c);

T = (ElemType)malloc(sizeof(Elem));

if(c=='#'){

T = NULL;

}

else{

T->data = c;

T->Lchild = CreateTree(T->Lchild);

T->Rchild = CreateTree(T->Rchild);

}

return T;

}

void InOrder(ElemType T,Pstack s){//非递归的中序遍历

//ABC###DE###

while(T!=NULL||!IsEmpty(s)){

if(T!=NULL){

Push(s,T);

T = T->Lchild;

}

else{

T = Pop(s);

printf("%c ",T->data);

T = T->Rchild;

}

}

}

void PreOrder(ElemType T){//递归的后续遍历

if(T!=NULL){

PreOrder(T->Lchild);

PreOrder(T->Rchild);

printf("%c ",T->data);

}

}

int NumOfNode(ElemType T){//求二叉树的节点个数

if(T==NULL)

return 0;

return NumOfNode(T->Lchild)+NumOfNode(T->Rchild)+1;

}

int NumOfLeave(ElemType T){//求二叉树的叶子节点个数

if(T==NULL)

return 0;

if(T->Lchild==NULL&&T->Rchild==NULL)

return 1;

return NumOfLeave(T->Lchild)+NumOfLeave(T->Rchild);

}

int DepthOfTree(ElemType T){//求二叉树的高度

if(T==NULL)

return 0;

return DepthOfTree(T->Lchild)>DepthOfTree(T->Rchild) ? DepthOfTree(T->Lchild)+1:DepthOfTree(T->Rchild)+1;

}

int main()

{

printf("请构建二叉树:\n");

ElemType T;

T = NULL;

Pstack S = Init();

T = CreateTree(T);

printf("非递归中序遍历结果为:\n");

InOrder(T,S);

printf("\n递归的后续遍历结果为:\n");

PreOrder(T);

printf("\n节点个数为:\n");

printf("%d",NumOfNode(T));

printf("\n叶子节点的个数为:\n");

printf("%d",NumOfLeave(T));

printf("\n树的深度为:\n");

printf("%d",DepthOfTree(T));

return 0;

}

**实验结果：**

请构建二叉树:

ABC###DE###

非递归中序遍历结果为:

C B A E D

递归的后续遍历结果为:

C B E D A

节点个数为:

5

叶子节点的个数为:

2

树的深度为:

3

