完成日期：2016.11.3

* **问题描述：**

1）用先序递归过程建立二叉树 (存储结构：二叉链表)

输入数据按先序遍历所得序列输入，当某结点左子树或右子树为空时，输入‘\*’号，如输入abc\*\*d\*\*e\*\*得到的二叉树为：（图略）

（选做：由二叉树的先序序列和中序序列建立一棵二叉树。）

2）编写递归算法，计算二叉树中叶子结点的数目。

3）按凹入表方式输出该二叉树。

* **算法思想：**

要求用二叉链表做，所以有左右孩子指针。创建二叉树时，用递归，分别递归左右孩子，然后读取\*时为空。叶子节点是左右孩子都为空时，所以依旧可以用递归，判断每个节点的左右孩子，若不为空，则判断其左右孩子的左右孩子。凹入表输出相当于一个遍历，也可以用递归，只不过深度不同，前面空格不同，所以可以根据深度输出空格。

* **设计描述：**

1）typedef struct BiTNode{

char data;

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

}BiTNode,\*BiTree;

2）有创建二叉树、计算叶子结点个数、输出二叉树三个函数。Main函数依次调用。

3）算法：

Status CreateBiTree(BiTree T)

{ 读取字符

if(字符为'\*')

T为空；

else

{

T=(BiTree)malloc(sizeof(BiTNode));

T的data=ch;

CreateBiTree(T->lchild);

CreateBiTree(T->rchild);

}

}

void leaf(BiTree Ta)

{ if(Ta不为空)

{ if(Ta的左右孩子都为空)

count++;//叶子数加一

leaf(Ta->lchild);

leaf(Ta->rchild);

}

}

void output(BiTree Tb,int depth)//凹入表输出

{

if(Tb不为空)

{ depth++;

for(i=1;i<depth;i++)

打印空格；

打印数据；

if(Tb左孩子不空)

output(Tb->lchild,depth);

if(Tb右孩子不空)

output(Tb->rchild,depth);

}

}

main()

{

BiTree Tr;

创建二叉树；

求叶子节点个数

凹入表输出；

}

* **源程序**

(实验03.c)

#include<stdio.h>

#include<stdlib.h>

#define OVERFLOW -1

typedef struct BiTNode{

char data;

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

}BiTNode,\*BiTree;

int count=0;

BiTree CreateBiTree(BiTree T)//创建二叉树

{ char ch;

scanf("%c",&ch);

if(ch=='\*')

{T=NULL;

}

else

{

T=(BiTree)malloc(sizeof(BiTNode)); //分配内存

T->data=ch;

//T->lchild=NULL;

//T->rchild=NULL;

T->lchild=CreateBiTree(T->lchild);// 要把T->lchild带出来

T->rchild=CreateBiTree(T->rchild);

}

return T;

}

/\*

int Leaves(BiTree T)

{ if(T==NULL)

return 0;

else if(T->lchild==NULL&&T->rchild==NULL)

return 1;

else

{ return Leaves(T->lchild)+Leaves(T->rchild);

}

}\*/

void leaf(BiTree Ta)

{ if(Ta!=NULL)

{ if((Ta->lchild==NULL)&&(Ta->rchild==NULL))//左右孩子都为空

count++;//叶子节点数加1

leaf(Ta->lchild);//继续求左孩子的

leaf(Ta->rchild);

}

}

void output(BiTree Tb,int depth)//凹入表输出

{ int i;

if(Tb!=NULL)

{

depth++;//深度加一

for(i=1;i<depth;i++)

printf(" ");//根据深度输出空格

printf("%c\n",Tb->data);

if(Tb->lchild!=NULL)

output(Tb->lchild,depth);

if(Tb->rchild!=NULL)

output(Tb->rchild,depth);

}

}

main()

{ printf("Please input the data:\n");

int i;

BiTree Tr;

// Tr=(BiTree)malloc(sizeof(BiTNode));

Tr=CreateBiTree(Tr);

leaf(Tr);//叶子节点个数

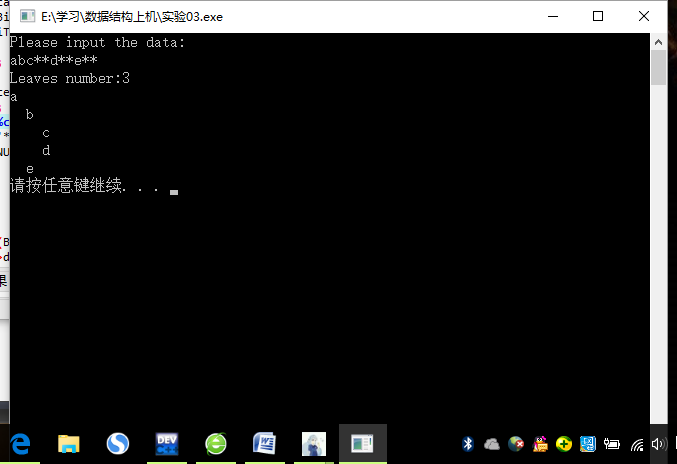
printf("Leaves number:%d\n",count);

output(Tr,0);

system("pause");

return 0;

}

* **测试结果：**
* **用户使用说明：**

运行环境Dev C++,按要求输入数据后回车。

* **心得体会：**

这个程序看似简单，但是因为一些错误花费了不少时间。创建二叉树时，递归中左孩子右孩子不读取返回指针的话，T的左孩子右孩子无法在之后的函数中读取。所以总是停止运行。试验了多遍之后才改过来。

还有数叶子节点的递归算法很多，也从网上看到不用全局变量的，都比较好理解。

凹入表输出用的有点类似先序遍历，但不同的是要计算深度输出空格。