姓名： 毛翊尧 班级： A1611 学号： 31 号

**实验三 二叉树**

一、实验目的

1、掌握二叉树的基本特性

2、掌握二叉树的先序、中序、后序的递归遍历算法

3、理解二叉树的先序、中序、后序的非递归遍历算法

4、通过求二叉树的深度、叶子结点数和层序遍历等算法，理解二叉树的基本特性

二、实验预习

说明以下概念

1、二叉树：是每个结点最多有两个子树的树结构。

2、递归遍历：遵从递归次序，遍访二叉树中的所有结点。

3、非递归遍历：遵从非递归次序，遍访二叉树中的所有结点。

4、层序遍历：从二叉树的根结点开始，自上向下、自左向右分层依次访问树中的各个结点。

三、实验内容和要求

1、阅读并运行下面程序，根据输入写出运行结果，并画出二叉树的形态。

#include<stdio.h>

#include<malloc.h>

#define MAX 20

typedef struct BTNode{ /\*节点结构声明\*/

char data ; /\*节点数据\*/

struct BTNode \*lchild;

struct BTNode \*rchild ; /\*指针\*/

}\*BiTree;

void createBiTree(BiTree \*t){ /\* 先序遍历创建二叉树\*/

char s;

BiTree q;

printf("\nplease input data:(exit for #)");

s=getche();

if(s=='#'){\*t=NULL; return;}

q=(BiTree)malloc(sizeof(struct BTNode));

if(q==NULL){printf("Memory alloc failure!"); exit(0);}

q->data=s;

\*t=q;

createBiTree(&q->lchild); /\*递归建立左子树\*/

createBiTree(&q->rchild); /\*递归建立右子树\*/

}

void PreOrder(BiTree p){ /\* 先序遍历二叉树\*/

if ( p!= NULL ) {

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

PreOrder( p->lchild ) ;

PreOrder( p->rchild) ;

}

}

void InOrder(BiTree p){ /\* 中序遍历二叉树\*/

if( p!= NULL ) {

InOrder( p->lchild ) ;

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

InOrder( p->rchild) ;

}

}

void PostOrder(BiTree p){ /\* 后序遍历二叉树\*/

if ( p!= NULL ) {

PostOrder( p->lchild ) ;

PostOrder( p->rchild) ;

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

}

}

void Preorder\_n(BiTree p){ /\*先序遍历的非递归算法\*/

BiTree stack[MAX],q;

int top=0,i;

for(i=0;i<MAX;i++) stack[i]=NULL;/\*初始化栈\*/

q=p;

while(q!=NULL){

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

if(q->rchild!=NULL) stack[top++]=q->rchild;

if(q->lchild!=NULL) q=q->lchild;

else

if(top>0) q=stack[--top];

else q=NULL;

}

}

void release(BiTree t){ /\*释放二叉树空间\*/

if(t!=NULL){

release(t->lchild);

release(t->rchild);

free(t);

}

}

int main(){

BiTree t=NULL;

createBiTree(&t);

printf("\n\nPreOrder the tree is:");

PreOrder(t);

printf("\n\nInOrder the tree is:");

InOrder(t);

printf("\n\nPostOrder the tree is:");

PostOrder(t);

printf("\n\n先序遍历序列（非递归）:");

Preorder\_n(t);

release(t);

return 0;

}

运行程序

输入：

ABC##DE#G##F###

运行结果：

2、在上题中补充求二叉树中求结点总数算法（提示：可在某种遍历过程中统计遍历的结点数），并在主函数中补充相应的调用验证正确性。

算法代码：

3、在上题中补充求二叉树中求叶子结点总数算法（提示：可在某种遍历过程中统计遍历的叶子结点数），并在主函数中补充相应的调用验证正确性。

算法代码：

4、在上题中补充求二叉树深度算法，并在主函数中补充相应的调用验证正确性。

算法代码：

四、实验小结