#include "stdio.h"

#include "stdlib.h"

//二叉树链式储存的结构体

typedef struct node{

int data;

struct node \*lchild,\*rchild;

}bintnode;

//全局变量

bintnode \*root; /\*指向二叉树跟结点的指针\*/

//顺序栈的头文件

typedef struct stack

{

bintnode \*data[100];

int tag[100]; /\*为栈中的每个元素设置的标记\*/

int top; /\*栈顶指针\*/

} seqstack;

//二叉树的递归算法

//前序遍历

void preorder(bintnode \*t)

{

if(t)

{

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

preorder(t->lchild);

preorder(t->rchild);

}

}

//二叉树的非递归实现

void push(seqstack \*s,bintnode \*t) /\*进栈\*/

{

s->data[s->top] = t;

s->top ++;

}

bintnode \*pop(seqstack \*s)

{

if(s->top != 0)

{

s->top --;

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

}

else

return NULL;

}

//根据前序遍历创建一棵二叉树

bintnode \*createbintree()

{

char ch;

bintnode \*t;

if((ch = getchar())== '#')

t = NULL;

else

{

t = (bintnode \*)malloc(sizeof(bintnode));

t->data = ch;

t->lchild = createbintree();

t->rchild = createbintree();

}

return t;

}

//递归统计二叉树中的叶子节点个数

int numofnode(bintnode \*t)

{

int count;

if(t == NULL)

return 0;

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

return 1;

else

return (numofnode(t->lchild) + numofnode(t->rchild));

return count;

}

//非递归统计二叉树中的叶子节点个数

int numofleaf(bintnode \*t)

{

int count = 0;

seqstack s;

s.top = 0;

while((t) || (s.top != 0))

{

if(t)

{

if(t->lchild == NULL && t->rchild == NULL)

count ++;

push(&s,t);

t = t->lchild;

}

else

{

t = pop(&s);

t = t->rchild;

}

}

return count;

}

void inorderlastnode(bintnode \*t)

{

seqstack s;

s.top = 0;

bintnode \*p = NULL;

while((t != NULL) || (s.top != 0))

{

if(t)

{

p = t;

push(&s,t);

t = t->lchild;

}

else

{

t = pop(&s);

t = t->rchild;

}

}

printf("\n中序遍历的最后一个结点为：%c\n",p->data);

}

int main()

{

printf("前序遍历方式创建一颗二叉树：\n");

root = createbintree();

printf("前序遍历递归输出这颗二叉树：\n");

preorder(root);

printf("\n递归该二叉树当中的叶子结点个数为：%d\n",numofnode(root));

printf("\n");

printf("\n非递归该二叉树当中的叶子结点个数为：%d\n",numofleaf(root));

//取得最后一个结点

inorderlastnode(root);

}

//cba###de#gf##h###

