13叶节点数

#include"stdio.h"

#include"string.h"

#include"stdlib.h"

typedef char datatype;

typedef struct bintnode{

datatype data;

struct bintnode \*lchild,\*rchild;

}node;

typedef struct stack

{

node \*data[100];

int tag[100];

int top;

} seqstack;

//前序递归创建

node \*create()

{

char c;

node \*t;

if((c=getchar())=='#')

t=NULL;

else

{

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

t->data=c;

t->lchild=create();

t->rchild=create();

}

return t;

}

//进栈

void push(seqstack \*s,node \*t )

{

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

}

//出栈

node \*pop(seqstack \*s)

{

if (s->top!=0)

{

s->top--;

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

}

else

return NULL;

}

//前序递归

int preorder1(node \*t)

{static int c=0;

if(t)

{

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

c++;

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

else

{

preorder1(t->lchild);

preorder1(t->rchild);

}

}

return c;

}

//前序遍历的非递归实现

void preorder(node \*t)

{

int a=0;

printf("非递归前序遍历");

seqstack s;

s.top=0;

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

{

if(t)

{

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

a++;

push(&s,t);

t=t->lchild;

}

else

{

t=pop(&s);

t=t->rchild;

}

}

printf("%d",a);

}

int main()

{

node \*t=NULL;

int c;

seqstack \*s=0;

t=create();

printf("前序递归：") ;

c=preorder1(t);

printf("%d",c);

printf("\n");

preorder(t);

printf("\n");

}

14中序遍历下的最后一个节点

#include"stdio.h"

#include"string.h"

#include"stdlib.h"

typedef char datatype;

typedef struct bintnode{

datatype data;

struct bintnode \*lchild,\*rchild;

}node;

typedef struct stack

{

node \*data[100];

int tag[100];

int top;

} seqstack;

//前序递归创建

node \*create()

{

char c;

node \*t;

if((c=getchar())=='#')

t=NULL;

else

{

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

t->data=c;

t->lchild=create();

t->rchild=create();

}

return t;

}

//进栈

void push(seqstack \*s,node \*t )

{

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

}

//出栈

node \*pop(seqstack \*s)

{

if (s->top!=0)

{

s->top--;

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

}

else

return NULL;

}

//前序递归

int preorder1(node \*t)

{static int c=0;

if(t)

{

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

c++;

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

else

{

preorder1(t->lchild);

preorder1(t->rchild);

}

}

return c;

}

//前序遍历的非递归实现

void preorder(node \*t)

{

int a=0;

printf("非递归前序遍历");

seqstack s;

s.top=0;

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

{

if(t)

{

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

a++;

push(&s,t);

t=t->lchild;

}

else

{

t=pop(&s);

t=t->rchild;

}

}

printf("%d",a);

}

int main()

{

node \*t=NULL;

int c;

seqstack \*s=0;

t=create();

printf("前序递归：") ;

c=preorder1(t);

printf("%d",c);

printf("\n");

preorder(t);

printf("\n");

}