**题目8：二叉树的遍历**

二叉树以lson-rson链接方式存储，以菜单方式设计并完成功能任务：建立并存储树、输出前序遍历结果、输出中序遍历结果、输出后序遍历结果、交换左右子树、统计高度，其中对于中序、后序的遍历运算要求采用非递归方式。

**代码：**

#include<stdio.h>

#include<stdlib.h>

#include <string.h>

#define M 100

typedef struct node//定义二叉树结点

{

char data;

struct node \*lchild,\*rchild;

} BTNode;

BTNode \*CreatBTree(char \*str)

{

char ch;

int top=-1,k,i,j=0;

BTNode \*T;

BTNode \*Stack[M],\*p=NULL;

for(i=0;i<M;i++)

Stack[i]=(BTNode \*)malloc(sizeof(BTNode));

T=(BTNode \*)malloc(sizeof(BTNode));

T=NULL;

ch=str[j];

while(ch!='\0')

{

switch(ch)

{

case '(':top++;Stack[top]=p;k=1;break;

//k是1，接下来的一个是左子树

case ')':top--;break;

case ',':k=2;break;

//k是2，接下来的一个是右子树

default://如果不是符号，说明是数据

{

p=(BTNode \*)malloc(sizeof(BTNode));

p->data=ch;

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

p->lchild=p->rchild=NULL;

if(T==NULL) //根结点

T=p;

else

{

switch(k)

{

case 1:Stack[top]->lchild=p;break;

case 2:Stack[top]->rchild=p;break;

}

}

}

}

j++;

ch=str[j]; //ch访问下一个字符串

}

return T;

}

void PreOrder(BTNode \*b)//递归先序遍历二叉树函数

{

if(b!=NULL)

{

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

PreOrder(b->lchild);

PreOrder(b->rchild);

}

}

void InOrder(BTNode \*b)//非递归中序遍历二叉树

{

BTNode \*stack[M],\*p;

int top=-1;

if(b!=NULL)

{

p=b;

while(top>-1||p!=NULL)

{

while(p!=NULL)//扫描p的所有左结点并入栈

{

top++;

stack[top]=p;

p=p->lchild;

}

if(top>-1)

{

p=stack[top];//出栈访问节点

top--;

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

p=p->rchild;//扫描p的右结点

}

}

printf("\n");

}

}

void PostOrder(BTNode \*b)//非递归后序遍历二叉树函数

{

BTNode \*stack[M],\*p;

int sign,top=-1;

if(b!=NULL)

{

do

{

while(b!=NULL)//b所有左结点入栈

{

top++;

stack[top]=b;

b=b->lchild;

}

p=NULL;//p指向栈顶前一个已访问节点

sign=1;//置b为已访问

while(top!=-1&&sign)

{

b=stack[top];//取出栈顶结点

if(b->rchild==p)//右孩子不存在或右孩子已访问则访问b

{

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

top--;

p=b;//p指向被访问的结点

}

else

{

b=b->rchild;//b指向右孩子结点

sign=0;//置未访问标记

}

}

}while(top!=-1);

printf("\n");

}

}

void change(BTNode \*b)//左右子树交换（递归）

{

BTNode \*r;

r=(BTNode \*)malloc(sizeof(BTNode));

int f1=0,f2=0;

if(b==0)return;//树为空时，跳出

if(b->lchild)

{

change(b->lchild);

r->lchild=b->lchild;

f1++;//有左子树，符号位不为空

}

if(b->rchild)

{

change(b->rchild);

r->rchild=b->rchild;

f2++;//有右子树，符号位不为空

}

if(f1==0)r->lchild=NULL;//否则，给中间变量赋空值

if(f2==0)r->rchild=NULL;

if(f1||f2)

{

b->rchild=r->lchild;//左右子树交换

b->lchild=r->rchild;

}

}

int max(int m,int n)

{

if(m>n)

return m;

else

return n;

}

int count(BTNode \*b)//计算树高（递归）

{

if(b==NULL)

return 0;

else

return(1+max(count(b->lchild),count(b->rchild)));

}

int main()

{

BTNode \*root=NULL;

char s[M];

printf("\*\*\*\*\*\*\*\*\*\*二叉树的遍历\*\*\*\*\*\*\*\*\*\*\n\n");

//printf("请按先序递归创建二叉树（结束符#）:\n");

printf("请用括号法输入一个二叉树:");

scanf("%s",s);

root=CreatBTree(s);

printf("\n递归先序遍历结果：\n");

PreOrder(root);

printf("\n非递归中序遍历结果：\n");

InOrder(root);

printf("\n非递归后序遍历结果：\n");

PostOrder(root);

printf("\n树高：%d\n",count(root));

printf("\n左右子树交换位置：");

change(root);

printf("\n递归先序遍历结果：\n");

PreOrder(root);

printf("\n非递归中序遍历结果：\n");

InOrder(root);

printf("\n非递归后序遍历结果：\n");

PostOrder(root);

return 0;

}

运行结果：

