数据结构实验报告

院系：信息工程学院

专业：软件工程二班

学号：20122480212

姓名：霍静静

二叉树递归遍历的实验报告

**一、实验目的**：了解二叉树存储结构，熟练掌握二叉树的先中后序的递归遍历；

**二、实验要求**：

1）定义二叉树结构

2）熟练掌握二叉树的先中后序递归遍历；

**三、实验内容：**

二叉树的先中后序递归遍历以及二叉树所有节点左右子树的互相交换，求树深度，判断二叉树是否相似等函数；

**四、算法思想：**

二叉树的结构特点是每个节点至多只有两棵子树，并且有左右之分，其次序不能任意颠倒；

**五、主要函数的实现：**

**//----常量头文件---**

**#ifndef \_CONSTANT\_H**

**#define \_CONSTANT\_H**

**typedef int ElemType;**

**typedef int Status;**

**#define TRUE 1**

**#define FALSE 0**

**#define OK 1**

**#define ERROR 0**

**#define INFEASIBLE -1**

**#define OVERFLOW -2**

**#endif**

//­­­­­---------二叉树递归遍历头文件----

#ifndef \_BITREE\_H

#define \_BITREE\_H

#include"constant.h"

#include<stdio.h>

typedef struct BiTNode{

TElemType data;

struct BiTNode \*lchild,\*rchild;

}BiTNode,\*BiTree;

Status CreateBiTree(BiTree \*T)

{

char ch;

scanf("%c",&ch);

if( ch==' ' )

(\*T )=NULL;

else {

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

(\*T)->data = ch;

CreateBiTree((&(\*T)->lchild));

CreateBiTree((&(\*T)->rchild));

}

return OK;

}

Status PrintElement(TElemType e){

printf("%c ",e);

return OK;

}

Status PreOrderTraverse(BiTree T,Status(\*Visit)(TElemType e)){

fflush(stdin);

if(T){

if(PrintElement(T->data)){

if(PreOrderTraverse(T->lchild,Visit)){

if(PreOrderTraverse(T->rchild,Visit))

return OK;

}

}else return ERROR;

}else return OK;

}

Status InOrderTraverse(BiTree T,Status(\*Visit)(TElemType e)){

fflush(stdin);

if(T){

if(InOrderTraverse(T->lchild,Visit)){

if(PrintElement(T->data)){

if(InOrderTraverse(T->rchild,Visit))

return OK;

}

}else return ERROR;

}else return OK;

}

Status PostOrderTraverse(BiTree T,Status(\*Visit)(TElemType e)){

fflush(stdin);

if(T){

if(PostOrderTraverse(T->lchild,Visit)){

if(PostOrderTraverse(T->rchild,Visit)){

if(PrintElement(T->data))

return OK;

}

}else return ERROR;

}else return OK;

}

int Deapth(BiTree bt)

{

//-------------------------------------

// TODO (#1#):计算二叉树的深度

int ldeapth,rdeapth;

if(bt==0)

return 0;

else

{

ldeapth=Deapth(bt->lchild);

rdeapth=Deapth(bt->rchild);

return(ldeapth>rdeapth? ldeapth:rdeapth)+1;

}

return 0;

//-------------------------------------

}

#endif

//————主函数————

#include <stdio.h>

#include <stdlib.h>

#include "constant.h"

#include "BiTree.h"

int LeafCount(BiTree bt)

{

//-------------------------------------

// TODO (#1#):计算二叉树中叶子结点的个数

int lleaf,rleaf;

if(bt->lchild==0 && bt->rchild==0)

return 1;

else

{

lleaf=LeafCount(bt->lchild);

rleaf=LeafCount(bt->rchild);

return lleaf+rleaf;

}

return 0;

//-------------------------------------

}

int BiTree\_Sim(BiTree b1,BiTree b2)/\*判断两个树是否相似\*/

{

if(b1==NULL&&b2==NULL)

return 1; /\*两个为空子树、相似\*/

else if(b1==NULL||b2==NULL)

return 0; /\*两个中某一个为空子树、不相似\*/

else

{

return(BiTree\_Sim(b1->lchild,b2->lchild)&&BiTree\_Sim(b1->rchild,b2->rchild));

}

}

void BiTree\_Revolute(BiTree T){

BiTree b;

b=T->lchild;T->lchild=T->rchild;T->rchild=b;

if(T->lchild) BiTree\_Revolute(T->lchild);

if(T->rchild) BiTree\_Revolute(T->rchild);

}

int Get\_Sub\_Depth(BiTree T,int x){

if(T->data==x){

printf("%d\n",Deapth(T));

return ;

}

else{

if(T->lchild) Get\_Sub\_Depth(T->lchild,x);

if(T->rchild) Get\_Sub\_Depth(T->rchild,x);

}

}

int main(int argc, char \*argv[])

{

BiTree T=0,T1=0;

char c,s;

int t;

printf("Creat BinTree;\nInput preorder:\n");

fflush(stdin);

if( CreateBiTree(&T)==ERROR ) {

printf("ERROR: call CreateBiTree\n");

system("pause");

exit(1);

}

printf("\n二叉树的深度: %d\n", Deapth(T));

printf("\n二叉树中叶子结点的个数: %d\n", LeafCount(T));

printf("Print BinTree Preorder:\n");

PreOrderTraverse(T,PrintElement);

printf("\n");

printf("Print BinTree Inorder:\n");

InOrderTraverse(T,PrintElement);

printf("\n");

printf("Print BinTree Postorder:\n");

PostOrderTraverse(T,PrintElement);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Creat Another BinTree;\nInput preorder:\n");

fflush(stdin);

if( CreateBiTree(&T1)==ERROR ) {

printf("ERROR: call CreateBiTree\n");

system("pause");

exit(1);

}

printf("\n二叉树的深度: %d\n", Deapth(T1));

printf("\n二叉树中叶子结点的个数: %d\n", LeafCount(T1));

printf("Print BinTree Preorder:\n");

PreOrderTraverse(T1,PrintElement);

printf("\n");

printf("Print BinTree Inorder:\n");

InOrderTraverse(T1,PrintElement);

printf("\n");

printf("Print BinTree Postorder:\n");

PostOrderTraverse(T1,PrintElement);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

t=BiTree\_Sim(T,T1);

if(t)

printf("T和T1相似！\n\n");

else

printf("T和T1不相似！\n\n");

printf("交换第一个二叉树的左右子树得到新的二叉树的先序遍历：\n");

BiTree\_Revolute(T);

printf("Print BinTree Preorder:\n");

PreOrderTraverse(T,PrintElement);

printf("\n");

printf("求第一个二叉树中以元素值为s的节点为根的子树的深度：\n请输入s：");

scanf("%c",&s);

printf("深度为：");

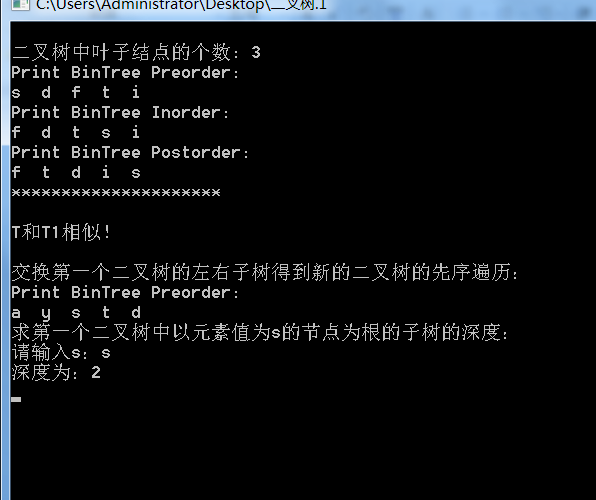
Get\_Sub\_Depth(T,s);

system("PAUSE");

return 0;

}

**六、测试结果：**

****

**七、实验总结：**

**1.逻辑不够清晰，如果再多考虑一下互交就更好了；**

**2.对常量的使用还是不够熟练；**