# 实验6 树子系统的实现

### 实验目的

1. 掌握二叉树的特点及其存储方式。
2. 掌握二叉树的创建和显示方法。
3. 复习二叉树遍历的概念，掌握二叉树遍历的基本方法。
4. 掌握求二叉树的叶结点数、总结点数的基本算法。

### 实验内容

1. 按屏幕提示用前序方法建立一棵二叉树，并能按凹入法显示二叉树的结构。
2. 编写前序遍历、中序遍历、后序遍历、层次遍历程序。
3. 编写求二叉树的叶结点数、总结点数程序。
4. 设计一个选择式菜单，以菜单方式选择下列操作程序。

### 3、设计思路提示：（编程基础好的同学按照自己的习惯来）

1. 先分块读懂参考程序；
2. 然后逐个输入验证该程序，记录bug；
3. 自行设计测试用例，注意极端情况，检验程序的健壮性。
4. 自行改进该参考程序。可以使用自己熟悉的编程语言。

### 参考程序（比较优劣）

### 程序1

#include<stdio.h>

#include<stdlib.h>

#define MAXLEN 100

typedef struct BT{

char data;

struct BT \*lchild,\*rchild;

}BT;

int count=0; //全局变量。。。

//1建二叉树

BT \*Createtree()

{

BT \*T;

char x;

scanf("%c",&x);

getchar(); //处理回车键

if(x=='0')

T=NULL;

else{

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

T->data=x;

printf("请输入%c节点的左子结点为：",T->data); //最好有提示，不然你会在建树.......

T->lchild=Createtree();

printf("请输入%c节点的右子结点为：",T->data);

T->rchild=Createtree();

}

return T;

}

//2先序遍历

void Preorder(BT \*T)

{

if(T==NULL) return;

{

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

Preorder(T->lchild);

Preorder(T->rchild);

}

}

//3中序遍历

void Inorder(BT \*T)

{

if(T==NULL) return;

{

Inorder(T->lchild);

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

Inorder(T->rchild);

}

}

//4后序遍历

void Postorder(BT \*T)

{

if(T==NULL) return;

{

Postorder(T->lchild);

Postorder(T->rchild);

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

}

}

//5层次遍历

void Levelorder(BT \*T)

{

int i,j;

BT \*q[100],\*p; //定义一个指针数组

p=T;

if(p!=NULL){

i=1;

q[i]=p;

j=2;

}

while(i!=j){

p=q[i];

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

if(p->lchild!=NULL){

q[j]=p->lchild;

j++;

}

if(p->rchild!=NULL){

q[j]=p->rchild;

j++;

}

i++;

}

}

//6求叶子数

int Leafnum(BT \*T)

{

if(!T)

return 0;

else{

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

count++;

Leafnum(T->lchild);

Leafnum(T->rchild);

return count;

}

}

int Nodenum(BT \*T)

{

if(!T)

return 0;

else{

count++;

Nodenum(T->lchild);

Nodenum(T->rchild);

return count;

}

}

//8求树深度

int TreeDepth(BT \*T)

{

int ldep,rdep,dep;

if(T==NULL)

return 0;

else{

ldep=TreeDepth(T->lchild);

rdep=TreeDepth(T->rchild);

dep=ldep>rdep?ldep:rdep;

return dep+1;

}

}

void ShowTree(BT \*T)

{

BT \*stack[MAXLEN],\*p;

int level[MAXLEN][2],top,n,i,width=4;

if(T!=NULL)

{

top=1;

stack[top]=T;

level[top][0]=width;

while(top>0)

{

p=stack[top];

n=level[top][0];

for(i=1;i<=n;i++)

printf(" ");

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

for(i=n+1;i<50;i+=2)

printf("▅");

printf("\n\t\t");

top--;

if(p->rchild!=NULL)

{

top++;

stack[top]=p->rchild;

level[top][0]=n+width;

level[top][1]=2;

}

if(p->lchild!=NULL)

{

top++;

stack[top]=p->lchild;

level[top][0]=n+width;

level[top][1]=1;

}

}

}

}

int main()

{

BT \*T;

int choice;

int j=1;

while(j){

printf(" 二叉树子系统 \n");

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

printf(" 1-------建二叉树 \n");

printf(" 2-------先序遍历 \n");

printf(" 3-------中序遍历 \n");

printf(" 4-------后序遍历 \n");

printf(" 5-------层次遍历 \n");

printf(" 6-------求叶子数 \n");

printf(" 7-------求节点数 \n");

printf(" 8-------求树深度 \n");

printf(" 9-------凹入显示 \n");

printf(" 0-------返 回 \n");

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

printf(" 请输出想要输入的数字: ");

scanf("%d",&choice);

getchar();

switch(choice){

case 1:

printf("请按先序序列输入二叉树的节点。说明：输入节点（‘0’表示后继节点为空）后按回车\n"); //提示不可少

printf("请输入根节点：");

T=Createtree();

printf("二叉树成功建立!\n");

break;

case 2:

printf("该二叉树的先序遍历序列为：");

Preorder(T);

printf("\n");

break;

case 3:

printf("该二叉树的中序遍历序列为：");

Inorder(T);

printf("\n");

break;

case 4:

printf("该二叉树的后序遍历序列为：");

Postorder(T);

printf("\n");

break;

case 5:

printf("该二叉树的层次遍历序列为：");

Levelorder(T);

printf("\n");

break;

case 6:

count=0;

printf("该二叉树有%d个叶子\n",Leafnum(T));

break;

case 7:

count=0;

printf("该二叉树总共有%d个节点\n",Nodenum(T));

break;

case 8:

printf("该二叉树的深度是%d：\n",TreeDepth(T));

break;

case 9:

printf("该二叉树的的凹入表示法为: \n");

ShowTree(T);

break;

case 0:j=0;

break;

}

}

return 0;

}

### 程序2

#include<stdio.h>

#include<stdlib.h>

#define datatype char

#define M 50

#define MAXLEN 50

int num=0, counter1=0,counter2=0;

typedef struct node

{

datatype data;

int lt,rt;

struct node \*lchild,\*rchild;

}BT;

BT\* createtree()

{

BT \*bt;

datatype x;

scanf("\n%c",&x);

if(x=='0')

{

bt=NULL;

}

else

{

bt=(BT\*)malloc(sizeof(BT));

bt->data=x;

bt->lchild=createtree();

bt->rchild=createtree();

}

return bt;

}

ShowTree(BT \*T)

{

int i;

if(T==NULL)

{

return;

}

else

{

num++;

printf("\t\t\t");

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

{

printf(" ");

}

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

{

printf("-");

}

printf("\n");

ShowTree(T->lchild);

ShowTree(T->rchild);

num=num-1;

}

}

void Preorder(BT \*T)

{

if(T==NULL)

{

return;

}

else

{

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

Preorder(T->lchild);

Preorder(T->rchild);

}

}

void Inorder(BT \*T)

{

if(T==NULL)

{

return;

}

else

{

Inorder(T->lchild);

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

Inorder(T->rchild);

}

}

void Postorder(BT \*T)

{

if(T==NULL)

{

return;

}

else

{

Postorder(T->lchild);

Postorder(T->rchild);

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

}

}

void Levelorder(BT \*T)

{

int i,j;

BT \*q[MAXLEN],\*p;

p=T;

if (p!=NULL)

{

i=1;

q[i]=p;

j=2;

}

while (i!=j)

{

p=q[i];

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

if( p->lchild!=NULL)

{

q[j]=p->lchild;

j++;

}

if(p->rchild!=NULL)

{

q[j]=p->rchild;

j++;

}

i++;

}

}

void Nodenum(BT \*T)

{

if(T==NULL)

{

return;

}

else

{

counter2++;

Nodenum(T->lchild);

Nodenum(T->rchild);

}

}

void Leafnum(BT \*T)

{

if(T==NULL)

{

return;

}

else

{

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

{

counter1++;

}

Leafnum(T->lchild);

Leafnum(T->rchild);

}

}

int TreeDepth(BT \*T)

{

int ldep,rdep;

if(T==NULL)

{

return 0;

}

else

{

ldep=TreeDepth(T->lchild);

rdep=TreeDepth(T->rchild);

if(ldep>rdep)

{

return ldep+1;

}

else

{

return rdep+1;

}

}

}

void Menu()

{

printf("\n\t\t\t 二 叉 树 子 系 统 \n");

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

printf("\t\* 1------建立二叉树的二叉链表存储结构 \*\n");

printf("\t\* 2------按凹入表显示二叉树 \*\n");

printf("\t\* 3------按先序遍历二叉树 \*\n");

printf("\t\* 4------按中序遍历二叉树 \*\n");

printf("\t\* 5------按后序遍历二叉树 \*\n");

printf("\t\* 6------按层次遍历二叉树 \*\n");

printf("\t\* 7------统计二叉树的叶子结点总数 \*\n");

printf("\t\* 8------统计二叉树的结点总数 \*\n");

printf("\t\* 9------统计二叉树的深度 \*\n");

printf("\t\* 0------返 回 \*\n");

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

printf("\t 请选择要执行的操作代号（0-9）: ");

}

void main()

{

int i,choose;

datatype x;

BT \*T,\*t;

while(1)

{

Menu();

scanf("%d",&choose);

if(choose==1)

{

printf("\n\t 按先序次序输入二叉树中结点的值,以0字符表示空树\n");

printf("\t请输入： ");

T=createtree();

printf("\t 数据输入完毕!\n");

}

else if(choose==2)

{

printf("\n\t\t\t凹入表显示二叉树\n");

ShowTree(T);

}

else if(choose==3)

{

printf("\n\t\t先序遍历二叉树: ");

Preorder(T);

printf("\n");

}

else if(choose==4)

{

printf("\n\t\t中序遍历二叉树: ");

Inorder(T);

printf("\n");

}

else if(choose==5)

{

printf("\n\t\t后序遍历二叉树: ");

Postorder(T);

printf("\n");

}

else if(choose==6)

{

printf("\n\t\t层次遍历二叉树: ");

Levelorder(T);

printf("\n");

}

else if(choose==7)

{

Leafnum(T);

printf("\n\t树的叶子节点数为: %d\n",counter1);

counter1=0;

}

else if(choose==8)

{

Nodenum(T);

printf("\n\t树的节点数为: %d\n",counter2);

counter2=0;

}

else if(choose==9)

{

printf("\n\t树的深度为: %d\n",TreeDepth(T));

}

else if(choose==0)

{

break;

}

else

{

printf("\n\t\t 输入有误，请重新选择操作代号\n");

}

}

}

### 5、存在问题及解决办法