**广 州 商 学 院**

**实验报告（第 4 次）**

**实验名称： 二叉树的基本操作 实验时间： 2021.4.23**

1. **实验目的**

1．掌握二叉树的定义；

2．掌握二叉树的基本操作，如二叉树的建立、遍历、结点个数统计、树的深度计算等。

1. **实验内容**

用递归的方法实现以下算法：

1．以二叉链表表示二叉树，建立一棵二叉树（算法5.3）；

2．输出二叉树的中序遍历结果（算法5.1）；

3．输出二叉树的前序遍历结果；

4．输出二叉树的后序遍历结果；

5．计算二叉树的深度（算法5.5）；

6．统计二叉树的结点个数（算法5.6）；

7．统计二叉树的叶结点个数；

8．统计二叉树的度为1的结点个数；

9．输出二叉树中从每个叶子结点到根结点的路径。

**源程序及主要算法说明//刘欢202001010193**

**#include <iostream>**

**using namespace std;**

**typedef struct Node**

**{//定义二叉树结构**

**char data;**

**struct Node \*lchild,\*rchild;**

**}\*BiTree,BiTNode;**

**void CreateBiTree(BiTree &T)**

**{//先序创建二叉树**

**char ch;**

**cin>>ch;**

**if(ch=='#') T=NULL;**

**else{**

**T=new BiTNode;**

**T->data=ch;**

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

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

**}**

**}**

**void InOrderTraverse(BiTree T)**

**{//中序遍历**

**if(T)**

**{**

**InOrderTraverse(T->lchild);**

**cout<<T->data;**

**InOrderTraverse(T->rchild);**

**}**

**}**

**void PreOrderTraverse(BiTree T)**

**{//先序遍历**

**if(T)**

**{**

**cout<<T->data; ;**

**PreOrderTraverse(T->lchild);**

**PreOrderTraverse(T->rchild);**

**}**

**}**

**void PostOrderTraverse(BiTree T)**

**{//后序遍历**

**if(T)**

**{**

**PostOrderTraverse(T->lchild);**

**PostOrderTraverse(T->rchild);**

**cout<<T->data;**

**}**

**}**

**void Copy(BiTree T,BiTree &NewT)**

**{//二叉树的复制**

**if(T==NULL){**

**NewT=NULL;**

**return;**

**}else**

**{**

**NewT=new BiTNode;**

**NewT->data=T->data;**

**Copy(T->lchild,NewT->lchild);**

**Copy(T->rchild,NewT->rchild);**

**}**

**}**

**int Depth(BiTree T)**

**{//树的深度**

**if(T==NULL)**

**return 0;**

**else**

**{**

**int m=Depth(T->lchild);**

**int n=Depth(T->rchild);**

**if( m>n) return (m+1);**

**else return (n+1);**

**}**

**}**

**int NodeCount(BiTree T)**

**{//统计二叉树中结点的个数**

**if(T==NULL) return 0;**

**else return NodeCount(T->lchild)+NodeCount(T->rchild)+1;**

**}**

**int LeafCount(BiTree T)**

**{//统计二叉树中叶子结点的个数**

**if(!T) return 0;**

**if(!T->lchild &&!T->rchild){//如果二叉树左子树和右子树皆为空,说明该二叉树根节点为叶子节点,加1.**

**return 1;**

**}else{**

**return LeafCount(T->lchild)+LeafCount(T->rchild);**

**}**

**}**

**int Node\_1\_Count(BiTree T)**

**{//统计二叉树的度为1的结点个数**

**if(!T) return 0;**

**if((!T->lchild)&&(T->rchild)||(T->lchild)&&(!T->rchild))**

**return 1+Node\_1\_Count(T->lchild)&&(!T->rchild);**

**else**

**return Node\_1\_Count(T->lchild) + Node\_1\_Count(T->rchild);**

**}**

**void PrintAllPath(BiTree T, char path[], int pathlen)**

**{//二叉树中从每个叶子结点到根结点的路径**

**int i;**

**if(T != NULL) {**

**path[pathlen] = T->data; //将当前结点放入路径中**

**if(T->lchild == NULL && T->rchild == NULL) {//叶子结点**

**for(i = pathlen; i >= 0; i--)**

**cout << path[i] << " " ;**

**cout << endl;**

**}else{**

**PrintAllPath(T->lchild, path, pathlen + 1);**

**PrintAllPath(T->rchild, path, pathlen + 1);**

**}**

**}**

**}**

**void ExChangeTree(BiTree &T)**

**{//构造函数，使用递归算法进行左右结点转换**

**BiTree temp;**

**if(T!=NULL){//判断T是否为空，非空进行转换，否则不转换**

**temp=T->lchild;**

**T->lchild =T->rchild;//直接交换节点地址**

**T->rchild=temp;**

**ExChangeTree(T->lchild);**

**ExChangeTree(T->rchild);**

**}**

**}**

**int main()**

**{**

**BiTree T;**

**//测试例子AB#CD##E##F#GH###**

**cout<<"先序遍历输入(以#结束):";**

**CreateBiTree(T);**

**cout<<"中序遍历输出:";**

**InOrderTraverse(T);**

**cout<<endl<<"先序遍历输出:";**

**PreOrderTraverse(T);**

**cout<<endl<<"后序遍历输出:";**

**PostOrderTraverse(T);**

**cout<<endl<<"树的深度:"<<Depth(T);**

**cout<<endl<<"结点的个数:"<<NodeCount(T);**

**cout<<endl<<"叶结点的个数:"<<LeafCount(T);**

**cout<<endl<<"度为1的结点个数:"<<Node\_1\_Count(T);**

**cout<<endl<<"二叉树中从每个叶子结点到根结点的所有路径："<<endl;**

**char path[256];**

**int pathlen=0;**

**PrintAllPath(T,path,pathlen);//**

**//交换二叉树每个结点的左孩子和右孩子**

**BiTree tem=T;//直接复制一颗树，在不改变原树的前提下，对临时树进行交换。**

**ExChangeTree(tem);**

**cout<<"先序遍历输出交换后的结果:";**

**PreOrderTraverse(tem);**

**return 0;**

**}**

**三、主要问题和解决方案**

**对概念还不是太熟悉，要不断翻书进行查找相关的细节。**

1. **测试数据及结果**

****

1. **心得体会与自我评价**

**虽然概念不是特别熟悉，但是最后能够运行出来还是挺开心的！**

**六、教师评分**