西安财经大学 信息学院

姓名：程驰

学号：1931054041

班级：计本1902

指导教师：李薇

成绩：

数据结构 实验报告

# 实验名称:二叉树的创建和遍历

实验日期：2020年11月15日

**一、实验目的：**

1、熟悉二叉树结点的结构和对二叉树的基本操作；

2、掌握对二叉树的基本操作；

3、学会递归方法和非递归方法实现二叉树的遍历操作。

**二、实验内容：**

1、建立一棵二叉树。

2、对二叉树进行先序、中序和后序遍历。

3、实现二叉树的基本应用。

三、实验要求：

编写实现二叉树建立和遍历的基本算法的函数，并在此基础上设计一个主程序完成如下功能：

⑴定义二叉树的二叉链表存储结构;

⑵以扩展先序遍历序列建立二叉树的二叉链表；

⑶分别实现二叉树的先序、中序和后续遍历，并打印输出；

⑷计算二叉树的高度；

⑸计算二叉树节点的个数。

⑹计算二叉树叶子节点的个数。

四、实验步骤：

1.确定实验环境：Windows10中文版、Dev-cpp、Visual Studio 2019

2.建立定义二叉树的二叉链表存储结构

typedef char TElemType;

typedef struct BiTNode {

TElemType data;

struct BiTNode \*lchild,\*rchild;

} BiTNode,\*BiTree;

3. 以扩展先序遍历序列建立二叉树的二叉链表

void CreateBiTree(BiTree &T) {

//先序

char ch;

cin>>ch;

if(ch=='#')

T=NULL;

else {

T=new BiTNode;

T->data=ch;

CreateBiTree(T->lchild);

CreateBiTree(T->rchild);

}

}

4.二叉树的先序、中序和后续遍历

void InOrderTraverse1(BiTree T) {

//先序遍历

if(T) {

cout<<T->data;

InOrderTraverse1(T->lchild);

InOrderTraverse1(T->rchild);

}

}

void InOrderTraverse2(BiTree T) {

//中序遍历

if(T) {

InOrderTraverse2(T->lchild);

cout<<T->data;

InOrderTraverse2(T->rchild);

}

}

void InOrderTraverse3(BiTree T) {

//后序遍历

if(T) {

InOrderTraverse3(T->lchild);

InOrderTraverse3(T->rchild);

cout<<T->data;

}

}

5. 计算二叉树的高度；

int Depth(BiTree T) {

int n,m;

if(T==NULL)

return 0;

else {

m=Depth(T->lchild);

n=Depth(T->rchild);

if(m>n)

return(m+1);

else

return(n+1);

}

}

6.计算二叉树节点的个数。

int NodeCount(BiTree T) {

if(T==NULL)

return 0;

else

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

}

7.计算二叉树叶子节点的个数。

int LeafNum(BiTree T) {

if (!T) {

return 0;

} else if (!T->lchild && !T->rchild) {

return 1;

} else {

return LeafNum(T->lchild) + LeafNum(T->rchild);

}

}

附程序代码清单：

#include <iostream>

#define MAXSIZE 100

using namespace std;

typedef char TElemType;

typedef struct BiTNode {

TElemType data;

struct BiTNode \*lchild,\*rchild;

} BiTNode,\*BiTree;

void InOrderTraverse1(BiTree T) {

//先序遍历

if(T) {

cout<<T->data;

InOrderTraverse1(T->lchild);

InOrderTraverse1(T->rchild);

}

}

void InOrderTraverse2(BiTree T) {

//中序遍历

if(T) {

InOrderTraverse2(T->lchild);

cout<<T->data;

InOrderTraverse2(T->rchild);

}

}

void InOrderTraverse3(BiTree T) {

//后序遍历

if(T) {

InOrderTraverse3(T->lchild);

InOrderTraverse3(T->rchild);

cout<<T->data;

}

}

void CreateBiTree(BiTree &T) {

//先序

char ch;

cin>>ch;

if(ch=='#')

T=NULL;

else {

T=new BiTNode;

T->data=ch;

CreateBiTree(T->lchild);

CreateBiTree(T->rchild);

}

}

int Depth(BiTree T) {

int n,m;

if(T==NULL)

return 0;

else {

m=Depth(T->lchild);

n=Depth(T->rchild);

if(m>n)

return(m+1);

else

return(n+1);

}

}

int NodeCount(BiTree T) {

if(T==NULL)

return 0;

else

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

}

int LeafNum(BiTree T) {

if (!T) {

return 0;

} else if (!T->lchild && !T->rchild) {

return 1;

} else {

return LeafNum(T->lchild) + LeafNum(T->rchild);

}

}

int main()

{

BiTree a;

int height,num,leafnum;

cout<<"请输入二叉树的元素!\n";

CreateBiTree(a);

cout<<"按先序遍历法输出该二叉树的元素为：\n";

InOrderTraverse1(a);

cout<<"\n按中序遍历法输出该二叉树的元素为：\n";

InOrderTraverse2(a);

cout<<"\n按后序遍历法输出该二叉树的元素为：\n";

InOrderTraverse3(a);

height=Depth(a);

num=NodeCount(a);

leafnum= LeafNum(a);

cout<<"\n该二叉树高度为：";

cout<<height<<endl;

cout<<"该二叉树结点数为：";

cout<<num;

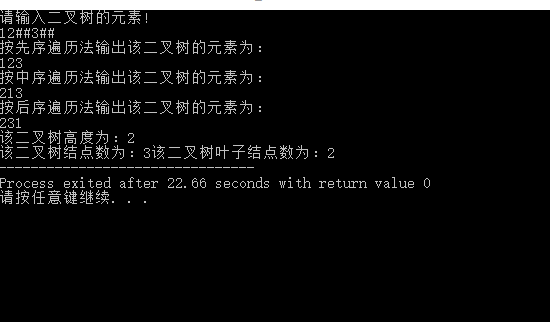
cout<<"该二叉树叶子结点数为：";

cout<<leafnum;

return 0;

}

五、实验结果：



六、实验总结：

“#”表示空，如果单纯的输入元素“1、2、3”，不能判断其是否有空节点，故程序无法继续运行。