数据结构实验报告

实验成绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 |  | 学号 |  | 专业班级 |  |
| 指导教师 | 张先宜 | 实验时间 |  | 实验地点 |  |

**实验名称： 树于森林实验**

1. **实验目标**

(1)掌握树和森林的孩子兄弟链表（二叉链表）表示方法。

(2)掌握树和二叉树的结构及算法之间的对应关系。

(3)掌握树的两种遍历算法及其应用。

1. **实验要求**
2. 结构定义和算法实现放入库文件，如“treeChildSibling.h”中；
3. 数和森林的测试数据用文本文件方式给出，例如测试数据名为tree10.tre的树或森林，可参考发来的树和森林形状和参考存储文件；
4. 数和森林创建方法可自行选择；
5. 可多次连续测试。
6. **实验内容**

设计、实现算法求解下列问题：

1. 按先序、后序、层次遍历森林。

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

1. 求森林的高度。

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

1. 求森林叶子结点数。

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

1. 求森林的度。

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

1. 先序输出结点值及其层次号。

例对图7-1所示森林，输出为：(A,1) (B,2) (E,3) (K,4) (F,3) (G,3) (C,2) (H,3) (I,3) (D,2) (J,3) (L,1) (M,2) (N,2) (O,1) (P,2)

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

1. 输出广义表表示的树。

例对图7-1所示森林，输出为：A( B(E(K),F,G),C(H,I),D(J)), L(M,N), O(P) )

实验测试数据基本要求：

第一组数据： tree11.tre

第二组数据： f20.tre

图7-1 一个森林示意图

A

B

D

F

I

E

G

J

K

H

C

N

M

L

O

P

1. **数据结构设计**

typedef char elementType;

typedef struct pNode

{

elementType data; //结点数据域

int parent; //父结点指针（下标）

}PTNode;

typedef struct pTree

{

PTNode node[MAXLEN]; //结点数组

int n; //结点总数

}pTree;

typedef struct csNode

{

elementType data;

struct csNode\* firstChild, \* nextSibling;

}csNode, \* csTree;

1. **算法设计**

1）,2）,3）操作与二叉树相同

4）在求度时，先求出所有结点的度然后以最大的度作为森林的度

int GetTreeDegree(csNode\* t) //4.求森林的度

{

if (t == NULL)

return 0;

int degree = GetNodeDegree(t);

int a = fmax(GetTreeDegree(t->firstChild), GetTreeDegree(t->nextSibling));

return fmax(a, degree);

}

5）先序输出结点值及其层次号操作与先序输出森林类似，只要加入一个表示层次的变量即可。

void PreOrderAndLeveal(csNode\* t, int i)//i的初始值为1 //5.先序输出结点值及其层次号

{

if (t)

{

printf("(%c,%d)", t->data, i);

PreOrderAndLeveal(t->firstChild, i + 1);

PreOrderAndLeveal(t->nextSibling, i);

}

return;

}

6）输出广义表表示的树时需要考虑每个结点的左孩子与右兄弟是否存在，以及各种情况下应该输出的符号。

char\* TransArray(csNode\* t) { //6.将树转换为广义表表示的形式

static int i = 0;

static char ch[100];

if (t == NULL)

{

return NULL;

}

ch[i++] = t->data;

if (t->firstChild)

{

ch[i++] = '(';

TransArray(t->firstChild);

if (t->nextSibling)

{

ch[i++] = ',';

}

else

{

ch[i++] = ')';

}

}

else

{

if (t->nextSibling)

{

ch[i++] = ',';

}

else

{

ch[i++] = ')';

}

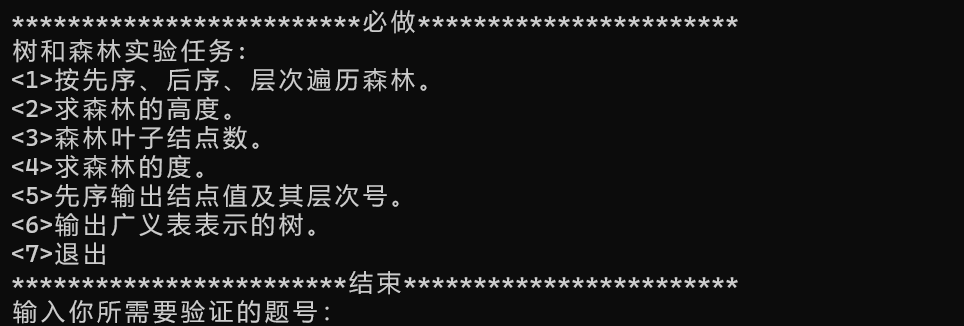
}

TransArray(t->nextSibling);

return ch;

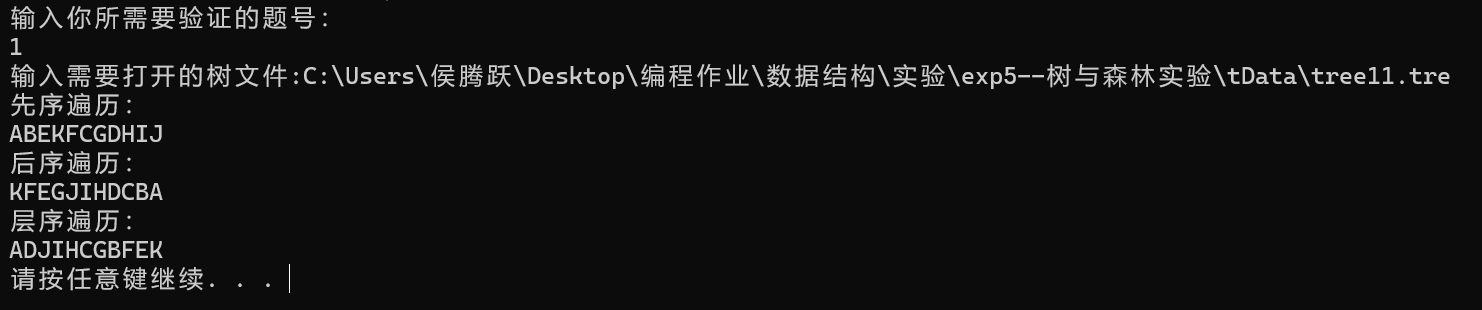
}

1. **运行和测试**

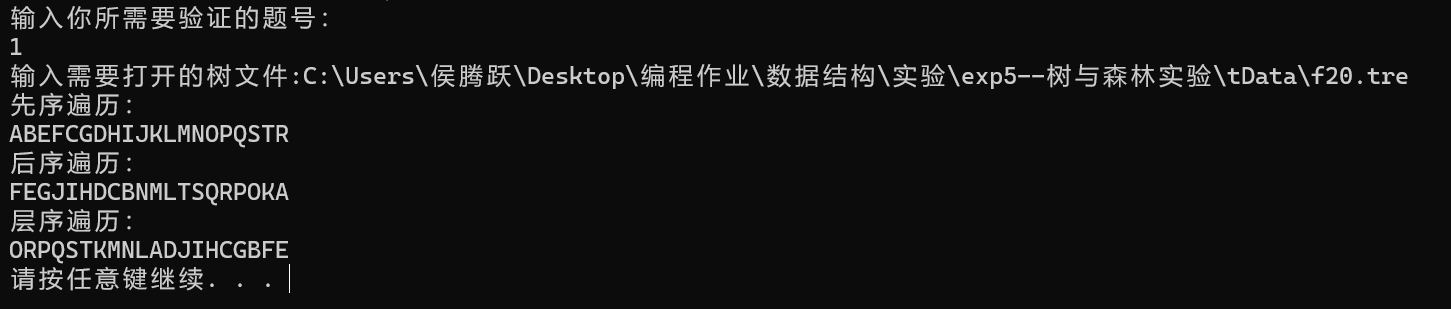


1）按先序、后序、层次遍历森林。

第一组数据： tree11.tre

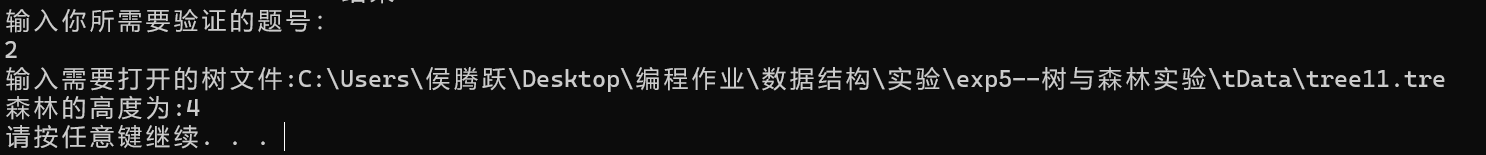


第二组数据： f20.tre

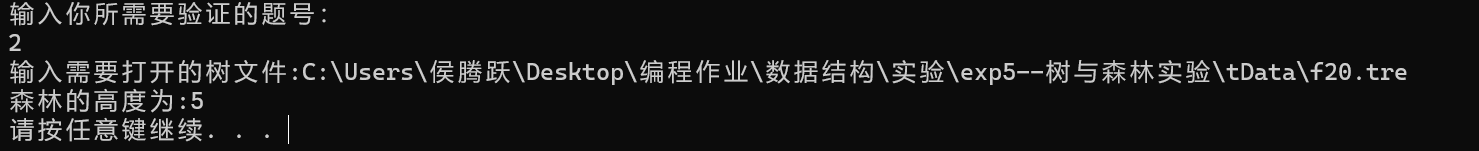


2）求森林的高度。

第一组数据： tree11.tre



第二组数据： f20.tre

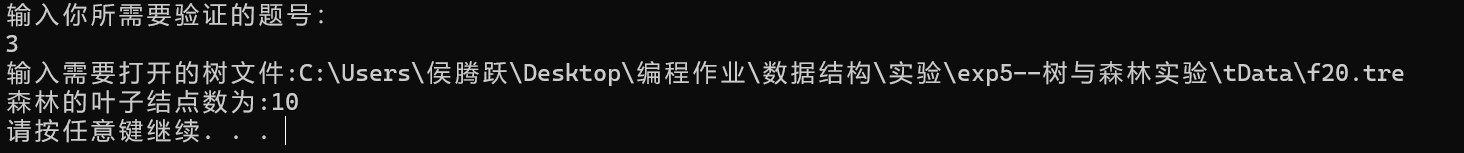


3）求森林叶子结点数。

第一组数据： tree11.tre

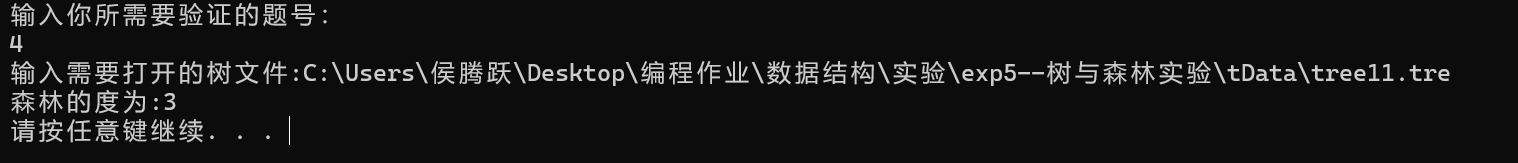


第二组数据： f20.tre

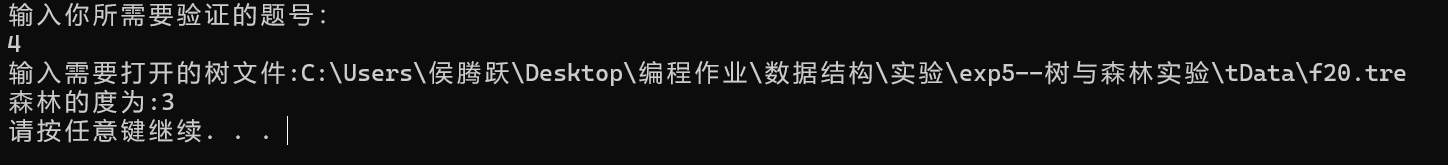


4）求森林的度。

第一组数据： tree11.tre

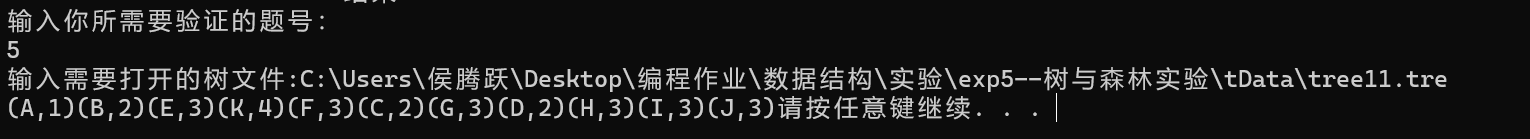


第二组数据： f20.tre

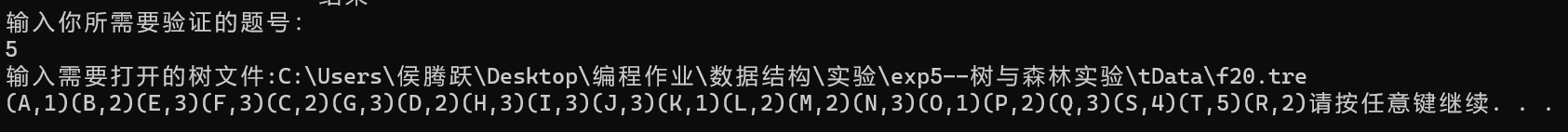


5）先序输出结点值及其层次号。

第一组数据： tree11.tre

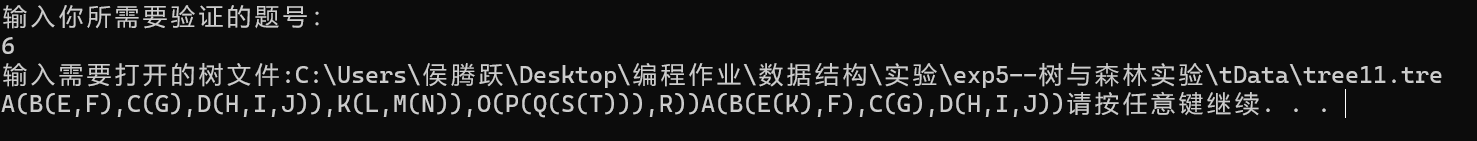


第二组数据： f20.tre

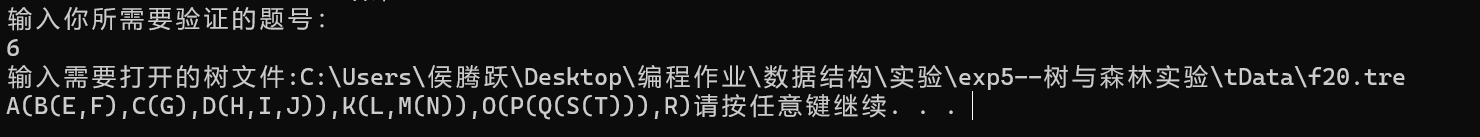


6）输出广义表表示的树。

第一组数据： tree11.tre



第二组数据： f20.tre



1. **总结、心得和建议**

在本次实验中，我对前面所学的链表、递归以及队列有了更加深刻的认识，更加熟悉了他们的使用方法；参考了老师提供的创建二叉链表表示的树（森林）的代码，理清了代码的实现原理；此次实验让我知道了森林的便利之处，有助于我以后日常工作中去完善项目，也促进我去学习更多知识去进步。