数据结构实验报告

实验成绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 邓凯文 | 学号 | 2021218149 | 专业班级 | 计算机21-3班 |
| 指导教师 | 张先宜 | 实验时间 | 2022.6.8 | 实验地点 |  |

**实验名称： 二叉树实验**

1. **实验目标**

掌握二叉树的动态链表存储结构及表示。

掌握二叉树的三种遍历算法（递归和非递归两类）。

运用二叉树三种遍历的方法求解有关问题。

1. **实验内容和要求**

结构定义和算法实现放入库文件，如“BiTree.h”中；

二叉树的测试数据用文本文件方式给出，例如测试数据名为bt151.btr的二叉树，可参考发来的二叉树形状和参考存储文件；

二叉树创建方法可自行选择；

可多次连续测试。

设计二叉树的二叉链表存储结构，编写算法实现下列问题的求解。

1. 设计算法按中序次序输出二叉树中各结点的值及其所对应的层次数。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 求二叉树的叶子结点数和1度结点数。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

1. 键盘输入一个元素x，求其父节点、兄弟结点、子结点的值，不存在时给出相应提示信息。对兄弟结点和孩子结点，存在时要明确指出是左兄弟、左孩子、右兄弟或右孩子。

实验测试数据基本要求：

第一组数据： bt31.btr

第二组数据： bt21.btr

1. 键盘输入一个元素x，求其在树中的层次。不存在时给出相应提示信息。

实验测试数据基本要求：

第一组数据： bt26.btr

第二组数据： bt21.btr

1. 将按顺序方式存储在数组中的二叉树转换为二叉链表形式。（数组中要扩展为完全二叉树）。

实验测试数据基本要求：

第一组数据： bt8.btr

第二组数据： bt14.btr

1. 输出二叉树从每个叶子结点到根结点的路径（经历的结点）。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

1. 对二叉链表表示的二叉树，按从上到下，从左到右打印结点值，即按层次遍历序打印。（提示：需要使用队列）

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

拓展题 非必做内容，有兴趣的同学选做，

1. 复制一棵二叉树T到T1。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 交换二叉树中每个结点的左右孩子指针的值。（即：左子树变为右子树，右子树变为左子树）。

实验测试数据基本要求：

第一组数据： bt151.btr

第二组数据： bt21.btr

1. 对二叉链表表示的二叉树，求2个结点最近的共同祖先。

实验测试数据基本要求：

第一组数据： bt261.btr

第二组数据： bt21.btr

1. **数据结构设计**

定义的树的节点的结构体是由数据域data 左孩子指针lc 右孩子指针rc以及父亲节点的指针parent组成的(方便于后续算法寻找父亲节点，兄弟节点的操作)，同时因为层序输出的程序是需要队列这一结构的，同时也定义了顺序队列，其中包含了初始化、出队、入队、判空的基本运算。

1. **算法设计**

（除书上给出的基本运算（这部分不必给出设计思想），其它实验内容要给出算法设计思想）

第一题：void类型函数 此题的思想是由中序遍历改变而来的，在遍历的时候添加一个计数器i，每当访问当前节点的孩子节点就将i+1，输入函数的i=1，这样每次访问孩子节点i都+1正好对应层数+1，符合题目的需求。

第二题:int类型函数 类似遍历的思想，当遍历到节点T时判断是否含有孩子节点，若左右孩子节点都不存在则为叶子节点返回值1，若含有左右孩子则递归，将节点指向左右孩子并且返回值0。这样可以达到对叶子节点的记数。

int类型函数对于度为1的节点的判断也是基于递归函数，判断当前节点是否为空，若存在左孩子或者右孩子之一，则返回1+对应的存在的孩子的递归函数寻找其余的节点，若左右孩子都不存在或者都存在则直接对其的左右孩子进行递归函数的判断。

第三题:void类型的函数 但含有一个node\* 类型的find函数跳板(遍历改造)，先找到是否存在输入值x对应的节点，如果find函数未找到则输出不存在该节点结束该函数，反之判断其是否含有父亲节点(使用自己数据结构设计中定义的parent指针指回判断是否为NULL)，若存在，则判断其父亲节点是否由左（右）孩子(需要判断当前节点是左孩子还是右孩子),判断后若含有兄弟节点则输出，反之则不存在其兄弟节点。对于其的孩子节点，直接使用lc，rc指针判断是否为空即可相对而言比较简单。

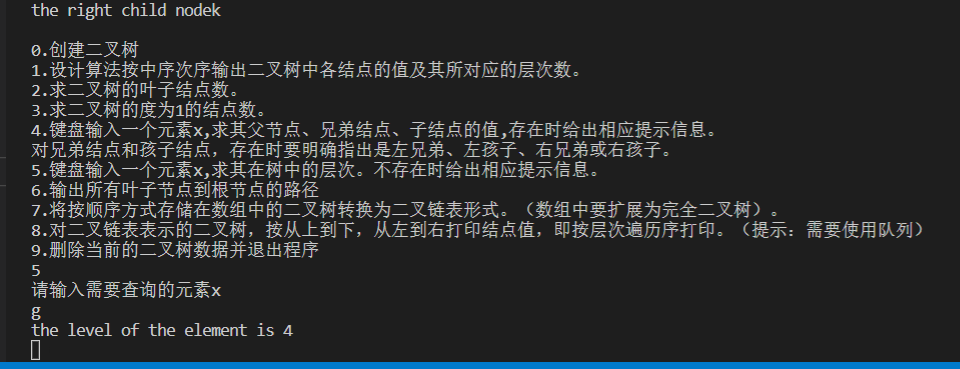
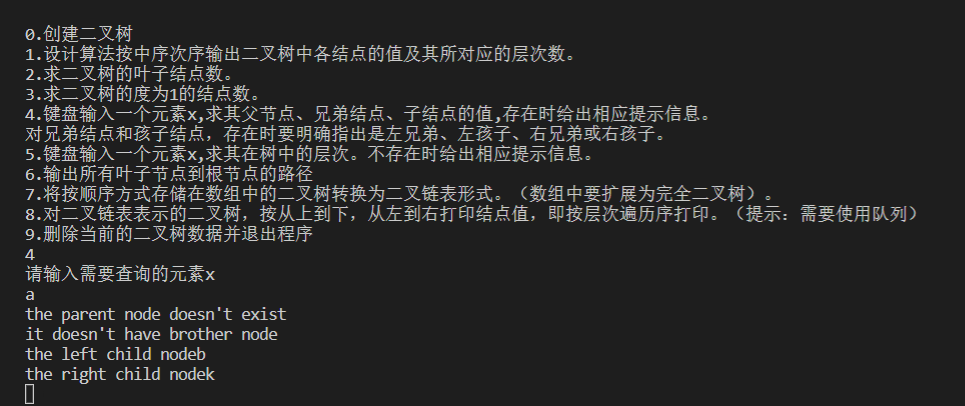
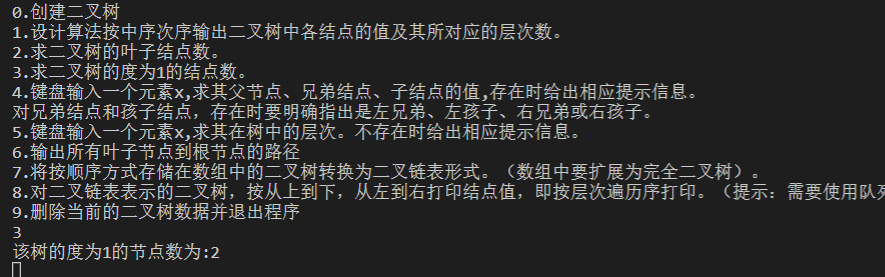
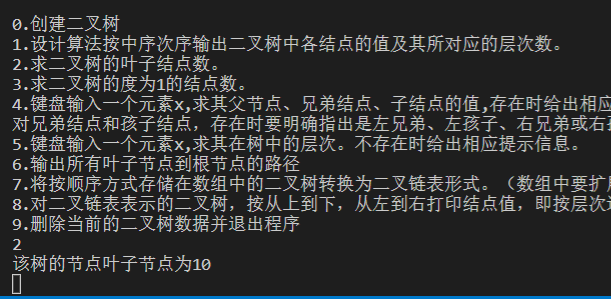
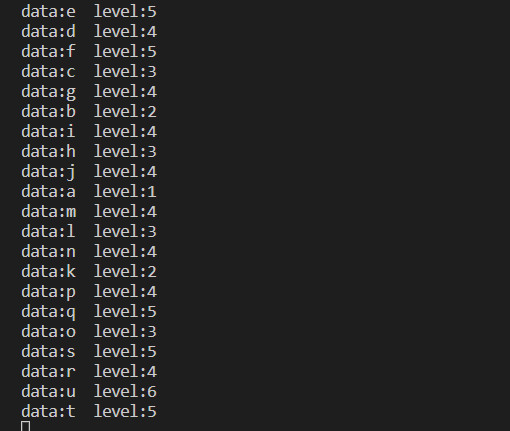
第四题:bool类型的函数（含有计数器i），基于先序遍历的改造，先对当前节点进行判断，是否不空且是否值等于e。若都满足则返回true，反之遍历寻找其的孩子节点，并且计数器+1，保证层数的对应，因为是bool类型的函数，所以递归也使用了if，若该递归能找到则会逐层返回true，实现最初的函数返回true(可能复杂度相对直接传值比较高)，若所有递归完成未找到则返回false。

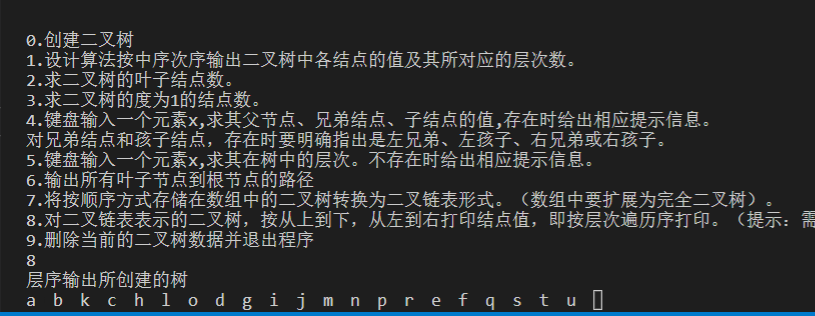
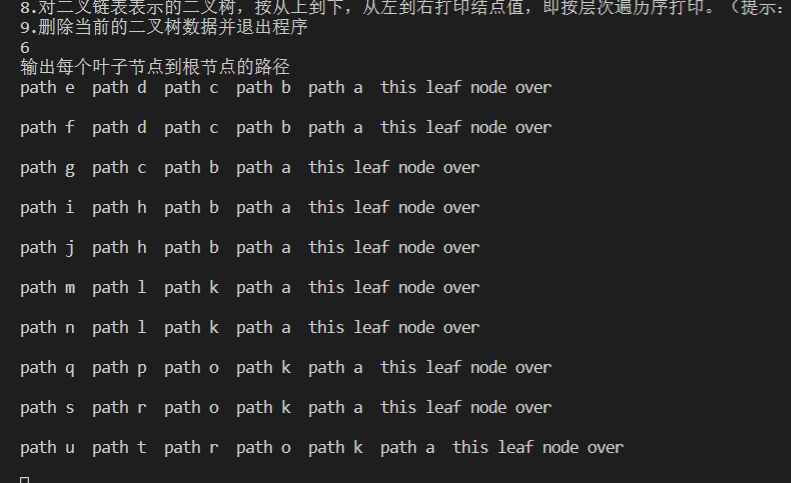
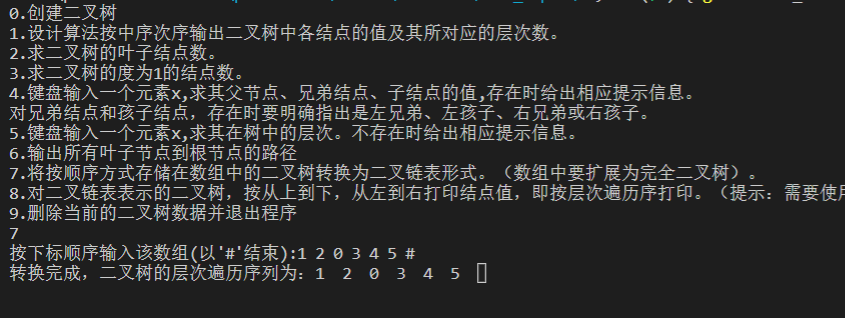
第五题:void类型函数，规定’^’和0为无效数据代表空子树，同时’#’代表输入数组时的终止符号，在函数内创建新节点，对应地接收A[i]元素，并且将其左右子树先置空，根据书中的推论，若当前i节点存在左右子树，则应当为2i和2i+1项，本题也是由此进行二叉树的创建的思想。

第六题:void类型函数，创建一个记录数据的节点，整体思想是先判断当前节点是否为根节点，是根节点则使用while循环输出其当前节点的值，并且将节点指向其parent，直到当前节点为根节点，停止输出且输出该叶子节点的路径已经输出完毕，进行路径的区分。至于递归的实现则是类似先序遍历，先进行判断再进行对其左右孩子的递归。

第七题:void类型函数 层序输出需要使用到队列这一数据结构，因为需要按FIFO原则进行输出，否则层序遍历难以实现，基本思路是当前节点入栈，若栈不空则使栈内队首元素出栈，并且判断该元素是否含有左右子树，若有则按左右的顺序入栈，这样可以保证是从上到下从左到右的顺序遍历二叉树。

**运行和测试**





1. **总结、心得和建议**

本次的二叉树实验相对前几次的实验而言更重要的是对递归的理解，如果没有递归的思想，在二叉树的实验中可能难以行动，相对于递归，非递归的方法代码量会更大。同时在本次二叉树的实验中穿插了队列的结构，用于层序遍历的输出。基本上大多数的实验题都是基于遍历改造而来的，在理解递归的工作后会变得比较简单。在实验中遇到的问题有在读取数据文件的时候我的电脑貌似会因为数据文件中的注释出现读不了文件的情况，也有在后续的层序遍历中出现的一些问题，因为开始没有想明白，对左右子树应该入栈的时候选择的是递归原函数，结果变成了先序遍历，不过之后通过调试也找出了此问题。同时，在本次的实验中，结构体的编写变得更加轻松，这也得益于前几次实验的结果。本次实验给我的最大收获就是递归的理解以及二叉树的形式，当然，也是对我编程调试的能力的一种历练。