|  |  |  |  |
| --- | --- | --- | --- |
| **《数据结构与算法》实验报告** | | | |
| **学生姓名** |  | **院（系）** | 计算学部 |
| **学 号** |  | **专 业** | 计算机类 |
| **实验时间** | 2021年10月13日（周三） | **实验地点** | 格物213室 |
| **实验项目** | **实验3-1：树型结构的建立与遍历（3学时）** | | |
| **实验目的：**将课程的基本原理、技术和方法与实际应用相结合，训练和提高学生组织、存储和处理信息的能力，以及复杂问题的数据结构设计能力和程序设计能力，培养软件设计与开发所需要的实践能力。  **实验要求：**灵活运用基本的数据结构和算法知识，对实际问题进行分析和抽象；结合程序设计的一般过程和方法为实际问题设计数据结构和有效算法；用高级语言对数据结构和算法进行编程实现、调试，测试其正确性和有效性。 | | | |
| **实验内容：**  **树型结构的遍历是树型结构算法的基础，本实验要求编写程序演示二叉树的存储结构的建立方法和遍历过程。**   1. **编写建立二叉树的二叉链表存储结构（左右链表示）的程序，并以适当的形式显示和保存二叉树；** 2. **采用二叉树的二叉链表存储结构，编写程序实现二叉树的先序、中序和后序遍历的递归和非递归算法以及层序遍历算法，并以适当的形式显示和保存二叉树及其相应的遍历序列；** 3. **给定一个二叉树，** 编写算法完成下列应用:（二选一）    1. 判断其是否为完全二叉树；    2. 求二叉树中任意两个结点的公共祖先。 | | | |
| 数据结构定义：  ①实验中采用二叉链表的方式构造二叉树，二叉树结点Node的结构体包含内容有：  *Char*类型变量*data*，存储二叉树结点的数据。  *Node*类型指针变量*lchild*和*rchild*，分别指向左孩子和右孩子结点。  同时定义该结构体的指针类型为*BTREE*  ②栈的数据结构定义：在实现二叉树的前序，中序，后序非递归算法时，引入了栈这一类型的数据结构，载入头文件#include <stack>。  ③队列的数据结构定义：在实现二叉树的层序遍历时，需要引入队列这一类型的数据结构，载入头文件#include <queue>。 | | | |
| 算法设计与分析（要求画出核心内容的程序流程图）：  ①**任务1的算法设计(建立二叉树的二叉链表存储结构)：**  任务1主要利用了递归建立二叉树的思想，输入要建立二叉树的带“#”的先序序列，逐一获取序列的字符来构造当前的根节点（相对于子树），然后递归调用该方法建立左孩子的子树，同样会以左孩子为其子树的根节点，右孩子类似。在遇到“#”后，说明该节点是空的，要做特殊判断并返回空给它的根节点，表明该结点无此孩子。  ②**任务2的算法设计：**  ⑴递归算法实现先序，中序，后序遍历：  使用递归算法实现遍历较为简单便捷，在先序遍历时，先访问根节点，再以先序遍历左孩子，返回后再先序遍历右孩子，这样在以孩子为根节点的下一层递归中，又可以重复此过程。当孩子为空时，直接返回即可。中序和后序遍历与先序遍历的实现方法类似，只需调换访问和遍历的顺序。  ⑵非递归算法实现先序，中序，后序遍历：  在实现非递归算法遍历时，需要使用栈这一数据结构，根据栈先进后出的特性，来完成遍历。  I先序遍历的非递归算法设计：  基本思想：根据先序遍历的次序为先访问根节点，然后访问左子树，右子树，可以借用栈这一数据结构，声明一个指针p指向根节点，先访问根节点，然后让根结点入栈并输出其值，并让指针p在左子树上滑动直到访问到空结点，令栈顶元素出栈，让p指向栈顶元素的右子树，继续重复该操作即可。  算法步骤的文字说明：  1)输出节点p，然后将其入栈，再看p的左孩子是否为空；  2)若p的左孩子不为空，则置p的左孩子为当前节点，重复1）的操作；  3)若p的左孩子为空，则将栈顶节点出栈，但不输出，并将出栈节点的右孩子置为当前节点，看其是否为空；  4)若不为空，则循环至1)操作；  5)如果为空，则继续出栈，但不输出，同时将出栈节点的右孩子置为当前节点，看其是否为空，重复4）和5）操作；  6)直到当前节点p为NULL并且栈空，遍历结束。  程序流程图：    II中序遍历的非递归算法设计：  基本思想：根据中序遍历的定义，先访问左子树，再访问根节点，最后访问右子树。那么依然可以借助栈实现遍历。首先声明指针p指向当前节点，若当前节点有左子树，则当前节点入栈，若无左子树，则打印当前节点，p指针进入右子树，若无右子树，则栈顶元素出栈，直到栈顶元素有右子树为止。  算法步骤的文字说明：  1）若P的左孩子不为空，则将P入栈并将P的左孩子置为当前节点，然后再对当前节点进行相同的处理；  2）若P的左孩子为空，则输出P节点，而后将P的右孩子置为当前节点，看其是否为空；  3）若不为空，则重复1）和2）的操作；  4）若为空，则执行出栈操作，输出栈顶节点，并将出栈的节点的右孩子置为当前节点，看起是否为空，重复3）和4）的操作；  5）直到当前节点P为NULL并且栈为空，则遍历结束。  程序流程图：    III后序遍历的非递归算法设计：  基本思想：根据后续遍历的定义，先访问其左子树，再访问其右子树，最后访问其根节点，依然借助栈实现，声明两个指针pr,p。p用于指向当前节点，pr用于指向p的右孩子结点。先让p指向根结点，当p不为空时，p进栈，然后不断让p指向自己的左孩子结点，pr指向p的右孩子结点，当p不存在时，用pr更新p，不断重复这个步骤可使栈顶元素为后序遍历的第一个元素。p指向栈顶，输出栈顶元素并出栈，如果此时p为栈顶的左孩子结点，则访问栈顶的右孩子结点（实现先访问左子树，然后再访问右子树的顺序），否则p指向空，再重复操作。  算法步骤的文字说明：  1）若p非空，让p进栈，pr指向p的右孩子结点，p指向自身的左孩子结点，若p为空则用pr的值更新p，不断重复1)操作直到p为空  2）令p指向栈顶元素，栈顶元素出栈，输出p的数据值，  3）判断是否栈非空且栈顶元素的左孩子结点恰好为p  4）若是，将p指向栈顶元素的右孩子结点  5）若否，将p指向NULL，若栈不为空，重回步骤2)，若栈也为空，则遍历结束  程序流程图：    ⑵层序遍历算法  实现二叉树的遍历算法需要引入队列这一数据结构。层序遍历过程，其实就是从上到下，从左到右依次将每个数放入到队列中，然后按顺序依次打印就是想要的结果。首先将二叉树的根节点push到队列中，判断队列不为NULL，就输出队头的元素，判断节点如果有孩子，就将孩子push到队列中（左孩子先进，右孩子后进），遍历过的节点出队列，循环以上操作，直到队列清空。  **③任务三的算法设计**  选题：给定一个二叉树，判断其是否为完全二叉树  算法的基本思想及步骤：根据完全二叉树的N 个结点对应于相同深度完美二叉树的层序遍历的前 N 个结点的性质，引入队列这一数据结构，通过对二叉树进行层序遍历，对于第一个出现空的结点，则按层序遍历的次序，其后的所有结点都应该为空，若不满足，则不是完全二叉树，反之则为完全二叉树。  程序流程图： | | | |
| 实验测试结果及结果分析：  ①根据任务一设计的输入函数，输入ABDH##I##E##CF##G#J##(表示的树如下图1所示）  QQ截图20211016185038  图 1  输出如下图2所示  QQ截图20211016190739  图 2  结果分析：先序递归顺序，正确；中序递归顺序，正确；后序递归顺序，正确；先序非递归顺序，正确；中序非递归顺序，正确；后序非递归顺序，正确；层序遍历顺序，正确；完全二叉树判断为FALSE，正确。  ②输入ABDH##I##EJ##K##CF##G## 构造树如图3所示  QQ截图20211016200047  图 3  输出如图4所示    图 4  结果分析：先序递归顺序，正确；中序递归顺序，正确；后序递归顺序，正确；先序非递归顺序，正确；中序非递归顺序，正确；后序非递归顺序，正确；层序遍历顺序，正确；完全二叉树判断为TRUE，正确。 | | | |
| 问题及解决方法：  **问题1： 对于后序遍历的非递归实现方法掌握不到位**  解决方法：经过对课本及上课所讲的知识进行复习总结，对栈的使用有了更加深入的理解，使用栈这一数据结构，将结点按照一定的顺序入栈，出栈，输出，理论上是可以完成后序遍历的非递归实现，最终通过思考与练习，成功设计出了算法，完成了后序遍历的非递归实现。  **问题2： 不知道该如何判断一棵树是完全二叉树**  解决方法：在网上查阅了资料，了解了判断的基本思路：在判断完全二叉树时，首先要先对二叉树进行层序遍历，把一个节点的左右节点都给入队，递归着遍历，直到找到了一个空节点，此时先把队列的第一个空节点出队，然后再对该队列进行遍历，如果一旦队列中有其他的不为NULL的节点，那么该树就不是完全二叉树，否则，该树就是完全二叉树，这是一种非常简便的判断方式。  **问题3： 一开始不知道该如何进行层序遍历**  解决方法：层序遍历需要“先进先出”，即层层遍历的时候，一个节点越靠前就越先被遍历，所以考虑到这个特性，则决定采取队列来存储需要遍历的节点，用递归的方法，遍历到某个点，则再把它的左右儿子节点入队，如此即可实现树的层序遍历。 | | | |
| 源程序名称：实验1.cpp | | | |

注意：正文文字为宋体小4号，图中文字为宋体5号。行距为多倍行距1.25。

源程序与此报告打包提交，压缩包采用学号命名。