 **广东理工学院实验(训)报告**

**姓 名： 叶锦亮 学 号： 1812402301044**

**专业、班级： 18软件工程1班 课程名称： 数据结构与算法(Java)**

**指导教师： 翟允赛 实验地点：**

**实验时间： 2020.05.05 实验学时： 2学时**

**实验项目： 08 二叉树的建造**  **成 绩：**

|  |
| --- |
| **实验目的：**  1.理解二叉树的类型定义与性质。  2.掌握二叉树的二叉链表存储结构的表示和实现方法。  3.掌握二叉树遍历操作的算法实现。  4.掌握二叉树的构造方法。 |
| 实验内容和要求:   1. 将该实验报告以“**08**+全部学号+姓名.doc”格式命名，完成后于本周周日24：00前交到班长处，班长再统一交给我，过期不候！！（以后实验都按照本次要求）。 2. 已知一棵树二叉树的先根遍历和中根遍历的序列分别为：A B D G H C E F I和G D H B A E C I F，请画出此二叉树，并写出它的后根遍的序列。 3. 已知一棵树二叉树的后根遍历和中根遍历的序列分别为：A C D B G I H F E和A B C D E F G H I，请画出该二叉树，并写出它的先根遍历的序列。   4、实现二叉树的建立、遍历基本操作。编程实现如下功能：  （1）假设二叉树的结点值是字符如下图所示，根据输出的一棵二叉树的标明了空子树的完整先根遍历序列，建立一棵以二叉链表表示的二叉树。  （2）对二叉树进行先根、中根、后根遍历操作，并输出遍历序列，同时观察输出序列是否与逻辑上的序列一致。 |
| **实验过程:**  **节点类：**  **package** tree;  **public** **class** BinaryTreeNode {  **private** BinaryTreeNode lChild;  **private** BinaryTreeNode rChild;  **private** **char** data;  **public** BinaryTreeNode(BinaryTreeNode lChild, BinaryTreeNode rChild, **char** data) {  **super**();  **this**.lChild = lChild;  **this**.rChild = rChild;  **this**.data = data;  }    **public** BinaryTreeNode(**char** data) {  **super**();  **this**.data = data;  }  **public** BinaryTreeNode getlChild() {  **return** lChild;  }  **public** **void** setlChild(BinaryTreeNode lChild) {  **this**.lChild = lChild;  }  **public** BinaryTreeNode getrChild() {  **return** rChild;  }    **public** **void** setrChild(BinaryTreeNode rChild) {  **this**.rChild = rChild;  }  **public** **char** getData() {  **return** data;  }  **public** **void** setData(**char** data) {  **this**.data = data;  }  @Override  **public** String toString() {  **return** "BinaryTreeNode [data=" + data + "]";  }  }  **方法类：**  **package** tree;  **public** **class** BinaryTree {  **private** BinaryTreeNode root;  **public** BinaryTreeNode getRoot() {  **return** root;  }  **public** **void** setRoot(BinaryTreeNode root) {  **this**.root = root;  }  **public** **void** preRootTraverse(BinaryTreeNode t) {  **if** (t != **null**) {  System.***out***.print(t.getData());  preRootTraverse(t.getlChild());  preRootTraverse(t.getrChild());  }  }  **public** **void** inRootTraverse(BinaryTreeNode t) {  **if** (t != **null**) {  inRootTraverse(t.getlChild());  System.***out***.print(t.getData());  inRootTraverse(t.getrChild());  }  }  **public** **void** postRootTraverse(BinaryTreeNode t) {  **if** (t != **null**) {  postRootTraverse(t.getlChild());  postRootTraverse(t.getrChild());  System.***out***.print(t.getData());  }  }  **public** **int** countNode(BinaryTreeNode t) {  **if** (t != **null**) {  **return** 1 + countNode(t.getlChild()) + countNode(t.getrChild());  }  **return** 0;  }  **public** **int** countLeafNode(BinaryTreeNode t) {  **if** (t != **null**) {  **if** (t.getlChild() == **null** && t.getrChild() == **null**) {  **return** 1;  } **else** {  **return** countLeafNode(t.getlChild()) + countLeafNode(t.getrChild());  }  }  **return** 0;  }  }  **测试类：**  **package** tree;  **public** **class** MainTest {  **public** **static** **void** main(String[] args) {  // 二叉树初始化  BinaryTree bt = **new** BinaryTree();  bt.setRoot(**new** BinaryTreeNode(  **new** BinaryTreeNode(  **new** BinaryTreeNode(  **null**,  **new** BinaryTreeNode('G'),  'D'),  **null**,  'B'),    **new** BinaryTreeNode(  **new** BinaryTreeNode('E'),  **new** BinaryTreeNode(  **new** BinaryTreeNode('H'),  **null**,  'F'),  'C')  , 'A'));    System.***out***.print("前根遍历：");  bt.preRootTraverse(bt.getRoot());  System.***out***.print("中根遍历：");  bt.inRootTraverse(bt.getRoot());  System.***out***.print("后根遍历：");  bt.postRootTraverse(bt.getRoot());    System.***out***.println();    }  } |
| **实验结果：**  2.  3.  4. |
| **实验总结**  **可以通过先根序列和后跟序列判断根节点然后在中跟序列中一步一步分解出树的结构。** |