**《数据结构》实验报告（第5章）**

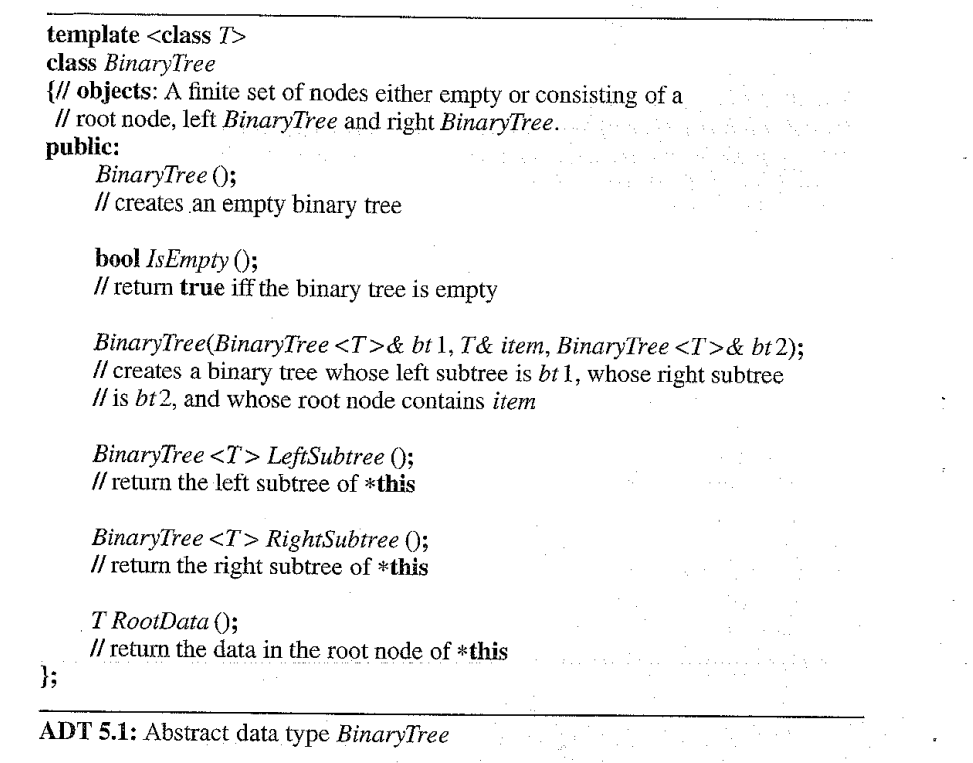
**学号：\_09020328\_\_\_\_ 姓名 ：\_ 王亮\_\_\_\_**

**实验题号： P267: 10; P272-273: 1, 2\_ 实验日期：\_2021.11.17\_\_\_**

**实验一**

**1．问题描述：**

定义完整的C++模板类二叉树，包括构造函数，拷贝构造函数，析构函数，四种迭代方法与相对应的前向迭代器，同时包括ADT5.1中函数。



**2．算法思想：**

copy():递归拷贝

deleteE():递归删除

拷贝函数：调用copy()

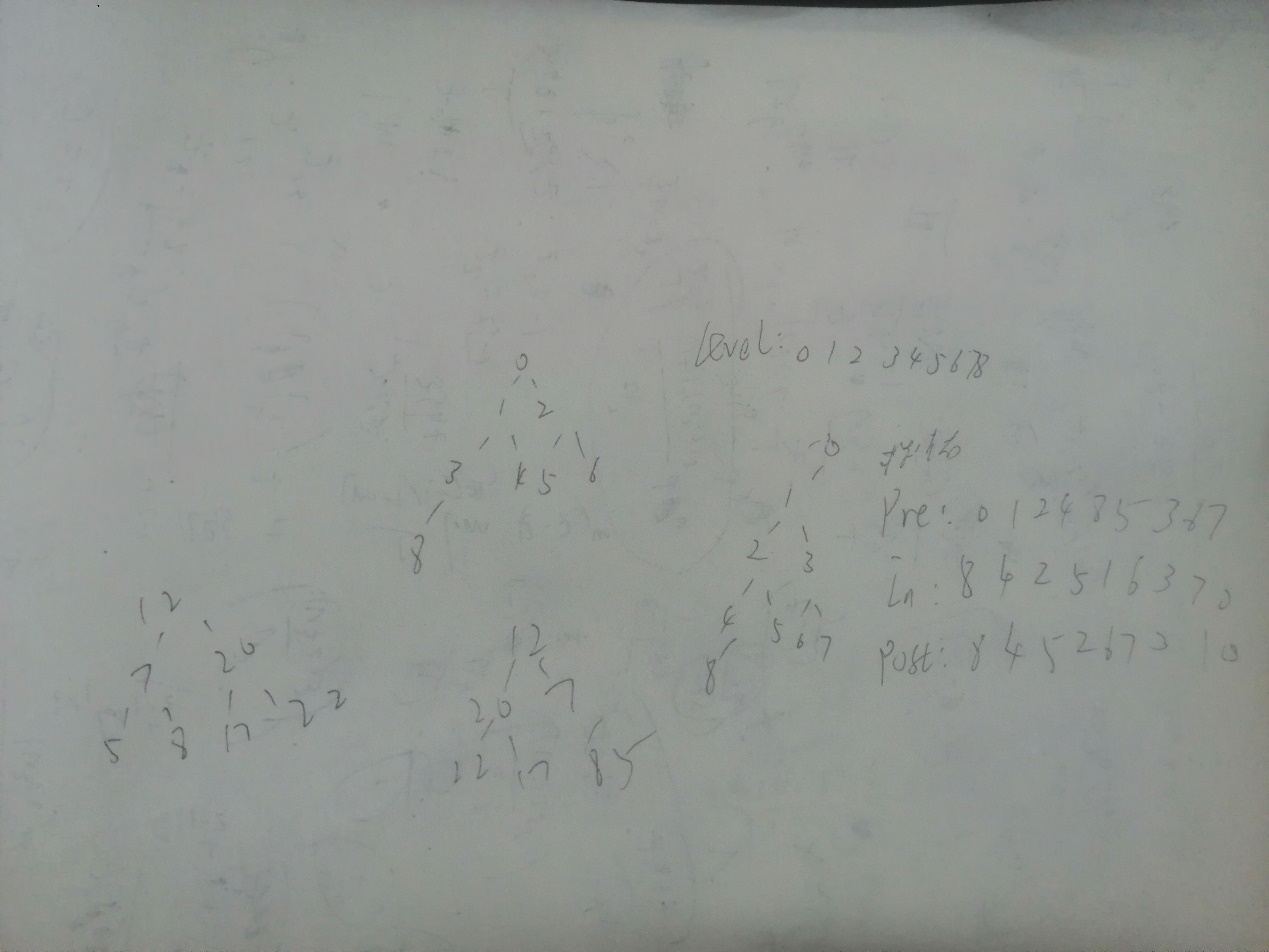
析构函数：调用deleteE()

四种遍历：均采用非递归

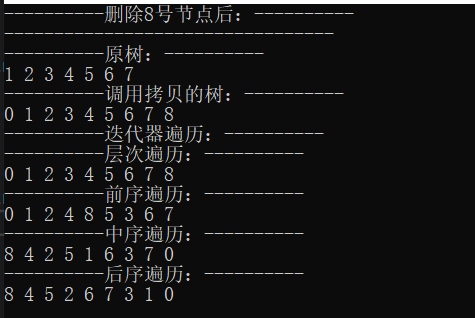
**3．功能函数：**

见上方

**4．测试数据：**

****

**5．测试情况：**



**6．实验总结：**

拷贝,析构,四种遍历均访问每个节点,时间复杂度均为O(n)

**7. 源代码：**

实验源程序清单

(1) As-Lab5.cpp

**实验二**

**1．问题描述：**

写一个能够计算二叉树中叶子节点个数的函数。考虑它的时间复杂度。

**2．算法思想：**

在实现层次遍历时，往往采用如下过程的循环：

队头元素出队

将队头元素的左右子节点入队

以一back作为队尾元素下标，对于任意

非叶节点而言，在左右子节点入队（不全为空）后，

此时的队尾back必然不等于原来的队尾back，

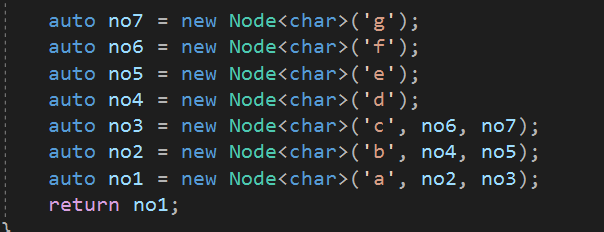
而对于叶节点，则有原来back==现在back，以此为据判断叶节点数。

**3．功能函数：**

pair<vector<T>, int> layerTraverse(Node<T>\* root)

层次遍历，顺带就求出叶节点个数。因为节点全部走一遍，时间复杂度O(n)

**4．测试数据：**





**5．测试情况：**

**对于上方完全二叉树共4个叶节点：**



**6．实验总结：**

层次遍历中加一条判断队头是否改变就可以做了。

时间复杂度O(n)

**7. 源代码：**

实验源程序清单

(1) As-LabP273.cpp

**实验三**

**1．问题描述：**

写一个SwapTree()，实现交换二叉树中每个节点的左右子树。

**2．算法思想：**

已知后序遍历遵循如下顺序：

左右根

在进行完一轮后序遍历之后，栈中会呈现左右根的排布，

此时依次将元素出栈，并将其左右子树进行交换即可实现。

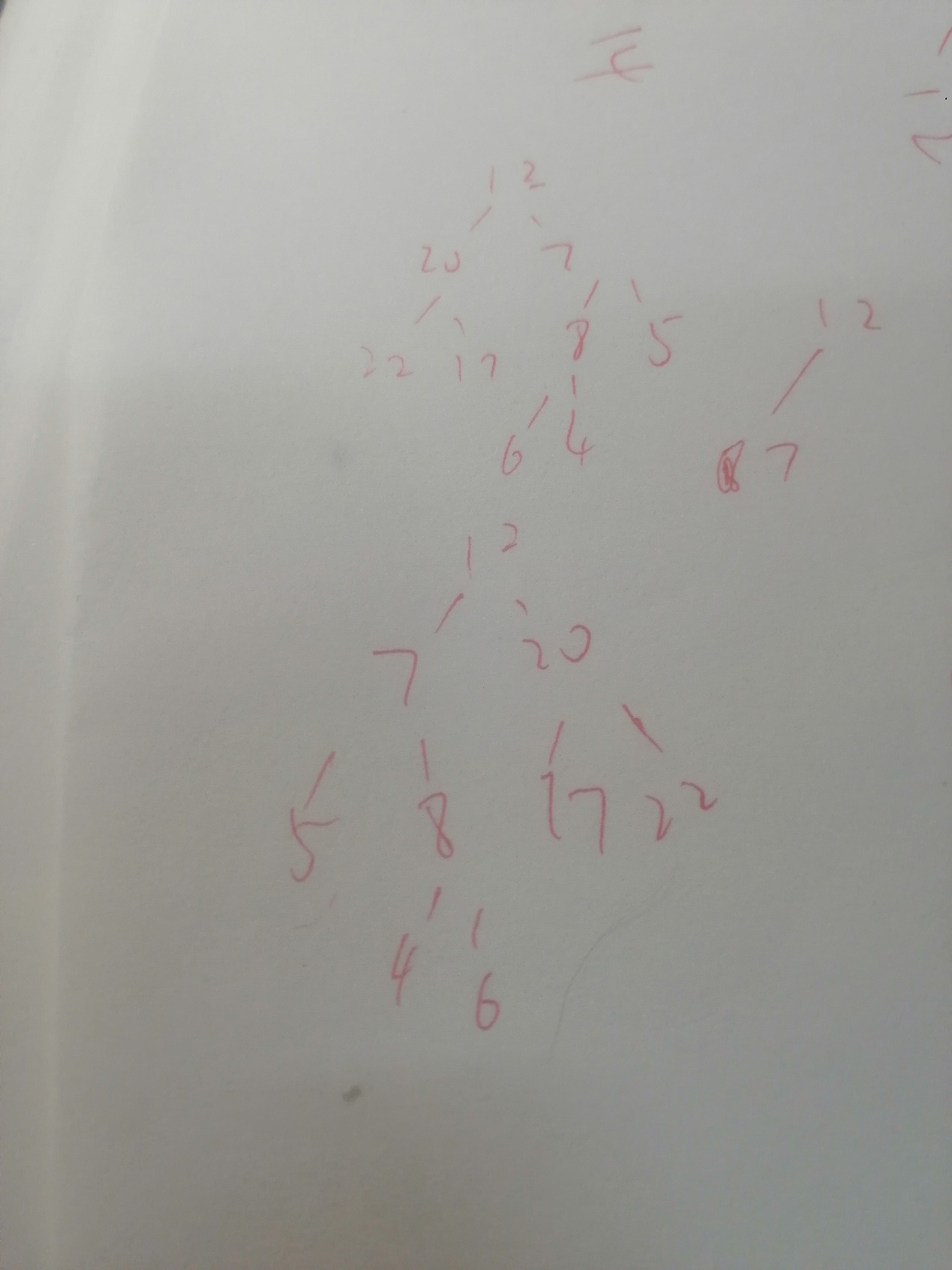
(此过程在后序遍历的过程中完成,具体在出栈阶段执行).

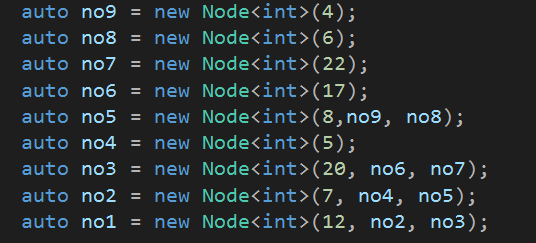
**3．功能函数：**

void swapTree(Node<T>\* root)

进行后序遍历,最后出栈访问时swap左右子树。

**4．测试数据：**





上图下1，2

**5．测试情况：**



**层次遍历验证正确**

**6．实验总结：**

一遍后序遍历,访问每个节点,时间复杂度O(n)

**7. 源代码：**

实验源程序清单

(1) As-labP274.cpp