**实验报告四**

1. **实验内容**

二叉树与二叉查找树的基本操作实现

1. **数据结构设计与思路分析**
2. **本题所采用的数据结构：**

二叉树与二叉查找树

**（二）问题分解与具体思路分析：**

（1）二叉树

本实验中包括的二叉树基本操作有：

1. InitBinTree():

功能：初始化二叉树

实现思路：根结点赋空并设定停止标记（stop flag）

1. CreatBinTree():

功能：构造二叉树

实现思路：递归方法构造二叉树（先创建根结点，然后创造左子树与右子树），每输入一个数就创造一个结点，遇到stop flag则赋空

1. PreOrder():

功能：先序遍历

实现思路：递归实现，在根结点不为空的条件下先访问根结点，接着自己调用自己，分别访问左子树与右子树

1. InOrder():

功能：中序遍历

实现思路：递归实现，在根结点不为空的条件下先访问左子树，接着访问根结点，最后自己调用自己访问右子树

1. PostOrder():

功能：后序遍历

实现思路：递归实现，在根结点不为空的条件下先访问左子树，接着访问右子树，最后访问根结点

1. LevelOrder():

功能：层次遍历

实现思路：借助队列，先将根结点入队，队不空出队，再将其左右结点（左右结点不为空的情况下）入队，以此类推

1. height():

功能：求树的深度

实现思路：递归求解，求出左子树的高度与右子树的高度，选择二者中较大值，最后加上根结点所在的一层。

1. width():

功能：求树的宽度

实现方式：借助队列，先将根结点入队。之后反复进行下面的三步操作直到队列为空：

遍历每一层：在每一轮循环中，首先获取当前队列的长度，这个长度即为当前层的节点数。

处理当前层：遍历当前层的每个节点，将每个节点的左右子节点（如果存在的话）加入队列。

更新宽度：用当前层的宽度更新最大宽度。

1. findDegreeTwoNodes():

功能：统计度数为2的结点个数

实现方式：采用递归的方法，先判断跟结点，然后自己调用自己，统计左子树中度数为2的结点数，接着统计右子树中度数为2的结点数。

1. findDegreeOneNodes():

功能：统计度数为1的结点个数

实现方式：采用递归的方法，先判断跟结点，然后自己调用自己，统计左子树中度数为2的结点数，接着统计右子树中度数为1的结点数。

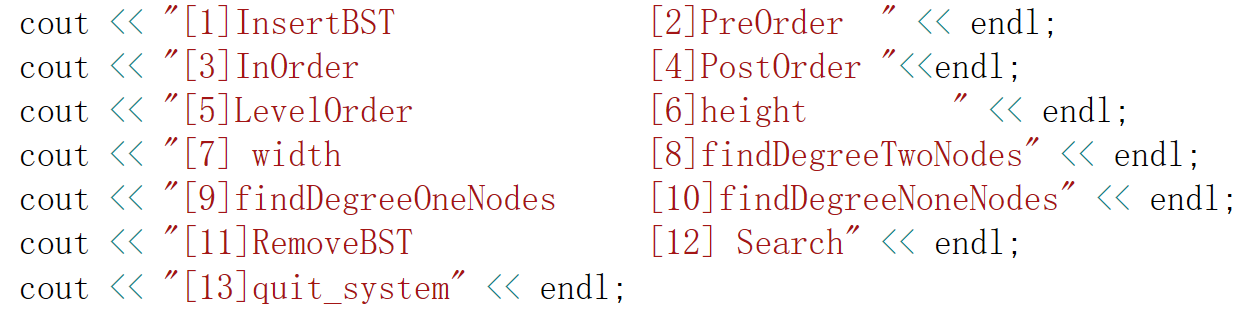
[11]findDegreeNoneNodes():

功能：统计度数为0的结点个数

实现方式：采用递归的方法，先判断跟结点，然后自己调用自己，统计左子树中度数为2的结点数，接着统计右子树中度数为0的结点数。

1. quit\_system():退出菜单
2. 二叉查找树

构造的菜单如下：



构造二叉查找树本实验中采取的方法为单个插入数据，也就是操作[1]

在二叉树操作的基础上增加的操作为：

1. InsertBST(BST &t, T x)：

功能：插入指定键

实现思路：递归实现，如果根结点为空，则创建结点后直接插入。如果插入键值小于根结点，则在其左树插入，大于则在其右插入。

1. void RemoveBST(BST&t,T key):

功能：删除指定键

实现思路：删除的结点会出现以下四种情况：（1）既没有左树也没有右树；（2）仅有左树；（3）仅有右树；（4）既有左树也有右树；应当对上述四种情况进行分类讨论分别解决。

1. BSTNode\* Search(BST t,T key):

功能：查找键

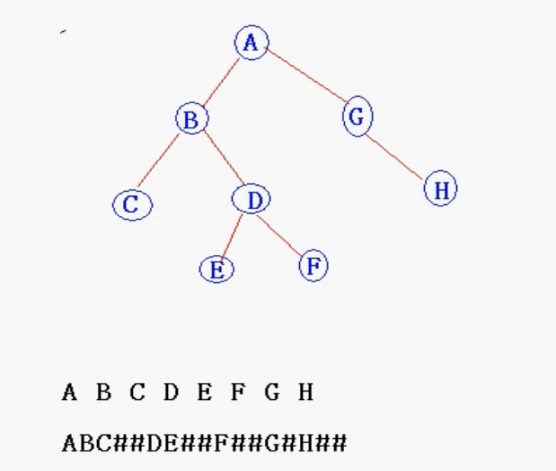
实现思路：采用递归的方法，先判断根结点是否为要寻找的结点，是直接返回；不是的话：如果所找的值比根结点的值小，则在左子树中寻找；否则则在右子树中寻找。

1. **实验结果与分析**

（一）二叉树基本操作

测试样例：

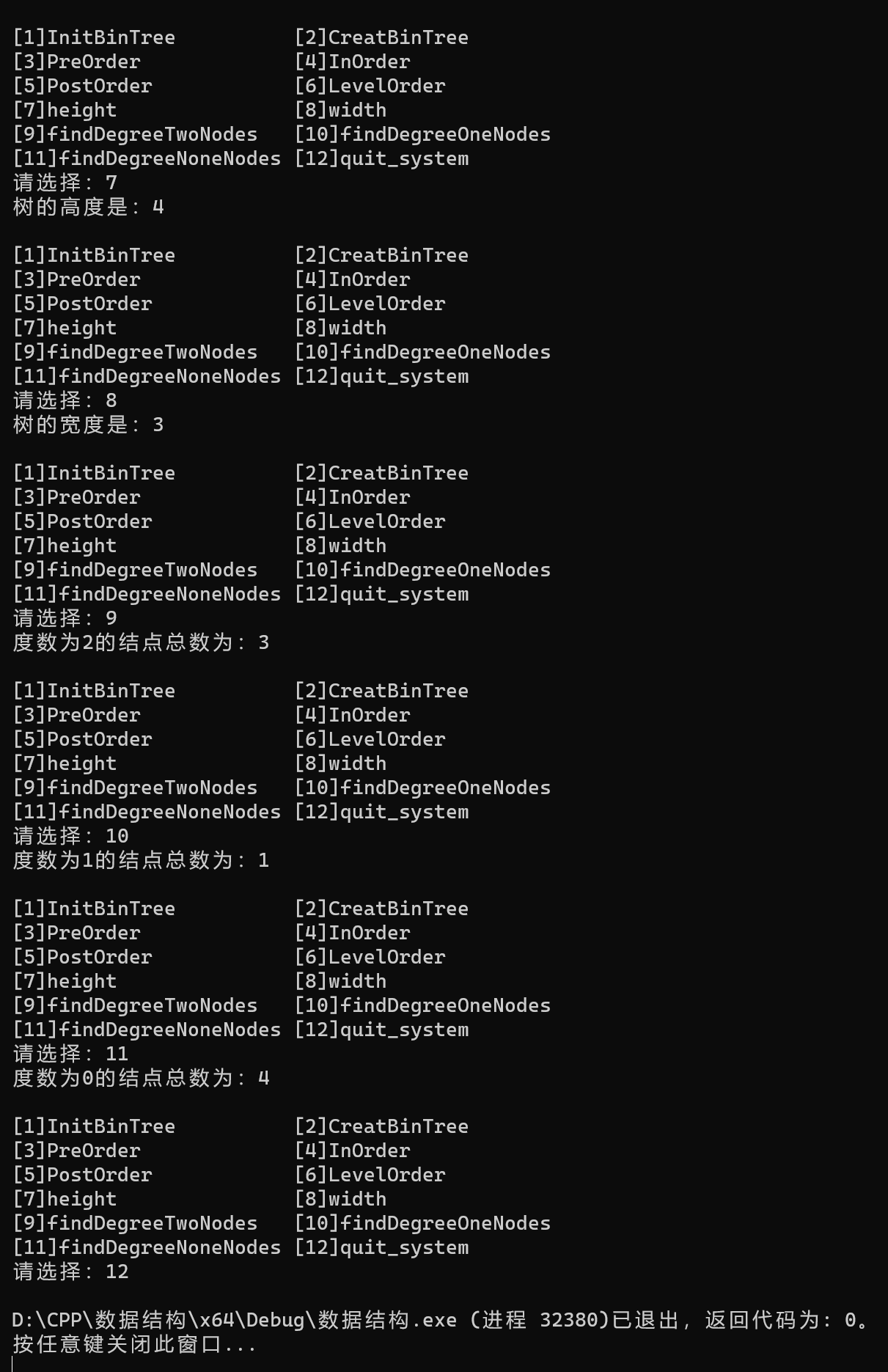
本测试案例中用来构造二叉树的序列为：ABC##DE##F##G#H##（按照前序序列来构造）



菜单驱动的程序结果截图如下（由于一张图片过长，将其分成两部分展示，图一为操作1-6，图二为操作7-12）：

通过与上方画出的二叉树比对，不难验证结果的正确性。

****

****

1. 二叉查找树

测试用例：利用{ 16,3,7,11,9,26,18,14,15 }这组数据构造二叉查找树，并根据菜单进行各功能实现。

实验结果如下：实验结果具有正确性





1. **实验小结**

通过本次实验，我加深了对二叉树以及二叉查找树基本操作的理解并能自己编写程序进行测试。同时也加深了对递归算法的理解，使得对递归算法的运用更加自如。