**《数据结构》实验报告3**

**学号：09021227 姓名 ：金桥**

**实验题号：3 实验日期：\_**2022.10.31**\_\_ 实验类型：** 必做

**1．问题描述：**描述实验内容和要求以及需要解决的问题。

**提示：**结合教师课堂讲授内容，仔细分析实验要求，对实验内容进行需求分析。

**准备工作：**

利用给定的中序和先序序列，构建一颗二叉树BTree，并实现构造空树、通过先序与中序序列构造、析构、四种遍历方式（中序遍历，先序遍历，后序遍历，层级遍历）、输出BTree、判断BTree是否为空、获取BTree树高、获取BTree结点数。

**实验任务：**

从文件获取数据并构建一颗二叉搜索树，并实现构造、析构、搜索、删除、插入功能。可以直接继承二叉树类。

**2．算法思想：**详细描述解决相应问题所需要的算法设计思想。

准备工作主要是使用二叉树实现。

实验任务使用二叉搜索树实现。

**这两者的主要算法都是递归，通过递归处理子树进而处理整棵树。**

**3．功能函数：**描述所设计的功能函数。如果有多个函数，需要描述它们之间的关系。

**主要包括四个类模板：**

BinaryTreeNode是二叉树的结点类型

BinaryTree是二叉树

KeyDataPair是为了继承二叉树而写的Key-Data对

BinarySearchTree是二叉搜索树

其中，BinaryTreeNode与KeyDataPair不再赘述，两者都是比较简单的类型。

BinaryTree包括如下函数：

BinaryTree(); 默认构造函数

BinaryTree(dataType\*, int); 通过一个序列进行构造

BinaryTree(dataType\*, dataType\*, int); 通过先序与中序构造

~BinaryTree(); 析构

void inOrder(void (\*)(dataType)); 中序遍历，传入操作的函数

void preOrder(void (\*)(dataType)); 先序遍历，同上

void postOrder(void (\*)(dataType)); 后序遍历，同上

void levelOrder(void (\*)(dataType)); 层级遍历，同上

bool isEmpty() const; 判空

int height() const; 获取树高

int size() const; 获取结点数量

void print() const; 输出树

void swapTree(); 交换所有节点的左右子节点

以下皆为上述函数对应的递归函数，实现上述函数的功能，上述函数大部分仅为入口。

BinaryTreeNode<dataType>\* constructRecursion(dataType\*, int, int, int);

BinaryTreeNode<dataType>\* constructRecursion(dataType\*, dataType\*, int, int);

void destructionRecursion(BinaryTreeNode<dataType>\*);

void inOrderRecursion(BinaryTreeNode<dataType>\*, void (\*)(dataType));

void preOrderRecursion(BinaryTreeNode<dataType>\*, void (\*)(dataType));

void postOrderRecursion(BinaryTreeNode<dataType>\*, void (\*)(dataType));

void printRecursion(std::string, BinaryTreeNode<dataType>\*, bool) const;

void swapTreeRecursion(BinaryTreeNode<dataType>\*);

**除去继承的函数，BinarySearchTree包括如下函数：**

BinarySearchTree(); 构造空树

BinarySearchTree(std::ifstream\*); 通过文件构造

~BinarySearchTree(); 析构

bool search(keyType) const; 搜索指定的Key

void makeEmpty(); 使用remove清空

void insert(keyType, dataType); 插入结点

void remove(keyType); 删除Key对应的结点

BinaryTreeNode<KeyDataPair<keyType, dataType>>\* removeRecursion(BinaryTreeNode<KeyDataPair<keyType, dataType>>\*&,

keyType);

BinaryTreeNode<KeyDataPair<keyType, dataType>>\* getLeftTreeMax(BinaryTreeNode<KeyDataPair<keyType, dataType>>\*) const; 获取左子树最大值

BinaryTreeNode<KeyDataPair<keyType, dataType>>\* getRightTreeMin(BinaryTreeNode<KeyDataPair<keyType, dataType>>\*) const; 获取右子树最小值

void updateHeight(); 更新树高

void updateHeightRecursion(BinaryTreeNode<KeyDataPair<keyType, dataType>>\*, int);

**更详细的描述见代码注释**

**4．测试数据：**设计测试数据，或具体给出测试数据。

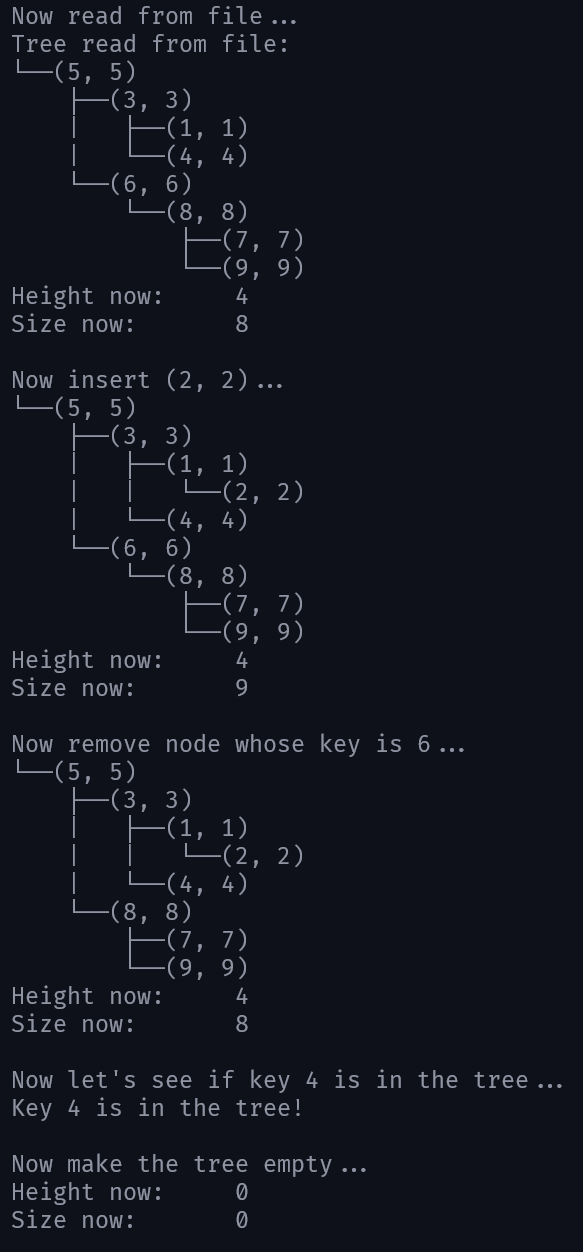
提示：要求测试数据能全面地测试所设计程序的功能。

使用BST依此插入如下结点：

5 3 6 8 9 1 4 7

之后测试各函数功能

**5．测试情况：**给出程序的测试情况，分析运行结果，显示实验结果截图。

****

**运行结果正常，符合预期。**

**6．实验总结：**写出实验过程中遇到的问题，以及问题的解决过程。分析算法的时间复杂度和空间复杂度，总结实验心得体会。

**复杂度分析：**

BST大部分操作（插入，删除，查询）单次时间复杂度为O(logn)，空间复杂度O(n)

**心得体会：**

对于C++写数据结构有了更深入的理解

**7. 源代码：**给出项目所有源程序清单。

建议：源程序中应有充分的注释，例如注释每个函数参数的含义、函数返回值的含义、函数的功能、主要语句段的功能，等等。

实验源程序清单

(1) BinaryTree.hpp

(2) BinarySearchTree.hpp

(3) main.cpp

(4) input.txt

…