**数据结构课程设计《二叉排序树》项目说明文档**

2253893 苗君文 软件工程

**1. 项目简介**

**1.1. 项目背景**

二叉排序树（Binary Search Tree，BST）是一种常见的二叉树数据结构，其特点是对于树中的每个节点，其左子树的所有节点的关键码都小于该节点的关键码，而右子树的所有节点的关键码都大于该节点的关键码。这种有序性质使得二叉排序树成为一种高效的数据结构，特别适用于搜索和排序操作。

二叉排序树的基本操作包括插入、删除和搜索。插入操作将新的节点按照关键码的大小有序地插入到树中；删除操作删除指定关键码的节点；搜索操作通过比较关键码在树中进行查找。这些基本操作使得二叉排序树可以用于实现各种高效的算法和数据结构，例如符号表、集合等。

**1.2. 项目要求**

本项目要求依次输入关键字并建立二叉排序树，实现二叉排序数的插入和查找功能。二叉排序树查找是指按照二叉排序树中结点的关系进行查找，查找关键自首先同根结点进行比较，如果相等则查找成功；如果比根节点小，则在左子树中查找；如果比根结点大，则在右子树中进行查找。这种查找方法可以快速缩小查找范围，大大减少查找关键的比较次数，从而提高查找的效率。

**1.3. 输入格式**

本项目要求用户首先输入要执行操作的选项。若要建立二叉排序树，则需要先输入关键码的个数以及各个关键码；若要插入元素，则需要输入要插入元素的关键码；若要查询元素，则需要输入要查询元素的关键码；若要删除元素，则需要输入要删除元素的关键码。

**1.4. 输出格式**

本项目针对输入有详细的输入错误处理，若用户输入错误，则程序会输出相关提示信息，并要求重新输入。而针对用户选择的操作，可以输出相应的操作结果并输出当前的二叉排序树的结果。

**2. 设计思路**

**2.1. 数据结构设计**

本项目选择二叉排序树作为数据结构，因为BST的核心特性是有序性。每个节点的左子树都包含小于节点值的元素，右子树包含大于节点值的元素。这种有序性使得对于排序和搜索操作更加方便高效。又由于有序性，BST的搜索操作非常高效。在每一步搜索中，可以通过比较当前节点的值与目标值的大小关系，确定是继续在左子树搜索还是在右子树搜索，从而在平均情况下实现较快的搜索。BST也能高效地完成插入和删除操作。在插入时，只需按照有序性找到合适的位置插入新节点，也就是插入的位置就是搜索失败的位置。在删除时，可以通过不同情况下节点的子树重组来维持有序性，实现高效的删除操作。

**2.2. 类设计**

**2.2.1. BSTNode类（二叉搜索树结点类）**

成员变量：

* key：节点的关键码。
* left：左子节点指针。
* right：右子节点指针。

成员函数：

* BSTNode()：构造函数，初始化节点。
* BSTNode(K x)：构造函数，初始化节点并设置关键码。

**2.2.2. BST类（二叉搜索树类）**

成员变量：

* root：指向根节点的指针。

成员函数：

* BST()：构造函数，初始化空树。
* ~BST()：析构函数，释放树的内存。
* Search(K x, BSTNode<K>\*& pr)：搜索关键码为 x 的节点，返回节点指针，并通过 pr 返回其父节点指针。
* createBSTree(K v[], int n)：根据数组 v 创建二叉排序树。
* printBSTree()：输出二叉排序树。
* Insert(K x)：插入关键码为 x 的节点。
* Remove(K& x)：删除关键码为 x 的节点。
* InOrder()：中序遍历输出二叉排序树。
* IsEmpty()：检查树是否为空。
* clearSubtree(BSTNode<K>\* node)：递归清空子树的内存。
* clearTree()：清空整个树的内存。

**2.3. 主程序设计**

程序先打印菜单，包括建立二叉排序树、插入元素、查询元素、删除元素以及退出程序的选项。创建一个BST类对象，该对象用于管理二叉排序树。进入一个循环，根据用户的选择执行对应的操作，循环中的关键步骤如下：

（1）选择建立二叉排序树：如果二叉排序树已经存在，输出提示信息。否则，接收用户输入的关键码个数和关键码数组，调用createBSTree函数建立二叉排序树，并输出当前树的状态。

（2）选择插入元素：接收用户输入要插入的关键码，调用Insert函数入元素，并输出当前树的状态。

（3）选择查询元素：接收用户输入要查询的关键码，调用Search函数查询元素，并输出查询结果。

（4）选择删除元素：接收用户输入要删除的关键码，调用Remove函数删除元素，并输出当前树的状态。

（5）选择退出程序：输出退出提示信息，结束循环，退出程序。

在程序结束前，还会释放动态分配的内存，确保没有内存泄露。

**2.4. 输入错误处理设计**

**2.4.1. 单个参数的输入错误处理**

函数dealInputError使用while循环不断尝试获取用户输入，直到输入满足要求。使用cin.fail()及min、max来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

此错误处理可以用在选择操作选项，输入关键码的个数，输入将插入元素的关键码，输入要查询的元素的关键码，输入要删除的元素的关键码。

**2.4.2. 二叉排序树相关的输入错误处理**

选择建立二叉排序树时，如果已建立过二叉排序树，则会提示用户选择其他操作。在输入关键码时，如果输入的个数超过用户规定的关键码个数则需要重新输入所有关键码，如果输入的关键码已经存在于树中，则会提示用户此关键码已存在于树中，但不会要求重新输入。选择插入元素时，同样输入的关键码已经存在于树中，则会提示相关信息。选择查询关键码时，树中无关键码，则会提示关键码不在树中，查询失败。选择删除元素时，若关键码不在树中，则会提示相关信息表明删除失败。

**3. 功能实现**

**3.1. Insert函数**

Insert函数完成向二叉排序树中插入新元素的操作。首先，函数需要检查根节点是否为空。如果根节点为空，说明这是一棵空树，直接将新元素作为根节点插入即可，插入成功后函数结束。

如果根节点不为空，进入搜索插入位置的阶段。从根节点开始，逐级比较新元素与当前节点的关键码大小，决定向左子树还是右子树移动。这一过程一直持续到找到一个空的位置，即叶子节点的左子节点或右子节点。

一旦找到空的位置，创建一个新的节点，将新元素存储在这个节点中。如果新元素小于当前节点的关键码，将新节点作为左子结点插入；如果新元素大于当前节点的关键码，将新节点作为右子结点插入。

插入完成后，返回插入成功的标志；如果新元素与树中已有的元素相等，说明插入失败，返回插入失败的标志。

**3.2. Remove函数**

Remove函数用于删除二叉排序树中含有指定关键码的节点。首先，函数需要检查根节点是否为空。如果根节点为空，说明这是一棵空树，删除操作无法执行，直接返回删除失败的标志。

从根节点开始，通过比较指定关键码与当前节点的关键码大小，逐级移动到要删除的节点。如果找到含有指定关键码的节点，执行删除操作；如果没有找到，返回删除失败的标志。

要删除的结点有三种情况：对于叶子节点（无子节点），则直接删除该节点。如果结点有一个子结点，则将父结点指向要删除节点的子节点，然后删除要删除的节点。如果结点有两个子结点，则找到该节点的中序遍历的直接前驱或直接后继节点，用其值覆盖要删除的节点的值，然后递归删除前驱或后继节点。

删除完成后，返回删除成功的标志；如果指定关键码在树中不存在，返回删除失败的标志。

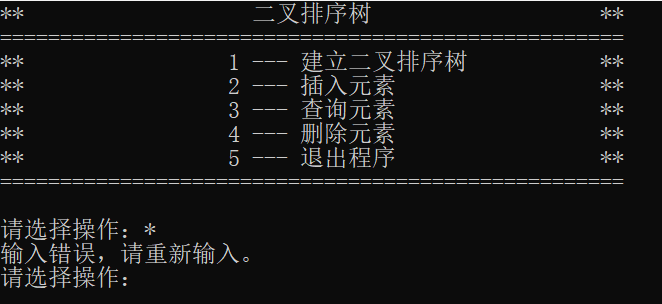
**3.3. printBSTree函数**

printBSTree函数用于输出二叉排序树的节点值，按照中序遍历的顺序输出。首先，函数需要检查根节点是否为空。如果根节点为空，说明这是一棵空树，直接返回。然后，中序遍历输出。采用递归的方式，对于每一个非空节点，递归调用printBSTree函数先输出左子树，再输出根节点，最后输出右子树。这样可以确保按照从小到大的顺序输出节点值。

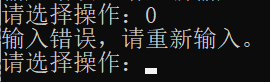
**4. 测试结果（包括边界测试）**

**4.1. 选择操作序号**

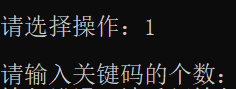
（1）输入的类型错误



（2）输入范围错误



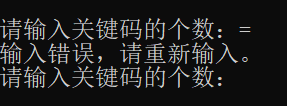
（3）输入正确即可进入下一环节



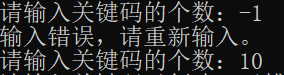
**4.2. 建立二叉排序树（选项1）**

**4.2.1. 输入关键码的个数**

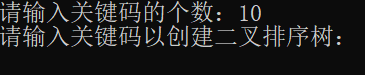
（1）输入的类型错误



（2）输入的范围错误

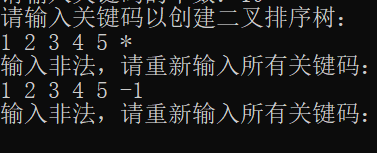


（3）输入正确即可进入下一环节

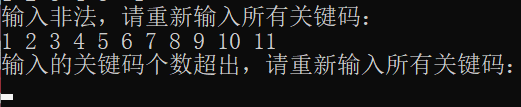


**4.2.2. 输入关键码以创建二叉排序树**

（1）输入的关键码中有非法字符（包括类型错误及范围错误）



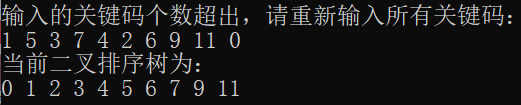
（2）输入的关键码的个数超过用户规定的个数



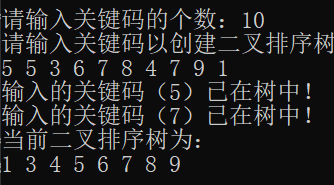
（3）输入正确即可输出二叉排序树的结果

由于程序中确保了输入错误后，即便之前有元素插入，也会将树清空，从而使合法的元素进入树中。

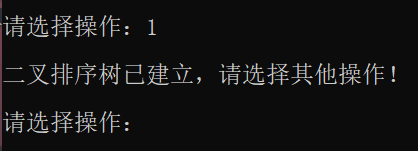
a. 用户输入的关键码中均不为树中的元素。



b. 用户输入的关键码中有已存在于树中的元素，则程序会输出相关提示信息。



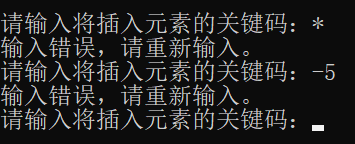
（4）若在已经建立二叉排序树后仍建立二叉排序树，会提示无需建立。



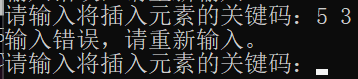
**4.3. 插入元素（选项2）**

**4.3.1. 输入将插入元素的关键码**

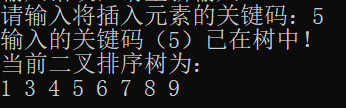
（1）输入的关键码非法（包括类型非法及内容非法）



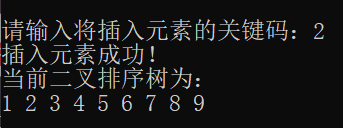
（2）输入的关键码个数超过一个



（3）输入的关键码已存在于二叉排序树中，则给出相关提示信息



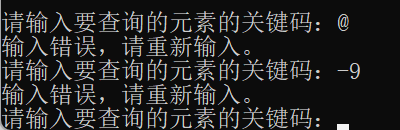
**4.3.2. 输入的关键码不存在于二叉排序树中，即可以插入。**



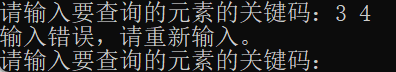
**4.4. 查询元素（选项3）**

**4.4.1. 输入要查询的元素的关键码**

（1）输入的关键码非法（包括类型非法及内容非法）



（2）输入的关键码个数超过一个

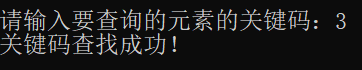


**4.4.2. 输入合法时，输出查找信息**

（1）查找的关键码不在树中，即查询失败。



（2）查找的关键码在树中，即查找成功



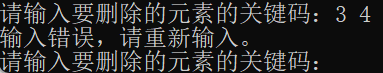
**4.5. 删除元素（选项4）**

**4.5.1. 输入要删除的元素的关键码**

（1）输入的关键码非法（包括类型非法及内容非法）

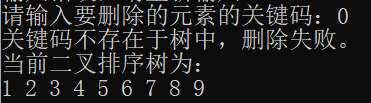


（2）输入的关键码个数超过一个

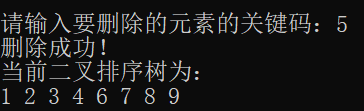


**4.5.2. 输入合法时，输出相关删除信息**

（1）删除元素的关键码不存在于树中，即删除失败。

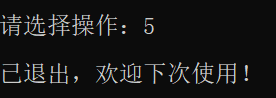


（2）删除元素的关键码存在于树中，即删除成功。



**4.6. 退出程序（选项5）**

输出退出提示。



**4.7. 程序测试结果总览（此处仅展示输入合法情况）**

