**西南石油大学实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目编号** | **001063000505** | **项目名称** | **查找及排序算法的综合应用** | **成绩** |  |
| **专业年级** | **计算机科学与技术** | | | **指导教师** | **胡卫东** |
| **姓名** | **王磊** | **学号** | **202231060435** | **实验日期** | **2023.06.03** | |

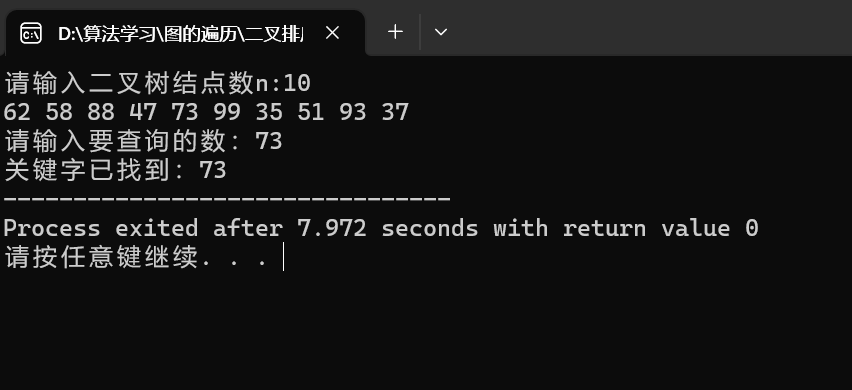
1. **实验目的**
2. 掌握顺序表查找中不同查找方法的查找思想，并能用C/C++语言实现。
3. 掌握树表查找中二叉排序树查找、平衡二叉树查找的查找思想，并能用C/C++语言实现。
4. 掌握Hash表查找中的查找思想，并能用C/C++语言实现。

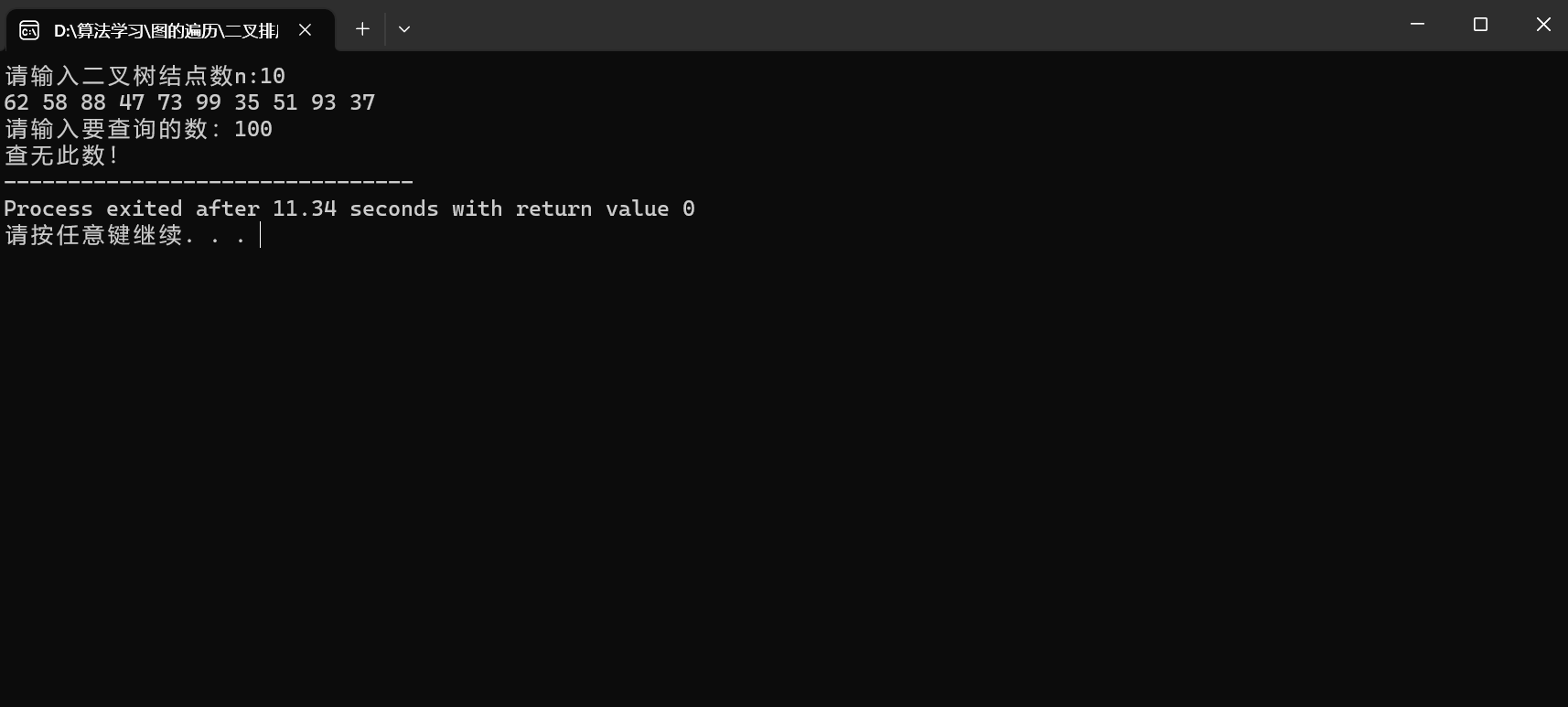
4、能够针对具体实际，灵活选用适宜的查找方法。

1. **实验工具**

PC微机，Windows，DOS，Turbo C或Visual C++

1. **实验步骤**
2. 定义二叉树结构
3. 创建查找函数searchBST函数，使用递归遍历循环二叉树进行寻找
4. 创建插入函数insertBST函数，在合适的位置插入数据，且使用searchBST函数查找确定位置和避免重复
5. 创建create函数构造二叉树，在其中调用insertBST函数有序插入数据（插入同时完成排序）
6. 最后打印出查找关键字数据
7. **实验结果**

****



1. **实验总结**

实验目标和意义：

目标：学习二叉排序树的原理和查找算法，并将其实现在代码中。

意义：二叉排序树是一种高效的查找结构，通过掌握其原理和实现方式，可以提高对数据的查找效率。

实验过程和方法：

创建二叉排序树：通过循环读取用户输入的关键字，并调用插入函数将关键字插入到二叉排序树中，从而创建了一个二叉排序树。

查找关键字：根据用户输入的关键字，调用查找函数在二叉排序树中进行查找。通过递归的方式，根据关键字和当前结点的大小关系，逐步缩小查找范围，直到找到目标结点或确定目标结点应该插入的位置。

实验收获和心得：

对二叉排序树的理解：通过实验，我深入理解了二叉排序树的概念和原理。二叉排序树的创建和查找过程相对简单，但可以提供较高的查找效率。

算法思维培养：在实验过程中，我运用递归思想解决问题，这培养了我对算法思维的训练和应用能力。

**1、二叉排序树查找**

**（1）问题描述**

查找是计算机操作中的一种重要应用技术，查找的方法有许多，不同的查找方法有不同的查找效率，而二叉排序树查找就是效率较高的查找方法之一。

所谓二叉排序树，就是指将原来已有数据根据大小构成一棵二叉树，二叉树中的所有结点数据满足一定的大小关系，所有左子树中的结点均比根结点小，所有右子树中的结点均比根结点大。

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，查找关键字首先同树根结点进行比较，如果相等则查找成功；如果比根结点小，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，大大减少了查找关键字的比较次数，从而提高了查找效率。

**（2）基本要求**

编程实现时，体现查找的全过程，即二叉排序树的创建、查找关键字的输入、查找关键字的查找、查找结果的输出等。

1. **算法描述**

**create: 用于创建二叉排序树。它首先从用户输入中获取结点数量，然后通过循环将每个输入的关键字插入到二叉排序树中。**

**searchBST: 用于在二叉排序树中查找关键字。它采用递归方式进行查找，如果找到了目标关键字，则返回1，并将指针p指向目标结点；如果未找到目标关键字，则返回0，并将指针p指向目标关键字应该插入的位置。**

**insertBST: 用于向二叉排序树中插入关键字。它首先调用searchBST函数来判断关键字是否已经存在，如果不存在则创建一个新结点，并根据大小关系将其插入到正确的位置。**

**在main函数中，首先调用create函数来创建二叉排序树，然后获取用户输入的关键字，并调用searchBST函数来查找关键字。最后根据查找结果输出相应的信息。如果找到了关键字，则输出关键字值；如果未找到关键字，则输出"查无此数"的提示信息。**

1. **算法实现——示例程序**

**#include <stdio.h>**

**#include <stdlib.h>**

**typedef struct tree {**

**int data;**

**struct tree \*lchild, \*rchild;**

**} \*BiTree, tree;**

**void create(BiTree \*T);//创建二叉树**

**int searchBST(BiTree T, int key, BiTree f, BiTree \*p);//查找**

**int insertBST(BiTree \*T, int key);//插入并排序**

**int main() {**

**BiTree T = NULL;**

**create(&T);**

**printf("请输入要查询的数：");**

**int key;**

**scanf("%d", &key);**

**BiTree p1;**

**int x = searchBST(T, key, NULL, &p1);**

**if (!x) {**

**printf("查无此数！");**

**} else {**

**printf("关键字已找到：%d", p1->data);**

**}**

**return 0;**

**}**

**void create(BiTree \*T) {**

**printf("请输入二叉树结点数n:");**

**int n;**

**scanf("%d", &n);**

**for (int i = 0; i < n; i++) {**

**int j;**

**scanf("%d", &j);**

**insertBST(T, j);**

**}**

**}**

**int searchBST(BiTree T, int key, BiTree f, BiTree \*p) {**

**if (!T) {**

**\*p = f;**

**return 0;**

**} else if (key == T->data) {**

**\*p = T;**

**return 1;**

**} else if (key < T->data) {**

**return searchBST(T->lchild, key, T, p);**

**} else {**

**return searchBST(T->rchild, key, T, p);**

**}**

**}**

**int insertBST(BiTree \*T, int key) {**

**BiTree p, s;**

**if (!searchBST(\*T, key, NULL, &p)) {**

**s = (BiTree)malloc(sizeof(tree));**

**s->data = key;**

**s->lchild = s->rchild = NULL;**

**if (!p) {**

**\*T = s;**

**} else if (key < p->data) {**

**p->lchild = s;**

**} else {**

**p->rchild = s;**

**}**

**return 1;**

**} else {**

**return 0;**

**}**

**}**