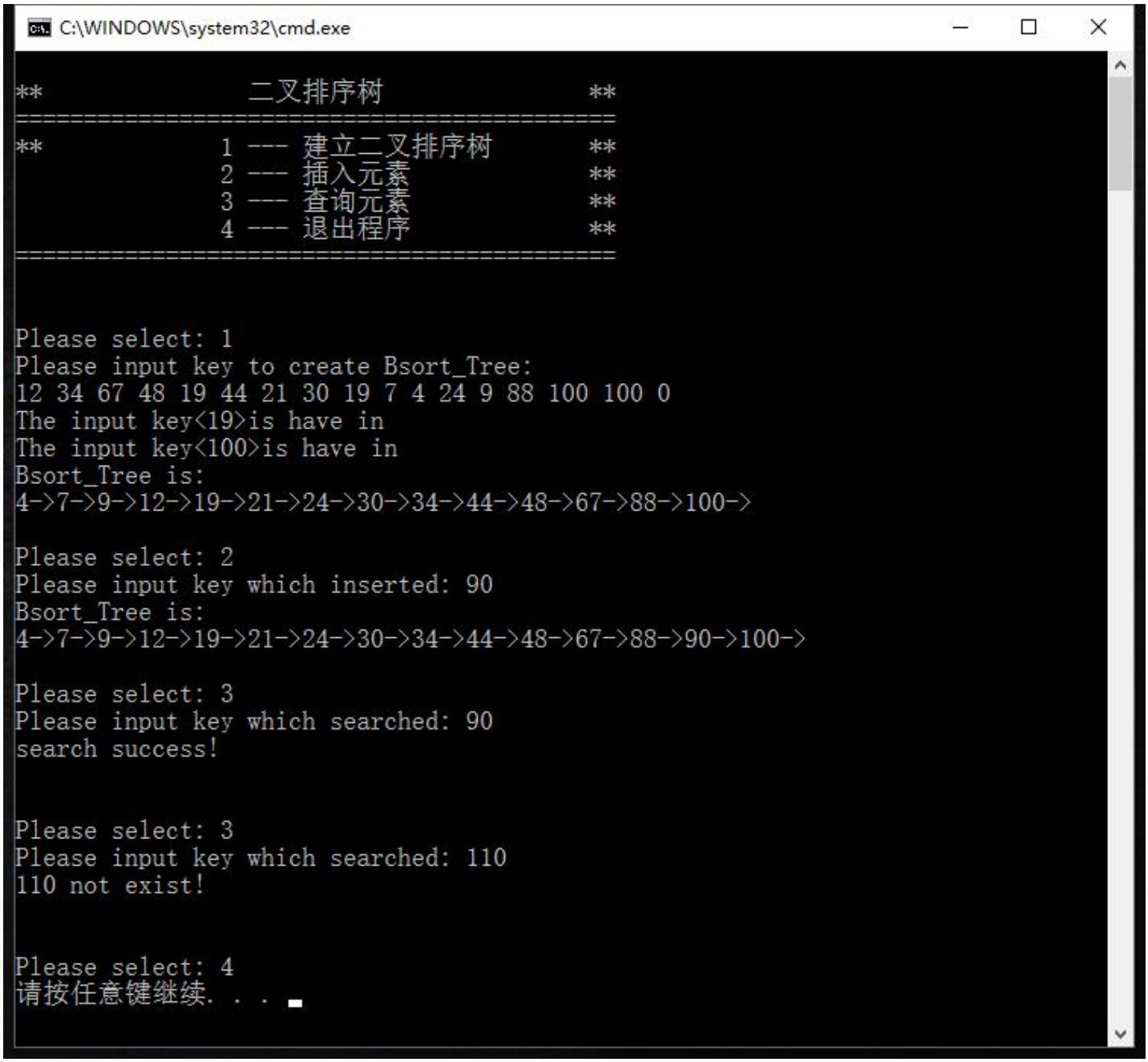
数据结构项目九文档

同济大学 软件学院 15级2班 1552651 王依睿

使用说明

操作手册



开始

运行程序后，显示二叉排序树系统的操作说明。

建立二叉排序树

- 用户选择操作1。
- 用户输入数列以创建一棵二叉排序树。（可重复）
- 输出重复的元素。
- 依据由小到大的顺序输出排序后的数列。（无重复）

```
Please select: 1
Please input key to create Bsort_Tree:
12 34 67 48 19 44 21 30 19 7 4 24 9 88 100 100 0
The input key<19>is have in
The input key<100>is have in
Bsort_Tree is:
4->7->9->12->19->21->24->30->34->44->48->67->88->100->
```

插入元素

- 用户选择操作2。
- 用户输入要插入的元素。
- 若已存在此元素，则提示用户此元素已存在。
- 依据由小到大的顺序输出插入元素后的数列。

```
Please select: 2
Please input key which inserted: 90
Bsort_Tree is:
4->7->9->12->19->21->24->30->34->44->48->67->88->90->100->
```

查询元素

- 用户选择操作3。
- 用户输入要查询的元素。
- 输出查询结果。

```
Please select: 3
Please input key which searched: 90
search success!
```

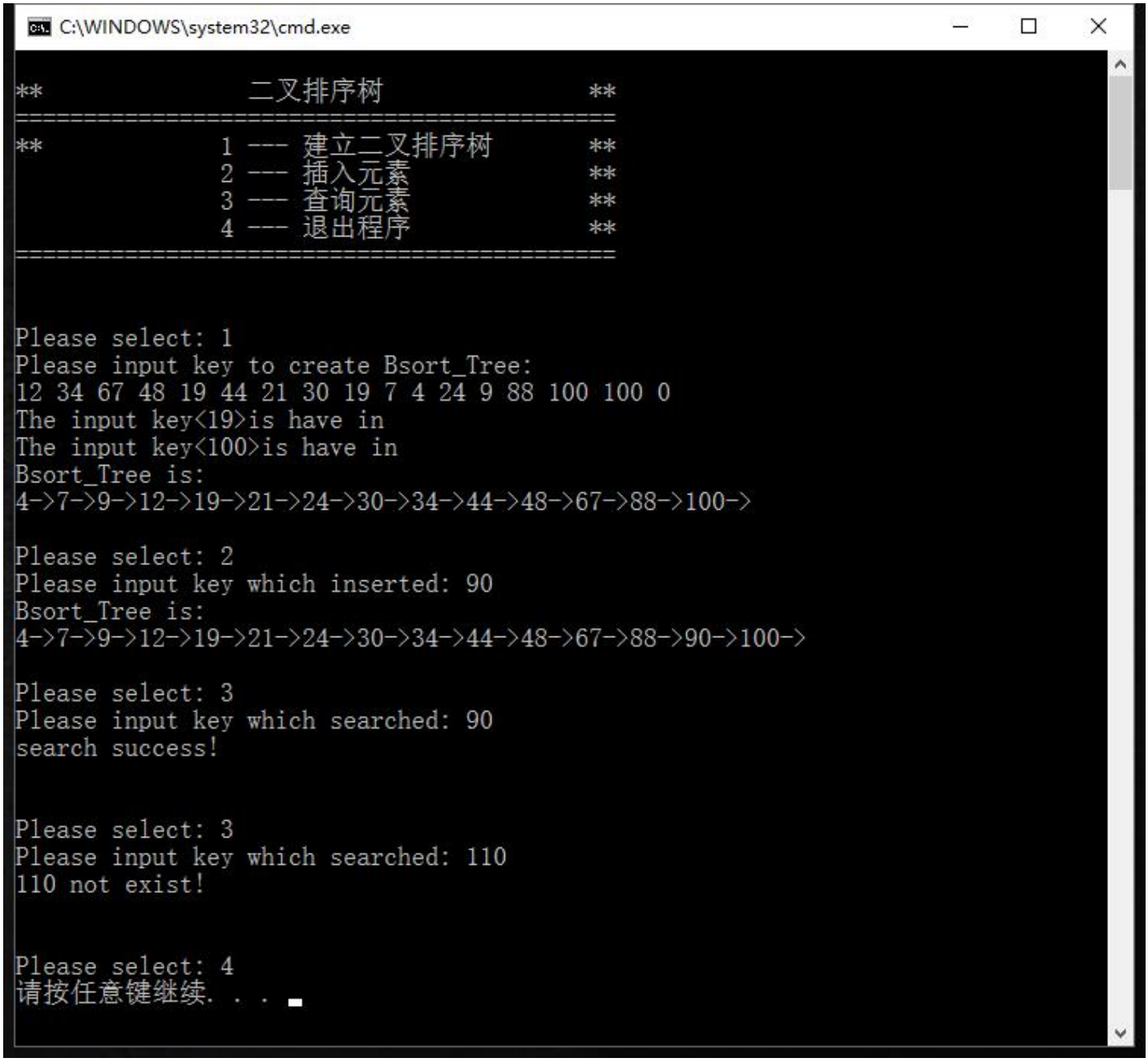
```
Please select: 3
Please input key which searched: 110
110 not exist!
```

退出程序

选择操作4。

```
Please select: 4
请按任意键继续. . .
```

整体预览



注意事项

- 用户输入用以创建一棵二叉排序树的数列中数字可重复。
- 二叉排序树中无重复的节点。

概述

> 依次输入关键字并建立二叉排序树，实现二叉排序树的插入和查找功能。

项目功能要求

二叉排序树就是指将原来已有的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，所有的左子树中的结点均比根结点小，所有的右子树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。

程序设计目的

实现一个利用二叉排序树对数列进行储存、插入和排列的简单程序，练习二叉排序树的编写实现，熟悉面向对象程序设计。

算法思路

以二叉排序树为数据结构，实现对数列进行储存、插入和排列。

数据结构

- 将二叉排序树节点封装成一个结构体。
- 将二叉排序树封装成一个类。

文件目录

- 9_1552651_wangyirui.cpp（主文件）
- tree_class.h（多叉树类）
- 9_1552651_wangyirui.exe（可执行文件）
- 9_1552651_wangyirui.pdf（项目文档）

函数接口

二叉排序树节点（BinNode）接口

二叉排序树节点类（BinNode）代码实现

```
struct BinNode {

    BinNode(int data) {

        data_ = data;
        lc = NULL;
        rc = NULL;
    }

    int data_;//关键码
    BinNodePosi lc = NULL;//左儿子
    BinNodePosi rc = NULL;//右儿子
};
```

二叉排序树（BST）接口

成员函数名	公有/私有性	功能	参数	返回值类型
~BST()	public	析构函数，通过调用函数Release，释放之前用new方法为节点分配的空间	/	/
BinNodePosi &SearchIn(BinNodePosi &v, const int e, BinNodePosi &hot)	pubic	递归：在子树v中查找关键码e	用于记录查找到与关键字匹配的节点指针v，待查找关键字e，用于记录上一层递归的的节点指针hot	若查找到，返回用于记录查找到与关键字匹配的节点指针BinNodePosi&，否则返回空指针
BinNodePosi &Search(const int e)	public	通过调用SearchIn算法，实现BST的标准接口Search()	待查找关键字e	若在二叉查找树中找到关键字e，返回用于记录查找到与关键字匹配的节点指针BinNodePosi&，否则返回空指针
		BST中插入		插入成功返回true，

bool Insert(const int e)	public	新节点e的过程	待插入的关键字e	若二叉查找树中已有待插入的关键字e，返回false
void Release()	public	通过调用BST的后序遍历删除BST中的节点	/	/
void TravIn_R(BinNodePosi x)	public	递归：BST的中序遍历（打印节点）	二叉查找树或其子树的根节点x	/
void TravPost_R(BinNodePosi x)	public	递归：BST的后序遍历（删除节点）	二叉查找树或其子树的根节点x	/
void PrintBST()	通过调用TravIn_R将BST中的关键码由小到大地打印出来	/	/	
bool empty()	public	判断BST是否为空	/	若为空，返回true，否则返回false

二叉排序树（BST）代码实现

```
class BST {

public:

    //析构函数，通过后序遍历删除节点
    ~BST();
    //递归：在子树v中查找关键码e
    BinNodePosi &SearchIn(BinNodePosi &v, const int e, BinNodePosi &hot);
    //通过调用SearchIn算法，实现BST的标准接口Search()
    BinNodePosi &Search(const int e);
```

```
    //BST中插入新节点e的过程
    bool Insert(const int e);
    //通过调用BST的后序遍历删除BST中的节点
    void Release();
    //递归：BST的中序遍历（打印节点）
    void TravIn_R(BinNodePosi x);
    //递归：BST的后序遍历（删除节点）
    void TravPost_R(BinNodePosi x);
    //通过中序遍历将BST中的关键码由小到大地打印出来
    void PrintBST();
    //判断BST是否为空
    bool empty() { return !root_; }

private:

    BinNodePosi root_ = NULL;//二叉查找树地根节点
    BinNodePosi hot_ = NULL;//在SearchIn函数中用于存储当前节点
};

#include<string>
#include"bst_class.h"

//析构函数，通过后序遍历删除节点
BST::~BST() {

    Release();
}

//递归：在子树v中查找关键码e
BinNodePosi &BST::SearchIn(BinNodePosi &v, const int e, BinNodePosi &hot) {

    //递归基：当节点为空或在节点中查找到关键码
    if (!v || (e == v->data_))
        return v;//当在二叉查找树中未查找到关键码时返回的是一个空指针

    hot = v;//一般情况：先记下当前节点

    //递归查找
    return SearchIn(((e < v->data_) ? v->lc : v->rc), e, hot);
}

//通过调用SearchIn算法，实现二叉查找树的标准接口Search()
//在二叉查找树中查找关键码e
BinNodePosi &BST::Search(const int e) {

    //返回目标节点的引用
    return SearchIn(root_, e, hot_=NULL);
}
```

```
//二叉查找树中插入新节点e的过程
bool BST::Insert(const int e) {

    //若不存在根节点，则将此节点作为根节点
    if (!root_) {
        auto root = new BinNode(e);
        root_ = root;
        return true;
    }

    //查找关键码e是否存在
    auto x = Search(e);
    //若存在，插入失败，返回
    if (x)
        return false;

    //若不存在，创造节点e
    x = new BinNode(e);

    //将节点e插入二叉查找树
    if (e < hot_->data_)
        hot_->lc = x;
    else
        hot_->rc = x;

    return true;//返回插入成功
}

void BST::Release() {

    if (root_)
        TravPost_R(root_);
    return;
}

//递归：二叉查找树的中序遍历(打印节点)
void BST::TravIn_R(BinNodePosi x) {

    if (!x)
        return;

    TravIn_R(x->lc);
    cout << x->data_ << "->";
    TravIn_R(x->rc);
    return;
}
```

```
//递归：BST的后序遍历（删除节点）
void BST::TravPost_R(BinNodePosi x) {

    if (!x)
        return;

    TravPost_R(x->lc);
    TravPost_R(x->rc);
    delete x;
    return;
}

//通过调用TravIn_R将二叉查找树中的关键码由小到大地打印出来
void BST::PrintBST() {

    TravIn_R(root_);
    return;
}
```

主文件代码实现

```
#include<string>
#include"bst_class.h"

int main() {

    cout << "\n**                二叉排序树                 **"
        << "\n============================================="
        << "\n**              1 --- 建立二叉排序树        **"
        << "\n                2 --- 插入元素              **"
        << "\n                3 --- 查询元素              **"
        << "\n                4 --- 退出程序              **"
        << "\n============================================="
        << endl;

    BST bst;

    while (true) {

        string operate;
        int num;

        cout << "\n\nPlease select: " << flush;
        cin >> operate;

        //若用户输入1，开始建立二叉排序树
```

```
        if (operate == "1") {

            cout << "Please input key to create Bsort_Tree:" << endl;
            while (cin >> num && num) {

                bool flag = bst.Insert(num);

                if (!flag)
                    cout << "The input key<" << num << ">is have in" << endl;
            }
            cout << "Bsort_Tree is:" << endl;
            if (bst.empty())
                cout << "The Bsort_Tree is empty!" << endl;
            else
                bst.PrintBST();
        }

        //若用户输入2，开始插入元素
        else if (operate == "2") {

            cout << "Please input key which inserted: " << flush;
            cin >> num;

            bool flag = bst.Insert(num);

            if (!flag)
                cout << "The input key<" << num << ">is have in" << endl;

            cout << "Bsort_Tree is:" << endl;
            if (bst.empty())
                cout << "The Bsort_Tree is empty!" << endl;
            else
                bst.PrintBST();
        }

        //若用户输入3，开始查询元素
        else if (operate == "3") {

            cout << "Please input key which searched: " << flush;
            cin >> num;

            bool flag = bst.Search(num);
            if (flag)
                cout << "search success!" << endl;
            else
                cout << num << " not exist!" << endl;
        }
```

```
        //若用户输入4，跳出循环，退出程序
        else if (operate == "4")
            break;

        else
            cout << "您输入的操作代码非法，请重新输入！" << endl;
    }
    return 0;
}
```