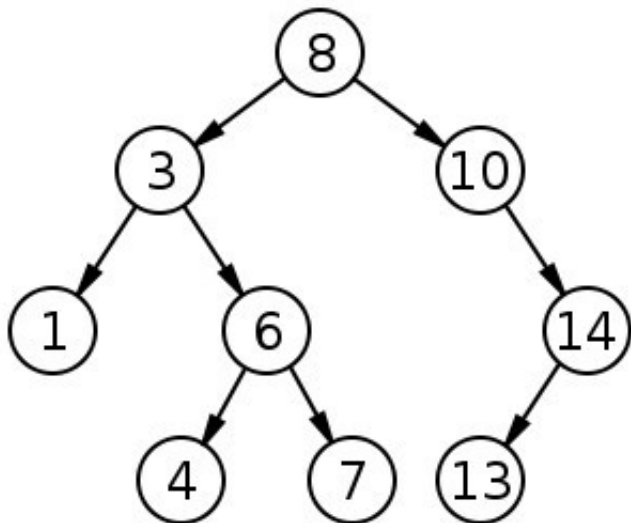


题目九 二叉排序树

一.设计思路

二叉排序树要求将所有数据整理为一棵二叉树，其任一节点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。

其原理可表示如下：



与一般的二叉树相比，其插入元素的规则较为特殊，在每次插入时，需要从根结点开始，不断比较，以决定其插入位置。因此非常适合用递归来实现。

而查找搜索操作本质上与插入操作是相同的，只是在搜索结束时将相应节点返回即可。

基于上述分析，需要构建一个二叉树类型数据结构，在其基础上，定制实现特殊的插入和查找操作。

二.数据结构实现

1.二叉排序树结点类型（TNode）

定义了结构体 `struct TNode` 作为二叉排序树的结点，存储其键值以及左右子女情况。

2.二叉排序树类型（familyTree）

定义了 `struct TNode` 的指针 `typedef struct TNode *BinTree` 作为二叉排序树类型，便于其他递归算法的实现，同时提供统一的接口。

3.元素键值类型（ElementType）

自定义了 `typedef int ElementType` 作为二叉排序树结点元素键值类型，暂定键值类型为 `int`，提供更好的封装性与可拓展性。

4.排序树结点成员

```
ElementType Data;    //节点关键字类型
BinTree Left;        //左子树
BinTree Right;       //右子树
```

`Date` 为 `ElementType` 类，代表元素键值类型。

`Left` 和 `Right` 均为 `BinTree` 类，即起左右子女均为二叉排序树类型。

5.插入函数

```
BinTree Insert( BinTree BST, ElementType X );    // 插入函数
```

向二叉排序树中插入节点，参数传入树的根节点以及被插入的元素键值，返回被操作节点的指针类型，便于递归操作。

6.查找函数

```
BinTree Find( BinTree BST, ElementType X );    //查找函数
```

在二叉排序树中查找元素，递归查找，返回被查找元素的指针。

7.二叉排序树输出函数

```
void printTree(BinTree BT);    //输出函数
```

打印二叉排序树，打印的顺序按照节点键值从小到大，内部调用中序遍历。

8.中序遍历函数

```
void InorderTraversal( BinTree BT );    //中序遍历
```

中序遍历二叉排序树，中序遍历的顺序就是节点键值大小排序。

7.辅助函数

```

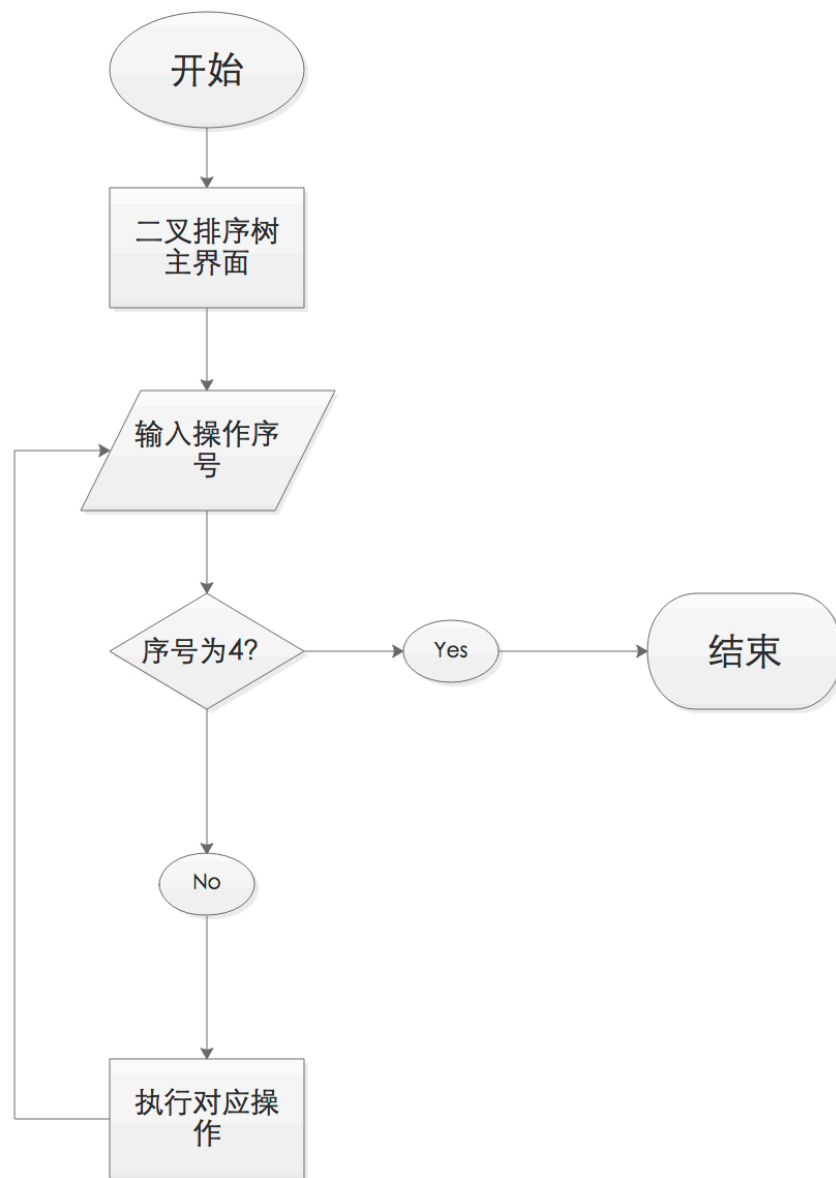
/*
 * 打印主菜单
 * */
void menu(){
    cout<<"**          二叉排序树          **\n";
    cout<<"===== \n";
    cout<<"**          1 --- 建立二叉排序树      **\n";
    cout<<"**          2 --- 插入元素              **\n";
    cout<<"**          3 --- 查询元素              **\n";
    cout<<"**          4 --- 退出程序              **\n";
    cout<<"===== \n";
}

```

打印主菜单。

三.系统实现

1.系统执行框架



首先调用 `void menu()` 函数初始化二叉排序树菜单界面。并进行辅助变量的初始化。

提示用户输入要进行的操作。

核心代码如下：

```

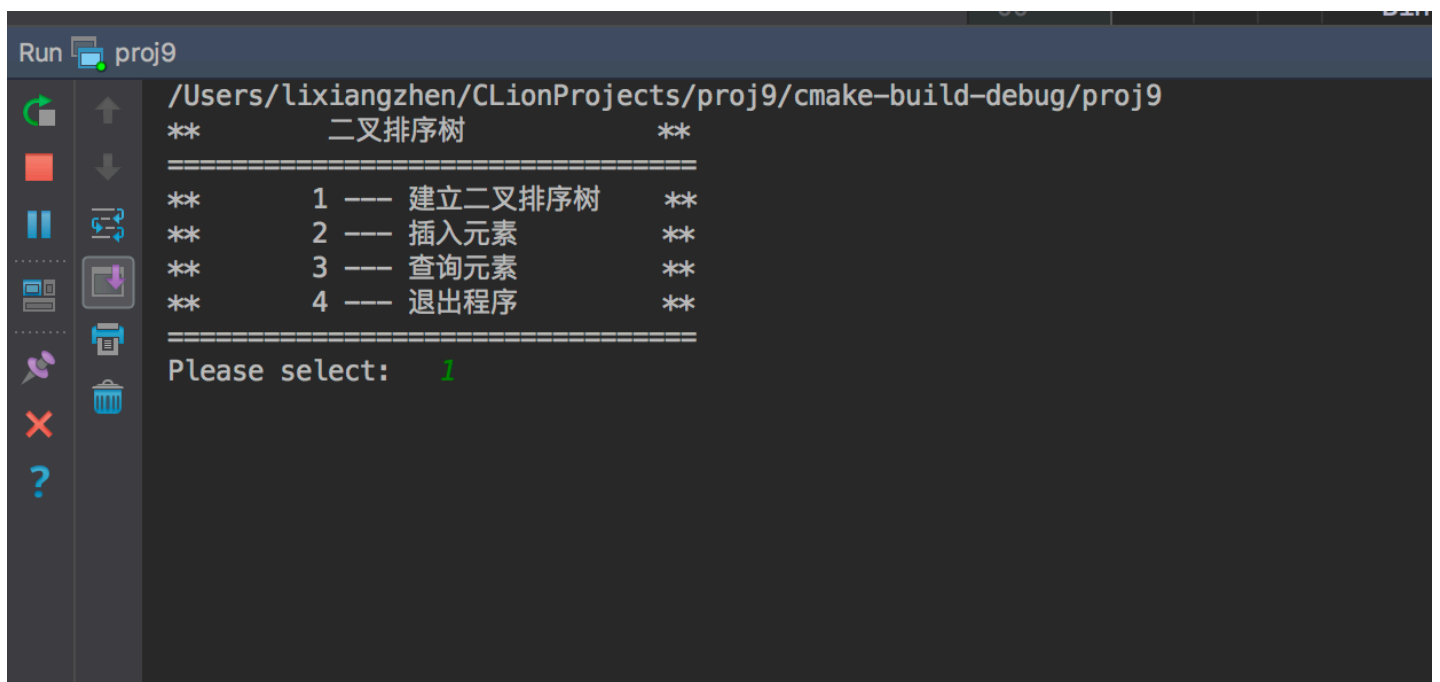
menu();      //打印主菜单

BinTree BST = NULL;      //创建新的空树
ElementType X;

stringstream ss;    // 字符流类型
string nodes;        // 以字符串形式读入一行二叉树节点
int temp;
int choice;           // 操作选择
cout<<"Please select:  ";
cin>>choice;

```

程序执行情况如下：



```

Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素      **
**      3 --- 查询元素      **
**      4 --- 退出程序      **
=====
Please select:  1

```

随后输入所选操作，利用 `switch--case` 语句判定不同操作，并跳转到相应位置。

核心代码如下：

```
while(choice != 4){          //循环输入所选操作
    switch (choice){
        case 1:
            cout<<"Please input key to create Bsort_Tree:\n";          // 读入字符
串所有节点

            getchar();
            getline(cin,nodes);
            ss<<nodes;

            while(ss>>X&&X != 0){          // 当读入元素为0时，插入停止
                BST = Insert(BST,X);
            }
            printTree(BST);
            break;
        case 2:
            cout<<"Please input key which is inserted:  ";          //插入单个节点
            cin>>temp;
            Insert(BST,temp);
            printTree(BST);
            break;
        case 3:
            cout<<"Please input key which is searched:  ";
            cin>>temp;
            BinTree found = Find(BST,temp);
            if(found == NULL)          // 处理元素不存在情况
                cout<<temp<<" not exist\n\n";
            else
                cout<<"search success!\n\n";
            break;
    }
    cout<<"Please select:  ";
    cin>>choice;
}
```

程序执行情况如下：

```
Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素      **
**      3 --- 查询元素      **
**      4 --- 退出程序      **
=====
Please select: 1
Please input key to create Bsort_Tree:
|
```

2.插入功能

核心代码如下：


```

/*
 * 向二叉排序树中插入节点
 * 参数传入树的根节点以及被插入的元素键值
 * 返回被操作节点的指针类型
 * 便于递归操作
 * */
BinTree Insert( BinTree BST, ElementType X ){
    if(!BST){          // 处理空树情况
        BST = new TNode;
        BST->Data = X;
        BST->Left = NULL;
        BST->Right = NULL;
    }
    else{              // 递归插入
        if(X > BST->Data)    // 根据节点键值选择插入位置
            BST->Right = Insert(BST->Right,X);
        else if(X < BST->Data)
            BST->Left = Insert(BST->Left,X);
        else
            cout<<"The input key"<<X<<" "<<"is have in!\n";    // 处理元素已经存在
的情况
    }

    return BST;
}

```

- 处理空树情况
- 根据节点键值选择插入位置
- 递归插入
- 提示元素已经存在

程序执行情况如下：

```
Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素      **
**      3 --- 查询元素      **
**      4 --- 退出程序      **
=====
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->

Please select:
```

3.查找功能

核心代码如下：

```

/*
 * 在二叉排序树中查找元素
 * 递归查找
 * 返回被查找元素的指针
 * */
BinTree Find( BinTree BST, ElementType X ){
    if(BST == NULL){
        return NULL;
    }
    if(BST->Data == X){

        return BST;
    }
    else if(BST->Data < X)
        return Find(BST->Right,X);
    else
        return Find(BST->Left,X);
}

```

- 类似于插入
- 根据节点键值选择下一步查找方向
- 返回结果
- 不存在返回空指针

程序执行情况如下：

```
Run proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素      **
**      3 --- 查询元素      **
**      4 --- 退出程序      **
=====
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->

Please select: 3
Please input key which is searched: 19
search success!

Please select:
```

Build finished in 2s 37ms (today 上午11:28)

4.中序遍历功能

核心代码如下：

```

/*
 * 中序遍历二叉排序树
 * 中序遍历的顺序就是节点键值大小排序
 * */
void InorderTraversal( BinTree BT ){
    if(!BT){
        return;
    }

    else{
        InorderTraversal(BT->Left);
        cout<<BT->Data<<"->";
        InorderTraversal(BT->Right);
    }
}

```

- 利用中序遍历下顺序就是节点键值大小排序的规律
- 遍历输出整棵树
- 输出方式按照格式化规范。

关联调用情况如下：

```

62
63
64         else{
65             cout<<"Bsort_Tree is:\n";
66             InorderTraversal(BT);
67             cout<<"\n\n";
68         }
69

```

5.二叉排序树输出功能

核心代码如下：

```
/*
 * 打印二叉排序树
 * 打印的顺序按照节点键值从小到大
 * 内部调用中序遍历
 * */
void printTree(BinTree BT){
    if(BT == NULL) {
        cout<<"Empty Tree\n";
        return;
    }
    else{
        cout<<"Bsort_Tree is:\n";
        InorderTraversal(BT);
        cout<<"\n\n";
    }
}
```

- 输出的接口
- 内部调用中序遍历
- 判断处理空树情况

关联调用情况如下：

```
        BST = Insert(BST,x);
    }
    printTree(BST);
    break;
case 2:
    cout<<"Please input key which is inserted: ";    //插入单个节点
    cin>>temp;
    Insert(BST,temp);
    printTree(BST);
    break;
case 3:
    cout<<"Please input key which is searched: ";
    cin>>temp;
```

四.测试

1.基本功能测试

测试用例

排序树键值： 3 22 101 55 43 6 1 65

建立二叉排序树

程序执行情况如下：

```
Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1  ---  建立二叉排序树      **
**      2  ---  插入元素              **
**      3  ---  查询元素              **
**      4  ---  退出程序              **
=====
Please select:  1
Please input key to create Bsort_Tree:
3 22 101 55 43 6 1 65 0
Bsort_Tree is:
1->3->6->22->43->55->65->101->
Please select:
```

插入元素

程序执行情况如下：


```
Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**          二叉排序树          **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素            **
**      3 --- 查询元素            **
**      4 --- 退出程序            **
=====
Please select: 1
Please input key to create Bsort_Tree:
3 22 101 55 43 6 1 65 0
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select: 2
Please input key which is inserted: 66
Bsort_Tree is:
1->3->6->22->43->55->65->66->101->

Please select:
```

查询元素

程序执行情况如下：

```

**      4 --- 退出程序      **
=====
Please select:  1
Please input key to create Bsort_Tree:
3 22 101 55 43 6 1 65 0
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select:  2
Please input key which is inserted:  66
Bsort_Tree is:
1->3->6->22->43->55->65->66->101->

Please select:  3
Please input key which is searched:  101
search success!

Please select:  3
Please input key which is searched:  21
21 not exist

Please select:

```

程序退出

程序执行情况如下：

```
Please input key which is inserted: 30
Bsort_Tree is:
1->3->6->22->43->55->65->66->101->

Please select: 3
Please input key which is searched: 101
search success!

Please select: 3
Please input key which is searched: 21
21 not exist

Please select: 4

Process finished with exit code 0
```

2.边界测试

插入重复元素

测试用例

排序树键值: 3 22 101 55 43 6 1 65

新插入元素: 101

程序执行情况如下:

```
Run proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树    **
**      2 --- 插入元素          **
**      3 --- 查询元素          **
**      4 --- 退出程序          **
=====
Please select:  1
Please input key to create Bsort_Tree:
3 22 101 55 43 6 1 65 0
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select:  2
Please input key which is inserted:  101
The input key<101>iS have in!
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select:
```

查询元素不存在

被查找元素：

程序执行情况如下：

```
**      4 --- 退出程序      **
=====
Please select:  1
Please input key to create Bsort_Tree:
3 22 101 55 43 6 1 65 0
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select:  2
Please input key which is inserted:  101
The input key<101>is have in!
Bsort_Tree is:
1->3->6->22->43->55->65->101->

Please select:  3
Please input key which is searched:  88
88 not exist

Please select:
```

空树

程序执行情况如下：

```
Run:  proj9  proj9
/Users/lixiangzhen/CLionProjects/proj9/cmake-build-debug/proj9
**      二叉排序树      **
=====
**      1 --- 建立二叉排序树      **
**      2 --- 插入元素      **
**      3 --- 查询元素      **
**      4 --- 退出程序      **
=====
Please select:  1
Please input key to create Bsort_Tree:
0
Empty Tree
Please select:
```

