项目文档

谈瑞

项目九：二叉排序树

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 专业班级：软件工程4班  电话：18936361545 | 学号：1452775  电子邮件：tanrui106@163.com | 课目：数据结构课程设计  个人网站：http://guitoubing.top |
|  |  |  |

目录

[项目简介 3](#_Toc501907638)

[项目概要 3](#_Toc501907639)

[项目功能及要求 3](#_Toc501907640)

[项目结构 4](#_Toc501907641)

[项目类的实现 5](#_Toc501907642)

[BSortTree类 5](#_Toc501907643)

[Passenger类 6](#_Toc501907644)

[代码分析 7](#_Toc501907645)

# 项目简介

## 项目概要

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。

## 项目功能及要求

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。

# 项目结构

本项目的基本架构是通过输入一串数字建立二叉搜索树，而后完成二叉搜索树的插入、删除、查找、替换、销毁等操作。二叉搜索树有一个很重要的特点，即对于任意一个结点（假设其左右子树都不为空），其左子树上所有结点的值都比右子树上的值要小，意即它是有一定顺序的存储方式。而且在中序遍历中，可得到其顺序排列。

本项目在基本要求外还实现了二叉树的前中后序的递归、非递归遍历方法，并在选择菜单提供了输出窗口，可随时输出二叉树。

# 项目类的实现

## BSortTree类

|  |  |
| --- | --- |
| 类成员 | 作用 |
| BSortTree(); | 构造函数 |
| ~BSortTree(); | 析构函数 |
| inline bool isEmpty()； | 判断二叉树是否为空 |
| void createBSortTree(); | 创建二叉树的入口函数 |
| bool setOperator(); | 选择对二叉树进行的操作 |
| void createTreeFormIn(); | 从输入端输入二叉树数值，这里的输入的是个文件，而文件内是1000个随机数 |
| bool insertTree(); | 往二叉树中添加结点 |
| bool searchTree(); | 在二叉树中寻找结点 |
| bool destroyTree(); | 销毁二叉树 |
| bool eraseTree(); | 在二叉树中删除结点 |
| void printTree(); | 输出二叉树 |
| void printTree(int op);  public成员 | 递归输出二叉树，op为选择输出方法（包括前序、中序、后序） |
| void printTreeNoRecursion(int op); | 非递归输出二叉树，op为选择输出方法（包括前序、中序、后序） |
| inline void printTreeInOrder  (BSortTreeNode \*\_current) | 中序遍历递归输出二叉树 |
| inline void printTreePreOrder  (BSortTreeNode \*\_current) | 前序遍历递归输出二叉树 |
| inline void printTreePostOrder  (BSortTreeNode \*\_current) | 后序遍历递归输出二叉树 |
| inline void printTreeInOrderNoRecursion  (BSortTreeNode \*\_current) | 中序遍历非递归输出二叉树 |
| inline void printTreePreOrderNoRecursion  (BSortTreeNode \*\_current) | 前序遍历非递归输出二叉树 |
| inline void printTreePostOrderNoRecursion  (BSortTreeNode \*\_current) | 后序遍历非递归输出二叉树 |
| BSortTreeNode \*getParent  (BSortTreeNode \*\_current) | 获取某结点的父亲结点 |
|  |  |