项目说明文档

数据结构课程设计

——二叉排序树

作 者 姓 名： 杨宇琨

学 号： 2252843

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 分析

## 1.1 背景分析

随着信息技术的发展，数据处理和存储需求不断增长。为了提高数据的检索效率和管理能力，开发了二叉排序树这样一种数据结构。二叉排序树是一种常见的数据结构，它能够高效地对数据进行插入和查找操作，尤其适用于有序数据的存储和检索。在实际的软件开发和数据处理中，二叉排序树被广泛应用于数据库索引、文件系统和搜索算法等领域。因此，对于二叉排序树的插入和检索功能的实现具有重要的实际意义。

## 1.2 功能分析

1. 二叉排序树的建立：根据输入的关键字，按照大小关系构建二叉排序树。对于每个新的关键字，根据其大小关系与已有的结点进行比较，插入到合适的位置以保持二叉排序树的有序性。

2. 二叉排序树的插入功能：用户可以输入新的关键字，程序将根据大小关系将其插入到已有的二叉排序树中的合适位置，以保持树的有序性。

3. 二叉排序树的查找功能：用户可以输入待查找的关键字，程序将按照二叉排序树中结点的大小关系进行查找。首先与根结点进行比较，然后根据比较结果在左子树或右子树中进行进一步的查找，直到找到目标结点或确认其不存在。

通过实现这些功能，用户可以方便地构建、插入和查找二叉排序树，从而加深对该数据结构的理解，提高数据的检索效率，满足实际应用中对数据处理和管理的需求。

# 2 设计

## 2.1 数据结构设计

该程序中使用了二叉树数据结构来存储和管理数据，其中每个节点包含一个整数值以及左右子节点指针。整体数据结构设计简洁明了，符合二叉树的基本特征。通过这样的设计，能够有效地支持插入和查找功能，同时保持了数据的有序性。

## 2.2 类结构设计

1. TreeNode类：表示二叉树的节点，包含数据值，左右子节点指针。通过构造函数初始化数据值，并提供了一个简洁的构造函数用于创建节点对象。

//Data structure of binary tree

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}

};

2. 主函数main：用于程序的入口和用户交互，包括创建二叉树、添加节点和查找节点等功能。

## 2.4 系统设计

1. 插入节点功能：insert函数通过递归的方式，在二叉排序树中插入新的节点，并根据节点值的大小关系选择合适的插入位置，以保持树的有序性。

2. 查找节点功能：search函数同样采用递归的方式，在二叉排序树中查找指定的节点，根据节点值的大小关系在左子树或右子树中进行查找，直到找到目标节点或确认其不存在。

3. 显示树的功能：treeDisplay函数使用中序遍历的方式，按照节点值的大小顺序输出整棵树，方便用户观察和验证树的结构。

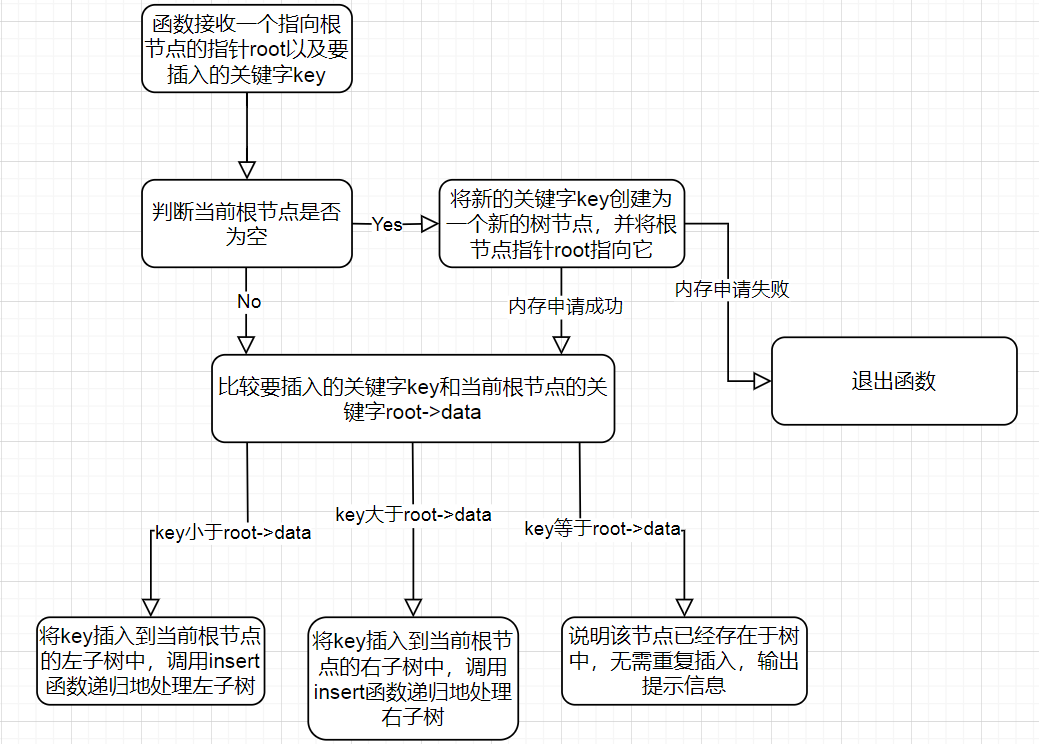
4. 用户交互功能：主函数main通过循环和交互界面，提供了创建二叉树、添加节点和查找节点等操作，使用户能够方便地使用二叉排序树的功能。

整体上，该程序通过合理的数据结构设计和功能实现，提供了一个简单而完整的二叉排序树操作系统，能够满足用户对二叉排序树的基本需求。

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图



### 3.1.2 插入功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:insert

Function:To insert a treenode

Input Parameters:The root of a tree and the key value

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void insert(TreeNode\*& root, int key)

{

//If finding the bottom of the tree,create a new treenode for the data

if (root == nullptr) {

root = new(std::nothrow) TreeNode(key);

if (root == NULL) {

std::cout << "叶子结点内存申请失败" << std::endl;

return;

}

}

else if (key < root->data)//If the key value is smaller than its root

insert(root->left, key);//Insert it in the left subtree of the root

else if (key > root->data)//If the key value is greater than its root

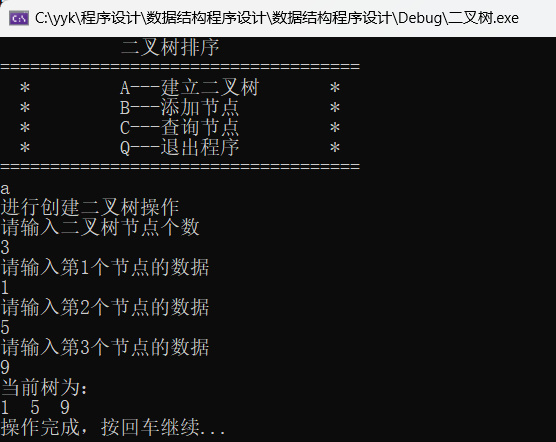
insert(root->right, key);//Insert it in the right subtree of the root

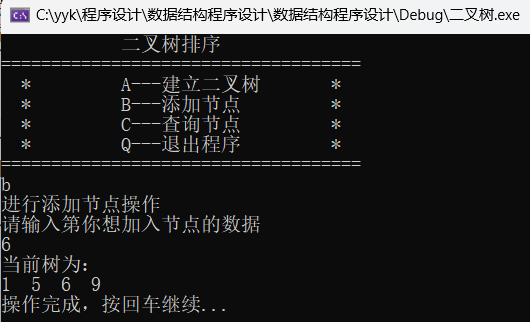
else

std::cout << "已有节点" << key << "在树中，将不重复插入！" << std::endl;

}

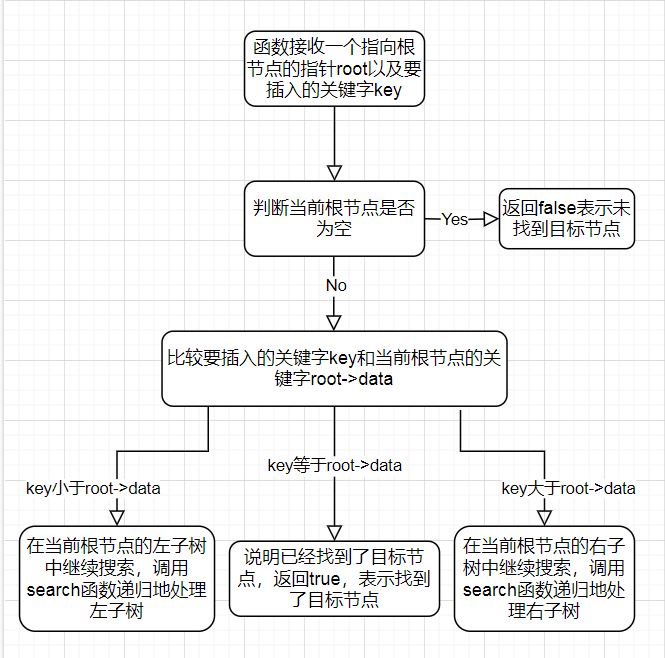
3.1.3 插入功能截屏示例





## 3.2 查找功能的实现

### 3.2.1 查找功能流程图



### 3.2.2 查找功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:search

Function:To search for a treenode

Input Parameters:The root of a tree

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool search(TreeNode\* root, int key)

{

if (root == nullptr)//Reaching the bottom

return false; //Fail to find

else if (key == root->data)//If the key equals to the root value

return true; //Found!

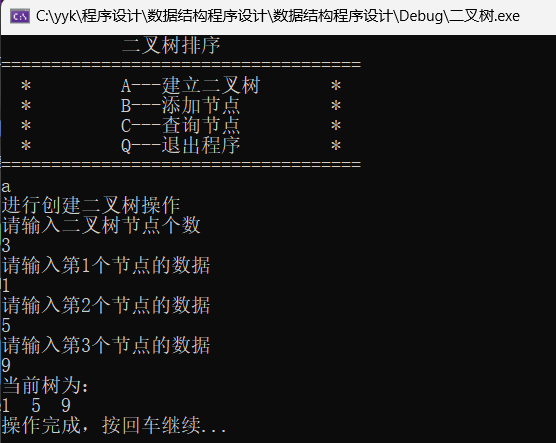
else if (key < root->data)//If the key is smaller than the root

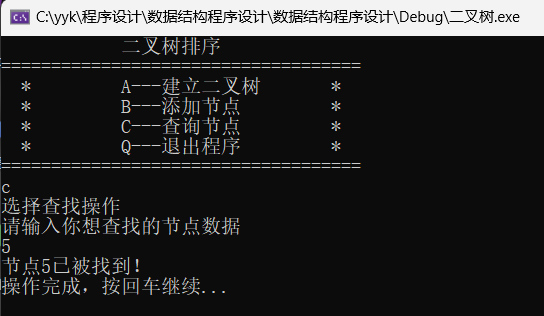
return search(root->left, key);//Search it in the left subtree

else //If the key is greater than the root

return search(root->right, key);//Search it in the right subtree

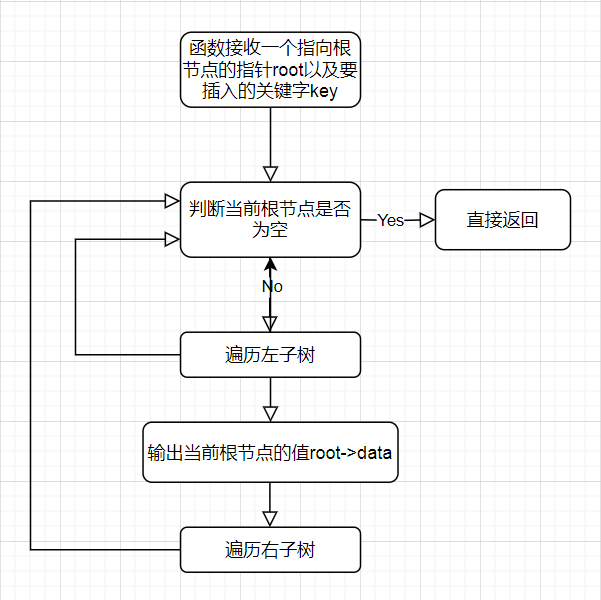
### }3.2.3查找功能截屏示例





## 3.3 显示功能的实现

### 3.3.1 显示功能流程图



### 3.3.2 显示功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:treeDisplay

Function:To display the tree in order

Input Parameters:TreeNode

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void treeDisplay(TreeNode\* root)

{

if (root == nullptr)

return;

else {

treeDisplay(root->left);

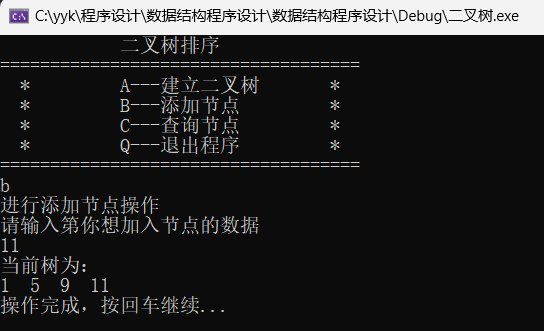
std::cout << root->data << " ";

treeDisplay(root->right);

}

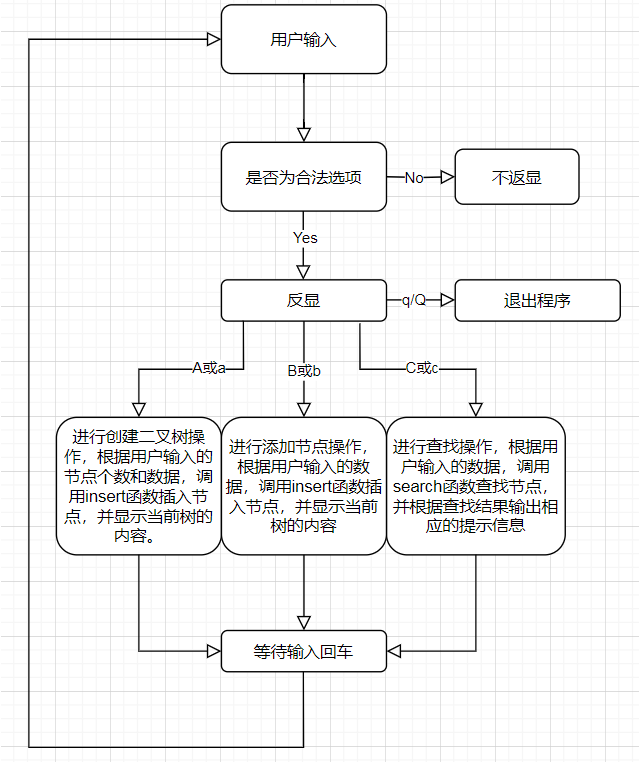
### }

### 3.3.3 查找功能截图示例



## 3.5 总体系统的实现

### 3.5.1 总体系统流程图



### 3.5.2 总体系统核心代码

TreeNode\* root = nullptr;//Create the root

//The loop can let the user execute the process repeatedly

while (true) {

//Clear our screen

system("cls");

//Display our list

menuDisplay();

char controller;//Create a variable to hold the choice we input

while (true) {

controller = \_getch();//\_getch can avoid displaying undesired input on the screen

if ((controller == 'Q' || controller == 'q') || (controller >= 'a' && controller <= 'c')|| (controller >= 'A' && controller <= 'C')) {

std::cout << controller << std::endl;//Display the choice on the screen

switch (controller)

{

case 'Q':

case 'q':

std::cout << "程序结束！" << std::endl;

return 0;

case 'A':

case 'a':

std::cout << "进行创建二叉树操作" << std::endl;

int n;

std::cout << "请输入二叉树节点个数" << std::endl;

n = input<int>(1, INT\_MAX);

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

int key;

std::cout << "请输入第" << i + 1 << "个节点的数据" << std::endl;

key = input<int>(INT\_MIN, INT\_MAX);

insert(root, key);//Add the node to the tree

}

std::cout << "当前树为：" << std::endl;

treeDisplay(root);

std::cout << std::endl;

std::cout << "操作完成，按回车继续..." << std::endl;

pause();

break;

case 'B':

case 'b':

std::cout << "进行添加节点操作" << std::endl;

int key;

std::cout << "请输入第你想加入节点的数据" << std::endl;

key = input<int>(INT\_MIN, INT\_MAX);

insert(root, key);//Add the node to the tree

std::cout << "当前树为：" << std::endl;

treeDisplay(root);

std::cout << std::endl;

std::cout << "操作完成，按回车继续..." << std::endl;

pause();

break;

case 'c':

case 'C':

std::cout << "选择查找操作" << std::endl;

int target;

std::cout << "请输入你想查找的节点数据" << std::endl;

target = input<int>(INT\_MIN, INT\_MAX);

if (search(root, target))

std::cout << "节点" << target << "已被找到！" << std::endl;

else

std::cout << "节点" << target << "未被找到" << std::endl;

std::cout << "操作完成，按回车继续..." << std::endl;

pause();

break;

default:

continue;

}

break;

}//end of if

}//end of while

}//end of while

### return 0;

### 3.5.3 总体系统截屏示例

