

# 数据结构课程设计报告

浙江工业大学 基础课程大型实验

实验学期：2019/2020(1)

实验题目：用户登录系统的模拟

学生姓名：陈昊天

学生学号：201806061201

学生班级：计算机科学与技术学院软件工程 1802 班

提交日期：2019 年 12 月 13 日

## 实验题目和要求

本节所有内容与课程设计任务书保持一致

### 问题描述

在登录服务器系统时，都需要验证用户名和密码，如 `telnet` 远程登录服务器。用户输入用户名和密码后，服务器程序会首先验证用户信息的合法性。由于用户信息的验证 频率很高，系统有必要有效地组织这些用户信息，从而快速查找和验证用户。另外，系统也会经常会添加新用户、删除老用户和更新用户密码等操作，因此，系统必须采用动态结构，在添加、删除或更新后，依然能保证验证过程的快速。请采用相应的数据结构模拟用户登录系统，其功能要求包括**用户登录、用户密码更新、用户添加和用户删除**等。

### 基本要求

1. 要求自己编程实现二叉树结构及其相关功能，以存储用户信息，**不允许使用标准模板类的二叉树结构和函数**。同时要求根据二叉树的变化情况，进行相应的平衡操作，即 AVL 平衡树操作，**四种平衡操作都必须考虑**。测试时，各种情况都需要测试，并附上测试截图。
2. 要求采用类的设计思路，不允许出现类以外的函数定义，但允许友元函数。主函数中只能出现类的成员函数的调用，不允许出现对其它函数的调用。
3. 要求采用多文件方式：**.h** 文件存储类的声明**.cpp** 文件存储类的实现，主函数 **main** 存储在另外一个单独的**.cpp** 文件中。如果采用类模板，则类的声明和实现都放在**.h** 文件中。
4. 不强制要求采用类模板，也不要求采用可视化窗口；要求源程序中有相应注释。

5. 要求测试例子要比较详尽，各种极限情况也要考虑到，测试的输出信息要详细易懂，表明各个功能的执行正确。
6. 建议采用 **Visual C++ 6.0** 及以上版本进行调试；

## 实现提示

1. 用户信息(即用户名和密码)可以存储在文件中，当程序启动时，从文件中读取所有的用户信息，并建立合适的查找二叉树。
2. 验证过程时，需要根据登录的用户名，检索整个二叉树，找到匹配的用户名，进行验证；更新用户密码时，也需要检索二叉树，找到匹配项后进行更新，同时更新文件中存储的用户密码。
3. 添加用户时，不仅需要在文件中添加，也需要在二叉树中添加相应的节点；删除用户时，也是如此。

## 运行结果要求

要求能够实现用户登录验证、添加用户、删除用户和更新用户密码功能，实验报告要求有详细的功能测试截图。

## 考核要求

要求程序能正常运行，全面完成题目要求

## 题目难度

难，成绩等级高

## 实验环境

用户登录系统的模拟在 JetBrains Clion 下进行开发，操作系统为 Window10 64 位操作系统。

## 实验进度说明

2019 年 11 月 25 日 完成 Beta1 版本

2019 年 11 月 27 日 完成 Beta3 版本

2019 年 12 月 6 日 完成 Beta5 版本 修复了删除导致程序崩溃的 Bug

2019 年 12 月 12 日 编写实验报告

## 设计思路

### 系统总体设计

在这里介绍整个内核的设计逻辑和主要的操作逻辑

本实验的基于 AVL 平衡树的登录系统（以下简称本系统）主要采用了面向对象、分而治之的思想进行整个系统的设计。按照系统的功能分块，我将整个系统分为 **UI 层**和**数据层**，并使用**消息队列**的方法实现 UI 层和数据层的交互。为了让该系统的界面交互更加友好，我们引入了**分角色**和**菜单**的概念，让用户的操作更加方便便捷，同时，也体现了编程人员一定的编程水平。

### 系统功能设计及实现

#### UI 层（用户界面）设计

UI 层是本系统面向用户的窗口，基于用户角色的区别，我们将用户分为**普通用户**和**管理员**两种角色，同时，还引入了**菜单**，和**页面**的概念。下面将介绍这个 UI 层的设计结构。



```
AVLTree x
C:\Users\cht\Program\C++\AVLTree\cmake-build-debug\AVLTree.exe
欢迎来到用户登录系统(AVL实验) VERSION:beta 5
page = main
*****
1. 导入                2. 导出                3. 转到普通用户界面
4. 转到管理员界面    5. 退出程序
*****
> 请选择菜单中的一项:
```

图 1 UI 用户设计

整个 UI 层设计的主要类为 **AppBase,App,Context,Menu** 以及辅助类 **consolecolorhelper,Log**。我将先介绍各个类的功能，然后讲解这个 UI 层交互的原理。

#### AppBase

这个类是整个 UI 层的核心，为 CUI 提供了基础的输入和输出分装，关键函数的功能和介绍如下。其中 **onTip()**和 **onLoad()**和 **onHandleInput()**都需要进行重写，以完善应用的功能。

函数签名	功能
string onTip()	显示提示字符串信息，由 handleInput()函数来调用，需要进行重写。
void onLoad()	在初始化的时候会被调用的函数，需要进行重写。
bool onHandleInput(string command)	处理控制字符信息，由 handleInput()函数来调用，需要进行重写。 @return 是否继续执行，若返回 false，则表示应用程序终止运行。
Context& getContext()	获取当前的上下文信息
<i>private:</i> void handleInput()	将系统的 cin 封装起来，用于和用户进行交互。

表 1 AppBase 的类的设计

我们看一下 handleInput()的代码，它是将整个应用运行起来的核心。

*//将系统的cin 封装起来，用于和用户进行交互*

```
void handleInput()
{
    while (true)
    {
        auto tip = onTip();
        cout << consoleforecolor::seablu << "> " << tip << ":" <<
consoleforecolor::normal;
        string cmd;
        getline(cin, cmd);
        if (!onHandleInput(cmd))
        {
            break;
        }
    }
}
```

下面介绍整个系统的调用逻辑。

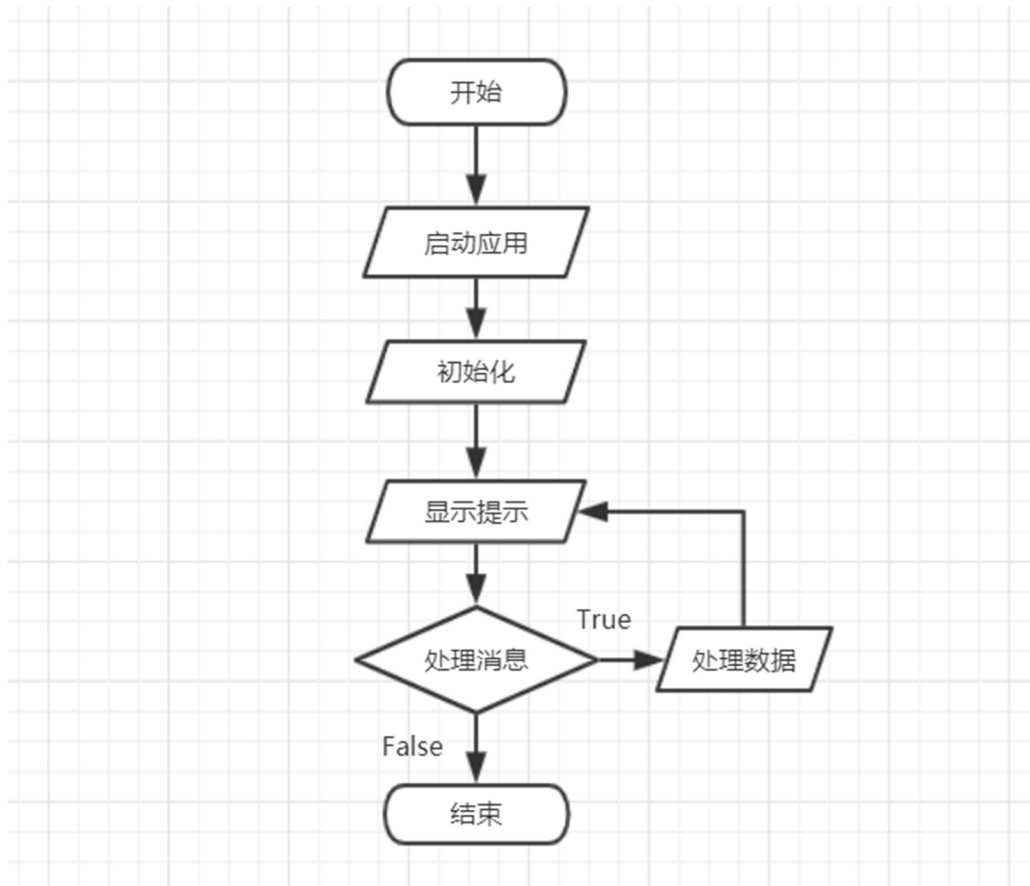


图 2 整个系统的调用逻辑

## Context

这个类可以通俗的理解为应用的上下文，其记录了程序在运行在任何一个时刻的运行状态，关键函数、字段以及其功能介绍如下。

函数签名/字段	功能
loginData	登录信息数据存储，以 <b>AVLTree</b> 的形式存储
page	主页面，主要用于区分不同的功能块。
subpage	此页面，主要用于记录多步操作时界面状态。
menus	以 map 的形式保存，用于存储友好的菜单
Menu& getCurrentMenu()	获取当前页面对应的菜单

函数签名/字段	功能
string getPage()	获取当前的页面，默认为 main
void setPage(string page)	设置当前的页面
bool isMenuIndexOf(const string& option, const string& index)	辅助函数，用于判断 index 是否和某选项对应

表 2 Context 的类的设计

其中 page 和 subpage 用于描述程序的运行状态，并进行相应的操作。是整个系统保持状态的关键数据。而 loginData 是登录的信息存储。

## App

是 AppBase 的派生类，重写了 AppBase 中的 onTip(),onLoad()和 onHandleInput()三个函数，实现了整个系统的功能。

## Menu 层（菜单）设计

### page:main

- import: 导入
- export: 导出
- client: 转到普通用户界面
- admin: 转到管理员界面
- exit: 退出程序

### page:client

- exit: 退回主界面
- register: 注册
- login: 登录

### page:logged

- logout: 退出登录
- hello: 欢迎
- changepw: 修改密码
- delete: 删除账号

### page:admin

- exit: 退回主界面
- register: 添加账号

- login: 查看账号信息
- changepw: 修改密码
- delete: 删除账号
- drop: 清除缓存
- debug|print: 调试打印
- debug|show: 显示所有信息

```

page = client
*****
1. 退回主界面      2. 注册      3. 登录
*****
> 请选择菜单中的一项:3
> 请输入用户名和密码(quit取消):hell 12345678
密码错误
> 请输入用户名和密码(quit取消):cht 12345678
cht登录成功
page = logged
*****
1. 退出登录      2. 欢迎      3. 修改密码
4. 删除账号
*****
> 请选择菜单中的一项:2
Welcome!cht
page = logged
*****
1. 退出登录      2. 欢迎      3. 修改密码
4. 删除账号
*****
> 请选择菜单中的一项:1
page = client
*****
1. 退回主界面      2. 注册      3. 登录
*****

```

图 3 菜单系统的设计

## Data 层（数据操作）设计

Data 层是本系统的功能核心实现部分，主要的三个类为 bnode,AVLTree 和 LoginData 其中，AVLTree 是一个模板类，提供了一系列通用的平衡二叉树功能，并使用面向对象的方式提供了封装，也是整个数据结构课程设计的考察核心。LoginData 是存储数据单元，存储单个的用户数据。

### bnode

标记为一个二叉树的节点，代码如下：

```
//  
// Created by cht on 2019/11/27.  
//  
template <typename T>  
class bnode {  
public:  
    explicit bnode(T  
value):value(value),left(nullptr),right(nullptr),height(1),flag(false){  
  
    }  
  
    T value;  
    bnode<T>* left;  
    bnode<T>* right;  
    int height = 0;  
    /**  
     * 用于记录该节点的最近状态 1:LL 2:RR 3:LR 4:RR  
     */  
    int flag = 0;  
};
```

## AVLTree

关键函数和功能介绍如下：（在源代码中已给出更加详细的注释）。

函数签名	功能
<i>field</i> : bnode<T>* root	根节点
bool loadFile(const string& fileName)	在文件中加载缓存
void saveFile(const string& fileName)	保存缓存到文件
void update(T value)	更新节点，若不存在则插入新节点，存在则替换节点的值，插入位置由==来决定。
bnode* find(T value)	依据==来定位到某一个节点
bool remove(T value)	删除某一个节点
void serialize(ostream& oStream)	将内存序列化到输出流



函数签名	功能
<code>void print()</code>	提供显示所有节点极其信息的功能
<code>bool deSerialize()</code>	从输入流中反序列化对象
<code>void clear()</code>	清空整棵 AVL 树
<code>void lPrint()</code>	提供横向打印输出的功能
<i>private:</i> <code>void _r(bnode&lt;T&gt;* &amp; node)</code>	在某个节点上施加右旋操作，属于原子操作
<i>private:</i> <code>void _l(bnode&lt;T&gt;* &amp; node)</code>	在某个节点上施加左旋操作，属于原子操作
<i>private:</i> <code>void _balance(bnode&lt;T&gt;* &amp; node)</code>	对某个节点进行平衡操作，自动判断不平衡类型并进行相应的旋转操作
<i>private:</i> <code>void _updateHeight(bnode&lt;T&gt;* node)</code>	对某个节点的高度重新计算
<i>private:</i> <code>void _getHeight(bnode&lt;T&gt;* node)</code>	获取某个节点的高度，当某节点为 <code>nullptr</code> 时，高度为 0
<i>private:</i> <code>void _bf(bnode&lt;T&gt;* node)</code>	获取一个节点的 BF 值
<i>private:</i> <code>void _lPrint(ostream&amp; oStream, bnode&lt;T&gt;* node, int start = 0, int spacing = 4)</code>	辅助函数，用于横向输出整个树
<i>private:</i> <code>void _lPrintSlicing(ostream&amp; oStream, bnode&lt;T&gt;* node, action action, int start = 0, int spacing = 4)</code>	辅助函数，用于横向输出整个树
<i>private:</i> <code>bnode&lt;T&gt;* _find(bnode&lt;T&gt;* node, T value)</code>	查找某一个节点
<i>private:</i> <code>bnode&lt;T&gt;* _print(ostream&amp; oStream, bnode&lt;T&gt;* node)</code>	中序输出节点
<i>private:</i> <code>void _serialize(ostream&amp; oStream, bnode&lt;T&gt;* node)</code>	序列化
<i>private:</i> <code>void _update(bnode&lt;T&gt;* &amp; node, T value)</code>	更新一个值，或替换或插入
<i>private:</i> <code>void _clearFlag(bnode&lt;T&gt;* node)</code>	清除 node 的 flag 标记
<i>private:</i> <code>void _findMax(bnode&lt;T&gt;* node)</code>	内部函数，找到某个子树的最大节点

函数签名	功能
<i>private</i> : void _findMin(bnode<T>* node)	内部函数，找到某个子树的最小节点
<i>private</i> : void _remove(bnode<T>*& node, T value)	内部函数：移除某一个值
<i>private</i> : void _clear(bnode<T>* node)	内部函数，删除所有节点

表 3 AVLTree 的类的设计

功能主要包括与外部缓存做交互的序列化和反序列化。插入节点、删除节点、查找节点的几大操作。下面主要介绍从插入节点、删除节点、调整平衡的原理。并给出尽量充分测试的测试数据样例。

同时，在 **bnode** 结构中我引入了 flag 来更加友好的显示上一步的操作。

## 平衡树的相关原理和操作方法

### 平衡树

平衡树的概念，可以使用递归的方式来进行定义。平衡树，要么是一颗空树，要么是一个满足一下条件的二叉搜索树。

1. 当前节点的左节点和右节点均为一个平衡树。
2. 左节点的高度与右节点的高度的差的绝对值小于 1。

平衡树是一种优化的二叉搜索树，常规的二叉搜索树可能为因为过于稀疏（接近链表）而导致树的深度过高，导致搜索效率降低的问题。在二叉搜索树建立平衡的算法中，引入了平衡因子的概念，通过判断 BF 因子来判断平衡树节点的平衡性，并对节点进行相应的操作。

### 插入与删除

插入与删除都使用了分而治之的概念，即将一个问题分解成一系列子问题的和。在程序中由递归函数来实现。插入和删除的调整过程都使用了递归函数来实现从下而上的平衡调整过程，从而保持整棵树的平衡性。

因此，我们可以将插入分为两步操作：在子树上进行插入，调整当前节点的平衡性。在删除的过程中，也可以使用类似的概念。

不一样的是，在删除操作中使用了特殊的小技巧，即转换的思想。因为删除叶子节点比删除分支节点来的容易。使用了用左子树的最大节点替代当前节点，然后删除左子树的最大节点的替换思想来实现删除操作。

### 平衡树的调整操作

平衡树的调整操作，是平衡树的关键算法。前人的经验总结，知道平衡树的不平衡性主要分为四种情况。而每种情况都可以使用若干步的左旋节点或者右旋节点来完成。按照步骤，调整主要分为判断平衡性和实施调整来进行。下面分别介绍各种不平衡性和操作方法。

对于节点的替换问题，采用优先旋转的原则重新调整各个节点之间的链接关系。

## LL 型

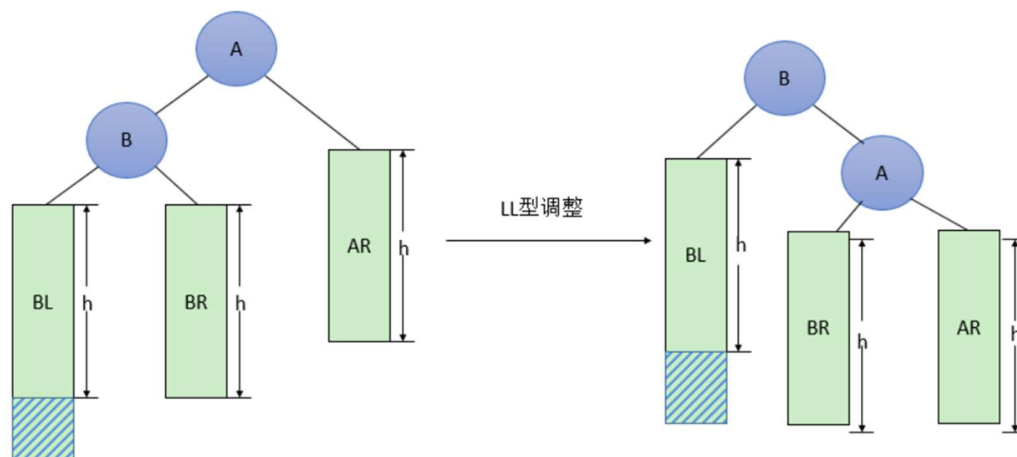


图 4 平衡树的不平衡情况——LL 型

判断方法： $bf(A) > 1 \ \&\& \ bf(B) > 0$

操作方法：右旋(A)

## RR 型

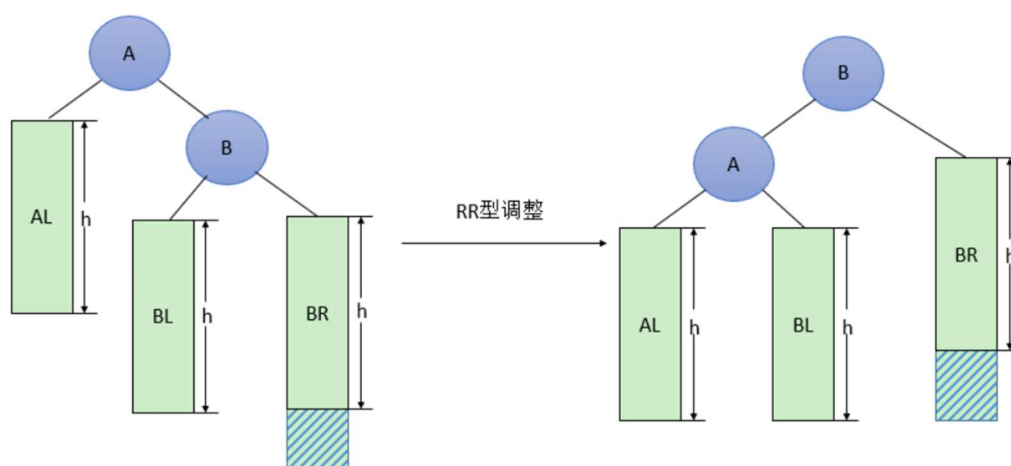


图 5 平衡树的不平衡情况——RR 型

判断方法： $bf(A) < -1 \ \&\& \ bf(B) < 0$

操作方法：左旋(A)

### LR 型

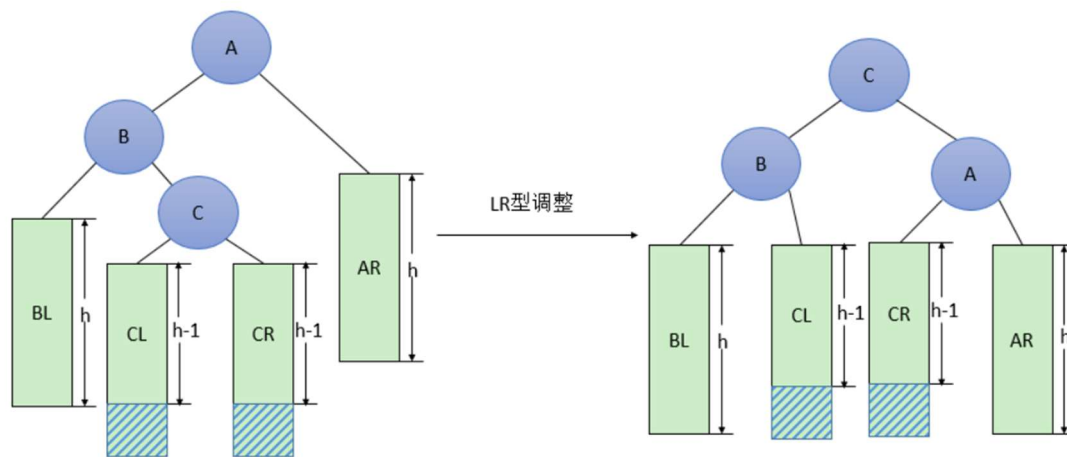


图 6 平衡树的不平衡情况——LR 型

判断方法： $bf(A) > 0 \ \&\& \ bf(B) < 0$

操作方法：左旋(B) 然后 右旋(A)

### RL 型

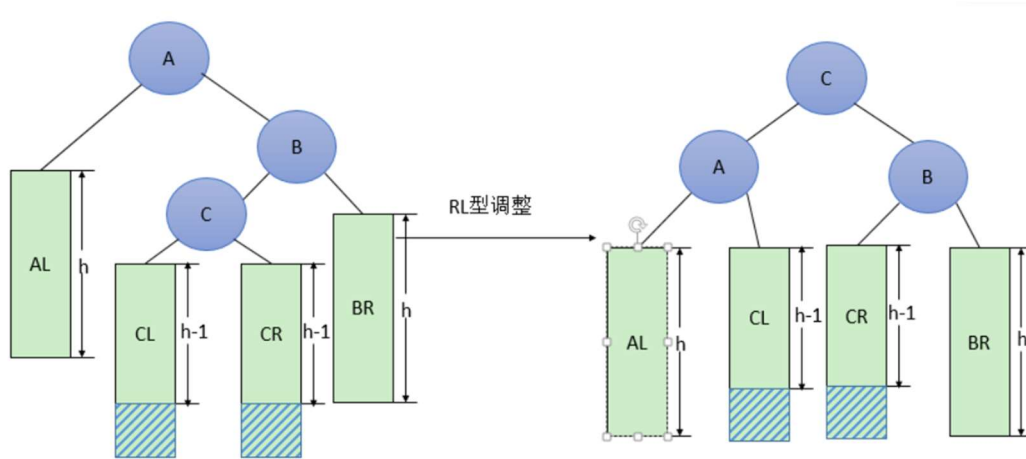


图 7 平衡树的不平衡情况——RL 型

判断方法： $bf(A) < 0 \ \&\& \ bf(B) > 0$

操作方法：右旋(B) 然后 左旋(A)

平衡树插入与删除的边界测试样例

为了确保系统的稳定性，需要对系统进行尽量完善的测试，本系统的测试样例如下，将从边界条件和测试样例的完整性出发进行测试。

测试范围

测试范围	1	2	3	4	5
插入	不调整 √	LL √	LR √	RL √	RR √
删除	根节点 √	叶节点 √	删除唯一的节点 √		

初始状态

```

        no
    mifeifei
        hello
hell
        devcheck
    cht
        abcd
159a
    1590
1234
    1233

page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项: 7
        no
    mifeifei
        hello
hell
        devcheck
    cht
        abcd
159a
    1590
1234
    1233
```

图 8 平衡树测试样例 1

插入(测试 LR 型，且调整子树为复杂节点)

测试样例：插入 aaaa,12345678 测试结果符合预期

```

              no
            mifeifei
              hello
        hell
      devcheck
(RL)cht
      abcd
          aaaa
159a
          1590
      1234
          1233
```

```
> 请选择菜单中的一项:2
> 请输入你想添加账号的用户名和密码(quit取消):aaaa 12345678
创建用户aaaa成功
page = admin
*****
1.退回主界面      2.添加账号      3.查看账号信息
4.修改密码        5.删除账号      6.清除缓存
7.调试/打印       8.调试/显示所有信息
*****
> 请选择菜单中的一项:7
              no
            mifeifei
              hello
        hell
      devcheck
(RL)cht
      abcd
          aaaa
159a
          1590
      1234
          1233
```

图 9 平衡树测试样例 2

### 插入(测试 LL 型, 简单节点)

测试样例: 插入 160a,12345678 测试结果符合预期

```
no
mifeifei
hello
hell
devcheck
cht
abcd
(LL)aaaa
160a
159a
1590
1234
1233
```


```
> 请选择菜单中的一项:2
> 请输入你想添加账号的用户名和密码(quit取消):160a 12345678
创建用户160a成功
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印      8. 调试/显示所有信息
*****
> 请选择菜单中的一项:7
no
mifeifei
hello
hell
devcheck
cht
abcd
(LL)aaaa
160a
159a
1590
1234
1233
```

图 10 平衡树测试样例 3

### 插入(测试不调整)

测试样例：依次插入 159f,12345678;160g,12345678;abca,12345678 测试结果符合预期

```
no
  mifeifei
    hello
hell
  devcheck
cht
    abcd
      abca
aaaa
      160g
      160a
      159f
159a
      1590
      1234
      1233
```



```
> 请选择菜单中的一项:2
> 请输入你想添加账号的用户名和密码(quit取消):abca 12345678
创建用户abca成功
page = admin
*****
1.退回主界面      2.添加账号      3.查看账号信息
4.修改密码        5.删除账号      6.清除缓存
7.调试/打印      8.调试/显示所有信息
*****
> 请选择菜单中的一项:7
no
  mifeifei
    hello
hell
  devcheck
cht
    abcd
      abca
aaaa
      160g
      160a
      159f
159a
      1590
      1234
      1233
```

图 11 平衡树测试样例 4

插入(测试 RR 型，且调整子树为复杂节点)



测试样例：依次插入 abb9,12345678;abb8,12345678

```
no
mifeifei
hello
hell
devcheck
cht
abcd
abca
abb9
abb8
(RR)aaaa
160g
160a
159f
159a
1590
1234
1233
```

```

> 请输入你想添加账号的用户名和密码(quit取消):abb8 12345678
创建用户abb8成功
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印        8. 调试/显示所有信息
*****
> 请选择菜单中的一项:7
no
mifeifei
hello
hell
devcheck
cht
abcd
abca
abb9
abb8
(RR)aaaa
160g
160a
159f
159a
1590
1234
1233

```

图 12 平衡树测试样例 5

插入(测试 RL 型，且调整子树为复杂节点)

测试样例：插入 hellc,12345678

```

no
mifeifei
(RL)hello
hellc
hell
devcheck
cht
abcd
abca
abb9
abb8
aaaa
160g

```

```

160a
159f
159a
1590
1234
1233

```

```

> 请输入你想添加账号的用户名和密码(quit取消):hellc 12345678
创建用户hellc成功
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印      8. 调试/显示所有信息
*****
> 请选择菜单中的一项:7
no
mifeifei
(RL)hello
hellc
hell
devcheck
cht
abcd
abca
abb9
abb8
aaaa
160g
160a
159f
159a
1590
1234
1233

```

图 13 平衡树测试样例 6

## 删除(根节点)

测试样例：删除 cht

```

no
mifeifei
hello
hellc
hell
devcheck
abcd
abca

```

```

(LL)abb9
      abb8
aaaa
      160g
      160a
      159f
159a
      1590
      1234
      1233

> 请选择菜单中的一项:5
> 请输入你想删除账号的用户名(quit取消):cht
删除用户成功
page = admin
*****
1. 返回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码      5. 删除账号      6. 清除缓存
7. 调试/打印      8. 调试/显示所有信息
*****
> 请选择菜单中的一项:7
      no
      mifeifei
hello
      hellc
      hell
      devcheck
abcd
      abca
      (LL)abb9
      abb8
      aaaa
      160g
      160a
      159f
159a
      1590
      1234
      1233

```

图 14 平衡树测试样例 7

## 删除(叶节点)

测试样例：删除 abcd

```

      mifeifei
hello
      hellc
      hell

```

```

devcheck
abcd
    abcd
        abca
            abb9
                abb8
                    aaaa
                        160g
                            160a
                                159f
                                    159a
                                        1590
                                            1234
                                                1233

```

```

> 请输入你想删除账号的用户名 (quit取消): no
删除用户成功
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印      8. 调试/显示所有信息
*****
> 请选择菜单中的一项: 7
    mifeifei
        hello
            hellc
                hell
                    devcheck
abcd
    abcd
        abca
            abb9
                abb8
                    aaaa
                        160g
                            160a
                                159f
                                    159a
                                        1590
                                            1234
                                                1233

```

图 15 平衡树测试样例 8

## 调试分析

## 技术上的难点

1. 平衡树的平衡判断和调整平衡操作。

平衡树的平衡判断主要依据节点的 BF 值和子节点的 BF 值。调整平衡的操作主要使用旋转操作，节点的调整顺序是关键的问题，否则可能会出现断链或者出现无限循环的情况。

2. 如何使用递归函数实现插入和删除的功能。

递归函数的停止调整，每次递归函数应该执行的操作顺序，都是本实验技术上的难点，通过多次测试，可以有效解决潜在的 bug。

3. 如何原模原样的将当前树保存到文件里。

保存二叉树的方法，一般涉及到遍历的操作。保存二叉树可以使用前序遍历+中序遍历或者带标记的前序遍历来完成。如何将整个树的结构信息保存到文件中是比较一个关键的地方。

## 调试错误分析

1. 文件读取时无法按照预期重构平衡二叉树

当时保存文件的方法是用前序遍历的方法，经过多次调试后发现，保存文件后再次读取并不会按照原来的顺序重新恢复二叉树。

解决方法是使用特殊的遍历方法来进行保存，考虑到二叉树的高度平衡性，故使用层序遍历的方法保存整棵二叉树。

对应的代码如下：

```
/**
 * 序列化
 * @param ostream 输出流
 * @param node 节点
 */
void _serialize(ostream& ostream, bnode<T>* node){
    if(node == nullptr){
        return;
    }

    queue<bnode<T>*> q;
    q.push(node);

    while(!q.empty()){
```

```

        bnode<T>* element = q.front();
        q.pop();
        ostream << element->value << endl;
        if(element->left != nullptr){
            q.push(element->left);
        }
        if(element->right != nullptr){
            q.push(element->right);
        }
    }

    //ostream << node->value << endl ;
    //_serialize(ostream,node->left);
    //_serialize(ostream,node->right);
}

```

## 2. 在删除某个节点后程序崩溃

程序崩溃发生在删除根节点的时候，导致程序直接崩溃。

程序崩溃发生的原因发生在重新调整节点的时候，由于判断 BF 值的算法存在漏洞，导致试图将一个节点赋值给一个空节点的左子树，导致程序崩溃。

之前判断不平衡情况是通过插入节点的路径来计算的，之后重新设计了判断方法，解决了这个 bug。

```

/**
 * 对当前节点进行调整平衡的操作 &reference type
 * @param node 要操作的节点
 */
void _balance(bnode<T>*& node){
    int bf = _bf(node);
    if(bf > 1){ // 节点左边更高，为LL型或者LR型
        int inBf = _bf(node->left);
        if(inBf < 0){
            // 考虑LR型需要旋转左分支节点
            _l(node->left);
        }
        _r(node);
    } else if(bf < -1){ // 节点右边更高，为RR型或者RL型
        int inBf = _bf(node->right);
        if(inBf > 0){
            // 考虑RL型需要旋转右分支节点

```

```

        _r(node->right);
    }
    _l(node);
}
}

```

#### 4. 在给操作节点设置 flag 时无法按照预期清除以前的 flag

之前是通过节点的是否旋转来动态调整节点的 flag 值，后来发现之后调整节点值的时候，无法消除原来的标记。

解决方法是在更新节点的时候，清除 flag，以达到较好的效果，可以更具节点的旋转情况，从而推断调整的类型，这样可以提高树结构的可读性。

设置 flag 标记的函数在\_l（左旋）和\_r（右旋）中。

#### 5. 在清空整棵树之后添加节点程序崩溃

忘记最后将根节点赋值成 nullptr，导致无法访问的内存地址的情况。

## 测试结果分析

测试与用户交互的各种逻辑

测试菜单之间的跳转功能



```
欢迎来到用户登录系统(AVL实验) VERSION:beta 5
page = main
*****|*****
1.导入          2.导出          3.转到普通用户界面
4.转到管理员界面 5.退出程序
*****
> 请选择菜单中的一项:1
> 请输入你想要导入的文件名(quit取消):test1
导入成功
page = main
*****
1.导入          2.导出          3.转到普通用户界面
4.转到管理员界面 5.退出程序
*****
> 请选择菜单中的一项:4
page = admin
*****
1.退回主界面    2.添加账号    3.查看账号信息
4.修改密码      5.删除账号    6.清除缓存
7.调试/打印     8.调试/显示所有信息
*****
```

图 16 功能测试样例 1

## 测试输入检查的功能

非法输入检查。

```
> 请选择菜单中的一项:2
> 请输入你想添加账号的用户名和密码(quit取消):a
用户名长度不合法,应在2-20位
> 请输入你想添加账号的用户名和密码(quit取消):aa
密码长度不合法,应在6-64位
> 请输入你想添加账号的用户名和密码(quit取消):aa ;;
密码长度不合法,应在6-64位
> 请输入你想添加账号的用户名和密码(quit取消):aa IHAVEABUH
创建用户aa成功
```

图 17 功能测试样例 2

检查存在的用户

```
> 请选择菜单中的一项:2
> 请输入你想添加账号的用户名和密码(quit取消):mifeifei 12345678
已存在该用户,无法创建用户
```

图 18 功能测试样例 3

### 测试查看账号信息的功能

```
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项:3
> 请输入你想查看账号的用户名(quit取消):hello
hello 123456
```

图 19 功能测试样例 4

### 测试修改密码的功能

```
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项:3
> 请输入你想查看账号的用户名(quit取消):hello
hello 123456
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项:4
> 请输入你想修改账号的用户名和新密码(quit取消):hello 12345678
修改密码hello成功
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项:3
> 请输入你想查看账号的用户名(quit取消):hello
hello 12345678
```

图 20 功能测试样例 5

## 测试输出所有信息的功能

```
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印       8. 调试/显示所有信息
*****
> 请选择菜单中的一项:8
1233 123456
1234 123456
1590 123456
159a 123456
159f 12345678
160a 12345678
160g 12345678
aa I HAVEABUH
aaaa 12345678
abb8 12345678
abb9 12345678
abca 12345678
abcd 123456
cht 12345678
devcheck 123456
hell 123456
hello 12345678
mifeifei 1234567890
no 12345678
```

图 21 功能测试样例 6

## 测试注册账号的功能

```
*****
1. 退回主界面      2. 注册      3. 登录
*****
> 请选择菜单中的一项:2
> 请输入你想注册账号的用户名和密码(quit取消):mime 12345678
创建用户成功
```

图 22 功能测试样例 7

## 测试登录、欢迎、修改密码功能

```

*****
1. 退回主界面      2. 注册      3. 登录
*****
> 请选择菜单中的一项:3
> 请输入用户名和密码(quit取消):mime 12345678
mime登录成功
page = logged
*****
1. 退出登录      2. 欢迎      3. 修改密码
4. 删除账号
*****
> 请选择菜单中的一项:2
Welcome!mime
page = logged
*****
1. 退出登录      2. 欢迎      3. 修改密码
4. 删除账号
*****
> 请选择菜单中的一项:3
> 请输入新密码(quit取消):123456
修改密码mime成功
page = logged

```

图 23 功能测试样例 8

## 测试注销账号功能

```

page = logged
*****
1. 退出登录      2. 欢迎      3. 修改密码
4. 删除账号
*****
> 请选择菜单中的一项:4
> 请确认你想删除该账号吗(y/n):y
删除用户成功
page = client
*****
1. 退回主界面      2. 注册      3. 登录
*****
> 请选择菜单中的一项:3
> 请输入用户名和密码(quit取消):mime 12345678
用户不存在

```

图 24 功能测试样例 9

## 测试平衡树各种操作（插入，删除）的完整性

此部分为了达到更好的效果，已在**测试样例**部分给出。

## 测试读取文件和保存文件的功能

保存文件之前

```
page = admin
*****
1. 退回主界面      2. 添加账号      3. 查看账号信息
4. 修改密码        5. 删除账号      6. 清除缓存
7. 调试/打印        8. 调试/显示所有信息
*****
> 请选择菜单中的一项: 7
no
mifeifei
hello
hell
devcheck
cht
abcd
abca
abb9
abb8
aaaa
aa
160g
160a
159f
159a
1590
1234
1233
```

图 25 功能测试样例 10

```
导入成功
page = main
*****
1.导入          2.导出          3.转到普通用户界面
4.转到管理员界面 5.退出程序
*****
> 请选择菜单中的一项:4
page = admin
*****
1.退回主界面      2.添加账号      3.查看账号信息
4.修改密码        5.删除账号      6.清除缓存
7.调试/打印      8.调试/显示所有信息
*****
> 请选择菜单中的一项:7
no
mifeifei
hello
hell
devcheck
cht
abcd
abca
abb9
abb8
aaaa
aa
160g
160a
159f
159a
1590
```

图 26 功能测试样例 11

## 实验总结

本次实验主要包含了界面的功能设计，面向对象的设计，数据结构的概念理解和程序编写功能。本次实验关于平衡树的操作过程主要参考了一篇[CSDN 博客](#)，并按照自己所学的知识重新整理了概念，并使用引用的功能重新设计了函数。并使用面向对象的思想对平衡二叉树结构重新封装。

当然，本实验的重点在于理解整个平衡二叉树的概念，除了概念之外，也需要对分治法和递归算法的思想有一定的了解。

虽然说平衡二叉树这个思想已经掌握了,但是在第一次设计这个数据结构时仍然存在看不懂代码,需要自己推敲的情况。在实验的过程中,也因为频繁出现的指针错乱问题打乱程序编写的节奏。

同时,在编写软件的时候,需要尽量完全的测试,尽量包含测试可能包含的各种情况,各种边界条件和一场条件,让程序变得更加完善,不容易崩溃。