

**数据结构课程设计报告**



学 号 201806061108

姓 名 胡皓睿

班 级 计实1801

完成日期 2019.12.16

目录

[0](#_Toc27405864)

[1 实验题目与要求 2](#_Toc27405865)

[2 设计思路 3](#_Toc27405866)

[2.1 系统总体设计 3](#_Toc27405867)

[2.2 系统功能设计 4](#_Toc27405868)

[2.2.1 用户登陆验证功能 4](#_Toc27405869)

[2.2.2 添加用户功能 4](#_Toc27405870)

[2.2.3 显示用户功能 4](#_Toc27405871)

[2.2.4 删除用户功能 4](#_Toc27405872)

[2.2.5 修改密码功能 4](#_Toc27405873)

[2.3 类的设计 5](#_Toc27405874)

[2.3.1 AvlTree类 5](#_Toc27405875)

[2.3.2 Node类 5](#_Toc27405876)

[2.3.3 ShadowTreeNode类 5](#_Toc27405877)

[2.4 主程序的设计 6](#_Toc27405878)

[3 调试分析 7](#_Toc27405879)

[3.1 技术难点分析 7](#_Toc27405880)

[3.2 调试错误分析 7](#_Toc27405881)

[4 测试结果分析 7](#_Toc27405882)

[4.1 登陆 7](#_Toc27405883)

[4.2 管理员界面 8](#_Toc27405884)

[4.2.1 正向打印树 8](#_Toc27405885)

[4.2.2 添加用户 8](#_Toc27405886)

[4.2.3 删除用户 9](#_Toc27405887)

[4.3 一般用户 10](#_Toc27405888)

[4.3.1 修改密码 10](#_Toc27405889)

[5 附录 11](#_Toc27405890)

[5.1 main.cpp 11](#_Toc27405891)

[5.2 AvlTree.h 11](#_Toc27405892)

[5.3 AvlTree.cpp 11](#_Toc27405893)

[5.4 Node.h 11](#_Toc27405894)

[5.5 Node.cpp 11](#_Toc27405895)

[5.6 ShadowTreeNode.h 11](#_Toc27405896)

# 实验题目与要求

**【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；

【**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。

**验收要求**：**1、通过管理员权限，正向打印用户信息的平衡二叉查找树；**

**2、管理员在删除、插入用户信息后，需重新旋转调整AVL树，并正向打印；**

**3、用户只能看到自己的信息，只能更改密码。**

**【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**

# 设计思路

## 系统总体设计

系统需要实现“用户登陆系统”的功能，由于线性表的时间复杂度为O(n)，效率较低，不足以满足大容量高并发的需求，所以使用二叉搜索树的结构进行数据存储。由于二叉存储树可能存在不平衡的问题，影响数据结构的效率，所以我们使用Avl树进行存储。

## 系统功能设计



### 用户登陆验证功能

用户输入用户名和密码，系统会在数据库中查找该用户。如果用户不存在或者密码错误，发出相应的提示，否则根据用户所属的类型（管理员/普通用户），完成登陆

### 添加用户功能

管理员用户有权限添加用户，输入要添加的用户名和密码，如果在数据库中存在同名用户，提示添加失败，否则将用户信息存放到数据库里

### 显示用户功能

管理员用户有权限显示用户，使用打印命令，可以正向打印出Avl树中存储的用户信息。

### 删除用户功能

管理员用户有权限删除用户，输入要删除的用户名称，如果数据库中存在该用户，就将其删除，否则提示错误。

### 修改密码功能

普通用户可以更改自己的密码

## 类的设计



### AvlTree类

AvlTree类定义了一棵Avl树，包含了Node节点和ShadowTreeNode节点，包含了用影子树正向打印树，旋转节点，添加节点等功能。

### Node类

Node类定义了Avl树的节点，包含了name，password，height三个字段，分别存储用户名，密码，以及这个节点所处的高度。基于数据访问控制的要求，这三个字段使用了private访问权限，所以各自实现了set和get方法，进行修改和输出。同时，包含了指向两个子节点的left和right指针，用于实现数据结构。

### ShadowTreeNode类

ShadowTreeNode类定义了影子树节点，存储了每个节点打印时应当存储的位置，从而实现在正向打印时对其的功能。

## 主程序的设计



# 调试分析

## 技术难点分析

## 调试错误分析

# 测试结果分析

## 登陆

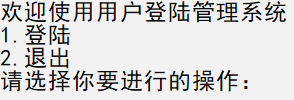


图4.1.1



图4.1.2

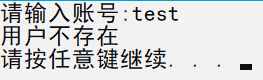


图4.1.3

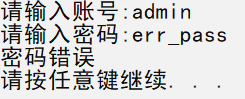


图4.1.4

在启动程序后，会先显示欢迎界面（图4.1.1），选择登陆，就会进入账号密码输入界面。如果输入正确的账号和密码，即可完成登陆过程。如果用户不存在或者密码错误，就会显示对应的错误信息（如图4.1.3，4.1.4）

## 管理员界面

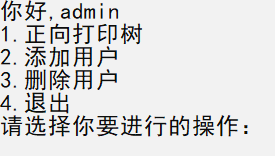


图4.2.1

如果用户名为“admin”，就会进入管理员界面（图4.2.1），可以在此界面选择要进行的操作。

### 正向打印树

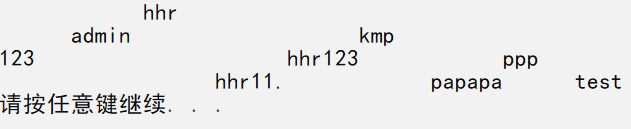


图4.2.1.1

选择正向打印树功能后，就会调用影子树的代码，在屏幕上生成一棵正向的树（图4.2.1.1）。为了让树可以对齐，我强制了用户名的最大长度如果超过该长度，就会将后面的内容变为点号，保证了名字字段等长。

### 添加用户

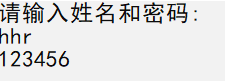


图4.2.2.1

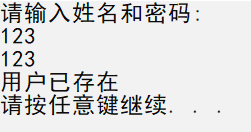


图4.2.2.2

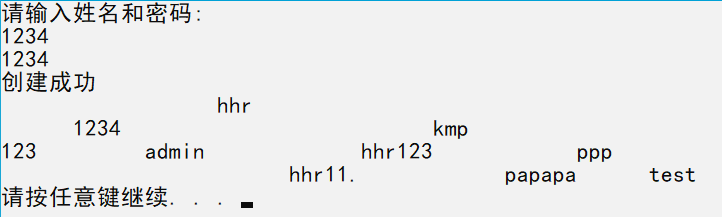


图4.2.2.3

选择添加用户功能后，会要求你输入用户名及其密码（图4.2.1.1）。如果已经存在同名用户，会提示你用户已存在（图4.2.2.2），否则就会将用户插入Avl树中，并打印这棵树（图4.2.2.3）。

### 删除用户

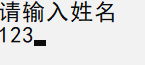


图4.2.3.1

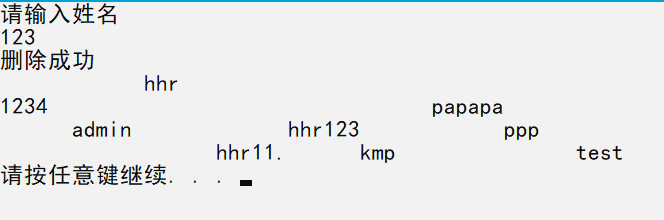


图4.2.3.2

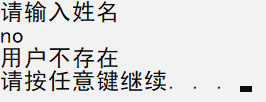


图4.2.3.3

选择删除用户功能后，系统会提示你输入用户名（图4.2.3.1），如果用户存在，就会将其删除，并打印出操作后的树（图4.2.3.2），否则提示该用户不存在（图4.2.3.3）。

## 一般用户

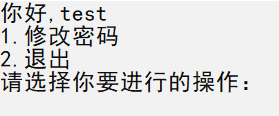


图4.3.1

如果用户为一般用户，则只允许修改密码（图4.3.1）。

### 修改密码



图4.3.1.1

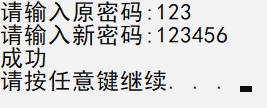


图4.3.1.2

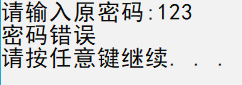


图4.3.1.3

进入修改密码界面后，系统会提示输入原密码（图4.3.1.1）。如果原密码错误，就会提示密码错误（图4.3.1.3），否则要求输入新密码，并完成修改（图4.3.1.2）。

# 附录

## main.cpp

## AvlTree.h

## AvlTree.cpp

## Node.h

## Node.cpp

## ShadowTreeNode.h