



东华理工大学
EAST CHINA INSTITUTE OF TECHNOLOGY

课程设计报告

C++面向对象程序设计

职工档案管理系统

学 号: 2022xxxxxx

学生姓名: Hard

专 业: 通信工程

班 级: XXXXXXXX

指导教师: XXX

2024 年 5 月 30 日

目录

一、课程设计目的和要求	1
二、需求分析	1
三、概要设计	1
1. 用户模块	2
2. 职工档案模块	2
四、详细设计和思路	3
1. 中英双语言功能	3
2. 背景音乐和语音播报	3
3. 注册	4
4. 登录	5
5. 忘记密码	7
6. 查询职工档案信息	8
7. 新增职工档案信息	9
8. 修改职工档案信息	10
9. 删除职工档案信息	15
五、运行环境及使用说明	15
六、项目总结与展望	16
参考文献	17
附录	18
1. Main.cpp	18
2. User.h	35
3. User.cpp	36
4. EmployeeProfile.h	43
5. EmployeeProfile.cpp	45
6. Language.h	57
7. Chinese.h	60
8. Chinese.cpp	60
9. English.h	64
10. English.cpp	65
11. Constant.h	69
12. MD5.h	70

一、课程设计目的和要求

C++面向对象课程设计是通信工程专业的集中实践性环节之一，是学习完《C++面向对象程序设计》课程后进行的一次全面的综合练习。通过 C++面向对象程序设计课程设计，让学生能全面理解、掌握面向对象的基本知识和技能，培养学生利用面向对象程序设计方法分析问题、解决问题的能力；培养学生针对具体的应用和实际问题，综合运用所学知识对问题进行抽象，分析、设计的能力，使学生掌握面向对象程序的编程方法。通过 C++面向对象课程设计教学，培养学生严谨求实的科学态度，激发学生的求知热情、探索精神、创新欲望，提高学生的综合素养。通过 C++面向对象程序设计课程设计，让学生熟悉面向对象基本理论和知识；掌握面向对象程序设计方法；初步掌握利用面向对象程序设计方法解决实际问题的技能。

设计职工档案管理系统，实现职工档案信息的显示、录入、删除、修改、排序、查询（可以按多种方式查询）等功能，数据存储于文件中。

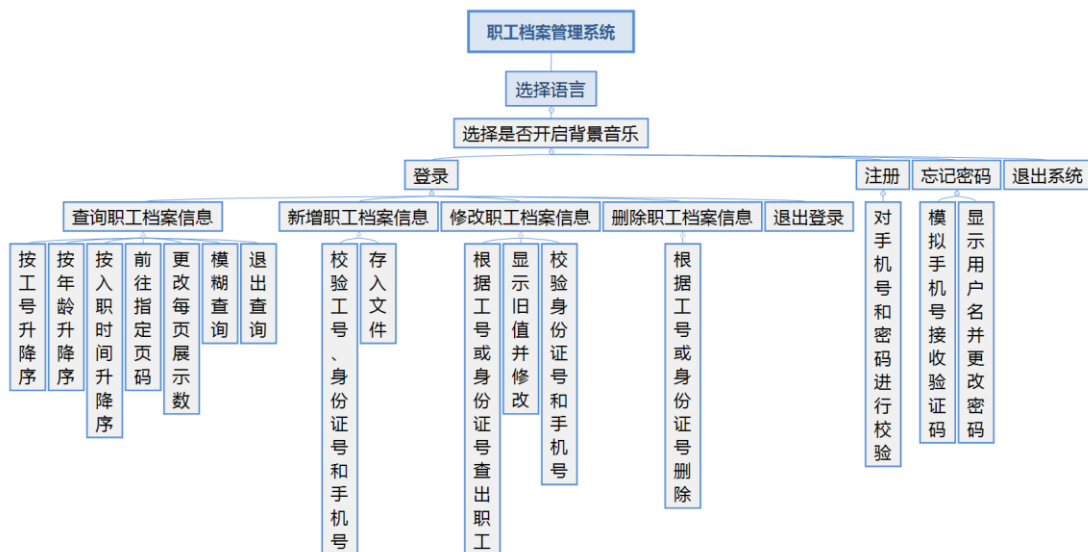
二、需求分析

本课程设计选题为“职工档案管理系统”。系统主要功能是管理职工档案信息，要求能实现但不限于：

1. 显示职工档案信息
2. 录入职工档案信息
3. 修改职工档案信息
4. 删除职工档案信息
5. 对员工工号、入职日期等进行排序
6. 通过工号、身份证查询职工档案信息
7. 模糊查询职工档案信息
8. 登录、注册
9. 忘记密码

三、概要设计

针对系统的设计要求，确定本系统的结构如下图：



此设计主要分了两个层面，一个是用户模块，一个是职工档案模块。

1. 用户模块

该模块可以为用户提供登录、注册和修改密码的功能。用户可以通过输入用户名和密码来进行登录，如果尚未拥有账户，则可以进行注册。同时，用户还可以在忘记密码时通过手机号进行修改密码。所有用户数据会被写入到名为“user.txt”的文件中，以便进行持久化存储。

2. 职工档案模块

该模块提供了对职工档案信息进行增加、删除、修改和查询的功能。用户可以根据需要对职工档案信息进行各种操作，包括对已有信息进行修改和删除不再需要的信息，或者新增新的档案信息。

在进行查询操作时，用户可以选择进行排序查询，按照指定的字段对档案信息进行排序，以便更加方便地浏览和管理数据。此外，模糊查询功能也是这个模块的一大特色，用户可以根据部分关键词来查询符合条件的档案信息，从而快速定位到需要的数据。

所有职工档案数据会被写入到名为“employee.txt”的文件中，以便进行持久化存储。这样不仅可以保证数据的安全性和可靠性，还可以方便后续的数据管理和分析工作。

四、详细设计和思路

1. 中英双语言功能

此功能鉴于本课程设计缺少亮点，思来想去决定实现一下双语言系统。此功能利用了诸多 C++ 课程知识，比如：继承、静态成员变量、纯虚函数和函数重写等等。除此之外，还使用了智能指针等。

此功能一开始是想通过简单的对变量名拼接实现，如：

```
string EnString = "_ENGLISH";
string resultString = "";
const string WELCOME_LOGIN = "欢迎登录职工档案管理系统";
const string WELCOME_LOGIN_ENGLISH = "Welcome to the Employee
Profile Management System";
```

对用户选择进行判断，当为英文时 resultString 被赋值为 "_ENGLISH" 得到最终的输出语言：

```
cout << WELCOME_LOGIN + resultString;
```

理想输出为：Welcome to the Employee Profile Management System

实际输出为：欢迎登录职工档案管理系统_ENGLISH

但是发现，将一个变量名和一个变量的值进行拼接得到一个新的变量名，再输出这个新变量的值的实现有点困难，查阅资料后发现可以改为工厂模式实现双语言功能。

在 main 函数中定义一个工厂函数和一个智能指针类型的全局变量，根据用户选择的语言创建不同的对象(自定义类 Chinese 或 English)，赋值给全局变量，然后利用这个全局变量调用类中的函数或获取类中的属性。并且 Chinese 类和 English 类继承于同一个基类 Language，便于统一函数和成员变量。

2. 背景音乐和语音播报

背景音乐通过 mciSendString 实现，用于播放 “.mp3” 格式的文件。背景音乐有五首音乐，如图所示：

名称	#	标题	参与创作的艺术家	唱片集
1.mp3		瞬间的永恒(钢琴曲)	赵海洋	夜色钢琴曲
2.mp3		いつも何度でも	宗次郎	Prime Selection
3.mp3		夜空中最亮的星(钢琴版)	赵海洋	
4.mp3		Eternity	David Foster	Eleven Words
5.mp3		Between Worlds	Roger Subirana	X II

在自定义函数中,通过设置随机数种子和 `rand()` 函数生成一个随机的数字,拼接到最终选择的文件名中。

语音播报通过 `PlaySound` 实现,用于播放 “.wav” 格式的文件。语音播报功能整合了一个函数,函数中会对传过来的文件名参数进行拼接,选择相应的语言和对应的语音进行播报。

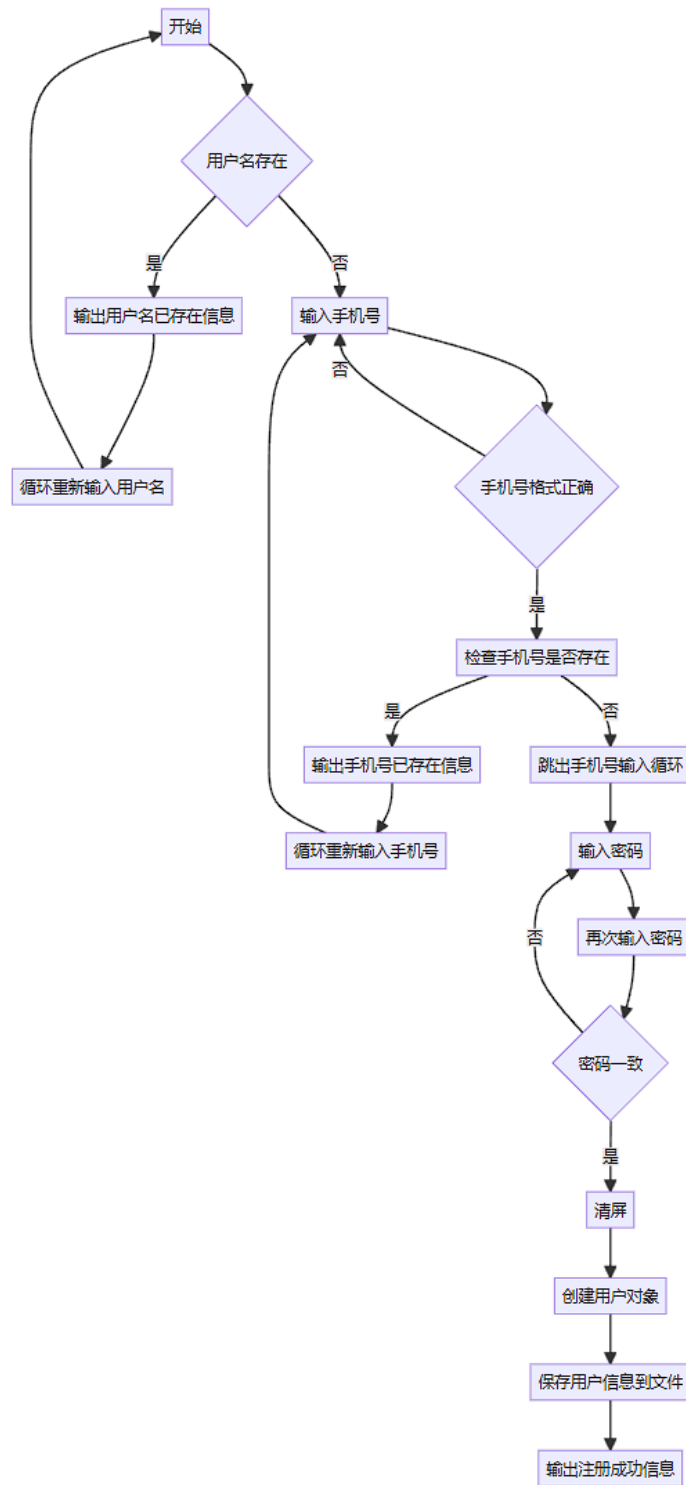
3. 注册

用户注册时,会通过自定义函数对用户名和手机号是否存在进行校验,并且也通过正则表达式校验手机号的格式。

在用户输入密码时,调用用户输入密码的函数,会使密码显示为 “*”,保护了用户的隐私,同时也会校验密码的长度,还会要求用户再次输入密码,以防输错。并且在存储密码时,也会使用 `MD5` 加密算法对密码进行加密。

```
欢迎注册职工档案管理系统
请输入用户名: admin
用户名已存在, 请重新输入
请输入用户名: admin1
请输入手机号: 18000001111
手机号已存在, 请重新输入
请输入手机号: 12345678910
手机号格式错误!
请输入手机号: 18011112222
请输入密码: ***
密码长度不能少于6位, 请重新输入: *****
密码长度不能超过20位, 请重新输入: *****
请再次输入密码: *****
两次密码不一致, 请重新输入!
请输入密码: *****
请再次输入密码: *****|
```

流程图:



4. 登录

输入用户名和密码进行登录。

当用户名不存在时会退出登录功能，效果如图：

```

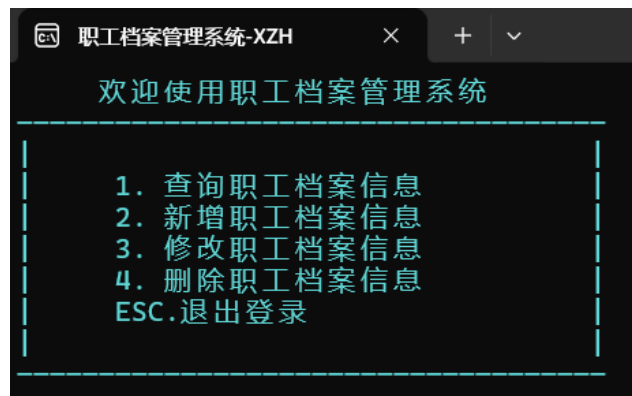
    欢迎登录职工档案管理系统
    请输入用户名: admin2
    用户名不存在, 请先注册
    2024/05/30 周四
    16:52
    欢迎使用职工档案管理系统
    *****
    *****
    *****      1. 登录      *****
    *****      *****
    *****      2. 注册      *****
    *****      *****
    *****      3. 忘记密码  *****
    *****      *****
    *****      ESC.退出系统 *****
    *****      *****
    *****
    *****
    
```

用户名和密码不匹配时会重新输入，效果如图：

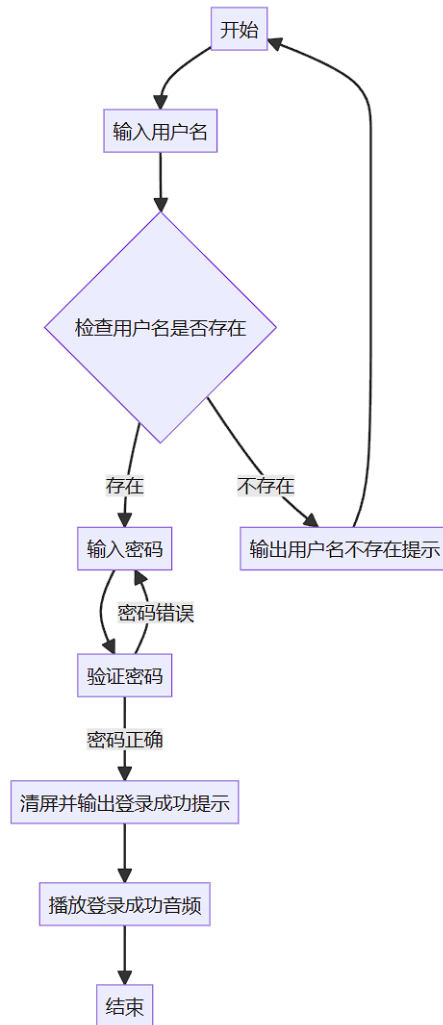
```

    欢迎登录职工档案管理系统
    请输入用户名: admin
    请输入密码: ***
    密码长度不能少于6位, 请重新输入: *****
    用户名或密码错误, 请重试!
    请输入用户名:
    
```

登录成功后，进入主界面



流程图：

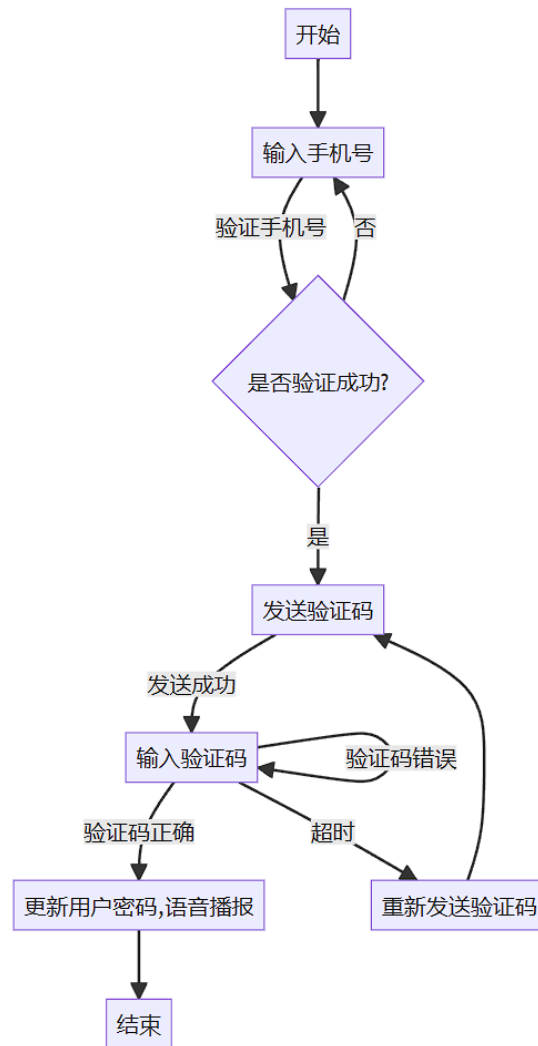


5. 忘记密码

该功能旨在解决用户忘记密码的问题。

用户输入手机号，先会利用正则表达式对手机号格式进行校验，并且在文件中查找是否存在此手机号，如何都通过了，则会模拟发送验证码，验证码设置了 60 秒的有效时间，通过当前时间减去发送时的时间计算出是否超过 60 秒，如果超过，则按回车键重新发送，否则显示用户名并输入新密码。

流程图：



6. 查询职工档案信息

(1) 自动全屏

当按“1”键进入查询职工档案信息功能时，会调用自定义函数 `fullScreen()` 时窗口最大化，便于显示数据。

(2) 分页查询

查询职工档案信息采用分页查询的方法。

通过自定义函数，将文件中的数据获取出来，放入一个 `EmployeeProfile` 类型的动态数组 `employeeProfiles` 中，计算分页参数页码 `pageNum` 和每页展示数 `pageSize`，获取对应的数据；

(3) 排序

工号、年龄和入职日期的升降序功能。

通过 `sort()` 函数和 `lambda` 函数对 `employeeProfiles` 进行排序, 实现正序和倒序功能。

(4) 页码和每页展示数

跳往页码、更改每页展示数。

通过修改 `pageNum` 和 `pageSize` 的值实现。

(5) 模糊查询

用户输入任何需要查询的内容, 系统会去文件中查询任何属性中包含了此查询内容的信息, 并且还计算了查询所需的时间。

通过自定义函数 `fuzzyQuery()` 实现此功能。其中在开始查询时记录开始时间, 查询结束之后记录结束时间, 再相减得到查询时间。在查询时, 会先通过 `regex_replace()` 函数将查询内容中的特殊字符进行转义, 防止在使用正则表达式时发生错乱。接下来通过 `regex_search()` 函数对查询内容和文件中的内容逐一进行匹配, 匹配成功则输出该条职工信息。

(6) 翻页功能

按 `A/←` 或 `D/→` 可以实现左右翻页功能。

通过 `_getch()` 获取用户按下的按键, 然后修改对应的 `pageNum` 实现该功能。

(7) 其余显示信息

在查询页面的尾部显示当前页、总页数、每页展示数和总职工数等信息。

1.按工号升序 2.按工号降序 3.按年龄升序 4.按年龄降序 5.按入职时间升序 6.按入职时间降序 7.前往页码 8.更改每页展示数 9.模糊查询 ESC键退出										
工号:	职工姓名:	身份证号:	性别:	年龄:	联系电话:	家庭地址:	学历:	职位:	入职日期:	所属部门:
2022040404	周正若	510104199608234524	女	27	15028182345	四川省成都市	硕士	人力资源专员	2023-10-12	人力资源部
2022050505	吴郭	31010119941010353X	男	29	18621101010	上海市黄浦区	本科	财务会计	2023-11-18	财务部
2022060606	张无忌	420102199312121817	男	30	13871212121	湖北省武汉市	硕士	研发工程师	2023-12-20	研发部
2022111111	刘洋	430103199709082723	女	26	18873109080	湖南省长沙市	本科	市场营销	2023-10-10	市场部
2022121212	孙丽	370203200306153247	女	20	15853261532	山东省青岛市	大专	客服代表	2023-11-20	客服部
2022123123	陈晨	330212200011110415	女	23	15957457111	浙江省宁波市	大专	UI设计师	2023-12-01	设计部
当前页: 1/11 6/页 向左: A/← 向右: D/→										
总职工数: 61										

7. 新增职工档案信息

在新增职工信息时, 会对职工的工号格式、身份证号格式、手机号格式以及其是否存在进行校验。信息输入完毕之后会将信息存储在文件中。

身份证号和手机号格式通过正则表达式进行校验; 校验是否存在时会通过自定义函数去文件中进行查询比对。

```

工号(长度为10位,前四位数为1956~当前年份之间):2022
工号格式有误!
工号(长度为10位,前四位数为1956~当前年份之间):1900212001
工号格式有误!
工号(长度为10位,前四位数为1956~当前年份之间):2025212001
工号格式有误!
工号(长度为10位,前四位数为1956~当前年份之间):2022212001
工号已存在!
工号(长度为10位,前四位数为1956~当前年份之间):2022212888
职工姓名:张三
身份证号(X为大写):362322222208091111
身份证号格式有误!
身份证号(X为大写):36232220040809111x
身份证号格式有误!
身份证号(X为大写):110101199507023512
身份证号已存在!
身份证号(X为大写):36232220040809111X
性别:男
年龄:23
联系电话:11122223333
手机号格式错误!
联系电话:18000001111
手机号已存在!
联系电话:18055556666
家庭地址:江西省南昌市
学历:本科
职位:软件开发
入职日期:2022-1-12
所属部门:开发岗|

```

8. 修改职工档案信息

先选择通过工号或者身份证号查询出职工信息

```

1.通过工号更新职工 2.通过身份证号更新职工 ESC键退出
|

```

信息查询出来之后会显示原来的值。工号不可修改,如果不进行修改按回车键即可,依旧会对职工的身份证号格式、手机号格式以及其是否存在进行校验,如图所示:

```

请输入工号：2022212001
按回车键则保留旧数据

旧值：2022212001
不可修改

旧值：张伟
新值：

旧值：362322200408121111
新值：

旧值：男
新值：

旧值：19
新值：

旧值：18000001111
新值：18055556666
手机号已存在！
请重新输入手机号：18088889999

旧值：江西省上饶市
新值：江西省南昌市

旧值：博士
新值：

旧值：算法开发
新值：

旧值：2024-4-2
新值：

旧值：人工智能岗
新值：

工号为 2022212001 的员工记录已成功修改。
1.通过工号更新职工 2.通过身份证号更新职工 ESC键退出

```

通过自定义函数实现，首先创建一个临时文件，再去原文件中查询是否存在该工号(或身份证号)，如果存在该工号(或身份证号)，则令标识 **found** 为 **true**。如果某个职工信息不是要更新的职工，则写入临时文件，如果是要更新的职工则将其信息获取出来，将每个属性以逗号为标识符分开存入 **employeeVector** 中便于操作，再逐一显示原信息并输入新信息，输入完毕后将 **employeeVector** 中的信息拼接成一个新的字符串存入文件。最后对 **found** 进行判断，如果为 **true** 则将原文件删除，将临时文件命名为原文件的名称，否则删除临时文件。

在实现不进行修改按回车键即可时，是本项目的难点之一，需要对 C++ 的输入字符流有比较熟练的掌握，并且还要熟练使用 **Debug** 功能，否则很容易出现不理想的效果。

当不清空输入缓冲区时，前面的输入会影响到后面的效果，比如注释掉以下代码之后

```

for (int j = 0; j < employeeVector.size(); j++) {
    if (j == 0) {
        cout << employeeLanguage->retainOldVal << employeeVector[0] << endl
            << employeeLanguage->catNotUpdate << endl << endl;
        j++;
        // 清除输入缓冲区
        //cin.ignore();
    }
}

```

便会出现以下效果，将直接跳过姓名的修改。

```

请输入工号： 2022212001
按回车键则保留旧数据

旧值： 2022212001
不可修改

旧值： 张伟
新值：
旧值： 362322200408121111
新值： |
  
```

再比如，注释掉以下代码之后

```

for (int j = 0; j < employeeVector.size(); j++) {
    if (j == 0) {
        cout << employeeLanguage->ratainOldVal << employeeVector[0] << endl
            << employeeLanguage->catNotUpdate << endl << endl;
        j++;
        // 清除输入缓冲区
        cin.ignore();
    }
    cout << employeeLanguage->oldVal << employeeVector[j] << endl << employeeLanguage->newVal;
    // 读取用户输入的字符
    char input = cin.get();
    // 如果不是回车键
    if (input != '\n') {
        // 将读取的字符放回输入流
        //cin.unget();
        // 进行输入操作
        cin >> newE;
    }
}
  
```

可以看到，输入中文时无法识别到信息

```

189 string newE;
190 for (int j = 0; j < employeeVector.size(); j++) {
191     if (j == 0) {
192         cout << employeeLanguage->ratainOldVal << employeeVector[0] << endl
193             << employeeLanguage->catNotUpdate << endl << endl;
194         j++;
195         // 清除输入缓冲区
196         cin.ignore();
197     }
198     cout << employeeLanguage->oldVal << employeeVector[j] << endl << employeeLanguage->newVal;
199     // 读取用户输入的字符
200     char input = cin.get();
201     // 如果不是回车键
202     if (input != '\n') {
203         // 将读取的字符放回输入流
204         //cin.unget();
205         // 进行输入操作
206         cin >> newE;
207     }
208 }
  
```

input: -43 '?'

newE: <字符串中的字符无效.>

职工档案管理系统 XZH

```

208 请输入工号： 2022212001
209 按回车键则保留旧数据
210
211 旧值： 2022212001
212 不可修改
213 旧值： 张伟
214 新值： 张三丰
215
  
```

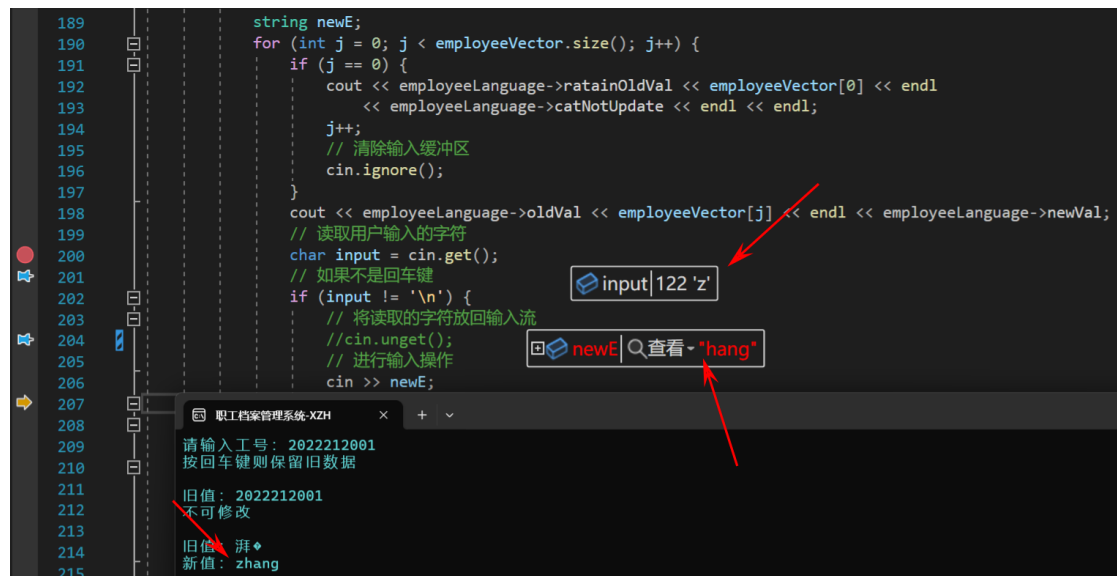
如果继续执行修改，最终保存的数据就会出现乱码

请输入工号：2022212001
按回车键则保留旧数据

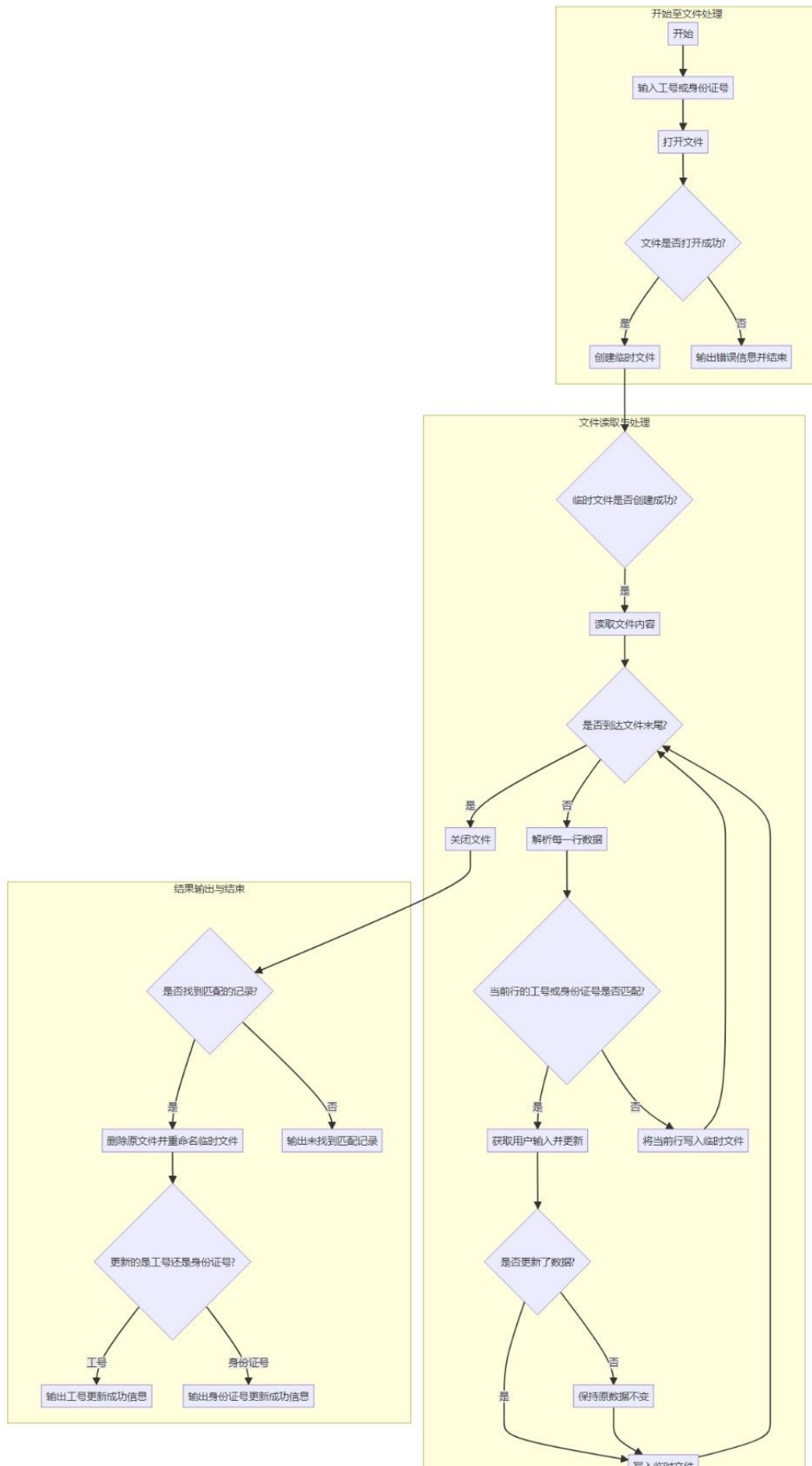
旧值：2022212001
不可修改

旧值：湃◆
新值：

如果输入的是英文时将会丢失第一个字符



流程图：



9. 删除职工档案信息

通过工号和身份证号进行删除。

通过自定义函数实现，首先创建一个临时文件，先去原文件中查询是否存在该工号(或身份证号)，如果存在该工号(或身份证号)，则令标识 **found** 为 **true**。如果某个职工信息不是要删除的职工，则写入临时文件，如果是要删除的职工则不写入临时文件。最后对 **found** 进行判断，如果为 **true** 则将原文件删除，将临时文件命名为原文件的名称，否则删除临时文件。

```
请输入工号：2022212881
未找到工号为 2022212881 的员工记录。
1.通过工号删除 2.通过身份证号删除 ESC键退出
```

```
请输入工号：2022212888
工号为 2022212888 的员工记录已成功删除。
1.通过工号删除 2.通过身份证号删除 ESC键退出
```

考虑到删除全部数据实现起来相对简单（只需创建临时文件，删除原文件，再将临时文件改名为原文件名即可），但是相对来说不太安全，容易将数据全部删除，故本项目不实现删除全部员工操作功能。

五、运行环境及使用说明

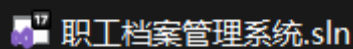
操作系统：Windows 11

运行软件：Microsoft Visual Studio 2022

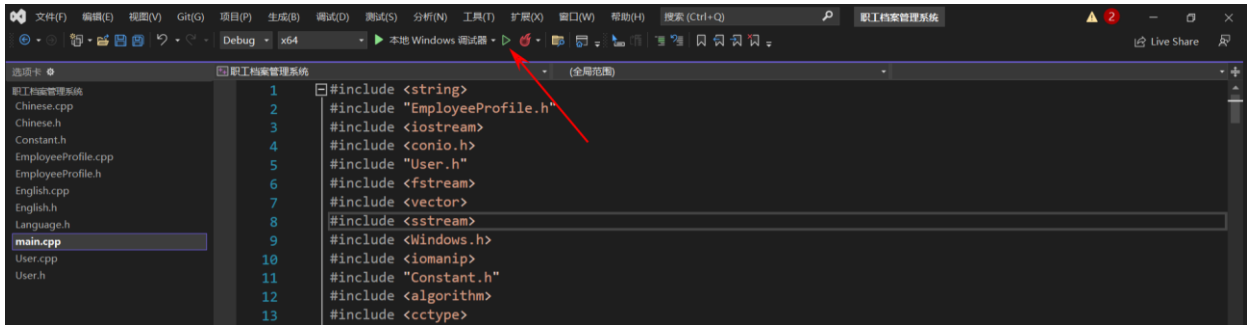
Gitee 地址：<https://gitee.com/Hard X/employee-profile.git>

Github 地址：<https://github.com/HardX8/employee-profile.git>

本项目主函数为 **main()** 函数。运行本项目，用 Microsoft Visual Studio 2022 打开“职工档案管理系统.sln”文件



直接运行即可



或者直接运行文件中的“职工档案管理系统.exe”

User.cpp	2024/5/31 23:51	C++ Source	9 KB
User.h	2024/6/1 23:17	C/C++ Header	2 KB
user.txt	2024/6/3 22:15	文本文档	1 KB
职工档案管理系统.exe	2024/6/3 22:09	应用程序	214 KB
职工档案管理系统.sln	2024/5/21 8:23	Visual Studio Sol...	2 KB
职工档案管理系统.vcxproj	2024/6/2 18:57	VC++ Project	8 KB

默认登录用户名：admin

登录密码：admin1

在运行本项目时，最好确保当前目录下存在“user.txt”和“employee.txt”文件，并且在当前目录下要存在 video 文件夹，将音频文件放入其中。

六、项目总结与展望

在“职工档案管理系统”的开发过程中，我深刻领悟到 C++ 面向对象编程的精髓与实践应用的价值。本课程设计项目是通信工程专业学生综合能力培养的重要环节，它要求我在掌握面向对象程序设计的基础之上，通过具体应用案例，将理论知识转化为实用技能。

项目伊始，我明确了职工档案管理系统的目标——实现职工档案信息的高效管理，包括显示、录入、删除、修改、排序及查询等功能。为了增强系统实用性与用户体验，我引入了中英双语支持、背景音乐与语音播报等特色功能，不仅丰富了系统的交互性，也提升了用户使用的便捷性和趣味性。

在功能实现上，我注重细节与用户体验。注册流程中，通过正则表达式验证手机号格式，确保信息准确无误；登录功能的校验机制，有效保障了账户安全；忘记密码功能通过验证码验证，既保证安全性又兼顾用户便利。查询职工档案信

息时，自动全屏与分页查询的结合，让大量数据的浏览变得井然有序；模糊查询与排序功能，极大地提高了数据检索的效率和准确性。

新增、修改与删除职工档案信息时，我采用了严谨的数据校验与处理策略，保证信息的完整性与一致性。特别是在修改信息时，对输入流的精准控制，避免了因缓冲区未清空导致的数据混乱。

在运行环境配置上，我选择了 Windows 11 操作系统与 Microsoft Visual Studio 2022 作为开发工具，确保了项目在主流环境下的稳定运行。项目中涉及的“user.txt”与“employee.txt”文件以及音频文件的妥善处理，为系统的正常运行提供了必要的技术支持。

项目总结阶段，我回顾整个开发历程，一次又一次的困难，一次又一次的查阅资料、解决 Bug，深感每一次挑战的克服都是个人技能提升的契机。本项目不仅是对我学习《C++面向对象程序设计》课程结果的检验，也是对我解决复杂问题和创新思维的全方位锻炼。

我期望，“职工档案管理系统”在未来可以结合 EasyX 图形库或 Qt 框架等，实现更加直观、友好的用户界面。更进一步可以结合 HTML、CSS 和 JavaScript 等相关的前端开发技术，可以将“职工档案管理系统”从桌面应用拓展至网络平台，让本系统变得更加实用和灵活。

最后，我期待通过此次项目，向涂老师展现我的努力与成果。

参考文献

- [1] 邵兰洁, 马睿, 徐海云, 母俐丽. (2020). C++面向对象程序设计(第二版). 清华大学出版社.
- [2] 陈炫文, 邹昶, 聂国健, 等. 基于面向对象思想的软件系统分析与设计[J]. 电子产品可靠性与环境试验, 2020, 38(06):24-28.
- [3] 余桂强. 成都市六医院职工档案管理系统设计与实现[D]. 电子科技大学, 2018.
- [4] 田婷. 贵医附院职工档案管理系统的研究与分析[D]. 云南大学, 2015.
- [5] 董引娣. 职工档案管理系统设计[J]. 软件导刊, 2013, 12(12):87-88.

附录

1. Main.cpp

```
1  #include <string>
2  #include "EmployeeProfile.h"
3  #include <iostream>
4  #include <conio.h>
5  #include "User.h"
6  #include <fstream>
7  #include <vector>
8  #include <sstream>
9  #include <Windows.h>
10 #include <iomanip>
11 #include "Constant.h"
12 #include <algorithm>
13 #include <cctype>
14 #include <regex>
15 #include <functional>
16 #include <chrono>
17 #include "Language.h"
18 #include "Chinese.h"
19 #include "English.h"
20 #include <memory>
21 #pragma comment(lib, "Winmm.lib")
22 using namespace std;
23 string EnString = "_ENGLISH";
24 string resultString = "";
25 // 工厂函数，根据需要返回相应的 Language 实例
26 shared_ptr<Language> createLanguage(const string& lang) {
27     if (lang == "_ENGLISH") {
28         return make_shared<English>();
29     } else {
30         // 默认中文或其他逻辑
31         return make_shared<Chinese>();
32     }
33 }
34 // shared_ptr 允许多个对象共享同一个资源的所有权，
35 // 当最后一个指向该资源的 shared_ptr 销毁时，资源会被自动释放
36 shared_ptr<Language> language;
```

```
37  /**
38  * @author XZH
39  */
40  bool login();
41  void myRegister();
42  void forgetPassword();
43  void selectMusic();
44  void menu();
45  void displayAllProfile();
46  void insertEmployeeProfile();
47  void updateEmployeeProfile(string filename);
48  void updateEmployeeProfileById(string filename);
49  void updateEmployeeProfileByIdNumber(string filename);
50  void deleteEmployeeProfile();
51  void deleteEmployeeProfileById(string filename);
52  void deleteEmployeeProfileByIdNumber(string filename);
53  void fuzzyQuery(vector<EmployeeProfile> employeeProfiles);
54  vector<EmployeeProfile> loadEmployeeProfiles(const string& filename);
55  EmployeeProfile createProfileFromLine(const std::string& line);
56  void fullScreen();
57  void closeBlackWindow();
58  void videoPath(wstring basePath);
59  void tableTitle();
60  // 是否选择过语言
61  bool selectLanguage = false;
62  int main() {
63      while (1) {
64          // 如果还没选择语言
65          if (!selectLanguage) {
66              while (1) {
67                  cout << "请选择语言\tPlease select language" << endl
68                      << "1.中文\t\t1.Chinese" << endl
69                      << "2.英文\t\t2.English" << endl;
70                  int languageChoice = _getch();
71                  if (languageChoice == '1') {
72                      std::system("cls");
73                      selectLanguage = true;
74                      // 选择语言对象
75                      language = createLanguage(resultString);
76                      // 是否开启背景音乐
77                      selectMusic();
78                      std::system("cls");
```

```
79         videoPath(L"video\\welcome");
80         break;
81     }
82     else if (languageChoice == '2') {
83         std::system("cls");
84         resultString = EnString;
85         selectLanguage = true;
86         // 选择语言对象
87         language = createLanguage(resultString);
88         // 是否开启背景音乐
89         selectMusic();
90         std::system("cls");
91         videoPath(L"video\\welcome");
92         break;
93     }
94     else {
95         std::system("cls");
96         cout << KEY_ERROR << endl;
97         cout << KEY_ERROR_ENG << endl << endl;
98     }
99 }
100 }
101 // 将语言对象传递给 User
102 User user(language);
103 // 将语言对象传递给 Employee
104 EmployeeProfile employee(language);
105 // 设置黑窗口标题
106 SetConsoleTitleA("职工档案管理系统-XZH");
107 // 设置黑窗口和字体颜色
108 system("color f0");
109 // 显示日期
110 system("date /T");
111 // 显示时间
112 system("TIME /T");
113 cout << language->loginAndRegisterPage() << endl;
114 char first;
115 first = _getch();
116 switch (first) {
117     case '1':
118         std::system("cls");
119         cout << language->welcomeLogin << endl;
120         // 如果登录成功, 进入主界面
```

```
121         if (login()) {
122             // 设置黑窗口和字体颜色
123             system("color 0B");
124             menu();
125             return 0;
126         }
127         break;
128     case '2':
129         std::system("cls");
130         cout << language->welcomeRegister << endl;
131         myRegister();
132         break;
133     case '3':
134         std::system("cls");
135         cout << language->forgetPassword << endl;
136         forgetPassword();
137         break;
138     case 27:
139         cout << language->isExit << endl << language->yesOrNo;
140         if (_getch() == '1') {
141             cout << language->exitSuccess << endl;
142             videoPath(L"video\\bye");
143             closeBlackWindow();
144             // 用 return 0 的话在 meun()中调用 main()之后还会再执行 meun()
145             // 虽然执行了关闭窗口函数，但是如果用户此时鼠标正在移动窗口
146             // 则不会直接关闭窗口
147             exit(0);
148         }
149         std::system("cls");
150         break;
151     default:
152         system("cls");
153         cout << language->KEY_ERROR << endl;
154     }
155 }
156 return 0;
157 }
158 // 登录
159 bool login() {
160     string name, password;
161     string filename = USER_FILENAME;
162     while (1) {
```

```
163     cout << language->inputUserName;
164     cin >> name;
165     // 检查用户名是否存在
166     bool nameExists = User::isUsernameExists(name, filename);
167     if (!nameExists) {
168         //std::system("cls");
169         cout << language->userNameNotExist << endl;
170         return false;
171     }
172     cout << language->inputPassword;
173     password = User::inputPassword();
174     User user(name, password);
175     if (user.isPasswordValid(filename)) {
176         std::system("cls");
177         cout << language->loginSuccess << endl;
178         videoPath(L"video\\login");
179         return true;
180     }
181     else {
182         cout << language->userNameOrPasswordError << endl;
183     }
184 }
185
186 }
187 // 注册
188 void myRegister() {
189     string name, password, confirmPassword, phone;
190     string filename = USER_FILENAME;
191     while (1) {
192         cout << language->inputUserName;
193         cin >> name;
194         // 检查用户名是否存在
195         bool nameExists = User::isUsernameExists(name, filename);
196         if (nameExists) {
197             cout << language->userNameAlreadyExist << endl;
198         }
199         else {
200             break;
201         }
202     }
203     while (1) {
204         cout << language->inputPhone;
```



```
205         cin >> phone;
206         // 如果手机号格式不正确
207         if (!EmployeeProfile::verifyPhone(phone)) {
208             continue;
209         }
210         // 检查手机号是否存在
211         bool phoneExists = User::isPhoneExists(phone, filename);
212         if (phoneExists) {
213             cout << language->phoneAlreadyExist << endl;
214         }
215         else {
216             break;
217         }
218     }
219     while (1) {
220         cout << language->inputPassword;
221         password = User::inputPassword();
222         cout << language->inputConfirmPassword;
223         confirmPassword = User::inputPassword();
224         if (password != confirmPassword) {
225             cout << language->passwordDifferent << endl;
226         }
227         else {
228             break;
229         }
230     }
231     User user(name, password, phone);
232     std::system("cls");
233     // 保存用户信息
234     user.saveUserToFile(filename);
235     cout << language->registerSuccess << endl;
236     videoPath(L"video\\register");
237 }
238 // 选择音乐
239 void selectMusic() {
240     const wstring fileExtension = L".mp3 repeat";
241     const wstring basePath = L"play video\\";
242     cout << language->music << endl << language->yesOrNo2 << endl;
243     int key = _getch();
244     // 不开启背景音乐
245     if (key == '2') {
246         return;
```

```
247     }
248     int i;
249     // 设置随机数生成器的种子
250     srand(time(0));
251     while (1) {
252         i = rand() % 10;
253         if (i <= 5 && i > 0) {
254             break;
255         }
256     }
257     // 拼接字符串
258     wstring filePath = basePath + to_wstring(i) + fileExtension;
259     mciSendString(filePath.c_str(), NULL, 0, NULL);
260 }
261 // 忘记密码
262 void forgetPassword() {
263     string phone, code, VerificationCode;
264     chrono::high_resolution_clock::time_point startTime, endTime;
265     while (1) {
266         cout << language->inputPhone;
267         cin >> phone;
268         if (!EmployeeProfile::verifyPhone(phone)) {
269             continue;
270         }
271         sendVerificationCode:
272         VerificationCode = User::findPhoneAndSendVerificationCode(phone, USER_F
ILENAME);
273         // 记录开始时间
274         startTime = chrono::high_resolution_clock::now();
275         if (VerificationCode != "") {
276             break;
277         }
278     }
279     cout << language->inputCode;
280     while (1) {
281         cin >> code;
282         // 记录结束时间
283         endTime = chrono::high_resolution_clock::now();
284         if (chrono::duration_cast<chrono::seconds>(endTime - startTime).count()
> 3) {
285             cout << language->overtime << endl << language->pressEnter << endl;
```

```
286         int i = _getch();
287         if (i == 13) {
288             goto sendVerificationCode;
289         }
290         break;
291     }
292     if (VerificationCode == code) {
293         User::updateUserByPhone(USER_FILENAME, phone);
294         videoPath(L"video\\password");
295         break;
296     }
297     cout << language->codeError;
298 }
299 }
300 // 功能菜单
301 void menu() {
302     while (1) {
303         language->menu();
304         string filename = EMPLOYEE_FILENAME;
305         char i;
306         i = _getch();
307         switch (i)
308         {
309             case 27:
310                 cout << language->isExit << endl << language->yesOrNo;
311                 if (_getch() == '1') {
312                     std::system("cls");
313                     main();
314                 }
315                 std::system("cls");
316                 break;
317             case '1':
318                 std::system("cls");
319                 displayAllProfile();
320                 break;
321             case '2':
322                 std::system("cls");
323                 insertEmployeeProfile();
324                 break;
325             case '3':
326                 std::system("cls");
327                 updateEmployeeProfile(filename);
```

```
328         break;
329     case '4':
330         std::system("cls");
331         deleteEmployeeProfile();
332         break;
333     default:
334         std::system("cls");
335         cout << language->KEY_ERROR << endl;
336         break;
337     }
338 }
339
340 }
341 // 显示所有职工信息，并提供排序、模糊查询等功能
342 void displayAllProfile() {
343     // 窗口最大化
344     fullScreen();
345     vector<EmployeeProfile> employeeProfiles = loadEmployeeProfiles(EMPLOYEE_FI
LENAME);
346     bool flag = true;
347     while (flag) {
348         cout << language->selectFunction << endl << endl;
349         // 输出表头
350         tableTitle();
351         int startRow = pageNum > 0 ? (pageNum - 1) * pageSize : 0;
352         int endRow = startRow + pageSize * (pageNum > 0 ? 1 : 0);
353         int total = employeeProfiles.size();
354         // 计算出来之后向上取整
355         int sumNum = static_cast<int>(ceil(static_cast<double>(total) / pageSiz
e));
356         // 输出每一个数据
357         for (int i = startRow; i < endRow && i < total; i++) {
358             cout << employeeProfiles[i] << endl;
359         }
360         cout << language->currentPage << pageNum << "/" << sumNum << "\t" << pa
geSize << language->page <<
361             language->leftOrRight << endl
362             << language->totalEmployee << total;
363         int i = _getch();
364         switch (i) {
365             case '1':
366                 std::system("cls");
```

```
367         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
368             // 按工号升序
369             return a.getId() < b.getId();
370         });
371         break;
372     case '2':
373         std::system("cls");
374         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
375             // 按工号降序
376             return a.getId() > b.getId();
377         });
378         break;
379     case '3':
380         std::system("cls");
381         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
382             // 按年龄升序
383             return a.getAge() < b.getAge();
384         });
385         break;
386     case '4':
387         std::system("cls");
388         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
389             // 按年龄序
390             return a.getAge() > b.getAge();
391         });
392         break;
393     case '5':
394         std::system("cls");
395         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
396             // 按入职时间升序
397             return a.getHireDate() < b.getHireDate();
398         });
399         break;
400     case '6':
401         std::system("cls");
402         sort(employeeProfiles.begin(), employeeProfiles.end(), [](EmployeeP
rofile& a, EmployeeProfile& b) {
```

```
403         // 按入职时间降序
404         return a.getHireDate() > b.getHireDate();
405     });
406     break;
407     case '7': {
408         cout << language->inputPageNum;
409         string inputStr;
410         cin >> inputStr;
411         std::system("cls");
412         try {
413             // 检测输入是否可以转成在范围内的数字
414             int input = stoi(inputStr);
415             // 如果输入页码小于等于 0 则为第一页
416             if (input <= 0) {
417                 pageNum = 1;
418             }
419             // 如果大于最大页码最为最后一页
420             else if (input > sumNum) {
421                 pageNum = sumNum;
422             }
423             else {
424                 pageNum = input;
425             }
426         }
427         catch (const invalid_argument& e) {
428             // 捕获输入不是数字的异常
429             cerr << language->inputNum << endl;
430         }
431         catch (const out_of_range& e) {
432             // 转换后的数字超出 int 范围
433             cerr << language->numTooLong << endl;
434         }
435         break;
436     }
437     case '8':
438         cout << language->inputNumPerPage;
439         cin >> pageSize;
440         std::system("cls");
441         // 回到第一页
442         pageNum = 1;
443         break;
444     case '9':
```

```
445         fuzzyQuery(employeeProfiles);
446         std::system("pause");
447         std::system("cls");
448         break;
449     case 27: // ESC 键
450         std::system("cls");
451         flag = false;
452         break;
453     case 77: // 右键
454         std::system("cls");
455         if (pageNum < sumNum) {
456             pageNum++;
457         }
458         break;
459     case 100: // D 键
460         std::system("cls");
461         if (pageNum < sumNum) {
462             pageNum++;
463         }
464         break;
465     case 75: // 左键
466         std::system("cls");
467         if (pageNum > 1) {
468             pageNum--;
469         }
470         break;
471     case 97: // A 键
472         std::system("cls");
473         if (pageNum > 1) {
474             pageNum--;
475         }
476         break;
477     default:
478         std::system("cls");
479         //cout << i;
480         cout << language->KEY_ERROR << endl;
481     }
482 }
483 // 重置为第一页
484 pageNum = PAGE_NUM;
485 // 重置每页展示数
486 pageSize = PAGE_SIZE;
```

```
487     }
488     // 新增职工
489     void insertEmployeeProfile() {
490         EmployeeProfile e;
491         cin >> e;
492         e.saveEmployeeToFile(EMPLOYEE_FILENAME);
493     }
494     // 修改职工信息
495     void updateEmployeeProfile(string filename) {
496         while (1) {
497             cout << language->updateByIdOrIdNumber << endl;
498             char i = _getch();
499             if (i == '1') {
500                 std::system("cls");
501                 updateEmployeeProfileById(EMPLOYEE_FILENAME);
502             }
503             else if (i == '2') {
504                 std::system("cls");
505                 updateEmployeeProfileByIdNumber(EMPLOYEE_FILENAME);
506             }
507             else if (i == 27) {
508                 std::system("cls");
509                 break;
510             }
511             else {
512                 cout << language->inputOneOrTwo << endl;
513             }
514         }
515     }
516     // 通过工号更新职工信息
517     void updateEmployeeProfileById(string filename) {
518         EmployeeProfile::updateProfileById(filename);
519     }
520     // 通过身份证号更新职工信息
521     void updateEmployeeProfileByIdNumber(string filename) {
522         EmployeeProfile::updateProfileByIdNumber(filename);
523     }
524     // 删除员工
525     void deleteEmployeeProfile() {
526         while (1) {
527             cout << language->deleteByIdOrIdNumber << endl;
528             char i = _getch();
```



```
529         if (i == '1') {
530             std::system("cls");
531             deleteEmployeeProfileById(EMPLOYEE_FILENAME);
532         }
533         else if (i == '2') {
534             std::system("cls");
535             deleteEmployeeProfileByIdNumber(EMPLOYEE_FILENAME);
536         }
537         else if (i == 27) {
538             std::system("cls");
539             break;
540         }
541         else {
542             cout << language->inputOneOrTwo << endl;
543         }
544     }
545 }
546 // 通过工号删除
547 void deleteEmployeeProfileById(string filename) {
548     EmployeeProfile::deleteProfileById(filename);
549 }
550 // 通过身份证号删除
551 void deleteEmployeeProfileByIdNumber(string filename) {
552     EmployeeProfile::deleteProfileByIdNumber(filename);
553 }
554 // 模糊查询
555 void fuzzyQuery(vector<EmployeeProfile> employeeProfiles) {
556     chrono::high_resolution_clock::time_point startTime, endTime;
557     string content;
558     // 记录查询结果数
559     int count = 0;
560     cout << language->inputSelectContent;
561     cin >> content;
562     std::system("cls");
563     // 转义 content 中的特殊正则表达式字符
564     string escapedContent = regex_replace(content, regex(R"([\^\$.|?*(+)])" ), R"
(\&)");
565     regex reg(escapedContent, regex_constants::icase);
566     // 输出表头
567     tableTitle();
568     // 记录开始时间
569     startTime = chrono::high_resolution_clock::now();
```

```

570     // 逐一匹配
571     for (int i = 0; i < employeeProfiles.size(); i++) {
572         if (regex_search(employeeProfiles[i].getId(), reg) ||
573             regex_search(employeeProfiles[i].getName(), reg) ||
574             regex_search(employeeProfiles[i].getIdNumber(), reg) ||
575             regex_search(employeeProfiles[i].getGender(), reg) ||
576             regex_search(to_string(employeeProfiles[i].getAge()), reg) ||
577             regex_search(employeeProfiles[i].getPhoneNumber(), reg) ||
578             regex_search(employeeProfiles[i].getAddress(), reg) ||
579             regex_search(employeeProfiles[i].getEducation(), reg) ||
580             regex_search(employeeProfiles[i].getPosition(), reg) ||
581             regex_search(employeeProfiles[i].getHireDate(), reg) ||
582             regex_search(employeeProfiles[i].getDepartment(), reg)) {
583             // 打印匹配到的员工信息
584             cout << employeeProfiles[i] << endl;
585             count++;
586         }
587     }
588     // 记录结束时间
589     endTime = chrono::high_resolution_clock::now();
590     cout << language->selectTime
591         << chrono::duration_cast<chrono::milliseconds>(endTime - startTime).count() << " 毫秒" << endl
592         << language->totalEmployee << count << endl;
593 }
594 // 从文件中获取每一行的内容，并构建成一个 EmployeeProfile 对象数组
595 vector<EmployeeProfile> loadEmployeeProfiles(const string& filename) {
596     vector<EmployeeProfile> profiles;
597     ifstream inFile(filename);
598     if (inFile.is_open()) {
599         std::string line;
600         while (std::getline(inFile, line)) {
601             // 确保行不为空
602             if (!line.empty()) {
603                 EmployeeProfile profile = createProfileFromLine(line);
604                 profiles.push_back(profile);
605             }
606         }
607         inFile.close();
608     }
609     else {
610         cerr << language->canNotOpen << filename << std::endl;

```

```
611     }
612     return profiles;
613 }
614 // 从一行数据中获取每个属性值，并构建 EmployeeProfile 对象
615 EmployeeProfile createProfileFromLine(const std::string& line) {
616     istringstream iss(line);
617     // 定义 EmployeeProfile 包含的成员变量
618     string id, name, idNumber, gender, ageStr, phoneNumber, address, education,
        position, hireDate, department;
619     int age;
620     // 每行中的每个数据以逗号分隔，解析字符串
621     getline(iss, id, ',');
622     getline(iss, name, ',');
623     getline(iss, idNumber, ',');
624     getline(iss, gender, ',');
625     // 对于整数非字符串类型，需要额外的转换
626     getline(iss, ageStr, ',');
627     getline(iss, phoneNumber, ',');
628     getline(iss, address, ',');
629     getline(iss, education, ',');
630     getline(iss, position, ',');
631     getline(iss, hireDate, ',');
632     getline(iss, department);
633     // 将字符串 ageStr 转化为 int 类型 age
634     age = stoi(ageStr);
635     // 创建 EmployeeProfile 对象
636     return EmployeeProfile(id, name, idNumber, gender, age, phoneNumber, address, education, position, hireDate, department);
637 }
638 // 窗口最大化
639 void fullScreen() {
640     // 按下 Alt 键
641     keybd_event(VK_MENU, 0, 0, 0);
642     // 按下空格键
643     keybd_event(VK_SPACE, 0, 0, 0);
644     // 按下 X 键
645     keybd_event(0x58, 0, 0, 0);
646     // 释放 X 键
647     keybd_event(0x58, 0, KEYEVENTF_KEYUP, 0);
648     // 释放空格键
649     keybd_event(VK_SPACE, 0, KEYEVENTF_KEYUP, 0);
650     // 释放 Alt 键
```

```
651     keybd_event(VK_MENU, 0, KEYEVENTF_KEYUP, 0);
652 }
653 // 关闭窗口
654 void closeBlackWindow() {
655     cout << language->close;
656     Sleep(1000);
657     cout << ".";
658     Sleep(1000);
659     cout << ".";
660     Sleep(1000);
661
662     // 按下 Alt 键
663     keybd_event(VK_MENU, 0, 0, 0);
664     // 按下 F4 键
665     keybd_event(VK_F4, 0, 0, 0);
666     // 释放 Alt 键
667     keybd_event(VK_MENU, 0, KEYEVENTF_KEYUP, 0);
668     // 释放 F4 键
669     keybd_event(VK_F4, 0, KEYEVENTF_KEYUP, 0);
670 }
671 // 播放不同语言的音频
672 void videoPath(wstring basePath) {
673     const wstring fileExtension = L".wav";
674     // 拼接字符串
675     wstring filePath = basePath
676         + wstring(resultString.begin(), resultString.end())
677         + fileExtension;
678     // 播放音频
679     PlaySound(filePath.c_str(), NULL, SND_FILENAME | SND_ASYNC);
680 }
681 // 打印表头
682 void tableTitle() {
683     const int SHORT_WIDTH = 10; // 短列宽
684     const int MIDDLE_WIDTH = 15; // 中列宽
685     const int LONG_WIDTH = 20; // 长列宽
686     const int LINE_LENGTH = 155; // 分割线长度
687     // 输出表头
688     language->tableTitle();
689     cout << string(LINE_LENGTH, '-') << endl;
690 }
```

2. User.h

```
691     #pragma once
692     #include <string>
693     #include "Language.h"
694     #include <memory>
695
696     using namespace std;
697     /**
698      * @author XZH
699      */
700     class User
701     {
702     private:
703         // 用户名
704         string name;
705         // 密码
706         string password;
707         // 手机号
708         string phone;
709         // userLanguage 是一个智能指针，用于管理 Language 类的对象
710         static shared_ptr<Language> userLanguage;
711         // 通过第 i 项内容查询并更新
712         static void updateUserByI(const string& filename, const string& phone, int
i);
713     public:
714         // 无参构造函数
715         User() = default;
716         // 有参构造函数
717         User(string na, string pa, string ph);
718         User(string na, string pa);
719         // 用于初始化语言的构造函数
720         User(shared_ptr<Language>& langPtr);
721
722         // 成员变量对应的 get 函数
723         string getName();
724         string getPassword();
725         // 将用户信息保存至文件中
726         void saveUserToFile(const string& filename);
727         // 判断用户名和密码是否匹配
728         bool isPasswordValid(const string& filename);
```

```

729         // -----static-----
730         // 判断是否存在该用户名
731         static bool isUsernameExists(const std::string& username, const std::string
& filename);
732         // 判断是否存在该手机号
733         static bool isPhoneExists(const std::string& phone, const string& filename)
;
734         // 手机号存在时发送验证码
735         static string findPhoneAndSendVerificationCode(const string& phone, const s
tring& filename);
736         // 根据手机号更新职工
737         static void updateUserByPhone(const string& filename, const string& phone);

738         // 输入密码
739         static string inputPassword();
740     };

```

3. User.cpp

```

741     #include "User.h"
742     #include <fstream>
743     #include <iostream>
744     #include <string>
745     #include "Constant.h"
746     #include <sstream>
747     #include <vector>
748     #include "MD5.h"
749     #include <conio.h>
750     #include <windows.h>
751
752     using namespace std;
753     /**
754      * @author XZH
755      */
756     // 初始化静态成员变量
757     shared_ptr<Language> User::userLanguage = nullptr;
758     // 通过第 i 项内容查询并更新
759     void User::updateUserByI(const string& filename, const string& phone, int i)
760     {
761         ifstream inFile(filename);
762         if (!inFile.is_open()) {

```

```
763         cerr << userLanguage->canNotOpen << endl;
764         return;
765     }
766     // 创建临时文件来保存修改后的数据
767     string tempFilename = "temp_" + filename;
768     ofstream outFile(tempFilename);
769     if (!outFile.is_open()) {
770         cerr << userLanguage->canCreateTemp << endl;
771         inFile.close();
772         return;
773     }
774     string line;
775     while (getline(inFile, line)) {
776         istringstream iss(line);
777         string currentProperty;
778         // 读取每行的第 i 个字段（工号）
779         int tempI = i;
780         while (tempI--) {
781             getline(iss, currentProperty, ',');
782         }
783         if (currentProperty != phone) {
784             // 如果当前行的属性不是要修改的属性，则写入临时文件
785             outFile << line << endl;
786         }
787         else {
788             // 使用 vector 来动态存储分割后的字符串
789             vector<string> userVector;
790             stringstream ss(line);
791             string item;
792             // 使用 getline 按逗号分割字符串，直到读取完所有元素
793             while (getline(ss, item, ',')) {
794                 // 将分割得到的元素添加到 vector 中
795                 userVector.push_back(item);
796             }
797             cout << userLanguage->userName << userVector[USER_NAME_SERIAL_NUMBE
R - 1] << endl;
798
799             string password, confirmPassword;
800             while (1) {
801                 cout << userLanguage->inputNewPassword;
802                 password = inputPassword();
803                 cout << userLanguage->inputConfirmPassword;
```

```
804         confirmPassword = inputPassword();
805         if (password != confirmPassword) {
806             cout << userLanguage->passwordDifferent << endl;
807         }
808         else {
809             break;
810         }
811     }
812
813     userVector[USER_PASSWORD_SERIAL_NUMBER - 1] = password;
814     // 拼接新的用户信息
815     string newLine = "";
816     for (int j = 0; j < userVector.size(); j++) {
817         newLine += userVector[j] + ",";
818     }
819     // 写入文件
820     outFile << newLine << endl;
821 }
822 }
823 inFile.close();
824 outFile.close();
825
826 // 删除原文件
827 remove(filename.c_str());
828 // 将临时文件重命名为原文件名
829 rename(tempFilename.c_str(), filename.c_str());
830 cout << userLanguage->phone << phone << userLanguage->updatePasswordSuccess
    << endl;
831 }
832 User::User(string na, string pa, string ph) : name(na), password(pa), phone(ph)
833 {
834 }
835 User::User(string na, string pa) : name(na), password(pa)
836 {
837 }
838 User::User(shared_ptr<Language>& langPtr)
839 {
840     userLanguage = langPtr;
841 }
842
843 string User::getName() {
```



```
844     return name;
845 }
846 string User::getPassword() {
847     return password;
848 }
849
850 // 将用户信息保存至文件中
851 void User::saveUserToFile(const string& filename) {
852     ofstream outFile(filename, ios_base::app);
853     if (!outFile.is_open()) {
854         cerr << userLanguage->canNotOpen << filename << endl;
855         return;
856     }
857     // 将用户名和密码写入文件，中间用逗号分开，便于之后解析
858     outFile << name << "," << password << "," << phone << endl;
859     outFile.close();
860     cout << userLanguage->saveUser << filename << endl;
861 }
862 // 判断用户名和密码是否匹配
863 bool User::isPasswordValid(const string& filename)
864 {
865     ifstream inFile(filename);
866     if (!inFile.is_open()) {
867         cerr << userLanguage->canNotOpen << filename << endl;
868         // 文件不存在或无法打开，无法验证
869         return false;
870     }
871     string line;
872     while (getline(inFile, line)) {
873         // 查找第一个逗号的位置
874         size_t pos1 = line.find(',');
875         if (pos1 != string::npos) {
876             // 获取第一个数据（用户名）
877             string storedUsername = line.substr(0, pos1);
878             // 从第一个逗号之后开始查找第二个逗号的位置
879             size_t pos2 = line.find(',', pos1 + 1);
880             if (pos2 != string::npos) {
881                 // 获取第二个数据（密码）
882                 string storedPassword = line.substr(pos1 + 1, pos2 - pos1 - 1);
883
884                 // 校验用户名和密码
885                 if (storedUsername == name && storedPassword == password) {
```

```
885         inFile.close();
886         // 用户名和密码匹配
887         return true;
888     }
889 }
890 }
891 }
892 inFile.close();
893 // 未找到匹配的用户名密码组合
894 return false;
895 }
896 // 判断是否存在该用户名
897 bool User::isUsernameExists(const string& username, const string& filename) {
898     ifstream inFile(filename);
899     if (!inFile.is_open()) {
900         cerr << userLanguage->canNotOpen << filename << endl;
901         // 文件不存在或无法打开, 视为用户名不存在
902         return false;
903     }
904     string line;
905     while (getline(inFile, line)) {
906         size_t pos = line.find(',');
907         if (pos != string::npos && line.substr(0, pos) == username) {
908             inFile.close();
909             // 用户名已存在
910             return true;
911         }
912     }
913     inFile.close();
914     // 读取完所有行, 未找到匹配的用户名
915     return false;
916 }
917 // 判断是否存在该手机号
918 bool User::isPhoneExists(const string& phone, const string& filename)
919 {
920     ifstream inFile(filename);
921     if (!inFile.is_open()) {
922         cerr << userLanguage->canNotOpen << filename << endl;
923         return false; // 文件不存在或无法打开, 视为用户名不存在
924     }
925     string line;
926     while (getline(inFile, line)) {
```

```
927         istream iss(line);
928         string currentProperty;
929         // 读取每行的 phone 属性
930         int tempI = USER_PHONE_SERIAL_NUMBER;
931         while (tempI--) {
932             getline(iss, currentProperty, ',');
933         }
934         if (currentProperty == phone) {
935             inFile.close();
936             // 手机号已存在
937             return true;
938         }
939     }
940     inFile.close();
941     // 读取完所有行，未找到匹配的用户名
942     return false;
943 }
944 // 手机号存在时发送验证码
945 string User::findPhoneAndSendVerificationCode(const string& phone, const string
& filename)
946 {
947     if (isPhoneExists(phone, filename)) {
948         srand(time(0));
949         // 生成一个四位数的验证码
950         int verificationCode = rand() % 9000 + 1000;
951         cout << userLanguage->wait << endl;
952         Sleep(1000);
953         cout << userLanguage->sending << endl;
954         Sleep(1000);
955         cout << userLanguage->sended << phone << userLanguage->sendCode <<
verificationCode << userLanguage->timeValid << endl;
956         return to_string(verificationCode);
957     }
958 }
959 else {
960     cout << userLanguage->phoneNotRegister << endl;
961 }
962 // 未找到对应用户
963 return "";
964 }
965 // 根据手机号更新职工
966 void User::updateUserByPhone(const string& filename, const string& phone)
967 {
```

```
968     updateUserByI(filename, phone, USER_PHONE_SERIAL_NUMBER);
969 }
970 // 输入密码
971 string User::inputPassword() {
972     char password[21]; // 20 位密码加上字符串结束符
973     int index = 0;
974     while (1) {
975         char ch;
976         ch = _getch();
977         if (ch == 8) { // 退格键
978             if (index != 0) {
979                 cout << "\b \b"; // 删除一个字符
980                 index--;
981             }
982         }
983         else if (ch == '\r') { // 回车键
984             password[index] = '\0';
985             cout << endl;
986             if (index < 6) {
987                 cout << userLanguage->passwordShort;
988                 index = 0; // 重置密码输入位置
989             }
990             else {
991                 break;
992             }
993         }
994         else {
995             if (index < 20) { // 密码长度限制为 20 位
996                 cout << "*";
997                 password[index++] = ch;
998             }
999             else {
1000                 cout << userLanguage->passwordLong;
1001                 index = 0; // 重置密码输入位置
1002             }
1003         }
1004     }
1005     string passwordStr(password);
1006     MD5 md5;
1007     return md5.calculate(passwordStr);
1008 }
```

4. EmployeeProfile.h

```

1009  #pragma once
1010  #include <string>
1011  #include "Language.h"
1012  #include <memory>
1013  using namespace std;
1014  /**
1015   * @author XZH
1016   */
1017  class EmployeeProfile
1018  {
1019  private:
1020      // 成员变量
1021      string id;           // 工号
1022      string name;        // 姓名
1023      string idNumber;    // 身份证号
1024      string gender;      // 性别
1025      int age;            // 年龄
1026      string phoneNumber; // 联系电话
1027      string address;     // 家庭地址
1028      string education;   // 学历
1029      string position;    // 职位
1030      string hireDate;    // 入职日期
1031      string department;  // 所属部门
1032      // EmployeeLanguage 是一个智能指针，用于管理 Language 类的对象
1033      static shared_ptr<Language> employeeLanguage;
1034      // 通过第 i 项内容删除
1035      static void deleteProfileByI(const string& filename, int i);
1036      // 通过第 i 项内容查询并更新
1037      static void updateProfileByI(const string& filename, int i);
1038  public:
1039      // 无参构造函数
1040      EmployeeProfile() = default;
1041      // 有参构造函数
1042      EmployeeProfile(
1043          string id,
1044          string name,
1045          string idNumber,
1046          string gender,
1047          int age,

```

```
1048         string phoneNumber,
1049         string address,
1050         string education,
1051         string position,
1052         string hireDate,
1053         string department
1054     );
1055     // 用于初始化语言的构造函数
1056     EmployeeProfile(shared_ptr<Language>& langPtr);
1057     // 成员变量对应的 get 函数
1058     string getId();
1059     string getName();
1060     string getIdNumber();
1061     string getGender();
1062     int getAge();
1063     string getPhoneNumber();
1064     string getAddress();
1065     string getEducation();
1066     string getPosition();
1067     string getHireDate();
1068     string getDepartment();
1069
1070     // 重载 >> 运算符
1071     friend istream& operator>>(istream& in, EmployeeProfile& profile);
1072     // 重载 << 运算符
1073     friend ostream& operator<<(ostream& out, EmployeeProfile& profile);
1074     // 将信息存入文件
1075     void saveEmployeeToFile(const string& filename);
1076     // -----static-----
1077     // 通过工号删除
1078     static void deleteProfileById(const string& filename);
1079     // 通过身份证号删除
1080     static void deleteProfileByIdNumber(const string& filename);
1081     // 根据工号更新职工
1082     static void updateProfileById(const string& filename);
1083     // 根据身份证号更新职工
1084     static void updateProfileByIdNumber(const string& filename);
1085     // 判断某个成员变量是否存在, i 表示要查询的成员变量的序号
1086     static bool isPropertyExists(const string& property, const string& filename
, const int i);
1087     // 验证手机号是否正确
1088     static bool verifyPhone(string phone);
```

```
1089     };
```

5. EmployeeProfile.cpp

```
1090     #include "EmployeeProfile.h"
1091     #include <iostream>
1092     #include <iomanip>
1093     #include <fstream>
1094     #include <conio.h>
1095     #include <sstream>
1096     #include <vector>
1097     #include "Constant.h"
1098     // #include <regex>
1099     #include <ctime>
1100     using namespace std;
1101     shared_ptr<Language> EmployeeProfile::employeeLanguage = nullptr;
1102     /**
1103      * @author XZH
1104      */
1105     // 通过第 i 个属性删除职工
1106     void EmployeeProfile::deleteProfileByI(const string& filename, int i)
1107     {
1108         string id;
1109         cin >> id;
1110         ifstream inFile(filename);
1111         if (!inFile.is_open()) {
1112             cerr << employeeLanguage->canNotOpen << endl;
1113             return;
1114         }
1115         // 创建临时文件来保存修改后的数据
1116         string tempFilename = "temp_" + filename;
1117         // 创建的就是新的空文件，不用 ios_base::app 追加形式
1118         ofstream outFile(tempFilename);
1119         if (!outFile.is_open()) {
1120             cerr << employeeLanguage->canCreateTemp << endl;
1121             inFile.close();
1122             return;
1123         }
1124         string line;
1125         // 标记是否找到并删除了指定的记录
1126         bool found = false;
```

```
1127     while (getline(inFile, line)) {
1128         istream iss(line);
1129         string currentId;
1130         // 读取每行的第 i 个字段
1131         int tempI = i;
1132         while (tempI--) {
1133             getline(iss, currentId, ',');
1134         }
1135
1136         if (currentId != id) {
1137             // 如果当前行的属性不是要删除的属性，则写入临时文件
1138             outFile << line << endl;
1139         }
1140         else {
1141             // 找到了匹配的工号
1142             found = true;
1143             // 此处进行 break 的话会导致要删除职工后面的职工没有写入新文件
1144         }
1145     }
1146     inFile.close();
1147     outFile.close();
1148     if (found) {
1149         // 如果找到了匹配的属性，删除原文件
1150         remove(filename.c_str());
1151         // 将临时文件重命名为原文件名
1152         rename(tempFilename.c_str(), filename.c_str());
1153         if (i == EMPLOYEE_ID_SERIAL_NUMBER) {
1154             cout << employeeLanguage->employeeId << id << employeeLanguage->employeeAlreadyDel << endl;
1155         }
1156         else {
1157             cout << employeeLanguage->idNum << id << employeeLanguage->employeeAlreadyDel << endl;
1158         }
1159     }
1160     else {
1161         // 如果没找到匹配项，删除临时文件
1162         remove(tempFilename.c_str());
1163         if (i == EMPLOYEE_ID_SERIAL_NUMBER) {
1164             cout << employeeLanguage->employeeIdNotFound << id << employeeLanguage->employeeRecord << endl;
1165         }
1166     }
```



```
1166         else {
1167             cout << employeeLanguage->idNumNotFound << id << employeeLanguage->
employeeRecord << endl;
1168         }
1169     }
1170 }
1171 // 通过工号删除
1172 void EmployeeProfile::deleteProfileById(const string& filename)
1173 {
1174     cout << employeeLanguage->inputEmployeeId;
1175     deleteProfileByI(filename, EMPLOYEE_ID_SERIAL_NUMBER);
1176 }
1177 // 通过身份证号删除
1178 void EmployeeProfile::deleteProfileByIdNumber(const string& filename)
1179 {
1180     cout << employeeLanguage->inputIdNum;
1181     deleteProfileByI(filename, EMPLOYEE_ID_NUMBER_SERIAL_NUMBER);
1182 }
1183 // 将信息存入文件
1184 void EmployeeProfile::saveEmployeeToFile(const string& filename)
1185 {
1186     // ios_base::app 为追加模式打开文件
1187     ofstream outFile(filename, ios_base::app);
1188     if (!outFile.is_open()) {
1189         cerr << employeeLanguage->canNotOpen << filename << endl;
1190         return;
1191     }
1192     // 在此再次判断, 如果工号存在则不保存到文件中
1193     if (EmployeeProfile::isPropertyExists(id, EMPLOYEE_FILENAME, EMPLOYEE_ID_SE
RIAL_NUMBER)) {
1194         return;
1195     }
1196     // 在此再次判断, 如果身份证号存在则不保存到文件中
1197     if (EmployeeProfile::isPropertyExists(idNumber, EMPLOYEE_FILENAME, EMPLOYEE
_ID_NUMBER_SERIAL_NUMBER)) {
1198         return;
1199     }
1200     // 将职工信息写入文件, 中间用逗号分开, 便于之后解析
1201     outFile << id << "," << name << "," << idNumber << "," << gender << "," <<
age << ","
1202         << phoneNumber << "," << address << "," << education <<
1203         << "," << position << "," << hireDate << "," << department << endl;
```

```
1204     outFile.close();
1205     cout << employeeLanguage->employeeSaveTo << filename << endl;
1206 }
1207 // 通过第 i 项内容查询并更新
1208 void EmployeeProfile::updateProfileByI(const string& filename, int i)
1209 {
1210     string id;
1211     cin >> id;
1212     ifstream inFile(filename);
1213     if (!inFile.is_open()) {
1214         cerr << employeeLanguage->canNotOpen << endl;
1215         return;
1216     }
1217     // 创建临时文件来保存修改后的数据
1218     string tempFilename = "temp_" + filename;
1219     // 创建的就是新的空文件，不用 ios_base::app 追加形式
1220     ofstream outFile(tempFilename);
1221     if (!outFile.is_open()) {
1222         cerr << employeeLanguage->canCreateTemp << endl;
1223         inFile.close();
1224         return;
1225     }
1226     string line;
1227     // 标记是否找到并修改了指定的记录
1228     bool found = false;
1229     while (getline(inFile, line)) {
1230         istringstream iss(line);
1231         string currentId;
1232         // 读取每行的第 i 个字段（工号）
1233         int tempI = i;
1234         while (tempI--) {
1235             getline(iss, currentId, ',');
1236         }
1237         if (currentId != id) {
1238             // 如果当前行的属性不是要修改的属性，则写入临时文件
1239             outFile << line << endl;
1240         }
1241         else {
1242             // 找到了匹配的工号
1243             found = true;
1244             // 使用 vector 来动态存储分割后的字符串
1245             vector<string> employeeVector;
```

```
1246         stringstream ss(line);
1247         string item;
1248         // 使用 getline 按逗号分割字符串，直到读取完所有元素
1249         while (getline(ss, item, ',')) {
1250             // 将分割得到的元素添加到 vector 中
1251             employeeVector.push_back(item);
1252         }
1253         string newE;
1254         for (int j = 0; j < employeeVector.size(); j++) {
1255             if (j == 0) {
1256                 cout << employeeLanguage->ratainOldVal << employeeVector[0]
1257                 << endl
1258                 << employeeLanguage->catNotUpdate << endl << endl;
1259                 j++;
1260                 // 清除输入缓冲区
1261                 cin.ignore();
1262             }
1263             cout << employeeLanguage->oldVal << employeeVector[j] << endl <
1264             < employeeLanguage->newVal;
1265             // 读取用户输入的字符
1266             char input = cin.get();
1267             // 如果不是回车键
1268             if (input != '\n') {
1269                 // 将读取的字符放回输入流
1270                 cin.unget();
1271                 // 进行输入操作
1272                 cin >> newE;
1273                 // 对身份证号进行校验
1274                 if (j == EMPLOYEE_ID_NUMBER_SERIAL_NUMBER - 1) {
1275                     while (1) {
1276                         // 如果格式有误则重新输入
1277                         if (!regex_match(newE, idNumberPattern)) {
1278                             cout << employeeLanguage->idNumFormatError;
1279                             cin >> newE;
1280                             continue;
1281                         }
1282                         // 身份证号已存在
1283                         if (EmployeeProfile::isPropertyExists(newE, EMPLOYEE_FILENAME, EMPLOYEE_ID_NUMBER_SERIAL_NUMBER)) {
1284                             cout << employeeLanguage->idNumAlreadyExist;
1285                             cin >> newE;
1286                         }
1287                     }
1288                 }
1289             }
1290         }
1291     }
```

```
1285         else {
1286             break;
1287         }
1288     }
1289 }
1290 // 对手机号进行校验
1291 if (j == EMPLOYEE_PHONE_SERIAL_NUMBER - 1) {
1292     while (1) {
1293         // 如果格式有误则重新输入
1294         if (!verifyPhone(newE)){
1295             cout << employeeLanguage->reinputPhone;
1296             cin >> newE;
1297             continue;
1298         }
1299         // 手机号号已存在
1300         if (EmployeeProfile::isPropertyExists(newE, EMPLOYEE_FILENAME, EMPLOYEE_PHONE_SERIAL_NUMBER)) {
1301             cout << employeeLanguage->employeePhoneAlreadyExist;
1302             cin >> newE;
1303         }
1304         else {
1305             break;
1306         }
1307     }
1308 }
1309 // 更新数据
1310 employeeVector[j] = newE;
1311 // 清除输入缓冲区
1312 cin.ignore();
1313 cout << endl;
1314 }
1315 else {
1316     cout << endl;
1317 }
1318 }
1319 // 拼接新的职工信息
1320 string newLine = "";
1321 for (int j = 0; j < employeeVector.size(); j++) {
1322     // 如果不是最后一个则后面加逗号
1323     if (j != employeeVector.size() - 1) {
1324         newLine += employeeVector[j] + ",";
```

```
1325         }
1326     else {
1327         newLine += employeeVector[j];
1328     }
1329 }
1330 // 写入文件
1331 outFile << newLine << endl;
1332 }
1333 }
1334 inFile.close();
1335 outFile.close();
1336 if (found) {
1337     // 如果找到了匹配的属性，删除原文件
1338     remove(filename.c_str());
1339     // 将临时文件重命名为原文件名
1340     rename(tempFilename.c_str(), filename.c_str());
1341     if (i == EMPLOYEE_ID_SERIAL_NUMBER) {
1342         cout << employeeLanguage->employeeId << id << employeeLanguage->emp
1343         loyeeUpdateSuccess << endl;
1344     }
1345     else {
1346         cout << employeeLanguage->idNum << id << employeeLanguage->employee
1347         UpdateSuccess << endl;
1348     }
1349 }
1350 else {
1351     // 如果没找到匹配项，删除临时文件
1352     remove(tempFilename.c_str());
1353     if (i == EMPLOYEE_ID_SERIAL_NUMBER) {
1354         cout << employeeLanguage->employeeIdNotFound << id << employeeLangu
1355         age->employeeRecord << endl;
1356     }
1357     else {
1358         cout << employeeLanguage->employeeIdNotFound << id << employeeLangu
1359         age->employeeRecord << endl;
1360     }
1361 }
1362 }
1363 // 根据工号更新职工 phonePattern
1364 void EmployeeProfile::updateProfileById(const string& filename)
1365 {
1366     cout << employeeLanguage->inputEmployeeId;
```

```
1363     updateProfileByI(filename, EMPLOYEE_ID_SERIAL_NUMBER);
1364 }
1365 // 根据身份证号更新职工
1366 void EmployeeProfile::updateProfileByIdNumber(const string& filename)
1367 {
1368     cout << employeeLanguage->inputIdNum;
1369     updateProfileByI(filename, EMPLOYEE_ID_NUMBER_SERIAL_NUMBER);
1370 }
1371 // 判断某个成员变量是否存在, i 表示要查询的成员变量的序号
1372 bool EmployeeProfile::isPropertyExists(const string& property, const string& filename, const int i)
1373 {
1374     ifstream inFile(filename);
1375     if (!inFile.is_open()) {
1376         cerr << employeeLanguage->canNotOpen << filename << endl;
1377         // 文件不存在或无法打开, 视为用户名不存在
1378         return false;
1379     }
1380     string line;
1381     while (getline(inFile, line)) {
1382         istringstream iss(line);
1383         string currentProperty;
1384         // 读取每行的第 i 个字段
1385         int tempI = i;
1386         while (tempI--) {
1387             getline(iss, currentProperty, ',');
1388         }
1389         if (currentProperty == property && i == EMPLOYEE_ID_SERIAL_NUMBER) {
1390             inFile.close();
1391             return true;
1392         }
1393         else if (currentProperty == property && i == EMPLOYEE_ID_NUMBER_SERIAL_NUMBER) {
1394             inFile.close();
1395             return true;
1396         }
1397         else if (currentProperty == property && i == EMPLOYEE_PHONE_SERIAL_NUMBER) {
1398             inFile.close();
1399             return true;
1400         }
1401     }
```

```

1402     inFile.close();
1403     // 读取完所有行，未找到匹配的工号
1404     return false;
1405 }
1406 // 验证手机号是否正确
1407 bool EmployeeProfile::verifyPhone(string phone) {
1408     // 如果手机号格式不正确
1409     if (!regex_match(phone, phonePattern)) {
1410         cout << employeeLanguage->phoneFormatError << endl;
1411         return false;
1412     }
1413 }
1414 // 重载 >> 运算符
1415 istream& operator>>(istream& in, EmployeeProfile& profile) {
1416     while (1) {
1417         cout << EmployeeProfile::employeeLanguage->employeeIdFormat;
1418         in >> profile.id;
1419         // 如果格式有误则重新输入
1420         if (profile.id.size() == 10) {
1421             // 提取前四位
1422             string yearStr = profile.id.substr(0, 4);
1423             // 转换为 int
1424             int yearInt = stoi(yearStr);
1425             // 获取当前时间
1426             time_t now = time(0);
1427             // 定义一个 tm 结构体变量来保存转换后的时间
1428             tm localTimeStruct;
1429             // 使用 localtime_s 进行安全的转换
1430             localtime_s(&localTimeStruct, &now);
1431             // 获取年份，tm_year 是从 1900 年开始的，所以需要加 1900
1432             int currentYear = localTimeStruct.tm_year + 1900;
1433             if (yearInt < 1956 || yearInt > currentYear) {
1434                 cout << EmployeeProfile::employeeLanguage->employeeIdFormatError
1435                 r << endl;
1436                 continue;
1437             }
1438             else {
1439                 cout << EmployeeProfile::employeeLanguage->employeeIdFormatError <<
1440                 endl;
1441                 continue;
1442             }
1443         }
1444     }
1445 }

```

```
1442         // 工号已存在
1443         if (EmployeeProfile::isPropertyExists(profile.id, EMPLOYEE_FILENAME, EM
1444             PLOYEE_ID_SERIAL_NUMBER)) {
1445             cout << EmployeeProfile::employeeLanguage->employeeIdAlreadyExist <
1446             < endl;
1447         }
1448         else {
1449             break;
1450         }
1451         cout << EmployeeProfile::employeeLanguage->employeeName;
1452         in >> profile.name;
1453         while (1) {
1454             cout << EmployeeProfile::employeeLanguage->inputIdNum2;
1455             in >> profile.idNumber;
1456             // 如果格式有误则重新输入
1457             if (!regex_match(profile.idNumber, idNumberPattern)) {
1458                 cout << EmployeeProfile::employeeLanguage->idNumFormatError2 << end
1459                 l;
1460                 continue;
1461             }
1462             // 身份证号已存在
1463             if (EmployeeProfile::isPropertyExists(profile.idNumber, EMPLOYEE_FILENA
1464             ME, EMPLOYEE_ID_NUMBER_SERIAL_NUMBER)) {
1465                 cout << EmployeeProfile::employeeLanguage->idNumAlreadyExist2 << en
1466                 dl;
1467             }
1468             else {
1469                 break;
1470             }
1471             cout << EmployeeProfile::employeeLanguage->gender;
1472             in >> profile.gender;
1473             cout << EmployeeProfile::employeeLanguage->age;
1474             in >> profile.age;
1475             while (1) {
1476                 cout << EmployeeProfile::employeeLanguage->employeePhone;
1477                 in >> profile.phoneNumber;
1478                 // 如果格式有误则重新输入
1479                 if (!EmployeeProfile::verifyPhone(profile.phoneNumber)) {
1480                     continue;
1481                 }
1482             }
1483         }
```



```
1479         // 身份证号已存在
1480         if (EmployeeProfile::isPropertyExists(profile.phoneNumber, EMPLOYEE_FIL
ENAME, EMPLOYEE_PHONE_SERIAL_NUMBER)) {
1481             cout << EmployeeProfile::employeeLanguage->phoneAlreadyExist2 << en
d1;
1482         }
1483         else {
1484             break;
1485         }
1486     }
1487
1488     cout << EmployeeProfile::employeeLanguage->address;
1489     in >> profile.address;
1490     cout << EmployeeProfile::employeeLanguage->education;
1491     in >> profile.education;
1492     cout << EmployeeProfile::employeeLanguage->position;
1493     in >> profile.position;
1494     cout << EmployeeProfile::employeeLanguage->hireDate;
1495     in >> profile.hireDate;
1496     cout << EmployeeProfile::employeeLanguage->department;
1497     in >> profile.department;
1498     return in;
1499 }
1500 // 重载 << 运算符
1501 ostream& operator<<(ostream& out, EmployeeProfile& profile)
1502 {
1503     const int SHORT_WIDTH = 10; // 短列宽
1504     const int MIDDLE_WIDTH = 15; // 中列宽
1505     const int LONG_WIDTH = 20; // 长列宽
1506     const int LINE_LENGTH = 155; // 分割线长度
1507     // 输出数据行
1508     out << left << setw(MIDDLE_WIDTH) << profile.id
1509         << setw(SHORT_WIDTH) << profile.name
1510         << setw(LONG_WIDTH) << profile.idNumber
1511         << setw(SHORT_WIDTH) << profile.gender
1512         << setw(SHORT_WIDTH) << profile.age
1513         << setw(MIDDLE_WIDTH) << profile.phoneNumber
1514         << setw(LONG_WIDTH) << profile.address
1515         << setw(SHORT_WIDTH) << profile.education
1516         << setw(MIDDLE_WIDTH) << profile.position
1517         << setw(MIDDLE_WIDTH) << profile.hireDate
1518         << setw(MIDDLE_WIDTH) << profile.department << endl;
```

```
1519         out << string(LINE_LENGTH, '-');
1520         return out;
1521     }
1522     // 有参构造函数定义
1523     EmployeeProfile::EmployeeProfile(
1524         string i,
1525         string n,
1526         string idN,
1527         string g,
1528         int a,
1529         string p,
1530         string addr,
1531         string e,
1532         string pos,
1533         string hd,
1534         string dep
1535     ) :
1536         id(i),
1537         name(n),
1538         idNumber(idN),
1539         gender(g),
1540         age(a),
1541         phoneNumber(p),
1542         address(addr),
1543         education(e),
1544         position(pos),
1545         hireDate(hd),
1546         department(dep)
1547     {
1548     }
1549     EmployeeProfile::EmployeeProfile(shared_ptr<Language>& langPtr) {
1550         employeeLanguage = langPtr;
1551     }
1552     // 成员变量对应的 get 函数
1553     string EmployeeProfile::getId()
1554     {
1555         return id;
1556     }
1557     string EmployeeProfile::getName()
1558     {
1559         return name;
1560     }
```

```
1561     string EmployeeProfile::getIdNumber()
1562     {
1563         return idNumber;
1564     }
1565     string EmployeeProfile::getGender()
1566     {
1567         return gender;
1568     }
1569     int EmployeeProfile::getAge()
1570     {
1571         return age;
1572     }
1573     string EmployeeProfile::getPhoneNumber()
1574     {
1575         return phoneNumber;
1576     }
1577     string EmployeeProfile::getAddress()
1578     {
1579         return address;
1580     }
1581     string EmployeeProfile::getEducation()
1582     {
1583         return education;
1584     }
1585     string EmployeeProfile::getPosition()
1586     {
1587         return position;
1588     }
1589     string EmployeeProfile::getHireDate()
1590     {
1591         return hireDate;
1592     }
1593     string EmployeeProfile::getDepartment()
1594     {
1595         return department;
1596     }
```

6. Language.h

```
1597     #pragma once
1598     #include <string>
```

```
1599     using namespace std;
1600     class Language
1601     {
1602     public:
1603         virtual ~Language() {}
1604         // 登录注册页面显示
1605         virtual string loginAndRegisterPage() const = 0;
1606         virtual string menu() const = 0;
1607         virtual string tableTitle() const = 0;
1608         string welcomeLogin;
1609         string welcomeRegister;
1610         string forgetPassword;
1611         string isExit;
1612         string yesOrNo;
1613         string selectFunction;
1614         string currentPage;
1615         string page;
1616         string leftOrRight;
1617         string totalEmployee;
1618         string updateByIdOrIdNumber;
1619         string deleteByIdOrIdNumber;
1620         string selectTime;
1621         string close;
1622         string userName;
1623         string music;
1624         string yesOrNo2;
1625         string exitSuccess;
1626         string loginSuccess;
1627         string registerSuccess;
1628         string inputUserName;
1629         string inputConfirmPassword;
1630         string inputPhone;
1631         string inputPassword;
1632         string inputCode;
1633         string inputPageNum;
1634         string inputNum;
1635         string inputNumPerPage;
1636         string inputOneOrTwo;
1637         string inputSelectContent;
1638         string userNameNotExist;
1639         string phoneAlreadyExist;
1640         string userNameAlreadyExist;
```

```
1641     string passwordDifferent;
1642     string userNameOrPasswordError;
1643     string overtime;
1644     string codeError;
1645     string numTooLong;
1646     string canNotOpen;
1647     string pressEnter;
1648     string KEY_ERROR;
1649
1650     // -----User-----
1651     string canCreateTemp;
1652     string inputNewPassword;
1653     string phone;
1654     string updatePasswordSuccess;
1655     string saveUser;
1656     string wait;
1657     string sending;
1658     string sended;
1659     string sendCode;
1660     string timeValid;
1661     string phoneNotRegister;
1662     string passwordShort;
1663     string passwordLong;
1664     // -----EmployeeProfile-----
1665     string employeeId;
1666     string employeeAlreadyDel;
1667     string idNum;
1668     string employeeIdNotFound;
1669     string employeeRecord;
1670     string idNumNotFound;
1671     string inputEmployeeId;
1672     string inputIdNum;
1673     string employeeSaveTo;
1674     string rатаinOldVal;
1675     string catNotUpdate;
1676     string oldVal;
1677     string newVal;
1678     string idNumFormatError;
1679     string idNumAlreadyExist;
1680     string reinputPhone;
1681     string employeePhoneAlreadyExist = "手机号已存在！\n请重新输入手机号：";
1682     string employeeUpdateSuccess;
```

```
1683     string phoneFormatError;
1684     string employeeIdFormat;
1685     string employeeIdFormatError;
1686     string employeeIdAlreadyExist;
1687     string employeeName;
1688     string inputIdNum2;
1689     string idNumFormatError2;
1690     string idNumAlreadyExist2;
1691     string gender;
1692     string age;
1693     string employeePhone;
1694     string phoneAlreadyExist2;
1695     string address;
1696     string education;
1697     string position;
1698     string hireDate;
1699     string department;
1700 };
```

7. Chinese.h

```
1701     #pragma once
1702     #include "Language.h"
1703
1704     using namespace std;
1705     class Chinese : public Language
1706     {
1707     public:
1708         string loginAndRegisterPage() const override;
1709         string menu() const override;
1710         string tableTitle() const override;
1711         Chinese();
1712     };
```

8. Chinese.cpp

```
1713     #include "Chinese.h"
1714     #include <string>
1715     #include <iostream>
1716     #include <iomanip>
```

```

1717     using namespace std;
1718     string Chinese::loginAndRegisterPage() const {
1719         cout << "    欢迎使用职工档案管理系统    " << endl;
1720         cout << "*****" << endl;
1721         cout << "*****" << endl;
1722         cout << "*****    1. 登录    *****" << endl;
1723         cout << "*****" << endl;
1724         cout << "*****    2. 注册    *****" << endl;
1725         cout << "*****" << endl;
1726         cout << "*****    3. 忘记密码    *****" << endl;
1727         cout << "*****" << endl;
1728         cout << "*****    ESC.退出系统    *****" << endl;
1729         cout << "*****" << endl;
1730         cout << "*****" << endl;
1731         return "";
1732     }
1733     string Chinese::menu() const
1734     {
1735         cout << "    欢迎使用职工档案管理系统    " << endl;
1736         cout << "—————" << endl;
1737         cout << "|                                |" << endl;
1738         cout << "|    1. 查询职工档案信息    |" << endl;
1739         cout << "|    2. 新增职工档案信息    |" << endl;
1740         cout << "|    3. 修改职工档案信息    |" << endl;
1741         cout << "|    4. 删除职工档案信息    |" << endl;
1742         cout << "|    ESC.退出登录          |" << endl;
1743         cout << "|                                |" << endl;
1744         cout << "—————" << endl;
1745         return "";
1746     }
1747     string Chinese::tableTitle() const
1748     {
1749         const int SHORT_WIDTH = 10; // 短列宽
1750         const int MIDDLE_WIDTH = 15; // 中列宽
1751         const int LONG_WIDTH = 20; // 长列宽
1752         const int LINE_LENGTH = 155; // 分割线长度
1753         // 输出表头
1754         cout << left
1755             << setw(MIDDLE_WIDTH) << "工号:"
1756             << setw(SHORT_WIDTH) << "职工姓名:"
1757             << setw(LONG_WIDTH) << "身份证号:"
1758             << setw(SHORT_WIDTH) << "性别:"

```

```

1759         << setw(SHORT_WIDTH) << "年龄:"
1760         << setw(MIDDLE_WIDTH) << "联系电话:"
1761         << setw(LONG_WIDTH) << "家庭地址:"
1762         << setw(SHORT_WIDTH) << "学历:"
1763         << setw(MIDDLE_WIDTH) << "职位:"
1764         << setw(MIDDLE_WIDTH) << "入职日期:"
1765         << setw(MIDDLE_WIDTH) << "所属部门:"
1766         << endl;
1767         return "";
1768     }
1769     Chinese::Chinese()
1770     {
1771         welcomeLogin = "  欢迎登录职工档案管理系统  ";
1772         welcomeRegister = "  欢迎注册职工档案管理系统  ";
1773         forgetPassword = "  忘记密码  ";
1774         isExit = "是否退出";
1775         yesOrNo = "1.确认 2.取消";
1776         selectFunction = "\t    1.按工号升序 2.按工号降序 3.按年龄升序 4.按年龄降序"
1777             "5.按入职时间升序 6.按入职时间降序 7.前往页码 8.更改每页展示数 9.模糊查"
1778             "询 ESC 键退出";
1779         currentPage = "\n 当前页: ";
1780         page = "/页";
1781         leftOrRight = "\t 向左: A/←\t 向右: D/→";
1782         totalEmployee = "总职工数: ";
1783         updateByIdOrIdNumber = "1.通过工号更新职工 2.通过身份证号更新职工  ESC 键退出"
1784             ";
1785         deleteByIdOrIdNumber = "1.通过工号删除 2.通过身份证号删除  ESC 键退出";
1786         selectTime = "本次查询时间: ";
1787         close = "3 秒后关闭窗口.";
1788         userName = "用户名: ";
1789         music = "是否开启背景音乐? ";
1790         yesOrNo2 = "1.是 2.否";
1791         exitSuccess = "\n 退出成功! ";
1792         loginSuccess = "登陆成功";
1793         registerSuccess = "注册成功";
1794         inputUserName = "请输入用户名: ";
1795         inputPassword = "请输入密码: ";
1796         inputConfirmPassword = "请再次输入密码: ";
1797         inputPhone = "请输入手机号: ";
1798         inputCode = "请输入验证码: ";
1799         inputPageNum = "\n 请输入页码: ";
1800         inputNum = "请输入数字! ";

```



```
1799     inputNumPerPage = "\n 请输入每页展示数: ";
1800     inputOneOrTwo = "请按 1 或 2 键";
1801     inputSelectContent = "\n 请输入查询的内容: ";
1802     userNameNotExist = "用户名不存在, 请先注册";
1803     userNameAlreadyExist = "用户名已存在, 请重新输入";
1804     phoneAlreadyExist = "手机号已存在, 请重新输入";
1805     passwordDifferent = "两次密码不一致, 请重新输入! ";
1806     userNameOrPasswordError = "用户名或密码错误, 请重试! ";
1807     overtime = "超过 60 秒, 操作超时";
1808     codeError = "验证码错误, 请重新输入: ";
1809     numTooLong = "数字太大, 超出了允许的范围! ";
1810     canNotOpen = "无法打开文件: ";
1811     pressEnter = "按回车键重新发送";
1812     KEY_ERROR = "请按正确的按键! ";
1813     // -----User-----
1814     canCreateTemp = "无法创建临时文件! ";
1815     inputNewPassword = "请输入新密码: ";
1816     phone = "手机号为 ";
1817     updatePasswordSuccess = " 的用户密码已成功修改。";
1818     saveUser = "用户数据已保存至: ";
1819     wait = "请稍等...";
1820     sending = "正在发送";
1821     sended = "已向手机号 ";
1822     sendCode = " 发送验证码: ";
1823     timeValid = "\t(60 秒有效)";
1824     phoneNotRegister = "该手机号未注册! ";
1825     passwordShort = "密码长度不能少于 6 位, 请重新输入: ";
1826     passwordLong = "\n 密码长度不能超过 20 位, 请重新输入: ";
1827     // -----EmployeeProfile-----
1828     employeeId = "工号为 ";
1829     employeeAlreadyDel = " 的员工记录已成功删除。";
1830     idNum = "身份证为 ";
1831     employeeIdNotFound = "未找到工号为 ";
1832     employeeRecord = " 的员工记录。";
1833     idNumNotFound = "未找到身份证号为 ";
1834     inputEmployeeId = "请输入工号: ";
1835     inputIdNum = "请输入身份证号: ";
1836     employeeSaveTo = "职工数据已保存至: ";
1837     rатаinOldVal = "按回车键则保留旧数据\n\n 旧值: ";
1838     catNotUpdate = "不可修改";
1839     oldVal = "旧值: ";
1840     newVal = "新值: ";
```

```

1841     idNumFormatError = "身份证号格式有误! \n 请重新输入身份证号: ";
1842     idNumAlreadyExist = "身份证号已存在! \n 请重新输入身份证号: ";
1843     reinputPhone = "请重新输入手机号: ";
1844     employeePhoneAlreadyExist = "手机号已存在! \n 请重新输入手机号: ";
1845     employeeUpdateSuccess = " 的员工记录已成功修改。";
1846     phoneFormatError = "手机号格式错误! ";
1847     employeeIdFormat = "工号(长度为 10 位,前四位数在 1956~当前年份之间):";
1848     employeeIdFormatError = "工号格式有误! ";
1849     employeeIdAlreadyExist = "工号已存在! ";
1850     employeeName = "职工姓名:";
1851     inputIdNum2 = "身份证号 (X 为 大写):";
1852     idNumFormatError2 = "身份证号格式有误! ";
1853     idNumAlreadyExist2 = "身份证号已存在! ";
1854     gender = "性别:";
1855     age = "年龄:";
1856     employeePhone = "联系电话:";
1857     phoneAlreadyExist2 = "手机号已存在! ";
1858     address = "家庭地址:";
1859     education = "学历:";
1860     position = "职位:";
1861     hireDate = "入职日期:";
1862     department = "所属部门:";
1863 }

```

9. English.h

```

1864 #pragma once
1865 #include "Language.h"
1866 #include <string>
1867 using namespace std;
1868 class English : public Language
1869 {
1870 public:
1871     string loginAndRegisterPage() const override;
1872     string menu() const override;
1873     string tableTitle() const override;
1874     English();
1875
1876 };

```

10. English.cpp

```

1877 #include "English.h"
1878 #include <string>
1879 #include <iostream>
1880 #include <iomanip>
1881 using namespace std;
1882 string English::loginAndRegisterPage() const {
1883     cout << "Welcome to use the employee file management system" << endl;
1884     cout << " *****" << endl;
1885     cout << " *****" << endl;
1886     cout << " ***** 1. login *****" << endl;
1887     cout << " *****" << endl;
1888     cout << " ***** 2. register *****" << endl;
1889     cout << " *****" << endl;
1890     cout << " ***** 3. forget password *****" << endl;
1891     cout << " *****" << endl;
1892     cout << " ***** ESC.exit system *****" << endl;
1893     cout << " *****" << endl;
1894     cout << " *****" << endl;
1895     return "";
1896 }
1897 string English::menu() const
1898 {
1899     cout << " Welcome to the Employee Profile Management System " << endl;
1900     ;
1901     cout << " _____" << endl;
1902     cout << "| " << endl;
1903     cout << "| 1. Query Employee Profiles |" << endl;
1904     cout << "| 2. Add Employee Profile |" << endl;
1905     cout << "| 3. Update Employee Profile |" << endl;
1906     cout << "| 4. Delete Employee Profile |" << endl;
1907     cout << "| ESC. Logout |" << endl;
1908     cout << "| " << endl;
1909     cout << " _____" << endl;
1910     return "";
1911 }
1912 string English::tableTitle() const
1913 {
1914     const int SHORT_WIDTH = 10; // 短列宽
1915     const int MIDDLE_WIDTH = 15; // 中列宽

```

```

1915     const int LONG_WIDTH = 20; // 长列宽
1916     const int LINE_LENGTH = 155; // 分割线长度
1917     // 输出表头
1918     cout << left
1919         << setw(MIDDLE_WIDTH) << "Employee ID:"
1920         << setw(SHORT_WIDTH) << "Name:"
1921         << setw(LONG_WIDTH) << "ID Number:"
1922         << setw(SHORT_WIDTH) << "Gender:"
1923         << setw(SHORT_WIDTH) << "Age:"
1924         << setw(MIDDLE_WIDTH) << "Phone:"
1925         << setw(LONG_WIDTH) << "Address:"
1926         << setw(SHORT_WIDTH) << "Education:"
1927         << setw(MIDDLE_WIDTH) << "Position:"
1928         << setw(MIDDLE_WIDTH) << "Hire Date:"
1929         << setw(MIDDLE_WIDTH) << "Department:"
1930         << endl;
1931     return "";
1932 }
1933 English::English()
1934 {
1935     welcomeLogin = " Welcome to Login the Employee Profile Management
1936 ";
1937     welcomeRegister = " Welcome to Register in the Employee Profile Management
1938 System ";
1939     forgetPassword = " Forgot Password ";
1940     isExit = "Exit?";
1941     yesOrNo = "1. Yes 2. No";
1942     selectFunction = "\t 1. Ascending by Employee ID 2. Descending by Employee
1943 ID 3. Ascending by Age 4. Descending by Age"
1944 "5. Ascending by Hire Date \n\t 6. Descending by Hire Date 7. Go to
1945 Page 8. Change Items Per Page 9. Fuzzy Query \t\tESC to Exit";
1946     currentPage = "\nCurrent Page: ";
1947     page = "/Page";
1948     leftOrRight = "\tLeft: A/←\tRight: D/→";
1949     totalEmployee = "Total Employees: ";
1950     updateByIdOrIdNumber = "1. Update Employee by ID 2. Update Employee by ID N
1951 umber Press \t\tESC to Exit";
1952     deleteByIdOrIdNumber = "1. Delete Employee by ID 2. Delete Employee by ID N
1953 umber Press \t\tESC to Exit";
1954     selectTime = "Current Query Time: ";
1955     close = "Window will close in 3 seconds.";
1956     userName = "Username: ";

```

```
1951     music = "Whether to enable background music?";
1952     yesOrNo2 = "1.YES 2.NO";
1953     exitSuccess = "\nExit successful!";
1954     loginSuccess = "Login successful";
1955     registerSuccess = "Registration successful";
1956     inputUserName = "Please enter username: ";
1957     inputPassword = "Please enter password: ";
1958     inputConfirmPassword = "Please enter password again: ";
1959     inputPhone = "Please enter phone number: ";
1960     inputCode = "Please enter verification code: ";
1961     inputPageNum = "\nPlease enter page number: ";
1962     inputNum = "Please enter a number!";
1963     inputNumPerPage = "\nPlease enter items per page: ";
1964     inputOneOrTwo = "Please press 1 or 2";
1965     inputSelectContent = "\nPlease enter query content: ";
1966     userNameNotExist = "Username does not exist, please register first";
1967     userNameAlreadyExist = "Username already exists, please enter a different o
ne";
1968     phoneAlreadyExist = "Phone number already exists, please enter a different
one";
1969     passwordDifferent = "Passwords do not match, please try again!";
1970     userNameOrPasswordError = "Incorrect username or password, please try again
!";
1971     overtime = "Operation timed out, over 60 seconds";
1972     codeError = "Incorrect verification code, please try again: ";
1973     numTooLong = "Number is too large, exceeds the allowed range!";
1974     canNotOpen = "Cannot open file: ";
1975     pressEnter = "Press Enter to resend";
1976     KEY_ERROR = "Please press the correct KEY!";
1977     // -----User-----
1978     canCreateTemp = "Cannot create temporary file!";
1979     inputNewPassword = "Please enter new password: ";
1980     phone = "Phone number is ";
1981     updatePasswordSuccess = " user's password has been successfully updated.";
1982     saveUser = "User data saved to: ";
1983     wait = "Please wait...";
1984     sending = "Sending";
1985     sendCode = "Code has been sent to phone number ";
1986     sendCode = " Send code: ";
1987     timeValid = "\t(Valid for 60 seconds)";
1988     phoneNotRegister = "This phone number is not registered!";
```

```
1989     passwordShort = "Password length should be at least 6 characters, please tr
      y again: ";
1990     passwordLong = "\nPassword length should not exceed 20 characters, please t
      ry again: ";
1991     // -----EmployeeProfile-----
1992     employeeId = "Employee ID is ";
1993     employeeAlreadyDel = " employee record has been successfully deleted.";
1994     idNum = "ID number is ";
1995     employeeIdNotFound = "Employee record with ID ";
1996     employeeRecord = " not found.";
1997     idNumNotFound = "Employee record with ID number ";
1998     inputEmployeeId = "Please enter employee ID: ";
1999     inputIdNum = "Please enter ID number: ";
2000     employeeSaveTo = "Employee data saved to: ";
2001     retainOldVal = "Press Enter to retain old data\n\nOld value: ";
2002     catNotUpdate = "Cannot update";
2003     oldVal = "Old value: ";
2004     newVal = "New value: ";
2005     idNumFormatError = "Invalid ID number format!\nPlease enter ID number again
      : ";
2006     idNumAlreadyExist = "ID number already exists!\nPlease enter ID number agai
      n: ";
2007     reinputPhone = "Please re-enter phone number: ";
2008     employeePhoneAlreadyExist = "Phone number already exists!\nPlease enter pho
      ne number again: ";
2009     employeeUpdateSuccess = " employee record has been successfully updated.";
2010     phoneFormatError = "Invalid phone number format!";
2011     employeeIdFormat = "Employee ID (length is 10 digits, first four digits are
      between 1956 and the current year):";
2012     employeeIdFormatError = "Invalid employee ID format!";
2013     employeeIdAlreadyExist = "Employee ID already exists!";
2014     employeeName = "Employee name: ";
2015     inputIdNum2 = "ID number (X is uppercase): ";
2016     idNumFormatError2 = "Invalid ID number format!";
2017     idNumAlreadyExist2 = "ID number already exists!";
2018     gender = "Gender: ";
2019     age = "Age: ";
2020     employeePhone = "Phone: ";
2021     phoneAlreadyExist2 = "Phone number already exists!";
2022     address = "Address: ";
2023     education = "Education: ";
```

```

2024     position = "Position: ";
2025     hireDate = "Hire Date: ";
2026     department = "Department: ";
2027 }

```

11. Constant.h

```

2028     #pragma once
2029     #include <string>
2030     #include <regex>
2031     using namespace std;
2032     /**
2033      * @author XZH
2034      */
2035     const string EMPLOYEE_FILENAME = "employee.txt";
2036     const string USER_FILENAME = "user.txt";
2037     const string KEY_ERROR = "请按正确的按键! ";
2038     const string KEY_ERROR_ENG = "Please press the correct KEY";
2039     const int PAGE_NUM = 1;
2040     const int PAGE_SIZE = 6;
2041     static int pageNum = 1;
2042     static int pageSize = 6;
2043     // 序号为该属性在一行中的位置, 从 1 开始
2044     // 职工 id 的序号
2045     static int EMPLOYEE_ID_SERIAL_NUMBER = 1;
2046     // 职工 phone 的序号
2047     static int EMPLOYEE_PHONE_SERIAL_NUMBER = 6;
2048     // 职工 idNumber 的序号
2049     static int EMPLOYEE_ID_NUMBER_SERIAL_NUMBER = 3;
2050     // 用户 phone 的序号
2051     static int USER_PHONE_SERIAL_NUMBER = 3;
2052     // 用户 password 的序号
2053     static int USER_PASSWORD_SERIAL_NUMBER = 2;
2054     // 用户 name 的序号
2055     static int USER_NAME_SERIAL_NUMBER = 1;
2056     // 身份证正则表达式
2057     static regex idNumberPattern("^[1-
2058     9]\\d{5}(?:18|19|20)\\d{2}(?:0\\d|10|11|12)(?:0[1-9]|[1-
2059     2]\\d|30|31)\\d{3}[\\dX]$");
2060     // 手机号正则表达式

```

```
2059 static regex phonePattern("^(:?(:\\+|00)86)?1(:?(:3[\\d])|(:4[5-7|9])|(:5[0-3|5-9])|(:6[5-7])|(:7[0-8])|(:8[\\d])|(:9[1|8|9]))\\d{8}$");
```

12. MD5.h

```
2060 #pragma once
2061 #include <iostream>
2062 #include <string>
2063 #include <cstring>
2064 #include <cmath>
2065 /**
2066  * @author XZH
2067  */
2068 // Functions for MD5 transformation
2069 #define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
2070 #define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
2071 #define H(x, y, z) ((x) ^ (y) ^ (z))
2072 #define I(x, y, z) ((y) ^ ((x) | (~z)))
2073 #define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))
2074 #define FF(a, b, c, d, x, s, ac) { \
2075     (a) += F((b), (c), (d)) + (x) + (ac); \
2076     (a) = ROTATE_LEFT((a), (s)); \
2077     (a) += (b); \
2078 }
2079 #define GG(a, b, c, d, x, s, ac) { \
2080     (a) += G((b), (c), (d)) + (x) + (ac); \
2081     (a) = ROTATE_LEFT((a), (s)); \
2082     (a) += (b); \
2083 }
2084 #define HH(a, b, c, d, x, s, ac) { \
2085     (a) += H((b), (c), (d)) + (x) + (ac); \
2086     (a) = ROTATE_LEFT((a), (s)); \
2087     (a) += (b); \
2088 }
2089 #define II(a, b, c, d, x, s, ac) { \
2090     (a) += I((b), (c), (d)) + (x) + (ac); \
2091     (a) = ROTATE_LEFT((a), (s)); \
2092     (a) += (b); \
2093 }
2094 class MD5 {
2095 public:
```



```
2096     MD5() {
2097         init();
2098     }
2099     std::string calculate(const std::string& input) {
2100         init();
2101         update(reinterpret_cast<const unsigned char*>(input.c_str()), input.length());
2102         finalize();
2103         char buf[33];
2104         for (int i = 0; i < 16; i++) {
2105             sprintf_s(&buf[i * 2], 3, "%02x", static_cast<unsigned int>(digest[i]));
2106         }
2107         buf[32] = '\0';
2108         return std::string(buf);
2109     }
2110 private:
2111     void init() {
2112         count[0] = count[1] = 0;
2113         state[0] = 0x67452301;
2114         state[1] = 0xefcdab89;
2115         state[2] = 0x98badcfe;
2116         state[3] = 0x10325476;
2117     }
2118     void update(const unsigned char* input, size_t input_len) {
2119         size_t i, index, part_len;
2120         index = (count[0] >> 3) & 0x3F;
2121         if ((count[0] += (input_len << 3)) < (input_len << 3))
2122             count[1]++;
2123         count[1] += (input_len >> 29);
2124         part_len = 64 - index;
2125         if (input_len >= part_len) {
2126             memcpy(&buffer[index], input, part_len);
2127             transform(buffer);
2128             for (i = part_len; i + 63 < input_len; i += 64) {
2129                 transform(&input[i]);
2130             }
2131             index = 0;
2132         }
2133         else {
2134             i = 0;
2135         }
```

```
2136         memcpy(&buffer[index], &input[i], input_len - i);
2137     }
2138     void finalize() {
2139         unsigned char bits[8];
2140         size_t index, pad_len;
2141         encode(bits, count, 8);
2142         index = (count[0] >> 3) & 0x3f;
2143         pad_len = (index < 56) ? (56 - index) : (120 - index);
2144         update(padding, pad_len);
2145         update(bits, 8);
2146         encode(digest, state, 16);
2147     }
2148     void transform(const unsigned char block[64]) {
2149         uint32_t a = state[0], b = state[1], c = state[2], d = state[3], x[16];
2150
2151         decode(x, block, 64);
2152         /* Round 1 */
2153         FF(a, b, c, d, x[0], 7, 0xd76aa478);
2154         FF(d, a, b, c, x[1], 12, 0xe8c7b756);
2155         FF(c, d, a, b, x[2], 17, 0x242070db);
2156         FF(b, c, d, a, x[3], 22, 0xc1bdceee);
2157         FF(a, b, c, d, x[4], 7, 0xf57c0faf);
2158         FF(d, a, b, c, x[5], 12, 0x4787c62a);
2159         FF(c, d, a, b, x[6], 17, 0xa8304613);
2160         FF(b, c, d, a, x[7], 22, 0xfd469501);
2161         FF(a, b, c, d, x[8], 7, 0x698098d8);
2162         FF(d, a, b, c, x[9], 12, 0x8b44f7af);
2163         FF(c, d, a, b, x[10], 17, 0xffff5bb1);
2164         FF(b, c, d, a, x[11], 22, 0x895cd7be);
2165         FF(a, b, c, d, x[12], 7, 0x6b901122);
2166         FF(d, a, b, c, x[13], 12, 0xfd987193);
2167         FF(c, d, a, b, x[14], 17, 0xa679438e);
2168         FF(b, c, d, a, x[15], 22, 0x49b40821);
2169         /* Round 2 */
2170         GG(a, b, c, d, x[1], 5, 0xf61e2562);
2171         GG(d, a, b, c, x[6], 9, 0xc040b340);
2172         GG(c, d, a, b, x[11], 14, 0x265e5a51);
2173         GG(b, c, d, a, x[0], 20, 0xe9b6c7aa);
2174         GG(a, b, c, d, x[5], 5, 0xd62f105d);
2175         GG(d, a, b, c, x[10], 9, 0x02441453);
2176         GG(c, d, a, b, x[15], 14, 0xd8a1e681);
2177         GG(b, c, d, a, x[4], 20, 0xe7d3fbc8);
```

```

2177     GG(a, b, c, d, x[9], 5, 0x21e1cde6);
2178     GG(d, a, b, c, x[14], 9, 0xc33707d6);
2179     GG(c, d, a, b, x[3], 14, 0xf4d50d87);
2180     GG(b, c, d, a, x[8], 20, 0x455a14ed);
2181     GG(a, b, c, d, x[13], 5, 0xa9e3e905);
2182     GG(d, a, b, c, x[2], 9, 0xfcefa3f8);
2183     GG(c, d, a, b, x[7], 14, 0x676f02d9);
2184     GG(b, c, d, a, x[12], 20, 0x8d2a4c8a);
2185     /* Round 3 */
2186     HH(a, b, c, d, x[5], 4, 0xfffa3942);
2187     HH(d, a, b, c, x[8], 11, 0x8771f681);
2188     HH(c, d, a, b, x[11], 16, 0x6d9d6122);
2189     HH(b, c, d, a, x[14], 23, 0xfde5380c);
2190     HH(a, b, c, d, x[1], 4, 0xa4beea44);
2191     HH(d, a, b, c, x[4], 11, 0x4bdecfa9);
2192     HH(c, d, a, b, x[7], 16, 0xf6bb4b60);
2193     HH(b, c, d, a, x[10], 23, 0xbee5fb70);
2194     HH(a, b, c, d, x[13], 4, 0x289b7ec6);
2195     HH(d, a, b, c, x[0], 11, 0xeea127fa);
2196     HH(c, d, a, b, x[3], 16, 0xd4ef3085);
2197     HH(b, c, d, a, x[6], 23, 0x04881d05);
2198     HH(a, b, c, d, x[9], 4, 0xd9d4d039);
2199     HH(d, a, b, c, x[12], 11, 0xe6db99e5);
2200     HH(c, d, a, b, x[15], 16, 0x1fa27cf8);
2201     HH(b, c, d, a, x[2], 23, 0xc4ac5665);
2202     /* Round 4 */
2203     II(a, b, c, d, x[0], 6, 0xf4292244);
2204     II(d, a, b, c, x[7], 10, 0x432aff97);
2205     II(c, d, a, b, x[14], 15, 0xab9423a7);
2206     II(b, c, d, a, x[5], 21, 0xfc93a039);
2207     II(a, b, c, d, x[12], 6, 0x655b59c3);
2208     II(d, a, b, c, x[3], 10, 0x8f0ccc92);
2209     II(c, d, a, b, x[10], 15, 0xffefff47d);
2210     II(b, c, d, a, x[1], 21, 0x85845dd1);
2211     II(a, b, c, d, x[8], 6, 0x6fa87e4f);
2212     II(d, a, b, c, x[15], 10, 0xfe2ce6e0);
2213     II(c, d, a, b, x[6], 15, 0xa3014314);
2214     II(b, c, d, a, x[13], 21, 0x4e0811a1);
2215     II(a, b, c, d, x[4], 6, 0xf7537e82);
2216     II(d, a, b, c, x[11], 10, 0xbd3af235);
2217     II(c, d, a, b, x[2], 15, 0x2ad7d2bb);
2218     II(b, c, d, a, x[9], 21, 0xeb86d391);

```

```

2219         state[0] += a;
2220         state[1] += b;
2221         state[2] += c;
2222         state[3] += d;
2223         memset(x, 0, sizeof(x));
2224     }
2225     void encode(unsigned char* output, const uint32_t* input, size_t len) {
2226         size_t i, j;
2227         for (i = 0, j = 0; j < len; i++, j += 4) {
2228             output[j] = static_cast<unsigned char>(input[i] & 0xff);
2229             output[j + 1] = static_cast<unsigned char>((input[i] >> 8) & 0xff);
2230             output[j + 2] = static_cast<unsigned char>((input[i] >> 16) & 0xff)
2231             ;
2232             output[j + 3] = static_cast<unsigned char>((input[i] >> 24) & 0xff)
2233             ;
2234         }
2235     }
2236     void decode(uint32_t* output, const unsigned char* input, size_t len) {
2237         size_t i, j;
2238         for (i = 0, j = 0; j < len; i++, j += 4) {
2239             output[i] = (static_cast<uint32_t>(input[j])) |
2240                 (static_cast<uint32_t>(input[j + 1]) << 8) |
2241                 (static_cast<uint32_t>(input[j + 2]) << 16) |
2242                 (static_cast<uint32_t>(input[j + 3]) << 24);
2243         }
2244     }
2245     uint32_t state[4];
2246     uint32_t count[2];
2247     unsigned char buffer[64];
2248     unsigned char digest[16];
2249     static const unsigned char padding[64];
2250 };
2251 const unsigned char MD5::padding[64] = {
2252     0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2253     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2254     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2255     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
2256 };

```

东华理工大学

课程设计评分表

学生姓名：

班级：

学号：

课程设计题目：

	项目内容	满分	实 评
选题	能结合所学课程知识、有一定的能力训练。符合选题要求（5 人一题）	10	
	工作量适中，难易度合理	10	
能力水平	能熟练应用所学知识，有一定查阅文献及运用文献资料能力	10	
	理论依据充分，数据准确，公式推导正确	10	
	能应用计算机软件进行编程、资料搜集录入、加工、排版、制图等	10	
	能体现创造性思维，或有独特见解	10	
成果质量	总体设计正确、合理，各项技术指标符合要求。	10	
	说明书综述简练完整，概念清楚、立论正确、技术用语准确、结论严谨合理；分析处理科学、条理分明、语言流畅、结构严谨、版面清晰	10	
	设计说明书栏目齐全、合理，符号统一、编号齐全。格式、绘图、表格、插图等规范准确，符合国家标准	10	
	有一定篇幅，字符数不少于 5000	10	
	总 分	100	

指导教师评语：

指导教师签名：

年 月 日