KC021-1

常州工学院

CHANGZHOU INSTITUTE OF TECHNOLOGY

课 程 设 计 说 明 书

课程名：《程序设计大作业》

题　目：ATM 管理系统

二级学院：计 算 机 信 息 工 程 学 院

专　　业：软 件 工 程

班　　级：22 软件三

学　　号：22030516

姓　　名：李硕

指导教师：曹 中 心

2023 年 6 月

一、课程认识

1. 要求

ATM（自动柜员机）模拟系统是一个基于 C 语言的大作业课题。该系统的主要目的是模拟真实世界中的 ATM 机的功能和操作，以提供用户进行账户管理和交易的便利性。本文将对该课题的设计过程进行总结，并按照层次化结构进行详细说明。

2. 目的

本课题的主要目的是设计和实现一个 ATM 模拟系统，其具体目标包括：

(1) 模拟真实世界中 ATM 机的基本功能，如取款、存款、查询余额、转账等。

(2) 提供用户管理功能，包括开户、销户、修改密码等。

(3) 实现基本的交易验证和账户安全措施，如密码验证、限制错误次数等。

(4) 提供用户友好的界面和操作方式，以便用户能够方便地使用系统进行操作。保持数据的持久性，即系统能够将用户的账户信息和交易记录保存在持久存储介质中，以便下次登录时能够恢复数据。

3. 系统结构

为了实现上述目标，ATM 模拟系统可以按照以下层次化结构进行设计：

(1) 用户界面层

用户界面层是用户与系统进行交互的界面，通常使用文本或图形界面。该层负责接收用户输入的命令和信息，并将其传递给下一层进行处理。在本系统中，用户界面层需要提供用户登录、账户管理和交易功能的界面，以及相应的输入和输出处理。

(2) 2. 业务逻辑层

业务逻辑层是系统的核心，负责处理用户输入的命令和信息，并进行相应的业务逻辑处理。该层包括账户管理模块和交易处理模块。账户管理模块用于处理用户账户的创建、删除、密码修改等操作，而交易处理模块负责处理用户的各种交易请求，如取款、存款、转账等。

(3) 数据访问层

数据访问层负责与系统的数据存储介质进行交互，将数据持久化保存。在本系统中，数据访问层需要实现用户账户信息和交易记录的读取和写入功能。常见的数据存储介质

可以是文件、数据库或内存等。

二、课题选择

1. 课程背景：

在现代社会中，自动柜员机（ATM）已经成为人们生活中不可或缺的一部分。它们提供了便利的金融服务，使人们能够随时随地进行取款、存款、查询余额和转账等操作。C语言作为一门广泛应用于嵌入式系统和操作系统开发的编程语言，是学习计算机科学和软件工程的基础之一。通过设计和实现一个ATM管理系统，我们可以将所学的C语言知识应用于实际场景中，深入理解计算机软件的工作原理，并提升我们的编程技能和系统设计能力。

2. 意义：

ATM管理系统的设计和实现具有重要的意义和价值。首先，它可以帮助我们深入理解现代金融系统的工作原理，包括账户管理、交易记录、安全性等方面。通过编写和调试ATM管理系统的代码，我们可以更好地理解计算机网络、数据库管理和数据结构等相关知识。其次，通过开发ATM管理系统，我们可以提高软件工程的实践能力，包括需求分析、系统设计、模块划分、代码编写和测试等方面。最后，ATM管理系统的设计和实现可以提供一个实践平台，使我们能够锻炼自己的团队协作和沟通能力，培养解决问题的能力和创新思维。

3. 实用性：

ATM管理系统具有实际应用的价值和实用性。在当今的金融服务行业中，ATM是一种广泛使用的自助服务工具，涉及到的技术和功能非常复杂。通过设计和实现一个完整的ATM管理系统，我们可以模拟和实现ATM机器上的各种功能，如账户管理、现金管理、交易记录、密码验证、网络通信等。这样的系统可以用于教育和培训目的，帮助学生和从业人员更好地理解和掌握 ATM 系统的工作原理，并为银行和金融机构提供一个可靠的软件平台，用于模拟和测试不同的操作场景和业务流程。

三、系统总体分析与设计

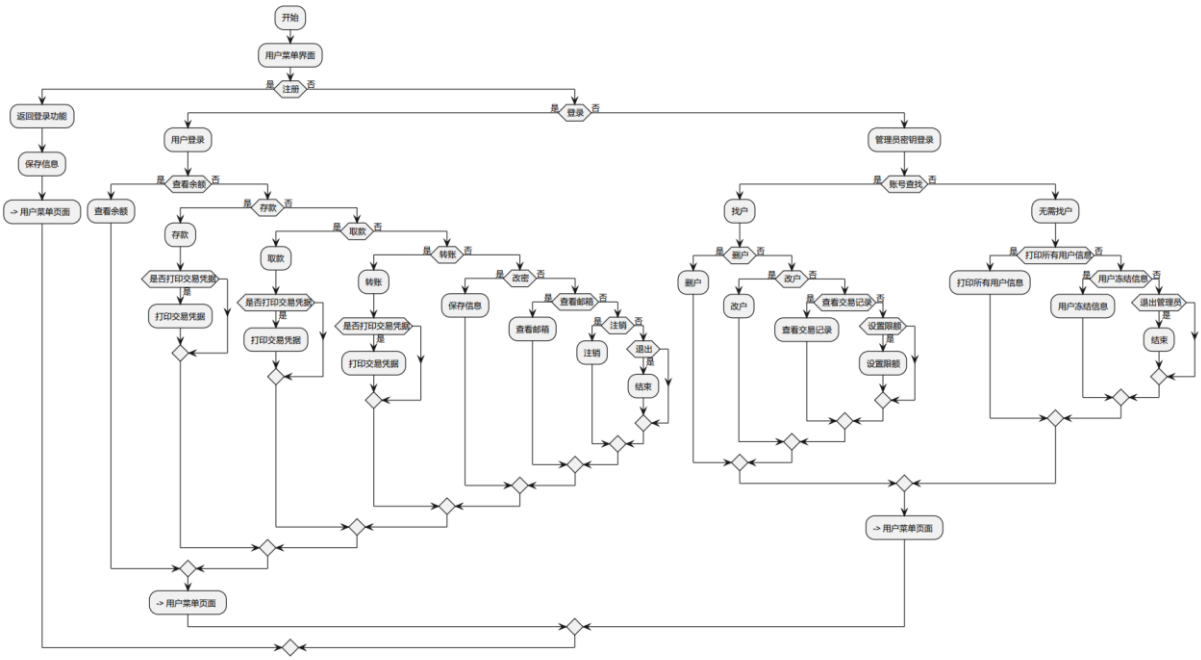

图 3-1

如图 3-1 所示，流程图功能:

1.用户操作层:

(1) 用户可以通过注册或登录进入系统
(2) 提供取款、存款、转账、改密、查看余额等基本功能
(3) 提供查看最近交易记录的功能

2.用户信息管理层:

(1) 管理和保存所有注册用户的个人信息和账户信息
(2) 用户进行交易时更新相应账户余额和交易记录
(3) 提供管理员修改和删除用户信息的功能

3.文件操作层:

(1) 使用文本文件保存所有用户信息和交易记录
(2) 注册新用户时添加信息到文本文件
(3) 用户登录时从文件读取用户信息
(4) 每次交易后更新用户余额并写入文件
(5) 保存交易记录到文本文件

4.交易记录查看层:

(1) 保存所有交易记录到文本文件
(2) 用户和管理员可以查看最近的一段交易记录
(3) 交易记录包括交易类型、金额、交易时间等信息

5.改密功能层:

(1) 用户可以通过改密功能修改登录密码
(2) 在文件中更新对应用户的密码信息
(3) 新密码需要符合一定规范,如长度、字母数字要求等

四、模块详细设计

1.改密模块设计:

(1) 获取旧密码和新密码的输入
(2) 从用户信息文件中读取对应用户的旧密码
(3) 进行密码匹配,如果匹配成功则继续,否则提示错误
(4) 生成新密码 hash 值,更新用户信息文件中的密码 hash
- 返回成功信息

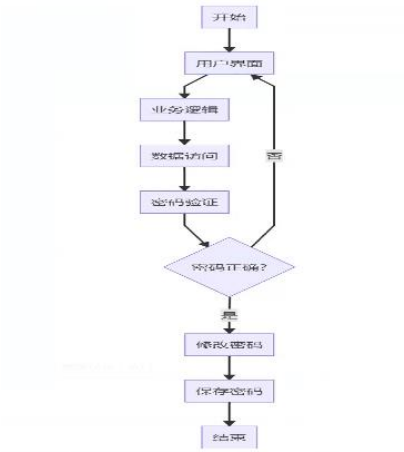

图 4-1

2.查看余额模块设计:

(1) 获取用户信息(账号)的输入
(2) 从用户信息文件中读取对应用户的余额
(3) 输出余额信息
(4) 为了安全,可显示前后若干位余额而不是具体数值

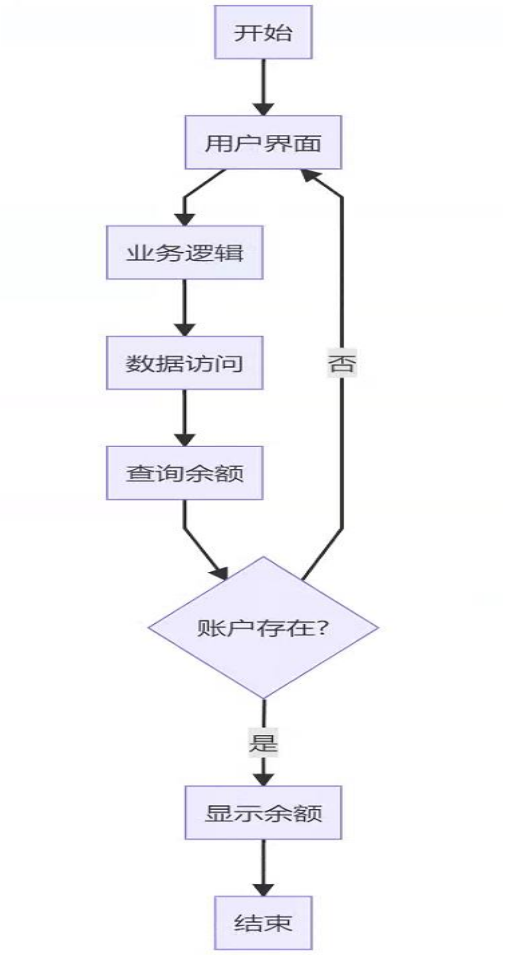

图 4-2

3.用户信息管理模块设计:

(1) 包含添加新用户、修改用户信息和删除用户三个功能

(2) 添加用户:向用户信息文件写入新的一条用户记录

(3) 修改用户:根据账号读取对应用户记录,更新部分字段,并重新写入文件

(4) 删除用户:根据账号搜索 User 记录,删除该记录

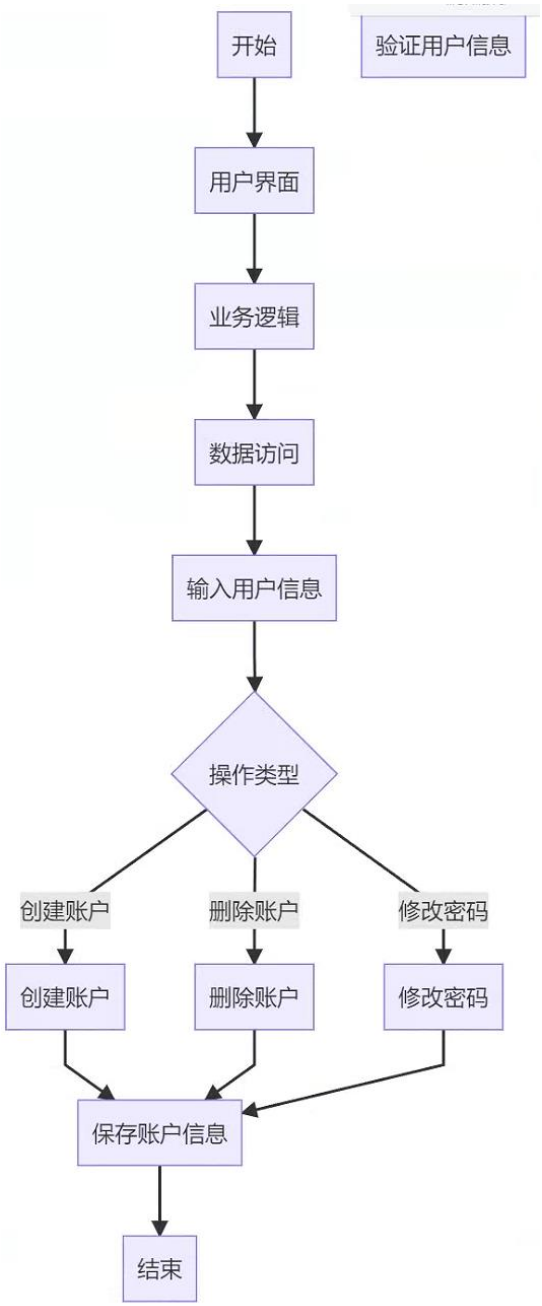

图 4-3

4.文件保存操作模块设计:

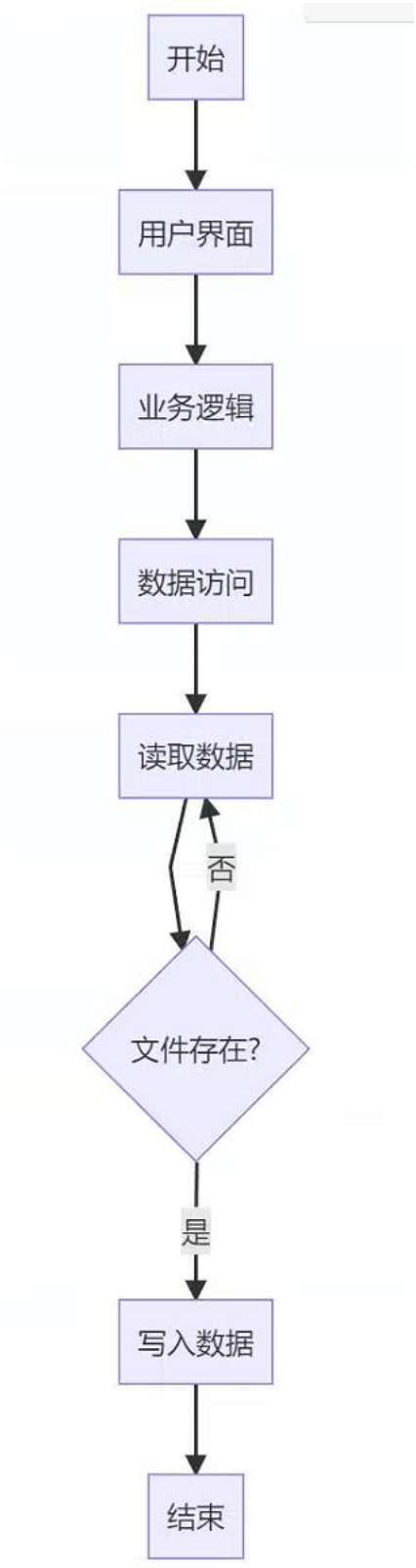

图 4-4

(1) 与用户信息管理模块 savedata

(2) 负责读取和写入用户信息文件

(3) 文件格式为固定字段的文本格式,方便读取和修改
(4) 用户信息字段包括:账号、密码 hash、名称、余额等
(5) 使用结构体表示一个 User 记录

5.交易记录查看模块设计:

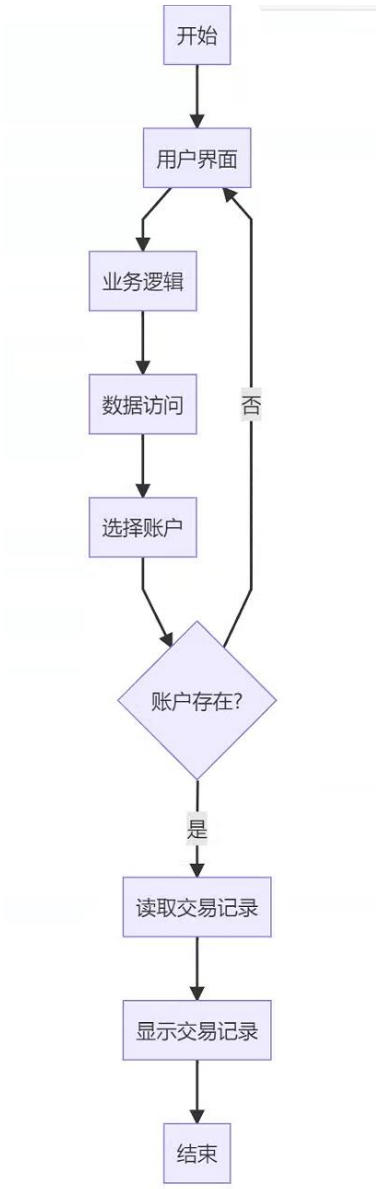

图 4-5

(1) 包含单次交易记录和最近多条记录两种查看方式
(2) 遍历交易记录文件,过滤符合要求的记录
(3) 交易记录文件存储格式类似用户信息文件
(4) 交易记录字段:账号、交易类型、金额、时间等

五、系统实现

1. 主要文件

文件名：*save.bat*

本文件主要存放银行用户的信息。

代码对应的结构体如下：

```
    typedef struct
{
    char account[max];
    char passwd[max];
    double balance;
    Record record[max];
    int NUM;
    bool Freeze;
    bool Submit;
    double limitCost;
    Verify verify;
} User;
```

代码对应的表格如下：

表 5 1 User

字段名	类型	说明
account	char[]	账户
passwd	char[]	密码
balance	double	余额
record	Record[max]	记录
NUM	int	数量
Freeze	bool	冻结状态
Submit	bool	提交状态

字段名	类型	说明
limitCost	double	限制成本
verify	Verify	验证

2. 主要函数

2.1 void changePassword()

(1) `Old` 数组存放用户输入的旧密码
(2) `New` 和 `Confirm` 数组分别存放两次输入的新密码
(3) 使用 `secret()` 函数隐藏用户输入的密码
(4) 通过 `strcmp()` 比较:
(5) 旧密码是否匹配 `pr->passwd`
(6) 两次新密码输入是否一致
(7) 如果通过验证,则使用 `strcpy()` 将新密码复制到:
pr->passwd
- `pr` 指向当前登录用户,`pr->passwd` 是其原密码字段
- 调用 `saveFile()` 函数将最新用户信息保存到文件
- 提示密码修改成功

其中:
- `users` 是 `User` 类型的数组,表示所有用户
- `count` 表示用户数量
- `pr` 指向当前登录用户

所以整体流程是:

(1) 获取旧密码
(2) 验证旧密码是否正确
(3) 获取并验证两次新密码输入
(4) 将新密码复制到对应用户的 `passwd` 字段
(5) 保存用户信息文件
(6) 提示密码修改成功

```
    void changePassword() //改密
    {
        char Old[MAX];
        char New[MAX];
        char Confirm[MAX];

        printf("请输入旧密码: \n");
        secret(Old);

        if (strcmp(Old, pr->passwd) !=
0)
        {
            printf("旧密码不正确,请
重新输入\n");
            return;
        }
```

这段代码定义了一个函数名为`changePassword`，其含义是实现密码修改的功能。

代码逻辑如下：

声明了三个字符数组变量`Old`、`New`和`Confirm`，用于存储旧密码、新密码和确认密码。使用`printf`函数输出提示信息，要求用户输入旧密码。调用一个名为`secret`的函数对用户输入的旧密码进行隐藏或保密操作。使用`strcmp`函数将用户输入的旧密码与`pr`结构体指针中的密码进行比较，如果不相等，则输出密码不正确的提示信息并返回。如果旧密码正确，则使用`printf`函数提示用户输入新密码。调用`secret`函数对用户输入的新密码进行隐藏或保密操作。使用`printf`函数提示用户再次输入新密码进行确认。调用`secret`函数对用户输入的确认密码进行隐藏或保密操作。使用`strcmp`函数将新密码和确认密码进行比较，如果不相等，则输出密码不一致的提示信息并返回。如果新密码和确认密码一致，则表示密码修改成功。

2.2 void printAllUsers()

```
    void printAllUsers() //打印所有用户
    {
        //内置管理员账号
        printf("所有用户信息：\n");
        for (int i = 0; i < count; i++)
        {
            printf("账号：%s\t\t\t 余额 ：%.2f\t\t\t", users[i].account,users[i].balance);
            printf("密码(已加密):");
            AddSecret(users[i].passwd);
        }
    }
```

这段代码定义了一个函数名为`printAllUsers`，其含义是打印所有用户的信息。

使用`printf`函数输出一个提示信息，表示要打印所有用户的信息。使用`for`循环遍历用户数组，循环变量`i`从 0 到`count-1`。 在循环内部，使用`printf`函数输出每个用户的账号、余额和加密后的密码。

(1) `%s`用于输出字符串，这里是输出用户的账号。

(2) `%.2f`用于输出浮点数，这里是输出用户的余额，保留两位小数。

(3) `AddSecret`函数用于加密用户的密码，并使用`printf`函数输出加密后的密码。 循环结束后，所有用户的信息都被打印出来。

2.3 void printAll()

```
//银行总金额
void printAll()
{
    double all=0.0;
    for(int i=0;i<count;i++)
    {
        all+=users[i].balance;
    }
     printf("银行的总存款为%lf\n",all);
}
```

首先定义了一个名为 printAll 的函数，没有参数和返回值。在函数内部，声明并初始化一个变量 all，用于存储总存款金额，初始值为 0.0。使用循环遍历所有的用户，通过访问每个用户的 balance 属性获取其存款金额，并将其累加到变量 all 中。最后，使用 printf 函数打印出银行的总存款金额，格式化输出 all 的值。

2.4 void saveFile(User* pr, int count)

```
void saveFile(User* pr, int count)
{
    //二进制写
    FILE* file = fopen("save.bat", "wb");
    //防御性编程
    if (file == NULL)
    {
        printf("无法打开文件。\n");
        return;
    }
    //将用户人数存进文件
    fwrite(&count, sizeof(int), 1, file);
    //将每个用户的个人信息存入文件
    //结构体数组名及地址
    fwrite(pr, sizeof(User), count, file);

    fclose(file);//好习惯，随手关闭文件
}
```

这段代码定义了一个函数名为`printAll`，其含义是打印银行的总金额，即所有用户的余额之和。声明了一个`double`类型变量`all`并初始化为 0.0，用于存储银行的总金额。 使用`for`循环遍历用户数组，循环变量`i`从 0 到`count-1`。在循环内部，将每个用户的余额累加到`all`变量中，即`all+=users[i].balance`。 循环结束后，所有用户的余额都被累加到`all`变量中，表示银行的总金额。使用`printf`函数输出银行的总存款金额，格式化字符串中的`%lf`用于输出`double`类型的变量。

2.5 void showBalance()

```
    void showBalance() //查看余额
    {
        printf("您的账户余额为: %.2f\n", pr->balance);
    }
```

这段代码定义了一个函数名为`showBalance`，其含义是显示用户的账户余额。

使用`printf`函数输出提示信息，表示要显示用户的账户余额。使用格式化字符串`%.2f`将`pr->balance`的值作为浮点数输出，保留两位小数。- `pr`是一个结构体指针，表示当前用户的信息。- `balance`是`pr`结构体中的一个字段，表示用户的账户余额。

六、课程设计总结

6.1 模块总体实现情况：

1. 用户认证模块：
   (1) 实现情况：该模块完成了用户身份验证功能，使用用户名和密码进行认证，确保只有授权用户可以访问系统。
   (2) 评价：用户认证模块的实现符合预期，并能够有效地确保系统的安全性。

2. 资金管理模块：
   (1) 实现情况：该模块成功管理用户的账户余额、存款和取款等资金操作，包括更新用户余额并保证事务的原子性。
   (2) 评价：资金管理模块的实现满足课程要求，能够准确地处理用户的资金操作，并确保数据的一致性。

3. 交易记录模块：
   (1) 实现情况：该模块能够记录用户的交易历史并生成交易日志，包括交易类型、金额、日期和时间等详细信息。
   (2) 评价：交易记录模块的实现较好，能够有效地记录用户的交易信息，方便

后续的查询和审计。

4. 并发控制模块：

(1) 实现情况：该模块处理多个用户同时访问系统的并发操作，使用互斥锁等并发控制机制确保操作的正确性。

(2) 评价：并发控制模块的实现满足课程要求，能够有效地处理并发操作，防止数据竞争和不一致性的问题。

5. 异常处理模块：

(1) 实现情况：该模块能够捕捉和处理用户输入错误、系统故障或其他异常情况，提供友好的错误提示信息。

(2) 评价：异常处理模块的实现良好，能够有效地处理各种异常情况，提高系统的稳定性和用户体验。

6.2 系统简要客观评价：

基于上述模块的实现情况，可以对整个系统进行简要客观评价：

1. 功能完整性：系统实现了 ATM 系统管理所需的基本功能，包括用户认证、资金管理、交易记录等，能够满足课程要求和课题功能。

2. 安全性：系统具备用户认证功能，使用用户名和密码进行身份验证，确保只有授权用户可以访问系统，提高了系统的安全性。

3. 数据一致性：资金管理模块保证了用户

账户余额的正确性，并采取了并发控制措施，确保数据的一致性。

4. 错误处理：系统能够捕捉和处理用户输入错误和其他异常情况，提供友好的错误提示信息，增强了系统的健壮性和用户体验。

5. 日志记录：交易记录模块能够准确地记录用户的交易历史和生成交易日志，便于后续的查询和审计。

综合来看，该 ATM 系统管理的 C 语言大作业在模块实现、功能完整性、安全性、数据一致性、错误处理和日志记录等方面都达到了预期要求，并能够满足课程要求和课题功能。然而，具体的评价还需要根据项目的实际情况和需求来进一步衡量。

6.3 存在问题及改进思路

在完成 C 语言的大作业，主题为 ATM 系统管理时，以下是一份整理出的条理清晰的结果，包括存在的问题或不足、分析原因以及提出改进思路或对策。

6.4 存在问题或不足

1. 用户界面设计不够友好：系统的用户界面可能存在不够直观和易用的问题，可能会给用户造成困扰或操作上的困难。

2. 错误处理不完善：系统对用户输入错误或其他异常情况的处理可能不够充分，导致错误信息不明确或无法及时解决问题。

3. 并发控制不够健壮：在多用户同时访问系统时，可能存在并发控制方面的问题，可能导致数据竞争或一致性问题。

6.5 分析原因：

(1) 技术限制：在完成大作业的过程中，可能受限于 C 语言的特性和功能，导致用户界面设计上的不足。

(2) 时间压力：在完成大作业时，时间可能是一项限制因素，可能导致在错误处理和并发控制方面无法充分考虑和实现。

(3) 缺乏合作与反馈：如果没有充分的合作与反馈机制，可能导致对问题和不足的发现和解决能力有所欠缺。

改进思路或对策：

1. 用户界面设计改进：
   (1) 采用更直观、易用的界面设计，考虑使用菜单、指令提示等方式，提高用户操作的可理解性和友好性。
   (2) 可以进行用户测试和反馈收集，以便根据用户意见和建议对界面进行改进。
2. 错误处理优化：
   (1) 完善错误处理机制，确保用户输入错误时能够提供明确的错误提示信息，帮助用户快速定位和解决问题。
   (2) 对于系统异常情况，如数据库连接失败等，应提供适当的错误处理和恢复机制，以确保系统的稳定性和可靠性。
3. 并发控制增强：
   (1) 采用更健壮的并发控制机制，如信号量、互斥锁等，以避免并发操作引起的数据竞争和一致性问题。
   (2) 进行充分的测试和性能评估，以确保并发控制机制能够在高负载和多用户环境下正常运行。
4. 加强合作与反馈：
   (1) 如果有团队合作的机会，可以与团队成员共同交流和合作，共同解决问题和

优化系统设计。

(2) 如果没有合作机会，可以主动寻求他人的意见和反馈，例如向同学、教师或论坛社区等请教和分享经验。

6.6 课程设计心得：

1.目的：通过完成 ATM 系统管理的大作业，我成功地将所学的 C 语言知识应用到实际项目中，并实践了课程所要求的软件设计和开发过程。

2. 学到了什么：通过这个大作业，我学到了很多关于 C 程序设计的技巧和方法。我深入了解了 C 语言的数据结构和函数的使用，学会了模块化设计和代码重用的重要性。

3. 对 C 程序设计的新认识：这个大作业让我更深入地理解了 C 程序设计的复杂性和挑战性。我意识到编写高质量、可靠和可维护的代码需要细致的规划、良好的架构和合理的代码组织。

4. 进度管理和计划准备：在完成大作业的过程中，我学会了更有效地管理时间和进度。我制定了详细的计划和任务分解，合理分配时间和资源，确保项目按时完成。

5. 团队合作的重要性：尽管这个大作业是个人完成的，但我也意识到团队合作在项目开发中的重要性。与同学、教师和论坛社区的交流和讨论，让我受益匪浅，从不同的角度获取了宝贵的反馈和建议。

6.7 感想：

完成这个 C 语言的大作业是一次充满挑战但又充满收获的经历。我意识到编程不仅仅是解决问题，更是一种创造和艺术的体现。我体会到了代码的美感和优雅，并更加注重代码的可读性和可维护性。

此外，大作业的完成也让我更深入地了解了软件开发的过程和步骤。从项目规划、需求分析到设计、实现和测试，我学会了如何进行全面的项目管理，确保项目的质量和进展。

最后，这个大作业让我更加热爱和深入探索编程的世界。我对 C 语言和程序设计有了更深入的认识，也明白了自己在编程领域的不断成长和进步。我期待在以后的学习和实践中继续提升自己的技能和能力，不断追求编程的卓越。

七、建议

1. 强化实践性教学，C 程序设计是一门实践性很强的学科，更多的实践和实验环节可以帮助学生更深入地理解和掌握知识。增加实际案例的分析和编码练习，让学生亲自动手实现具体的应用程序，将理论知识与实际应用结合起来。

2. 提供更多的案例和示例，为了激发学生的学习兴趣和培养实际问题解决能力，可以提供更多的案例和示例，涵盖不同领域和应用场景。这样可以帮助学生更好地理解概念，并能够将学到的知识应用到实际项目中。

3. 引入团队合作和项目管理，C 程序设计往往涉及到大型项目的开发，引入团队合作和项目管理的概念和实践，可以培养学生的合作能力、沟通能力和项目管理能力。通过小组项目或跨学科合作项目，让学生体验实际项目开发过程中的挑战和机会。

4. 加强实时反馈和指导，及时的反馈对学生的学习和进步至关重要。教师可以通过定期的代码评审、作业批改和个别指导，给予学生具体的反馈和建议。这样可以帮助学生及时纠正错误、改进代码，并在学习过程中不断提升。

5. 鼓励创新和开放思维，C 程序设计是一个充满创造力和开放性的领域，鼓励学生提出自己的创新想法和解决方案。教师可以鼓励学生参与编程竞赛、开源项目或开发个人项目，培养学生的创新意识和实际问题解决能力。