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：22030531

姓　名：杨熙承

指导教师：曹中心

2023 年 6 月

ATM 模拟系统

一、课程认识

1.1 目的

ATM 管理系统的目的在于模拟银行 ATM 的常见功能,如账户注册、登录、存款、取款、转账、改密等,并实现一定的用户管理和数据持久化。本报告旨在介绍 ATM 模拟管理系统的设计与实现。通过这个课程项目，我们将深入了解 C 语言的应用，并通过实践掌握软件开发过程中的基本原理和技能。该课程旨在培养学生的编程能力、系统设计能力以及解决实际问题的能力。

1.2 性质

本课程是一门实践性较强的课程，要求学生结合理论知识和实际应用，通过设计和实现 ATM 管理系统，掌握软件开发的流程和方法。通过此项目，学生将运用 C 语言进行软件开发，了解与实际工程开发相结合的软件设计方法。本课程属于应用型课程,要求学生熟练掌握 C 语言的基本语法和常用函数,并能运用所学知识开发一个完整的，实用的程序项目。

1.3 任务

1.3.1 基本要求

(1) 实现用户注册、登录功能

(2) 实现账户余额查询、存款、取款、转账功能

(3) 实现密码修改功能

(4) 实现简单的数据持久化,保存用户数据到二进制文件

(5) 实现简单的用户管理功能,如用户删除

(6) 实现验证码和密码加密功能,提高系统安全性

(7) 实现用户交易记录和银行终总额查询功能

(8) 实现超限提醒和账户冻结功能

1.3.2 扩展要求

(1) 实现定期邮件提醒用户余额和近期交易情况

(2) 实现账户注销功能

(3) 实现可视化图形化界面

(4) 增加更丰富的管理员功能

(5) 增加管理员的查，删，找功能

(6)增加管理员的解封，限额，查看交易记录等功能

二、课题选择

2.1 课题背景

随着科技的发展和社会的进步，ATM（自动取款机）已经成为了现代金融领域中不可或缺的一部分。ATM 的普及和应用方便了人们的日常生活，使得金融交易更加便捷和高效。然而，随之而来的是 ATM 管理的复杂性和挑战性。为了确保 ATM 系统的安全性、可靠性和高效性，需要一个有效的管理系统来监控和控制 ATM 的运作。如今，ATM 自动提款机系统已经广泛应用于各大银行,方便了人们的财务管理和交易。

本课题拟基于 C 语言开发一个 ATM 管理系统,实现 ATM 的常见功能,加深对 C 语言的理解运用。

2.2 意义

(1)熟练掌握 C 语言的基本语法和常用函数。通过开发一个完整的 ATM 管理系统项目,可以深入理解 C 语言各种数据类型、运算符、控制流语句、函数、指针、结构体等概念。

(2)锻炼编程思维和代码实现能力。实现 ATM 各项功能需要思考界面设计、程序流程和算法,编写清晰的代码。这有助于培养分析问题和编程解决问题的能力。

(3)加深对 ATM 工作原理的理解。开发 ATM 管理系统可以加深对 ATM 机理、交易流程和安全机制，用户认证、账户管理的理解。

(4)实现简单的数据持久化。利用文件操作将用户数据保存到文件,可以理解文件的读写原理。

(5)实际应用的意义。一个高效、安全的 ATM 管理系统可以提升金融机构的运营效率，减少人力资源的消耗。同时，它还能够提供可靠的账户管理和交易记录，增强用户对于金融机构的信任度，促进金融行业的发展。

2.3 实用性

(1)模拟真实 ATM 机的使用场景和流程,对新手更加友好。可以作为 C 语言入门的学习项目。

(2)数据存储使用简单文件操作,没有复杂的数据库,易于理解和修改。可以作为文件操作的示例程序。

(3)具有登录验证、账户管理、存取款转账、改密等基本功能,涵盖常用的结构体、函数、文件操作使用示例。可以作为 C 语言语法的综合练习项目。

(4)有一定的扩展性,可以加入更多功能、优化界面和数据存储、支持多用户等来丰富项目。

(5)ATM 管理系统在实际应用中具有广泛的实用性。它可以用于各类金融机构，包括银行、信用社、金融公司等，用于管理和监控他们的 ATM 网络。通过该系统，机构可以方便地进行账户管理、交易监控、故障排查等操作，提高运营效率和服务质量。

(6)ATM 管理系统还可以为用户提供更便捷的金融服务。用户可以通过 ATM 机进行存款、取款、查询余额等操作，而管理系统可以实时更新用户账户信息，确保交易的准确性和安全性。因此，该系统的实用性不仅体现在金融机构内部管理方面，也直接关系到用户的金融体验和便利性。

三、系统总体分析与设计

3.1 系统总体框架图

如下图所示:

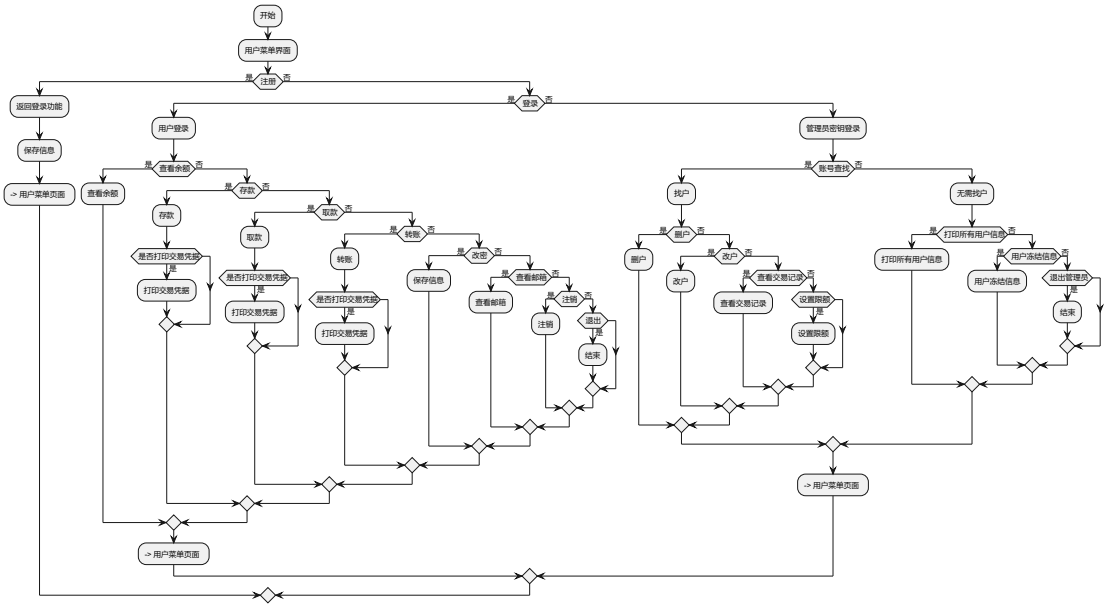

图 3- 1 总系统流程图

3.2 简要功能说明

如图 3-1 所示，这是一个简化版的银行管理系统。主要功能包括用户的注册、登录、存款、取款、转账、改密、查询余额、打印交易凭据等操作。代码中使用了结构体数组来存储用户信息，包括账号、密码、余额、交易记录等字段。同时，还提供了管理员系统，管理员可以查看所有用户信息、删除账户、冻结账户、设置限额等功能。

3.3 功能模块

1. 注册账号:用于注册系统的新用户,要输入账号、密码、验证码等信息。
2. 登录系统:输入账号和密码,并校验验证码登录系统。登录成功后可以进行各种交易操作和信息查询。
3. 查看余额:查看登录用户的账户余额信息。
4. 存款:向登录用户的账户进行存款操作。需要输入存款金额,可以选择是否打印存款凭据。

5. 取款:从登录用户的账户进行取款操作。需要输入取款金额,可以选择是否打印取款凭据。
6. 转账:从登录用户的账户向其他用户账户进行转账操作。需要输入转账金额和目标用户账号,可以选择是否打印转账凭据。
7. 打印交易凭据：用进行交易时，可以打印交易的金额，类型，时间。
8. 改密:修改登录用户的密码。需要输入原密码和新的密码。
9. 查看交易记录:查看最近 20 条登录用户的交易记录,包括交易时间、交易类型、交易金额等信息。
10. 登出:退出当前登录用户,返回到系统主界面。
11. 管理员系统，需要密钥登录

    (1) 查询所有用户

    (2) 修改用户信息

    (3) 查询指定用户

    (4) 冻结/解冻用户账号

    (5) 删除指定用户

    (6) 查看指定用户交易记录

    (7) 退出管理员，回到用户界面

    (8) 查看银行总余额

    (9) 修改指定用户限额

整个系统使用结构体、文件操作、验证码、加密显示等技术,实现了一个基本的 ATM 管理系统。

3.4 开发环境

环境：vscode、neovim、devcpp

编译器：mingw64/bin/gcc(或 devcpp 集成的 gcc)

流程图实现：www.plantuml.com

3.5 运行界面

代码运行后会得到以下界面,以下是部分界面的展示：

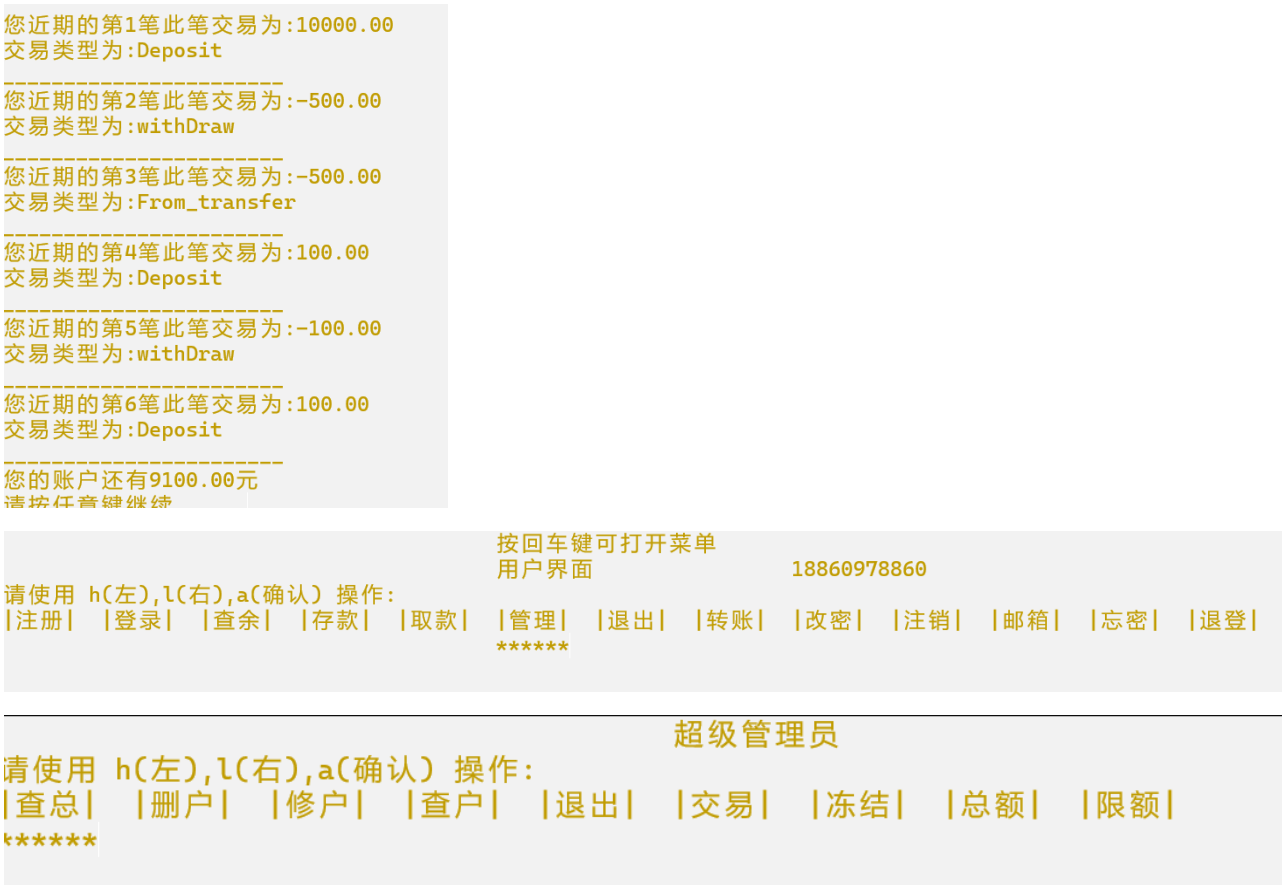

图 3- 2 运行界面展示

小组分工：

杨熙承（组长）：整体规划、界面设计、用戶注册登录（部分）、密码加密、存款取款转账（部分）、交易记录查询、管理员系统、文件读取操作、延迟显示、删除操作、打印凭据操作、注销、冻结、限额、交易记录实现、忘记密码实现。

王语桐：用戶注册与登录、存款、取款、转账、余额查询、交易记录查询、余额查询。

李硕：改密、用戶信息管理、文件操作、一种改密、二种改密。

四、模块详细设计

4.1 加密输入模块

加密输入模块主要实现输入用户密码，管理员密钥时，在终端只显示*号，增强管理系统的安全性，输入后可以存放到字符数组中。加强系统安全性。

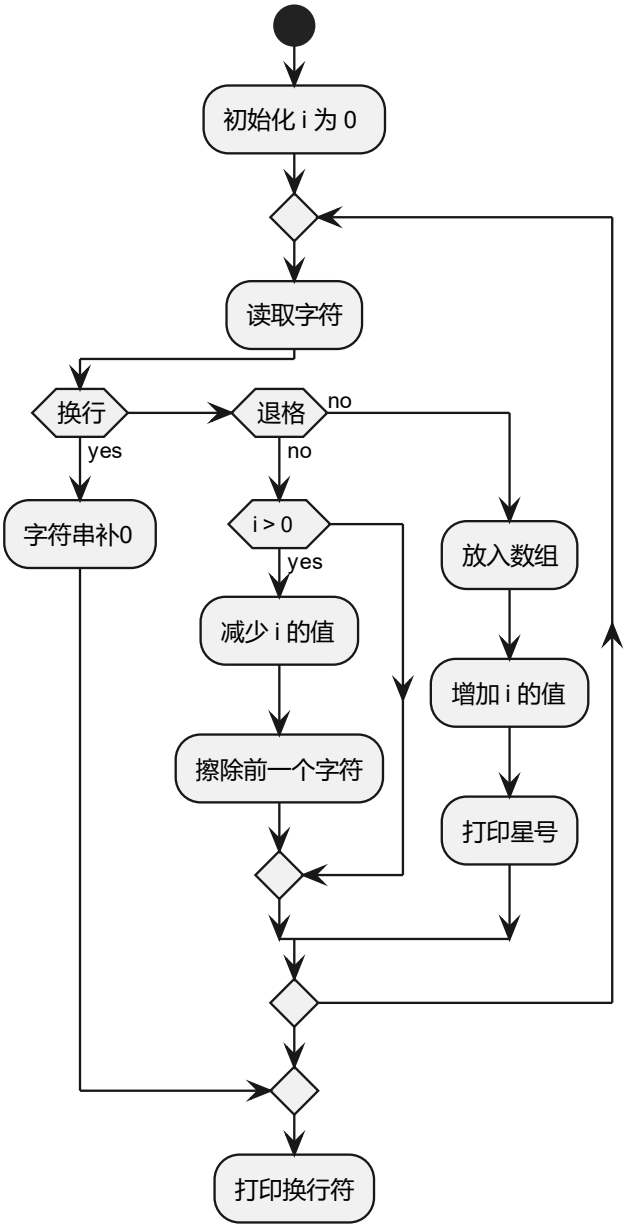

图 4- 1 加密流程图

如图 4-1 所示：

初始化变量 i 为 0，用于记录字符在数组中的索引位置。进入一个循环，在循环中，首先获取一个字符 c。使用条件判断语句"if"检查字符 c 是否等于回车符'\r'。如果字符 c 等于回车符，则执行":结束循环"操作，即跳出循环。如果字符 c 不等于回车符，则进入下一个条件判断语句。使用条件判断语句"if"检查字符 c 是否等于退格符'\b'。如果字符 c 等于退格符，则执行":擦除前一个字符"操作，即删除数组中的前一个字符。如果字符 c 不等于退格符，则执行":保存字符 c 到 s 数组"操作，即将字符 c 保存到一个名为 s 的数组中。执行"i 自增"操作，将变量 i 的值加 1，用于指向下一个字符在数组中的位置。执行"打印 *号"操作，即输出一个星号。结束条件判断语句。回到步骤 4，继续下一次循环，读取下一个字符并重复以上操作。当遇到回车符时，跳出循环。程序结束

4.2 验证码模块

验证码模块通过 time.h 库文件的调用，实现了生成随机四位的验证码，用户在登录和注册时需要验证。加强系统安全性。

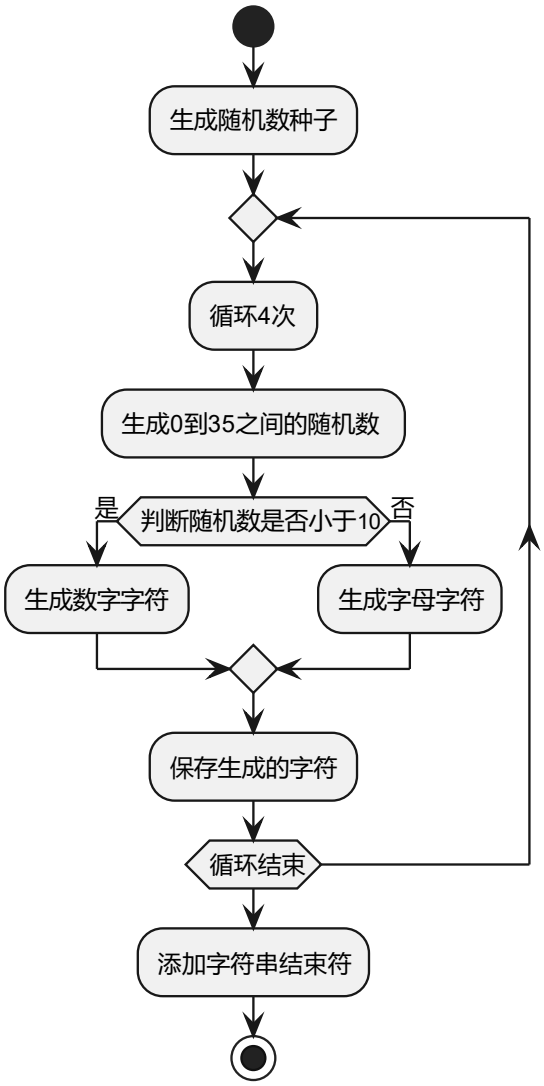

图 4- 2 验证码流程图

如图 3 所示

(1) 生成随机数种子：生成一个随机数种子，用于生成伪随机数序列。

(2) 重复循环：进行以下步骤 4 次。

(3) 生成 0 到 35 之间的随机数：生成一个介于 0 到 35 之间的随机数。

(4) 判断随机数是否小于 10：判断上一步生成的随机数是否小于 10。

(5) 如果是：执行以下步骤。

(6) 生成数字字符：生成一个数字字符。

(7) 否则：执行以下步骤。

(8) 生成字母字符：生成一个字母字符。

(9) 保存生成的字符：将生成的数字字符或字母字符保存起来。

(10) 循环结束：判断是否完成了 4 次循环。

(11) 如果循环未结束，则回到步骤 4 继续执行。

(12) 添加字符串结束符：在生成的字符序列末尾添加一个字符串结束符。

4.3 队列加密模块

运用数据结构中的队列对密码进行加密，管理员查看用户密码时只可看到加密后的密码，增强了安全性。

加密原理如下表所示：

Table 1 加密表格

q[0]	q[1]	q[2]	q[3]	q[4]	q[5]	q[6]	q[7]	q[8]	q[9]	q[10]	q[11]
0	6	3	1	7	5	8	9	2	4		
	head									tail	
0	6	3	1	7	5	8	9	2	4		
		head								tail	
0	6	3	1	7	5	8	9	2	4	3	
		head									tail
0	6	3	1	7	5	8	9	2	4	3	
			head								tail

4.4 打印凭据模块

在用户进行转账，存款，取款后，可以选择答应交易凭据，增强了代码的实用性

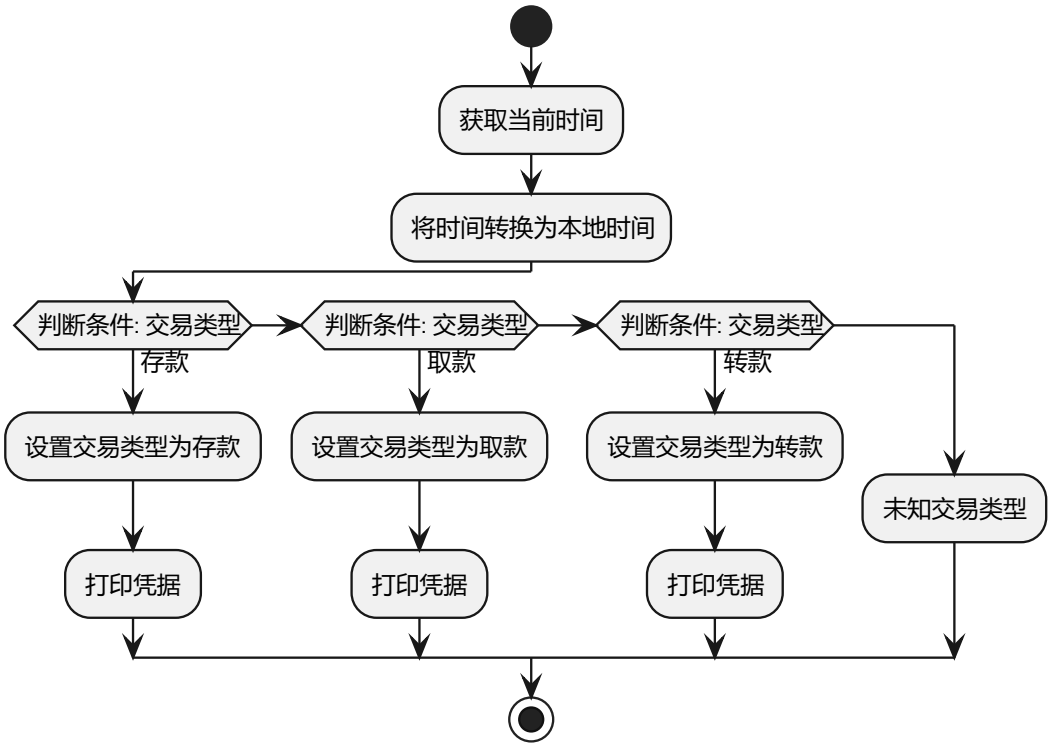

图 4- 3 打印流程图

如图 4-3 所示：

(1) 获取当前时间。

(2) 将时间转换为本地时间。

(3) 如果判断条件指示交易类型为存款，则执行以下步骤：

(4) 设置交易类型为存款,打印凭据。

(5) 否则，如果判断条件指示交易类型为取款，则执行以下步骤：

(6) 设置交易类型为取款,打印凭据。

(7) 否则，如果判断条件指示交易类型为转款，则执行以下步骤：

(8) 设置交易类型为转款,打印凭据。

(9) 否则，执行其他步骤。

4.5 删除模块

4.5.1 交易记录更新

删除模块的本质是一种伪删除，将某个数组删除，整体向前，永远保持最近的 20 条记录。

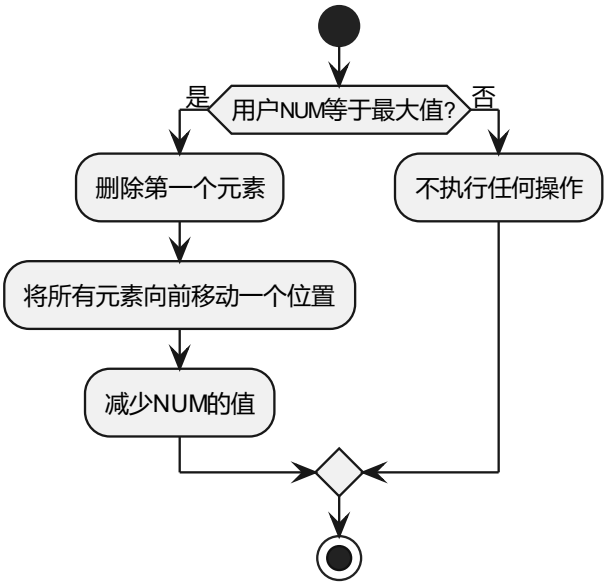

图 4- 4 交易记录流程图

如图 4-4 所示

(1) 判断用户 NUM 是否等于最大值。

(2) 如果用户 NUM 等于最大值：删除第一个元素：删除数据中的第一个元素。将所有元素向前移动一个位置：将数据中的所有元素往前移动一个位置，覆盖删除的第一个元素。减少 NUM 的值：将 NUM 的值减少一个。

(3) 如果用户 NUM 不等于最大值，则不执行任何操作。

4.5.2 删除账号

管理员系统中，查找到相应的账号后进行删除，本质也是伪删除。

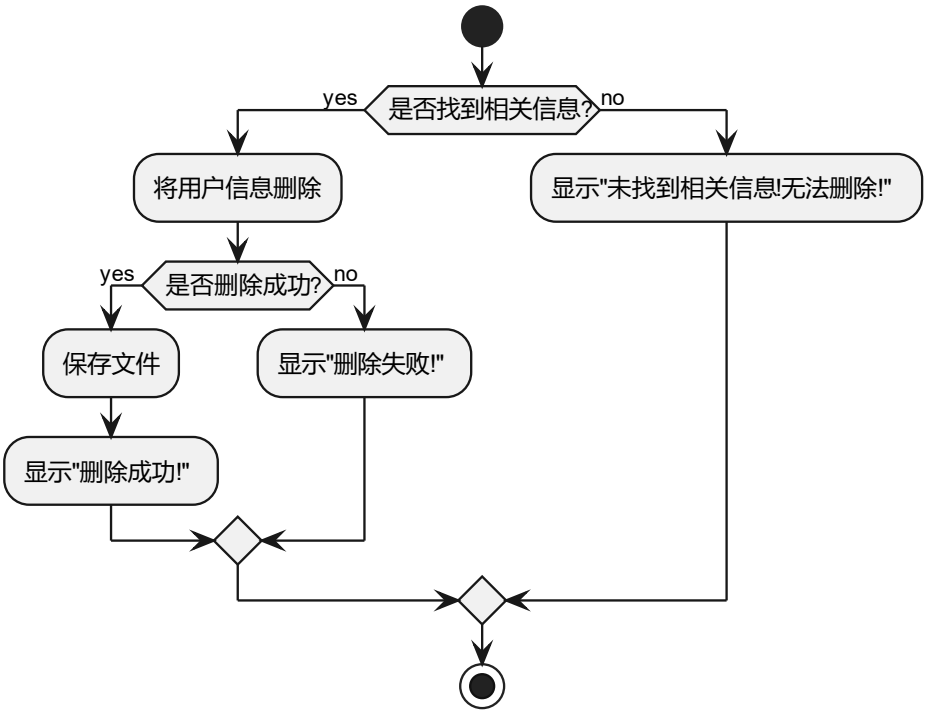

图 4- 5 删除账号流程图

如图 4-5 所示

检查是否找到相关信息。如果是，执行以下操作：将用户信息删除。检查是否删除成功。如果删除成功，执行以下操作：保存文件。显示"删除成功!"。如果删除失败，执行以下操作：显示"删除失败!"。如果不是，执行以下操作：显示"未找到相关信息!无法删除!"。

4.6 判断强密码模块

当用户注册账号时，需要一个同时包含大写字母，小写字母，以及数字强密码，用此函数做判断，增加安全性。

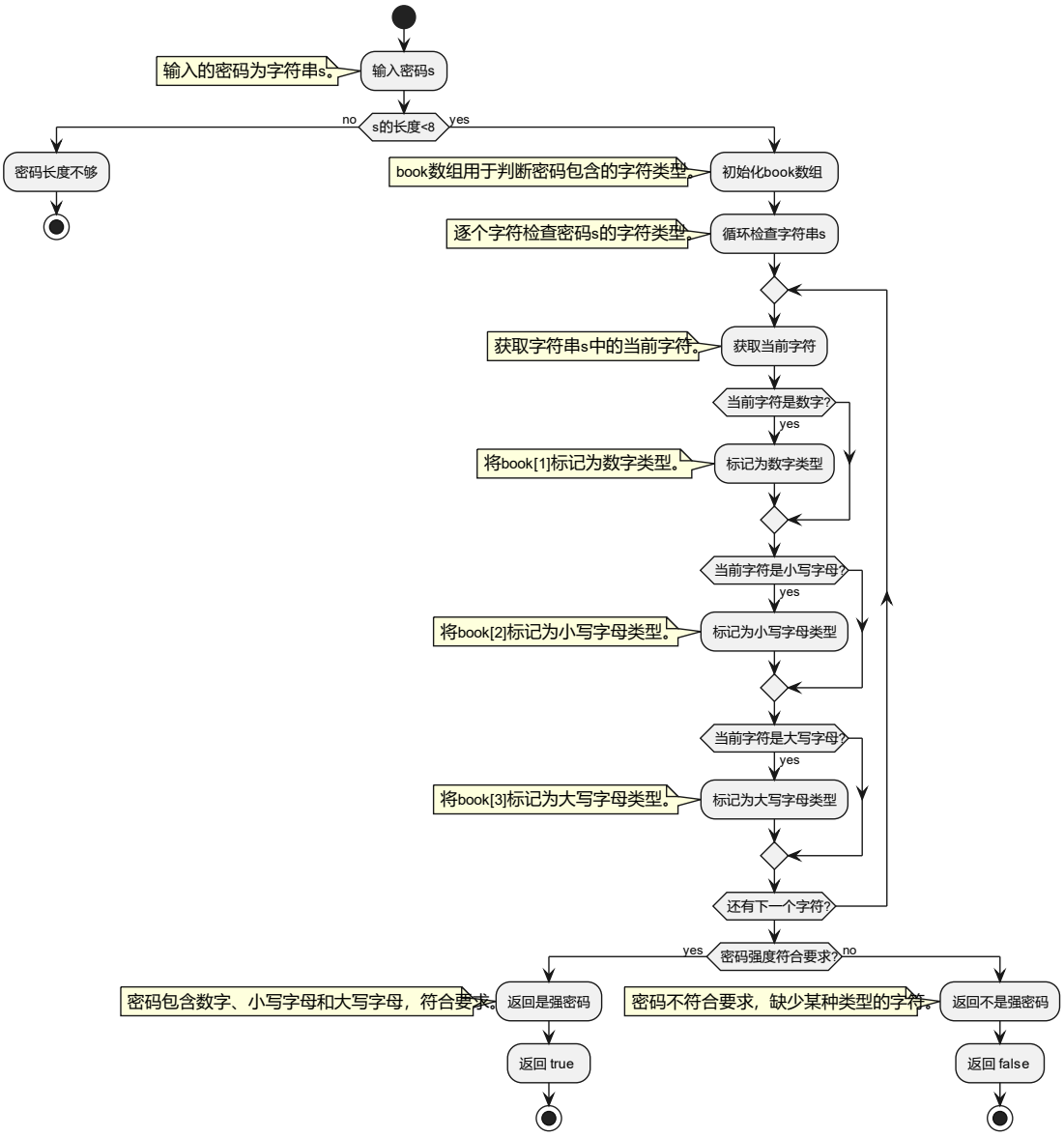

图 4- 6 强密码流程图

如图 4-6 所示

(1) 这段代码实现了一个密码强度检查的逻辑。它接受一个密码作为输入，并根据一定的规则判断密码是否符合强密码的要求。

(2) 代码开始处使用了一个起始符号，表示程序的开始。

(3) 然后，代码会检查输入的密码的长度。如果密码长度小于 8 个字符，就会输出"密码长度不够"并停止程序。如果密码长度大于等于 8 个字符，则会执行后续步骤。

(4) 接下来，代码会初始化一个名为"book"的数组，用于记录密码中包含的字符类型。这里使用了一个便签来解释数组的作用。

(5) 然后，代码会进入一个循环，逐个检查密码的字符类型。

(6) 在循环中，代码会获取当前字符，并根据字符的特征判断它是数字、小写字母还是大写字母，并相应地将"book"数组中对应的位置标记为相应的字符类型。这里也使用了便签来解释每个条件判断的作用。

(7) 循环会一直执行，直到检查完密码中的所有字符。

(8) 在循环结束后，代码会判断密码的强度是否符合要求。如果"book"数组中标记了数字、小写字母和大写字母类型，即密码包含这三种字符类型，那么就会输出"是强密码"并停止程序，同时返回 true。如果"book"数组中有某种类型的字符标记缺失，即密码不符合要求，那么就会输出"不是强密码"并停止程序，同时返回 false。

4.7 可视化模块

用于在 cmd 中尽可能设计一点可视化界面，使得程序更逼真

4.7.1 延迟函数

通过延迟函数，模拟 ATM 加载的过程。

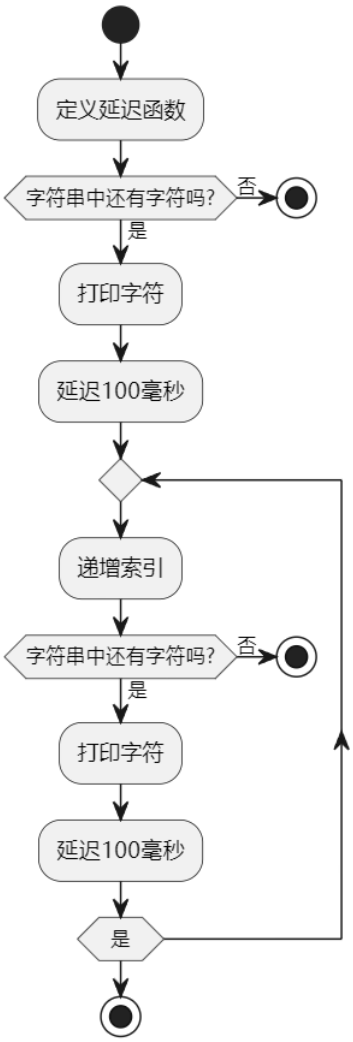

图 4- 7 延迟流程图

如图 4-7 所示

(1) 定义延迟函数：定义了一个函数，用于延迟一段时间后执行下一步操作。

(2) 打印字符：检查字符串中是否还有字符，如果有，则打印当前字符。

(3) 延迟 100 毫秒：等待 100 毫秒，以便字符能够逐渐打印出来。

(4) 递增索引：增加索引，以便获取下一个字符。

(5) 重复步骤 3-5：重复执行步骤 3 至 5，直到字符串中没有剩余字符为止。

4.7.3 特色菜单模块

使用一种全新的特色菜单界面，用户通过 h 左移，l 左移，a 确认。增加一定的趣味性和真实性。

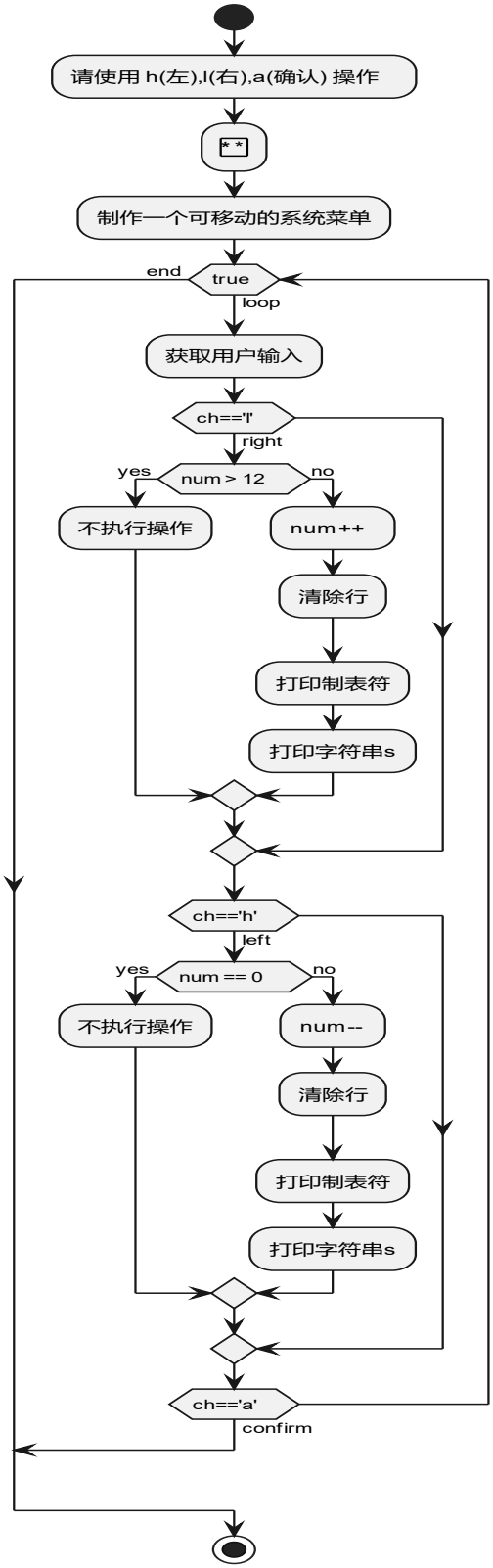
请使用 h(左),l(右),a(确认) 操作
制作一个可移动的系统菜单
end
true
loop
获取用户输入
ch=='l'
right
yes
num > 12
no
不执行操作
num ++
清除行
打印制表符
打印字符串s
ch=='h'
left
yes
num == 0
no
不执行操作
num --
清除行
打印制表符
打印字符串s
ch=='a'
confirm

图 4- 8 特色菜单流程图

如图 4-8 所示

(1) 提示用户使用"h"（向左移动）、"l"（向右移动）和"a"（确认）操作。

(2) 绘制菜单界面。

(3) 进入循环，不断执行以下步骤：

(4) 获取用户输入。

(5) 如果用户输入为"l"（向右移动）：

(6) 如果当前位置（num）大于 12，则不执行任何操作。

(7) 如果当前位置（num）小于等于 12：

(8) 增加当前位置（num）的值。

(9) 清除当前行的内容。

(10) 打印制表符。

(11) 打印字符串 s。

(12) 如果用户输入为"h"（向左移动）：

(13) 如果当前位置（num）等于 0，则不执行任何操作。

(14) 如果当前位置（num）不等于 0：

(15) 减少当前位置（num）的值。

(16) 清除当前行的内容。

(17) 打印制表符。

(18) 打印字符串 s。

(19) 如果用户输入为"a"（确认），则跳出循环。

4.6.4 换色模块

用新色调代替一贯的白字黑框界面

```
    //换色，模拟进入了系统
    system("color f6");
```

4.6.7 密码错误锁定模块

可以动态显示密码错误锁定后的剩余时间。

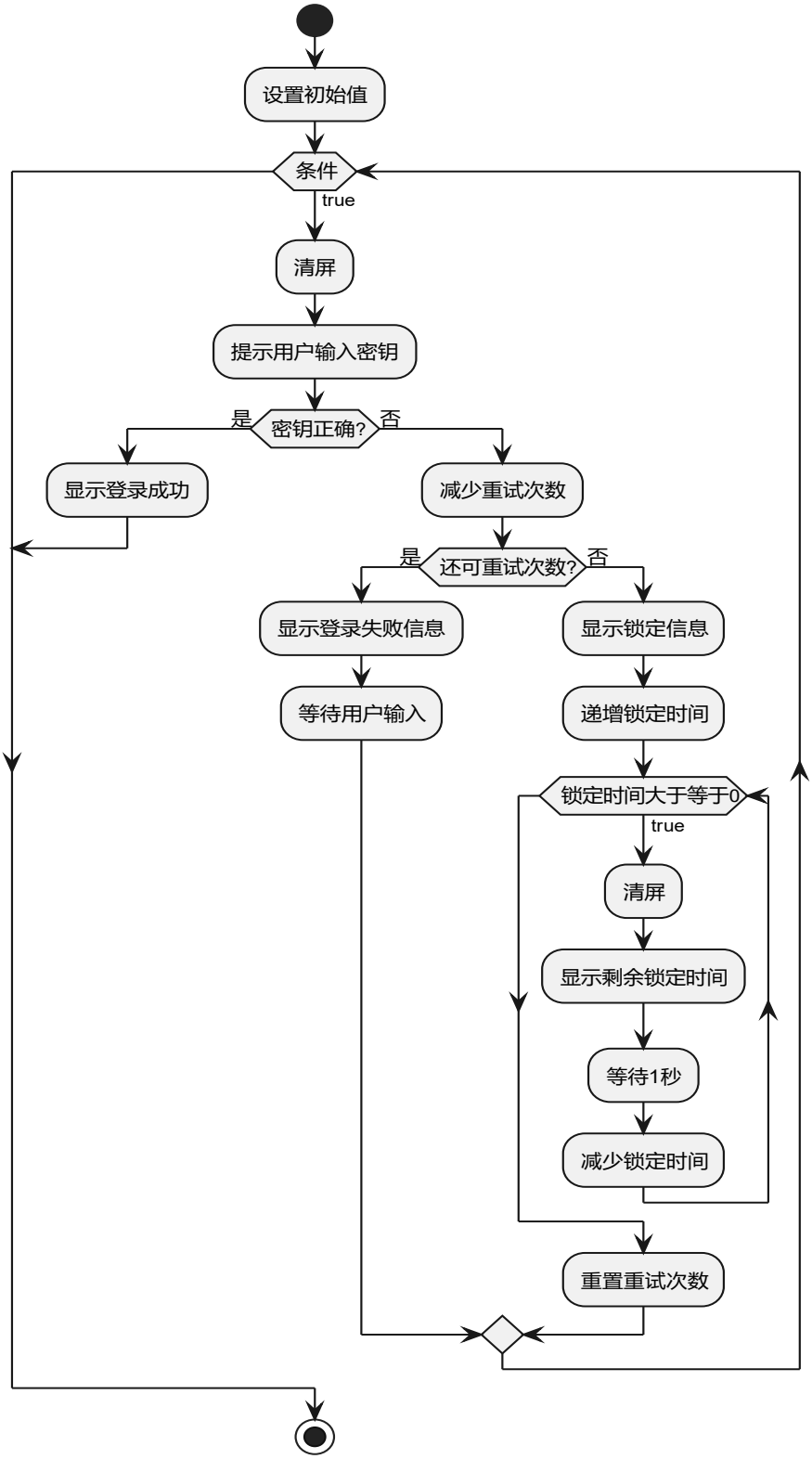
设置初始值
条件
true
清屏
提示用户输入密钥
是
密钥正确?
否
显示登录成功
减少重试次数
是
还可重试次数?
否
显示登录失败信息
等待用户输入
显示锁定信息
递增锁定时间
锁定时间大于等于0
true
清屏
显示剩余锁定时间
等待1秒
减少锁定时间
重置重试次数

图 4- 9 密码锁定流程图

如图 4-9 所示

(1) 系统开始运行，并设置初始值。

(2) 进入循环，检查条件是否为真（通常是检查用户是否尝试登录）。

(3) 清屏并提示用户输入密钥。

(4) 如果密钥正确，显示登录成功，并跳出循环。

(5) 如果密钥不正确：

(6) 减少重试次数。

(7) 如果还有可重试次数，显示登录失败信息，并等待用户输入新的密钥。

(8) 如果没有可重试次数：

(9) 显示锁定信息,递增锁定时间。

4.8 菜单栏模块

4.8.1 普通的用户菜单栏

普通的用户菜单栏引导，操作者可以根据选项进行选择并操作。

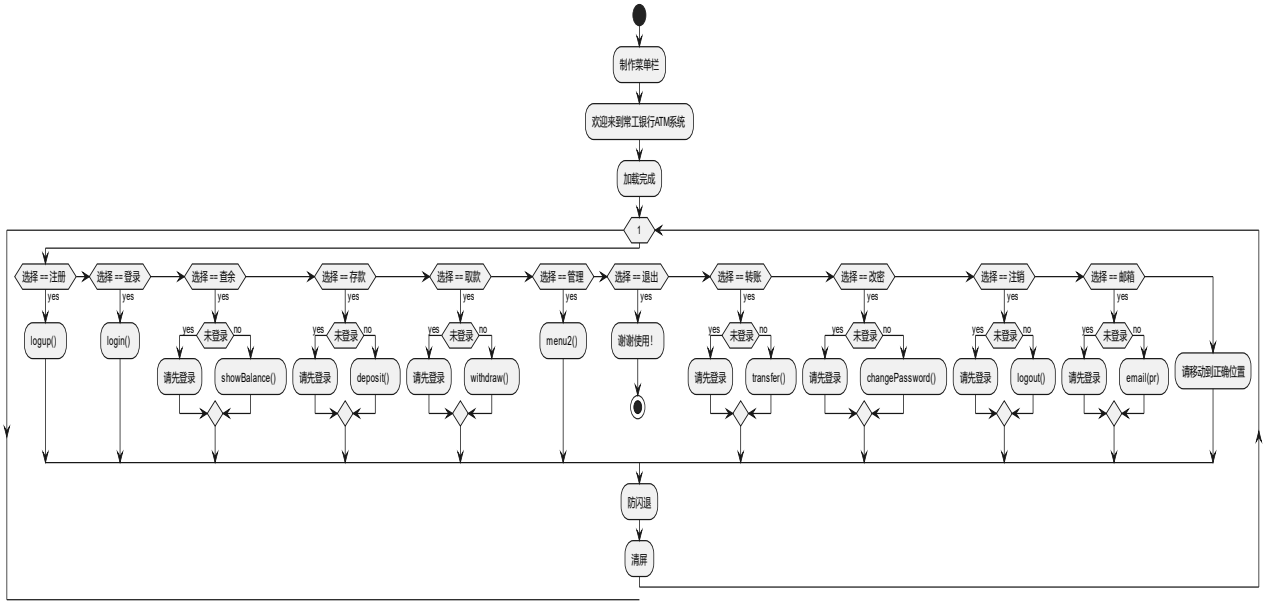

图 4- 10 用户菜单流程图

如图 4-10 所示

(1) 系统开始，并显示制作菜单栏和欢迎信息，然后加载完成。

(2) 进入一个无限循环，直到选择退出。

(3) 根据用户的选择进行不同的操作：

(4) 如果选择是注册，则执行 logup()函数，即用户注册操作。

(5) 如果选择是登录，则执行 login()函数，即用户登录操作。

(6) 如果选择是查余额，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 showBalance()函数，显示余额。

(7) 如果选择是存款，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 deposit()函数，进行存款操作。

(8) 如果选择是取款，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 withdraw()函数，进行取款操作。

(9) 如果选择是管理，则执行 menu2()函数，即进入管理菜单操作。

(10) 如果选择是退出，则显示"谢谢使用！"，并停止循环。

(11) 如果选择是转账，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 transfer()函数，进行转账操作。

(12) 如果选择是改密，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 changePassword()函数，进行改密操作。

(13) 如果选择是注销，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 logout()函数，进行注销操作。

(14) 如果选择是邮箱，则检查是否已登录。如果未登录，则显示提示信息"请先登录"；否则执行 email(pr)函数，进行邮件操作。

(15) 如果选择是其他任何选项，则显示提示信息"请移动到正确位置"。

(16) 在每次循环的末尾，执行防止闪退的操作和清屏操作。

(17)　　循环继续，等待下一次用户的选择。

4.8.2 管理员菜单栏引导

通过普通界面进入管理员系统，操作者输入密钥后可以进行相应操作。

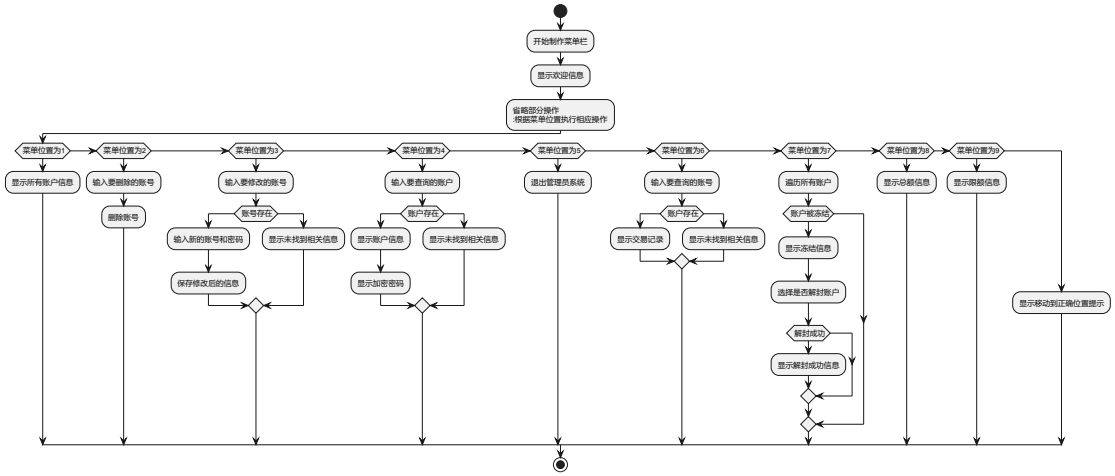

图 4- 11 管理员菜单流程图

如图 4-11 所示

4.9 交易模块

4.9.1 邮箱函数

如图 4-12 所示，用户登录后可查看邮箱，获得交易的记录信息和剩余余额，方便用户查看钱的走向。

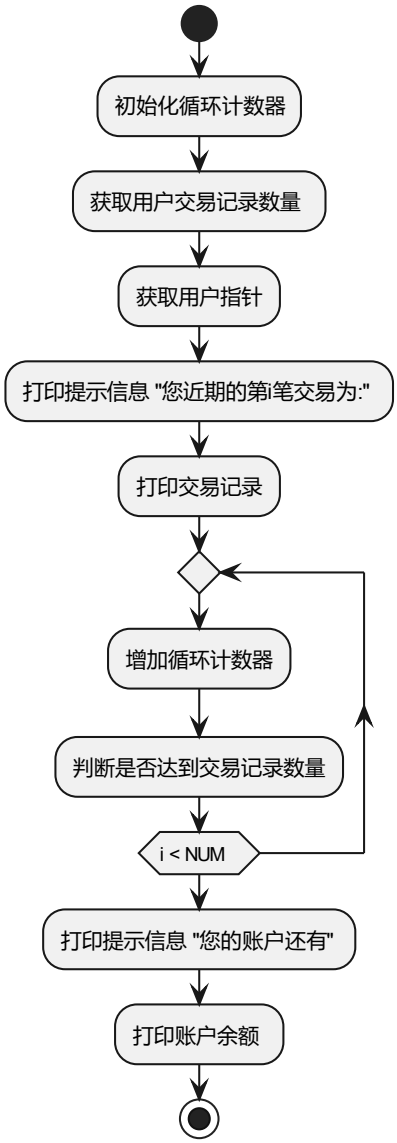

图 4- 12 邮箱流程图

4.9.2 查找交易记录函数

管理员可以指定账号并且打印最近 20 条记录，5 个一行打印。

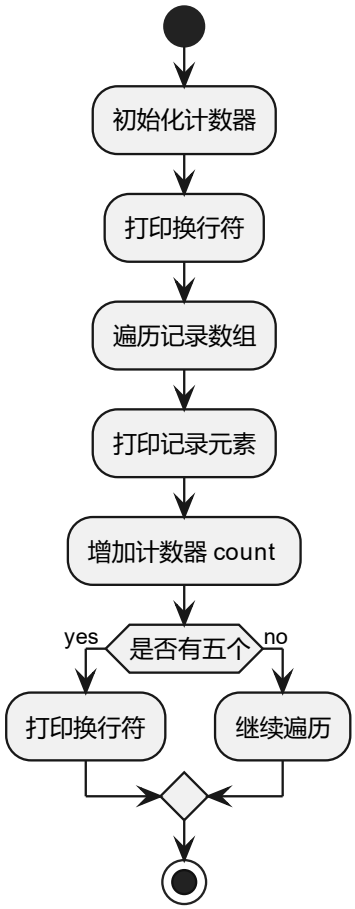

图 4- 13 打印记录流程图

如图 4-13 所示

(1) 初始化变量 i 和 count。打印记录标题。将计数器 count 设置为 0。然后，使用 repeat 关键字表示一个循环块。在循环中，以下操作会被重复执行，直到满足循环条件不再成立：

(2) 打印 s->record[i]，其中 s 是一个对象，表示某个记录，record 是 s 的属性，[i] 表示索引为 i 的元素。增加计数器 count 的值。使用 if 语句检查计数器 count 是否是 5 的倍数。如果是 5 的倍数，则执行以下操作：打印换行符。
如果不是 5 的倍数，则执行以下操作：继续打印当前行。

(3) 打印换行符。

4.10 注销模块

用户可以自行销户，但是需要输入密码

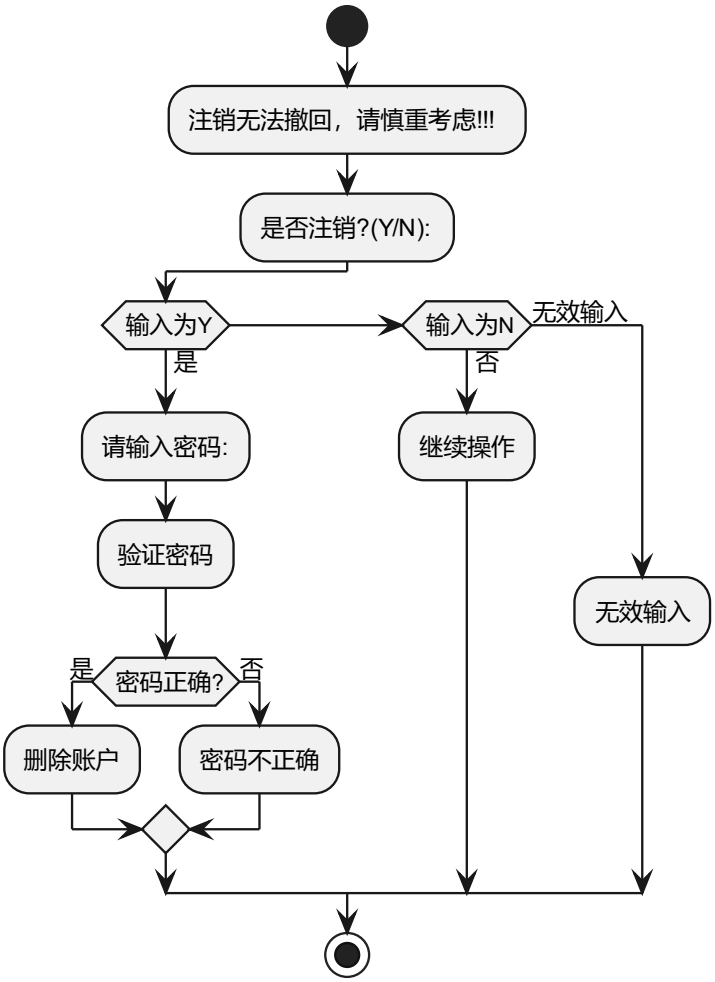

图 4- 14 注销流程图

如图 4-14 所示

(1) 显示提示信息：“注销无法撤回，请慎重考虑!!!”。

(2) 提示用户输入是否要注销账户，用户可以输入 Y（是）或 N（否）。

(3) 如果用户输入为 Y（是），则执行以下步骤：

(4) 提示用户输入密码。

(5) 验证密码是否正确。

(6) 如果密码正确，则删除账户。

(7) 如果密码不正确，则显示提示信息：“密码不正确”。

(8) 如果用户输入为 N（否），则显示提示信息：“继续操作”。

(9) 如果用户输入既不是 Y（是）也不是 N（否），则显示提示信息：“无效输入”。

4.11 限额模块

管理员可以设置指定用户的单次额度。

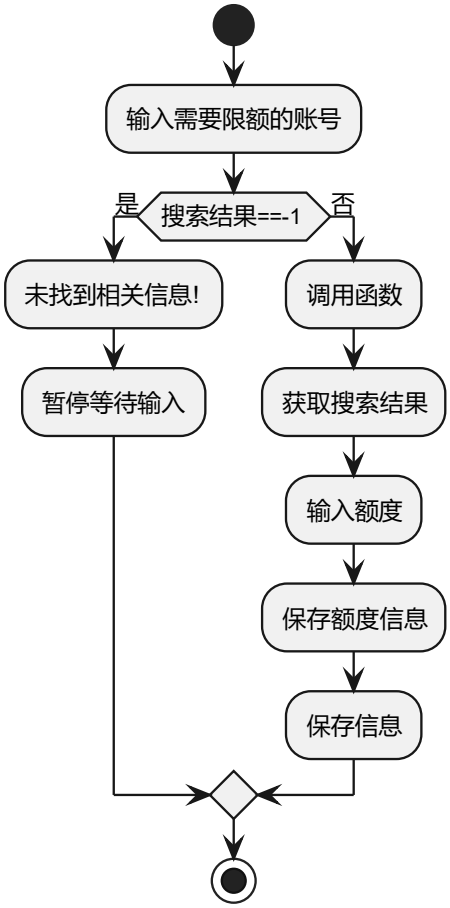

图 4- 15 限额流程图

如图 16 所示

(1) 输入需要限额的账号：这是一个用户输入步骤，用于获取需要设置限额的账号。

(2) 调用 searchResult 函数：这是一个函数调用步骤，用于搜索并检查是否存在与输入的账号相关的信息。

(3) 如果搜索结果等于-1，则执行以下操作：

(4) 未找到相关信息：这是一个输出步骤，表示未找到与账号相关的信息。
暂停等待输入：这是一个暂停步骤，等待用户进行下一步操作。
否则，如果搜索结果不等于-1，则执行以下操作：

(5) 调用 searchResult 函数：再次调用 searchResult 函数来获取搜索结果。
获取搜索结果：这是一个步骤，用于获取与账号相关的搜索结果。
输入额度：这是一个用户输入步骤，用于输入限额信息。
保存额度信息：这是一个步骤，用于将输入的额度信息保存。
保存信息：这是一个步骤，用于保存其他相关信息。

4.12 冻结模块

用户具有是否被冻结的状态，管理员可以解冻用户

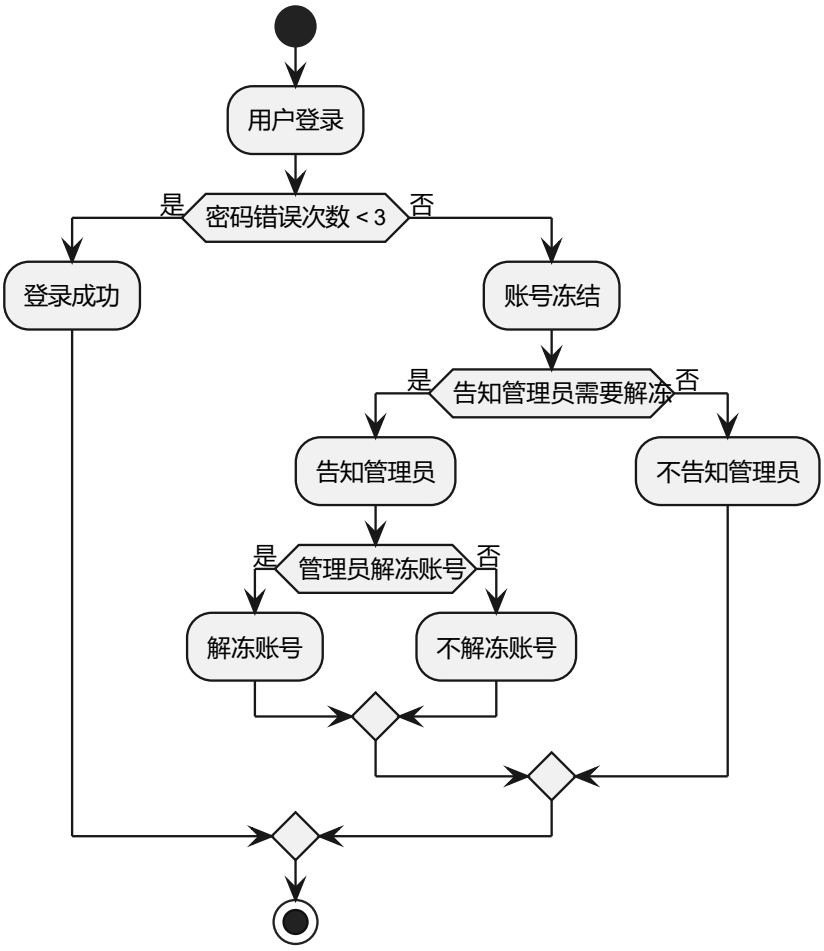

图 4- 16 冻结模块

如图 4-16 所示

用户登录时如果超过三次密码错误，账号会被冻结，之后用户可以选择是否告诉管理员需要解冻，选择’是‘可以告诉管理员，管理员可以知道你有没有被冻结，需不需要解冻，并且可以选择将你解冻。

4.13 注册模块（与组员合作完成）

用户注册账户功能，以及对用户信息的保存和初始化。

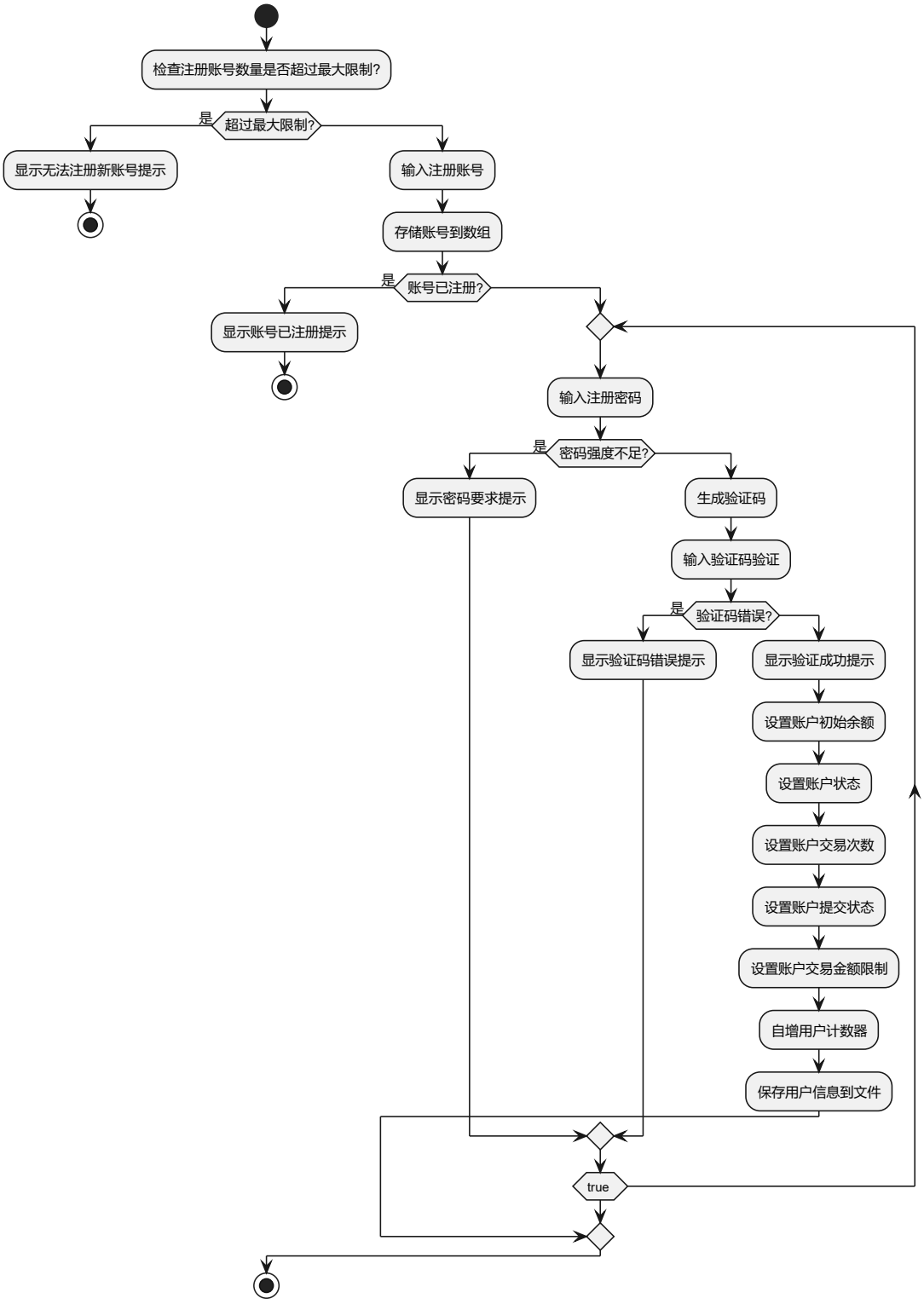
检查注册账号数量是否超过最大限制?
是
超过最大限制?
显示无法注册新账号提示
输入注册账号
存储账号到数组
是
账号已注册?
显示账号已注册提示
输入注册密码
是
密码强度不足?
显示密码要求提示
生成验证码
输入验证码验证
是
验证码错误?
显示验证码错误提示
显示验证成功提示
设置账户初始余额
设置账户状态
设置账户交易次数
设置账户提交状态
设置账户交易金额限制
自增用户计数器
保存用户信息到文件
true

图 4- 17 注册流程图

如图 4-17 所示

(1) 检查注册账号数量是否超过最大限制。

(2) 如果超过最大限制，则显示无法注册新账号的提示，停止注册流程。

(3) 如果没有超过最大限制，则输入注册账号。

(4) 将账号存储到数组中。

(5) 检查账号是否已经注册过。

(6) 如果账号已注册，则显示账号已注册的提示，停止注册流程。

(7) 如果账号未注册过，则重复以下步骤：

输入注册密码。如果密码强度不足，则显示密码要求的提示。否则，生成验证码并输入进行验证。如果验证码错误，则显示验证码错误的提示。如果验证码验证成功，则显示验证成功的提示，并进行以下操作：设置账户的初始余额。设置账户的状态。设置账户的交易次数。设置账户的提交状态。设置账户的交易金额限制。自增用户计数器。保存用户信息到文件。结束循环。

4.14 登录模块（与组员合作完成）

用户登录功能，并进行一系列后续的操作

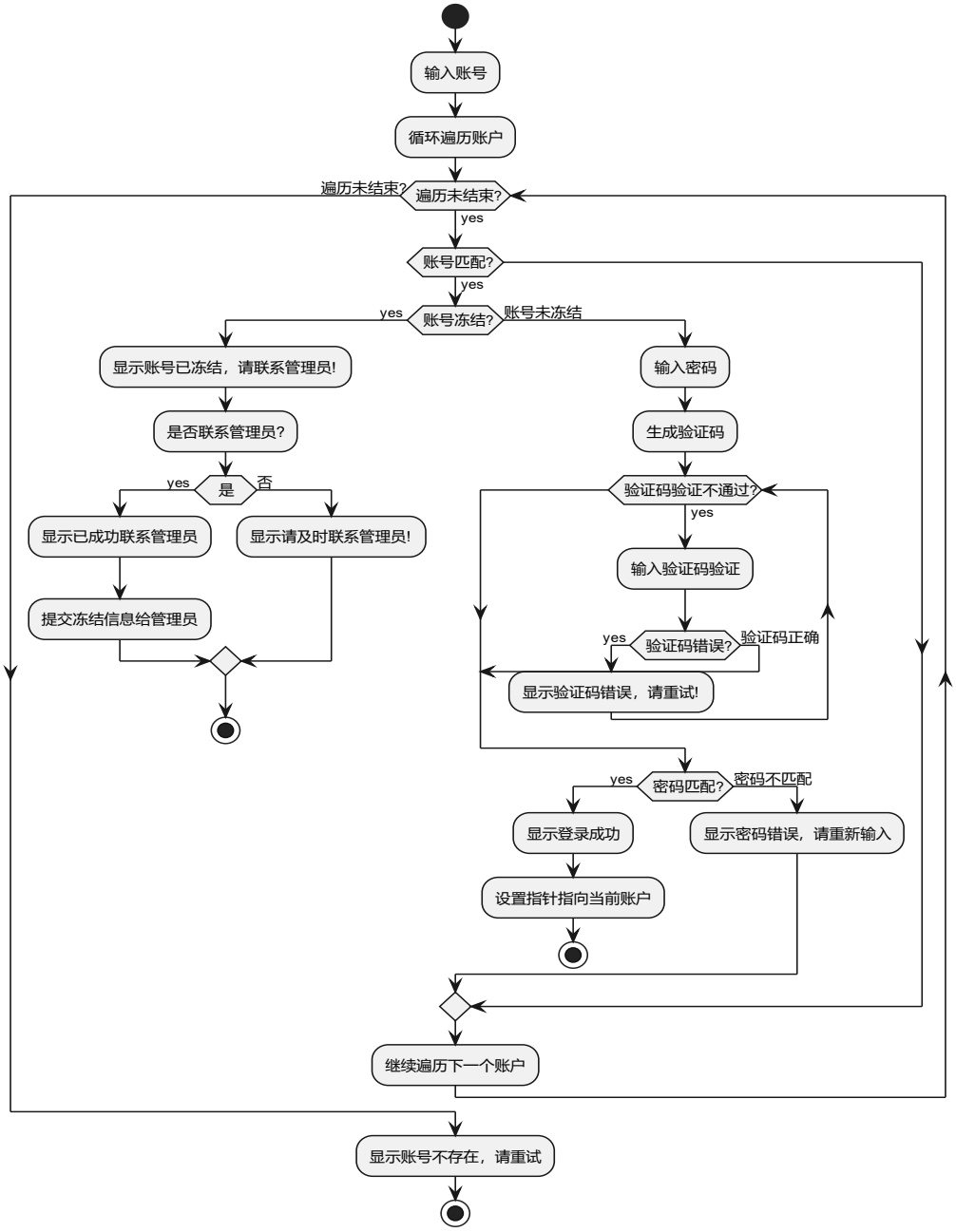

图 4- 18 登录流程图

如图 19 所示

(1) 用户开始操作，输入账号。

(2) 开始循环遍历账户列表。

(3) 如果当前账户与输入的账号匹配，则进入判断账号冻结的条件。

(4) 如果账号已冻结，则显示账号已冻结的提示信息，并询问用户是否联系管理员。

(5) 如果用户选择联系管理员，则显示已成功联系管理员，并提交冻结信息给管理员。

(6) 如果用户选择不联系管理员，则显示请及时联系管理员的提示信息。

(7) 结束程序。

(8) 如果账号未冻结，则要求用户输入密码，并生成验证码。

(9) 进入验证码验证的循环，直到验证码验证通过或用户选择退出。

(10) 如果验证码错误，则显示验证码错误的提示信息，让用户重新输入。

(11) 如果验证码正确，则跳出验证码验证循环。

(12) 进行密码匹配的判断。

(13) 如果密码匹配，则显示登录成功的提示信息，并设置指针指向当前账户。

(14) 结束程序。

(15) 如果密码不匹配，则显示密码错误的提示信息。

(16) 继续遍历下一个账户。

(17) 如果遍历结束仍未找到匹配的账号，则显示账号不存在的提示信息。

4.15 存款，取款，转账模块（部分功能）

存款，取款，转账的**保存，检验记录，打印凭据功能**

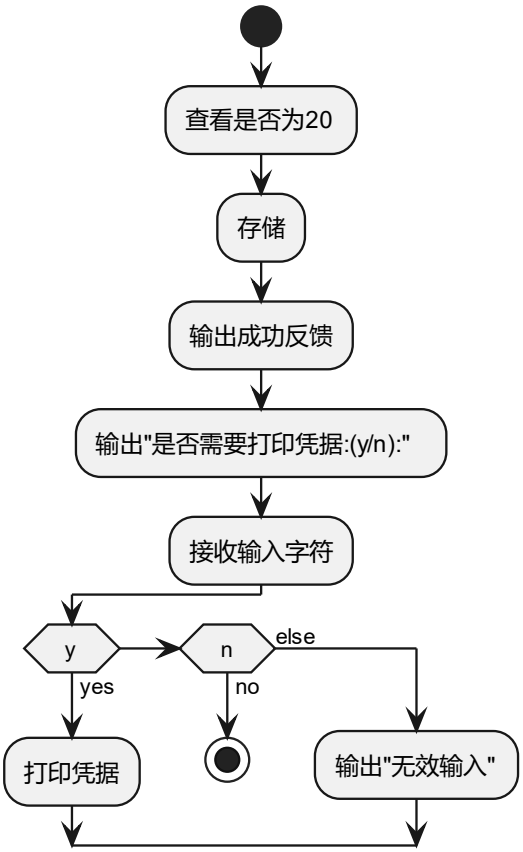

图 4- 19 存，取，转的部分流程图

如图 4-19 所示：

(1) 检查是否为第 20 次操作。

(2) 调用 lastRecord(pr)函数，可能是用来获取上一次操作的记录。

(3) 调用 saveFile(users, count)函数，可能是用来保存用户信息和计数。

(4) 输出"存款/取款/转账成功！"，提示操作成功。

(5) 输出"是否需要打印凭据:(y/n):"，询问用户是否需要打印凭据。

(6) 从用户输入中读取一个值，存储为变量 a。

(7) 如果

(8) a 等于 y 或 Y，则进入下面的判断分支：

调用 printReceipt(amount, 1)函数，可能是用来打印凭据。如果 a 等于'n'或'N'，则停止执行代码。如果以上条件都不满足，则执行下面的操作：输出"无效输入"，提示用户输入无效。

4.16 读取文件模块

用户保存文件后，再次打开程序时，读取文件来预加载信息。

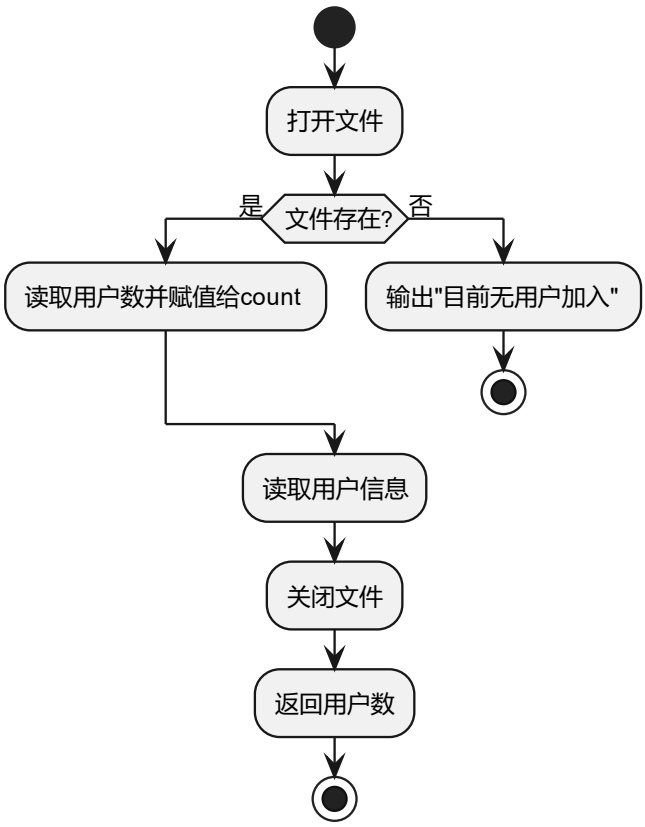

图 4-20 读取文件流程图

如图 4-20 所示：

(1) 打开文件：这个步骤表示代码正在尝试打开一个文件。

(2) 如果文件存在（条件判断），则执行以下步骤：

(3) 读取用户数并将其赋值给变量"count"：代码会从文件中读取用户数，并将该值赋给名为"count"的变量。

(4) 如果文件不存在（条件判断），则执行以下步骤：

(5) 输出"目前无用户加入"：代码会在屏幕上打印出一条消息，表示当前没有任何用户加入。

(6) 终止流程（stop）：代码执行到此处会停止运行。

(7) 无论文件存在与否，接下来的步骤都会执行：

(8) 读取用户信息：代码会读取文件中的用户信息。

(9) 关闭文件：代码在完成文件读取后会关闭该文件。

(10) 返回用户数：代码会返回用户的数量。

4.17 忘记密码模块

当用户忘记密码时，可以通过初始用户注册时的附加条件修改密码。

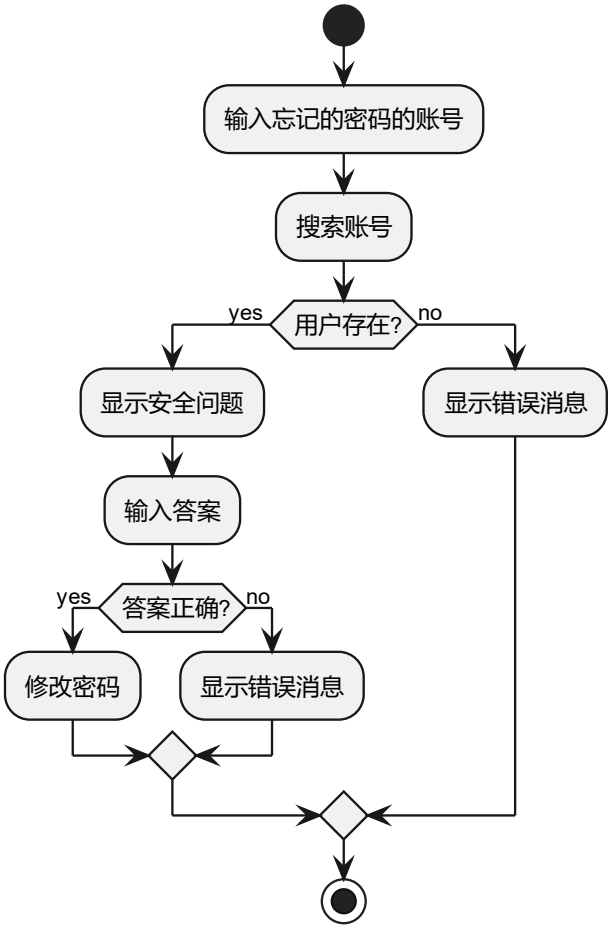

图 4-21 忘记密码流程图

如图 4-21 所示，用户输入忘记密码的账号可以通过以下步骤修改：

(1) 程序搜索账号，检查用户是否存在。

(2) 如果用户存在，程序显示安全问题，并要求用户输入答案。

(3) 如果用户输入的答案与存储的安全问题答案匹配，程序执行密码修改操作。

(4) 如果用户输入的答案不正确，程序显示错误消息，并密码修改失败。

(5) 如果用户不存在，程序显示错误消息。

五、系统实现

1. 模块中涉及的主要文件

文件名：save.bat

（1）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;
```

```
typedef struct
{
   double history;
   const char *Ptype;
} Record;
```

```
typedef struct
{
   int answer;
   char question[max];
} Verify;
```

代码对应的表格如下：

(2) 表 5 1 Record

字段名	类型	说明
history	double	历史记录
Ptype	const char *	P 类型

(3) 表 5 2 Verify

字段名	类型	说明

字段名	类型	说明
answer	int	答案
question	char[]	问题

(4) 表 5 3 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
limitCost	double	限制成本
verify	Verify	验证

2. 主要函数

(1)Code

函数声明

```
void Code(char* code)
```

函数功能说明

该函数用于生成一个随机的四位验证码，验证码由数字和字母组成。

主要代码段如下 ：

```
srand(time(NULL));
for (int i = 0; i < 4; i++) {
    int r = rand() % 36;
    if (r < 10) {
        code[i] = '0' + r;
    } else {
        code[i] = 'A' + r - 10;
    }
}
code[4] = '\0';
```

通过时间种子和随机数生成方法实现了生成随机四位验证码的功能。

使用 srand(time(NULL))将当前时间作为随机数生成器的种子，确保每次生成的随机数序列都不同。循环执行四次生成四位验证码。最后，在验证码的末尾添加字符串结束符\0，使其成为一个有效的 C 字符串。

(2)printReceipt

函数声明

```
void printReceipt(double amount, int judge)
```

函数功能说明 ** **

该函数用于打印交易凭据，包括交易时间、交易类型和交易金额。

```
time_t currentTime;
time(&currentTime);
struct tm* localTime = localtime(&currentTime);
char timeString[50];
strftime(timeString, sizeof(timeString), "%Y-%m-%d %H:%M:%S", localTime);
const char *TransType=NULL;
//……
printf("------------------------------\n");
printf("       ATM 交易凭据       \n");
printf("------------------------------\n");
printf("时间: %s\n", timeString);
printf("交易类型: %s\n", TransType);
printf("交易金额: %.2f\n", amount);
printf("------------------------------\n");
```

主要代码段如下 ：

通过 strftime 函数将时间格式化为字符串，将当前时间转换为本地时间，并使用条件语句判断交易类型。根据不同的交易类型，将相应的字符串赋值给 TransType。然后使用 printf 函数打印交易凭据的各项信息，包括分隔线、标题、时间、交易类型和交易金额。最后再打印一条分隔线，完成凭据的打印输出。

(3)lastRecord

函数声明

```
void lastRecord(User* user)
```

函数功能说明 ** **
该函数用于保留用户最近的 20 条交易数据，并删除旧的交易数据。函数接受一个指向用户结构体的指针作为参数。

主要代码段如下 ：

```
if (user->NUM == max)
{
  // 删除第一个元素
  for (int i = 0; i < max-1; i++)
  {
    user->record[i] = user->record[i + 1];
  }
  user->NUM--;//本质是一种伪删除
}
```

该代码段首先检查用户的交易数据数量是否达到最大值（20 条）。如果数量达到最大值，就会执行删除操作。删除操作是通过将第一个元素后面的所有元素向前移动一位来实现的，即将索引大于等于 1 的元素复制到前一个索引的位置上。最后，将交易数据数量减 1，以表示删除了一个旧的交易数据。

通过循环移位的方法实现了只保留最近的 20 条交易数据。当交易数据数量达到最大值时，即需要删除最旧的一条数据。该方法通过循环将每个元素后面的元素向前移动一位，以实现删除第一个元素的效果。这种实现方法简单高效，并且不需要额外的内存空间。

(4)secret

函数声明

```
void secret(char* s)
```

```
else
   {
      s[i] = c;
      i++;
      printf("*");
   }
}
printf("\n");
```

函数功能说明 *** *此函数用于加密用户输入的字符串，通过将用户输入的字符显示为星号()的形式，实现对输入的保密。*

主要代码段如下 ：

```
int i = 0;
while (1)
{
   char c = getch();
   if (c == '\r')
   {
      s[i] = '\0';
      break;
   }

  else if (c == '\b')
   {
      if (i > 0)
      {
         i--;
         printf("\b \b");
      }
   }
```

通过以下方法实现了加密的功能：

a) 使用 getch() 函数读取用户输入的字符，而不是常规的 scanf() 函数，从而在屏幕上不显示用户输入的字符。

b) 利用回车键 \r 作为输入结束的标志，当用户按下回车键时，将字符串末尾设置为 \0，表示字符串的结束。

c) 当用户按下退格键 \b 时，检查是否已经输入了字符，如果是，则将索引 i 减一，并在屏幕上擦除前一个字符的显示。

d) 对于其他非特殊字符，将其存储在字符串中，递增索引 i，并在屏幕上显示星号 * 代替实际字符的显示。

通过以上方法，实现了对用户输入字符串的加密，保护了用户输入的机密信息。

(5)AddSecret

函数声明

```
void AddSecret(char* s)
```

函数功能说明

该函数用于对密码进行加密。它接受一个字符指针 s，表示待加密的密码字符串。

主要代码段如下 ：

```
int head=0;
int tail=strlen(s)+1;
while(head!=tail)
{
   printf("%c",s[head]);
   head++;
   s[tail]=s[head];
   tail++;
   head++;
}
```

通过指定首尾索引来实现字符串的循环加密。该算法通过不断出队和将元素加到队尾的操作，对输入的密码字符串进行加密。首先，初始化 head 和 tail 分别为首个元素和字符串的后一个元素的索引。然后，使用一个循环来遍历字符串的元素。在每次循环中，

首先打印并出队第一个元素，然后将出队的元素加到队尾，最后再次出队。通过这样的操作，可以实现对密码字符串的加密。

(5)StrongPasswd

函数声明：

bool StrongPasswd(char *s)

函数功能说明 ：

该函数用于判断给定字符串是否为强密码。强密码的定义是：包含至少一个数字、一个小写字母和一个大写字母。

主要代码段如下 ：

```
int book[4] = {0};
for (int i = 0; s[i] != 0; i++) {
    if (isdigit(s[i]))
        book[1] = 1;
    if (islower(s[i]))
        book[2] = 1;
    if (isupper(s[i]))
        book[3] = 1;
}
if ((book[1] == 1) && (book[2] == 1) && (book[3] == 1))
    return true;
else
    return false;
```

通过以下方法实现了判断是否为强密码：

a) 使用一个长度为 4 的整型数组 book，用于记录数字、小写字母和大写字母是否出现过。

b) 遍历给定字符串 s，通过 isdigit()、islower()和 isupper()函数判断字符的类型，若是数字、小写字母或大写字母，则将相应的 book 数组元素置为 1。

c) 最后，检查 book 数组的元素是否都为 1，如果是，则说明给定字符串包含至少一个数字、一个小写字母和一个大写字母，返回 true 表示是强密码；否则返回 false 表示不是强密码。

该方法通过遍历字符串一次，记录字符类型的出现情况，并通过检查记录的结果进行判断，从而判断给定字符串是否为强密码。

(6)printSlowly

函数声明

```
void printSlowly(const char* str);
```

函数功能说明

该函数用于逐字打印字符串，并在每个字符之间增加延迟，以增加输出的真实感。

```
for (int i = 0; str[i] != 0; i++)
{
    printf("%c", str[i]);
    Sleep(100);
}
```

主要代码段如下 ：

在这段代码中，使用 for 循环逐个遍历字符串 str 中的字符。通过 printf 函数将每个字符打印出来。然后使用 Sleep 函数使程序暂停 100 毫秒，以实现字符之间的延迟输出。

通过延迟输出字符的方式，实现了逐字打印字符串并增加真实感的效果。在每个字符输出后都暂停一段时间，可以模拟人类逐字阅读的速度，使输出更加逼真。该方法适用于需要在控制台或终端界面逐个显示字符的场景。

(7)clearLine

函数声明

```
void clearLine();
```

函数功能说明

该函数用于清空当前行，在可视化界面中使用。它通过将光标移动到行首并清除行内容来实现。

主要代码段如下

```
printf("\r\033[K");
fflush(stdout);
```

通过方法、技巧实现了

该函数通过以下方法和技巧实现了清空当前行的功能：

a) 使用转义字符序列 \r 将光标移动到行首。

b) 使用转义字符序列 \033[K 清除行内容。

c) 调用 fflush(stdout) 刷新标准输出缓冲区，确保立即清空当前行的内容。

这种方法可以在可视化界面中实现清空当前行的效果，使得下一行的输出可以覆盖当前行的内容，从而达到清除当前行的目的。

(8)searchResult

函数声明

```
int searchResult(char* s);
```

函数功能说明

该函数用于在账号数组中查询指定账号的位置。

主要代码段如下 ：

```
for (int i = 0; i < count; i++)
{
   if (strcmp(users[i].account, s) == 0)
   {
      return i;
   }
}
return -1;
```

通过遍历账号数组，将每个账号与目标账号进行比较，若相等则返回该账号在数组中的索引位置。如果遍历完整个数组仍未找到匹配的账号，则返回-1 表示未找到。

该函数使用了字符串比较函数 strcmp 来比较账号字符串。如果两个字符串相等，strcmp 函数返回 0，进而判断账号是否匹配。

(9)Logup（与组员合作完成）

函数声明：void logup(int* pp)

函数功能说明 ：该函数用于注册新账号，并将注册信息保存到文件中。

主要代码段如下 ：

```
while(1)
{
   printf("请输入注册密码:\n");
   secret(users[*pp].passwd);

   if(StrongPasswd(users[*pp].passwd)==false)
   {
      printf("请输入含有数字，大写字母，小写字母的强密码\n");
      continue;
   }
}
```

这段代码用来判断是否为强密码

```
 Code(code);
printf("请输入验证码验证【%s】:\n",code);
char ccode[5];
scanf("%s",ccode);

if(strcmp(code,ccode)!=0)
{
    printf("验证码错误,请重试!\n");

   continue;
}
else
   printf("验证成功\n");
users[*pp].balance=0.0;
users[*pp].Freeze=false;
users[*pp].NUM=0;
users[*pp].Submit=false;
users[*pp].limitCost=20000;
(*pp)++;
saveFile(users, *pp);
break;
```

通过以上代码实现了以下功能：

a） 检查注册账号数量是否超过最大限制。

b） 接收用户输入的注册账号并检查是否已被注册。

c） 要求用户输入符合要求的强密码。

d） 生成一个随机验证码并要求用户输入进行验证。

e） 将注册信息保存到文件中。

（10）Login（与组员合作）

函数声明:

```
void login()
```

函数功能说明 ：

该函数用于用户登录功能。用户需要输入账号和密码，系统会进行验证。如果账号存在且密码正确，则登录成功，否则根据情况进行相应的处理。

主要代码段如下 ：

```
if (Fridge(users[i].Freeze) == true)       {
 printf("账号已冻结，请联系管理员!\n");
 printf("是否联系管理员(y/n):\n");
 scanf(" %c", &a);

 if ((a == 'y') || (a == 'Y'))
 {
  printf("已成功联系管理员\n");
  submit(i);
 }

printf("请输入密码: \n");
secret(passwd);
char code[5];
    Code(code);
                printf("请输入验证码验证【%s】:\n",code);
                char ccode[5];
    scanf("%s",ccode);
printf("密码重试次数超过限制，账号已冻结，请联系管理员!\n");
users[i].Freeze=true;
saveFile(users, count);
printf("是否联系管理员(y/n):\n");

scanf(" %c", &a);

if ((a == 'y') || (a == 'Y'))
{
 printf("已成功联系管理员\n");
 submit(i);
}
```

通过循环遍历账号列表，逐个对比输入的账号和密码进行验证。通过调用 Fridge()函数判断账号是否被冻结，如果是则提示用户联系管理员。如果账号存在且密码正确，则要求用户输入验证码，通过调用 Code()函数生成验证码，并与用户输入的验证码进行比较。如果验证通过，则登录成功，将指针 pr 指向该用户。如果密码错误次数超过限制，则冻结账号并提示用户联系管理员。如果账号不存在，则提示用户重试。

（11）deposit,withdraw,transfer（完成部分内容）

函数声明

```
void deposit,withdraw,transfer()
```

函数功能说明

该函数用于实现存款功能。它接受一个指向账户结构体的指针 pr，一个指向用户结构体的指针 users，以及表示用户数量的整数 count。

主要代码段如下 ：

```
lastRecord(pr);
saveFile(users, count);
printf("xx 成功！ \n");
printf("是否需要打印凭据:(y/n):\n");
char a;
scanf(" %c", &a);
if ((a == 'y') || (a == 'Y')) {
    printReceipt(amount, 1);
} else if ((a == 'n') || (a == 'N')) {
    return;
} else {
    printf("无效输入\n");
}
```

通过以下方法、技巧实现了存款功能：

a) lastRecord(pr)：调用了一个函数 lastRecord，该函数用于记录最后一次操作的时间和日期。

b) saveFile(users, count)：调用了一个函数 saveFile，该函数用于将用户数据保存到文件中。

c) printf("存款成功！\n")：打印提示消息，通知用户存款成功。

d) 用户输入是否需要打印凭据，通过使用条件语句进行判断：

（12）timetable

函数声明

```
void timetable(int a)
```

函数功能说明

该函数用于实现倒计时功能，并在命令行界面显示倒计时的剩余时间。

```
while(a>=0)
{
   system("cls"); // 清屏
   printf("账号已锁定，还剩%d 秒", a);
   Sleep(1000); // 暂停 1 秒
   a = a - 1;
}
```

主要代码段如下 ：

这段代码使用了一个 while 循环，当倒计时时间 a 大于等于 0 时，会执行循环体内的操作。在每次循环开始时，使用 system("cls")清屏，然后通过 printf 函数在命令行界面显示当前倒计时的剩余时间。接着使用 Sleep(1000)函数暂停 1 秒，实现每秒更新一次倒计时。最后，将倒计时时间 a 减 1，以进行下一次循环。

通过 system("cls")函数清屏和 printf 函数显示倒计时时间，结合 Sleep(1000)函数实现了倒计时功能。这种实现方法简单直观，每秒更新一次倒计时时间并在命令行界面显示。

（13）Delete

函数声明

```
void Delete(char* s1);
```

函数功能说明

该函数用于删除一个字符串在数组中的匹配项。

主要代码段如下 ：

```
int result = searchResult(s1);
if (result == -1) {
    printf("未找到相关信息!无法删除!\n");
    return;
} else {
    for (int i = result; i < count; i++) {
        users[i] = users[i + 1];
    }
    count--;
    printf("删除成功!\n");
}
saveFile(users, count);
```

通过方法、该函数通过以下步骤实现了删除功能：

a） 调用 searchResult() 函数查找字符串在数组中的索引。

b） 如果找不到匹配项（即返回值为 -1），则打印错误信息并返回。

c） 如果找到匹配项，从该索引开始，将后面的元素向前移动一位，覆盖匹配项。

d） 数组的元素数量减一。

e） 打印删除成功的提示信息。

f） 调用 saveFile() 函数保存更新后的数组到文件中。

（14）printRecord

函数声明

```
void printRecord(User* s);
```

函数功能说明，该函数用于打印记录。接受一个指向 User 结构体的指针参数，该结构体包含一个记录数组和记录数量。

主要代码段如下

```
for (int i = 0; i < s->NUM; i++)
{
    int count=0;
    printf("%.2lf\t",s->record[i]);
    count++;
    if(count%5==0)
    {
        putchar('\n');
    }
}
putchar('\n');
```

通过技巧实现了

该函数通过遍历记录数组并使用 printf 函数打印每个记录。在每个记录之后，使用计数器 count 进行计数，每 5 个记录换行打印。最后，在所有记录打印完毕后，打印一个换行符。

（15）logout

函数声明：

```
void logout();
```

函数功能说明 ：

该函数用于实现用户注销账号功能，包括确认注销、验证密码和删除账户等步骤。

主要代码段如下 ：

```
printf("注销无法撤回，请慎重考虑!!!\n");
printf("是否注销?(Y/N):\n");
char a;
scanf(" %c", &a);

if ((a == 'y') || (a == 'Y'))
{
    printf("请输入密码:");
    char Old[MAX];
    secret(Old);
    if (strcmp(Old, pr->passwd) != 0)
    {
        printf("密码不正确\n");
        return;
    }
    Delete(pr->account);
}
return;
```

通过以下方法实现了注销账号功能：

用户在函数运行时首先需要确认是否注销账号，输入字符'Y'或'y'表示确认注销，输入字符'N'或'n'表示取消注销。

a) 首先，函数会提示用户输入密码，并通过调用 secret 函数将用户输入的密码进行隐藏处理。

b) 然后，通过比较用户输入的密码与 pr->passwd 的值，判断密码是否正确。如果不正确，将输出提示信息并结束函数。

c） 如果密码正确，则调用 Delete 函数删除当前用户的账户。

（16）limit

函数声明

```
void limit();
```

函数功能说明

该函数用于设置账号的限额，即对指定账号的消费金额进行限制。

主要代码段如下 ：

```
char s1[20];
printf("请输入需要限额的账号:\n");
scanf("%s",s1);
if (searchResult(s1)==-1)
{
   printf("未找到相关信息!\n");
   getchar();
}
else
{
   int i=searchResult(s1);
   double limitCost2;
   printf("请输入额度:\n");
   scanf("%lf",&limitCost2);
   users[i].limitCost=limitCost2;
   //保存信息
   saveFile(users,count);
```

通过以下方法实现了对账号限额的设置：

首先，从用户输入中获取需要限额的账号，并将其存储在字符数组 s1 中。接着，通过调用 searchResult(s1) 函数来查找账号在用户数组中的索引位置。如果返回值为 -1，则表示未找到相关信息，输出提示信息 "未找到相关信息!"。否则，表示找到了对应的账号，继续执行下面的代码。在找到相关信息的情况下，再次调用 searchResult(s1) 函数，将返回的索引值赋给变量 i。用户被要求输入限额金额，将其存储在 double 类型变量 limitCost2 中。将 limitCost2 的值赋给对应账号在用户数组中的 limitCost 字段，实现限额设置。最后，调用 saveFile(users, count) 函数保存更新后的用户信息。

（17）menu2

函数声明

```
void menu2();
```

函数功能说明

该函数是一个管理员系统的菜单，用于管理账户信息和执行不同的操作。函数包含以下功能：

(5) 显示管理员系统欢迎信息和进入系统提示。

(6) 设置密钥，并进行密钥验证。

(7) 当密钥验证成功后，进入管理员系统菜单。

(8) 管理员系统菜单显示不同的选项，并根据用户的选择执行相应的操作。

主要代码段如下 ：

查看冻结账号：

```
int i;
for (i = 0; i < count; i++)
{
 if (users[i].Freeze)
 {
   printf("编号为%d,账号为%s 的用户冻结,是否希望解除冻结(是为 1,否为 0):%d\n",   i,
users[i].account,issubmit(i));
   FriNum++;
 }

}
```

菜单栏：

```
void menu2()
{
   // ...
   while (1)
   {
      // ...
      while(1)
      {
         // ...
         if(ch=='a')
            break;
      }
            switch (num+1)
      {
         case 1:
            printAllUsers();
            getchar();printf("按回车键继续");
            break;
         case 2:
            printf("请输入要删除的账号:\n");
            scanf("%s",s1);
            Delete(s1);
            //.....
            break;
         case 3:
            printf("请输入想要修改的账号:\n");

            scanf("%s",s1);
            // ...
         case 4:
            printf("请输入查询的账户:\n");
            scanf("%s",s1);
            // ...
         case 5:
            return;
           break;
         case 6:
            printf("请输入查找的账号:\n");
            gets(s1);
            // ...
         case 7:
            //.....
         case 8:
            printAll();
            break;
```

```
            case 9:
                limit();
                break;
            default:
                //.....
                break;
        }
        system("pause");    }
}
```

通过方法、技巧实现了管理员系统菜单的交互和功能选择。在主循环中，用户通过按下不同的键来选择菜单项，并根据选择执行相应的操作。程序使用了嵌套的循环来实现一个可移动的系统菜单，用户可以使用 "h" 键向左移动选项，"l" 键向右移动选项，"a" 键确认选择。

（18）menu1

函数声明： void menu()

函数功能说明 ： 该函数实现了一个模拟银行 ATM 系统的菜单功能，用户可以通过按键选择不同的操作，包括注册、登录、查余额、存款、取款、管理等功能。

主要代码段如下 ：

可视化操作

```
char ch=getch();
        if(ch=='l')
        {
          if(num>12)
            ;
          else
          {
          num++;
          clearLine();
          for (int i = 0; i < num; i++)
            printf("\t");
          printf("%s",s);
          }
        }
        if(ch=='h')
        {
          if(num==0);
          else
          {
          num--;
```

菜单栏

```
          switch (num+1)
      {
        case 1:
          logup(&count);
          break;
        case 2:
          login();
          break;

        case 3:
          if(pr==NULL)
            printf("请先登录");
          else
            showBalance();
          break;
```

```
case 4:
        if(pr==NULL)
            printf("请先登录");
        else
            deposit();
        break;
    case 5:
        if(pr==NULL)
            printf("请先登录");
        else
            withdraw();
        break;
    case 6:
        system("cls");
        menu2();
        break;
    case 7:
        printf("谢谢使用！\n");
        exit(0);
        break;
    case 8:
        if(pr==NULL)
            printf("请先登录");
        else
            transfer();
        break;
    case 9:
        if(pr==NULL)
            printf("请先登录");
        else changePassword();
        break;
```

```
case 10:
            if(pr==NULL)
               printf("请先登录");
            else logout();
            break;
 case 11:
            if(pr==NULL)
               printf("请先登录");
            else email(pr);
            break;
default:
            printf("请移动到正确位置");
            break;
```

通过在控制台打印相应的信息和使用特定的按键操作方式，实现了一个具有菜单功能的银行 ATM 系统。用户可以通过向左或向右按键移动菜单选项，按确认键进行选择不同的操作。其中，各个选项对应的操作包括注册、登录、查余额、存款、取款、管理等。具体实现方法为：

a） 首先，在菜单开始处进行一些初始化的显示，包括打印欢迎信息和加载完成提示，并改变控制台的颜色。

b） 然后，进入一个循环，在循环内部通过按回车键打开菜单，并在屏幕上显示菜单选项。

c） 接下来，使用'h'和'l'按键进行左右移动，同时更新菜单的显示位置。

d） 当用户按下确认键('a')后，根据当前选择的菜单选项，执行相应的操作。例如，选择注册时调用 logup()函数进行用户注册，选择登录时调用 login()函数进行用户登录，依此类推。

(19)loadFile(User*pr)

函数声明： int loadFile(User* pr)

函数功能说明 ：该函数实现了一个模拟银行 ATM 系统的菜单功能，用户可以通过按键选择不同的操作，包括注册、登录、查余额、存款、取款、管理等功能。

主要代码段如下

```
int loadFile(User* pr)
{
    FILE* file = fopen("save.bat", "rb");
    if (file == NULL)
    {
        printf("目前无用户加入\n");
        return 0;
    }
    int number;
    fread(&number, sizeof(int), 1, file);
    fread(pr, sizeof(User), number, file);
    fclose(file);
    return number;
}
```

a) 打开文件：使用 `fopen` 函数打开名为 "save.bat" 的二进制文件，模式为 "rb"（读取二进制文件）。如果文件打开失败（返回值为 `NULL`），则输出一条提示信息 "目前无用户加入"，并返回 0

b) 读取用户数量：使用 `fread` 函数从文件中读取一个 `int` 类型的数据，将其存储到变量 `number` 中。该数据表示文件中包含的用户数量。

c) 读取用户数据：使用 `fread` 函数从文件中读取 `number` 个 `User` 结构体大小的数据，并将其存储到以 `pr` 为首地址的内存空间中。这样就将文件中的用户数据加载到了 `pr` 所指向的 `User` 结构体数组中。

d) 关闭文件：使用 `fclose` 函数关闭已打开的文件。

e) 返回用户数量：返回变量 `number` 的值，表示成功加载的用户数量。

(20)forgotPassword()

函数声明： void forgotPassword()

函数功能说明 ：该函数实现了一个模拟银行 ATM 系统的菜单功能，用户可以通过按键选择不同的操作，包括注册、登录、查余额、存款、取款、管理等功能。

主要代码段如下

```
void forgotPassword()
{
    printf("输入忘记的密码的账号:\n");
    char forgotAccount[20];
    scanf("%s",forgotAccount);
    int i=searchResult(forgotAccount);
    if(i==-1)
    {
        printf("没有找到此用户");
        return;
    }
    printf("问题:%s\n", users[i].verify.question);
    int answer;
    printf("请输入答案:\n");
    scanf("%d", &answer);
    if (answer == users[i].verify.answer)
    {
        changePassword2(i);
        return;
    }
    else
    {
        printf("答案错误，密码修改失败。\n");
        return;
    }
}
```

a) 首先，代码通过 printf 打印提示信息，要求用户输入忘记密码的账号。

b) 然后，声明一个名为 forgotAccount 的字符数组变量，用于存储用户输入的账号信息。

c) 使用 scanf 函数从用户输入中读取账号信息，并将其存储在 forgotAccount 变量中。

d) 调用 searchResult 函数，将 forgotAccount 作为参数传递进去，以查找账号在用户数组中的索引。

e) 如果 searchResult 函数返回值为 -1，则表示没有找到对应的用户，此时代码会打印提示信息 "没有找到此用户" 并结束函数的执行。

f) 如果 searchResult 函数返回的索引值不为 -1，则表示找到了对应的用户。代码会打印该用户的安全问题 users[i].verify.question。

g) 声明一个名为 answer 的整数变量，用于存储用户输入的答案。

h) 使用 printf 打印提示信息，要求用户输入答案。

i) 使用 scanf 函数从用户输入中读取答案，并将其存储在 answer 变量中。

j) 如果用户输入的答案与 users[i].verify.answer 相等，代码会调用 changePassword2 函数来修改用户的密码，并结束函数的执行。

k) 如果用户输入的答案与 users[i].verify.answer 不相等，代码会打印提示信息 "答案错误，密码修改失败。"，并结束函数的执行。

六. 课程设计总结

1、问题及解决方法

在 ATM 系统的设计与实现过程中，我们成功完成了整体规划、界面设计、用户注册与登录、密码加密、存款、取款、转账、改密、余额查询、交易记录查询、管理员系统、用户信息管理、文件操作、延迟显示、删除操作、打印凭据操作、注销、冻结、限额、交易记录等功能。同时，我们也遇到了一些问题：

（1）缓冲区溢出导致 scanf 输入异常

问题描述：使用 scanf 函数接收字符输入时，发生了缓冲区溢出，导致输入异常。

问题分析：scanf 函数在接收字符时，会将换行符或空格等字符留在缓冲区中，下次调用 scanf 时会直接从缓冲区读取这些字符，导致输入异常。

解决方案：将 scanf 的格式字符串修改为" %c"，在%c 前加一个空格，这样可以消耗掉缓冲区中的空白字符，确保下次读取的是正确的输入。

（2）静态成员初始化问题

问题描述：在注册过程中，静态成员的初始化出现问题。

问题分析：静态成员在声明时会被初始化为零，但在注册时需要为其赋予相应的值。

解决方案：将静态成员的初始化放在注册过程中，即在注册函数内为静态成员进行赋值操作。

（3）加密函数输入字符串后异常

问题描述：在加密函数中，输入字符串后出现异常情况。

问题分析：在加密函数中，可能忘记在加密后的字符串末尾添加'\0'作为字符串结束的标志。

解决方案：在加密函数的最后添加'\0'，确保加密后的字符串以'\0'结尾。

（4）密码加密输入后无法用 backspace 撤销

问题描述：在密码输入时，使用 backspace 键无法撤销已输入的字符。

问题分析：在密码输入时，未对 backspace 键进行相应的处理。

解决方案：在密码输入的过程中，增加相应的条件语句，当检测到输入为 backspace 时，可以删除已输入的字符。

（5）冻结账户无法正确匹配到二进制文件

问题描述：冻结账户功能无法正确匹配到相应的二进制文件。

问题分析：在冻结账户功能中，可能存在逻辑错误或者结构体定义不完整的问题。

解决方案：检查冻结账户功能的逻辑，确保正确匹配到相应的二进制文件。如果需要，对结构体进行相应的修改和完善。

（6）锁定密码后再次等待时间结束后未进行任何操作又死循环锁定。

问题分析：在解锁后未添加判断，导致死循环。

解决方案：在解锁后添加判断，避免死循环。

（7）可视化菜单的操作无法对应。

问题分析：switch 语句的条件错误。

解决方案：将 switch 语句的条件改为 num+1。

（8）可视化菜单的操作没有限制导致异常。
问题分析：h,l 移动时未设置限制。

解决方案：设置 h,l 移动时的限制。

（9）程序在运行后出现崩溃。
问题分析：结构体数组的宏值过大，导致内存不足。

解决方案：减少结构体数组的宏值，将 MAX 设置为 50。

（10）是否告诉管理员冻结函数异常。
问题分析：逻辑错误，需要改正。

解决方案：修改相关逻辑错误，删掉繁琐的逻辑，在结构体中添加相关信息。

（11）交易记录查询出错。
问题分析：缓冲区异常。

解决方案：处理缓冲区异常问题。

（12）登录时程序异常

问题分析：结构体变量的指针未赋值造成了野指针。

解决方案：将指针 pr 指向空。

2、模块总体实现情况

我的任务分配为**整体规划、界面设计、用户注册与登录、密码加密、存款、取款、转账、改密、余额查询、交易记录查询、管理员系统、用户信息管理、文件保存操作、延迟显示**

后续在次基础上添加了**删除操作、打印凭据操作、注销、冻结、限额、交易记录。**

经过小组的努力，ATM 系统各模块已基本实现，符合课程要求和课题功能。在实际使用过程中，系统运行稳定，功能完善。我们对系统的实现和模块功能进行了充分的测试，确保其符合预期。

3、存在问题及改进思路

尽管我们的 ATM 系统已经实现了许多功能，但仍存在一些问题和不足。

(1) 变量命名可以更清晰一些,比如 pr 可以命名为 currentUser 等。

(2) 可以使用更严谨的类型,比如 MAX 可以定义为 const int MAX = 50 等。

(3) 可以添加更多的错误处理,比如输入非法字符时的处理等。

(4) 可以使用面向对象的编程思想,定义 User、ATM 等类,提高代码复用性和扩展性。

(5) 可以添加日志记录功能,记录每次交易信息等。

(6) 可以添加安全机制,比如交易次数限制、安全验证码等。

(7) 可以分文件编写,比如将用户管理、交易功能等分开,提高程序可维护性。

(8) 可以添加计算机网络知识,实现 ATM 联网、离线交易等功能。

(9) 代码使用了全局变量，尽量不要去使用全局变量，目前全局变量的坏处并不明显，但是未来在多人设计工程大项目时，会造成不可逆的后果。

为了改进系统。我们可以采取以下策略：

(1) 继续学习和提高 C 语言编程能力，以便更好地解决遇到的问题。

(2) 优化代码结构，提高代码可读性和可维护性。

(3) 加强团队沟通与协作，确保每个成员在项目中发挥最大作用。

(4) 在项目初期制定更详细的计划与分工，确保整个开发过程有序进行。

4、课程总结

(1) 达到了课程的学习目的。通过这个项目,熟练掌握了 C 语言的基本语法和数据结构,学习了如何设计和实现一个较为复杂的程序。同时也加深了对面向对象编程和软件工程的理解。

(2) 学到了许多编程技巧和方法。比如如何设计用户接口,如何实现类和对象,如何进行模块化设计,如何进行错误处理和异常捕捉,如何进行代码重构和重构等。这些技巧和方法对我以后设计和实现程序都非常有帮助。

(3) 对 C 语言程序设计有了更深入的认识。实现这个项目让我了解到 C 语言除了基本语法之外,还需要掌握数据结构、算法、软件工程等更高级的技能。只有熟练掌握各种工具和技能,才能设计出健壮高效的程序。

(4) 对团队合作也有了新的认识。这个项目的大小适合几个人共同完成,让我体会到了团队合作的重要性。通过分工协作,充分发挥每个人的专长,可以更快更好地完成项目。同时,团队中也需要选定一个负责人,统筹进度和质量,妥善解决意见分歧,这也是我的一大收获。

(5) 进度管理和计划很重要。为了按时完成这个较大的项目,我们制定了详细的进度计划和时间表,并定期跟进进度,避免出现瓶颈,这些管理经验也可以运用到以后项目中。

通过这次课程设计，我们对 C 程序设计有了更深入的了解，学到了许多有关文件操作、结构体、函数等方面的知识。同时，我们也认识到了进度管理、计划与准备、团队合

作等方面的重要性。这次课程设计使我们更加明确了自己的不足之处，也为我们提供了一个宝贵的实践机会。在今后的学习和工作中，我们将努力克服不足，不断提高自己的能力，更好地应对各种挑战。

七. 建议

作为学习 C 程序设计和参与 ATM 模拟系统大作业的学生，我想分享一些关于教学、实验和实践性环节的改进建议，以期提高教学效果和学习体验。

1. 引入更多实践性编程项目：除了 ATM 模拟系统大作业，将更多的实践性编程项目纳入课程中。这样可以让学生在实际项目中应用所学的知识，加深对 C 程序设计的理解。例如，可以设计一些简单的小游戏或应用程序，让学生在课程期间逐步完成，从而培养他们的编程技能和解决问题的能力。

2. 设计更具挑战性的大作业：尽管 ATM 模拟系统是一个不错的项目，但在大作业的设计上可以增加一些更具挑战性的要求。这可以激发学生的学习兴趣和动力，促使他们更深入地研究和应用 C 程序设计的概念。例如，可以要求学生实现一些高级功能，如多线程处理、数据加密或网络通信等，以提升他们的技术水平和解决实际问题的能力。

3. 提供更多的实验机会：在课程中增加更多的实验环节，让学生通过实际操作来巩固所学的知识。这可以包括编写和调试简单的程序、运行调试工具和观察程序执行过程等。通过这些实验，学生可以更好地理解 C 程序设计的工作原理，并培养他们的调试能力和错误处理能力。

4. 提供定期的代码评审和反馈：在大作业过程中，定期进行代码评审，并给予详细的反馈。这有助于学生发现和纠正自己代码中的问题，同时学习他人的编程技巧和优秀实践。通过定期的评审和反馈，学生可以不断改进自己的编程风格和质量意识。

5. 引入实际案例和应用场景：将 C 程序设计与实际案例和应用场景结合起来，可以增加学习的实用性和趣味性。例如，可以引入一些真实的金融系统案例，让学生设计和实现相关功能。这样可以帮助学生将所学的知识与实际问题相联系，加深对 C 程序设计的理解和应用能力。

6. 采用一些合作办公平台，可以使得小组成员在一个服务器中开发代码，增加工作的连贯性。

7. 多多进行小组交流，定期开展小组讨论会，每个小组可以分享自己的代码成果并且提出宝贵意见。

8. 开设类似图形库 easyx 之类的讲解课程，使学生在标准 C 语言外使用其他图形库搭建一个美化的可视化窗口程序。