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：22030527

姓　　名：王语桐

指导教师：曹中心

2023 年 6 月

一、课程认识

1. 目的：

C 语言程序设计课程的目的是为了帮助我们掌握编程基础知识并培养解决问题的能力。通过学习 C 语言，我可以了解计算机程序的执行原理，学习如何编写具体的程序来解决实际问题。这将为我未来的学习和职业发展打下坚实的基础。

2. 性质：

C 语言程序设计课程的性质既有实践性又有抽象性。我们需要通过编写具体的 C 程序来实践所学知识，这锻炼了我的动手能力和实际操作经验。同时，C 语言作为一种较底层的编程语言，需要我们具备一定的抽象思维能力，理解和运用变量、函数、指针等概念。

3. 任务：

在 C 语言程序设计课程中，我需要完成一系列任务，这些任务帮助我逐步提高编程能力和解决问题的能力。这些任务包括：

（1） 学习 C 语言的基本语法和数据类型，掌握变量、运算符、控制结构等基本概念。

（2）理解函数的定义和调用，学会编写函数，并掌握函数参数传递的方法。

（3）学习数组、字符串等数据结构，运用它们解决实际问题。

（4）掌握指针的概念和使用方法，了解内存管理和动态内存分配的原理。

（5）学习文件操作，包括读写文件、文件指针的使用等。

（6）进行简单的算法设计和程序调试，培养问题分析和解决的能力。

通过完成这些任务，我能够逐渐提高自己的编程能力，并能够独立设计和实现简单的 C 语言程序。这不仅让我对计算机的工作原理有更深入的理解，也增强了我的自信心和解决实际问题的能力。

二、课题选择

课题：ATM 管理系统

课题背景：

随着金融业务的快速发展和用户需求的增加，ATM 作为一种重要的现金管理和交易工具，已经成为银行和金融机构不可或缺的一部分，因此开发一个高效、安全、可靠的 ATM 管理系统变得尤为重要。本课题旨在设计一个简单的 C 语言管理系统，实现用户的注册、登录、存款、取款、转账等基本功能，以满足日常金融操作的需求。

意义：

本课题的意义在于通过编写一个简单的 ATM 管理系统，可以让我们在实践中巩固和应用所学的 C 语言知识，培养我们的编程能力和问题解决能力。同时，该管理系统具有一定的实用性，可以模拟真实的金融操作，让用户能够通过系统进行账户管理和交易操作。

实用性：

1. 基本功能实现：该管理系统实现了用户的注册、登录、存款、取款、转账等基本功能，用户可以通过系统进行账户管理和交易操作，提供了方便和便捷的金融服务。

2. 用户信息管理：系统通过结构体存储用户的个人信息，包括账号、密码和余额，并使用文件操作实现用户信息的保存和加载，确保用户数据的持久性和安全性。

3. 安全保障措施：系统具备一定的安全保障措施，如密码输入错误次数限制、账号冻结、联系管理员解冻等，保护用户账户的安全。

4. 交易记录记录：系统记录用户的交易记录，并只保留最近的 20 条记录，方便用户查阅自己的交易历史。

5. 界面美化：系统包含一些辅助函数和装饰图案函数，通过延迟输出和清空行的技术，实现了一些提示信息的显示效果，提升了用户界面的美观度。

通过本课题的设计与实现，我们可以提升自己的 C 语言编程能力，加深对 C 语言的理解，同时在模拟金融操作场景中培养问题解决能力。

三、系统总体分析与设计

系统总体框架图如图 1 所示。

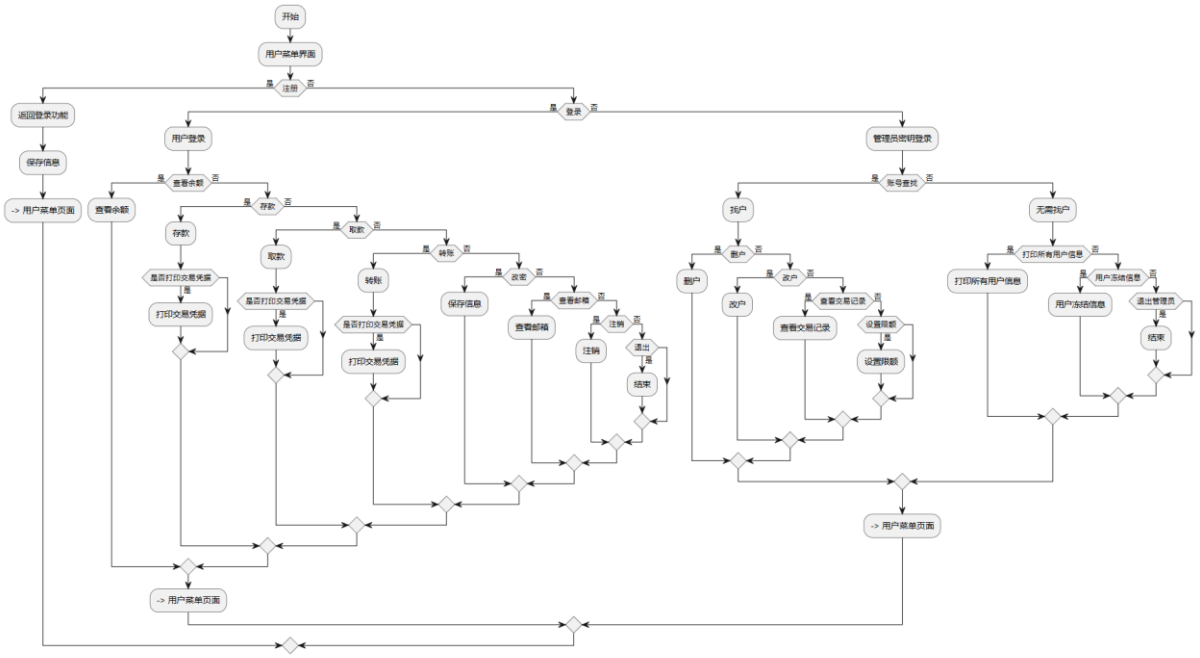

图 1 系统总体结构图

简要功能说明：

1. 注册模块：用户可以输入个人信息，包括账号、密码等进行注册。

2. 登录模块：已注册用户可以通过输入正确的账号和密码进行登录。

3. 存款模块：登录后的用户可以进行存款操作，将指定金额存入账户余额中。

4. 取款模块：登录后的用户可以进行取款操作，从账户余额中扣除指定金额。

5. 转账模块：登录后的用户可以进行转账操作，将指定金额从自己的账户转至指定账户。

6. 余额查询模块：登录后的用户可以查询自己的账户余额。

7. 交易记录模块：系统会记录用户的交易记录，并显示最近的 20 条记录。

8. 密码冻结解冻模块：当用户登录时连续输入错误密码达到一定次数时，账号会被冻结，需要联系管理员进行解冻。

小组具体担当情况为：

杨熙承：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

王语桐：用户注册与登录、存款、取款、转账、余额查询、交易记录查询

李硕：改密、余额查询、交易记录查询、用户信息管理、文件操作

四、模块详细设计

1. 用户注册模块

该模块主要实现用户注册功能，确保用户可以通过有效的账号和密码进行登录并完成注册流程。

功能说明：

（1）用户输入账号、密码、余额等个人信息。

（2）系统验证账号是否已存在，如果已存在，则要求用户重新输入。

（3）系统验证密码强度，确保密码符合要求。

（4）系统生成随机验证码，并显示给用户。

（5）用户输入验证码，系统进行验证，如果验证码匹配成功，则进行下一步。

（6）系统将用户信息保存到文件中，用于后续登录和其他操作的验证。

（7）显示注册成功信息给用户。

（8）结束注册模块。

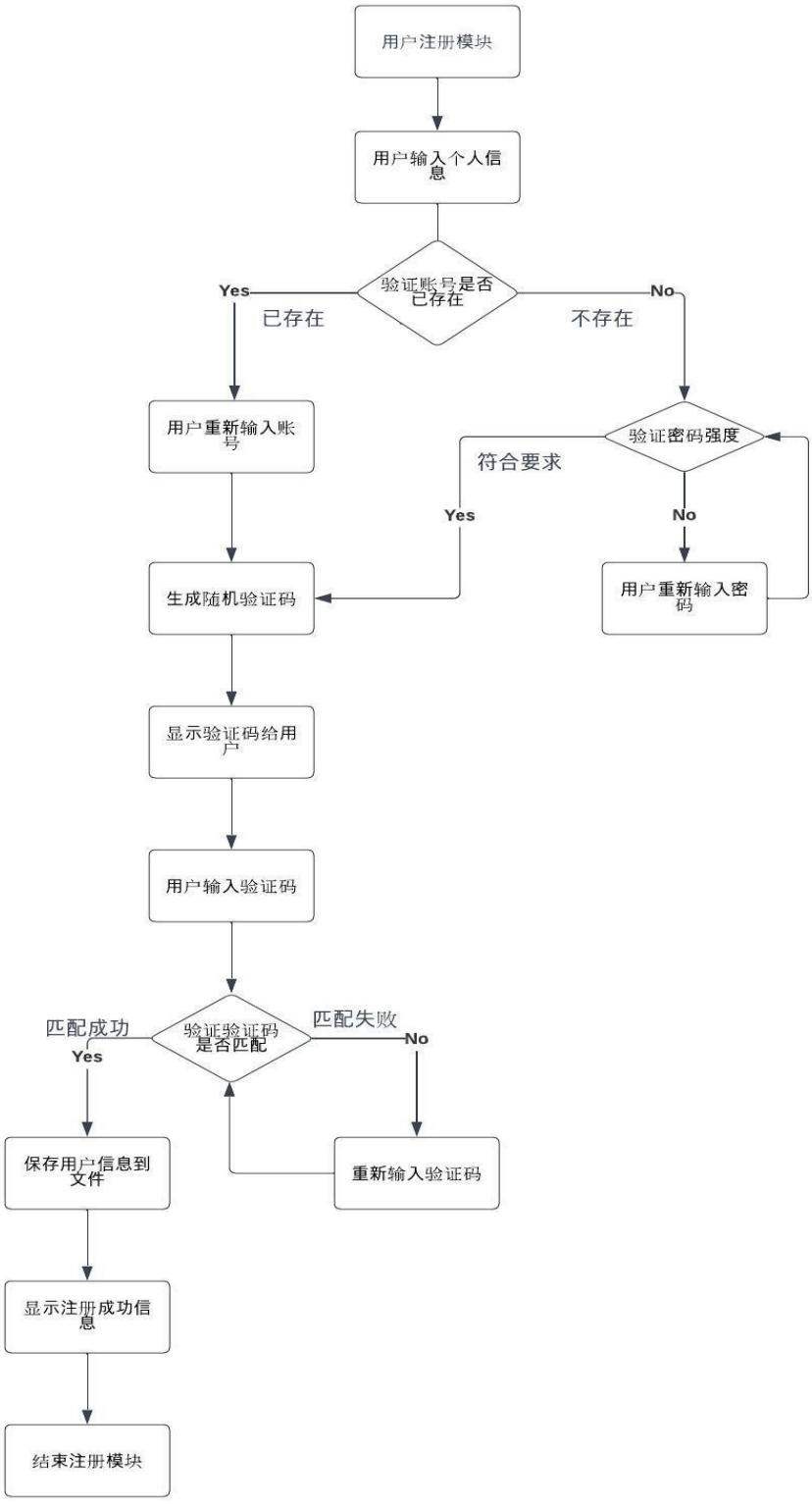
用户注册模块
用户输入个人信息
验证账号是否已存在
Yes
已存在
No
不存在
用户重新输入账号
验证密码强度
符合要求
Yes
No
生成随机验证码
用户重新输入密码
显示验证码给用户
用户输入验证码
匹配成功
验证验证码是否匹配
匹配失败
Yes
No
保存用户信息到文件
重新输入验证码
显示注册成功信息
结束注册模块

图 2 注册流程图

2、用户登录模块

该模块负责用户登录功能，让用户可以使用已注册的账号和密码进行登录。

功能说明：

（1）用户输入账号和密码。

（2）系统验证用户输入的账号和密码是否与注册时的信息匹配。

（3）如果匹配成功，继续下一步；否则，返回登录界面或显示错误信息。

（4）系统判断账号是否被冻结，如果冻结，则提供联系管理员解冻的选项。

（5）显示登录成功信息给用户。

（6）结束用户登录模块。

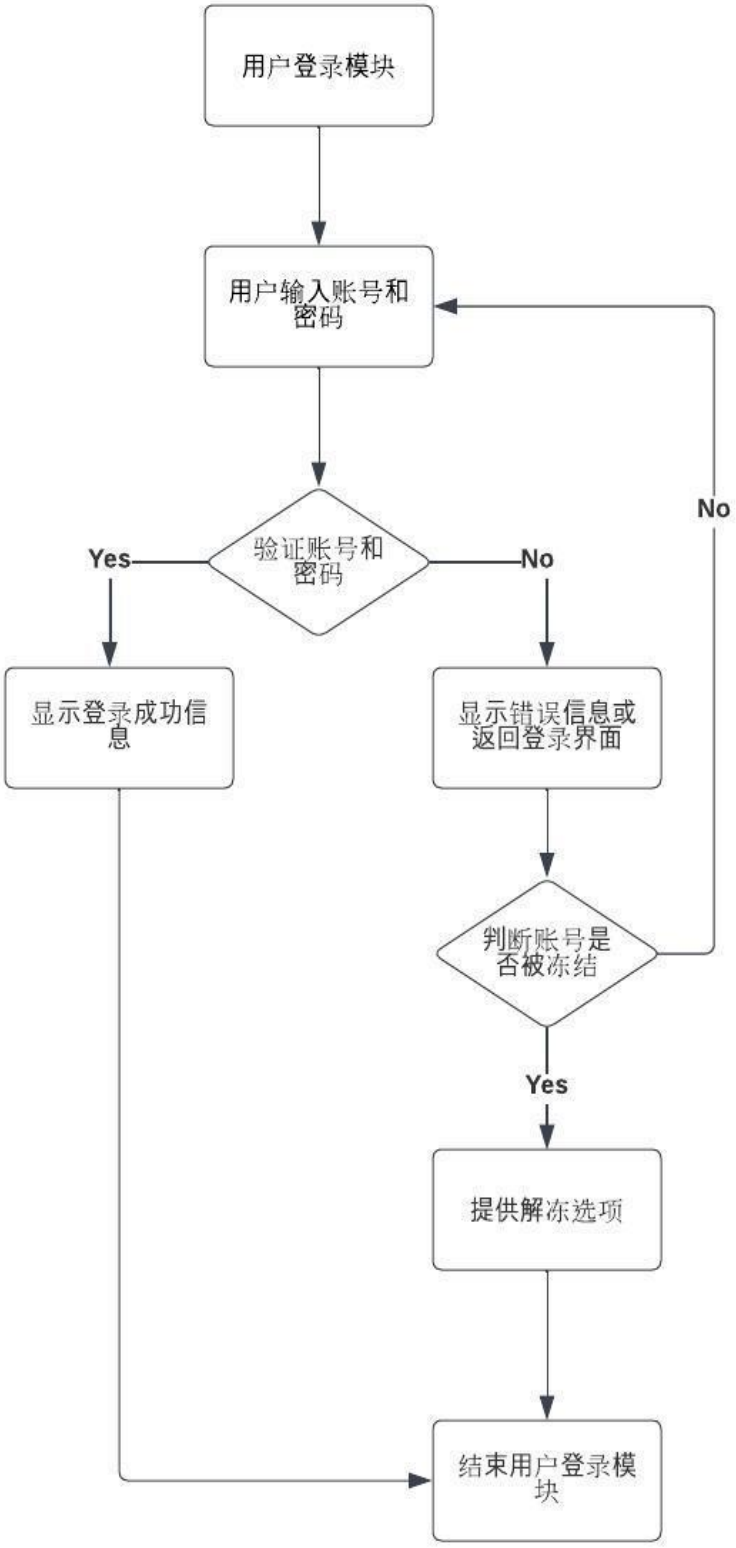
用户登录模块
用户输入账号和密码
验证账号和密码
Yes
No
No
显示登录成功信息
显示错误信息或返回登录界面
判断账号是否被冻结
Yes
提供解冻选项
结束用户登录模块

图 3 登录流程图

3、存款模块

该模块负责实现用户向其账户存款的功能。

功能说明：

（1）获取用户账号和存款金额：模块开始时，从用户输入中获取账号和存款金额的值。

（2）存款操作：根据用户输入的账号，查找对应的用户信息，并将存款金额加到用户的余额上。

（3）更新用户余额：将更新后的余额保存回用户信息中。

（4）记录存款交易记录：在系统的交易记录中添加一条存款记录，包括交易类型、账号、交易时间和金额。

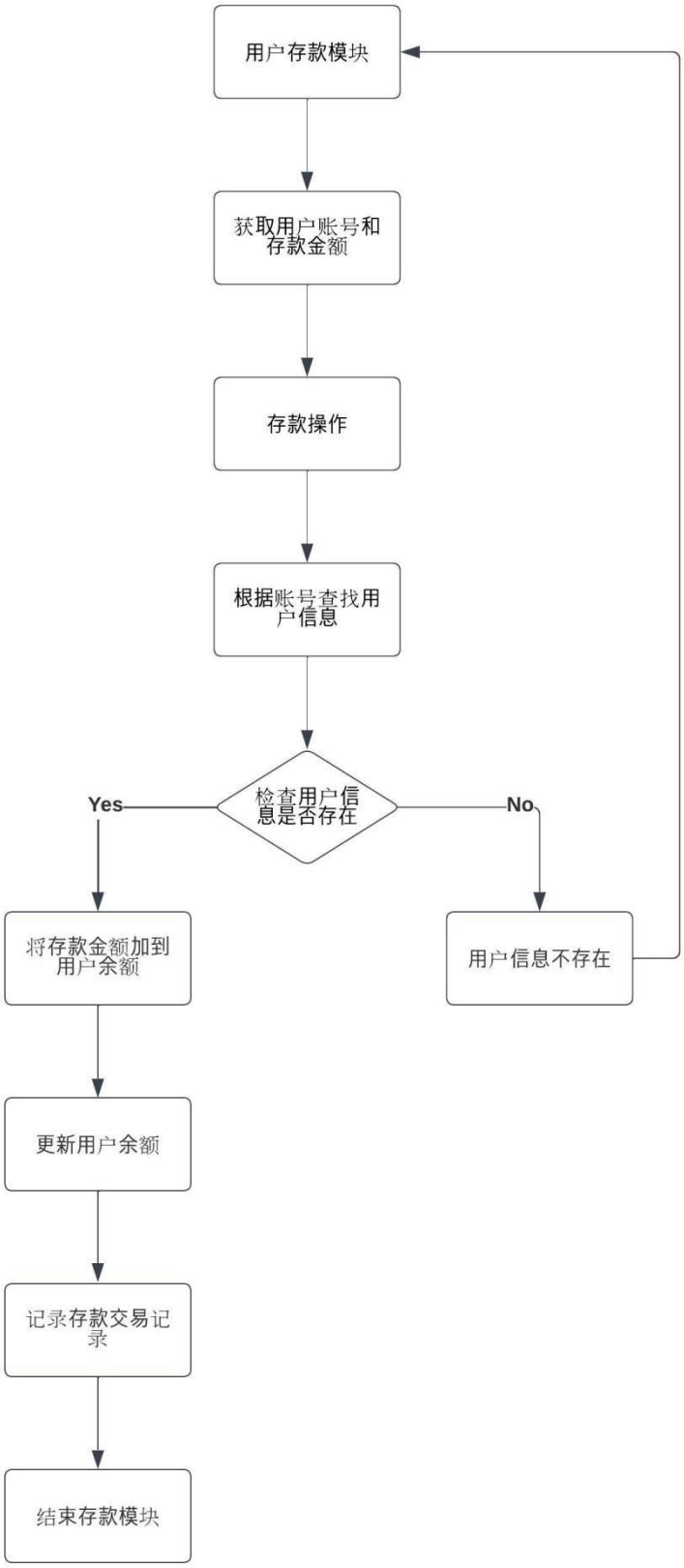
用户存款模块
获取用户账号和存款金额
存款操作
根据账号查找用户信息
检查用户信息是否存在
Yes
No
将存款金额加到用户余额
用户信息不存在
更新用户余额
记录存款交易记录
结束存款模块

图 4 存款流程图

4.取款模块

功能说明：

（1）用户输入账号和取款金额：用户通过界面输入账号和欲取款的金额，提供必要的输入信息。

（2）验证账号和取款金额：模块首先验证用户输入的账号是否存在于系统中，并检查该账号是否被冻结。如果账号不存在或已被冻结，则操作失败。接着，验证取款金额是否小于等于用户的余额。如果取款金额超过用户的余额，则操作失败。

（3）更新用户余额：如果账号和金额验证通过，系统将从用户的余额中减去取款金额，更新用户的余额信息。

（4）记录交易记录：在系统的交易记录中添加一条取款记录，包括账号、交易时间和金额。这样可以追踪用户的取款操作并保留历史记录。

（5）显示操作结果：模块向用户显示取款操作的结果。如果操作成功，系统会显示成功的消息，并提供更新后的余额信息。如果操作失败，系统会显示相应的错误消息，指示失败的原因。

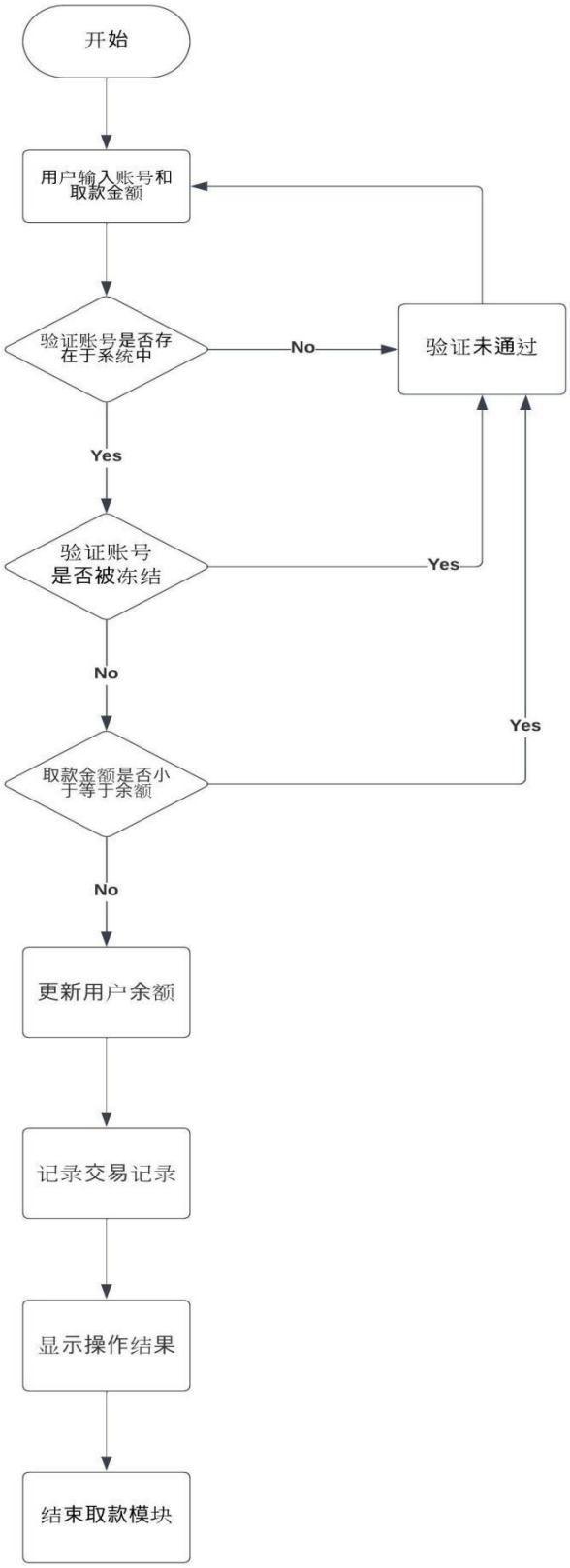
开始
用户输入账号和取款金额
验证账号是否存在于系统中
No
验证未通过
Yes
验证账号是否被冻结
Yes
No
取款金额是否小于等于余额
Yes
No
更新用户余额
记录交易记录
显示操作结果
结束取款模块

图 5 取款流程图

5. 转账模块

功能说明：

（1）用户输入转出账号和转入账号以及转账金额：用户通过界面输入转出账号、转入账号和转账金额，提供必要的输入信息。

（2）验证账号和金额：模块首先验证转出账号和转入账号是否存在于系统中，并检查转出账号是否被冻结。如果账号不存在或已被冻结，则操作失败。接着，验证转账金额是否小于等于转出账号的余额。如果转账金额超过转出账号的余额，则操作失败。

（3）更新账户余额：如果账号和金额验证通过，系统将从转出账号的余额中减去转账金额，并将转账金额加到转入账号的余额中，完成转账操作。

（4）记录交易记录：在系统的交易记录中添加一条转账记录，包括转出账号、转入账号、交易时间和金额。这样可以追踪用户的转账操作并保留历史记录。

（5）显示操作结果：模块向用户显示转账操作的结果。如果操作成功，系统会显示成功的消息，并提供更新后的余额信息。如果操作失败，系统会显示相应的错误消息，指示失败的原因。

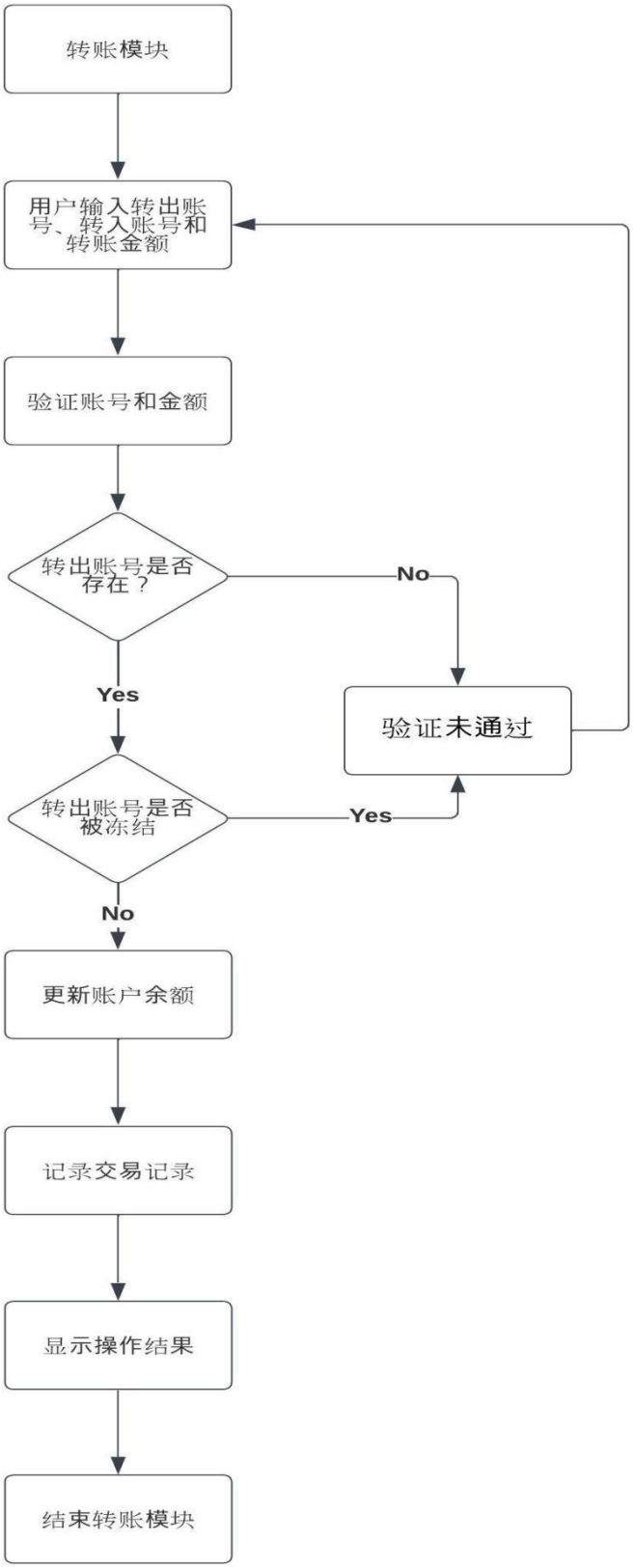
转账模块
用户输入转出账号、转入账号和转账金额
验证账号和金额
转出账号是否存在？
No
Yes
验证未通过
转出账号是否被冻结
Yes
No
更新账户余额
记录交易记录
显示操作结果
结束转账模块

图 6 转账流程图

6. 余额查询模块

功能说明：

（1）用户输入账号：用户通过界面输入要查询余额的账号。

（2）验证账号：模块首先验证输入的账号是否存在于系统中。如果账号不存在，则查询失败。

（3）显示余额：如果账号验证通过，系统将显示该账号的当前余额信息。

（4）提供操作选项：系统可以提供用户选择是否继续进行其他操作，如存款、取款、转账等，或退出余额查询模块返回主菜单。

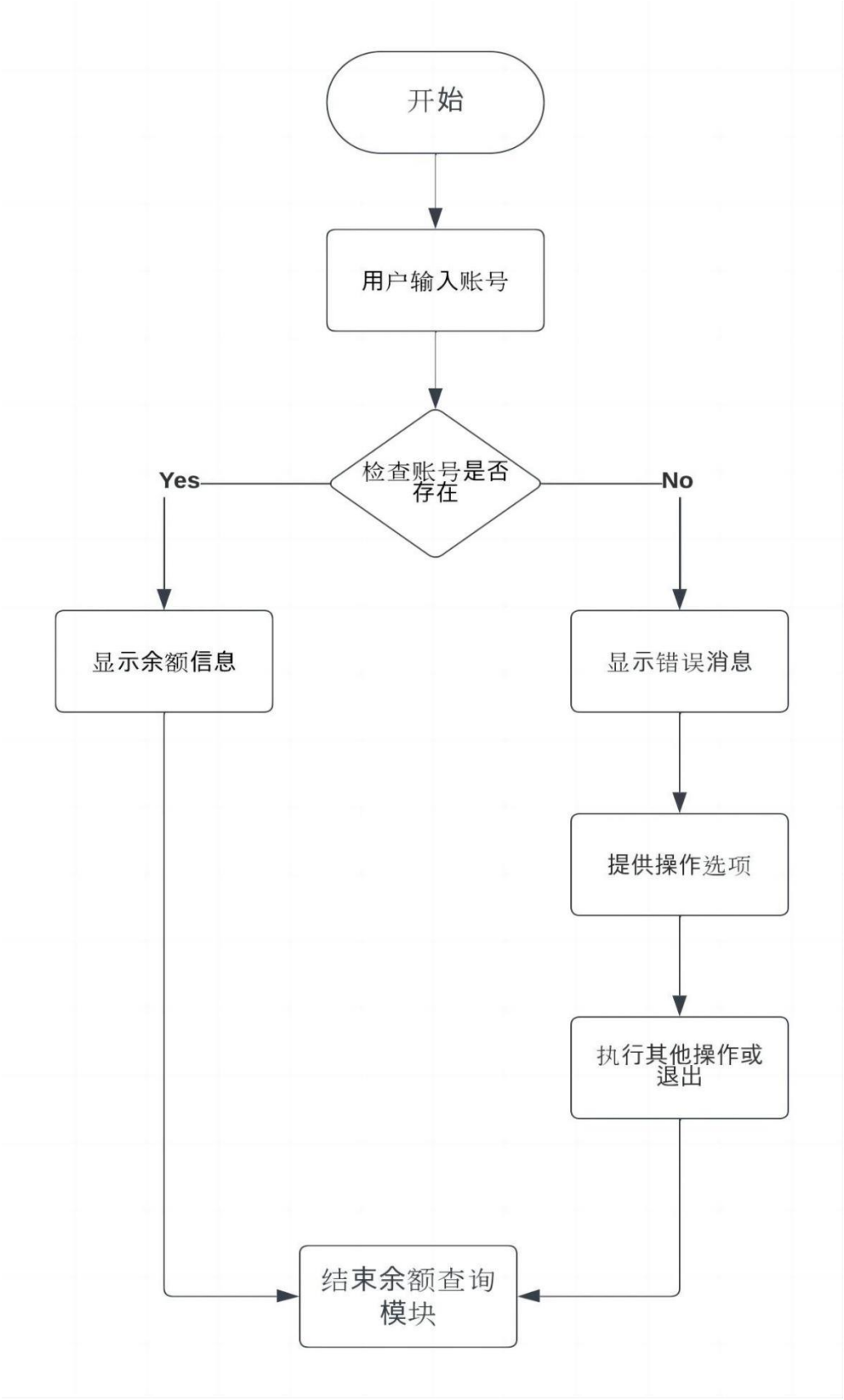
开始
用户输入账号
检查账号是否存在
Yes
No
显示余额信息
显示错误消息
提供操作选项
执行其他操作或退出
结束余额查询模块

图 7 余额查询流程图

7. 交易记录查询模块

功能说明：

（1）用户输入账号：用户通过界面输入要查询交易记录的账号。

（2）验证账号：模块首先验证输入的账号是否存在于系统中。如果账号不存在，则查询失败。

（3）显示交易记录：如果账号验证通过，系统将显示该账号的最近的交易记录信息。通常，系统会限制显示最近的 20 条记录，以保持界面的简洁性。

（4）提供操作选项：系统可以提供用户选择是否继续进行其他操作，如查询其他账号的交易记录，返回主菜单等。

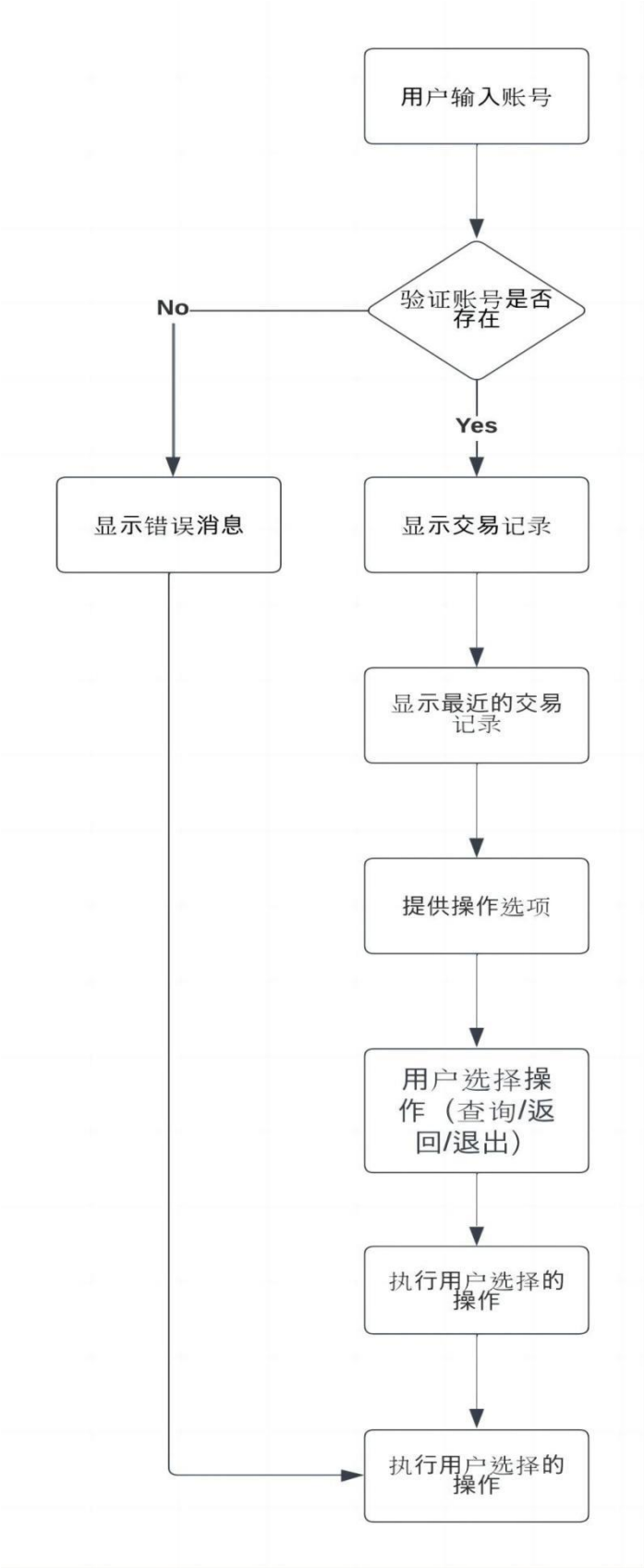
用户输入账号
验证账号是否存在
No
Yes
显示错误消息
显示交易记录
显示最近的交易记录
提供操作选项
用户选择操作（查询/返回/退出）
执行用户选择的操作
执行用户选择的操作

图 8 余额查询流程图

五、系统实现

1、模块中涉及的主要文件

文件名：save.bat

本文件主要存放银行用户的信息，对应的结构体为:

```
  typedef   struct {
char account[max];   // 账号
char passwd[max] ;   //密码
double balance;   // 余额
double record max ;   //记录，最多保存 20 条
int NUM;     //交易记录数量
bool Freeze;   //是否冻结状态
bool Submit;     //是否告诉管理员
double limitCost;   //限额
}User;
```

代码对应的表格如下：

表 2-1 银行用户信息文件

字段名	类型	备注
account	char[]	账号
passwd	char []	密码
balance	double	余额
record	double[]	记录，最多 20 条
NUM	int	交易记录数量
Freeze	bool	是否冻结状态
Submit	bool	是否告诉管理员
limitCost	double	限额

2、主要函数

（1）void logup()

该函数用于用户注册账号，并对密码强度进行检查，生成验证码进行验证。主要代码段如下：

```
if (count > MAX)
{
    printf("抱歉，目前系统无法注册新账号");
    return;
}
printf("请输入注册账号:\n");
scanf("%s", account);
strcpy(users[count].account, account);

if (searchResult(users[count].account) != -1)
{
    printf("该账号已经注册");
    return;
}
```

通过以下方法和技巧，上述代码段实现了账号注册前的检查和验证：

I. 检查注册数量限制：

通过比较`count`和`MAX`来检查已注册用户数量是否超过最大限制。

如果超过限制，即注册数量达到系统上限，函数返回并输出相应提示信息。

II. 输入和存储注册账号：

使用`scanf`函数要求用户输入注册账号，并将其存储在`account`数组中。

使用`strcpy`函数将账号从`account`数组复制到`users[count].account`数组中，实现账号的存储。

III. 检查账号是否已注册：

调用`searchResult`函数搜索已注册的账号数组`users`，以判断输入的账号是否已经在已注册的账号列表中存在。

如果`searchResult`返回值不为-1，表示账号已经注册过，函数返回并输出相应提示信息。

（2）void login()

该函数的主要作用是实现用户登录功能，包括输入账号和密码进行验证，处理账号冻结和密码错误等情况，并与管理员进行联系。

```
void login()
{
    char account[MAX];
    char passwd[MAX];
    char a;
    printf("请输入账号: \n");
    scanf("%s", account);

    for (int i = 0; i < count; i++)
    {
        if (strcmp(users[i].account, account) == 0)
          {
            if (Fridge(users[i].Freeze) == true)
            {
               ....
    }
              printf("请输入密码: \n");
            secret(passwd);
            char code[5];
```

```
        for (int j = 0; j < 5; j++)
        {
            if (strcmp(users[i].passwd, passwd) == 0)
               {    ...
                }
                    printf("登陆成功\n");
                       pr = users + i;
                         return;
                }
               else
                {
                 printf("密码错误，请重新输入(还可输入%d 次): \n",5-j);
                 secret(passwd);
                 }
        }
                 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("账号不存在，请重试\n");
        return;

}
```

通过以下方法和技巧，上述代码段实现了用户登录的功能：

I. 用户输入账号：使用`scanf`函数从用户输入中获取账号信息，并将其存储在`account`字符数组中。

II. 遍历用户列表：通过一个`for`循环，遍历用户列表中的每个用户，以查找与输入的账号匹配的用户。

III. 检查账号冻结：通过调用`Fridge`函数，判断当前用户的账号是否已被冻结。如果冻结，则向用户显示相应的提示信息，并询问是否联系管理员。

IV. 联系管理员：根据用户输入的选择，判断是否联系管理员。如果用户选择联系管理员，则调用`submit`函数，将冻结的账号信息传递给管理员。如果用户选择不联系管理员，则显示相应的提示信息。

V. 输入密码：通过`secret`函数，接收用户输入的密码，并将其存储在`passwd`字符数组中。

VI. 验证码验证：使用一个循环来生成和验证验证码。在每次循环中，调用`Code`函数生成一个随机验证码，并要求用户输入验证码进行验证。如果输入的验证码与生成的验证码不匹配，则显示错误提示，并继续生成和验证验证码，直到输入正确的验证码为止。

VII.密码验证：通过一个`for`循环，将用户输入的密码与当前用户的密码进行比较。如果匹配成功，则登录成功，并将指针`pr`指向当前用户。

VIII.密码错误处理：如果密码验证失败，循环将给出密码错误的提示，并允许用户重新输入密码。在每次循环中，用户最多可以尝试 5 次输入密码。

IX.账号冻结处理：如果密码验证失败次数超过了限制，将冻结当前账号。将账号的`Freeze`字段设置为`true`，并将用户信息保存到文件中。然后向用户显示相应的提示信息，并询问是否联系管理员。

X. 返回结果：在以上情况下，函数将提前返回相应的结果，结束登录过程。如果遍历用户列表后仍未找到匹配的账号，则向用户显示账号不存在的提示信息。

总体而言，该函数通过遍历用户列表、冻结账号、验证码验证等方法，实现了安全的用户登录功能。使用了循环、条件判断、函数调用等基本编程技巧，以处理不同的情况，并向用户提供相应的反馈和选择。

（3）void deposit()

该函数的主要功能是允许用户进行存款操作，用户可以输入存款金额，将金额累加

到账户余额中，并记录存款金额到用户的交易记录中，还可以选择是否打印存款凭据。主要代码段如下：

```
    if ((a == 'y') || (a == 'Y'))
    {
        printReceipt(amount,1);
    }
    else if ((a == 'n') || (a == 'N'))
    {
        return;
    }
    else
    {
        printf("无效输入\n");
    }
}
```

```
void deposit()
{
	double amount;

	printf("请输入存款金额: \n");
	scanf("%lf", &amount);
	if(amount >= pr->limitCost)
		printf("金额过大，无效存款\n");
	if (amount <= 0)
	{
		printf("请输入正确的存款金额\n");
		return;
	}
	pr->balance += amount;
	pr->record[pr->NUM]=amount;
	pr->NUM++;
	lastRecord(pr);
	saveFile(users, count);
	printf("存款成功！ \n");

	printf("是否需要打印凭据:(y/n):\n");
	char a;
	scanf(" %c",&a);
```

该函数实现了存款操作的功能，并使用了一些方法和技巧来实现特定的功能。下面是对该函数的实现方法和技巧的说明：

I. 输入存款金额：通过`scanf`函数，获取用户输入的存款金额，并将其存储在`amount`变量中。

II. 检查存款金额限制：使用条件判断，检查用户输入的存款金额是否超过了账户的限制金额`limitCost`。如果超过限制，则显示相应的提示信息，并终止存款操作。

III.检查存款金额有效性：再次使用条件判断，检查用户输入的存款金额是否小于等于零。如果是，则显示相应的提示信息，并终止存款操作。

IV. 更新账户余额：将存款金额累加到当前用户的账户余额`balance`中，使用`+=`运算符实现累加操作。

V. 记录存款金额：将存款金额记录到当前用户的交易记录数组`record`中，使用用户的`NUM`字段作为索引，将存款金额存储在相应的位置上。然后，将`NUM`字段自增，以便下一次记录使用。

VI. 检查是否达到交易记录上限：调用`lastRecord`函数，检查用户的交易记录数量是否达到上限（20 次）。如果达到上限，将执行相应的操作，根据具体实现可能包括清空记录、覆盖最旧记录等。

VII. 保存用户数据：调用`saveFile`函数，将更新后的用户信息保存到文件中。这可能涉及整体保存用户列表或只保存当前用户的信息，具体实现可能有所不同。

VIII. 显示存款成功信息：向用户显示存款成功的提示信息，表明存款操作已成功完成。

IX. 询问是否打印凭据：根据用户的选择，使用条件判断，判断是否需要打印存款凭据。根据用户的输入，调用`printReceipt`函数进行凭据打印操作。

X. 返回结果：根据用户的选择或输入，函数可能提前返回或终止操作，或者继续执行后续的代码。

总体而言，该函数通过输入金额、检查限制、更新余额、记录交易、保存数据等方法，实现了存款操作的功能。使用了条件判断、累加运算、数组操作等基本编程技巧，以处理不同的情况，并向用户提供相应的反馈和选择。

（4）void withdraw()

该函数的主要功能是允许用户进行取款操作，用户可以输入取款金额，从账户余额中扣除相应的金额，并记录取款金额到用户的交易记录中，还可以选择是否打印取款凭据。主要代码段如下：

```
void withdraw()
{
    double amount;
    printf("请输入取款金额: \n");
    scanf("%lf", &amount);

    if (amount <= 0)
    {
        printf("取款金额无效\n");
        return;
    }

    if(amount > pr->balance)
    {
        printf("取款金额无效或余额不足。\n");
        return;
    }
pr->balance -= amount;
    pr->record[pr->NUM]=-amount;
    pr->NUM++;
    lastRecord(pr);
saveFile(users, count);
    printf("取款成功！\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,2);
    }
    else if ((a == 'n') || (a == 'N'))
    {
        return;
    }
    else
    {
        printf("无效输入\n");
    }
}
```

该函数实现了取款操作的功能，并使用了一些方法和技巧来实现特定的功能。下面是对该函数的实现方法和技巧的说明：

I 输入取款金额：通过`scanf`函数，获取用户输入的取款金额，并将其存储在`amount`变量中。

II 检查取款金额有效性：使用条件判断，检查用户输入的取款金额是否小于等于零。如果是，则显示相应的提示信息，并终止取款操作。

III 检查余额是否足够：再次使用条件判断，检查用户输入的取款金额是否大于账户的余额`balance`。如果大于余额，则显示相应的提示信息，并终止取款操作。

IV 更新账户余额：从当前用户的账户余额中减去取款金额，使用`-=`运算符实现扣款操作。

V 记录取款金额：将取款金额记录到当前用户的交易记录数组`record`中，使用用户的`NUM`字段作为索引，并将取款金额存储在相应的位置上。然后，将`NUM`字段自增，以便下一次记录使用。

VI 检查是否达到交易记录上限：调用`lastRecord`函数，检查用户的交易记录数量是否达到上限（20 次）。如果达到上限，将执行相应的操作，可能包括清空记录、覆盖最旧记录等。

VII 保存用户数据：调用`saveFile`函数，将更新后的用户信息保存到文件中。具体实现可能有所不同，可能是整体保存用户列表或只保存当前用户的信息。

VIII 显示取款成功信息：向用户显示取款成功的提示信息，表明取款操作已成功完成。

IX 询问是否打印凭据：根据用户的选择，使用条件判断，判断是否需要打印取款凭据。根据用户的输入，调用`printReceipt`函数进行凭据打印操作。

X 返回结果：根据用户的选择或输入，函数可能提前返回或终止操作，或者继续执行后续的代码。

总体而言，该函数通过输入金额、检查有效性、更新余额、记录交易、保存数据等方法，实现了取款操作的功能。它使用了条件判断、扣款运算、数组操作等基本编程技巧，以处理不同的情况，并向用户提供相应的反馈和选择。

（5）void transfer()

该函数的主要作用是允许用户进行转账操作,用户可以输入目标账号和转账金额，从当前账户余额中扣除相应的金额，并将金额转入目标账号的余额中，同时记录转账操作到双方的交易记录中，并可选择是否打印转账凭据。主要代码段如下：

```
void transfer()
{
    char s[MAX];
    printf("请输入目标账号: \n");
    scanf("%s", s);
    int i = searchResult(s);
    if (i == -1)
    {
        printf("目标账号不存在\n");
        return;
    }
```

```
            double amount;
            printf("请输入转账金额: \n");
            scanf("%lf", &amount);
            if(amount >= pr->limitCost)
                printf("金额过大，无效转账\n");
            if (amount <= 0)
            {
                printf("请输入正确的转账金额\n");
                return;
            }
            if (amount > pr->balance)
            {
                printf("转账金额超过账户余额\n");
                return;
            }
            pr->balance -= amount;
            users[i].balance += amount;
            users[i].record[users[i].NUM]=amount;
            users[i].NUM++;
            lastRecord(users+i);
            pr->record[pr->NUM]=-amount;
            pr->NUM++;
            lastRecord(pr);
            saveFile(users, count);
            printf("转账成功！ \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,3);
            }
            else if ((a == 'n') || (a == 'N'))
            {
                return;
            }
            else
            {
                printf("无效输入\n");
            }
    }
```

该函数的实现方法与技巧如下：

I 用户输入目标账号，并通过`searchResult()`函数查找目标账号的索引位置。

II 如果目标账号不存在（索引为-1），则输出提示信息并返回。

III 用户输入转账金额，并进行有效性检查：如果金额超过当前账户的限制金额(`pr->limitCost`)，则输出提示信息并返回；如果金额小于等于零，则输出提示信息并返回。

IV 如果转账金额大于当前账户余额，则输出提示信息并返回。

V 执行转账操作：从当前账户余额中减去转账金额(`pr->balance -= amount`)，并将转账金额加到目标账号的余额中(`users[i].balance += amount`)。

VI 在目标账号的交易记录中添加转入记录(`users[i].record[users[i].NUM]=amount`)，并递增记录数量(`users[i].NUM++`)。

VII 在当前账号的交易记录中添加转出记录(`pr->record[pr->NUM]=-amount`)，并递增记录数量(`pr->NUM++`)。

VIII 检查交易记录是否达到上限（20 条）并进行相应处理，保持最近的 20 条记录。

IX 保存用户信息到文件中(`saveFile(users, count)`)。

X 输出转账成功的提示信息。

XI 用户选择是否打印转账凭据，并进行相应处理：如果选择打印，则调用`printReceipt()`函数打印凭据；如果选择不打印，则返回；如果输入无效，则输出提示信息。

通过上述方法和技巧，该函数实现了在用户账户之间进行转账的功能，包括输入目标账号、转账金额的验证和处理，账户余额的更新，交易记录的记录和保存，以及可选择是否打印转账凭据。

（6）void printRecord(User* ss)

该函数用来打印登录用户的信息，最多打印最近 50 条

```
void printRecord(User* ss)
{
    for (int i = 0; i < ss->NUM; i++)
    {
        printf("%.2lf\t",ss->record[i].history);
        printf("%s\t\n",ss->record[i].Ptype);
        printf("_____________");
        putchar('\n');
    }
    printf("目前余额:%.2lf",ss->balance);
    putchar('\n');
}
```

这段代码定义了一个名为 printRecord 的函数，它接受一个指向 User 类型的指针作为参数。函数的目的是打印用户的记录信息。函数通过一个循环遍历用户的记录数组，并使用 printf 函数打印每个记录的历史值和类型。具体来说，代码中的 ss->record[i].history 表示访问指针 ss 所指向的 User 结构体中的记录数组的第 i 个元素的 history 成员。类似地，ss->record[i].Ptype 用于访问记录数组的第 i 个元素的 Ptype 成员。在每个记录打印完成后，代码使用 printf 函数打印一条分隔线，并使用 putchar('\n')函数输出一个换行符。最后，函数打印用户的余额值，使用 printf 函数输出 ss->balance 的值，并使用 putchar('\n')函数输出一个换行符。

六、课程设计总结

1、问题及解决方法

在系统设计和代码实现过程中，我们遇到了一些问题并采取了相应的解决方法。其中，一些主要问题及解决方案如下：

(1) 问题：如何实现用户账号的冻结和解冻功能？

解决方案：我们在用户登录模块中添加了密码错误次数的统计功能。当密码错误次数超过一定阈值时，将用户账号标记为冻结状态。用户在登录界面可以选择联系管理员解冻账号，管理员通过验证后可以解冻账号。

(2) 问题：如何限制交易记录的数量并保留最近的 20 条记录？

解决方案：我们使用一个数组来存储用户的交易记录。每次进行交易时，将新的交易记录添加到数组的开头。当数组长度超过 20 时，删除最旧的一条记录，以保持最近的 20 条记录。这样可以限制交易记录的数量，并确保只保留最近的 20 条记录。

(3) 问题：如何确保用户输入的合法性和数据的完整性？

解决方案：我们在各个功能模块中增加了输入验证机制。通过验证用户的输入格式、范围和类型等，可以确保用户输入的合法性。例如，在存款模块中，我们验证存款金额是否为正数，以防止无效的输入。在转账模块中，我们验证转账目标账号的存在性，以确保交易的准确性。

(4) 问题：如何提高密码的安全性？

解决方案：我们实现了加密的 scanf 函数 secret，用于在命令行中输入密码时显示*号，增加输入的安全性。

2、模块总体实现情况

我们团队根据任务分工，各自承担了不同的模块实现。总体而言，我们成功实现了用户的注册、登录、存款、取款、转账等功能模块，并且使用结构体、文件操作、数组和指针等技术来管理用户信息和实现用户操作。系统具备较好的稳定性和可用性，能够满足基本的管理系统需求。

3、存在问题及改进思路

在课程设计过程中，我们也遇到了一些问题和不足之处。其中一些存在的问题及改进思路如下：

（1）问题：用户界面的交互性和友好性有待改进，提供更清晰的提示信息和操作指引。

改进思路：通过增加更详细的提示信息和用户指南，提升系统的易用性和用户体验。

（2）问题：代码的可扩展性和维护性有待提高，随着功能的增加，代码的复杂度可能会增加。

改进思路：采用模块化的设计和合理的代码结构，提高代码的可读性和可维护性，方便后续功能的扩展和维护。

4、课程总结

通过这次课程设计，我们深入学习了 C 语言编程和软件开发的基本原理和技术，并将所学知识应用到实际项目中。我们学会了团队合作、任务分工和协调沟通，提高了项目管理和进度控制的能力。在完成设计过程中，我们更加深入理解了 C 程序设计的重要性和灵活性，并且意识到了团队协作的重要性。

通过本次课程设计，我们不仅掌握了 C 语言编程的技术和方法，还培养了解决问

题、合作协调和自我管理的能力。同时，我们也认识到了项目开发过程中的一些挑战和改进空间，这对我们今后的学习和职业发展都具有积极的意义。

七、建议

1. 实际案例的引入：我觉得在教学中引入更多实际案例会很有帮助。通过实际案例，我可以更好地理解和应用所学的 C 程序设计知识。这样的学习方式可以增强我的兴趣，因为我可以看到这些知识在实际生活中的应用场景。

2. 渐进式的学习：我认为将 C 程序设计和大作业的学习内容分成适当的阶段是很重要的。这样我可以逐步学习和掌握各个概念和技能。从简单到复杂的学习过程可以帮助我建立起对 C 编程的自信心，不至于一下子感到太过困难和压力。

3. 实验和实践性任务的增加：我觉得增加更多的实验和实践性任务对于学习 C 程序设计非常重要。通过亲自动手编写 C 程序，我可以更深入地理解和应用所学的知识。同时，这也可以培养我解决问题和调试代码的能力，因为实践是最好的学习方式。

4. 小组合作项目的开展：我希望能有机会和其他同学一起合作完成一个 C 程序设计项目。通过小组合作，我可以与他人共同学习、交流和解决问题。这样的合作可以帮助我提高我的团队合作和沟通能力，而且通过与他人的互动，我可以学到不同的思维方式和解决问题的方法。

5. 及时的反馈和评估：我认为建立有效的反馈和评估机制非常重要。及时地得到关于我的 C 程序设计和大作业的指导和建议，可以帮助我了解自己的优点和不足，并进行及时的改进和提高。这样的反馈可以让我知道自己的进步，并鼓励我继续努力学习。

6. 实用工具和资源的提供：在学习 C 程序设计和完成大作业时，我希望能够得到必要的实用工具和资源支持，如集成开发环境（IDE）、调试工具和在线参考资料等。这些工具和资源可以帮助我更高效地编写和调试 C 程序，提高我的学习效率。

7. 鼓励创新和实践：我觉得在学习 C 程序设计和完成大作业时，鼓励我们展示创新思维和实践能力是很重要的。给予我们一定的自由度，让我们能够在实践中尝试新的想法和解决方案。这样的学习环境可以激发我的学习兴趣，同时培养我解决问题和创造力。

总而言之，我认为通过实际案例的引入、渐进式的学习、增加实验和实践性任务、开展小组合作项目、及时的反馈和评估、提供实用工具和资源以及鼓励创新和实践，可以让我在 C 程序设计和大作业中获得更好的教学效果，提高我的学习成果和实际能力。