

高级语言程序设计

课程设计报告

|  |  |  |
| --- | --- | --- |
| 课程设计成绩（总分20） | | 分数 |
| 程序 | 程序设计完整，功能齐全（8-10分） |  |
| 程序设计基本完整，功能基本完成（0-7分） |
| 报告 | 结构清晰完整，报告格式正确，语言通顺（8-10分） |  |
| 格式不正确，封面或字体有误，语言欠通顺（0-7分） |
|  | 总分 |  |

|  |  |
| --- | --- |
| 课程名称 | 高级语言程序设计 |
| 课程编号 | 08304135 |
| 报告题目 | 银行账户管理系统 |
| 序 号 | 51 |
| 姓 名 | 夏磊 |
| 学 号 | 21122926 |
| 所在学院 | 计算机工程与科学学院 |
| 专业名称 | 智能科学与技术 |

2021 年 11 月

目录

[1. 数据格式与功能描述 3](#_Toc87372465)

[1.1 数据文件格式 3](#_Toc87372466)

[1.2 功能描述 3](#_Toc87372467)

[1.3 要求 5](#_Toc87372468)

[2. 问题分析 6](#_Toc87372469)

[3. 总体设计 6](#_Toc87372470)

[3.1 功能模块设计 6](#_Toc87372471)

[3.2 系统界面设计 7](#_Toc87372472)

[3.3 数据结构设计 7](#_Toc87372473)

[3.4 函数设计 8](#_Toc87372474)

[4. 详细设计 9](#_Toc87372475)

[4.1 各个函数的调用关系图 9](#_Toc87372476)

[4.2 各个函数功能描述 10](#_Toc87372477)

[5. 功能测试 37](#_Toc87372478)

[5.1 初始化界面 37](#_Toc87372479)

[5.2 控制台打印所有账户 38](#_Toc87372480)

[5.3 查询账户信息 38](#_Toc87372481)

[5.4 开户 39](#_Toc87372482)

[5.5 销户 40](#_Toc87372483)

[5.6 修改密码 42](#_Toc87372484)

[5.7 存款 42](#_Toc87372485)

[5.8 取款 42](#_Toc87372486)

[5.9 转账 43](#_Toc87372487)

[5.10 排序 44](#_Toc87372488)

[5.11 计算利息 47](#_Toc87372489)

[5.12 退卡（更换银行） 47](#_Toc87372490)

[5.13 导出账户信息到文件 47](#_Toc87372491)

[5.14 错误信息 48](#_Toc87372492)

[6. 总结 48](#_Toc87372493)

银行账户管理系统

银行账户管理系统是金融机构进行用户管理的必备工具，它能够对海量的储户信息规范管理，使管理工作系统化、自动化、提高管理工作的效率。账户信息包括账号，开户人姓名、密码、金额、开户日期等。鉴于此，银行账户信息需要使用结构体存放。

本实验要求实现一个简单的账户管理系统，涉及函数、结构体、链表、文件等方面的知识，学习利用链表处理数据的方法，熟练掌握文件操作，构建综合程序设计的思路及框架，提高综合设计软件系统的能力。

1. 数据格式与功能描述
   1. 数据文件格式

每条银行账户信息包括：账号，开户人姓名、性别、密码、金额、开户日期，账户状态(活期/定期)。多条银行账户信息存储在一个txt文件中，本系统涉及多个这样的txt文件，分别表示不同的银行。单条账户信息示例格式如图1-1所示。

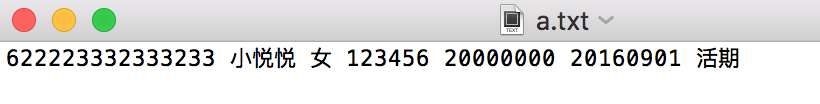


图1-1 单条银行账户信息

本实验中，要求自行创建5个txt文件，分别命名为a.txt, b.txt.. e.txt,分别表示五家银行的银行账户信息。各个银行具体的账户信息可以自行设定。

## 1.2 功能描述

设计一个银行账户管理系统，利用单链表来处理银行账户信息，要求实现如下系统功能。

1. 初始化菜单

程序开始运行后显示菜单，要求菜单能完全展示本系统的功能，尽力做到设计美观。

1. 选择待操作的文件

考虑到本系统有多个文件，需要提示用户输入来选择需要操作的银行信息文件。

(3) 查询账户信息

按账号、开户人姓名或开户日期查询银行账户信息。提示用户输入待查账户的账号、开户人姓名或开户日期，若找到该账户，则输出账户信息，若没有找到，则输出提示信息。

可输出全部账户信息，若当前银行没有账户，输出提示信息。

(4) 开户

通过控制台增加账户信息，提示用户输入账户的编号，如果编号存在，提示编号已经存在；如果不存在，则逐项录入银行账户的每一条信息。

(5) 销户

删除指定账户的信息。提示用户输入待销户账号，若找到该账户，提示输入，密码确认后，将该账户金额清零并删除账户信息；若密码错误或未找到，则输出提示信息。

(6) 修改账户密码

修改指定账户的密码。提示用户输入待修改账户的编号，若找到该账户，若密码确认正确，提示修改账户密码；若密码确认错误或者未找到，则输出提提示信息。

(7) 账户排序

按账号、开户人姓名、金额或开户日期进行排序。提示用户选择的排序类型，按照用户所选择的类型进行排序并输出结果。

(8) 存款

向指定账户存款。提示用户输入存款账户的账号和密码，若该账户存在且密码正确，提示输入存款金额，否则输出提示信息。

(9) 取款

从指定账户取款。提示用户输入取款账户的账号和密码，若该账户存在且密码正确，提示输入取款金额，若取款金额小于余额，输出取款成功并显示取款后的个人信息，若取款金额大于余额，输出提示余额不足，若账户不存在或密码错误，输出提示信息。

(10) 转账

提示用户输入转账方和收款方的所在银行()。若双方在同一家银行，则只需要判断转账方金额是否满足转账条件，如果不满足，输出提示信息，如果满足转账条件，输出转账后的账户信息。若双方不在同一银行，应按照规定收取1%的手续费且最高不超过50，而后判断支付手续费后，转账放是否满足转账要求，若满足，输出转账后的账户信息，若不满足，输出提示信息。

(11) 计算利息

账户的存款方式分为活期和定期两种，若为活期存款，年利率为0.35%，若为定期存款，第一年年利率为2.75%，第二年为3.35%，第三年之后每年为4%，不足一年的按照半年计算，用户控制台输入存款到期的时间(格式为20171031)，开户日期算做存款日期，由此计算某账户在此期间内获得的利息。

(12) 保存账户信息

建立一个数据文件，将账户信息写入到数据文件中进行保存。

(13) 退出系统

用户输入指定的内容，程序结束，退出系统。

1.3 要求

(1)源程序编写要求

根据系统功能描述，采用模块化程序设计方法进行程序设计，要求程序结构清晰。上述各个功能分别用函数实现，在主函数中通过调用这些函数，完成系统功能的要求。代码书写要规范，有简要的注释，给出函数说明。

(2)设计报告撰写要求

设计报告内容包括题目内容和要求、总体设计、详细设计、源代码、调试过程中的问题、总结等。

总体设计：对程序的整体设计思路进行描述，画出图书管理系统的总体功能模块图，说明系统使用的主要数据结构，列表给出需要用到的函数并描述其功能。

详细设计：分析实现各函数功能的算法，描述函数的功能。

调试过程中的问题：记录程序编写和调试过程中遇到的各种问题，以及解决这些问题的途径和方法。

总结：回顾整个综合程序设计的过程，对学习到的设计方法和思路进行总结，写出个人体会。

1. 问题分析

银行账户信息的数据为文本格式，由多个数据项组成。在程序设计中，可以用结构体来储存，由于系统功能涉及账户信息的插入与删除，使用链表结构储存信息更有利于插入和删除处理。因此，本实验将账户信息存储在单链表的结点中，由数据域和指针域构成。数据部分包括：账号、开户人姓名、性别、密码、金额、开户日期，账户状态(活期/定期)共七个成员；指针域存储其直接后继结点的地址。

账户信息的查找、销户、修改密码、开户、利息计算、转账、存取款都需要先查找，找到目标结点后再进行相应操作。对账户信息的排序，可采用选择排序、冒泡排序、插入排序等，针对单链表结构的特点，银行账户管理系统可采用选择排序的方法实现按不同字段的排序。存款时，找到目标结点后，需根据余额判断剩余金额是否支持取款，再完成后续操作。保存信息后，修改后的账户信息被写入数据文件。

1. 总体设计
   1. 功能模块设计

根据系统功能描述和问题分析，可将系统功能划分为若干模块，如图3-1所示。



图 3-1 系统功能模块图

* 1. 系统界面设计

进入银行账户管理系统后，首先要求用户选择账户所在银行，接着显示主菜单，菜单列出12个选项：展示所有账户、退卡、导出账户信息、查找账户、开户、销户、修改密码、存款、取款、转账、排序、计算利息，编号分别为\*号，#号，p，1～9，0为退出程序。

用户根据主菜单中显示的功能模块及其相应编号，选择相应编号，执行相应的功能。所选模块执行完毕后，返回系统主菜单。

* 1. 数据结构设计

实验内容涉及到多条记录的处理，每条记录含有不同类型的数据项，使用单链表来储存较为方便，链表节点设计如下：

struct Account

{

string a\_number;

string a\_name;

string a\_sex;

int a\_key;

long long a\_sum;

int a\_date;

string a\_state;

Account\* a\_next;

};

单链表结构中，每个节点的a\_next作为指针域，用于存储其后续节点的地址，其他部分为数据域，存储节点中的数据。

* 1. 函数设计

函数设计的主要内容包括函数的名称，函数的功能与函数中参数类型的说明。

|  |  |  |
| --- | --- | --- |
| 函数名称 | 函数功能 | 参数说明 |
| bool operator<(string s1, string s2) | 对两字符串大小判断（<运算符）进行定义 | s1,s2表示两个不同字符串 |
| bool comp\_date(int a, int b) | 对两日期大小判断进行定义 | a,b表示两个日期 |
| Account\* CreateAccount(ifstream& fcin) | 从文件读取账户信息到单链表 | fcin表示读入文件的文件流对象名 |
| void ShowAccount(Account\*& Head) | 控制台打印输出全部账户信息 | Head表示账户链表的头指针 |
| void InputAccount(Account\* Head, char& bank) | 导出账户信息到文件 | Head表示账户链表的头指针，bank表示当前所在银行 |
| Account\* SearchAccount(Account\*& Head,string number) | 根据账号查询 | Head表示账户链表的头指针，number表示搜索的账号 |
| void AddAccount(Account\*& Head,char& bank) | 开户 | Head表示账户链表的头指针，bank表示当前所在银行 |
| void DeleteAccount(Account\*& Head,char& bank) | 销户 | Head表示账户链表的头指针，bank表示当前所在银行 |
| void SortAccount\_Number(Account\*& Head) | 按照账号排序 | Head表示账户链表的头指针 |
| void SortAccount\_Name(Account\*& Head) | 按照姓名排序 | Head表示账户链表的头指针 |
| void SortAccount\_Sum(Account\*& Head) | 按照金额排序 | Head表示账户链表的头指针 |
| void SortAccount\_Date(Account\*& Head) | 按照开户日期排序 | Head表示账户链表的头指针 |
| void ShowSearchAccount(Account\*& Head) | 控制台打印输出搜索到的某个账户信息 | Head表示账户链表的头指针 |
| void ChangeKey(Account\*& Head,char& bank) | 修改密码 | Head表示账户链表的头指针，bank表示当前所在银行 |
| void AddMoney(Account\*& Head,char& bank) | 存款 | Head表示账户链表的头指针 |
| void GetMoney(Account\*& Head,char& bank) | 取款 | Head表示账户链表的头指针 |
| void Interest(Account\* Head) | 计算利息 | Head表示账户链表的头指针 |
| void Transfer(Account\*& Head,char& bank) | 转账 | Head表示账户链表的头指针，bank表示当前所在银行 |
| void DeleteAll(Account\*& Head) | 删除整个链表 | Head表示账户链表的头指针 |

1. 详细设计
   1. 各个函数的调用关系图

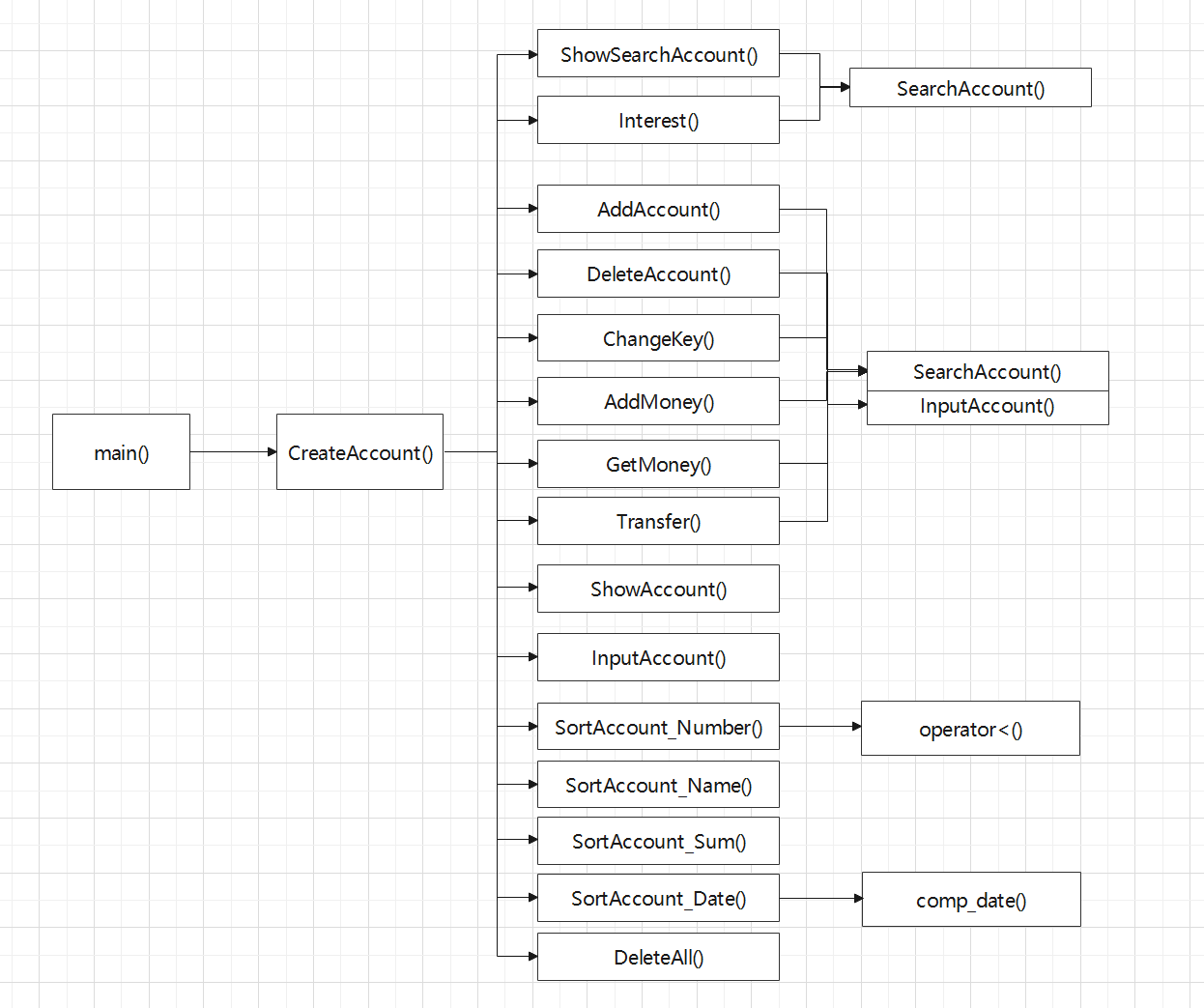


图 4-2 函数调用关系图

* 1. 各个函数功能描述
     1. 初始化系统界面

/\*

在控制台界面要求用户选择银行并输出各个功能模块的名称，要求用户选择功能，退卡或退出程序

\*/

int main()

{

ifstream fcin;

cout << "请输入您的账户所在银行：";

char bank;

cin >> bank;

switch (bank)

{

case 'a':fcin.open("a.txt", ios::in ); break;

case 'b':fcin.open("b.txt", ios::in ); break;

case 'c':fcin.open("c.txt", ios::in ); break;

case 'd':fcin.open("d.txt", ios::in ); break;

case 'e':fcin.open("e.txt", ios::in ); break;

default:cout << "不存在该银行。" << endl; break;

}

Account\* Head = CreateAccount(fcin);

fcin.close();

while (1)

{

cout << "-------------------------------------------------------" << endl;

cout << "\n输入\*展示所有账户；输入#退卡；输入p输出账户信息到文件\n1.查找账户；2.开户；3.销户\n4.修改密码；5.存款；6.取款\n7.转账；8.排序；9.计算利息\n输入0退出\n";

char i;

cin >> i;

switch (i)

{

case '\*':ShowAccount(Head); break;//展示账户信息

case 'p':InputAccount(Head, bank); break;//将单链表导出到文件

case '1':ShowSearchAccount(Head); break;//查找账户

case '2':AddAccount(Head,bank); break;//开户

case '3':DeleteAccount(Head,bank); break;//销户

case '4':ChangeKey(Head,bank); break;//修改密码

case '5':AddMoney(Head,bank); break;//存款

case '6':GetMoney(Head,bank); break;//取款

case '7':Transfer(Head, bank); break;//转账

case '8':

{

cout << "请选择排序依据（从小到大）：\n1.账号；2.姓名（非中文）；3.金额；4.开户日期\n" << endl;

int j;

cin >> j;

switch (j)

{

case 1:SortAccount\_Number(Head); break;

case 2:SortAccount\_Name(Head); break;

case 3:SortAccount\_Sum(Head); break;

case 4:SortAccount\_Date(Head); break;

default:cout << "无法识别的字符。" << endl; break;

}

break;

}

case '9':Interest(Head); break;

case '#':

{

DeleteAll(Head);

cout << "退卡成功。" << endl;

cout << "请输入您的账户所在银行：";

cin >> bank;

switch (bank)

{

case 'a':fcin.open("a.txt", ios::in); break;

case 'b':fcin.open("b.txt", ios::in); break;

case 'c':fcin.open("c.txt", ios::in); break;

case 'd':fcin.open("d.txt", ios::in); break;

case 'e':fcin.open("e.txt", ios::in); break;

default:cout << "不存在该银行。" << endl; break;

}

Head = CreateAccount(fcin);

fcin.close();

break;

}

case '0':cout << "程序已退出。" << endl; return 0;//退出

default:cout << "无法识别的字符，已默认退出。" << endl; return 0;//退出

}

}

return 0;

}

* + 1. 具体功能的实现

/\*

运算符重载，使用小于号<比较两字符串大小时采用该方法。

\*/

bool operator<(string s1, string s2)

{

int min\_size = min(s1.size(), s2.size());

for (int i = 0; i < min\_size; i++)

{

if (s1[i] < s2[i])

{

return true;

}

else if (s1[i] > s2[i])

{

return false;

}

if (i == min\_size - 1 and s1[i] == s2[i])

{

if (s1.size() < s2.size())

{

return true;

}

else

{

return false;

}

}

}

}

/\*

日期比较，比较两个八位日期(yyyymmdd)时采用该方法。

\*/

bool comp\_date(int a, int b)//小的日期排在前

{

if (a / 10000 < b / 10000)

{

return true;

}

else if (a / 10000 == b / 10000 and (a / 100 - a / 10000) < (b / 100 - b / 10000))

{

return true;

}

else if (a / 10000 == b / 10000 and (a / 100 - a / 10000) == (b / 100 - b / 10000) and (a - a / 100 - a / 10000) < (b - b / 100 - b / 10000))

{

return true;

}

else

{

return false;

}

}

/\*

从文件导入账户信息到单链表。

变量定义；

如果银行内没有账户，则输出"该银行不存在账户。"；

如果有账户，则将各个账户的信息导入到单链表中，并返回头指针。

\*/

Account\* CreateAccount(ifstream& fcin)

{

Account\* Head = nullptr;

Account\* q = nullptr;

if (fcin.peek() == -1)

{

cout << "该银行不存在账户。" << endl;

return Head;

}

while (fcin.peek() != -1)

{

Account\* p = new Account;

fcin >> p->a\_number >> p->a\_name >> p->a\_sex >> p->a\_key >> p->a\_sum >> p->a\_date >> p->a\_state;

if (Head == nullptr)

{

Head = p;

q = p;

if (fcin.peek() == '\n')

{

fcin.get();

}

continue;

}

q->a\_next = p;

q = p;

if (fcin.peek() == '\n')

{

fcin.get();

}

}

if (q != nullptr)

{

q->a\_next = nullptr;

}

return Head;

}

/\*

展示所有账号。

如果头指针为空指针，则输出"该银行内不存在账号。"；

变量定义；

如果头指针不为空，则按链表输出所有的账号信息。

\*/

void ShowAccount(Account\*& Head)

{

if (Head == nullptr)

{

cout << "该银行内不存在账号。" << endl;

return;

}

Account\* p = Head;

while (p != nullptr)

{

cout << p->a\_number << " " << p->a\_name << " " << p->a\_sex << " " << p->a\_key << " " << p->a\_sum << " " << p->a\_date << " " << p->a\_state << endl;

p = p->a\_next;

}

}

/\*

将账户信息导出到文件。

打开文件；

将链表内容依次输出到文件；

关闭文件。

\*/

void InputAccount(Account\* Head, char& bank)

{

ofstream fcout;

switch (bank)

{

case 'a':fcout.open("a.txt", ios::out); break;

case 'b':fcout.open("b.txt", ios::out); break;

case 'c':fcout.open("c.txt", ios::out); break;

case 'd':fcout.open("d.txt", ios::out); break;

case 'e':fcout.open("e.txt", ios::out); break;

default:cout << "不存在此银行。" << endl; return;

}

Account\* p = Head;

while (p != nullptr)

{

if (p == Head)

{

fcout << p->a\_number << " " << p->a\_name << " " << p->a\_sex << " " << p->a\_key << " " << p->a\_sum << " " << p->a\_date << " " << p->a\_state;

}

else

{

fcout << endl << p->a\_number << " " << p->a\_name << " " << p->a\_sex << " " << p->a\_key << " " << p->a\_sum << " " << p->a\_date << " " << p->a\_state;

}

p = p->a\_next;

}

cout << "bank " << bank << ":Input Successfully.";

fcout.close();

}

/\*

搜索账号。

变量定义；

如果头指针为空指针，或搜索不到对应账号，则返回空指针；

如果搜索到了对应账号，则返回该账号数据的地址。

\*/

Account\* SearchAccount(Account\*& Head,string number)

{

if (Head == nullptr)

{

return nullptr;

}

Account\* p = Head;

bool flag = false;

while (p != nullptr)

{

if (p->a\_number == number)

{

break;

}

p = p->a\_next;

}

if (p == nullptr)

{

return nullptr;

}

else

{

return p;

}

}

/\*

开户。

变量定义；

如果输入的账号已存在，则打印"账户已存在。"；

如果不存在，则输入账户信息，将账户加到链表末尾，并将账户链表保存到文件。

\*/

void AddAccount(Account\*& Head,char& bank)

{

cout << "请输入您的账号：";

string num;

cin >> num;

Account\* temp = SearchAccount(Head, num);

if (temp != nullptr)

{

cout << "账户已存在。" << endl;

}

else

{

Account\* p = new Account;

p->a\_number = num;

cout << "请输入姓名、性别、密码、金额、开户日期、账户状态" << endl;

cin >> p->a\_name >> p->a\_sex >> p->a\_key >> p->a\_sum >> p->a\_date >> p->a\_state;

p->a\_next = nullptr;

if (Head == nullptr)

{

Head = p;

cout << "开户成功。" << endl;

return;

}

Account\* temp0 = Head;

while (temp0 != nullptr)

{

if (temp0->a\_next == nullptr)

{

temp0->a\_next = p;

break;

}

temp0 = temp0->a\_next;

}

InputAccount(Head, bank);

cout << "开户成功。" << endl;

}

}

/\*

销户。

变量定义；

如果输入的账号不存在，则打印"账户不存在。"；

如果存在，则输入密码；

如果密码错误，则输出"密码错误！"；

若密码正确，则将账户从链表删除，并将账户链表保存到文件。

\*/

void DeleteAccount(Account\*& Head,char& bank)

{

cout << "请输入您的账号：";

string num;

cin >> num;

Account\* temp = SearchAccount(Head, num);

if (temp == nullptr)

{

cout << "账户不存在。" << endl;

}

else

{

cout << "请输入密码：";

int k;

cin >> k;

if (k != temp->a\_key)

{

cout << "密码错误！" << endl;

return;

}

else

{

Account\* p = Head;

while (p != nullptr)

{

if (p == temp)

{

break;

}

if (p->a\_next == temp)

{

break;

}

p = p->a\_next;

}

if (temp == Head)

{

Head = Head->a\_next;

temp->a\_sum = 0;

delete temp;

InputAccount(Head, bank);

cout << "销户成功" << endl;

}

else if (p != nullptr)

{

p->a\_next = temp->a\_next;

temp->a\_sum = 0;

delete temp;

InputAccount(Head, bank);

cout << "销户成功" << endl;

}

}

}

}

/\*

按账号排序。

变量定义；

如果头指针为空指针或链表仅含一条数据，则返回；

如果含多条数据，按15位账号的大小对各个数据进行选择排序。

\*/

void SortAccount\_Number(Account\*& Head)

{

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

if (Head->a\_next == nullptr)

{

return;

}

Account\* p = Head;

Account\* min\_ac = Head;

Account\* sorted = nullptr;

Account\* now\_ac = Head;

while (now\_ac->a\_next != nullptr)

{

while (p != nullptr)

{

if (p->a\_number < min\_ac->a\_number)

{

min\_ac = p;//min\_ac指向待测数据中的最小值

}

p = p->a\_next;

}

Account\* min\_last = Head;//指针min\_last指向最小值数据的前一个数据

if (Head == min\_ac)

{

min\_last = nullptr;//如果最小值就是头部，min\_last为空指针

}

else

{

while (min\_last->a\_next != min\_ac)

{

min\_last = min\_last->a\_next;

}

}

if (sorted == nullptr)//刚开始排序，未排第一个

{

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//（头部与非下一个数据的某一个数据）两个不连在一起的位置交换

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

if (min\_last != nullptr)

{

min\_last->a\_next = now\_ac;

}

now\_ac->a\_next = temp;

Head = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)//（头部与下一个数据）两个连在一起的数据交换

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

Head = min\_ac;

}

}

else//排完第一个

{

Account\* now\_last = Head;//是否可以把now\_last用sorted替代

while (now\_last->a\_next != now\_ac)

{

now\_last = now\_last->a\_next;

}

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//两个不连在一起的数据交换位置

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

now\_ac->a\_next = temp;

min\_last->a\_next = now\_ac;

now\_last->a\_next = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

now\_last->a\_next = min\_ac;

}

}

sorted = min\_ac;

now\_ac = min\_ac->a\_next;

p = sorted->a\_next;

min\_ac = min\_ac->a\_next;

}

cout << "已完成按账号排序。" << endl;

}

/\*

按姓名排序。

变量定义；

如果头指针为空指针或链表仅含一条数据，则返回；

如果含多条数据，按姓名对各个数据进行选择排序。

\*/

void SortAccount\_Name(Account\*& Head)

{

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

if (Head->a\_next == nullptr)

{

return;

}

Account\* p0 = Head;

while (p0 != nullptr)

{

for (int i = 0; i < p0->a\_name.size(); i++)

{

if (p0->a\_name[i] < 0)

{

cout << "暂不支持以汉字排序。" << endl;

return;

}

}

p0 = p0->a\_next;

}

Account\* p = Head;

Account\* min\_ac = Head;

Account\* sorted = nullptr;

Account\* now\_ac = Head;

while (now\_ac->a\_next != nullptr)

{

while (p != nullptr)

{

if (p->a\_name < min\_ac->a\_name)

{

min\_ac = p;//min\_ac指向待测数据中的最小值

}

p = p->a\_next;

}

Account\* min\_last = Head;//指针min\_last指向最小值数据的前一个数据

if (Head == min\_ac)

{

min\_last = nullptr;//如果最小值就是头部，min\_last为空指针

}

else

{

while (min\_last->a\_next != min\_ac)

{

min\_last = min\_last->a\_next;

}

}

if (sorted == nullptr)//刚开始排序，未排第一个

{

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//（头部与非下一个数据的某一个数据）两个不连在一起的位置交换//min\_last!=now可行

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

if (min\_last != nullptr)

{

min\_last->a\_next = now\_ac;

}

now\_ac->a\_next = temp;

Head = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)//（头部与下一个数据）两个连在一起的数据交换

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

Head = min\_ac;

}

}

else//排完第一个

{

Account\* now\_last = Head;//是否可以把now\_last用sorted替代

while (now\_last->a\_next != now\_ac)

{

now\_last = now\_last->a\_next;

}

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//两个不连在一起的数据交换位置

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

now\_ac->a\_next = temp;

min\_last->a\_next = now\_ac;

now\_last->a\_next = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

now\_last->a\_next = min\_ac;

}

}

sorted = min\_ac;

now\_ac = min\_ac->a\_next;

p = sorted->a\_next;

min\_ac = min\_ac->a\_next;

}

cout << "已完成按姓名排序。" << endl;

}

/\*

按金额排序。

变量定义；

如果头指针为空指针或链表仅含一条数据，则返回；

如果含多条数据，按金额对各个数据进行选择排序。

\*/

void SortAccount\_Sum(Account\*& Head)

{

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

if (Head->a\_next == nullptr)

{

return;

}

Account\* p = Head;

Account\* min\_ac = Head;

Account\* sorted = nullptr;

Account\* now\_ac = Head;

while (now\_ac->a\_next != nullptr)

{

while (p != nullptr)

{

if (p->a\_sum < min\_ac->a\_sum)

{

min\_ac = p;//min\_ac指向待测数据中的最小值

}

p = p->a\_next;

}

Account\* min\_last = Head;//指针min\_last指向最小值数据的前一个数据

if (Head == min\_ac)

{

min\_last = nullptr;//如果最小值就是头部，min\_last为空指针

}

else

{

while (min\_last->a\_next != min\_ac)

{

min\_last = min\_last->a\_next;

}

}

if (sorted == nullptr)//刚开始排序，未排第一个

{

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//（头部与非下一个数据的某一个数据）两个不连在一起的位置交换//min\_last!=now可行

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

if (min\_last != nullptr)

{

min\_last->a\_next = now\_ac;

}

now\_ac->a\_next = temp;

Head = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)//（头部与下一个数据）两个连在一起的数据交换

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

Head = min\_ac;

}

}

else//排完第一个

{

Account\* now\_last = Head;//是否可以把now\_last用sorted替代

while (now\_last->a\_next != now\_ac)

{

now\_last = now\_last->a\_next;

}

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//两个不连在一起的数据交换位置

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

now\_ac->a\_next = temp;

min\_last->a\_next = now\_ac;

now\_last->a\_next = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

now\_last->a\_next = min\_ac;

}

}

sorted = min\_ac;

now\_ac = min\_ac->a\_next;

p = sorted->a\_next;

min\_ac = min\_ac->a\_next;

}

cout << "已完成按金额排序。" << endl;

}

/\*

按开户日期排序。

变量定义；

如果头指针为空指针或链表仅含一条数据，则返回；

如果含多条数据，按日期先后对各个数据进行选择排序。

\*/

void SortAccount\_Date(Account\*& Head)

{

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

if (Head->a\_next == nullptr)

{

return;

}

Account\* p = Head;

Account\* min\_ac = Head;

Account\* sorted = nullptr;

Account\* now\_ac = Head;

while (now\_ac->a\_next != nullptr)

{

while (p != nullptr)

{

if (comp\_date(p->a\_date, min\_ac->a\_date))

{

min\_ac = p;//min\_ac指向待测数据中的最小值

}

p = p->a\_next;

}

Account\* min\_last = Head;//指针min\_last指向最小值数据的前一个数据

if (Head == min\_ac)

{

min\_last = nullptr;//如果最小值就是头部，min\_last为空指针

}

else

{

while (min\_last->a\_next != min\_ac)

{

min\_last = min\_last->a\_next;

}

}

if (sorted == nullptr)//刚开始排序，未排第一个

{

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//（头部与非下一个数据的某一个数据）两个不连在一起的位置交换//min\_last!=now可行

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

if (min\_last != nullptr)

{

min\_last->a\_next = now\_ac;

}

now\_ac->a\_next = temp;

Head = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)//（头部与下一个数据）两个连在一起的数据交换

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

Head = min\_ac;

}

}

else//排完第一个

{

Account\* now\_last = Head;//是否可以把now\_last用sorted替代

while (now\_last->a\_next != now\_ac)

{

now\_last = now\_last->a\_next;

}

if (now\_ac != min\_ac and now\_ac->a\_next != min\_ac)//两个不连在一起的数据交换位置

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac->a\_next;

now\_ac->a\_next = temp;

min\_last->a\_next = now\_ac;

now\_last->a\_next = min\_ac;

}

else if (now\_ac->a\_next == min\_ac)

{

Account\* temp = min\_ac->a\_next;

min\_ac->a\_next = now\_ac;

now\_ac->a\_next = temp;

now\_last->a\_next = min\_ac;

}

}

sorted = min\_ac;

now\_ac = min\_ac->a\_next;

p = sorted->a\_next;

min\_ac = min\_ac->a\_next;

}

cout << "已完成按开户日期排序。" << endl;

}

/\*

展示搜索到的账号。

变量定义；

当头指针为空指针时，输出"该银行内不存在账户。"；

当搜索到的指针为空指针时，输出"账户不存在。"；

当搜索到账号时，输出账号信息。

\*/

void ShowSearchAccount(Account\*& Head)

{

string tempstr;

cout << "请输入查询的账号：" << endl;

cin >> tempstr;

if (Head == nullptr)

{

cout << "该银行内不存在账户。" << endl;

return;

}

Account\* temp = SearchAccount(Head, tempstr);

if (temp == nullptr)

{

cout << "账户不存在。" << endl;

}

else

{

cout << "账号：" << temp->a\_number << "\n姓名：" << temp->a\_name << "\n性别：" << temp->a\_sex << "\n余额：" << temp->a\_sum << "\n开户日期：" << temp->a\_date << endl;

}

}

/\*

修改密码。

定义变量；

当头指针为空指针时，输出"该银行不存在账户。"，返回；

当搜索到账户时，要求用户输入原密码；

若密码错误，输出"密码错误！"；

若密码正确，要求用户输入新密码，用新密码代替旧密码，并将账户信息更新至文件。

\*/

void ChangeKey(Account\*& Head,char& bank)

{

Account\* p = Head;

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

cout << "请输入待修改密码的账号：";

string num;

cin >> num;

Account\* temp = SearchAccount(Head, num);

if (temp == nullptr)

{

cout << "账户不存在。" << endl;

}

else

{

cout << "请输入原密码：";

int c\_key;

cin >> c\_key;

if (c\_key != temp->a\_key)

{

cout << "密码错误！" << endl;

}

else

{

cout << "请输入新密码：";

int new\_key;

cin >> new\_key;

temp->a\_key = new\_key;

InputAccount(Head, bank);

cout << "修改成功。" << endl;

}

}

}

/\*

存款。

变量定义；

当头指针为空指针时，输出"该银行不存在账户。"，返回；

当搜索到账户时，要求用户输入密码；

若密码错误，输出"密码错误！"；

若密码正确，要求用户输入存款金额，将存款金额加到余额，并将账户信息更新至文件。

\*/

void AddMoney(Account\*& Head,char& bank)

{

Account\* p = Head;

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

cout << "请输入存款账号：";

string num;

cin >> num;

Account\* temp = SearchAccount(Head, num);

if (temp == nullptr)

{

cout << "账户不存在。" << endl;

}

else

{

cout << "请输入密码：";

int c\_key;

cin >> c\_key;

if (c\_key != temp->a\_key)

{

cout << "密码错误！" << endl;

}

else

{

cout << "请输入存款金额：";

long long money\_add;

cin >> money\_add;

temp->a\_sum += money\_add;

InputAccount(Head, bank);

cout << "存款成功。" << endl;

}

}

}

/\*

取款

变量定义；

当头指针为空指针时，输出"该银行不存在账户。"，返回；

当搜索不到账户，输出"账户不存在。"，返回；

当搜索到账户时，要求用户输入密码；

若密码错误，输出"密码错误！"；

若密码正确，要求用户输入取款金额；

若取款金额大于余额，输出"余额不足。"；

若取款金额大于等于余额，将取款金额从余额减去，并将账户信息更新至文件。

\*/

void GetMoney(Account\*& Head,char& bank)

{

cout << "请输入取款账号：";

string num;

cin >> num;

Account\* temp = SearchAccount(Head, num);

if (temp == nullptr)

{

cout << "账户不存在。" << endl;

}

else

{

cout << "请输入密码：";

int c\_key;

cin >> c\_key;

if (c\_key != temp->a\_key)

{

cout << "密码错误！" << endl;

}

else

{

cout << "请输入取款金额：";

long long money\_get;

cin >> money\_get;

if (money\_get > temp->a\_sum)

{

cout << "余额不足。" << endl;

}

else

{

temp->a\_sum -= money\_get;

InputAccount(Head, bank);

cout << "取款成功。" << endl;

}

}

}

}

/\*

计算利息

当头指针为空指针时，输出"该银行不存在账户。"，返回；

当搜索不到账户，输出"账户不存在。"，返回；

当找到账户，要求输入存款到期日期；

若到期日期早于存款日期，输出"日期有误。"，返回；

若到期日期晚于存款日期或为存款日期当天，计算存款年数，根据活期或定期计算利息。

\*/

void Interest(Account\* Head)

{

if (Head == nullptr)

{

cout << "该银行不存在账户。" << endl;

return;

}

cout << "请输入您的账号：";

string num;

cin >> num;

Account\* result = SearchAccount(Head, num);

if (result == nullptr)

{

cout << "未找到账户。" << endl;

return;

}

if (result->a\_state != "活期" and result->a\_state != "定期")

{

cout << "无法识别存款类型" << endl;

return;

}

cout << "请输入存款到期日期：";

int date\_arrive;

cin >> date\_arrive;

if (date\_arrive < result->a\_date)

{

cout << "日期有误。" << endl;

return;

}

int year0 = (date\_arrive - result->a\_date) / 10000;

double year = year0;

int left\_res = result->a\_date % 10000, left\_arr = date\_arrive % 10000;

if (left\_arr < left\_res)

{

year -= 0.5;

}

else if (left\_arr > left\_res)

{

year += 0.5;

}

double s;

if (result->a\_state == "活期")

{

s = year \* result->a\_sum \* 0.0035;

}

else

{

if (year <= 1)

{

s = year \* result->a\_sum \* 0.0275;

}

else if (year <= 2)

{

s = result->a\_sum \* 0.0275 + (year - 1) \* result->a\_sum \* 0.0335;

}

else if (year > 2)

{

s = result->a\_sum \* 0.0275 + result->a\_sum \* 0.0335 + (year - 2) \* result->a\_sum \* 0.04;

}

}

cout << fixed << setprecision(2) << "预计总利息为：" << s << "元" << endl;

}

/\*

转账。

定义变量；

若收款方与转账方所在银行不存在账户，输出"当前/该银行不存在账户。"，返回；

若收款方所在银行与转账方所在银行相同，在当前链表内操作；

若收款方所在银行与转账方所在银行不同，打开收款方银行文件并导入至新链表；

要求输入转账方与收款方账户，若账户不正确，输出"您的/对方账户不存在。"，返回；

要求输入转账方密码，若密码错误，输出"密码错误！"，返回；

若账号均存在且密码正确，输入转账金额，若金额小于转账方余额，输出"余额不足。"；

若余额大于等于转账金额，从转账方余额减去转账金额，并向收款方余额加上转账金额，输出"转账成功。"，并将账户链表导入文件。

\*/

void Transfer(Account\*& Head,char& bank)

{

if (Head == nullptr)

{

cout << "当前银行不存在账户。" << endl;

return;

}

cout << "请输入对方账户的银行：";

char bank\_t;

cin >> bank\_t;

if (bank\_t == bank)

{

cout << "请输入对方账号：";

string num\_p;

cin >> num\_p;

Account\* q = SearchAccount(Head, num\_p);

if (q == nullptr)

{

cout << "对方账户不存在。" << endl;

return;

}

else

{

Account\* p = nullptr;

cout << "请输入您的账号：";

string num\_q;

cin >> num\_q;

p = SearchAccount(Head, num\_q);

if (p == nullptr)

{

cout << "您的账户不存在。" << endl;

return;

}

else

{

cout << "请输入密码：";

int key\_q;

cin >> key\_q;

if (key\_q != p->a\_key)

{

cout << "密码错误。" << endl;

return;

}

else

{

cout << "请输入转账金额：";

long long money\_transfer\_p;

cin >> money\_transfer\_p;

if (money\_transfer\_p > p->a\_sum)

{

cout << "余额不足。" << endl;

}

else

{

p->a\_sum -= money\_transfer\_p;

q->a\_sum += money\_transfer\_p;

InputAccount(Head, bank);

cout << "转账成功。" << endl;

}

return;

}

}

}

}

ifstream fcin\_t;

ofstream fcout\_t;

switch (bank\_t)

{

case 'a':fcin\_t.open("a.txt", ios::in); break;

case 'b':fcin\_t.open("b.txt", ios::in); break;

case 'c':fcin\_t.open("c.txt", ios::in); break;

case 'd':fcin\_t.open("d.txt", ios::in); break;

case 'e':fcin\_t.open("e.txt", ios::in); break;

default:cout << "不存在该银行。" << endl; return;

}

Account\* Head\_t = CreateAccount(fcin\_t);

if (Head\_t == nullptr)

{

cout << "该银行内不存在账户。" << endl;

return;

}

cout << "请输入对方账号：";

string num\_t;

cin >> num\_t;

Account\* q = SearchAccount(Head\_t, num\_t);

if (q == nullptr)

{

cout << "对方账户不存在。" << endl;

return;

}

else

{

Account\* p = nullptr;

cout << "请输入您的账号：";

string num;

cin >> num;

p = SearchAccount(Head, num);

if (p == nullptr)

{

cout << "您的账户不存在。" << endl;

return;

}

else

{

cout << "请输入密码：";

int key;

cin >> key;

if (key != p->a\_key)

{

cout << "密码错误。" << endl;

return;

}

else

{

cout << "请输入转账金额：";

long long money\_transfer;

cin >> money\_transfer;

if (money\_transfer > p->a\_sum)

{

cout << "余额不足。" << endl;

}

else

{

p->a\_sum -= money\_transfer;

q->a\_sum += money\_transfer;

InputAccount(Head, bank);

InputAccount(Head\_t, bank\_t);

cout << "转账成功。" << endl;

}

}

}

}

}

/\*

删除整个链表。

从堆区内依次删去各个账户数据，并将头指针赋值为空指针。

\*/

void DeleteAll(Account\*& Head)

{

Account\* p = Head;

while (p != nullptr)

{

Account\* temp = p->a\_next;

delete p;

p = temp;

}

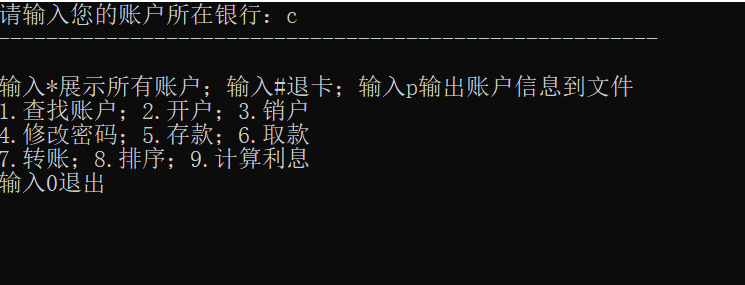
Head = nullptr;

}

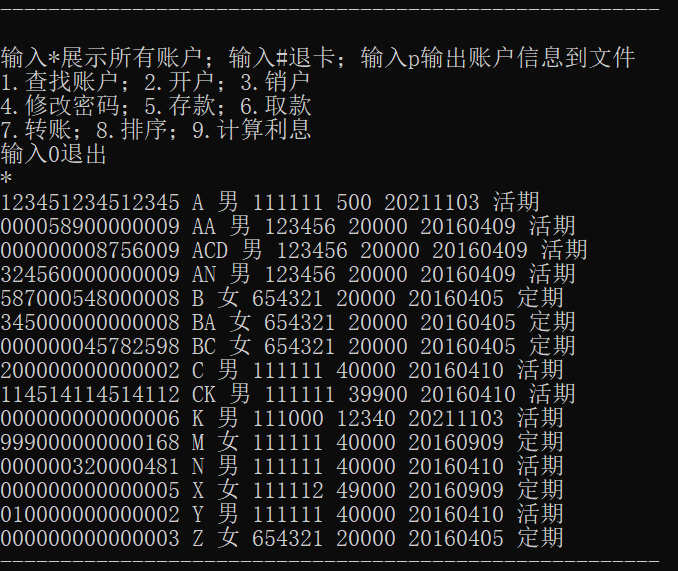
1. 功能测试

## 5.1 初始化界面

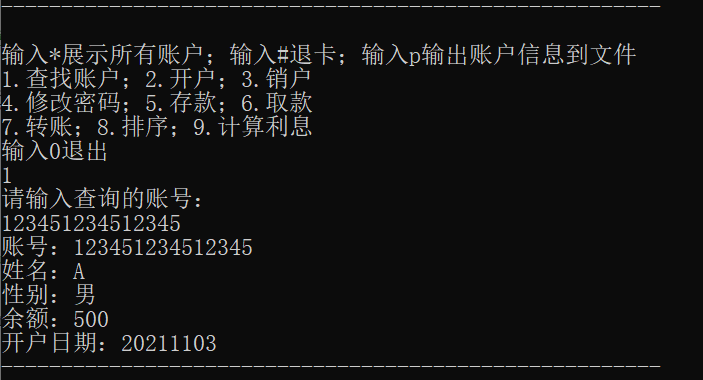




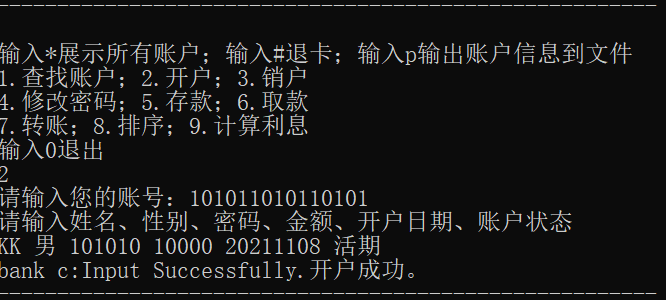
## 5.2 控制台打印所有账户



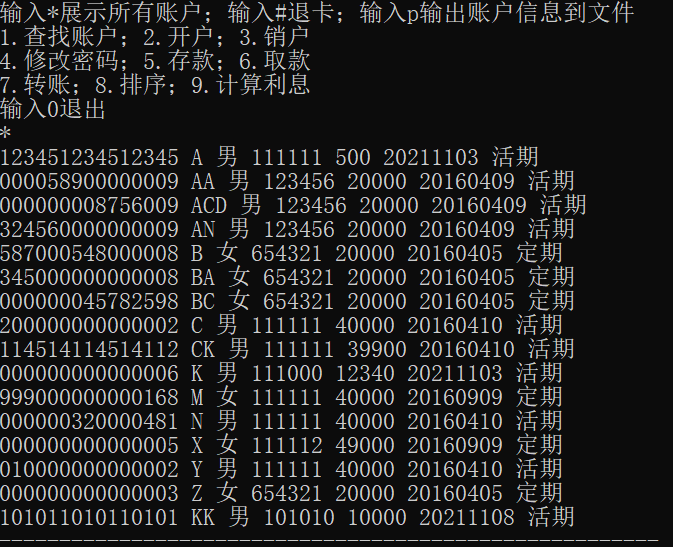
## 5.3 查询账户信息

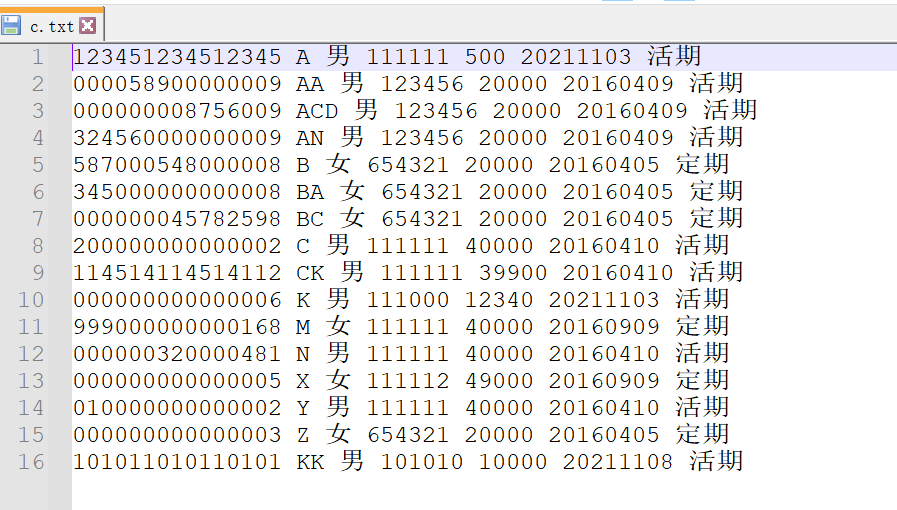


## 5.4 开户

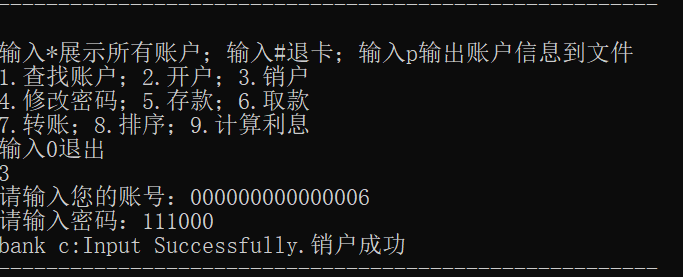


此时新账户已存放至链表及文件中

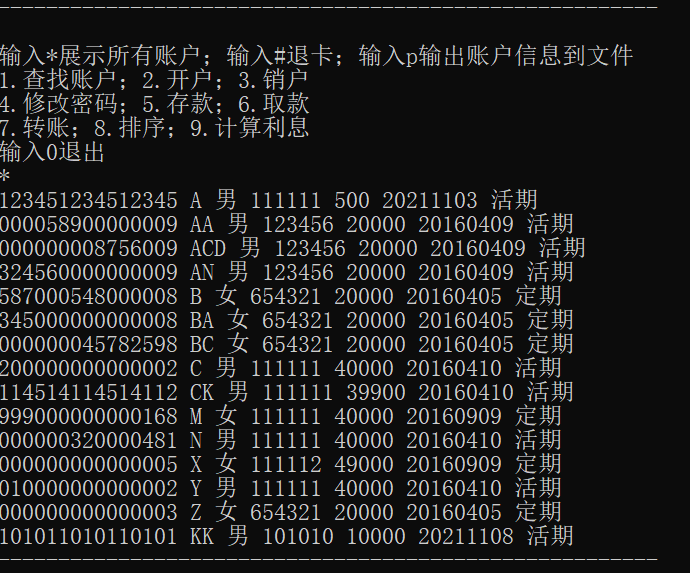


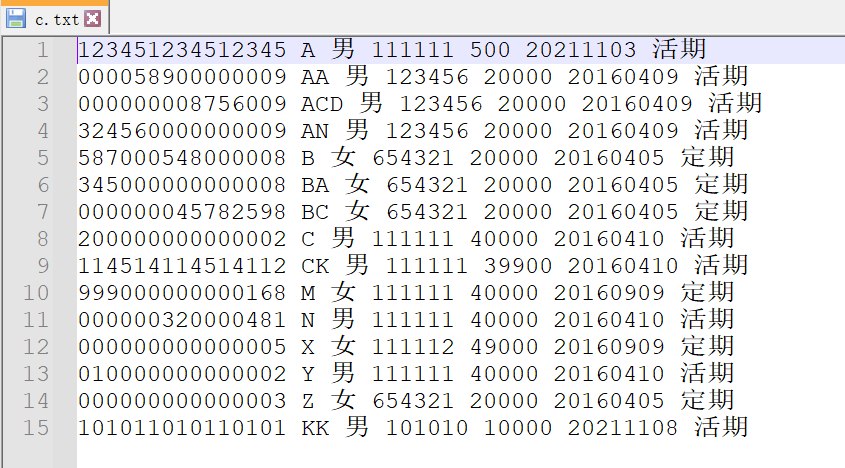


## 5.5 销户

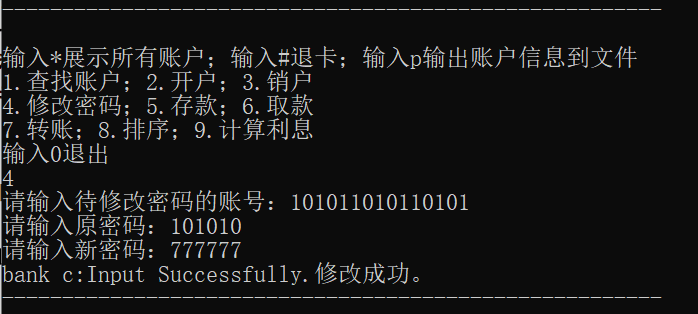


此时账户已从链表及文件中删除

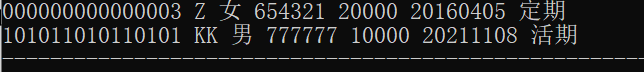


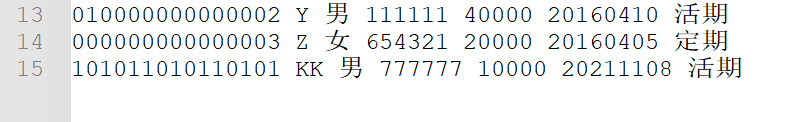


## 5.6 修改密码

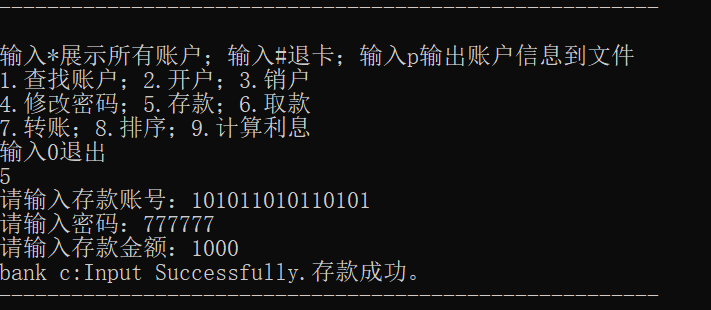


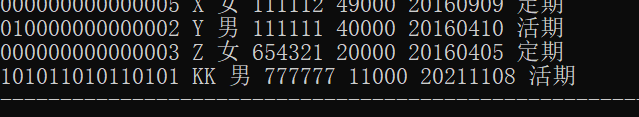
此时密码已更新至链表及文件中





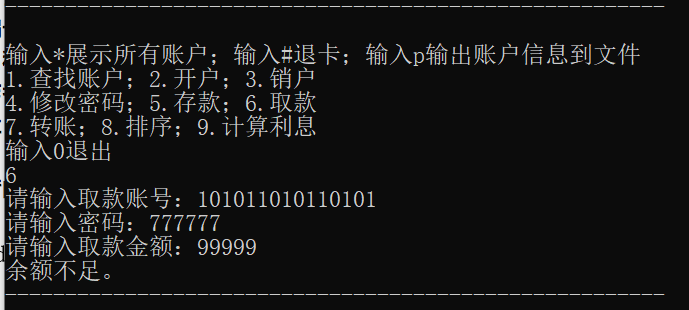
## 5.7 存款



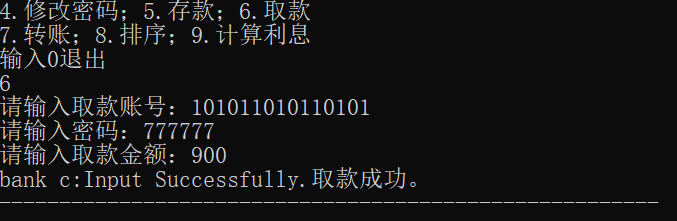


## 5.8 取款

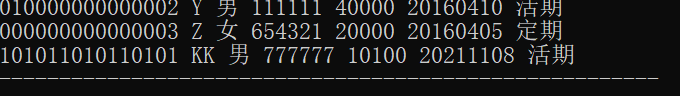
若取款金额大于余额，输出余额不足

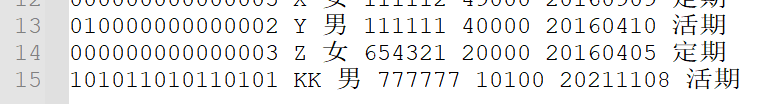


若可以取款，输出取款成功

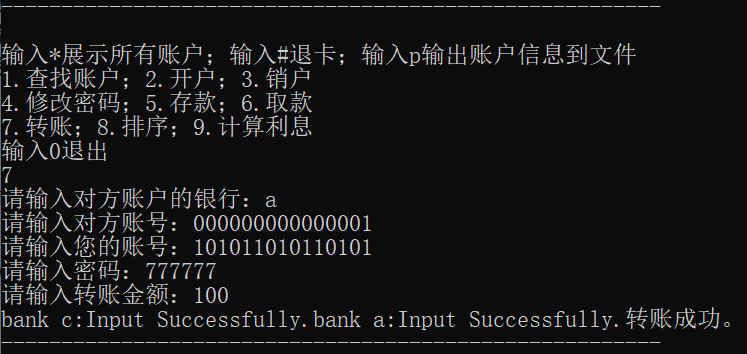


此时余额已更新至链表及文件中

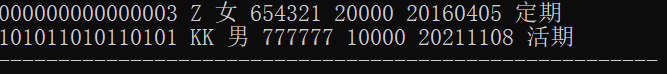


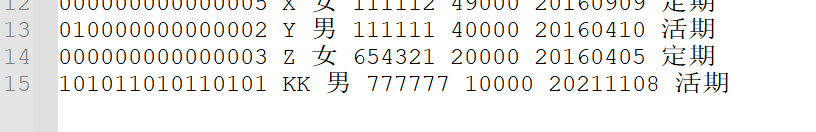


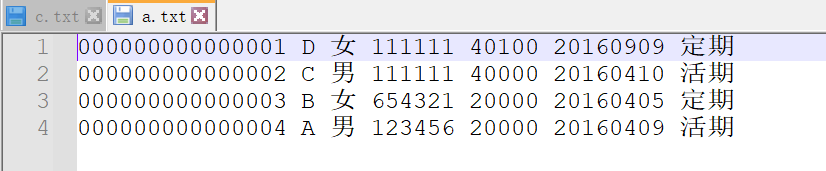
## 5.9 转账



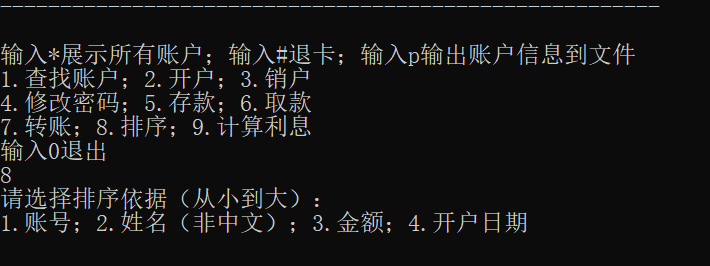
此时链表及文件中内容都已更新



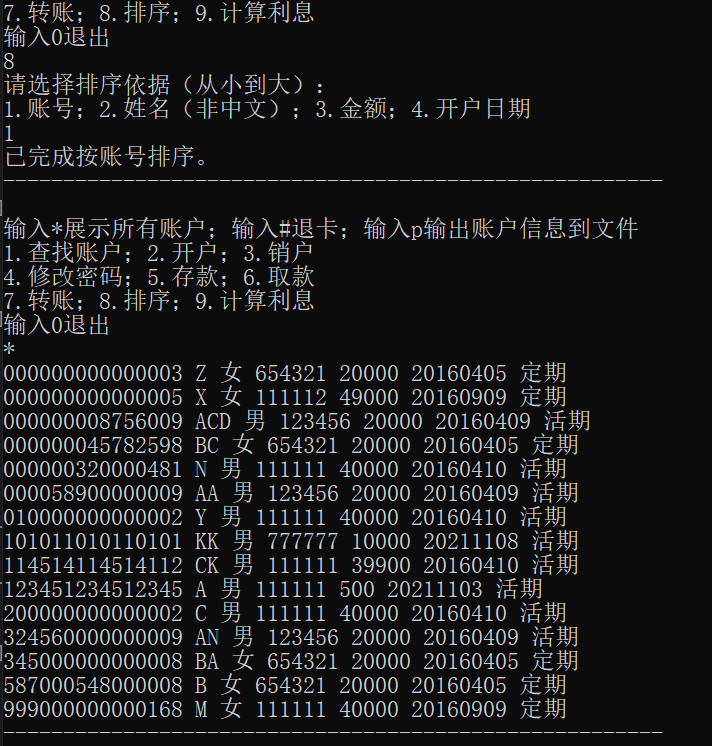




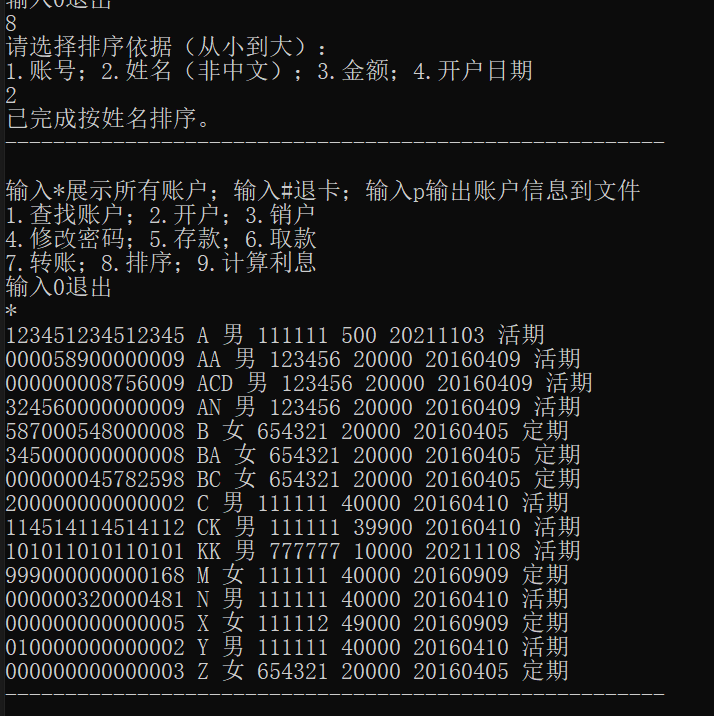
## 5.10 排序



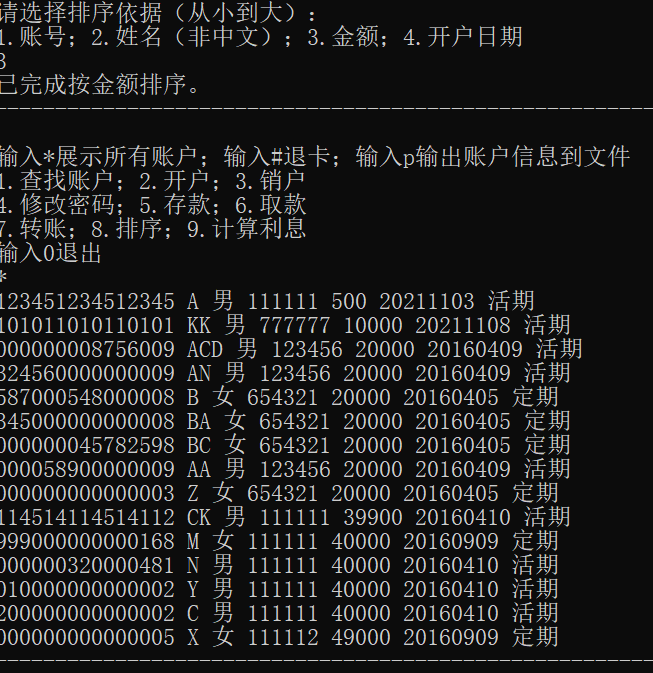
5.10.1 根据账号排序



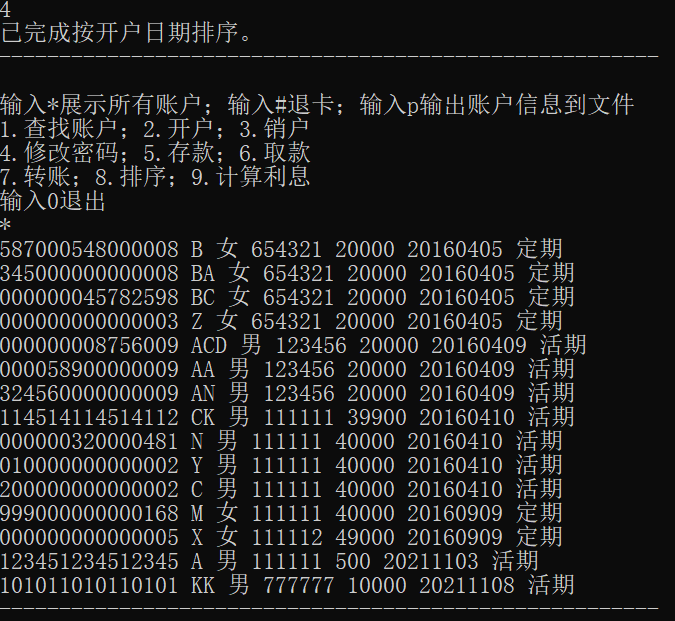
5.10.2 根据姓名排序



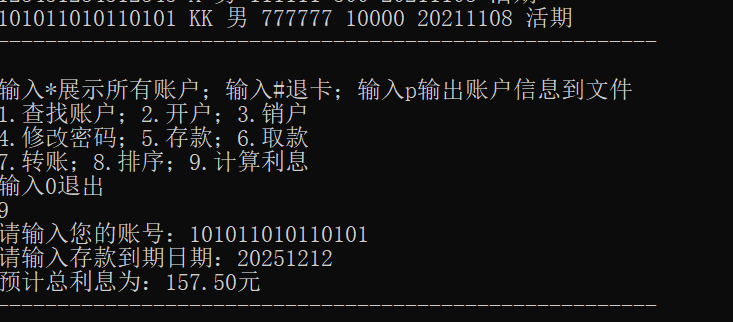
5.10.3 根据金额排序



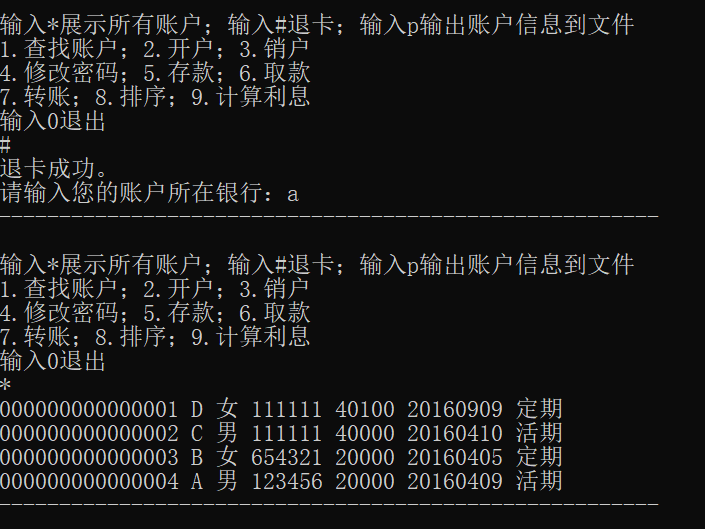
5.10.4 根据开户日期排序



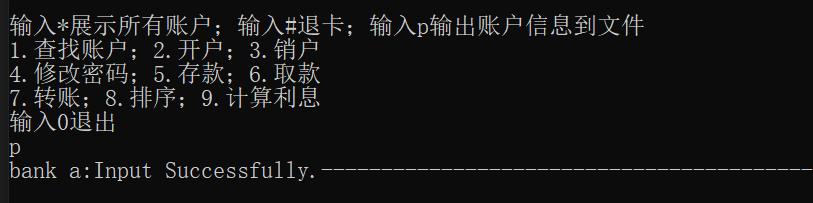
## 5.11 计算利息



## 5.12 退卡（更换银行）

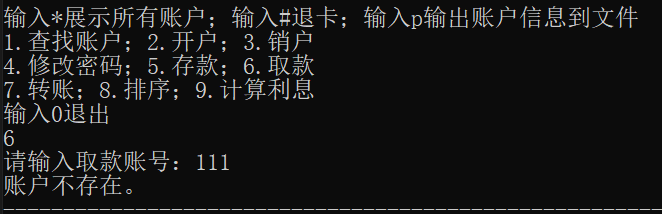


## 5.13 导出账户信息到文件

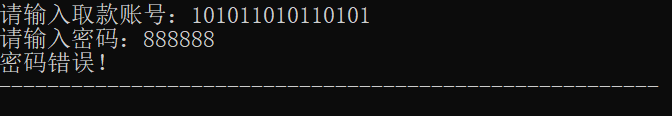


## 5.14 错误信息

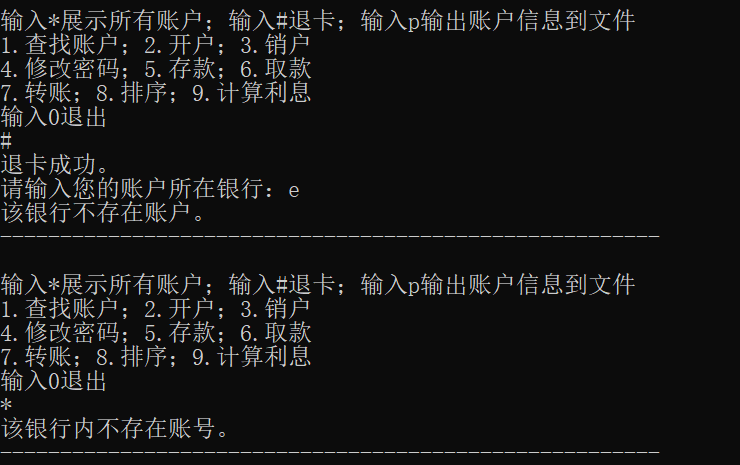
5.14.1 账户不存在



5.14.2 密码错误



5.14.3 银行内不存在账户



1. 总结

在项目设计过程中，我首先遇到的问题是如何对文件进行读取。当文件为空时，只有先进行一次读取操作后，下一次读取才能真正读到EOF。在多次尝试与验证后，我采用了函数cin.peek()，当其返回值为-1时表明读到了EOF，停止读取。其次，我遇到了对账户进行排序的问题，我对于将链表数据进行排序的操作并不是很熟练，因此我进行了多次尝试，补充排序过程中的各种分支结构，最终完善了排序操作。

现在的项目中，最主要的缺陷是不能完成按中文姓名进行排序；同时，对于界面美观度方面，我的程序仍然有不足之处，因此，对这一系统进行优化可以从这两方面入手。

在本次项目设计里，我学会了如何使用单链表结构，如何对文件进行读入和输出操作，以及如何对不同于数组的单链表结构进行排序，这对我程序设计学习起到了良好的复习与提高效果。