数据结构与算法设计（C++描述）

专题报告

实验名称：　银行账户管理系统设计

	姓名	学号	专业
小组成员	[redacted]	[redacted]	电子信息工程（徐特立英才班）
	[redacted]	[redacted]	电子信息工程（徐特立英才班）
	[redacted]	[redacted]	电子信息工程（徐特立英才班）
	[redacted]	[redacted]	电子信息工程（徐特立英才班）
	[redacted]	[redacted]	电子信息工程（徐特立英才班）
任课教师：	[redacted]	备注：	

版本

版本号	描述/改动	日期	责任人
1.0	初版	[redacted]	[redacted]
2.0	编写转账函数和文件读写函数	[redacted]	[redacted]
3.0	修复代码中存在的一些 bug	[redacted]	[redacted]
4.0	进一步完善系统操作流程与提示信息	[redacted]	[redacted]

一、基本情况

题目内容	设计并实现一个银行账户管理系统，支持客户和管理员两种身份登录，客户具备开户、查询、修改、存款、取款、转账、销户等功能，管理员还可浏览、查找账户信息等。
题目要求及约束条件	银行客户信息以一个账户一条记录的形式存储，每个账户记录包含的信息有身份标识号（ID）、姓名、电话、电子邮箱、身份证号、银行卡号、账户余额等信息。并实现题目要求所有功能。
成员分工	[redacted]

二、设计与实现

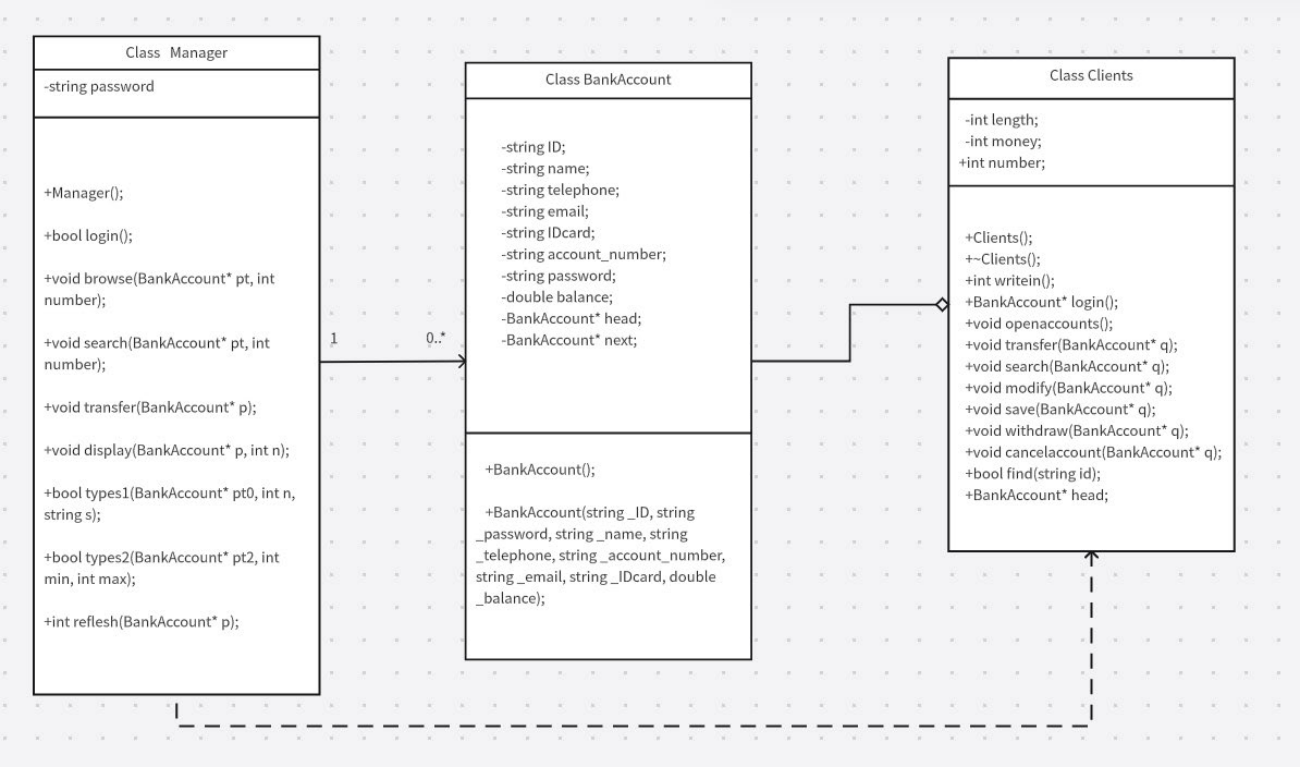
摘要：

本系统基于C++语言设计了一个银行账户管理系统，支持客户和管理员两种身份登录，具备开户、查询、存款、取款、转账、销户等核心功能。在这个项目中我们采用采用单向链表数据结构动态管理账户数据，以便于对账户信息进行增删改查。系统通过文件读写实现数据持久化，确保账户信息的安全存储与及时恢复。与此同时我们对该系统进行了测试，测试结果表明，我们设计的系统不仅可以实现题目要求的基本功能，而且能对各种异常情况进行处理，便于用户的操作并且提升体验感。

1、设计思想

系统以链表作为主要的数据结构，实现动态增删查改，能够动态管理客户账户信息，确保操作的高效性与灵活性；管理员与客户权限分离，文件读写保证数据持久化，实现数据的安全操作与存储。

2、类结构



账户类：相当于链表的节点，用于存储账户的基本信息，同时存储指针指向链表的下一个节点。

客户类：定义客户对银行账户的各种操作，并且存放账户数量，存款总额等重要信息。

管理员类：定义管理员对银行账户的各种操作。

3、主要数据结构

数据结构选择分析:

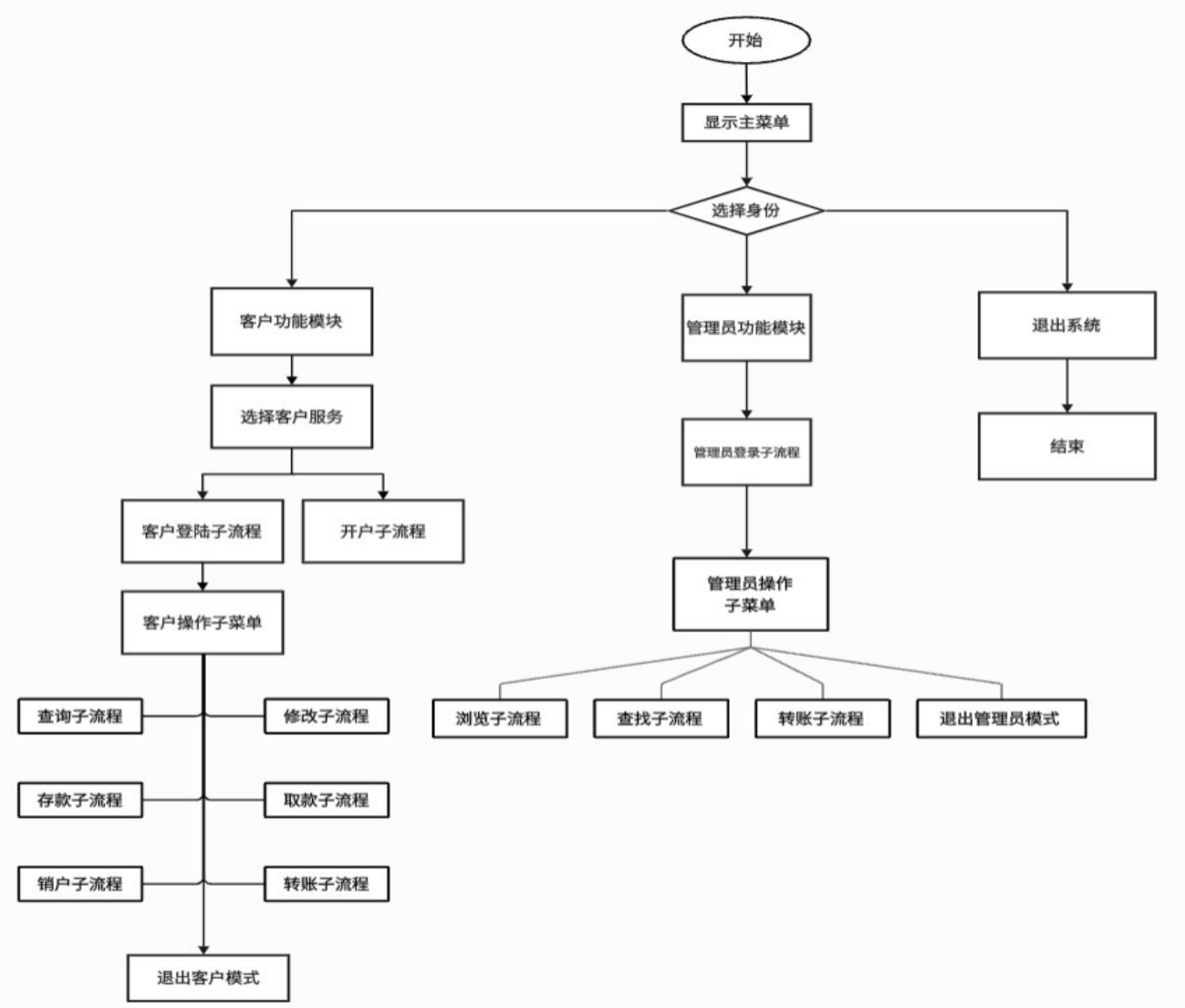
本系统采用**单向链表**存储账户数据，主要基于以下考虑：

1. **动态管理**：银行账户数量可能频繁变化，链表支持高效的动态增删操作。使用链表操作可以存储更多的银行账户，不会像数组存在账户数量限制。
2. **内存利用率高**：链表无需提前分配内存，避免数组预分配可能造成的空间浪费。
3. **实现简单**：链表结构易于实现和维护，便于对各种账户信息进行增删改查。

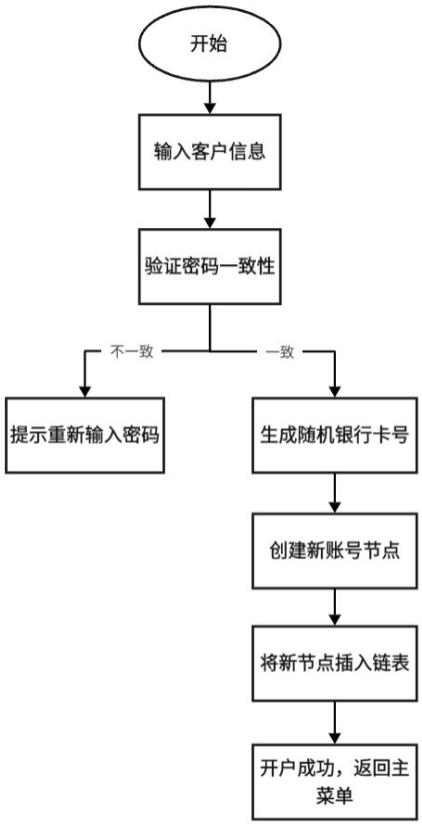
权衡分析：

链表的查询效率为 O(n)，但由于银行账户系统对查询的实时性要求不高，且可通过优化遍历方式（如记录高频访问节点）提升性能，因此链表仍是最优选择。

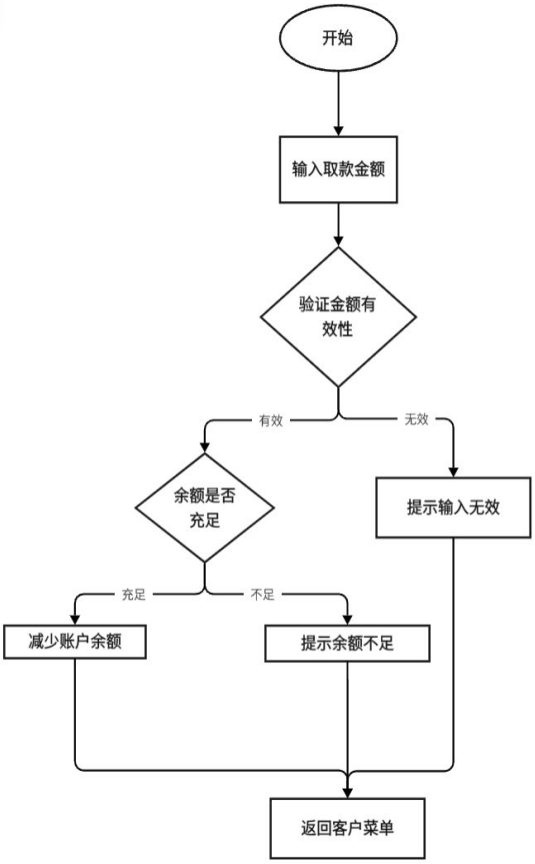
4、算法设计



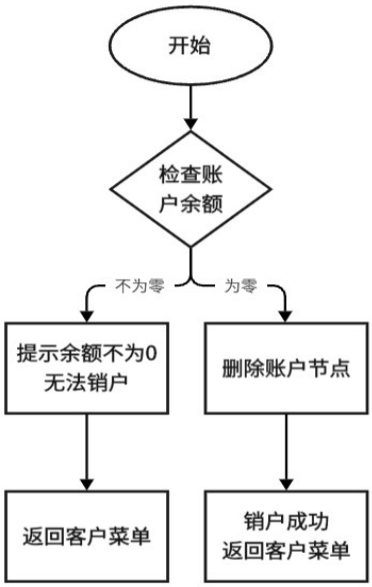
主代码实现银行账户管理系统的主要操作流程，定义了系统的显示界面以及操作方式，并在每一步操作都给予用户充分的提示信息便于用户操作。



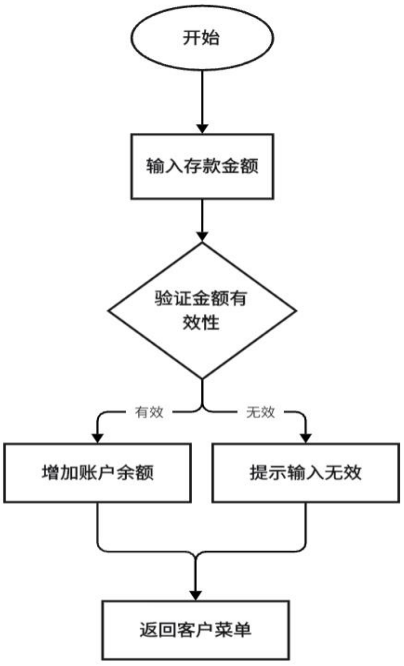
开户时先输入 ID 等账户信息，并检验 ID 的唯一性，与此同时在设置密码时要求用户两次输入密码并验证一致性，增强系统安全性。然后将客户信息作为链表节点插入到表头节点之后完成开户操作。



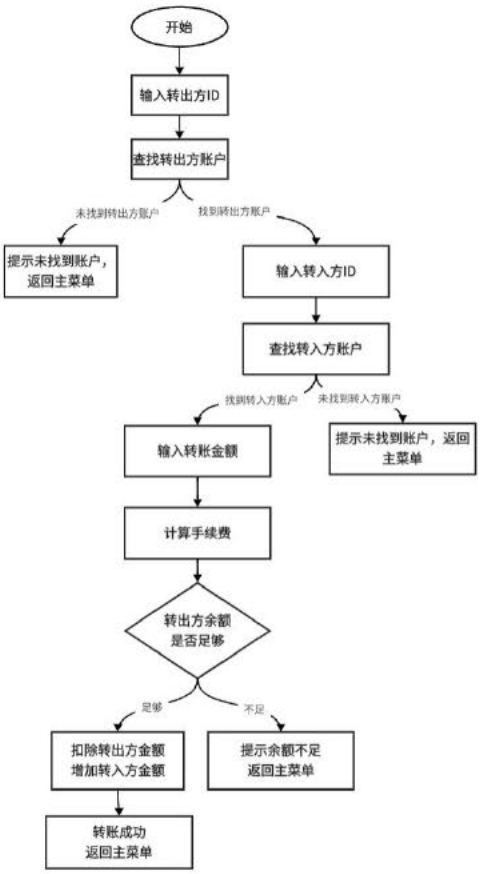
取款时先验证输入金额的有效性，然后检测账户余额是否充足，充足则减少账户余额，反之则提示余额不足。



选择销户后，系统先检查账户内是否还有余额，没有则销户，反之提示用户销户失败。

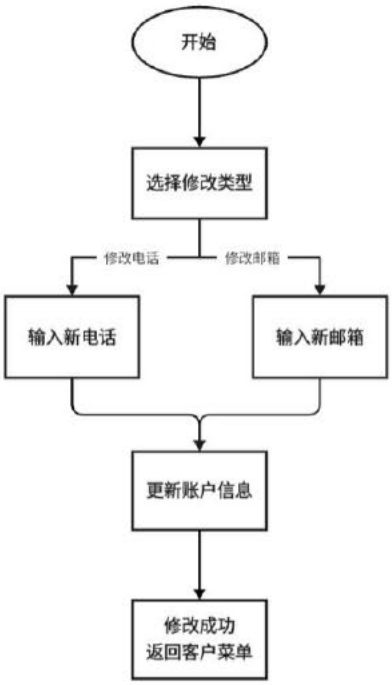


登陆成功后，可选择存款服务，输入金额后验证有效性，成功后则增加账户余额，否则提示用户重新输入。

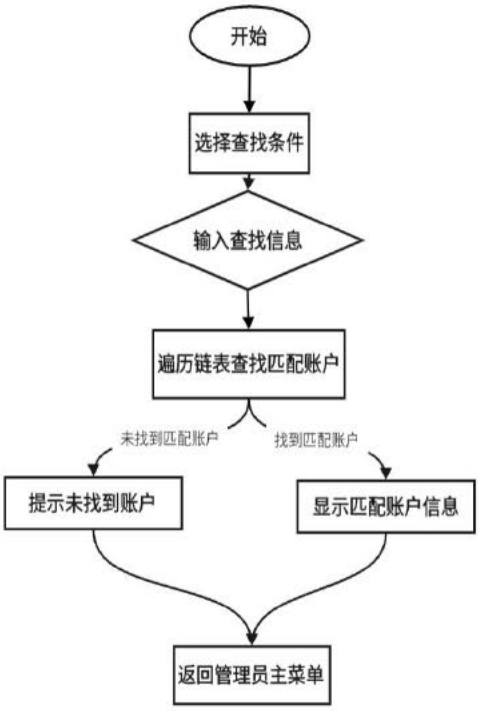


客户与管理员均可进行转账操作。其中客户需要先输入对方银行卡号，若不存在则提示用户不存在。接下来输入转账金额并扣除手续费，双方的存款金额均发生相应变化。管理员转账则需输入转入方和转出方的银行卡号，其余操作与客户一致。

通过遍历链表进行账户信息输出，同时统计客户数量和存款总数。



修改仅能修改账户的电话和邮箱，修改之后系统会自动更新账户的信息并保存。



选择查找条件，然后遍历链表查找账户，成功找到就显现账户信息，反之就提示未找到账户。

5、核心代码展示

开户功能（链表插入）：将输入的客户信息作为一个节点插入链表的表头节点后，对用户的每一步操作进行提示，同时纠正用户的错误操作。

```
#include"Clients.h"
#include <random>
//开户
void Clients::openaccounts()
{
	system("cls");
	string name0;
	string password01;
	string password02;
	string id0;
	string email0;
	string telephone0;
	string idcard0;
	string account_number = "";
	cout << "请输入您的姓名：" << endl;
	cin >> name0;
	for (;;){
		cout << "请输入您的密码：" << endl;
		cin >> password01;
		cout << "请再次输入您的密码：" << endl;
		cin >> password02;
		if (password01 != password02){
			cout << "两次输入密码不一致，请重新输入：" << endl;
		}
		else break;
	}
	cout << "请输入您的 ID 号码：" << endl;
	cin >> id0;
	while (find(id0)) {
		cout << "该 ID 号已被占用，请重新输入您的 ID" << endl;
		id0.clear();
		cin >> id0;
	}
	cout << "请输入您的电子邮箱：" << endl;
	cin >> email0;
	cout << "请输入您的电话号码：" << endl;
	cin >> telephone0;
	cout << "请输入您的身份证号：" << endl;
	cin >> idcard0;
	random_device rd; // 随机设备产生种子
	mt19937 gen(rd()); // 梅森旋转引擎
	// 2. 均匀分布的整数
```

```
43.         uniform_int_distribution<> dist_int(0, 9);
44.         for (int i = 0; i < 16; i++){
45.                     account_number += '0' + dist_int(gen);
46.         }
47.         cout << "您的银行卡号是：" << account_number << endl;
48.         /*创建新账户的结点*/
49.         BankAccount* p = new BankAccount(id0, password01, name0, telephone0, account_number, email0, idcard0, 0.0);
50.         p->next = head->next;
51.         head->next = p;
52.         length++;
53.         number++;
54.         cout << "开户成功！" << endl;
55. }
56.
```

登录查找（链表查找）:通过遍历链表找到对应账户节点。

```
1. #include"Clients.h"
2. bool Clients::find(string id)
3. {
4.         BankAccount* pf;
5.         pf = head;
6.         while (pf->ID != id && pf->next != NULL){
7.                     pf = pf->next;
8.         }
9.         if (pf->ID == id)
10.                    return true;
11.        else
12.                    return false;
13. }
14.
```

销户（链表删除）：将一个账户节点从链表中删除，并对销户成功与否进行提示。

```
1. #include"Clients.h"
2. /*销户*/
3. void Clients::cancelaccount(BankAccount* q) {
4.         system("cls");
5.         BankAccount* pt = head;
6.         if (q->balance != 0)
7.                     cout << "余额不为零，无法办理销户！" << endl;
8.         /*删除用户 q 所在结点*/
9.         else {
10.                    while (pt->next != q && pt != NULL) {
11.                                pt = pt->next;
12.                    }
13.                    if (pt == NULL) {
14.                                cout << "系统错误！" << endl;
15.                    }
```

```
16.                     if (pt->next == q) {
17.                             pt->next = q->next;
18.                             delete q;
19.                             cout << "销户成功！" << "\n";
20.                     }
21.         }
22.         length--;
23.         number--;
24. }
25.
```

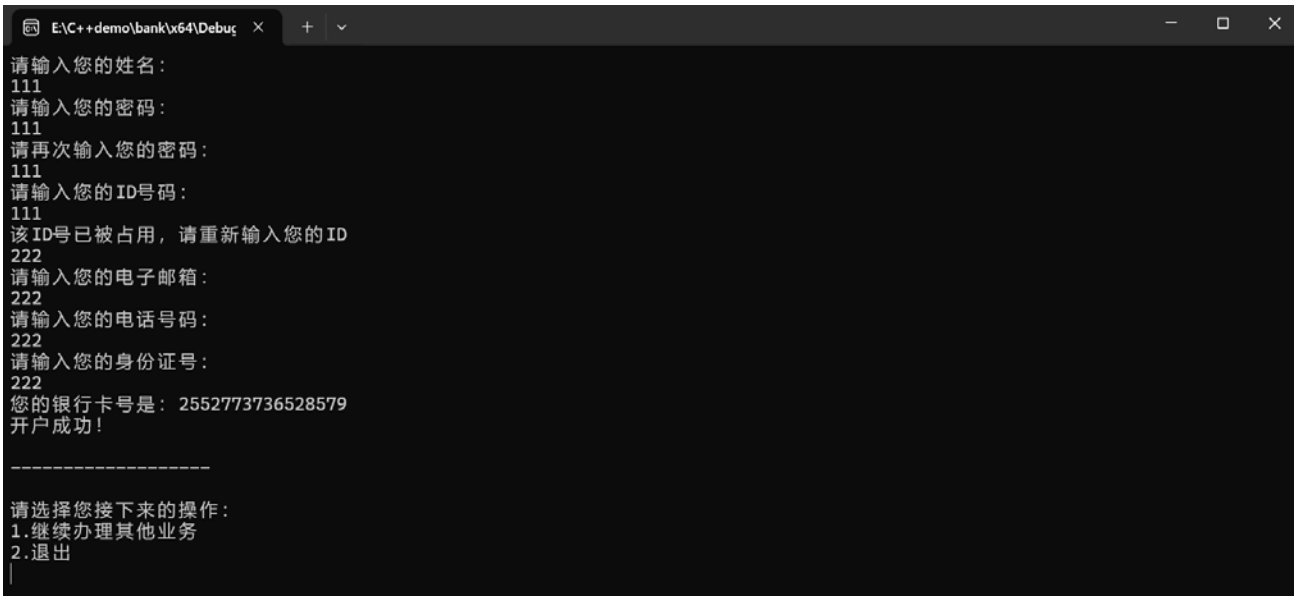
三、测试与结论

1、测试环境与数据

环境：Windows 11，Visual Studio 2022
数据：人为模拟客户和管理员，生成多条账户记录。

2、测试用例

（1）客户开户



输入必要信息进行开户，银行卡号随机生成

（2）查询

使用卡号和密码登录账户后可以查看该客户所有信息。

（3）修改

登录账户后可以选择修改电话和电子邮箱，直接输入新电话或电子邮箱就可修改成功。

（4）存款

登录账户，输入存款数字即可存款

（5）取款

登录账户，此时账户中必须有一定数量存款方可取出，否则无效；取款失败可以选择重新输入数额或退出

（6）销户

若是账户中有余额，则无法销户

无余额则销户成功

（7）浏览

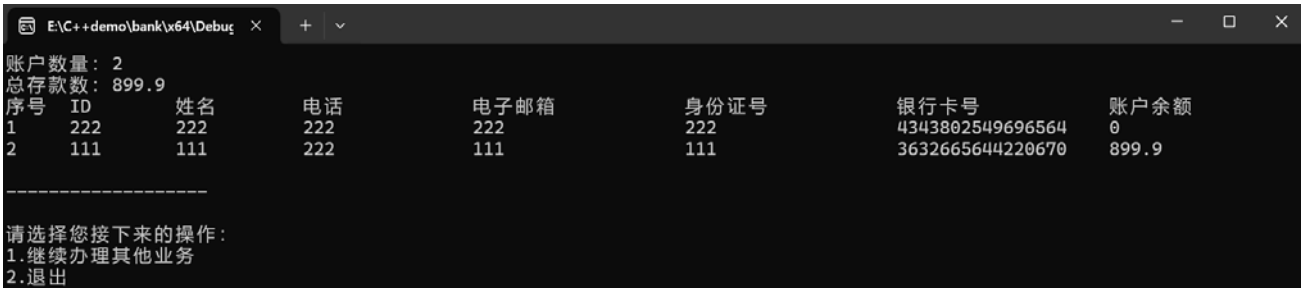
没有账户时输出

有账户时输出，可见账户数量、总存款数、所有账户信息

（8）查找

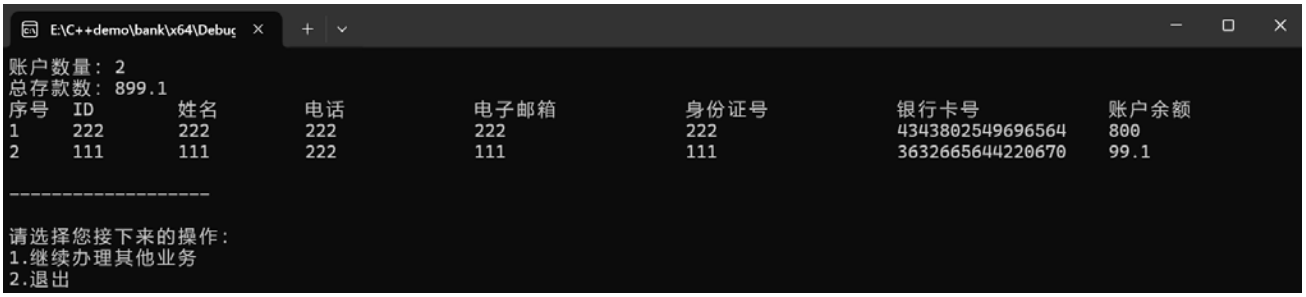
这是通过 ID 遍历链表进行查找，还能通过姓名，电话等进行查找，输出账户所有信息

（9）转账



转账前的账户信息

进行转账并扣除手续费



转账后的账户信息，成功转账 800 元

（10）文件的导入更新

文件账户数据以 txt 格式保存

3、测试结论

程序整体上满足题目需求，十项功能全部可以实现。设计的银行账户管理系统有着清晰的操作流程，对用户的每一步操作都有详细的提示，能够对出现的各种异常情况进行处理，有助于提升用户的使用体验感。系统实现了文件的读写操作，有助于账户信息的持久化保存，避免了数据丢失的问题。

四、 总结与思考

1、题目难点要点

1. 链表动态管理账户的增删查改，需要严谨的指针管理，稍有不慎会导致内存泄漏或指针指向错误等问题从而导致系统崩溃。、
2. 设计系统时要充分考虑用户的使用体验感，为用户的每一步操作提供充分的提示信息，并且可以防止由于误操作导致的错误。
3. 文件读写必须与内存数据实时同步，否则可能导致数据丢失。
4. 客户和管理员的操作权限需严格区分（如管理员可浏览所有账户）。

2、本组工作特点

本组设计了一个银行账户管理系统，实现了开户、查询、存款、取款、转账、销户等核心功能，并通过对文件的读写操作提供了完整的数据备份与恢复功能。本组的工作主要有以下几个特点：

模块化增强

将客户(Clients)和管理员(Manager)功能分离，通过不同类实现，代码结构更清晰。使用 client.writein() 进行银行账户的文件读入以初始化数据，避免主函数臃肿。

用户交互优化

我们设计的银行账户管理系统对用户的每一步操作都有充分的提示信息，并且对用户的误操作进行及时的纠正，从而提升了用户的体验感。与此同时，增加输入校验（如 while (identity < 1 || identity>3)），防止无效输入导致程序崩溃。通过 system("cls") 清屏以提升界面整洁度。

数据持久化

在退出时调用 manager.reflesh(client.head) 保存数据到文件，避免数据丢失，提升管

理系统的安全性。客户登录失败后能安全退出（if (p == NULL) break），避免空指针异常。

这些特点使得输入校验和错误处理更加完善，模块化设计增强了程序的可拓展性，便于后续新增功能，连贯的业务流程更加接近真实网银系统。

3、本组改进方向

1. 程序编写时可能未考虑到随机生成的 16 位卡号是否会出现相同的情况。
2. 账户安全性有待增强：输入的密码为明文显示，且没有登录失败的次数限制。改进思路可以将密码输入隐藏显示（如****），连续输入错误密码后锁定账户
3. 编写的管理系统的各种操作仍通过命令行窗口实现，可考虑使用图形界面进一步简化用户的操作。