|  |
| --- |
| 姓名：赵诗文  班级：21104502  学号：2110450209 |

|  |
| --- |
| 程序设计基础 |
| **课程设计报告** |
| 个人收支管理系统 |

目录

[个人收支管理 1](#_Toc106198442)

[1 设 计 目 的 1](#_Toc106198443)

[2 功 能 描 述 1](#_Toc106198444)

[2.1 初始化账户模块 2](#_Toc106198445)

[2.2 系统主界面显示模块 2](#_Toc106198446)

[2.3 退出系统模块 2](#_Toc106198447)

[3 总 体 设 计 2](#_Toc106198448)

[3.1 功能模块设计 2](#_Toc106198449)

[3.1.1 程序执行主流程 2](#_Toc106198450)

[3.1.2 系统初始化账户 3](#_Toc106198451)

[3.1.3 系统收支管理 4](#_Toc106198452)

[3.1.4 系统退出程序 4](#_Toc106198453)

[3.2 数据结构设计 4](#_Toc106198454)

[3.2.1 Date日期结构体 4](#_Toc106198455)

[3.2.2 Record收支记录结构体 4](#_Toc106198456)

[3.2.3 Account账户结构体 4](#_Toc106198457)

[3.2.4 全局变量 5](#_Toc106198458)

[3.3 函数功能设计 5](#_Toc106198459)

[3.3.1 initializeAccount() 5](#_Toc106198460)

[3.3.2 showMenu() 5](#_Toc106198461)

[3.3.3 sortRecord() 5](#_Toc106198462)

[3.3.4 calculateBalance() 5](#_Toc106198463)

[3.3.5 addRecord() 6](#_Toc106198464)

[3.3.6 showRecord() 6](#_Toc106198465)

[3.3.7 amendRecord() 6](#_Toc106198466)

[3.3.8 deleteRecord() 6](#_Toc106198467)

[3.3.9 countRecord() 6](#_Toc106198468)

[3.3.10 cleanRecord() 6](#_Toc106198469)

[4 程 序 实 现 7](#_Toc106198470)

[4.1 源码分析 7](#_Toc106198471)

[4.1.1 程序预处理 7](#_Toc106198472)

[4.1.2 主函数main() 8](#_Toc106198473)

[4.1.3 账户初始化函数 9](#_Toc106198474)

[4.1.4 主界面显示函数 11](#_Toc106198475)

[4.1.5 记录按时间排序函数 11](#_Toc106198476)

[4.1.6 余额计算函数 12](#_Toc106198477)

[4.1.7 收支记录函数 12](#_Toc106198478)

[4.1.8 查看明细函数 14](#_Toc106198479)

[4.1.9 修改记录函数 15](#_Toc106198480)

[4.1.10 删除记录函数 17](#_Toc106198481)

[4.1.11 查询统计记录函数 17](#_Toc106198482)

[4.1.12 清空记录函数 21](#_Toc106198483)

[4.2 运行结果 22](#_Toc106198484)

[4.2.1 系统初始状体 22](#_Toc106198485)

[4.2.2 主界面显示状态 23](#_Toc106198486)

[4.2.3 记录收支状态 23](#_Toc106198487)

[4.2.4 查看明细状态 24](#_Toc106198488)

[4.2.5 修改记录状态 24](#_Toc106198489)

[4.2.6 删除记录状态 25](#_Toc106198490)

[4.2.7 记录统计状态 26](#_Toc106198491)

[4.2.8 清空记录状态 27](#_Toc106198492)

[4.2.9 退出系统状态 27](#_Toc106198493)

[5 课 程 设 计 总 结 27](#_Toc106198494)

个人收支管理

收支管理就是把一个企事业单位或者个人家庭发生的所有经济业务运用一定的记账方法在账簿上记录，必要时对记录进行查询统计等管理，方便理清收入、支出的具体情况。

旧石器时代的人们通常以坚硬的石器作为刻划的工具，在石片、骨片等载体之上刻划出一排排单线条的浅纹道，或者是在树木或木板上刻出若干重复的缺口，形成通常只有刻划者自己可以体会出来的代表一定数量的标记，或是记载某种事物的标记。

原始社会末期的人们通过结绳记数的方式对经济事项进行计量、记录。

从商朝开始，人们开始使用文字记账。文字叙述记录法不拘一格，并没有固定记录格式。这种方法一直延续到现在，仍会有人选择手写账本的方式来记录收支。随着科技的发展，不同风格的电子账本层出不穷。它们的出现使人们在手机上可以随时随地的记账，大大地改变了人们的记账方式。

# 设 计 目 的

采用面向过程的程序设计方法，设计一个能够记录、修改、删除、查询、统计收支等功能的个人收支管理（以下简称：系统）程序，使用户管理收支更加高效便捷。

# 功 能 描 述

如图1所示，本系统由以下4个大功能模块组成。



图1 个人收支管理系统四大功能模块图

## 初始化账户模块

本系统是基于一个设置好的账户进行工作的。用户在正式使用本系统之前，需要对账户做初始化工作，包括给账户命名，输入账户无记录时的余额以及修改账户记录币种。

## 系统主界面显示模块

此模块的作用是向用户展示程序的具体功能，有记录收支、查看明细、修改记录、删除记录、查询统计、清空记录。每一个具体的功能都有一个代码。用户在进行具体功能选择时，按照系统主界面显示的提示输入代码，即可进入相对应的功能区域。

## 退出系统模块

系统退出系统模块主要执行退出系统操作，用户无法再次选择使用具体功能模块，随后退出程序。

# 总 体 设 计

## 功能模块设计

### 程序执行主流程

此个人收支管理系统执行主流程图如图2所示。

首先，程序调用initializeAccount()函数，使系统进入初始化账户模块。然后调用showMenu()函数，显示系统主界面。在用户输入具体功能编号之前，程序还会调用sortRecord()函数和calculateBalance()函数，对当前记录进行按时间排序和计算余额操作，以起到更新记录的作用。用户输入具体功能编号之后，程序就会按编号所对应的具体模块调用函数：1，记录收支addRecord()函数；2，查看明细showRecord()函数；3，修改记录amendRecord()函数；4，删除记录deleteRecord()函数；5，记录统计countRecord()函数；6，清空记录cleanRecord()函数。在正式进入修改记录和删除记录模块前，程序还会再调用一次查看明细showRecord()函数，方便用户做出要修改和删除的记录选择。在进入记录统计模块后，系统还为用户提供三个具体功能模块，包括输入起止日期查出这段时间内的收支明细、按年统计总收支、按月统计总收支。以上六个收支管理具体功能模块使用完毕后，程序都会回到主界面显示模块，供用户选择下次使用的功能。当在输入编号时，输入“0”，程序会执行退出系统模块。

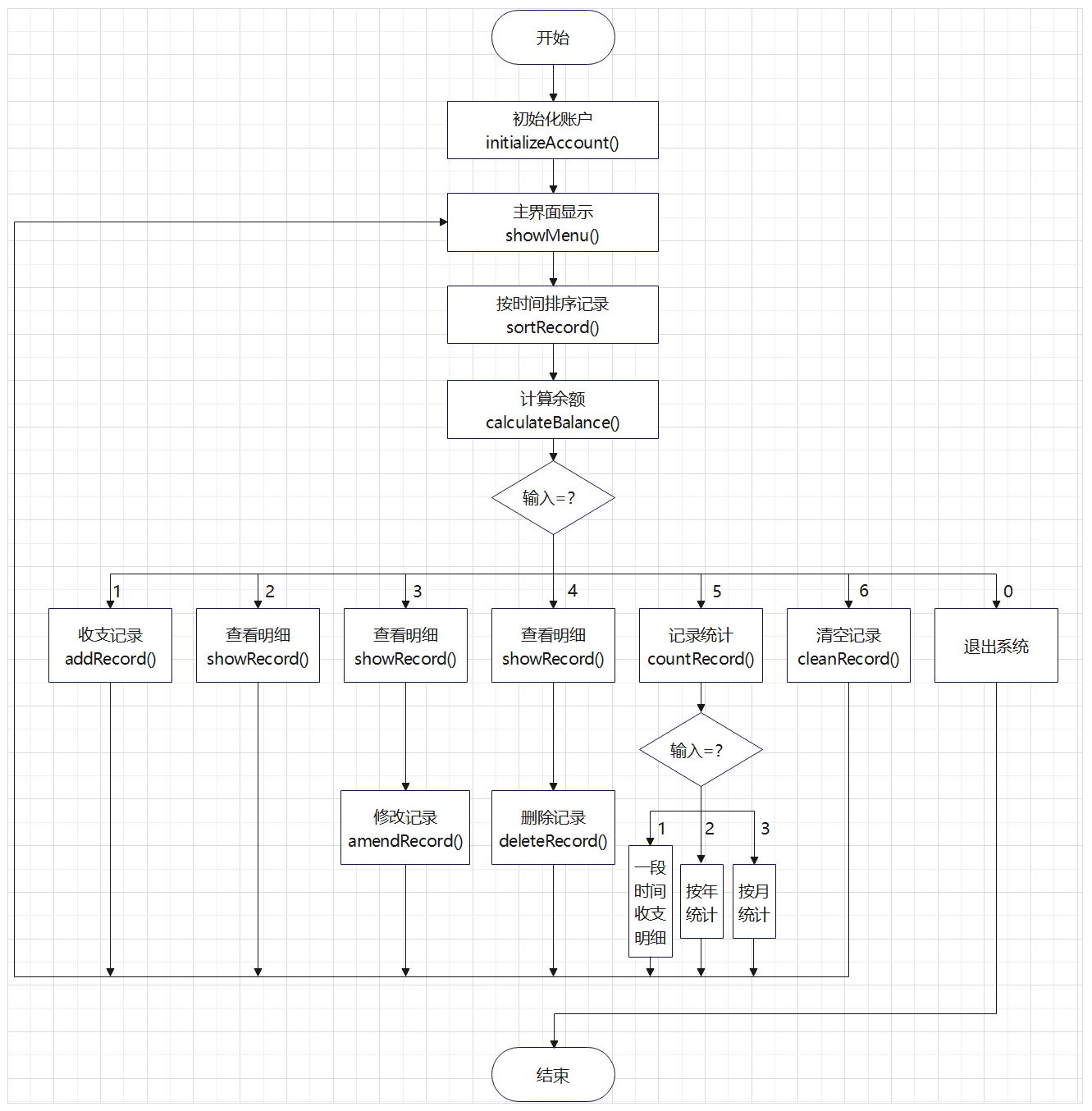


图2 程序执行主流程

### 系统初始化账户

用户进入初始化账户模块后，需按提示输入账户名称、账户余额，并选择是否更改币种（默认币种是人民币）。

### 系统收支管理

用户进入系统收支管理模块后，可选择六个具体功能模块。输入六个具体功能模块代码后，程序会调用相对应的函数。在正式进入修改记录和删除记录模块前，程序还会再调用一次查看明细showRecord()函数，方便用户做出要修改和删除的记录选择。

### 系统退出程序

此模块较为简单，即退出main()即可。

## 数据结构设计

此程序中，定义了三个结构体。

### Date日期结构体

struct Date {

int m\_Year;//年

int m\_Month;//月

int m\_Day;//日

};

定义日期结构体，方便管理。

### Record收支记录结构体

struct Record {

Date m\_Time;//日期

string m\_Remark;//备注

string m\_Variety;//收入或支出

double m\_InMoney;//收入金额

double m\_OutMoney;//支出金额

double m\_Balance;//余额

};

因为每一条收支记录中包含不同数据类型的值，所以定义一个收支记录结构体即可。

### Account账户结构体

struct Account {

string m\_Name;//账户名称

string m\_Currency;//币种

double m\_Balance;//账户余额

struct Record AccountBill[MAX];//收支记录数组

int m\_Size;//账户记录条数

};

定义一个账户结构体，其中包含了账户的属性和收支记录的数组。

### 全局变量

创建A1账户结构体变量和为用户提供输入的整形select变量。

## 函数功能设计

### initializeAccount()

函数原型：void initializeAccount(Account\*)

initializeAccount()函数用于初始化账户，完成用户对账户命名、输入余额、修改币种的操作。

### showMenu()

函数原型：void showMenu()

showMenu()函数用于显示系统主界面。

### sortRecord()

函数原型：void sortRecord(Account\*)

sortRecord()函数用于对已存在的记录进行按时间排序操作。

### calculateBalance()

函数原型：void calculateBalance(Account\*)

calculateBalance()函数用于对已按时间排好的记录计算其余额。

### addRecord()

函数原型：void addRecord(Account\*)

addRecord()函数用于实现增加收支记录功能。

### showRecord()

函数原型：void showRecord(Account\*)

showRecord()函数用于实现显示记录，即查看明细表功能。

### amendRecord()

函数原型：void amendRecord(Account\*)

amendRecord()函数用于实现修改收支记录功能。

### deleteRecord()

函数原型：void deleteRecord (Account\*)

deleteRecord()函数用于实现删除收支记录功能。

### countRecord()

函数原型：void countRecord (Account\*)

countRecord()函数用于实现查询与统计收支记录功能。

### cleanRecord()

函数原型：void cleanRecord (Account\*)

cleanRecord()函数用于实现清空所有记录功能。

# 程 序 实 现

## 源码分析

### 程序预处理

包括加载头文件，定义结构体、宏常量，声明函数。

#include <iostream>

#include <string>//用到string数据类型

using namespace std;

#define MAX 100//宏常量，用于定义账单最大容量，可更改

//声明日期结构体

struct Date {

int m\_Year;//年

int m\_Month;//月

int m\_Day;//日

};

//声明收支记录结构体

struct Record {

Date m\_Time;//日期

string m\_Remark;//备注

string m\_Variety;//收入或支出

double m\_InMoney;//收入金额

double m\_OutMoney;//支出金额

double m\_Balance;//余额

};

//设计账户结构体

struct Account {

string m\_Name;//账户名称

string m\_Currency;//币种

double m\_Balance;//账户余额

struct Record AccountBill[MAX];//收支记录数组

int m\_Size;//账户记录条数

};

void initializeAccount(Account\*);//初始化账户函数

void showMenu();//主界面显示函数

void sortRecord(Account\*);//记录按时间排序函数

void calculateBalance(Account\*);//余额计算函数

void addRecord(Account\*);//记录收支函数

void showRecord(Account\*);//查看明细函数

void amendRecord(Account\*);//修改记录函数

void deleteRecord(Account\*);//删除记录函数

void countRecord(Account\*);//查询统计记录函数

void cleanRecord(Account\*);//清空记录函数

### 主函数main()

main()函数主要实现了对整个程序的运行控制，以及相关功能模块的调用。

int main() {

Account A1;//创建账户

initializeAccount(&A1);//调用初始化账户函数

A1.m\_Size = 0;//初始化账户账单个数

int select = 0;//创建用户输入变量

while (true) {//用while循环检测输入是否合法

showMenu();//调用主界面显示函数

sortRecord(&A1);//调用记录按时间排序函数

calculateBalance(&A1);//调用余额计算函数

cout << "账户名称：" << A1.m\_Name << "\t";//显示账户名称

//显示账户余额

if (A1.m\_Size == 0)cout << "当前账户余额：" << A1.m\_Balance << endl;

else cout << "账户余额：" << A1.AccountBill[A1.m\_Size - 1].m\_Balance << endl;

cout << "#请选择功能序号（1/2/3/4/5/6/0）：";

cin >> select;//用户输入功能序号

switch (select) {

case 1://1.记录收支

addRecord(&A1);//调用记录收支函数

break;

case 2://2.查看明细

showRecord(&A1);//调用查看明细函数

system("pause");//请按任意键继续

system("cls");//清屏操作

break;

case 3://3.修改记录

cout << "#已进入“修改记录”功能！" << endl;

showRecord(&A1);//调用查看明细函数

amendRecord(&A1);//调用修改记录函数

break;

case 4://4.删除记录

cout << "#已进入“删除记录”功能！" << endl;

showRecord(&A1);//调用查看明细函数

deleteRecord(&A1);//调用删除记录函数

break;

case 5://5.记录统计

countRecord(&A1);//调用记录统计函数

break;

case 6://6.清空记录

cleanRecord(&A1);//调用清空记录函数

break;

case 0://0.退出系统

cout << "#欢迎下次使用！" << endl;

system("pause");

return 0;

break;

default:

system("cls");//清屏操作

cout << "#您输入有误！" << endl;

break;

}

}

system("pause");

return 0;

}

### 账户初始化函数

void initializeAccount(Account\* A) {

string name;//创建局部变量账户名字

double balance;//创建局部变量余额

int Currency = 0;//创建局部变量币种代号

cout << "欢迎使用个人收支管理系统！" << endl;

cout << "请先初始化账户：" << endl;

cout << "#请输入账户名称：";

cin >> name;

A->m\_Name = name;//将用户输入的局部变量的值赋值给真正的值

cout << "#请输入账户余额：";

cin >> balance;

A->m\_Balance = balance;//将用户输入的局部变量账户余额赋值给真正的账户余额

cout << "!提示：" << endl;

cout << " 账本默认记录币种是人民币CNY！" << endl;

cout << " 如需更改请输入1，不更改请输入0！" << endl;

cout << "#是否更改（1/0)：";

while (true) {//用while循环检测输入是否合法

cin >> Currency;

if (Currency == 0) {

A->m\_Currency = "CNY";//将用户输入的局部变量的值赋值给真正的值

break;

}

else if (Currency == 1) {//修改账户币种

int selectCurrency = 0;

cout << "#选择币种" << endl;

cout << " 人民币CNY - 0" << endl;

cout << " 美元USD ----1" << endl;

cout << " 欧元EUR --- 2" << endl;

cout << " 日元JPY ----3" << endl;

cout << " 英镑GPB ----4" << endl;

while (true) {//用while循环检测输入是否合法

cout << "请选择（0/1/2/3/4）：";

cin >> selectCurrency;

if (selectCurrency == 0) {

A->m\_Currency = "CNY";

break;

}

if (selectCurrency == 1) {

A->m\_Currency = "USD";

break;

}

if (selectCurrency == 2) {

A->m\_Currency = "EUR";

break;

}

if (selectCurrency == 3) {

A->m\_Currency = "JPY";

break;

}

if (selectCurrency == 4) {

A->m\_Currency = "GPB";

break;

}

cout << "输入不合法！";

}

break;

}

cout << "输入不合法！是否更改（1/0)：";

}

cout << "#初始化账户成功！" << endl;

cout << "“" << A->m\_Name << "”" << "账户当前余额为："

<< A->m\_Currency <<" "<< A->m\_Balance << endl;

system("pause");//请按任意键继续

system("cls");//清屏操作

}

### 主界面显示函数

void showMenu() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* \*" << endl;

cout << "\* 个人收支管理系统 \*" << endl;

cout << "\* \*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 1.记录收支 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 2.查看明细 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 3.修改记录 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 4.删除记录 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 5.查询统计 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 6.清空记录 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\* 0.退出系统 \*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl<< endl;

}

### 记录按时间排序函数

void sortRecord(Account\* A) {

for (int i = 0; i < A->m\_Size - 1; i++) {

for (int j = 0; j < A->m\_Size - i - 1; j++) {

if (A->AccountBill[j].m\_Time.m\_Year > A->AccountBill[j + 1].m\_Time.m\_Year) {

Record temp = A->AccountBill[j];

A->AccountBill[j] = A->AccountBill[j + 1];

A->AccountBill[j + 1] = temp;

}

}

}

for (int k = 0; k < A->m\_Size - 1; k++) {

if (A->AccountBill[k].m\_Time.m\_Year == A->AccountBill[k + 1].m\_Time.m\_Year) {

if (A->AccountBill[k].m\_Time.m\_Month > A->AccountBill[k + 1].m\_Time.m\_Month) {

Record temp = A->AccountBill[k];

A->AccountBill[k] = A->AccountBill[k + 1];

A->AccountBill[k + 1] = temp;

}

}

}

for (int m = 0; m < A->m\_Size - 1; m++) {

if (A->AccountBill[m].m\_Time.m\_Year == A->AccountBill[m + 1].m\_Time.m\_Year) {

if (A->AccountBill[m].m\_Time.m\_Month == A->AccountBill[m + 1].m\_Time.m\_Month) {

if (A->AccountBill[m].m\_Time.m\_Day > A->AccountBill[m + 1].m\_Time.m\_Day) {

Record temp = A->AccountBill[m];

A->AccountBill[m] = A->AccountBill[m + 1];

A->AccountBill[m + 1] = temp;

}

}

}

}

}

### 余额计算函数

void calculateBalance(Account\* A) {

double C = 0;

if (A->AccountBill[0].m\_Variety == "+") {

C = A->m\_Balance + A->AccountBill[0].m\_InMoney;

A->AccountBill[0].m\_Balance = C;

}

else {

C = A->m\_Balance - A->AccountBill[0].m\_OutMoney;

A->AccountBill[0].m\_Balance = C;

}

for (int i = 1; i < A->m\_Size; i++) {

if (A->AccountBill[i].m\_Variety == "+") {

C += A->AccountBill[i].m\_InMoney;

A->AccountBill[i].m\_Balance = C;

}

else {

C -= A->AccountBill[i].m\_OutMoney;

A->AccountBill[i].m\_Balance = C;

}

}

}

### 收支记录函数

void addRecord(Account\* A) {

cout << "#已进入“记录收支”功能！" << endl;

if (A->m\_Size == MAX) {//检测账户记录是否已达到最大容量

cout << "该账户记录已经达到最大容量，无法继续添加！" << endl;

system("pause");

system("cls");

}

else {

//输入日期

Date time;//创建Date结构体类型局部变量

cout << "#请输入日期" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 年：";

cin >> time.m\_Year;

//检测日期输入是否合法

if (0 < time.m\_Year && time.m\_Year < 9999) {

A->AccountBill[A->m\_Size].m\_Time.m\_Year = time.m\_Year;//将用户输入的局部变量的值赋值给真正的值

break;

}

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 月：";

cin >> time.m\_Month;

//检测日期输入是否合法

if (0 < time.m\_Month && time.m\_Month < 13) {

A->AccountBill[A->m\_Size].m\_Time.m\_Month = time.m\_Month;//将用户输入的局部变量的值赋值给真正的值

break;

}

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 日：";

cin >> time.m\_Day;

//检测日期输入是否合法

if (0 < time.m\_Day && time.m\_Day < 32) {

A->AccountBill[A->m\_Size].m\_Time.m\_Day = time.m\_Day;//将用户输入的局部变量的值赋值给真正的值

break;

}

cout << " 输入不合法！";

}

//输入摘要

string remark;

cout << "#请输入摘要：";

cin >> remark;

A->AccountBill[A->m\_Size].m\_Remark = remark;//将用户输入的局部变量的值赋值给真正的值

//输入收入或支出标志

string variety;

cout << "#选择记录类型(收入 +/支出 -)" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 请输入（+/-）：";

cin >> variety;

if (variety == "+" || variety == "-") {

A->AccountBill[A->m\_Size].m\_Variety = variety;//将用户输入的局部变量的值赋值给真正的值

break;

}

cout << " 输入不合法！";

}

//输入金额

double money = 0;

cout << "#请输入金额：";

cin >> money;

if (A->AccountBill[A->m\_Size].m\_Variety == "+") {

A->AccountBill[A->m\_Size].m\_InMoney = money;//将用户输入的局部变量的值赋值给真正的值

A->AccountBill[A->m\_Size].m\_OutMoney = 0;//收入模式，支出为0

}

else {

A->AccountBill[A->m\_Size].m\_InMoney = 0;//支出模式，收入为0

A->AccountBill[A->m\_Size].m\_OutMoney = money;//将用户输入的局部变量的值赋值给真正的值

}

//更新账单

A->m\_Size++;

cout << "#添加账单成功！" << endl;

system("pause");//请按任意键继续

system("cls");//清屏操作

}

}

### 查看明细函数

void showRecord(Account\* A) {

if (A->m\_Size == 0) {//提示记录为0

cout << "#当前记录为空！" << endl;

}

else {

for (int i = 0; i < A->m\_Size; i++) {

cout << i+1 << "\t";

cout << A->AccountBill[i].m\_Time.m\_Year << "年";

A->AccountBill[i].m\_Time.m\_Month > 0 && A->AccountBill[i].m\_Time.m\_Month < 10 ?

cout << "0" << A->AccountBill[i].m\_Time.m\_Month : cout << A->AccountBill[i].m\_Time.m\_Month; cout << "月";

A->AccountBill[i].m\_Time.m\_Day > 0 && A->AccountBill[i].m\_Time.m\_Day < 10 ?

cout << "0" << A->AccountBill[i].m\_Time.m\_Day : cout << A->AccountBill[i].m\_Time.m\_Day; cout << "日" << "\t";

cout << " " << A->m\_Currency << " ";

A->AccountBill[i].m\_Variety == "+" ?

cout << " " << A->AccountBill[i].m\_InMoney << "\t": cout << "-" << A->AccountBill[i].m\_OutMoney << "\t";

cout << "余额：" << A->m\_Currency << " " << A->AccountBill[i].m\_Balance << "\t";

cout << "摘要：" << A->AccountBill[i].m\_Remark << endl;

}

}

}

### 修改记录函数

void amendRecord(Account\* A) {

if (A->m\_Size == 0) {//提示记录为0

cout << "#无可修改账单！" << endl;

}

else {

int amendNumber;

while (true) {

cout << "#请输入要修改的记录的序号：";

cin >> amendNumber;

if (amendNumber > A->m\_Size) {//检测查询序号是否存在

cout << "#您查询的记录不存在，请确认后重新输入！" << endl;

}

else {

Date time;

cout << "#请输入日期" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 年：";

cin >> time.m\_Year;

if (0 < time.m\_Year && time.m\_Year < 9999) {

A->AccountBill[amendNumber - 1].m\_Time.m\_Year = time.m\_Year;

break;

}

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 月：";

cin >> time.m\_Month;

if (0 < time.m\_Month && time.m\_Month < 13) {

A->AccountBill[amendNumber - 1].m\_Time.m\_Month = time.m\_Month;

break;

}

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 日：";

cin >> time.m\_Day;

if (0 < time.m\_Day && time.m\_Day < 32) {

A->AccountBill[amendNumber - 1].m\_Time.m\_Day = time.m\_Day;

break;

}

cout << " 输入不合法！";

}

//输入摘要

string remark;

cout << "#请输入摘要：";

cin >> remark;

A->AccountBill[amendNumber - 1].m\_Remark = remark;

//输入收入或支出标志

string variety;

cout << "#选择记录类型(收入 +/支出 -)" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 请输入（+/-）：";

cin >> variety;

if (variety == "+" || variety == "-") {

A->AccountBill[amendNumber - 1].m\_Variety = variety;

break;

}

cout << " 输入不合法！";

}

//输入金额

double money = 0;

cout << "#请输入金额：";

cin >> money;

if (A->AccountBill[amendNumber - 1].m\_Variety == "+") {

A->AccountBill[amendNumber - 1].m\_InMoney = money;

A->AccountBill[amendNumber - 1].m\_OutMoney = 0;

}

else {

A->AccountBill[amendNumber - 1].m\_InMoney = 0;

A->AccountBill[amendNumber - 1].m\_OutMoney = money;

}

break;

}

}

cout << "修改记录成功！" << endl;

}

system("pause");//请按任意键继续

system("cls");//清屏操作

}

### 删除记录函数

void deleteRecord(Account\* A) {

if (A->m\_Size == 0) {//提示记录为0

cout << "#无可删除账单！" << endl;

}

else {

int deleteNumber;

cout << "#请输入要删除的记录的序号：";

while (true) {//用while循环检测输入序号是否存在

cin >> deleteNumber;

if (0 < deleteNumber && deleteNumber < A->m\_Size) {

for (int i = deleteNumber - 1; i < A->m\_Size; i++)

A->AccountBill[i] = A->AccountBill[i + 1];

A->m\_Size--;

break;

}

cout << " 找不到对应序号！请确认：";

}

cout << "#删除记录成功！" << endl;

}

system("pause");//请按任意键继续

system("cls");//清屏操作

}

### 查询统计记录函数

//查询统计

void countRecord(Account\* A) {

cout << "#已进入“查询统计”功能！" << endl;

if (A->m\_Size == 0) {//提示记录为0

cout << "#当前记录为空！" << endl;

cout << "#无可查询或统计的账单！" << endl;

system("pause");

system("cls");

}

else {

int n;

cout << "#请选择具体功能：" << endl;

cout << " 输入起止日期查询该段时间的收支明细 --- 1" << endl;

cout << " 按年统计总收入和总支出 --------------- 2" << endl;

cout << " 按月统计总收入和总支出 --------------- 3" << endl;

cout << "#请选择（1/2/3）：";

while (true) {//用while循环检测输入是否有误

cin >> n;

if (n == 1) {

Date beginningTime, endTime;

int b = -1, e = A->m\_Size, k = 0;

cout << "#请输入查询开始日期" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 年：";

cin >> beginningTime.m\_Year;

if (0 < beginningTime.m\_Year && beginningTime.m\_Year < 9999)

break;

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 月：";

cin >> beginningTime.m\_Month;

if (0 < beginningTime.m\_Month && beginningTime.m\_Month < 13)

break;

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 日：";

cin >> beginningTime.m\_Day;

if (0 < beginningTime.m\_Day && beginningTime.m\_Day < 32)

break;

cout << " 输入不合法！";

}

cout << "#请输入查询结束日期" << endl;

while (true) {//用while循环检测输入是否合法

cout << " 年：";

cin >> endTime.m\_Year;

if (0 < endTime.m\_Year && endTime.m\_Year < 9999)

break;

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 月：";

cin >> endTime.m\_Month;

if (0 < endTime.m\_Month && endTime.m\_Month < 13)

break;

cout << " 输入不合法！";

}

while (true) {//用while循环检测输入是否合法

cout << " 日：";

cin >> endTime.m\_Day;

if (0 < endTime.m\_Day && endTime.m\_Day < 32)

break;

cout << " 输入不合法！";

} //日期比较 20030121<20030122<20030124

int m = beginningTime.m\_Year \* 10000 + beginningTime.m\_Month \* 100 + beginningTime.m\_Day,

n = endTime.m\_Year \* 10000 + endTime.m\_Month \* 100 + endTime.m\_Day;

for (int i = 0; i < A->m\_Size - 1; i++) {

int p = A->AccountBill[i].m\_Time.m\_Year \* 10000 + A->AccountBill[i].m\_Time.m\_Month \* 100 + A->AccountBill[i].m\_Time.m\_Day,

q = A->AccountBill[i + 1].m\_Time.m\_Year \* 10000 + A->AccountBill[i + 1].m\_Time.m\_Month \* 100 + A->AccountBill[i + 1].m\_Time.m\_Day;

if (p < m && m <= q) {

b = i + 1;

break;

}

}

for (int j = 0; j < A->m\_Size - 1; j++) {

int p = A->AccountBill[j].m\_Time.m\_Year \* 10000 + A->AccountBill[j].m\_Time.m\_Month \* 100 + A->AccountBill[j].m\_Time.m\_Day,

q = A->AccountBill[j + 1].m\_Time.m\_Year \* 10000 + A->AccountBill[j + 1].m\_Time.m\_Month \* 100 + A->AccountBill[j + 1].m\_Time.m\_Day;

if (p <= n && n < q) {

e = j;

break;

}

}

if (b == -1) {//检测是否有误

cout << "#出现错误！" << endl;

}

else {

cout << "#您从" << beginningTime.m\_Year << "年" << beginningTime.m\_Month << "月" << beginningTime.m\_Day << "日"

<< "到" << endTime.m\_Year << "年" << endTime.m\_Month << "月" << endTime.m\_Day << "日"

<< "的收支明细如下：" << endl;

for (int i = b; i <= e; i++) {

cout << k + 1 << "\t";

cout << A->AccountBill[i].m\_Time.m\_Year << "年";

A->AccountBill[i].m\_Time.m\_Month > 0 && A->AccountBill[i].m\_Time.m\_Month < 10 ?

cout << "0" << A->AccountBill[i].m\_Time.m\_Month : cout << A->AccountBill[i].m\_Time.m\_Month; cout << "月";

A->AccountBill[i].m\_Time.m\_Day > 0 && A->AccountBill[i].m\_Time.m\_Day < 10 ?

cout << "0" << A->AccountBill[i].m\_Time.m\_Day : cout << A->AccountBill[i].m\_Time.m\_Day; cout << "日" << "\t";

cout << " " << A->m\_Currency << " ";

A->AccountBill[i].m\_Variety == "+" ?

cout << " " << A->AccountBill[i].m\_InMoney << "\t" : cout << "-" << A->AccountBill[i].m\_OutMoney << "\t";

cout << "余额：" << A->m\_Currency << " " << A->AccountBill[i].m\_Balance << "\t";

cout << "摘要：" << A->AccountBill[i].m\_Remark << endl;

k++;

}

}

break;

}

if (n == 2) {

int Y;

double YsumIn = 0, YsumOut = 0;

cout << "#请输入想要查询的年份：";

cin >> Y;

for (int i = 0; i < A->m\_Size; i++) {

if (Y == A->AccountBill[i].m\_Time.m\_Year) {

YsumIn += A->AccountBill[i].m\_InMoney;

YsumOut += A->AccountBill[i].m\_OutMoney;

}

}

cout << "#您在" << Y << "年，";

cout << "总收入为:" << A->m\_Currency << " " << YsumIn << "\t";

cout << "总支出为:" << A->m\_Currency << " " << YsumOut << endl;

break;

}

if (n == 3) {

int Y, M;

double YMsumIn = 0, YMsumOut = 0;

cout << "#请输入想要查询的年份：";

cin >> Y;

cout << "#请输入" << Y << "年中想查询的月份：";

cin >> M;

for (int i = 0; i < A->m\_Size; i++) {

if (Y == A->AccountBill[i].m\_Time.m\_Year && M == A->AccountBill[i].m\_Time.m\_Month) {

YMsumIn = YMsumIn + A->AccountBill[i].m\_InMoney;

YMsumOut = YMsumOut + A->AccountBill[i].m\_OutMoney;

}

}

cout << "#您在" << Y << "年" << M << "月，" << endl;

cout << "总收入为:" << A->m\_Currency << " " << YMsumIn << "\t";

cout << "总支出为:" << A->m\_Currency << " " << YMsumOut << endl;

break;

}

cout << "#您输入有误！请重新选择（1/2/3）：";

}

system("pause");

system("cls");

}

}

### 清空记录函数

void cleanRecord(Account\* A) {

cout << "#已进入“清空记录”功能！" << endl;

if (A->m\_Size == 0) {//提示记录为0

cout << "#当前账单记录已为空！" << endl;

}

else {

string select;

cout << "您是否要清空所有记录（是-y/否-n）:" << endl;

while (true) {//用while循环检测输入是否合法

cin >> select;

if (select == "y") {

A->m\_Size = 0;//逻辑上清空

cout << "记录已清空！" << endl;

break;

}

if (select == "n")

break;

cout << "您输入有误！请重新选择（是-y/否-n）：";

}

system("pause");

system("cls");

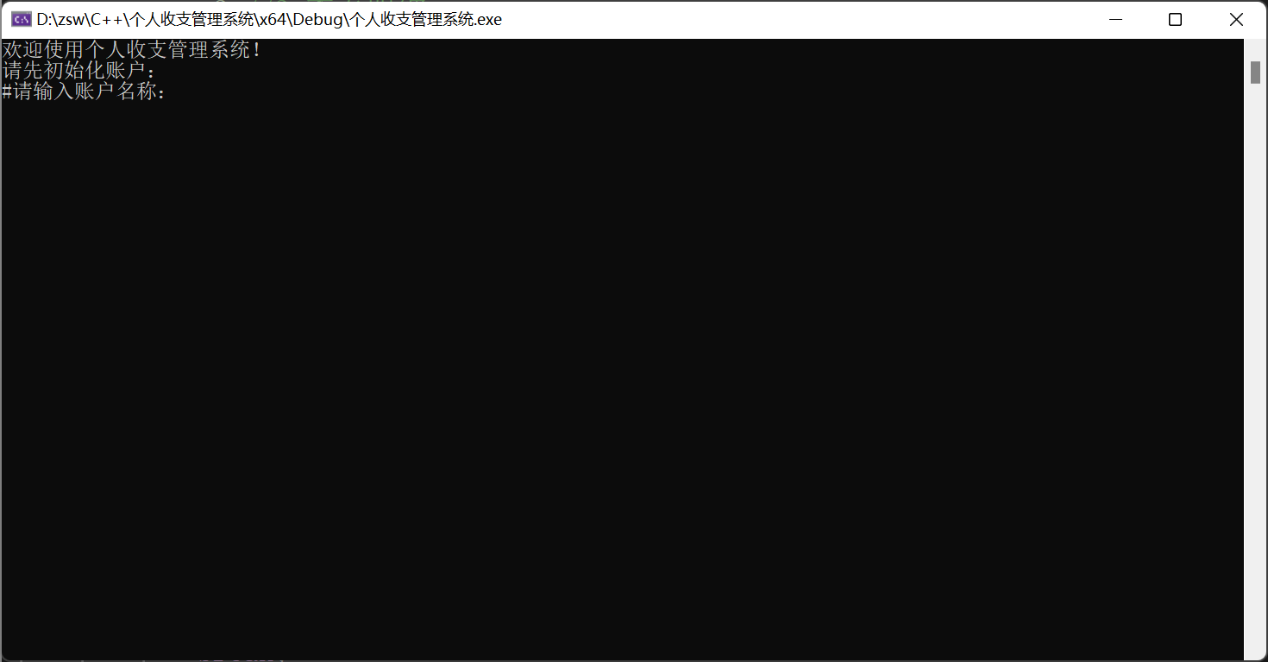
}

}

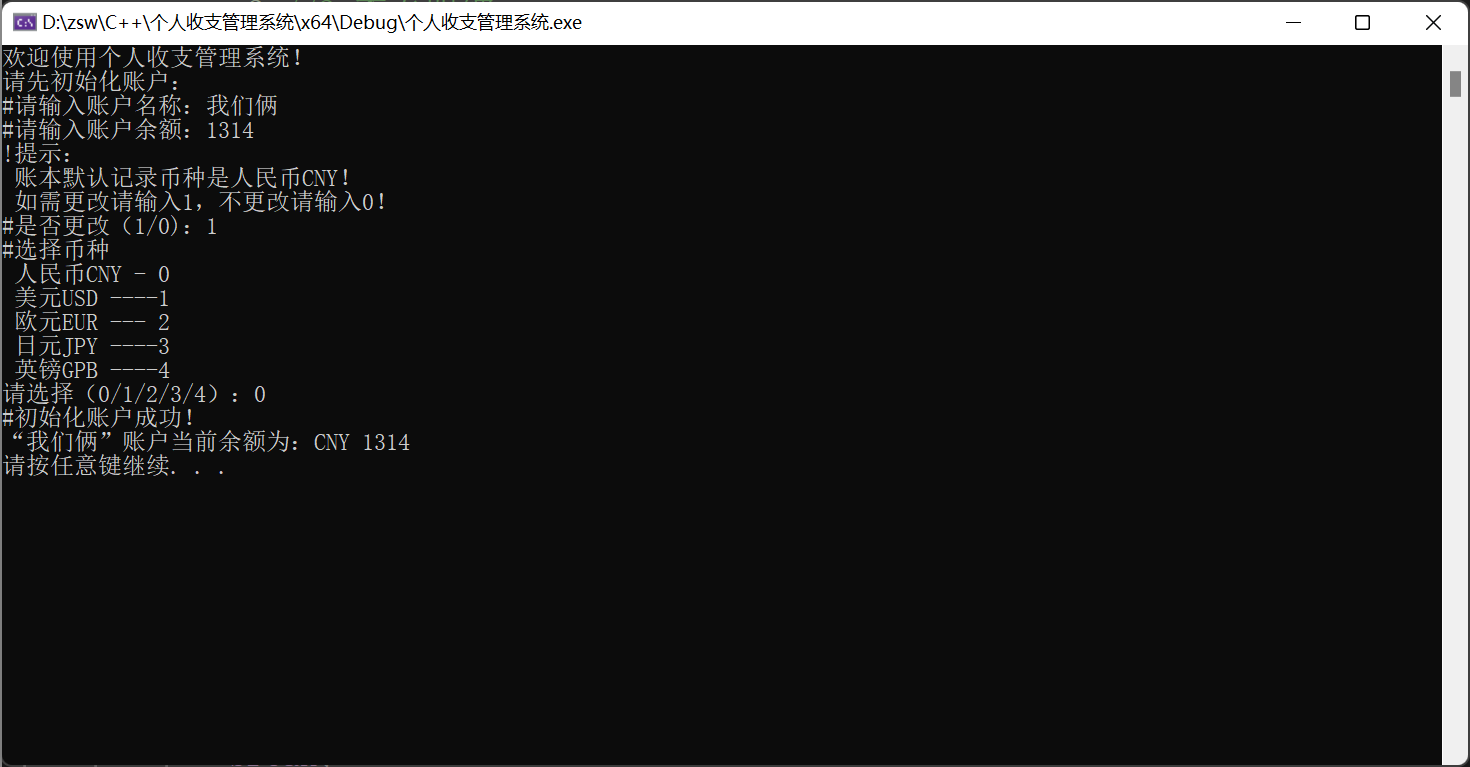
## 运行结果

### 系统初始状体

刚进入，系统直接提示用户需要初始化账户。



初始化账户完成。



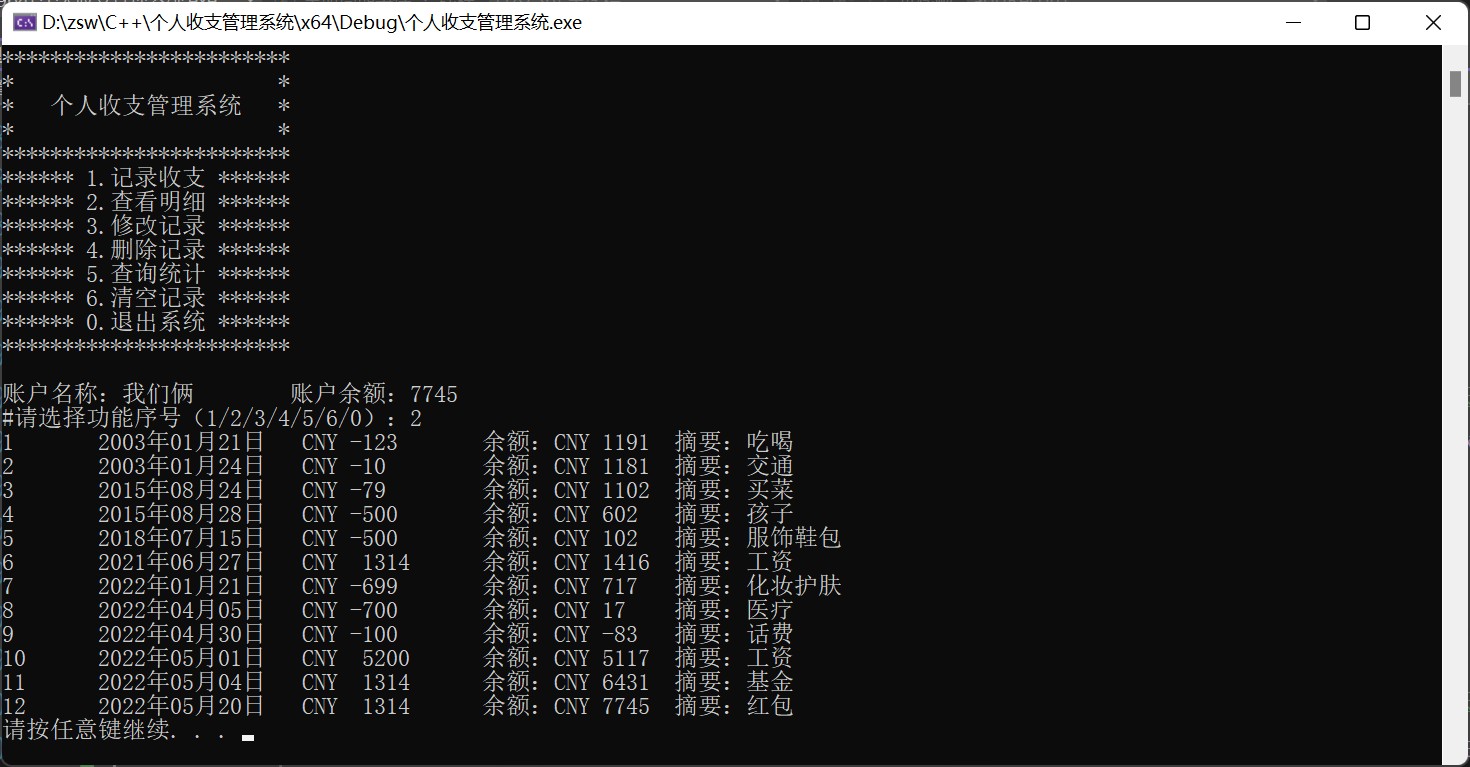
### 主界面显示状态



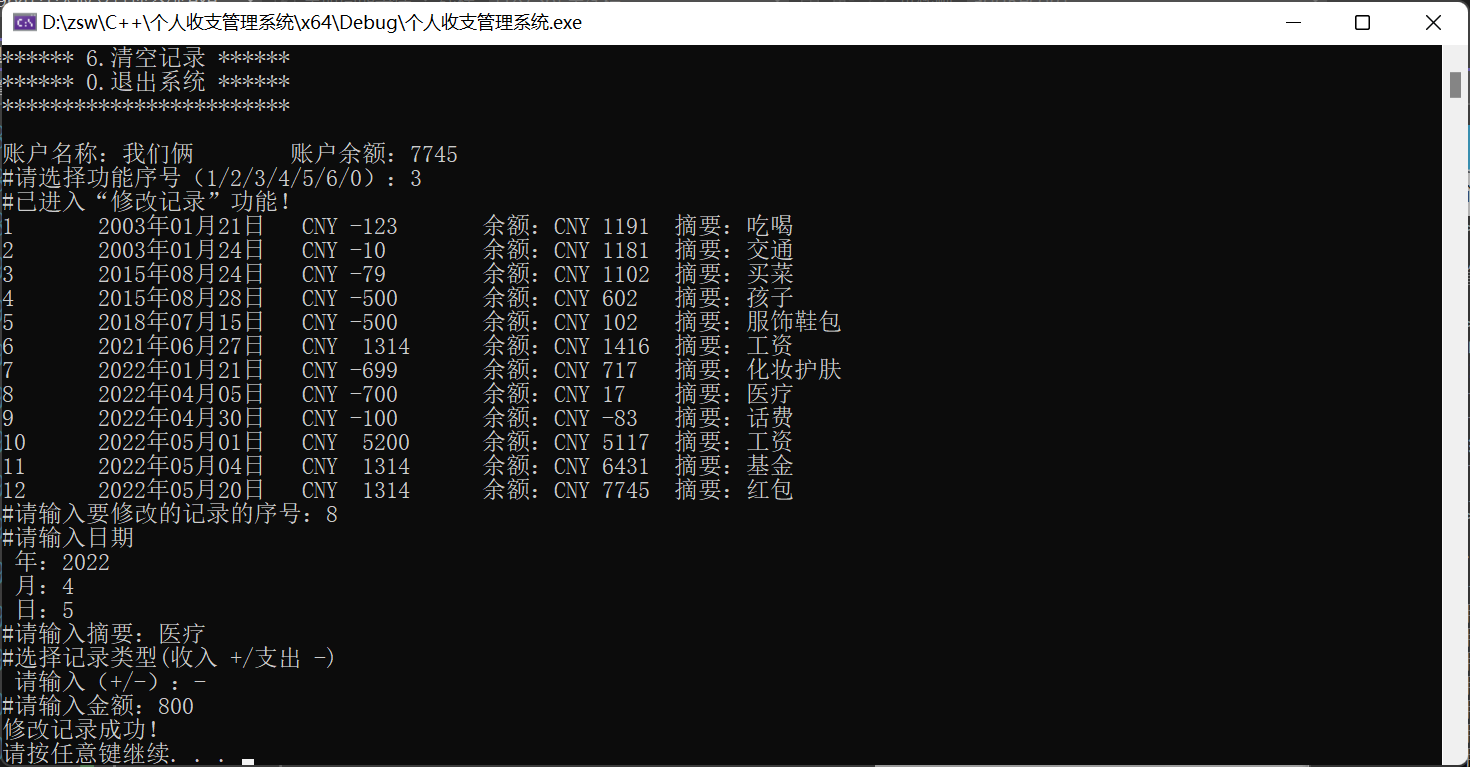
### 记录收支状态



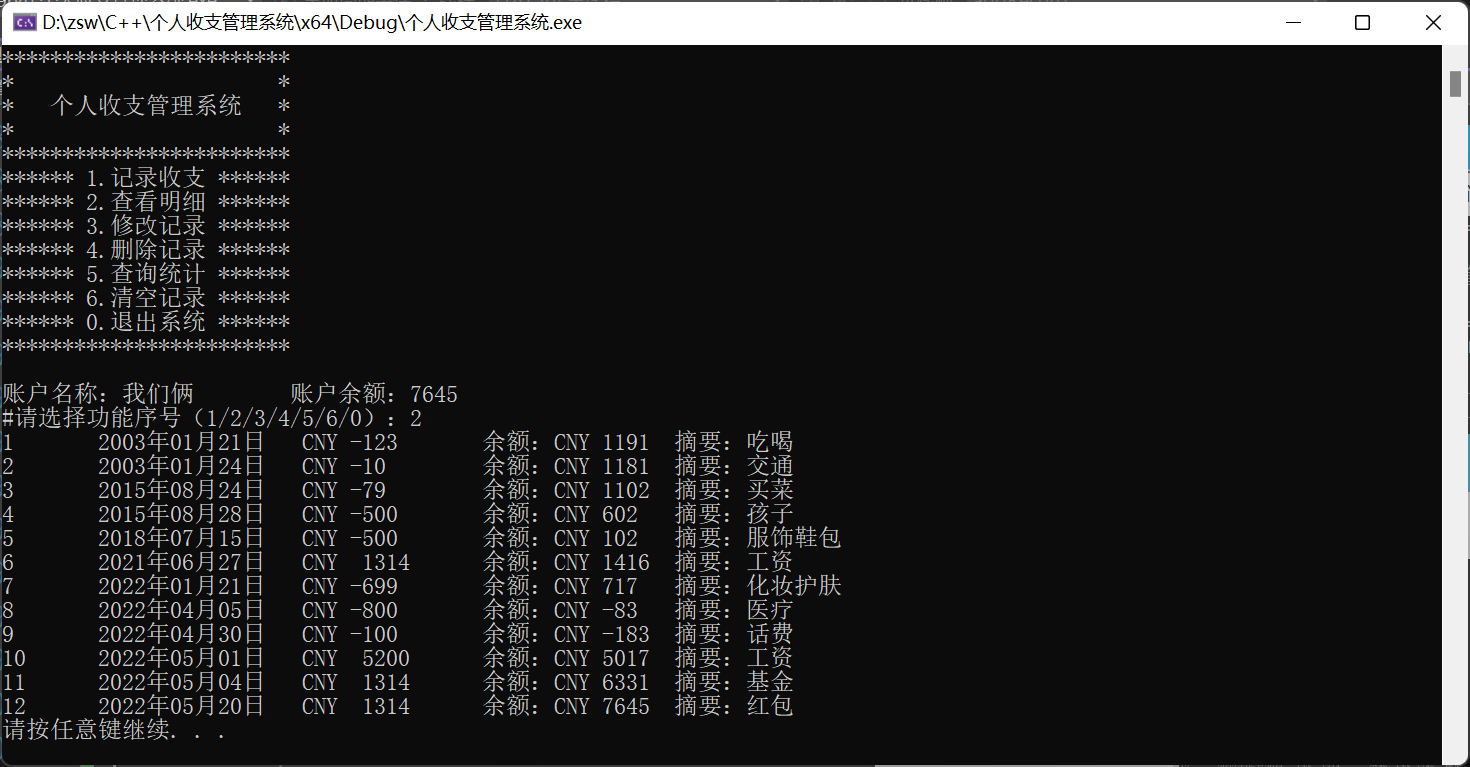
### 查看明细状态



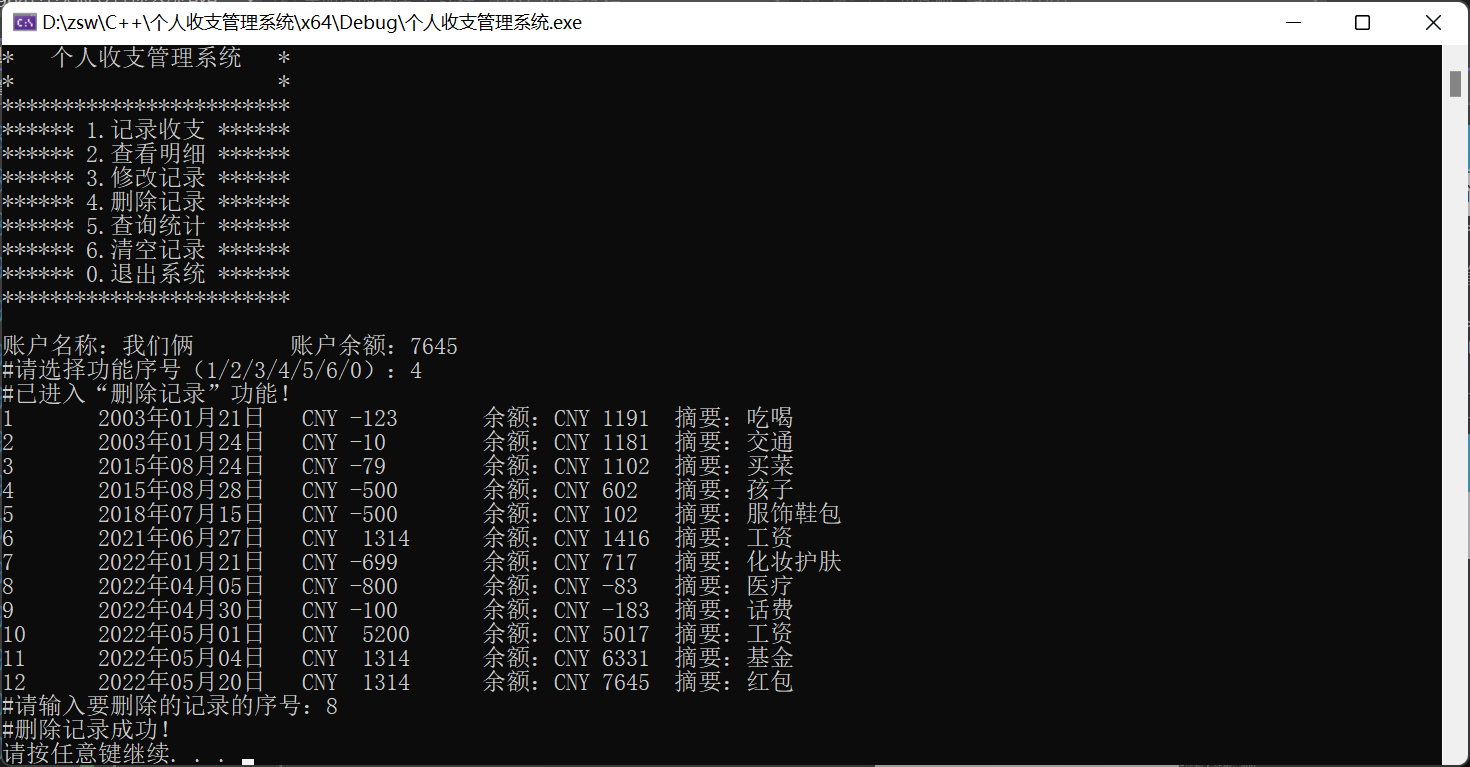
### 修改记录状态



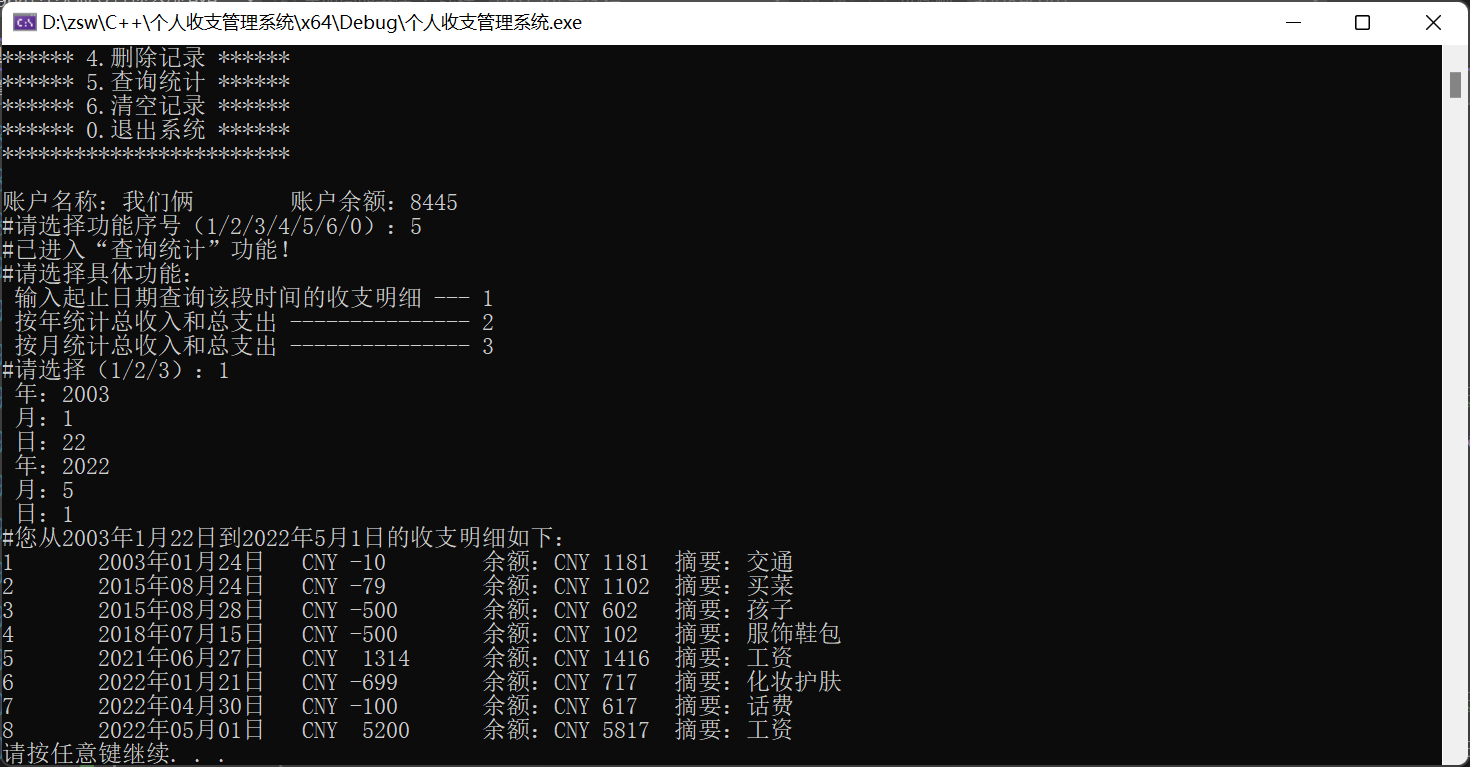
修改后。



### 删除记录状态



### 记录统计状态



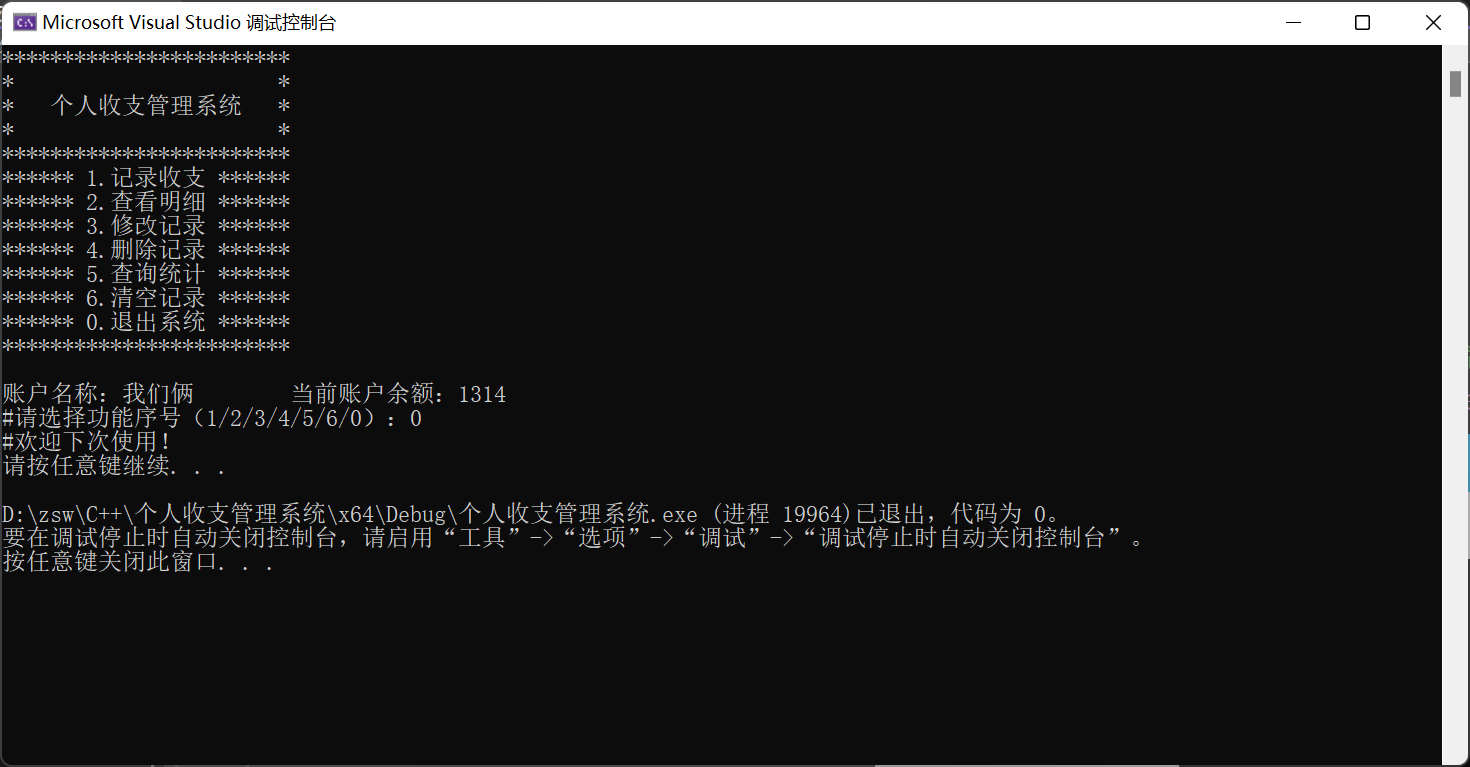




### 清空记录状态



### 退出系统状态



# 课 程 设 计 总 结

通过两周课程设计，我认识到了自己的不足，也接触到了之前没有接触过的函数和语法。在对记录按时间排序和判断时间范围我都遇到了不小的困难，甚至有过放弃的念头。但在老师的帮助下和自己的深入思考，最终将问题解决。我也意识到自己现在的水平有限，以后更需要加大努力，多学习一些新知识。