**项目一 小谷记账簿软件**

## 一、整体介绍

### 1.1 需求和目标

项目的需求，实现一个叫做“小谷记账簿”的基于命令行界面的家庭记账软件。

我们的目标，是通过项目练习，综合运用在C++第一部分课程中学习到的各种知识，初步掌握编写软件的方法和技巧。

### 1.2 整体功能描述

这个软件相对简单，只需要基于命令行做纯文本的交互，不涉及图形化界面。作为一个记账软件，主要有两大功能：记账和查询。也就是说，软件需要能够记录家庭中每一笔收入、支出的账目，并提供查询收支明细的功能。而且为了让软件有真正的实用价值，需要对数据做持久化保存；我们这里只需要存放在一个指定的文本文件中。

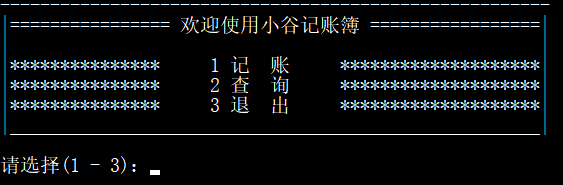
项目采用分级菜单方式，每一级菜单应该有“返回主菜单”功能；主菜单有“退出”选项。

## 二、页面及功能描述

### 2.1 主菜单

运行软件后首先应该显示主菜单。主菜单提供三个选项：“记账”、“查询”和“退出”，并在下方提示用户输入1-3的数字，选择不同的功能。

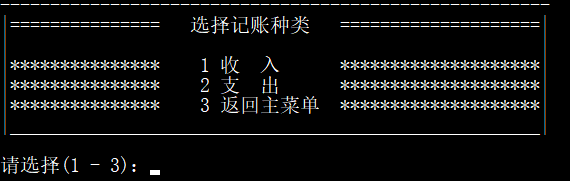
主菜单界面如下：



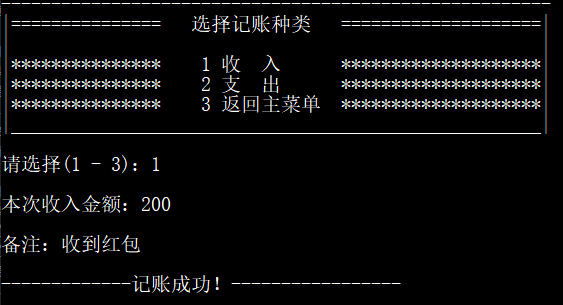
### 2.2 记账菜单

在主菜单页面选择“1”，可以进入记账菜单（二级菜单）。记账菜单提供三个选项：“收入”、“支出”和“返回主菜单”；并在下方提示用户按对应的数字进行功能选择。

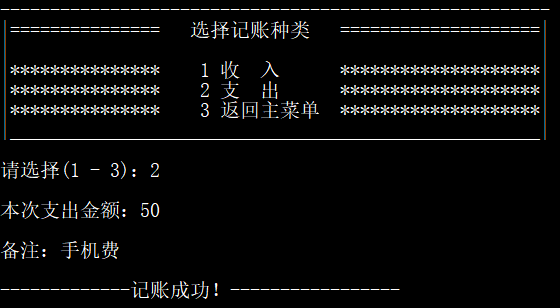
记账菜单界面如下：



在记账菜单页面选择“1”，可以记录一笔收入账目。账目信息包括：类型（收入/支出）、金额、备注。这里需要提示用户输入账目的金额和备注信息，然后显示完成记账。



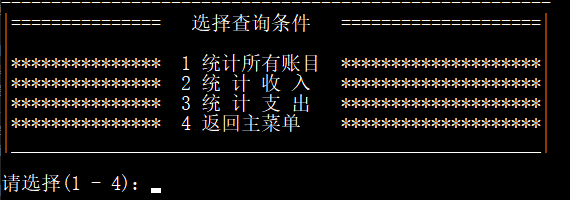
类似地，在记账菜单页面选择“2”，可以记录一笔支出账目。提示用户输入账目的金额和备注信息，然后显示完成记账。



### 2.3 查询菜单

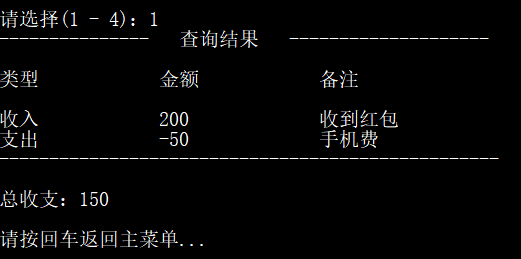
在主菜单页面选择“2”，可以进入查询菜单。查询菜单提供四个选项：“统计所有账目”、“统计收入”、“统计支出”和“返回主菜单”；并在下方提示用户按对应的数字进行功能选择。

查询菜单界面如下：



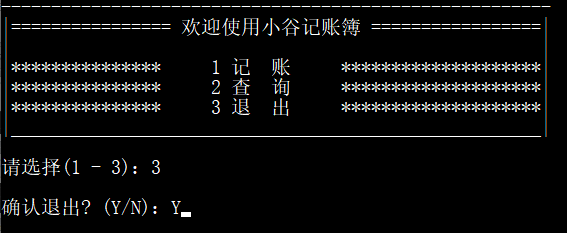
在查询菜单页面，选择“1”可以查询所有账目；选择“2”可以查询所有收入；选择“3”可以查询所有支出。用户选择之后，除了应该显示对应的账目明细外，还应该对所有账目进行统计汇总。

对应地，选择“1”之后列出所有账目，统计的是总收支；选择“2”之后列出所有收入，统计总收入；选择“3”之后列出所有支出，统计总支出。



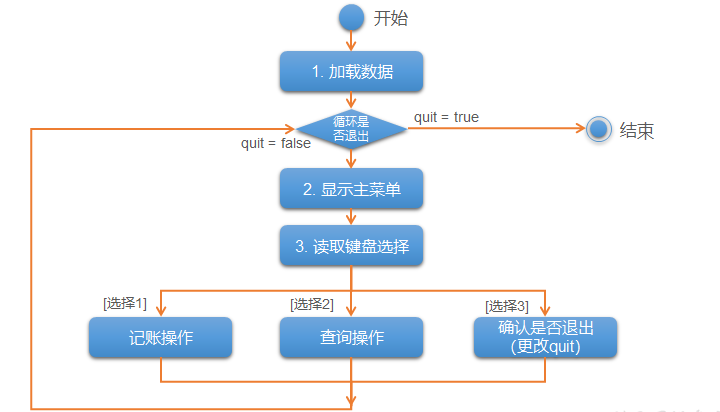
### 2.4 退出功能

在主菜单页面，选择“3”可以退出软件。页面将做确认退出的对话提醒：如果用户输入“Y”则退出，输入“N”则返回主菜单页面。

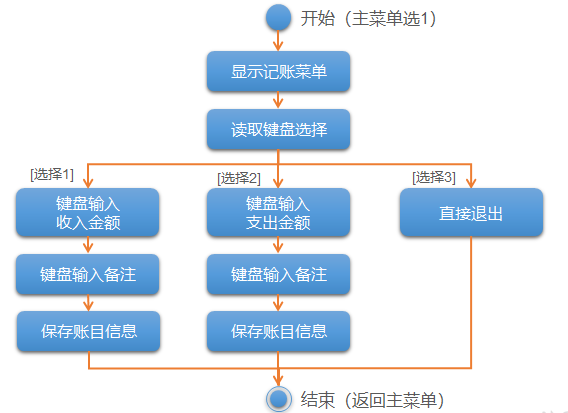


## 三、流程设计

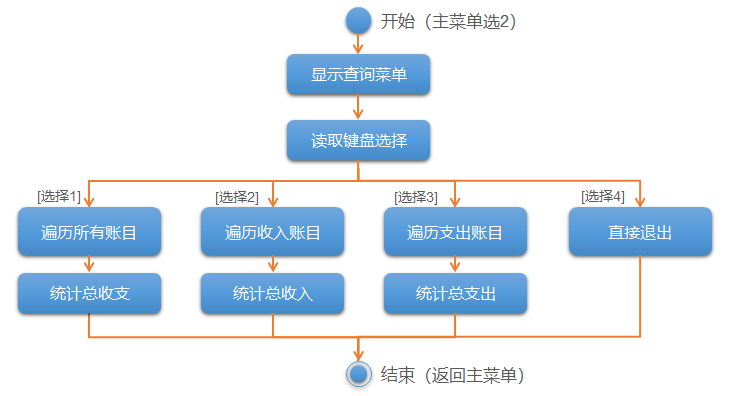
### 3.1 主流程



### 3.2 记账操作流程



### 3.3 查询操作流程



## 四、代码设计

### 4.1 核心思路

记账软件处理的数据，就是收入或者支出的“账目”。每一条账目数据，都应该包含收支类型、金额、备注三部分，这可以构建一个结构体类型 AccountItem 来表示。

而程序运行的开始，应该读取保存在文件中的数据，读取之后应该是一个 AccountItem 类型的数组。

在流程控制方面，如果用户不选择退出，程序就不会结束，所以应该用一个while循环来处理整个流程；当用户确认退出时，更改一个标志位，用来退出循环。

每一级菜单做键盘选择后，可以用 switch 分支语句来处理；不同的功能模块，可以包装成函数。

### 4.2 项目文件分类设计

#### 4.2.1 头文件

对于一个C++项目来说，构建整体架构很重要的一步就是定义头文件。一般我们会把全局变量、函数声明、以及结构体的定义都放在头文件中。

本项目可以设计两个头文件，来管理不同的内容：

* 结构体 AccountItem 的定义，以及涉及到账目操作的函数声明，都可以放在一个头文件 account\_item.h 中；
* 其它一些通用的设置和功能性函数，可以放在另一个头文件common.h 中

#### 4.2.2 源文件

真正的代码实现都放在源文件中。根据不同的用途，项目中可以用四个源文件来实现对应的功能

* 主体代码可以放在一个源文件 xiaogu\_acount.cpp 中
* 绘制菜单的函数都放在menus.cpp中
* 通用函数（比如读取键盘输入）放在common.cpp中
* 针对账目的所有操作函数（比如记账、查询）全部放在operations.cpp中

### 4.3 代码具体实现

#### 4.3.1 定义头文件

首先可以在common.h中，引入相应的库，定义出需要的全局变量，以及声明绘制菜单和读取键盘输入的通用函数。

#pragma once

#include<iostream>

#include<fstream>

#include<string>

#include<vector>

#define FILENAME "D:\\data\\AccountBook.txt" // 文件路径注意是双反斜杠分隔

#define INCOME "收入"

#define EXPAND "支出"

using namespace std;

// 读取键盘输入并校验的通用函数

char readMenuSelection(int options);

int readAmount();

char readQuitConfirm();

// 绘制菜单的函数

void showMainMenu();

void showAccountingMenu();

void showQueryMenu();

然后在account\_item.h定义核心数据结构：结构体 AccountItem ，并声明涉及到账目操作的函数。

#pragma once

#include "common.h"

struct AccountItem

{

string itemType;

int amount;

string detail;

};

// 打印输出一条账目

void printAccountItem(const AccountItem& item);

// 针对账目AccountItem的操作函数

void loadDataFromFile(vector<AccountItem>& items);

void insertItemIntoFile(const AccountItem& item);

// 记账

void accounting(vector<AccountItem>& items);

void income(vector<AccountItem>& items);

void expand(vector<AccountItem>& items);

// 查询

void query(vector<AccountItem>& items);

void queryItems(vector<AccountItem>& items);

void queryItems(vector<AccountItem>& items, const string accoutType);

#### 4.3.2 实现主体流程

在源文件 xiaogu\_acount.cpp 的主函数中，参照主流程图实现主体流程。

代码如下：

#include "common.h"

#include "account\_item.h"

int main()

{

vector<AccountItem> items;

// 1. 加载文件中数据

loadDataFromFile(items);

// 为了方便控制循环退出，设置一个标志位

bool quit = false;

while (!quit)

{

// 2. 显示主菜单

showMainMenu();

// 3. 从键盘获取选择，由于需要做异常判断，所以包装成函数

// 这里的3是最多三个选项

switch (readMenuSelection(3))

{

case '1': // (1) 记账

showAccountingMenu();

accounting(items);

break;

case '2': // (2) 查询

showQueryMenu();

query(items);

break;

case '3': // (3) 退出

cout << "\n确认退出? (Y/N)：";

if (readQuitConfirm() == 'Y')

quit = true;

break;

default:

break;

}

cout << endl;

}

}

#### 4.3.3 绘制菜单

在 menus.cpp 中定义绘制各级菜单的函数；对应的函数声明放在 common.h里。

代码如下：

// 显示菜单的函数

#include "common.h"

void showMainMenu()

{

system("cls"); // 调用system函数，清屏，对windows有效

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

cout << "|================ 欢迎使用小谷记账簿 =================|" << endl;

cout << "| |" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1 记 账 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2 查 询 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3 退 出 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "\n请选择(1 - 3)：";

}

void showAccountingMenu()

{

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

cout << "|=============== 选择记账种类 ====================|" << endl;

cout << "| |" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1 收 入 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2 支 出 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3 返回主菜单 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "\n请选择(1 - 3)：";

}

void showQueryMenu()

{

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

cout << "|=============== 选择查询条件 ====================|" << endl;

cout << "| |" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1 统计所有账目 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2 统 计 收 入 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3 统 计 支 出 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4 返回主菜单 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

cout << "\n请选择(1 - 4)：";

}

#### 4.3.4 读取键盘输入

在 common.cpp 中定义读取键盘输入的函数，包括读取菜单选择、读取输入的金额数、读取确认退出信息；这些函数都需要对输入做合法性校验，并返回正确的值。

代码如下：

#include "common.h"

// 读取键盘输入的菜单选择，参数为选项个数

char readMenuSelection(int options)

{

string str;

while (true)

{

getline(cin, str);

if (str.size() > 1 ||

str[0] - '0' <= 0 ||

str[0] - '0' > options)

{

cout << "输入错误，请重新选择：";

}

else

break;

}

return str[0];

}

// 读取键盘输入的金额数

int readAmount()

{

int amount;

string str;

while (true)

{

getline(cin, str);

try {

return stoi(str);

}

catch (invalid\_argument e) {

cout << "输入错误，请正确输入数字：";

}

}

}

// 读取确认退出信息

char readQuitConfirm()

{

string str;

char ch;

while (true)

{

getline(cin, str);

if (str.empty())

continue;

ch = toupper(str[0]);

if (str.size() > 1 ||

ch != 'Y' && ch != 'N')

{

cout << "\n输入错误，请重新输入（Y/N）：";

}

else

break;

}

return ch;

}

#### 4.3.5 实现具体操作功能

在 operations.cpp 中，定义从文件加载数据、记账、查询等操作对应的函数。

（1）从文件加载数据

逐行读取文件中数据，按空格将每个词赋值给AccountItem中的每个成员，得到一个AccountItem对象，并把它保存到数组中。

由于数组长度是固定的，这里应该用可变长的容器对象vector<AccountItem>来保存；为了能够在函数中修改vector，应该把它的引用作为函数参数传入。

（2）记账功能

首先需要读取键盘输入的金额和备注信息；然后将其保存在一个AccountItem 对象中，并添加到数组，同时写入文件。

（3）查询功能

需要用for循环遍历整个数组，通过条件筛选需要的账目打印输出，并统计总和。

具体代码如下：

// 执行操作的函数

#include "common.h"

#include "account\_item.h"

// 读取文件

void loadDataFromFile(vector<AccountItem>& items)

{

ifstream input(FILENAME);

// 逐行读取

AccountItem item;

while (input >> item.itemType && input >> item.amount && input >> item.detail)

{

items.push\_back(item);

}

input.close();

}

// 将一条账目写入文件

void insertItemIntoFile(const AccountItem& item)

{

ofstream output(FILENAME, ios::out | ios::app);

output << item.itemType << "\t" << item.amount << "\t" << item.detail << endl;

output.close();

}

// ------------1. 记账操作------------//

void accounting(vector<AccountItem>& items)

{

switch (readMenuSelection(3))

{

case '1':

income(items);

break;

case '2':

expand(items);

break;

default:

break;

}

}

// 记账操作-收入

void income(vector<AccountItem> & items)

{

AccountItem item;

item.itemType = INCOME;

cout << "\n本次收入金额：";

item.amount = readAmount();

cout << "\n备注：";

getline(cin, item.detail);

items.push\_back(item);

insertItemIntoFile(item);

cout << "\n-------------记账成功！-----------------" << endl;

cout << "\n请按回车返回主菜单..." << endl;

string line;

getline(cin, line);

}

// 记账操作-支出

void expand(vector<AccountItem>& items)

{

AccountItem item;

item.itemType = EXPAND;

cout << "\n本次支出金额：";

item.amount = - readAmount();

cout << "\n备注：";

getline(cin, item.detail);

items.push\_back(item);

insertItemIntoFile(item);

cout << "\n-------------记账成功！-----------------" << endl;

cout << "\n请按回车返回主菜单..." << endl;

string line;

getline(cin, line);

}

// ------------2. 查询操作------------//

void query(vector<AccountItem>& items)

{

switch (readMenuSelection(4))

{

case '1':

queryItems(items);

break;

case '2':

queryItems(items, INCOME);

break;

case '3':

queryItems(items, EXPAND);

break;

default:

break;

}

}

void queryItems(vector<AccountItem> & items)

{

// 记录总收支

int total = 0;

cout << "--------------- 查询结果 --------------------" << endl;

cout << "\n类型\t\t金额\t\t备注\n" << endl;

// 遍历所有账目

for (auto item: items)

{

printAccountItem(item);

total += item.amount;

}

cout << "--------------------------------------------------\n" << endl;

cout << "总收支：" << total << endl;

cout << "\n请按回车返回主菜单..." << endl;

string line;

getline(cin, line);

}

void queryItems(vector<AccountItem>& items, const string accoutType)

{

// 记录总和

int total = 0;

cout << "--------------- 查询结果 --------------------" << endl;

cout << "\n类型\t\t金额\t\t备注\n" << endl;

for (auto item : items)

{

if (item.itemType != accoutType)

continue;

printAccountItem(item);

total += item.amount;

}

cout << "--------------------------------------------------\n" << endl;

cout << (accoutType == INCOME ? "总收入：" : "总支出：") << total << endl;

cout << "\n请按回车返回主菜单..." << endl;

string line;

getline(cin, line);

}

// 打印输出一条账目

void printAccountItem(const AccountItem& item)

{

cout << item.itemType << "\t\t" << item.amount << "\t\t" << item.detail << endl;

}