实验三：软件实现与构建

231220080 丁一鸣

1.根据UML图实现代码

代码实现的类主要是根据UML图中的类图中的具体划分实现，即分为Category、Transaction、StatisticService、UIService和SearchService进行，每个类中进行的操作也参考了类图中的操作。需要注意的是只实现了主体部分的内容，对于User的登录部分没有进行实现。

```
// 分类类
class Category {
public:
 Category(std::string id, std::string name)
     : categoryId(id), categoryName(name) {}
```

```
// 交易类
class Transaction {
public:
 Transaction()
     : transactionId(""), amount(0.0), transactionTime(""),
       categoryId(""), remarks(""),
       transactionType(TransactionType::Expense) {}
```

```
// 统计服务
class StatisticService {
public:
 explicit StatisticService(const std::vector<Transaction>& data)
     : transactions(data) {}
```

```
// 搜索服务
class SearchService {
public:
 explicit SearchService(const std::vector<Transaction>& data)
     : transactions(data) {}
```

```
// UI 服务
class UIService {
public:
 static std::vector<Transaction> sortTransactionList(
     std::vector<Transaction> list, SortField field, SortDirection direction) {
```

代码main函数中实现了在终端中的简单的依托于字符的图形化，这个程序进行的流程是参考UML图中的活动图进行，分为记一笔、统计和查找三个主要功能，这里和UML图不一样的是增加了一个管理分类的模块专门进行分类的增删改查。

代码实现还有一点和类图不一样，在进行统计的时候不将统计数据单独存储，而是在调取时实时计算，这样简化了代码的实现，在数据量较小时基本不影响效率。

程序运行依赖和修改的数据直接存在了一个和程序同名的db文件中，但是不是通过现成的数据库管理实现的，而是直接通过读取文件，将其中的内容保存在vector中实现了类似的功能。

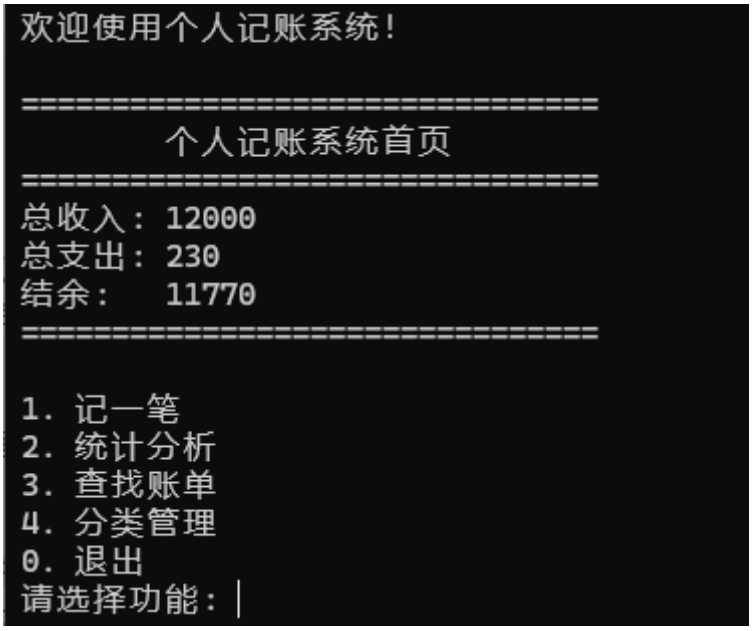
关于大模型的使用，在实验初期使用了Claude4.5模型，根据类图生成了几个类的大致框架，但是实现的代码过于复杂，每个功能之间的实现都是相对独立的，很难进行整合，然后我通过具体的需求简化了一些类的功能和内容，比如删去了存储统计数据的功能和用户登录的功能，简化了增加和删除的逻辑等。

还有一点用到大模型的是数据的存取部分，本身只是通过vector实现，每次进入程序的时候数据都是全新的，后来直接让大模型增加了一个数据的存取功能，可以将数据存在程序同名的db文件中进行数据库管理。在使用过程中使用了GPT5和Claude4.5，综合下来Claude4.5的代码能力更强一点。

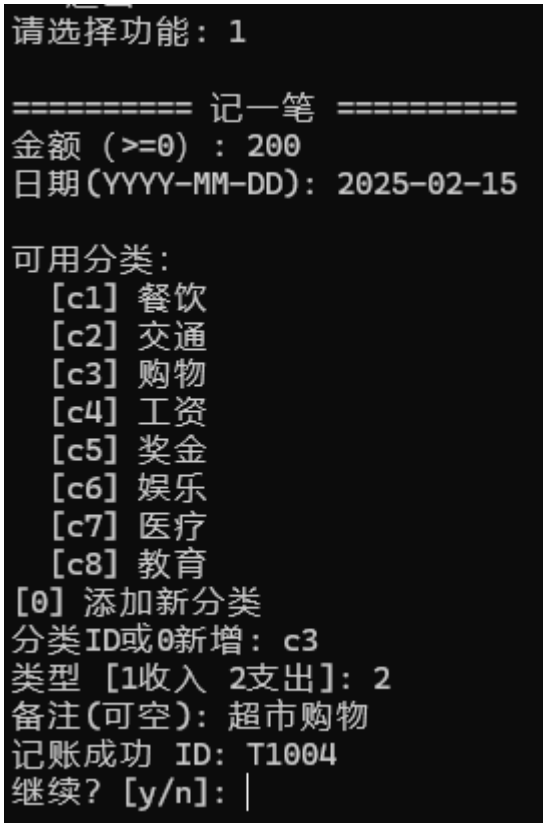
2.代码的编译与运行结果

使用`g++ -std=c++17 -O2 -Wall -Wextra -pedantic -o bookkeeping code.cpp`进行编译，编译后生成bookkeeping.exe即为可执行文件，程序运行中产生的数据存储在bookkeeping.db中。这里编译是在windows系统上进行的，在windows的终端中没有乱码出现。具体的运行结构如下：

首页



输入数据进行功能的选择，即活动图中活动的选择，这里分别演示一下：



```
请选择功能: 2

========== 统计分析 ==========

1. 按类型
2. 按时间
3. 按分类
4. 按金额区间
0. 返回
选项: 2
1日 2月 3年
粒度: 2
开始日期(可空): 2022-01-01
结束日期(可空): 2026-01-01

========== 统计结果 ==========
总收入: 12000
总支出: 430
结余: 11570

时间段统计:
时间     收入     支出     结余
------------------------------------
2025-01 10000   30       9970
2025-02 0       400      -400
2023-12 2000    0        2000
继续统计? [y/n]:
```

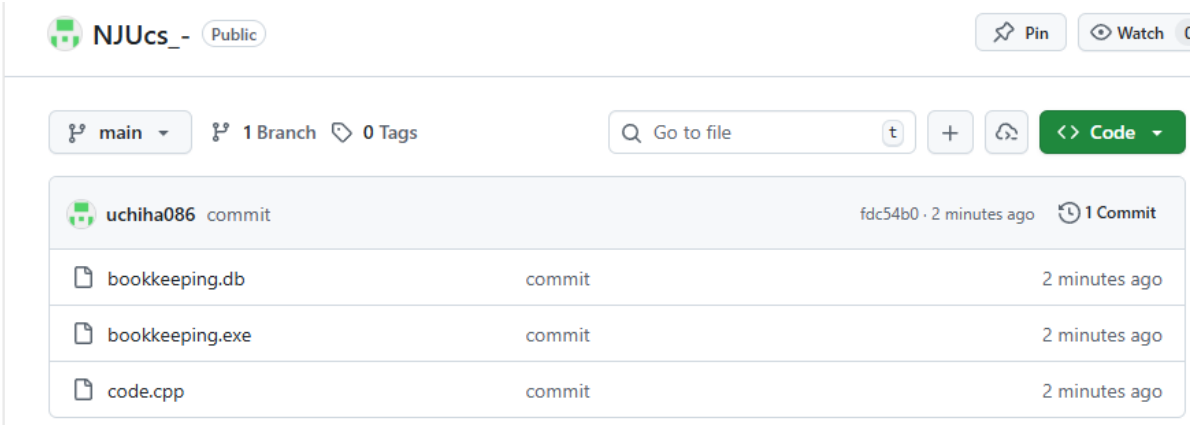
```
请选择功能: 3

========== 查找账单 ==========
起始日期(可空):
结束日期(可空):
最小金额(可空):
最大金额(可空):

可用分类:
  [c1] 餐饮
  [c2] 交通
  [c3] 购物
  [c4] 工资
  [c5] 奖金
  [c6] 娱乐
  [c7] 医疗
  [c8] 教育
分类ID(可空):
关键词(可空):

========== 交易列表 ==========
1. 交易[T1000] 金额=10000.000000, 时间=2025-01-01, 分类=工资(c4), 类型=收入, 备注=
2. 交易[T1001] 金额=200.000000, 时间=2025-02-01, 分类=餐饮(c1), 类型=支出, 备注=晚饭
3. 交易[T1002] 金额=30.000000, 时间=2025-01-15, 分类=购物(c3), 类型=支出, 备注=水
4. 交易[T1003] 金额=2000.000000, 时间=2023-12-01, 分类=奖金(c5), 类型=收入, 备注=
5. 交易[T1004] 金额=200.000000, 时间=2025-02-15, 分类=购物(c3), 类型=支出, 备注=超市购物
共 5 条
排序? [y/n]:
操作[1编辑 2删除 0跳过]:
```

```
1. 添加分类
2. 删除分类
3. 返回
请输入选项: 1
分类名称（必填）: 123
添加成功 ID: c100
```

以上就是主要功能的演示，生成的数据存储在bookkeeping.db中：

```
[CATEGORIES]
c1|餐饮
c2|交通
c3|购物
c4|工资
c5|奖金
c6|娱乐
c7|医疗
c8|教育
c100|123
[TRANSACTIONS]
T1000|10000|2025-01-01|c4|0|
T1001|200|2025-02-01|c1|1|晚饭
T1002|30|2025-01-15|c3|1|水
T1003|2000|2023-12-01|c5|0|
T1004|200|2025-02-15|c3|1|超市购物
```

3.Git远程代码管理展示



将代码托管到自己的Github仓库中了