**北京科技大学实验报告**

学院：计算机与通信工程学院 专业： 计算机类 班级；计2401

姓名： 孙煜欣 学号： U202442404 实验日期：2025 年 5 月 17 日

**实验名称：个人银行账户管理系统的改进**

**实验目的：**

完善银行账户系统，并实现以下功能：

1. 增加登录注册界面，更加生活化
2. 增加了用户界面，可以通过点击的方式创建和管理账户

3.增加查询排序功能，更直观显示交易记录

**实验仪器：**

计算机：联想小新2018

CPU：1.60GHz IntelCore i5

内存：20GB 2133MHz LPDDR

硬盘：256GB

显卡：Intel(R) UHD Graphics 620

操作系统：Windows 11

编译器：visual Studio 2022

**实验内容与步骤：**

**【**功能1：建立简单的储蓄账户**】**

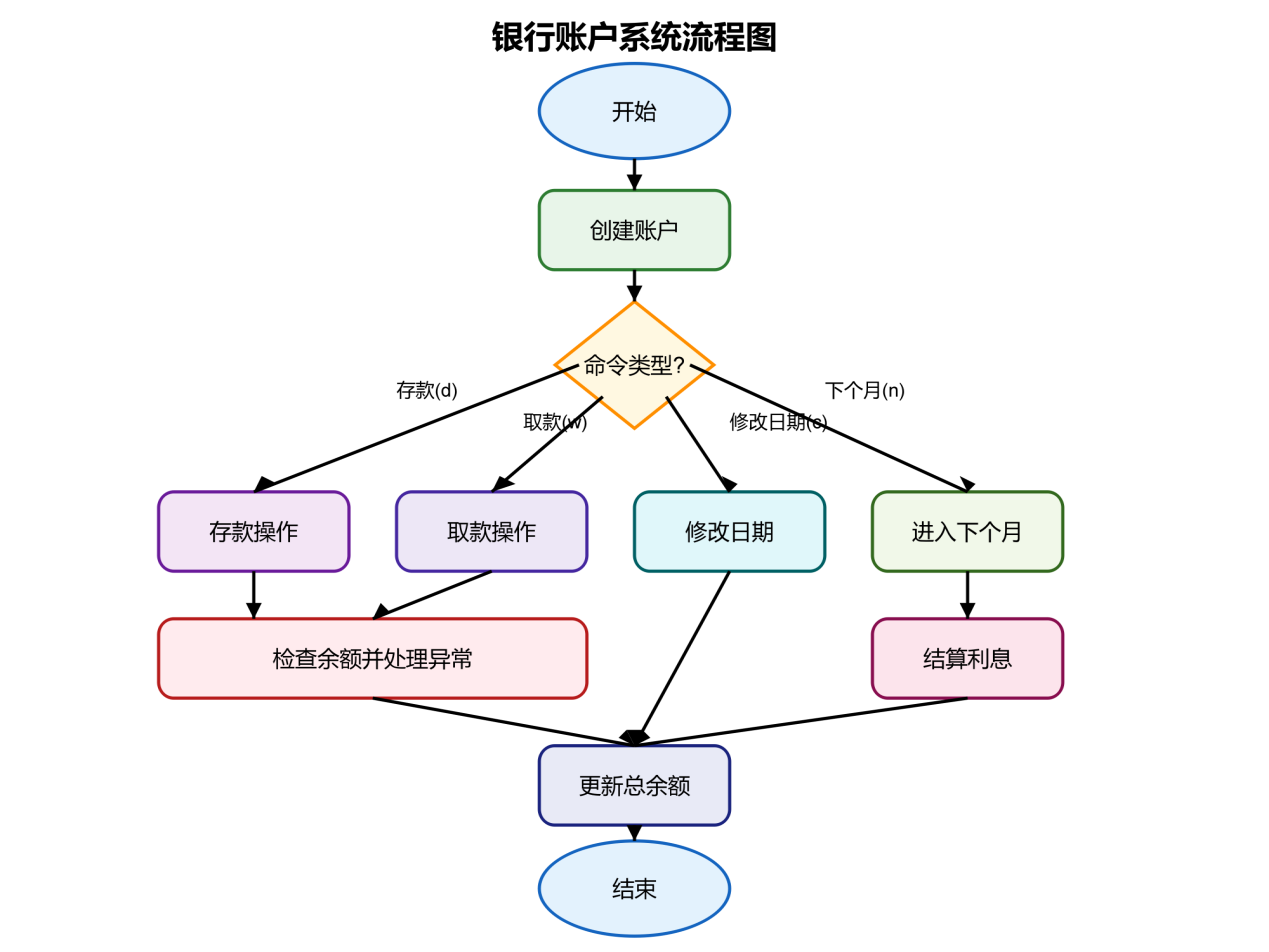
1.设计

1. Date类,Account 类,AccountRecord 类,SavingsAccount 类,CreditAccount 类,AccountException 类.

2.算法：（1）日期处理算法包括两部分：一是闰年判断，方法是看年份能否被4整除但不能被100整除，或者能被400整除；二是日期间隔的计算，把两个日期都转换成从公元1年1月1日开始的总天数，然后求出它们之间的差。

（2）利息计算方面，储蓄账户通过“天数×余额”的累积值（accumulation）来计算利息，每年1月1日进行一次结算；而信用账户则是对每天的负余额按利率计算利息，到了每年1月1日还会收一次年费。

（3）账户查询使用 multimap 来按照日期保存所有交易记录，查找时可以用 lower\_bound() 找到起始日期之后的第一个记录，然后依次遍历并显示指定时间范围内的所有交易。



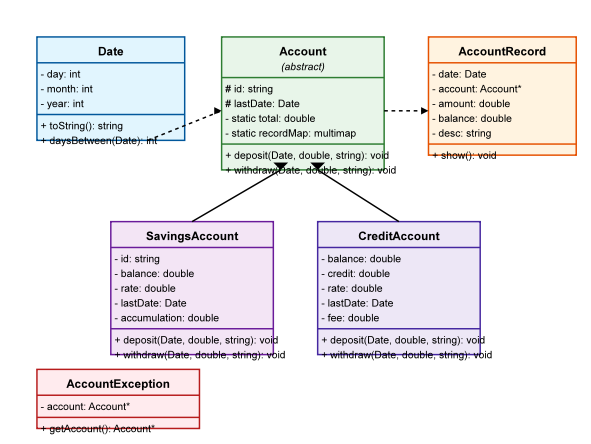
Date 类负责日期的表示和操作，提供如日期间隔计算、是否为闰年的判断等功能。

Account 类是一个抽象基类，用于定义所有账户共有的操作接口，同时维护一个静态变量 total 来记录所有账户的总余额，并使用静态 multimap 来保存所有账户的交易记录。

AccountRecord 类用来记录每一笔交易的详细信息，包含日期、所属账户、交易金额、交易后余额以及交易描述等，并提供 show() 方法用于显示交易内容。

SavingsAccount 类是储蓄账户，继承自 Account 类，实现了具体的存取款操作和利息计算方式，它通过一个 accumulation 变量来累加“天数×余额”，用于按年计算利息。

CreditAccount 类是信用账户，同样继承自 Account 类，用于管理信用额度和可能出现的负余额，每天计算负余额的利息，并在每年初收取一次年费。AccountException 类用于处理账户操作中可能出现的异常情况，比如取款金额超过账户可用余额等。



银行账户系统的主要操作流程包括以下几个方面：1.首先是创建账户，系统根据用户的命令创建储蓄账户或信用账户；2.随后是命令处理，解析用户输入的操作指令存款操作会增加账户余额，同时更新系统的总余额，并记录该笔交易；3.取款操作则需先检查余额是否足够，若不足会抛出异常，否则正常扣减余额、更新总余额并记录交易。4.系统还支持修改当前日期，以及进入下个月的操作，后者会将日期更新至下月1日，并触发账户结算：储蓄账户在每年1月1日结算利息，信用账户则计算每日负余额利息，并在年初收取年费。5.整个系统还包括异常处理机制，用于应对如余额不足等操作异常的情况

2.实现

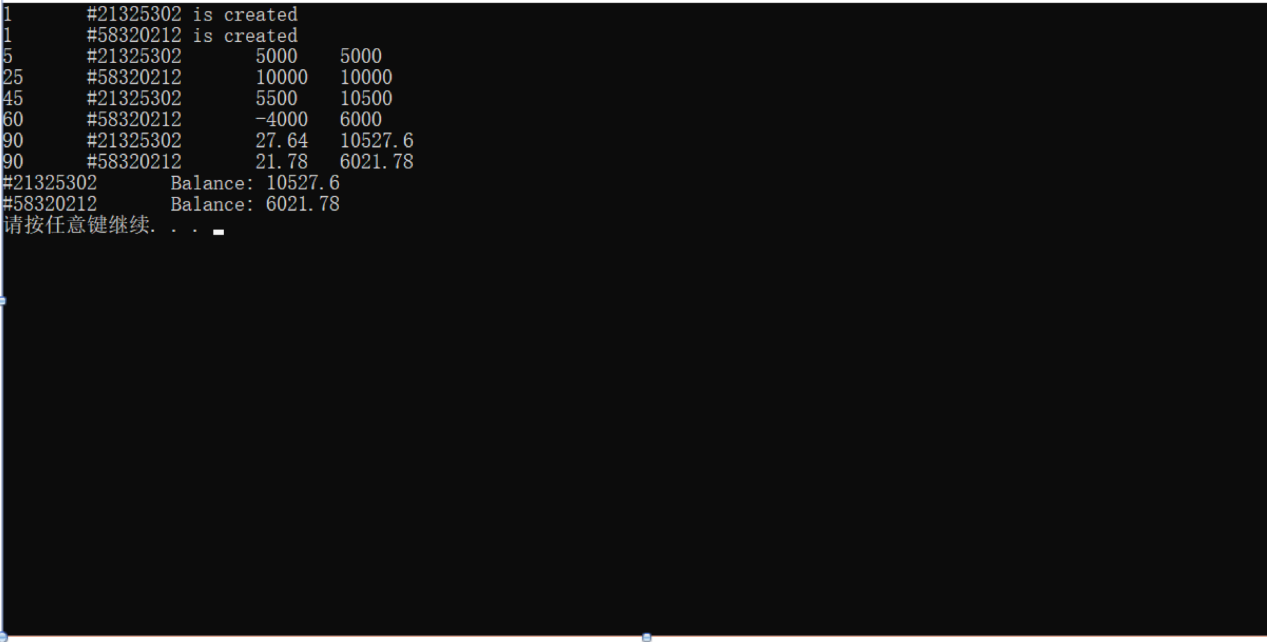
|  |
| --- |
| class SavingsAccount : public Account {  private:  std::string id; // 账户ID  double balance; // 账户余额  double rate; // 年利率  Date lastDate; // 上次操作日期  double accumulation; // 累积金额，用于利息计算  public:  // 构造函数  SavingsAccount(Date startDate, std::string id, double rate);  // 存款  void deposit(Date date, double amount, std::string description);  // 取款  void withdraw(Date date, double amount, std::string description);  // 结算  void settle(Date date);  // 获取所有账户的总余额  static double getTotal();  // 显示账户信息  void show() const;  // 计算并更新积数  double accumulate(Date date);  }; |

代码片段1 储蓄账户类的关键代码

3.测试

（1）测试用例1：a s S3755217 0.015（用户输入正确数据）

得到的运行截图如下：

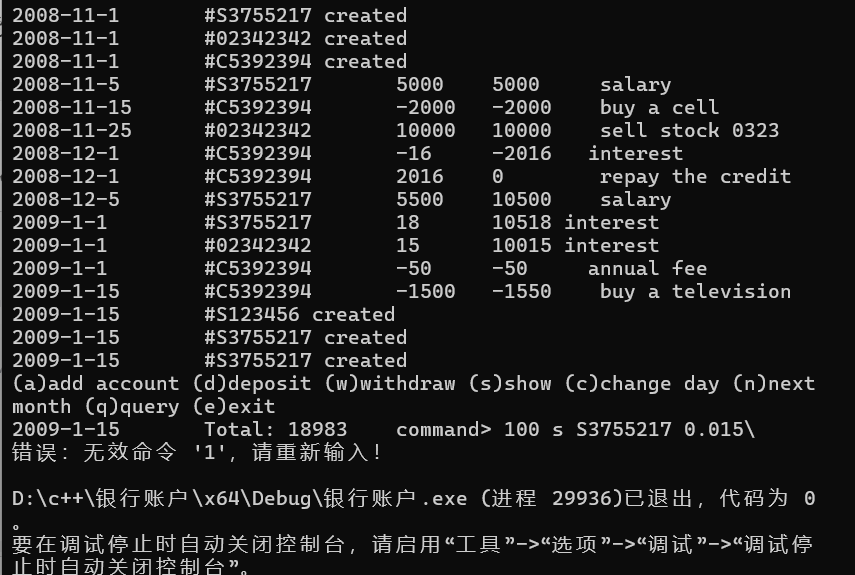


分析：设计这个测试用例的目的是什么？测出的结果表明什么？

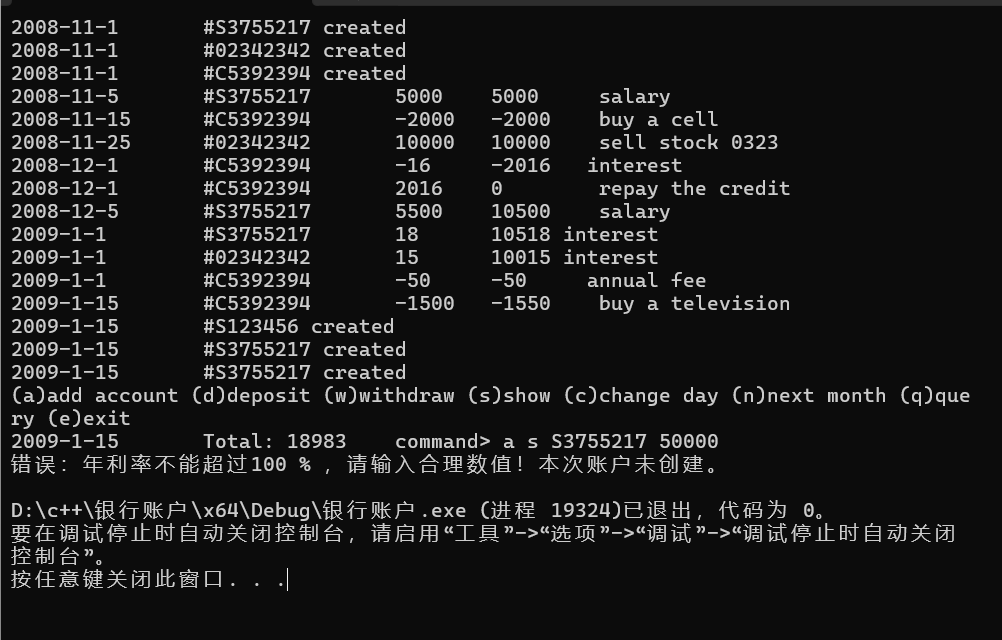
1：验证系统是否能够根据用户输入的指令正确创建储蓄账户，并设置相应的年利率。

2：结果表明系统未正确创建账户 S3755217

1. 测试用例2：100 s S3755217 0.015（用户输入了错误的选项）



（3）测试用例5: a s S3755217 50000（用户输入了错误的利率）



**【**功能2：建立简单的信用账户**】**

1. 设计：

程序中通过类的继承来建立信用账户。首先有一个基类 Account，它规定了账户该有的功能，比如存钱、取钱、结算等。信用账户是从这个基类继承来的，用 CreditAccount 类来表示。这个类中增加了信用额度、利率、年费等信息，并实现了存取款和结算的方法。

1. 实现：

|  |
| --- |
| class CreditAccount :public Account  {  private:  double balance;  double credit;//贷款额度  double rate;//每日利率  Date lastDate;  double fee;//年费，每年都要扣  public:  CreditAccount(Date date, std::string id, double credit, double rate, double fee);//构造函数  void deposit(Date date, double amount, std::string desc);//存款  void withdraw(Date date, double amount, std::string desc);//取款  void settle(Date date);//结算  void show() const;//显示账户信息  }; |

代码片段2 信用账户类的关键代码