



|  |  |  |  |
| --- | --- | --- | --- |
| **学号** |  | **姓名** |  |
| **班级** |  | **任课教师** |  |
| **实验名称** | **网上购物仿真系统** | | |
| **实验学期** | **2024 – 2025 学年第2学期** | | |
| **实验日期** |  | **实验地点** |  |
| **报告成绩** |  | | |

西安电子科技大学计算机科学与技术学院

1. **实验环境**

硬件环境：个人计算机

软件环境：Windows 11 操作系统，Visual Studio Code开发环境，g++编译器、C++ 语言标准库（C++11），文件存储采用INI、CSV、SL（购物清单）、OL（订单清单）等文本格式。

1. **实验内容**

本次实验目标是实现一个网上购物仿真系统，采用面向对象程序设计范型和C++语言开发，支持管理员后台管理与顾客前端购物功能。系统需满足 6 次上机任务的递进需求，涵盖用户管理、商品管理、购物车操作、订单流程、促销活动及数据持久化等核心功能。

六次上机实现的核心功能：

1. 用户管理：管理员（预设账户）与顾客（需注册）的登录、密码修改、账户切换。
2. 商品管理：管理员对商品的增删改查，顾客对商品的浏览、精确/模糊查询。
3. 购物车操作：登录顾客对购物车的添加、删除、修改数量及查看。
4. 订单流程：生成订单（库存校验）、订单状态流转（已付款→待发货→已发货→已签收）、订单查询/修改/删除。
5. 促销活动：限时折扣、满减活动、叠加优惠，促销标签显示及有效期管理。
6. 数据持久化：用户、商品、购物车、订单信息存储至文件，操作与文件同步。
7. **实验步骤**
8. **实验思路**

采用分层设计，将系统划分为用户模块、商品模块、购物车模块、订单模块、促销模块及数据持久化模块，各模块通过类与对象交互实现总体功能。

1. **关键数据结构与类定义**
   1. 用户模块：

User类：表示用户（管理员 / 顾客），包含用户名、密码、是否为管理员 / 访客等属性。

UserInfo结构体：存储顾客的购物车与订单文件路径，用于账户映射表。

System类（主控类）：管理用户登录、账户映射表（unordered\_map<string, UserInfo>），协调各模块功能。

* 1. 商品模块

Goods类：表示商品，包含名称、价格、库存、ID、描述、促销标签（promotion\_tag结构体）等属性。

GoodsManager类：管理商品集合（set<Goods>），支持增删改查、模糊搜索（基于 Jaro-Winkler 相似度算法）。

* 1. 购物车模块

ShoppingList类：管理购物车（map<uint32\_t, uint32\_t>，键为商品 ID，值为数量），支持添加、删除、修改数量及查看。

* 1. 订单模块

OrderItem结构体：表示订单项（商品 ID、名称、价格、数量、促销标签）。

Order类：表示订单，包含订单 ID、时间、状态、收件信息、订单项列表、总价等属性，支持支付、取消、状态更新。

OrderManager类：管理订单列表（vector<Order>），支持订单的加载、保存、查询、删除。

* 1. 促销模块

promotion\_tag结构体：记录促销类型（折扣/满减）、力度、有效期、是否可叠加等信息。

促销计算逻辑嵌入OrderManager::update\_total()方法，根据商品促销是否叠加计算（先折扣后满减），不叠加只计算折扣。

* 1. 数据持久化模块

文件存储格式：配置文件（INI）、用户信息（CSV）、商品信息（CSV）、购物车（SL）、订单（OL）。

各管理器（GoodsManager、ShoppingList、OrderManager）在析构时自动保存数据至文件，启动时加载。

1. **主要算法流程**

登录流程：用户输入账号密码→验证管理员/顾客身份→切换活跃状态（管理员/当前用户/访客）。

商品模糊搜索：计算用户输入与商品名称/描述的Jaro相似度（阈值0.75），返回高相似度结果。

订单生成：校验购物车/商品列表中商品库存→生成订单ID→计算总价（考虑促销）→更新库存→支付成功则清空购物车对应商品。

促销计算：遍历订单项，先应用折扣（price \*= (1.0 - item.tag.discount / 100.0)），再应用满减（total -= (uint32\_t(item\_total / item.tag.spend) \* item.tag.reduction)）。

1. **实验结果**

**功能覆盖测试**

|  |  |  |  |
| --- | --- | --- | --- |
| **功能** | **测试用例** | **预期结果** | **实际结果** |
| 用户管理 | 管理员登录、管理员账号信息变更、用户注册（包括重名）、用户登录、用户密码修改 | 正确登录，注册成功，错误提示明确 | 与预期一致 |
| 商品管理 | 添加商品（管理员）、修改商品信息（管理员）、移除商品（管理员）、搜索商品 | 商品列表显示新增商品，修改后库存更新，删除后商品消失 | 与预期一致 |
| 购物车操作 | 加入购物车、减少车内商品数量、增加车内商品数量、移除车内商品、清空购物车 | 购物车显示数量变化，删除后商品消失 | 与预期一致 |
| 订单管理 | 从购物车内下单、直接下单某一种商品、历史订单浏览、订单信息变更、删除已完成的订单 | 库存不足提示，地址修改成功，取消后库存恢复，已签收订单可删除 | 与预期一致 |
| 促销活动 | 管理员设置促销活动、下订单（活动过期、没过期） | 有效期内显示打折和满减标签，计算总价正确 | 与预期一致 |
| 数据持久化 | 上述测试结束后退出系统查看文件保存情况、全部删除后重新运行，退出后再次查看 | 文件内容与系统数据一致，没有文件时不会报错，而是创建 | 与预期一致 |

测试细节见附录。

1. **实验总结**
2. **对面向对象设计的理解**

通过本次实验，深刻体会到面向对象设计的核心——封装、继承、多态。

封装：所有类数据成员设为private，仅通过public方法访问（如Goods::get\_info()获取商品详情）。

继承：虽未显式使用继承，但是理解了管理员是用户这种“is a”的关系。

多态：主要是编译时多态，如get\_input函数通过模板的使用，达成了对不同类型输入的支持。

1. **对软件开发流程的认识**

系统开发需遵循需求分析→设计→实现→测试→优化的流程。

例如初期仅实现内存存储，后期通过文件读写实现持久化，避免过早陷入复杂 IO 操作。测试需覆盖功能点、边界条件及集成场景（如修改商品后验证购物车与订单的一致性），确保系统健壮性。

1. **改进方向**

用户体验和性能优化：选用更低时间复杂度的算法，更快地响应用户操作。

扩展功能：支持顾客评价、管理员统计（如商品销量排行），增强系统实用性。

**附录 测试样例与结果**

测试方法：使用freopen(“test.txt”,”r”,stdin);freopen(“output.txt”,”w”,stdout);

// 由于用到了标准io，要把c++的取消同步流先注释掉

**数据：**

help

li su

admin

3434

li su

admin

admin

cp

admin

newadmin

ca

cp

admin

123456

123456

lo

sp

user1

sp

user

user

user

li

user

user

cp

user

123

123

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

vig

shg

paper

adg

rmg

upg

ca su

admin

123456

adg

computer

5000

50

a kind of personal device

n

upg

0

vig

upg

Computer

6000

100

y

y

10

y

1000

30

y

vig

rmg

0

rmg

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

li

user

123

vis

vig

ads

11

500

vis

sbs

11

100

vis

sts

11

600

vis

sts

11

0

n

vis

ads

1

2

rms

1

rms

0

cls

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ads

1

1

plo

name

address

12323432

y

y

buy 11 200

name1

address1

123234321

y

cho

27

name2

add2

1234311

rmo

27

vio

sho

25

rmo

25

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ca su

admin

123456

upg

11

y

y

25

y

100

20

y

ca

user

123

buy 11 500

name5

add5

6278362

y

ads

1

75

ads

11

100

plo

name6

add6

666666

y

vio

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

exit

**输出：**

System has been initialized.

Welcome to shopping system! Enter "help" for more information.

>>> Available commands:

help: show help information

exit: exit the system

li: login

li su: login as admin

lo: logout

sp: sign up

cp: change password

ca: change account

ca su: change account as admin

vig: view available goods

shg: search for goods

ads: add a kind of goods to shopping list

sbs: subtract amount of a kind of goods in shopping list

sts: set amount of a kind of goods in shopping list

rms: remove a kind of goods from shopping list

cls: clear shopping list

vis: view shopping list

buy: buy [id] [amount]: place an order for specific goods

plo: place an order

vio: check order status

sho: search for orders

rmo: remove an order(if possible)

clo: cancel an order(if possible)

cho: change receiver, address and phone of an order

>>> Please enter your username: Please enter your password: Wrong username or password. Please try again.

>>> Please enter your username: Please enter your password: Wrong username or password. Please try again.

>>> You are not logged in. Please login first.

>>> Unknown command: admin

Enter "help" to show available commands.

>>> Unknown command: newadmin

Enter "help" to show available commands.

>>> You are not logged in. Please login first.

>>> You are not logged in. Please login first.

>>> Unknown command: admin

Enter "help" to show available commands.

>>> Unknown command: 123456

Enter "help" to show available commands.

>>> Unknown command: 123456

Enter "help" to show available commands.

>>> You are not logged in.

>>> Please enter your username: User already registered. Please try another username.

>>> Please enter your username: Please enter your password: Please cofirm your password: Sign up successful.

>>> Please enter your username: Please enter your password: Filename has been changed to data/user.sl

Filename has been changed to data/user.ol

Login successful.

>>> Please enter your old password: Please enter your new password: Please cofirm your new password: Password changed successfully.

>>> Unknown command: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter "help" to show available commands.

>>> Available goods:

----------------------------------------------------------------

name: apple

price: 1.5

stock: 76

id: 1

description: A round fruit

promotion:

discount: %80 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are stackable

----------------------------------------------------------------

name: banana

price: 0.5

stock: 50

id: 2

description: A long curved fruit

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: orange

price: 2

stock: 75

id: 3

description: A round citrus fruit

promotion:

none

----------------------------------------------------------------

name: grape

price: 3

stock: 619

id: 4

description: A small sweet fruit

promotion:

none

----------------------------------------------------------------

name: pear

price: 2.5

stock: 120

id: 5

description: A juicy fruit

promotion:

none

----------------------------------------------------------------

name: pencil

price: 1.25

stock: 9500

id: 9

description: normal writing tool

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 150 minus 25

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: paper

price: 0.25

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

----------------------------------------------------------------

>>> Please enter the keyword to search for: ----------------------------------------------------------------

name: apple

price: 1.5

stock: 76

id: 1

description:A round fruit

promotion:

discount: %80 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are stackable

----------------------------------------------------------------

name: pear

price: 2.5

stock: 120

id: 5

description:A juicy fruit

promotion:

none

----------------------------------------------------------------

name: paper

price: 0.25

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

----------------------------------------------------------------

>>> You are not authorized to add goods.

>>> You are not authorized to remove goods.

>>> You are not authorized to update goods.

>>> Logout successful.

Please enter your username: Please enter your password: Login successful.

Welcome admin!

>>> Please enter the name of the goods: Please enter the price of the goods: Please enter the stock of the goods: Please enter the description of the goods: Do you want to add a promotion tag? (y/n): Goods added successfully.

>>> Please enter the id of the goods to update: Invalid input. Please try again.

>>> Available goods:

----------------------------------------------------------------

name: apple

price: 1.5

stock: 76

id: 1

description: A round fruit

promotion:

discount: %80 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are stackable

----------------------------------------------------------------

name: banana

price: 0.5

stock: 50

id: 2

description: A long curved fruit

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: orange

price: 2

stock: 75

id: 3

description: A round citrus fruit

promotion:

none

----------------------------------------------------------------

name: grape

price: 3

stock: 619

id: 4

description: A small sweet fruit

promotion:

none

----------------------------------------------------------------

name: pear

price: 2.5

stock: 120

id: 5

description: A juicy fruit

promotion:

none

----------------------------------------------------------------

name: pencil

price: 1.25

stock: 9500

id: 9

description: normal writing tool

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 150 minus 25

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: paper

price: 0.25

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

----------------------------------------------------------------

name: computer

price: 0

stock: 50

id: 17

description: a kind of personal device

promotion:

none

----------------------------------------------------------------

>>> Please enter the id of the goods to update: Invalid input. Please try again.

>>> Unknown command: 6000

Enter "help" to show available commands.

>>> Unknown command: 100

Enter "help" to show available commands.

>>> >>> Unknown command: y

Enter "help" to show available commands.

>>> Unknown command: y

Enter "help" to show available commands.

>>> Unknown command: 10

Enter "help" to show available commands.

>>> >>> Unknown command: y

Enter "help" to show available commands.

>>> Unknown command: 1000

Enter "help" to show available commands.

>>> Unknown command: 30

Enter "help" to show available commands.

>>> Unknown command: y

Enter "help" to show available commands.

>>> Available goods:

----------------------------------------------------------------

name: apple

price: 1.5

stock: 76

id: 1

description: A round fruit

promotion:

discount: %80 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are stackable

----------------------------------------------------------------

name: banana

price: 0.5

stock: 50

id: 2

description: A long curved fruit

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: orange

price: 2

stock: 75

id: 3

description: A round citrus fruit

promotion:

none

----------------------------------------------------------------

name: grape

price: 3

stock: 619

id: 4

description: A small sweet fruit

promotion:

none

----------------------------------------------------------------

name: pear

price: 2.5

stock: 120

id: 5

description: A juicy fruit

promotion:

none

----------------------------------------------------------------

name: pencil

price: 1.25

stock: 9500

id: 9

description: normal writing tool

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 150 minus 25

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: paper

price: 0.25

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

----------------------------------------------------------------

name: computer

price: 0

stock: 50

id: 17

description: a kind of personal device

promotion:

none

----------------------------------------------------------------

>>> Please enter the id of the goods to remove: Failed to remove goods. No such kind of goods.

>>> Please enter the id of the goods to remove: Goods removed successfully.

>>> Unknown command: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter "help" to show available commands.

>>> You are already logged in as admin.

>>> Unknown command: user

Enter "help" to show available commands.

>>> Unknown command: 123

Enter "help" to show available commands.

>>> Shopping list is empty.

>>> Available goods:

----------------------------------------------------------------

name: apple

price: 1.5

stock: 76

id: 1

description: A round fruit

promotion:

discount: %80 off

end time: 2025-05-21 23:59:59

reduction: every 100 minus 10

end time: 2025-05-21 23:59:59

discounts are stackable

----------------------------------------------------------------

name: orange

price: 2

stock: 75

id: 3

description: A round citrus fruit

promotion:

none

----------------------------------------------------------------

name: grape

price: 3

stock: 619

id: 4

description: A small sweet fruit

promotion:

none

----------------------------------------------------------------

name: pear

price: 2.5

stock: 120

id: 5

description: A juicy fruit

promotion:

none

----------------------------------------------------------------

name: pencil

price: 1.25

stock: 9500

id: 9

description: normal writing tool

promotion:

discount: %90 off

end time: 2025-05-21 23:59:59

reduction: every 150 minus 25

end time: 2025-05-21 23:59:59

discounts are not stackable

----------------------------------------------------------------

name: paper

price: 0.25

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

----------------------------------------------------------------

name: computer

price: 0

stock: 50

id: 17

description: a kind of personal device

promotion:

none

----------------------------------------------------------------

>>> Please enter the id of the goods to add: Please enter the amount of the goods to add: success

>>> Shopping list:

----------------------------------------------------------------

name: paper

price: 0.250000

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

amount : 500

----------------------------------------------------------------

>>> Please enter the id of the goods to subtract: Please enter the amount of the goods to subtract: success

>>> Shopping list:

----------------------------------------------------------------

name: paper

price: 0.250000

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

amount : 400

----------------------------------------------------------------

>>> Please enter the id of the goods to set: Please enter the amount of the goods to set: success

>>> Shopping list:

----------------------------------------------------------------

name: paper

price: 0.250000

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

amount : 600

----------------------------------------------------------------

>>> Please enter the id of the goods to set: Please enter the amount of the goods to set: Are you sure want to remove this item from shopping list? (y/n):

Operation cancelled.

>>> Shopping list:

----------------------------------------------------------------

name: paper

price: 0.250000

stock: 7000

id: 11

description: a kind of material

promotion:

discount: %20 off

end time: 2025-05-30 19:35:58

reduction: every 100 minus 10

end time: 2025-05-30 19:36:04

discounts are stackable

amount : 600

----------------------------------------------------------------

>>> Please enter the id of the goods to add: Please enter the amount of the goods to add: success

>>> Please enter the id of the goods to remove: Item 1 has been removed from shopping list.

>>> Please enter the id of the goods to remove: Item 0 not found in shopping list.

>>> Shoppinglist has been cleared successfully.

>>> Unknown command: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter "help" to show available commands.

>>> Please enter the id of the goods to add: Please enter the amount of the goods to add: success

>>> Please enter the receiver's name: Please enter the receiver's address: Please enter the receiver's phone number: Do you want to remove these items from shopping list? (y/n):

Shoppinglist has been cleared successfully.

Order summary:1.5

Order created successfully. Please pay for it. Y/N:

Payment successful.

>>> Please enter the receiver's name: Please enter the receiver's address: Please enter the receiver's phone number: Order summary:50

Order created successfully. Please pay for it. Y/N:

Payment successful.

>>> Please enter the id of the order to change: No such order.

>>> Unknown command: name2

Enter "help" to show available commands.

>>> Unknown command: add2

Enter "help" to show available commands.

>>> Unknown command: 1234311

Enter "help" to show available commands.

>>> Please enter the id of the order to remove: No such order.

>>> ================================================================

id: 30

order\_time: 2025-06-07 19:05:21

paid\_time: 2025-06-07 19:05:21

receiver: name

address: address

phone: 12323432

status: PAID

goods\_id: 1

name: apple

price: 1.5

amount: 1

total: 1.5

================================================================

id: 31

order\_time: 2025-06-07 19:05:21

paid\_time: 2025-06-07 19:05:21

receiver: name1

address: address1

phone: 123234321

status: PAID

goods\_id: 11

name: paper

price: 0.25

amount: 200

total: 50

================================================================

>>> Please enter the id of the order to check: ================================================================

>>> Please enter the id of the order to remove: No such order.

>>> Unknown command: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter "help" to show available commands.

>>> Logout successful.

Please enter your username: Please enter your password: Login successful.

Welcome admin!

>>> Please enter the id of the goods to update: Please enter the new name of the goods(Press enter to skip): Please enter the new price of the goods(Press enter to skip): Please enter the new stock of the goods(Press enter to skip): Please enter the new description of the goods(Press enter to skip): Do you want to update the promotion tag? (y/n): Change discount? (y/n): Please enter new discount(Press enter to cancel discount): Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): Change reduction? (y/n): Please enter new reduction(Press enter to cancel reduction):

spend: reduction: Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): stackable? (y/n): Goods updated successfully.

>>> Logout successful.

Please enter your username: Please enter your password: Filename has been changed to data/user.sl

Filename has been changed to data/user.ol

Login successful.

>>> Please enter the receiver's name: Please enter the receiver's address: Please enter the receiver's phone number: Order summary:93.75

Order created successfully. Please pay for it. Y/N:

Payment successful.

>>> Please enter the id of the goods to add: Please enter the amount of the goods to add: success

>>> Please enter the id of the goods to add: Please enter the amount of the goods to add: success

>>> Please enter the receiver's name: Please enter the receiver's address: Please enter the receiver's phone number: Do you want to remove these items from shopping list? (y/n):

Shoppinglist has been cleared successfully.

Order summary:131.25

Order created successfully. Please pay for it. Y/N:

Payment failed. Order cancelled.

>>> Unknown command: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter "help" to show available commands.

>>> System has exited.

**附录 源程序清单**

**main.cpp**

#include "config.h"

#include "system.h"

#include "user.h"

#include "goods.h"

#include "functions.h"

#include "shoppinglist.h"

int main()

{

    Config config = Config("./config.ini");

    System system = System(config);

    system.run("Welcome to shopping system! Enter \"help\" for more information.");

    return 0;

}

**config.h**

#ifndef CONFIG\_H

#define CONFIG\_H

#pragma once

#include <string>

#include <unordered\_map>

using std::string;

using std::unordered\_map;

string strip(const string &str);

class Config

{

private:

    string filepath;

    unordered\_map<string, string> config;

    const string &get(const string &key) const;

    void set(const string &key, const string &value);

    void save() const;

    bool change;

public:

    Config();

    Config(const string &filepath);

    ~Config();

    const string &get\_admin\_username() const { return get("admin\_username"); }

    const string &get\_admin\_password() const { return get("admin\_pwd"); }

    const string &get\_accounts\_path() const { return get("accounts\_path"); }

    const string &get\_goods\_path() const { return get("goods\_path"); }

    const string &get\_goods\_id() const { return get("goods\_id"); }

    const string &get\_shopping\_list\_path() const { return get("shopping\_list\_path"); }

    const string &get\_order\_list\_path() const { return get("order\_list\_path"); }

    const string &get\_order\_id() const { return get("order\_id"); }

    void set\_admin\_username(const string &value) { set("admin\_username", value); }

    void set\_admin\_password(const string &value) { set("admin\_pwd", value); }

    void set\_accounts\_path(const string &value) { set("accounts\_path", value); }

    void set\_goods\_path(const string &value) { set("goods\_path", value); }

    void set\_goods\_id(const string &value) { set("goods\_id", value); }

    void set\_shopping\_list\_path(const string &value) { set("shopping\_list\_path", value); }

    void set\_order\_list\_path(const string &value) { set("order\_list\_path", value); }

    void set\_order\_id(const string &value) { set("order\_id", value); }

};

#endif // CONFIG\_H

**config.cpp**

#include "config.h"

#include "functions.h"

#include <fstream>

#include <string>

using std::ifstream;

using std::ofstream;

using std::string;

using std::to\_string;

const char endl = '\n';

Config::Config() : Config("./config.ini") {}

Config::Config(const string &filepath)

{

    change = false;

    this->filepath = filepath;

    ifstream file(filepath, ifstream::in);

    if (!file.is\_open())

    {

        // Handle error

        config["admin\_username"] = "";

        config["admin\_pwd"] = "";

        config["accounts\_path"] = "";

        config["goods\_path"] = "";

        config["goods\_id"] = to\_string(0);

        config["shopping\_list\_path"] = "./data/shopping\_list.sl";

        config["order\_list\_path"] = "./data/order\_list.ol";

        config["order\_id"] = to\_string(0);

        change = true;

        // 不存在则创建文件

        ofstream file(filepath, ofstream::out);

        file.close();

        return;

    }

    string line;

    int epos;

    // admin

    getline(file, line); // admin段，跳过

    getline(file, line); // 管理员username

    line = strip(line);

    epos = line.find('=');

    config["admin\_username"] = line.substr(epos + 1);

    getline(file, line); // 管理员password

    line = strip(line);

    epos = line.find('=');

    config["admin\_pwd"] = line.substr(epos + 1);

    // accounts\_path

    getline(file, line); // accounts\_path段，跳过

    getline(file, line); // 数据库路径

    line = strip(line);

    epos = line.find('=');

    config["accounts\_path"] = line.substr(epos + 1);

    // goods\_path

    getline(file, line); // goods\_path段，跳过

    getline(file, line); // 商品信息文件路径

    line = strip(line);

    epos = line.find('=');

    config["goods\_path"] = line.substr(epos + 1);

    // goods\_id

    getline(file, line); // goods\_id段，跳过

    getline(file, line); // 商品id

    line = strip(line);

    epos = line.find('=');

    config["goods\_id"] = line.substr(epos + 1);

    // shopping\_list\_path

    getline(file, line); // shopping\_list\_path段，跳过

    getline(file, line); // 购物清单文件路径

    line = strip(line);

    epos = line.find('=');

    config["shopping\_list\_path"] = line.substr(epos + 1);

    // order\_list\_path

    getline(file, line); // order\_list\_path段，跳过

    getline(file, line); // 订单清单文件路径

    line = strip(line);

    epos = line.find('=');

    config["order\_list\_path"] = line.substr(epos + 1);

    // order\_id

    getline(file, line); // order\_id段，跳过

    getline(file, line); // 订单id

    line = strip(line);

    epos = line.find('=');

    config["order\_id"] = line.substr(epos + 1);

    file.close();

}

Config::~Config()

{

    if (change)

        save();

}

const string &Config::get(const string &key) const

{

    auto const &it = config.find(key);

    static const string empty\_str = "";

    if (it == config.cend())

        return empty\_str;

    return it->second;

}

void Config::set(const string &key, const string &value)

{

    change = true;

    config[key] = value;

}

void Config::save() const

{

    ofstream file(filepath, ofstream::out);

    if (!file.is\_open())

    {

        // Handle error

        return;

    }

    file << "[admin]\n"

         << "admin\_username=" << get\_admin\_username() << endl

         << "admin\_pwd=" << get\_admin\_password() << endl

         << "[accounts\_path]" << endl

         << "accounts\_path=" << get\_accounts\_path() << endl

         << "[goods\_path]" << endl

         << "goods\_path=" << get\_goods\_path() << endl

         << "[goods\_id]" << endl

         << "goods\_id=" << get\_goods\_id() << endl

         << "[shopping\_list\_path]" << endl

         << "shopping\_list\_path=" << get\_shopping\_list\_path() << endl

         << "[order\_list\_path]" << endl

         << "order\_list\_path=" << get\_order\_list\_path() << endl

         << "[order\_id]" << endl

         << "order\_id=" << get\_order\_id() << endl;

    file.close();

}

**user.h**

#ifndef USER\_H

#define USER\_H

#pragma once

#include "config.h"

#include <string>

#include <iostream>

using std::cout;

using std::string;

class User

{

private:

    bool is\_admin;   // 是否为管理员

    bool is\_guest;   // 是否为访客

    string username; // 用户名

    string pwd;      // 密码

public:

    User();

    User(bool is\_admin\_user, bool is\_guest\_user);

    User(bool is\_admin\_user, bool is\_guest\_user, const string &un, const string &pd);

    void set\_username(const string &new\_username) { username = new\_username; } // 修改用户名

    void set\_pwd(const string &new\_pwd) { pwd = new\_pwd; }                     // 修改密码

    const string &get\_username() const { return username; }                    // 获取用户名

    const string &get\_pwd() const { return pwd; }                              // 获取密码

};

#endif // USER\_H

**user.cpp**

#include "user.h"

#include <string>

using std::string;

User::User() : User(false, false) {}

User::User(bool is\_admin\_user, bool is\_guest\_user) : User(is\_admin\_user, is\_guest\_user, "", "") {}

User::User(bool is\_admin\_user, bool is\_guest\_user, const string &un, const string &pd)

{

    is\_admin = is\_admin\_user;

    is\_guest = is\_admin\_user ? false : is\_guest\_user;

    username = un;

    pwd = pd;

}

**goods.h**

#ifndef GOODS\_H

#define GOODS\_H

#pragma once

#include "config.h"

#include "promotion.h"

#include <string>

#include <set>

#include <fstream>

#include <cstdint>

using std::ostream;

using std::set;

using std::string;

class Goods

{

private:

    string name;

    float price;

    uint32\_t stock;          // 库存

    const uint32\_t id;       // 区分不同商品的唯一标识，不可修改

    string description;      // 商品描述

    promotion\_tag promo\_tag; // 促销标签

public:

    Goods(Config &cfg,

          const string &name,

          float price,

          uint32\_t stock = 0,

          const string &description = "",

          const promotion\_tag &promo\_tag = promotion\_tag());

    Goods(const string &name,

          float price,

          uint32\_t stock,

          uint32\_t id,

          const string &description,

          const promotion\_tag &promo\_tag);

    const string &get\_name() const { return name; }

    float get\_price() const { return price; }

    uint32\_t get\_stock() const { return stock; }

    const uint32\_t get\_id() const { return id; }

    const promotion\_tag &get\_promotion() const { return promo\_tag; }

    const string &get\_description() const { return description; }

    const string &get\_info() const;

    void set\_name(const string &new\_name) { name = new\_name; }

    void set\_price(float new\_price) { price = new\_price; }

    void set\_stock(uint32\_t new\_stock) { stock = new\_stock; }

    void set\_description(const string &new\_description) { description = new\_description; }

    // 重载<，为了使用set容器

    bool operator<(const Goods &other) const { return id < other.id; }

};

class GoodsManager

{

private:

    set<Goods> goods\_set; // 商品集合

    string filepath;

    bool change;

public:

    GoodsManager();

    GoodsManager(const string &path);

    ~GoodsManager();

    void list\_goods() const;

    void add\_goods(const Goods &good);

    bool remove\_goods(uint32\_t id);

    bool update\_goods(uint32\_t id,

                      const string &name, bool update\_name,

                      float price, bool update\_price,

                      int stock, bool update\_stock,

                      const string &description, bool update\_description,

                      const promotion\_tag &promo\_tag, bool update\_promo\_tag);

    void search\_goods(const string &words) const;

    const string &search\_goods(uint32\_t id) const;

    const Goods &search\_goods(uint32\_t id, bool) const;

    void save(const string &path) const;

};

#endif // GOODS\_H

**goods.cpp**

#include "goods.h"

#include "functions.h"

#include "config.h"

#include "promotion.h"

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <cstdint>

using std::cout;

using std::getline;

using std::ifstream;

using std::ofstream;

using std::stof;  // 字符串转浮点数

using std::stoul; // 字符串转无符号整数

using std::string;

using std::to\_string; // 数字转字符串

using std::vector;

const char endl = '\n';

/\* 商品ID分配函数 \*/

uint32\_t auto\_id(Config &cfg)

{

    uint32\_t id = stoul(cfg.get\_goods\_id());

    cfg.set\_goods\_id(to\_string(id + 1));

    return id;

}

/\* 新建商品用法 \*/

Goods::Goods(Config &cfg,

             const string &name,

             float price,

             uint32\_t stock,

             const string &description,

             const promotion\_tag &promo\_tag)

    : id(auto\_id(cfg)), // const成员变量初始化在参数列表中进行

      name(name),       // 其他变量顺便也写在参数列表里

      price(price),

      stock(stock),

      description(description),

      promo\_tag()

{

}

/\* 读取已有商品用法 \*/

Goods::Goods(const string &name,

             float price,

             uint32\_t stock,

             uint32\_t id,

             const string &description,

             const promotion\_tag &promo\_tag)

    : name(name), // 其他变量顺便也写在参数列表里

      price(price),

      stock(stock),

      id(id), // const成员变量初始化在参数列表中进行

      description(description),

      promo\_tag(promo\_tag)

{

}

/\* GoodsManager无参数构造函数，委托GoodsManager(const string &path)构造函数 \*/

GoodsManager::GoodsManager() : GoodsManager::GoodsManager("./data/goods.csv") {}

/\* GoodsManager有参构造 \*/

GoodsManager::GoodsManager(const string &path)

{

    change = false; // 初始化未修改标志

    filepath = path;

    ifstream file(path, ifstream::in);

    if (!file.is\_open())

    {

        goods\_set.clear();

        // 不存在文件，创建新文件

        ofstream file(path, ofstream::out);

        file.close();

        return;

    }

    string line;

    vector<string> tmp;  // 存放一行里面的信息

    getline(file, line); // 跳过第一行

    int start, end;

    while (getline(file, line)) // 读取每一行

    {

        line = strip(line); // 去除换行符

        start = 0;

        end = line.find(',', start);

        for (int i = 0; i < 13; ++i)

        {

            tmp.push\_back(line.substr(start, end - start));

            start = end + 1;

            end = line.find(',', start);

        }

        promotion\_tag tmp\_tag = {

            stoul(tmp[5]),

            stoul(tmp[6]),

            stoul(tmp[7]),

            stoul(tmp[8]),

            stoul(tmp[9]),

            stoul(tmp[10]),

            strtoll(tmp[11].c\_str(), nullptr, 10),

            strtoll(tmp[12].c\_str(), nullptr, 10),

        };

        goods\_set.insert(Goods(tmp[0], stof(tmp[1]), stoul(tmp[2]), stoul(tmp[3]), tmp[4], tmp\_tag)); // 插入商品

        tmp.clear();

    }

}

GoodsManager::~GoodsManager()

{

    if (change)

        save(filepath);

}

const string &Goods::get\_info() const

{

    static string goods\_info;

    goods\_info.clear();

    goods\_info = "name: " + name + endl +

                 "price: " + to\_string(price) + endl +

                 "stock: " + to\_string(stock) + endl +

                 "id: " + to\_string(id) + endl +

                 "description: " + description + endl +

                 "promotion: " + endl;

    if (promo\_tag.is\_discount)

        goods\_info += "    discount: %" + to\_string(promo\_tag.discount) + " off" + endl +

                      "    end time: " + strtime(promo\_tag.discount\_end\_time) + endl;

    if (promo\_tag.is\_reduction)

        goods\_info += "    reduction: every " + to\_string(promo\_tag.spend) +

                      " minus " + to\_string(promo\_tag.reduction) + endl +

                      "    end time: " + strtime(promo\_tag.reduction\_end\_time) + endl;

    if (!promo\_tag.is\_discount && !promo\_tag.is\_reduction)

        goods\_info += "    none\n";

    if (promo\_tag.is\_discount && promo\_tag.is\_reduction)

    {

        if (promo\_tag.both)

            goods\_info += "    discounts are stackable";

        else

            goods\_info += "    discounts are not stackable";

    }

    return goods\_info;

}

/\* 添加商品 \*/

void GoodsManager::add\_goods(const Goods &goods)

{

    goods\_set.insert(goods);

    change = true; // 修改标志置为true

}

/\* 打印商品信息 \*/

void GoodsManager::list\_goods() const

{

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

    for (auto const &goods : goods\_set)

    {

        cout << "name:        " << goods.get\_name() << endl

             << "price:       " << goods.get\_price() << endl

             << "stock:       " << goods.get\_stock() << endl

             << "id:          " << goods.get\_id() << endl

             << "description: " << goods.get\_description() << endl

             << "promotion: " << endl;

        if (goods.get\_promotion().is\_discount)

            cout << "    discount: %" << to\_string(goods.get\_promotion().discount) << " off\n"

                 << "    end time: " + strtime(goods.get\_promotion().discount\_end\_time) + endl;

        if (goods.get\_promotion().is\_reduction)

            cout << "    reduction: every " << to\_string(goods.get\_promotion().spend)

                 << " minus " << to\_string(goods.get\_promotion().reduction) << endl

                 << "    end time: " + strtime(goods.get\_promotion().reduction\_end\_time) + endl;

        if (!goods.get\_promotion().is\_discount && !goods.get\_promotion().is\_reduction)

            cout << "    none\n";

        if (goods.get\_promotion().is\_discount && goods.get\_promotion().is\_reduction)

        {

            if (goods.get\_promotion().both)

                cout << "    discounts are stackable\n";

            else

                cout << "    discounts are not stackable\n";

        }

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

    }

}

/\* 保存商品信息到文件 \*/

void GoodsManager::save(const string &path) const

{

    ofstream file(path, ofstream::out);

    if (!file.is\_open())

    {

        return;

    }

    file << "name,price,stock,id,description,"

         << "is\_discount,is\_reduction,both,discount,spend,reduction,"

         << "discount\_end\_time,reduction\_end\_time" << endl;

    for (auto const &goods : goods\_set)

    {

        file << goods.get\_name() << ','

             << goods.get\_price() << ','

             << goods.get\_stock() << ','

             << goods.get\_id() << ','

             << goods.get\_description() << ','

             << goods.get\_promotion() << endl;

    }

    file.close();

}

bool GoodsManager::remove\_goods(uint32\_t id)

{

    for (auto it = goods\_set.begin(); it != goods\_set.end(); ++it)

        if (it->get\_id() == id) // 这里删除完之后直接返回，不存在迭代器失效问题

        {

            goods\_set.erase(it);

            change = true;

            change = true; // 修改标志置为true

            return true;

        }

    return false;

}

bool GoodsManager::update\_goods(uint32\_t id,

                                const string &name, bool update\_name,

                                float price, bool update\_price,

                                int stock, bool update\_stock,

                                const string &description, bool update\_description,

                                const promotion\_tag &promo\_tag, bool update\_promo\_tag)

{

    auto it = goods\_set.begin();

    for (; it != goods\_set.end(); ++it)

        if (it->get\_id() == id)

            break;

    if (it == goods\_set.cend())

        return false;

    auto name\_ = update\_name ? name : it->get\_name();

    auto price\_ = update\_price ? price : it->get\_price();

    auto stock\_ = update\_stock ? stock : it->get\_stock();

    auto description\_ = update\_description ? description : it->get\_description();

    auto id\_ = it->get\_id();

    auto promo\_tag\_ = update\_promo\_tag ? promo\_tag : it->get\_promotion();

    goods\_set.erase(it);

    goods\_set.insert(Goods(name\_, price\_, stock\_, id\_, description\_, promo\_tag\_));

    change = true; // 修改标志置为true

    return true;

}

void GoodsManager::search\_goods(const string &words) const

{

    bool is\_find = false;

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

    for (auto const &goods : goods\_set)

        if (goods.get\_name().find(words) != string::npos ||

            goods.get\_description().find(words) != string::npos) // 名称或描述中直接包含搜索词

        {

            is\_find = true;

            cout << "name:        " << goods.get\_name() << endl

                 << "price:       " << goods.get\_price() << endl

                 << "stock:       " << goods.get\_stock() << endl

                 << "id:         " << goods.get\_id() << endl

                 << "description: " << goods.get\_description() << endl

                 << "promotion: " << endl;

            if (goods.get\_promotion().is\_discount)

                cout << "    discount: %" << to\_string(goods.get\_promotion().discount) << " off\n"

                     << "    end time: " + strtime(goods.get\_promotion().discount\_end\_time) + endl;

            if (goods.get\_promotion().is\_reduction)

                cout << "    reduction: every " << to\_string(goods.get\_promotion().spend)

                     << " minus " << to\_string(goods.get\_promotion().reduction) << endl

                     << "    end time: " + strtime(goods.get\_promotion().reduction\_end\_time) + endl;

            if (!goods.get\_promotion().is\_discount && !goods.get\_promotion().is\_reduction)

                cout << "    none\n";

            if (goods.get\_promotion().is\_discount && goods.get\_promotion().is\_reduction)

            {

                if (goods.get\_promotion().both)

                    cout << "    discounts are stackable\n";

                else

                    cout << "    discounts are not stackable\n";

            }

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

        }

        else // 启用模糊搜索

        {

            is\_find = true;

            auto simlarity1 = jaro\_similarity(words, goods.get\_name());

            auto simlarity2 = jaro\_similarity(words, goods.get\_description());

            if (simlarity1 > 0.75 || simlarity2 > 0.75)

            {

                cout << "name:       " << goods.get\_name() << endl

                     << "price:      " << goods.get\_price() << endl

                     << "stock:      " << goods.get\_stock() << endl

                     << "id:         " << goods.get\_id() << endl

                     << "description:" << goods.get\_description() << endl

                     << "promotion: " << endl;

                if (goods.get\_promotion().is\_discount)

                    cout << "    discount: %" << to\_string(goods.get\_promotion().discount) << " off\n"

                         << "    end time: " + strtime(goods.get\_promotion().discount\_end\_time) + endl;

                if (goods.get\_promotion().is\_reduction)

                    cout << "    reduction: every " << to\_string(goods.get\_promotion().spend)

                         << " minus " << to\_string(goods.get\_promotion().reduction) << endl

                         << "    end time: " + strtime(goods.get\_promotion().reduction\_end\_time) + endl;

                if (!goods.get\_promotion().is\_discount && !goods.get\_promotion().is\_reduction)

                    cout << "    none\n";

                if (goods.get\_promotion().is\_discount && goods.get\_promotion().is\_reduction)

                {

                    if (goods.get\_promotion().both)

                        cout << "    discounts are stackable\n";

                    else

                        cout << "    discounts are not stackable\n";

                }

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

            }

        }

    if (!is\_find)

        cout << "No goods found." << endl

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

}

const string &GoodsManager::search\_goods(uint32\_t id) const

{

    static string empty = "";

    bool is\_find = false;

    for (auto const &goods : goods\_set)

        if (goods.get\_id() == id)

            return goods.get\_info();

    return empty;

}

const Goods &GoodsManager::search\_goods(uint32\_t id, bool) const

{

    static Goods empty("", 0, 0, 0, "", promotion\_tag{});

    for (auto const &goods : goods\_set)

        if (goods.get\_id() == id)

            return goods;

    return empty;

}

**shoppinglist.h**

#ifndef SHOPPINGLIST\_H

#define SHOPPINGLIST\_H

#pragma once

#include "goods.h"

#include <string>

#include <map>

#include <cstdint>

using std::map;

using std::string;

class ShoppingList

{

private:

    map<uint32\_t, uint32\_t> shopping\_list; //<id, amount>

    string filename;

    const GoodsManager &goods\_manager;

    bool changed;

public:

    ShoppingList(const string &filename, const GoodsManager &goods\_manager);

    ~ShoppingList();

    const map<uint32\_t, uint32\_t> &get\_shopping\_list() const { return shopping\_list; }

    void set\_filename(const string &new\_filename); // 设置存档文件名

    void add(uint32\_t id, uint32\_t amount);

    void sub(uint32\_t id, uint32\_t amount);

    void set(uint32\_t id, uint32\_t amount);

    void remove(uint32\_t id);

    void clear();

    bool empty() const { return shopping\_list.empty(); }

    void load();       // 从文件中读取数据

    void save() const; // 将数据保存到文件中

    void list() const;

};

#endif // SHOPPINGLIST\_H

**shoppinglist.cpp**

#include "shoppinglist.h"

#include "goods.h"

#include <iostream>

#include <fstream>

#include <string>

#include <cstdint>

using std::cout;

using std::ifstream;

using std::ofstream;

using std::string;

const char endl = '\n';

ShoppingList::ShoppingList(const string &filename, const GoodsManager &goods\_manager)

    : filename(filename), changed(false), goods\_manager(goods\_manager)

{

    load();

}

void ShoppingList::load()

{

    shopping\_list.clear();

    ifstream in(filename, ifstream::in);

    if (!in.is\_open())

    {

        ofstream out(filename, ofstream::out); // 不存在则新建文件

        out.close();

        return;

    }

    int id, amount;

    while (in >> id >> amount)

        if (goods\_manager.search\_goods(id) != "") // 商品存在

            shopping\_list[id] = amount;

        else

            changed = true; // 商品不存在，标记为需要保存

    in.close();

}

ShoppingList::~ShoppingList()

{

    if (changed)

        save();

}

void ShoppingList::save() const

{

    ofstream out(filename, ofstream::out);

    if (!out.is\_open())

    {

        cout << "Failed to save shopping list to file " << filename << endl;

        return;

    }

    for (const auto &item : shopping\_list)

        out << item.first << ' ' << item.second << endl;

    out.close();

}

void ShoppingList::set\_filename(const string &new\_filename)

{

    if (changed)

        save();

    filename = new\_filename;

    changed = false;

    load();

    cout << "Filename has been changed to " << filename << endl;

}

void ShoppingList::add(uint32\_t id, uint32\_t amount)

{

    if (goods\_manager.search\_goods(id) == "") // 商品不存在

    {

        cout << "Item " << id << " not found in goods manager." << endl;

        return;

    }

    if (shopping\_list.find(id) == shopping\_list.end())

        shopping\_list[id] = amount;

    else

        shopping\_list[id] += amount;

    changed = true;

    cout << "success" << endl;

}

void ShoppingList::sub(uint32\_t id, uint32\_t amount)

{

    if (shopping\_list.find(id) == shopping\_list.end())

    {

        cout << "Item " << id << " not found in shopping list." << endl;

        return; // 购物车里没有直接返回

    }

    if (shopping\_list[id] <= amount)

        shopping\_list.erase(id); // 购物车里的数量小于等于要减的数量，直接删除

    else

        shopping\_list[id] -= amount;

    changed = true;

    cout << "success" << endl;

}

void ShoppingList::set(uint32\_t id, uint32\_t amount)

{

    if (shopping\_list.find(id) == shopping\_list.end())

    {

        cout << "Item " << id << " not found in shopping list." << endl;

        return; // 购物车里没有直接返回

    }

    if (amount == 0)

        shopping\_list.erase(id); // 要设置的数量为0，直接删除

    else

        shopping\_list[id] = amount;

    cout << "success" << endl;

    changed = true;

}

void ShoppingList::remove(uint32\_t id)

{

    if (shopping\_list.erase(id))

    {

        changed = true;

        cout << "Item " << id << " has been removed from shopping list." << endl;

    }

    else

        cout << "Item " << id << " not found in shopping list." << endl;

}

void ShoppingList::clear()

{

    shopping\_list.clear();

    changed = true;

    cout << "Shoppinglist has been cleared successfully." << endl;

}

void ShoppingList::list() const

{

    if (shopping\_list.empty())

        cout << "Shopping list is empty." << endl;

    else

    {

        cout << "Shopping list:" << endl

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

        for (const auto &item : shopping\_list) // 遍历购物车

            cout << goods\_manager.search\_goods(item.first) << endl

                 << "amount : " << item.second << endl

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

    }

}

**order.h**

#ifndef ORDER\_H

#define ORDER\_H

#pragma once

#include "goods.h"

#include "promotion.h"

#include "config.h"

#include <string>

#include <map>

#include <vector>

#include <ctime>

#include <cstdint>

using std::map;

using std::pair;

using std::string;

using std::time\_t;

using std::vector;

enum OrderStatus : int

{

    UNPLACED,  // 未下单

    UNPAID,    // 未支付

    PAID,      // 已支付

    UNSHIPPED, // 未发货

    SHIPPING,  // 运输中

    DELIVERED, // 已送达

    CANCELLED  // 已取消

};

struct OrderItem // 订单项，保存下单时商品的快照

{

    uint32\_t goods\_id;

    string name;

    float price;

    uint32\_t amount;

    promotion\_tag tag;

    OrderItem(uint32\_t id, const string &n, float p, uint32\_t a)

        : goods\_id(id), name(n), price(p), amount(a), tag() {}

    OrderItem(uint32\_t id, const string &n, float p, uint32\_t a, const promotion\_tag &t)

        : goods\_id(id), name(n), price(p), amount(a), tag(t) {}

    OrderItem(const Goods &g, uint32\_t a) : amount(a)

    {

        goods\_id = g.get\_id();

        name = g.get\_name();

        price = g.get\_price();

        tag = g.get\_promotion();

    }

};

class Order

{

private:

    uint32\_t order\_id;           // 订单号

    time\_t order\_time;           // 创建订单时间

    time\_t paid\_time;            // 支付时间

    string order\_receiver;       // 订单收件人

    string order\_address;        // 订单地址

    string order\_phone;          // 订单联系方式

    OrderStatus order\_status;    // 订单状态

    vector<OrderItem> item\_list; // 订单列表

    double order\_total;          // 订单总价

    bool sub\_stock;              // 是否已经扣除库存

    GoodsManager \*goods\_manager; // 商品管理器

private:

    static uint32\_t generate\_id(Config &cfg); // 生成订单号

    Order(GoodsManager \*gm,

          uint32\_t order\_id,

          time\_t order\_time,

          time\_t paid\_time,

          const string &receiver,

          const string &address,

          const string &phone,

          OrderStatus order\_status,

          const vector<OrderItem> &item\_list,

          double order\_total); // 私有构造函数，仅供内部使用

public:

    Order(GoodsManager \*gm,

          const string &receiver = "",

          const string &address = "",

          const string &phone = ""); // 创建新订单

    static Order load\_order(GoodsManager \*gm, const string &line); // 读取已有订单

    /\* 状态查询 \*/

    OrderStatus check\_status(bool) const { return order\_status; }

    const string &check\_status() const;

    uint32\_t get\_id() const { return order\_id; }

    const string &get\_order\_time() const;

    const string &get\_paid\_time() const;

    time\_t get\_order\_time(bool) const { return order\_time; };

    time\_t get\_paid\_time(bool) const { return paid\_time; };

    double get\_total() const { return order\_total; }

    const string &get\_receiver() const { return order\_receiver; }

    const string &get\_address() const { return order\_address; }

    const string &get\_phone() const { return order\_phone; }

    const vector<OrderItem> &get\_item\_list() const { return item\_list; }

    /\* 订单操作 \*/

    double update\_total();

    bool change\_receiver(const string &receiver);

    bool change\_address(const string &address);

    bool change\_phone(const string &phone);

    bool add\_goods(uint32\_t goods\_id, uint32\_t amount);

    bool remove\_goods(uint32\_t goods\_id);

    bool place\_order(Config &cfg);

    bool pay();

    bool cancel\_order();

    bool update\_status();

};

class OrderManager

{

private:

    vector<Order> order\_list;    // 订单列表

    string filename;             // 订单文件名

    GoodsManager &goods\_manager; // 商品管理器

    bool change;                 // 是否需要保存

public:

    OrderManager(const string &filename, GoodsManager &goods\_manager);

    ~OrderManager();

    void set\_filename(const string &new\_filename); // 设置存档文件名

    void remove\_order(uint32\_t order\_id);

    void add\_order(const Order &order);

    void view\_order(uint32\_t order\_id = 0) const;

    pair<vector<Order>::iterator, bool> find\_order(uint32\_t order\_id);

    void update\_all\_order\_status();

    void load();

    void save() const;

};

#endif // ORDER\_H

**order.cpp**

#include "order.h"

#include "goods.h"

#include "config.h"

#include <string>

#include <map>

#include <list>

#include <ctime>

#include <cstdlib>

#include <fstream>

#include <iostream>

#include <sstream>

#include <cstdint>

using std::cout;

using std::ifstream;

using std::istringstream;

using std::list;

using std::localtime;

using std::map;

using std::ofstream;

using std::stoul;

using std::string;

using std::time\_t;

using std::to\_string;

using std::vector;

const char endl = '\n';

uint32\_t Order::generate\_id(Config &cfg)

{

    uint32\_t id = stoul(cfg.get\_order\_id());

    cfg.set\_order\_id(to\_string(id + 1));

    return id;

}

Order::Order(GoodsManager \*gm, const string &receiver, const string &address, const string &phone)

    : order\_id(0),

      order\_time(0),

      paid\_time(0),

      order\_receiver(receiver),

      order\_address(address),

      order\_phone(phone),

      order\_status(OrderStatus::UNPLACED),

      item\_list(),

      order\_total(0),

      sub\_stock(false),

      goods\_manager(gm)

{

}

Order::Order(GoodsManager \*gm, uint32\_t order\_id, time\_t order\_time, time\_t paid\_time,

             const string &receiver, const string &address, const string &phone,

             OrderStatus order\_status, const vector<OrderItem> &item\_list, double order\_total)

    : order\_id(order\_id),

      order\_time(order\_time),

      paid\_time(paid\_time),

      order\_receiver(receiver),

      order\_address(address),

      order\_phone(phone),

      order\_status(order\_status),

      item\_list(item\_list),

      order\_total(order\_total),

      sub\_stock(true),

      goods\_manager(gm)

{

}

Order Order::load\_order(GoodsManager \*gm, const string &line)

{

    vector<string> tokens;

    vector<OrderItem> item\_list;

    istringstream ss(line);

    string tmp;

    // 读取并解析

    while (getline(ss, tmp, ','))

        tokens.push\_back(tmp);

    for (auto pos = 7; pos < tokens.size() - 1; pos += 4)

    {

        auto id = static\_cast<uint32\_t>(stoul(tokens[pos]));

        auto name = tokens[pos + 1];

        auto price = stof(tokens[pos + 2]);

        auto amount = static\_cast<uint32\_t>(stoul(tokens[pos + 3]));

        item\_list.push\_back(OrderItem(id, name, price, amount));

    }

    return Order(gm,

                 static\_cast<uint32\_t>(stoul(tokens[0])),

                 static\_cast<time\_t>(stoll(tokens[1])),

                 static\_cast<time\_t>(stoll(tokens[2])),

                 tokens[3],

                 tokens[4],

                 tokens[5],

                 static\_cast<OrderStatus>(stoi(tokens[6])),

                 item\_list,

                 stod(tokens.back()));

}

const string &Order::check\_status() const

{

    static vector<string> status\_str = {

        "UNPLACED", "UNPAID", "PAID", "UNSHIPPED", "SHIPPING", "DELIVERED", "CANCELLED"};

    return status\_str[order\_status];

}

const string &Order::get\_order\_time() const

{

    char buffer[64];

    std::strftime(buffer, 64, "%Y-%m-%d %H:%M:%S", localtime(&order\_time));

    static string str;

    str = buffer;

    return str;

}

const string &Order::get\_paid\_time() const

{

    char buffer[64];

    std::strftime(buffer, 64, "%Y-%m-%d %H:%M:%S", localtime(&paid\_time));

    static string str;

    str = buffer;

    return str;

}

bool Order::change\_receiver(const string &receiver)

{

    if (order\_status <= OrderStatus::UNSHIPPED)

    {

        order\_receiver = receiver;

        return true;

    }

    return false;

}

bool Order::change\_address(const string &address)

{

    if (order\_status <= OrderStatus::UNSHIPPED)

    {

        order\_address = address;

        return true;

    }

    return false;

}

bool Order::change\_phone(const string &phone)

{

    if (order\_status <= OrderStatus::UNSHIPPED)

    {

        order\_phone = phone;

        return true;

    }

    return false;

}

bool Order::add\_goods(uint32\_t goods\_id, uint32\_t amount)

{

    auto const goods = goods\_manager->search\_goods(goods\_id, true);

    if (goods.get\_id() == 0)

        return false;

    item\_list.push\_back(OrderItem(goods, amount));

    return true;

}

double Order::update\_total()

{

    order\_total = 0;

    for (auto const &item : item\_list)

    {

        auto price = item.price;

        if (item.tag.is\_discount && order\_time <= item.tag.discount\_end\_time)

            price \*= (1.0 - item.tag.discount / 100.0);

        auto item\_total = price \* item.amount;

        if (item.tag.is\_reduction && order\_time <= item.tag.reduction\_end\_time)

            item\_total -= (uint32\_t(item\_total / item.tag.spend) \* item.tag.reduction);

        order\_total += item\_total;

    }

    return order\_total;

}

bool Order::pay()

{

    if (order\_status == OrderStatus::UNPAID)

    {

        order\_status = OrderStatus::PAID;

        paid\_time = time(nullptr);

        for (auto const &item : item\_list) // 检测库存是否够用

        {

            auto goods = goods\_manager->search\_goods(item.goods\_id, true);

            if (goods.get\_stock() <= item.amount) // 至少剩一件

                return false;

        }

        for (auto const &item : item\_list) // 扣除库存

        {

            auto goods = goods\_manager->search\_goods(item.goods\_id, true);

            uint32\_t new\_stock = goods.get\_stock() - item.amount;

            goods\_manager->update\_goods(item.goods\_id,

                                        "", false,

                                        0, false,

                                        new\_stock, true,

                                        "", false,

                                        goods.get\_promotion(), false);

        }

        return sub\_stock = true;

    }

    return false;

}

bool Order::place\_order(Config &cfg)

{

    order\_id = generate\_id(cfg);

    if (order\_status != OrderStatus::UNPLACED)

        return false;

    order\_status = OrderStatus::UNPAID;

    order\_time = time(nullptr);

    update\_total();

    return true;

}

bool Order::cancel\_order()

{

    if (order\_status > OrderStatus::UNSHIPPED)

        return false;

    if (order\_status == OrderStatus::UNPAID)

    {

        paid\_time = time(nullptr); // 取消订单时，支付时间设置为取消时间

        order\_status = OrderStatus::CANCELLED;

        return true;

    }

    order\_status = OrderStatus::CANCELLED;

    if (sub\_stock)

        for (auto const &item : item\_list) // 增加库存

        {

            auto goods = goods\_manager->search\_goods(item.goods\_id, true);

            uint32\_t new\_stock = goods.get\_stock() + item.amount;

            goods\_manager->update\_goods(item.goods\_id,

                                        "", false,

                                        0, false,

                                        new\_stock, true,

                                        "", false,

                                        goods.get\_promotion(), false);

        }

    item\_list.clear();

    return true;

}

/\* 重载 ++ 运算符，方便更新状态 \*/

OrderStatus &operator++(OrderStatus &other)

{

    if (other != OrderStatus::DELIVERED)

        return other = OrderStatus(int(other) + 1);

    return other;

}

bool Order::update\_status()

{

    // 已送达或已取消订单状态不再更新

    if (order\_status == OrderStatus::DELIVERED || order\_status == OrderStatus::CANCELLED)

        return false;

    auto sub = time(nullptr) - paid\_time; // 计算时间差

    if (sub < 10)                         // 时间太短，不更新状态

        return false;

    else if (sub < 20) // 状态更新一次

        ++order\_status;

    else if (sub < 30) // 状态更新两次

        ++ ++order\_status;

    else if (sub < 40) // 状态更新三次

        ++ ++ ++order\_status;

    else // 状态更新三次以上，订单完成

        order\_status = OrderStatus::DELIVERED;

    return true;

}

OrderManager::OrderManager(const string &filename, GoodsManager &goods\_manager)

    : order\_list(), filename(filename), goods\_manager(goods\_manager), change(false)

{

    load();

}

OrderManager::~OrderManager()

{

    if (change)

        save();

}

void OrderManager::load()

{

    order\_list.clear();

    ifstream in(filename, ifstream::in);

    if (!in.is\_open())

    {

        ofstream out(filename, ofstream::out); // 不存在则新建文件

        out.close();

        return;

    }

    string line;

    while (getline(in, line))

        order\_list.push\_back(Order::load\_order(&goods\_manager, line));

    in.close();

}

void OrderManager::save() const

{

    ofstream out(filename, ofstream::out);

    if (!out.is\_open())

    {

        cout << "Failed to save order list to file " << filename << '.' << endl;

        return;

    }

    for (const auto &item : order\_list)

    {

        out << item.get\_id() << ','

            << item.get\_order\_time(true) << ','

            << item.get\_paid\_time(true) << ','

            << item.get\_receiver() << ','

            << item.get\_address() << ','

            << item.get\_phone() << ','

            << item.check\_status(true) << ',';

        for (auto const &order\_item : item.get\_item\_list())

            out << order\_item.goods\_id << ','

                << order\_item.name << ','

                << order\_item.price << ','

                << order\_item.amount << ',';

        out << item.get\_total() << endl;

    }

    out.close();

}

void OrderManager::set\_filename(const string &new\_filename)

{

    if (change)

        save();

    filename = new\_filename;

    change = false;

    load();

    cout << "Filename has been changed to " << filename << endl;

}

void OrderManager::remove\_order(uint32\_t order\_id)

{

    for (auto it = order\_list.begin(); it != order\_list.end(); ++it)

        if (it->get\_id() == order\_id)

        {

            order\_list.erase(it);

            change = true;

            break;

        }

}

void OrderManager::add\_order(const Order &order)

{

    order\_list.push\_back(order);

    change = true;

}

void OrderManager::view\_order(uint32\_t order\_id) const

{

    cout << "================================================================\n";

    if (order\_list.empty())

        return;

    if (order\_id == 0)

    {

        for (const auto &item : order\_list)

        {

            cout << "id:         " << item.get\_id() << endl

                 << "order\_time: " << item.get\_order\_time() << endl

                 << "paid\_time:  " << item.get\_paid\_time() << endl

                 << "receiver:   " << item.get\_receiver() << endl

                 << "address:    " << item.get\_address() << endl

                 << "phone:      " << item.get\_phone() << endl

                 << "status:     " << item.check\_status() << endl;

            for (auto const &order\_item : item.get\_item\_list())

                cout << "goods\_id:   " << order\_item.goods\_id << endl

                     << "name:       " << order\_item.name << endl

                     << "price:      " << order\_item.price << endl

                     << "amount:     " << order\_item.amount << endl;

            cout << "total:      " << item.get\_total() << endl

                 << "================================================================\n";

        }

    }

    else

        for (const auto &item : order\_list)

            if (item.get\_id() == order\_id)

            {

                cout << "id:         " << item.get\_id()

                     << "order\_time: " << item.get\_order\_time() << endl

                     << "paid\_time:  " << item.get\_paid\_time() << endl

                     << "receiver:   " << item.get\_receiver() << endl

                     << "address:    " << item.get\_address() << endl

                     << "phone:      " << item.get\_phone() << endl

                     << "status:     " << item.check\_status() << endl;

                for (auto const &order\_item : item.get\_item\_list())

                    cout << "goods\_id:   " << order\_item.goods\_id << endl

                         << "name:       " << order\_item.name << endl

                         << "price:      " << order\_item.price << endl

                         << "amount:     " << order\_item.amount << endl;

                cout << "total:      " << item.get\_total() << endl

                     << "================================================================\n";

            }

}

pair<vector<Order>::iterator, bool> OrderManager::find\_order(uint32\_t order\_id)

{

    for (auto it = order\_list.begin(); it != order\_list.end(); ++it)

        if (it->get\_id() == order\_id)

            return pair<vector<Order>::iterator, bool>(it, true);

    return pair<vector<Order>::iterator, bool>(order\_list.end(), false);

}

void OrderManager::update\_all\_order\_status()

{

    for (auto &item : order\_list)

        if (item.update\_status())

            change = true;

}

**promotion.h**

#ifndef DISCOUNT\_H

#define DISCOUNT\_H

#pragma once

#include <cstdint>

#include <iostream>

#include <ctime>

using std::ostream;

struct promotion\_tag

{

    bool is\_discount = false;      // 是否参与折扣

    bool is\_reduction = false;     // 是否参与满减

    bool both = false;             // 是否同时参与折扣和满减

    uint32\_t discount = 100;       // 折扣力度

    uint32\_t spend = 0;            // 满

    uint32\_t reduction = 0;        // 减

    time\_t discount\_end\_time = 0;  // 折扣结束时间

    time\_t reduction\_end\_time = 0; // 满减结束时间

    friend ostream &operator<<(ostream &os, const promotion\_tag &p)

    {

        os << p.is\_discount << ','

           << p.is\_reduction << ','

           << p.both << ','

           << p.discount << ','

           << p.spend << ','

           << p.reduction << ','

           << p.discount\_end\_time << ','

           << p.reduction\_end\_time;

        return os;

    }

};

#endif // DISCOUNT\_H

**system.h**

#ifndef SYSTEM\_H

#define SYSTEM\_H

#pragma once

#include "config.h"

#include "user.h"

#include "goods.h"

#include "shoppinglist.h"

#include "order.h"

#include <cstdint>

#include <string>

#include <unordered\_map>

using std::pair;

using std::string;

using std::unordered\_map;

struct UserInfo

{

    string password;

    string shopping\_list\_file;

    string order\_list\_file;

};

class System

{

private:

    Config &config;                              // 配置信息

    bool running;                                // 系统运行状态

    User admin;                                  // 管理员

    bool is\_admin\_active;                        // 管理员活跃状态

    User guest;                                  // 访客

    bool is\_guest\_active;                        // 访客活跃状态

    User current;                                // 当前用户

    bool is\_current\_active;                      // 当前用户活跃状态

    unordered\_map<string, UserInfo> account\_map; // 账号映射表

    GoodsManager goods\_manager;                  // 商品管理器

    ShoppingList shopping\_list;                  // 购物车

    OrderManager order\_manager;                  // 订单管理器

private:

    /\* 账号相关操作 \*/

    void

    save\_account\_map() const;                   // 保存账号映射表

    void load\_account\_map();                    // 加载账号映射表

    void help() const;                          // 帮助信息

    void exit();                                // 退出系统

    void login(bool is\_admin = false);          // 登录

    void logout();                              // 登出

    void sign\_up();                             // 注册

    void change\_password();                     // 修改密码

    void change\_account(bool is\_admin = false); // 更换账号

    /\* 商品相关操作 \*/

    void goods\_view() const; // 浏览商品

    void goods\_search();     // 搜索商品

    void goods\_add();        // 添加商品，需要管理员权限

    void goods\_remove();     // 移除商品，需要管理员权限

    void goods\_update();     // 修改商品，需要管理员权限

    /\* 购物车相关操作 \*/

    void shopping\_list\_add();        // 添加商品到购物车

    void shopping\_list\_sub();        // 从购物车减少商品数量

    void shopping\_list\_set();        // 设置购物车中某种商品的数量

    void shopping\_list\_remove();     // 从购物车移除商品

    void shopping\_list\_clear();      // 清空购物车

    void shopping\_list\_view() const; // 查看购物车

    /\* 订单相关操作 \*/

    void order\_place(bool flag = false, const string &ins = ""); // 下订单，默认从购物车

    void order\_cancel();                                         // 取消订单

    void order\_change();                                         // 修改订单

    void order\_check() const;                                    // 查看订单

    void order\_find();                                           // 查找订单

    void order\_remove();                                         // 删除订单

public:

    System(Config &cfg);

    ~System();

    void handle\_input(const string &input);           // 处理用户输入

    void run(const string &welcome\_msg = "Welcome!"); // 运行系统

    void stop() { running = false; }                  // 停止系统

};

#endif // SYSTEM\_H

**system.cpp**

#define elif else if

#include "system.h"

#include "config.h"

#include "user.h"

#include "functions.h"

#include "goods.h"

#include "shoppinglist.h"

#include "order.h"

#include <iomanip>

#include <iostream>

#include <sstream>

#include <fstream>

#include <string>

#include <vector>

#include <unordered\_map>

#include <stdexcept>

#include <cstdlib>

#include <ctime>

#include <cstdint>

using std::cerr;

using std::cin;

using std::cout;

using std::exception;

using std::getline;

using std::ifstream;

using std::istringstream;

using std::ofstream;

using std::pair;

using std::runtime\_error; // 抛出运行时异常

using std::stoul;

using std::string;

using std::unordered\_map; // 无序映射表

using std::vector;

const char endl = '\n';

System::System(Config &cfg)

    : config(cfg),                                                            // 引用传递设置类

      running(false),                                                         // 系统运行状态

      admin(true, false, cfg.get\_admin\_username(), cfg.get\_admin\_password()), // 创建管理员账户

      is\_admin\_active(false),                                                 // 管理员默认不活跃

      guest(false, true),                                                     // 创建访客账户

      is\_guest\_active(true),                                                  // 访客默认活跃

      current(),                                                              // 创建普通账号

      is\_current\_active(false),                                               // 普通账号默认不活跃

      account\_map(),                                                          // 创建账号映射表

      goods\_manager(cfg.get\_goods\_path()),                                    // 创建商品管理器

      shopping\_list(cfg.get\_shopping\_list\_path(), goods\_manager),             // 创建购物车

      order\_manager(cfg.get\_order\_list\_path(), goods\_manager)                 // 创建订单管理器

{

    /\* 解绑stdio \*/

    std::ios::sync\_with\_stdio(false);

    /\* 加载账号映射表 \*/

    load\_account\_map();

    /\* 初始化随机数种子 \*/

    srand(time(nullptr));

    cout << "System has been initialized." << endl;

}

void System::load\_account\_map()

{

    ifstream file(config.get\_accounts\_path(), ifstream::in);

    if (!file.is\_open())

        cerr << "Failed to open data file." << endl;

    else

    {

        string line;

        int start, end;

        vector<string> tmp;

        getline(file, line);        // 跳过第一行

        while (getline(file, line)) // 读取每一行

        {

            line = strip(line); // 去除换行符

            start = 0;

            end = line.find(',', start);

            for (int i = 0; i < 4; ++i) // 解析行

            {

                tmp.push\_back(line.substr(start, end - start));

                start = end + 1;

                end = line.find(',', start);

            }

            account\_map[tmp[0]] = UserInfo{tmp[1], tmp[2], tmp[3]}; // 存入映射表

            tmp.clear();

        }

        file.close();

    }

}

void System::save\_account\_map() const

{

    ofstream file(config.get\_accounts\_path(), ofstream::out);

    if (!file.is\_open())

        cerr << "Failed to open data file." << endl;

    else

    {

        file << "username,password,shoppinglist\_path" << endl;

        for (auto const &item : account\_map)

            file << item.first << ','

                 << item.second.password << ','

                 << item.second.shopping\_list\_file << ','

                 << item.second.order\_list\_file << endl;

        file.close();

    }

}

System::~System()

{

    cout << "System has exited." << endl;

}

void System::run(const string &welcome\_msg)

{

    cout << welcome\_msg << endl;

    running = true;

    string input;

    while (running)

    {

        order\_manager.update\_all\_order\_status(); // 更新订单状态

        cout << ">>> ";

        if (!get\_input(\*this, input))            // 读取用户输入

            stop();                              // 退出系统

        order\_manager.update\_all\_order\_status(); // 更新订单状态

        handle\_input(input);                     // 处理用户输入

    }

}

void System::handle\_input(const string &input)

{

    if (input.empty())

        return;

    // 账号相关指令

    elif (input == "help") help();

    elif (input == "exit") exit();

    elif (input == "li") login();

    elif (input == "li su") login(true);

    elif (input == "lo") logout();

    elif (input == "sp") sign\_up();

    elif (input == "cp") change\_password();

    elif (input == "ca") change\_account();

    elif (input == "ca su") change\_account(true);

    // 商品相关指令

    elif (input == "adg") goods\_add();

    elif (input == "rmg") goods\_remove();

    elif (input == "vig") goods\_view();

    elif (input == "upg") goods\_update();

    elif (input == "shg") goods\_search();

    // 购物车相关指令

    elif (input == "ads") shopping\_list\_add();

    elif (input == "sbs") shopping\_list\_sub();

    elif (input == "sts") shopping\_list\_set();

    elif (input == "rms") shopping\_list\_remove();

    elif (input == "cls") shopping\_list\_clear();

    elif (input == "vis") shopping\_list\_view();

    // 订单相关指令

    elif (input == "plo") order\_place();

    elif (start\_with(input, "buy")) order\_place(true, input);

    elif (input == "vio") order\_check();

    elif (input == "sho") order\_find();

    elif (input == "rmo") order\_remove();

    elif (input == "clo") order\_cancel();

    elif (input == "cho") order\_change();

    else cout

        << "Unknown command: " << input << endl

        << "Enter \"help\" to show available commands." << endl;

}

void System::exit()

{

    running = false;

}

void System::help() const

{

    cout << "Available commands:" << endl

         << "    help:  show help information" << endl

         << "    exit:  exit the system" << endl

         << "    li:    login" << endl

         << "    li su: login as admin" << endl

         << "    lo:    logout" << endl

         << "    sp:    sign up" << endl

         << "    cp:    change password" << endl

         << "    ca:    change account" << endl

         << "    ca su: change account as admin" << endl

         << "    vig:   view available goods" << endl

         << "    shg:   search for goods" << endl

         << "    ads:   add a kind of goods to shopping list" << endl

         << "    sbs:   subtract amount of a kind of goods in shopping list" << endl

         << "    sts:   set amount of a kind of goods in shopping list" << endl

         << "    rms:   remove a kind of goods from shopping list" << endl

         << "    cls:   clear shopping list" << endl

         << "    vis:   view shopping list" << endl

         << "    buy:   buy [id] [amount]: place an order for specific goods" << endl

         << "    plo:   place an order" << endl

         << "    vio:   check order status" << endl

         << "    sho:   search for orders" << endl

         << "    rmo:   remove an order(if possible)" << endl

         << "    clo:   cancel an order(if possible)" << endl

         << "    cho:   change receiver, address and phone of an order" << endl;

    if (is\_admin\_active) // 管理员特供指令

        cout << "    adg:   add a kind of goods" << endl

             << "    rmg:   remove a kind of goods" << endl

             << "    upg:   update goods information" << endl;

}

void System::login(bool is\_admin)

{

    if (is\_current\_active) // 普通用户已登录

    {

        cout << "You are already logged in as " << current.get\_username() << "." << endl;

        return;

    }

    if (is\_admin\_active) // 管理员登录

    {

        cout << "You are already logged in as admin." << endl;

        return;

    }

    string username, password;

    cout << "Please enter your username: ";

    get\_input(\*this, username);

    cout << "Please enter your password: ";

    get\_input(\*this, password);

    if (!is\_admin) // 普通用户登录

    {

        auto const it = account\_map.find(username);

        if (it == account\_map.cend()) // 用户不存在

        {

            cout << "User not registered. Please sign up." << endl;

            return;

        }

        if (it->second.password == password) // 密码正确

        {

            current.set\_username(username);

            current.set\_pwd(password);

            is\_current\_active = true;

            is\_admin\_active = false;

            is\_guest\_active = false;

            shopping\_list.set\_filename(it->second.shopping\_list\_file); // 设置购物清单文件名

            order\_manager.set\_filename(it->second.order\_list\_file);    // 设置订单清单文件名

            cout << "Login successful." << endl;

        }

        else // 密码错误

            cout << "Wrong password. Please try again." << endl;

    }

    else // 管理员登录

    {

        if (admin.get\_username() == username && admin.get\_pwd() == password) // 用户名密码正确

        {

            is\_current\_active = false;

            is\_admin\_active = true;

            is\_guest\_active = false;

            cout << "Login successful." << endl

                 << "Welcome " << admin.get\_username() << '!' << endl;

        }

        else // 用户名或密码错误

            cout << "Wrong username or password. Please try again." << endl;

    }

}

void System::logout()

{

    if (is\_guest\_active) // 当前用户未登录

    {

        cout << "You are not logged in." << endl;

        return;

    }

    current.set\_username("");

    current.set\_pwd("");

    is\_current\_active = false;

    is\_admin\_active = false;

    is\_guest\_active = true;

    cout << "Logout successful." << endl;

}

void System::sign\_up()

{

    if (is\_current\_active) // 当前用户已登录

    {

        cout << "You are already logged in as " << current.get\_username() << "." << endl;

        return;

    }

    if (is\_admin\_active) // 管理员登录

    {

        cout << "You are already logged in as admin." << endl;

        return;

    }

    string username, password, confirm\_password;

    cout << "Please enter your username: ";

    get\_input(\*this, username);

    auto const it = account\_map.find(username);

    if (it != account\_map.cend()) // 用户已存在

    {

        cout << "User already registered. Please try another username." << endl;

        return;

    }

    cout << "Please enter your password: ";

    get\_input(\*this, password);

    cout << "Please cofirm your password: ";

    get\_input(\*this, confirm\_password);

    if (password != confirm\_password) // 两次密码不一致

    {

        cout << "Passwords do not match. Please try again." << endl;

        return;

    }

    string shoppinglist\_path = "data/" + username + ".sl";

    ofstream file(shoppinglist\_path, ofstream::out); // 创建购物清单文件

    file.close();

    string orderlist\_path = "data/" + username + ".ol";

    file.open(orderlist\_path, ofstream::out); // 创建订单清单文件

    file.close();

    account\_map[username] = UserInfo{password, shoppinglist\_path, orderlist\_path}; // 存入账户信息映射表

    cout << "Sign up successful." << endl;

    // 保存数据

    save\_account\_map();

}

void System::change\_account(bool is\_admin)

{

    if (is\_guest\_active) // 当前用户未登录

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    logout();

    login(is\_admin);

}

void System::change\_password()

{

    if (is\_guest\_active) // 当前用户未登录

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    if (is\_admin\_active) // 管理员

    {

        string old\_password, new\_password, confirm\_password;

        cout << "Please enter your old password: ";

        get\_input(\*this, old\_password);

        if (old\_password != admin.get\_pwd()) // 旧密码错误

        {

            cout << "Wrong password. Please try again." << endl;

            return;

        }

        cout << "Please enter your new password: ";

        get\_input(\*this, new\_password);

        cout << "Please cofirm your new password: ";

        get\_input(\*this, confirm\_password);

        if (new\_password != confirm\_password) // 两次密码不一致

        {

            cout << "Passwords do not match. Please try again." << endl;

            return;

        }

        admin.set\_pwd(new\_password);

        config.set\_admin\_password(new\_password); // 更新配置文件

        cout << "Password changed successfully." << endl;

    }

    else

    {

        string old\_password, new\_password, confirm\_password;

        cout << "Please enter your old password: ";

        get\_input(\*this, old\_password);

        if (old\_password != current.get\_pwd()) // 旧密码错误

        {

            cout << "Wrong password. Please try again." << endl;

            return;

        }

        cout << "Please enter your new password: ";

        get\_input(\*this, new\_password);

        cout << "Please cofirm your new password: ";

        get\_input(\*this, confirm\_password);

        if (new\_password != confirm\_password) // 两次密码不一致

        {

            cout << "Passwords do not match. Please try again." << endl;

            return;

        }

        current.set\_pwd(new\_password);

        account\_map[current.get\_username()].password = new\_password; // 更新账户信息映射表

        save\_account\_map();                                          // 保存数据

        cout << "Password changed successfully." << endl;

    }

}

// ==============================

void System::goods\_view() const

{

    cout << "Available goods:" << endl;

    goods\_manager.list\_goods(); // 只向管理员输出商品id信息

}

void System::goods\_add()

{

    if (!is\_admin\_active)

    {

        cout << "You are not authorized to add goods." << endl;

        return;

    }

    cout << "Please enter the name of the goods: ";

    string name;

    get\_input(\*this, name);

    cout << "Please enter the price of the goods: ";

    float price;

    get\_input(\*this, price);

    cout << "Please enter the stock of the goods: ";

    uint32\_t stock;

    get\_input(\*this, stock);

    cout << "Please enter the description of the goods: ";

    string description;

    get\_input(\*this, description);

    // 促销信息

    cout << "Do you want to add a promotion tag? (y/n): ";

    string yes\_or\_no;

    bool r = get\_input(\*this, yes\_or\_no);

    bool update\_tag = r && (yes\_or\_no == "Y" || yes\_or\_no == "y");

    promotion\_tag tag;

    if (update\_tag)

    {

        cout << "Set discount? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        if (r && (yes\_or\_no == "Y" || yes\_or\_no == "y"))

        {

            cout << "Please enter discount(Press enter to cancel discount): ";

            tag.is\_discount = get\_input(\*this, tag.discount);

            if (tag.is\_discount)

            {

                cout << "Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): ";

                if (!get\_input(\*this, tag.discount\_end\_time))

                    tag.discount\_end\_time = time(nullptr) + 3600 \* 24; // 默认一天

            }

        }

        cout << "Set reduction? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        if (r && (yes\_or\_no == "Y" || yes\_or\_no == "y"))

        {

            cout << "Please enter new reduction(Press enter to cancel reduction): \n"

                 << "spend: ";

            tag.is\_reduction = get\_input(\*this, tag.spend);

            if (tag.is\_reduction)

            {

                cout << "reduction: ";

                get\_input(\*this, tag.reduction);

                cout << "Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): ";

                if (!get\_input(\*this, tag.reduction\_end\_time))

                    tag.reduction\_end\_time = time(nullptr) + 3600 \* 24; // 默认一天

            }

        }

        cout << "stackable? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        tag.both = r && (yes\_or\_no == "Y" || yes\_or\_no == "y");

    }

    goods\_manager.add\_goods(Goods(config, name, price, stock, description, tag));

    cout << "Goods added successfully." << endl;

}

void System::goods\_remove()

{

    if (!is\_admin\_active)

    {

        cout << "You are not authorized to remove goods." << endl;

        return;

    }

    cout << "Please enter the id of the goods to remove: ";

    uint32\_t id = 0u;

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    if (goods\_manager.remove\_goods(id))

        cout << "Goods removed successfully." << endl;

    else

        cout << "Failed to remove goods. No such kind of goods." << endl;

}

void System::goods\_update()

{

    if (!is\_admin\_active)

    {

        cout << "You are not authorized to update goods." << endl;

        return;

    }

    // 输入id

    cout << "Please enter the id of the goods to update: ";

    uint32\_t id = 0u;

    if (!get\_input(\*this, id) || id == 0) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    // 更新名称

    cout << "Please enter the new name of the goods(Press enter to skip): ";

    string name;

    get\_input(\*this, name);

    bool update\_name = !name.empty();

    // 更新价格

    cout << "Please enter the new price of the goods(Press enter to skip): ";

    float price = 0.0f;

    bool update\_price = get\_input(\*this, price);

    // 更新库存

    cout << "Please enter the new stock of the goods(Press enter to skip): ";

    uint32\_t stock = 0u;

    bool update\_stock = get\_input(\*this, stock);

    // 更新描述

    cout << "Please enter the new description of the goods(Press enter to skip): ";

    string description;

    get\_input(\*this, description);

    bool update\_description = !description.empty();

    // 更新促销信息

    cout << "Do you want to update the promotion tag? (y/n): ";

    string yes\_or\_no;

    bool r = get\_input(\*this, yes\_or\_no);

    bool update\_tag = r && (yes\_or\_no == "Y" || yes\_or\_no == "y");

    promotion\_tag tag;

    if (update\_tag)

    {

        cout << "Change discount? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        if (r && (yes\_or\_no == "Y" || yes\_or\_no == "y"))

        {

            cout << "Please enter new discount(Press enter to cancel discount): ";

            tag.is\_discount = get\_input(\*this, tag.discount);

            if (tag.is\_discount)

            {

                cout << "Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): ";

                if (!get\_input(\*this, tag.discount\_end\_time))

                    tag.discount\_end\_time = time(nullptr) + 3600 \* 24; // 默认一天

            }

        }

        cout << "Change reduction? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        if (r && (yes\_or\_no == "Y" || yes\_or\_no == "y"))

        {

            cout << "Please enter new reduction(Press enter to cancel reduction): \n"

                 << "spend: ";

            tag.is\_reduction = get\_input(\*this, tag.spend);

            if (tag.is\_reduction)

            {

                cout << "reduction: ";

                tag.is\_reduction = get\_input(\*this, tag.reduction);

                cout << "Please enter the end time(%Y-%m-%d %H:%M:%S, default one day): ";

                if (!get\_input(\*this, tag.reduction\_end\_time))

                    tag.reduction\_end\_time = time(nullptr) + 3600 \* 24; // 默认一天

            }

        }

        cout << "stackable? (y/n): ";

        r = get\_input(\*this, yes\_or\_no);

        tag.both = r && (yes\_or\_no == "Y" || yes\_or\_no == "y");

    }

    bool result = goods\_manager.update\_goods(id,

                                             name, update\_name,

                                             price, update\_price,

                                             stock, update\_stock,

                                             description, update\_description,

                                             tag, update\_tag);

    if (result)

        if (update\_name || update\_price || update\_stock || update\_description || update\_tag)

            cout << "Goods updated successfully." << endl;

        else

            cout << "Nothing to update." << endl;

    else

        cout << "Failed to update goods. No such kind of goods." << endl;

}

void System::goods\_search()

{

    cout << "Please enter the keyword to search for: ";

    string keyword;

    if (get\_input(\*this, keyword))

        goods\_manager.search\_goods(keyword);

    else

        cout << "Invalid input. Please try again." << endl;

}

// ==============================

void System::shopping\_list\_add()

{

    uint32\_t id, amount;

    cout << "Please enter the id of the goods to add: ";

    if (!get\_input(\*this, id))

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    cout << "Please enter the amount of the goods to add: ";

    if (!get\_input(\*this, amount))

        amount = 1u;

    shopping\_list.add(id, amount);

}

void System::shopping\_list\_sub()

{

    uint32\_t id, amount;

    cout << "Please enter the id of the goods to subtract: ";

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    cout << "Please enter the amount of the goods to subtract: ";

    if (!get\_input(\*this, amount))

        amount = 1u;

    shopping\_list.sub(id, amount);

}

void System::shopping\_list\_set()

{

    uint32\_t id, amount;

    cout << "Please enter the id of the goods to set: ";

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    cout << "Please enter the amount of the goods to set: ";

    if (!get\_input(\*this, amount))

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    if (amount <= 0)

    {

        cout << "Are you sure want to remove this item from shopping list? (y/n): " << endl;

        string tmp;

        get\_input(\*this, tmp);

        if (tmp == "y" || tmp == "Y")

            shopping\_list.remove(id);

        else

        {

            cout << "Operation cancelled." << endl;

            return;

        }

    }

    shopping\_list.set(id, amount);

}

void System::shopping\_list\_remove()

{

    cout << "Please enter the id of the goods to remove: ";

    uint32\_t id = 0u;

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    shopping\_list.remove(id);

}

void System::shopping\_list\_clear()

{

    shopping\_list.clear();

}

void System::shopping\_list\_view() const

{

    shopping\_list.list();

}

// ==============================

void System::order\_place(bool flag, const string &ins)

{

    if (is\_guest\_active) // 未登录拒绝

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    // 生成订单

    Order order(&goods\_manager);

    if (flag == false) // 从购物车下单

    {

        if (shopping\_list.empty()) // 空购物车拒绝

        {

            cout << "Your shopping list is empty. Please add some goods first." << endl;

            return;

        }

        // 填写信息

        string receiver, address, phone;

        cout << "Please enter the receiver's name: ";

        get\_input(\*this, receiver);

        order.change\_receiver(receiver);

        cout << "Please enter the receiver's address: ";

        get\_input(\*this, address);

        order.change\_address(address);

        cout << "Please enter the receiver's phone number: ";

        get\_input(\*this, phone);

        order.change\_phone(phone);

        // 遍历购物车

        for (auto const &item : shopping\_list.get\_shopping\_list())

            order.add\_goods(item.first, item.second); // 添加商品

        // 询问是否清空购物车

        cout << "Do you want to remove these items from shopping list? (y/n): " << endl;

        string tmp;

        get\_input(\*this, tmp);

        if (tmp == "y" || tmp == "Y")

            shopping\_list\_clear();

    }

    else // 直接下单

    {

        vector<string> args;

        istringstream ss(ins);

        string arg;

        ss >> arg; // 跳过buy字段

        while (ss >> arg)

            args.push\_back(arg);

        if (args.size() & 1 || args.size() < 2) // 奇数个参数或少于两个参数拒绝

        {

            cout << "Arguments mistake. Please try again." << endl;

            return;

        }

        // 填写信息

        string receiver, address, phone;

        cout << "Please enter the receiver's name: ";

        get\_input(\*this, receiver);

        order.change\_receiver(receiver);

        cout << "Please enter the receiver's address: ";

        get\_input(\*this, address);

        order.change\_address(address);

        cout << "Please enter the receiver's phone number: ";

        get\_input(\*this, phone);

        order.change\_phone(phone);

        // 遍历参数添加商品

        for (auto it = args.begin(); it != args.end(); it += 2)

            order.add\_goods(stoul(\*it), stoul(\*(it + 1)));

    }

    // 下单

    order.place\_order(config);

    // 输出总价

    cout << "Order summary:" << order.get\_total() << endl;

    // 支付

    cout << "Order created successfully. Please pay for it. Y/N: " << endl;

    string tmp;

    get\_input(\*this, tmp);

    if (tmp == "y" || tmp == "Y") // 支付

    {

        bool result = order.pay();

        if (result == false) // 支付失败

        {

            cout << "Stocks not enough. Order cancelled." << endl;

            order.cancel\_order();

        }

        else // 支付成功

            cout << "Payment successful." << endl;

    }

    else // 取消支付

    {

        cout << "Payment failed. Order cancelled." << endl;

        order.cancel\_order();

    }

    order\_manager.add\_order(order); // 保存订单

}

void System::order\_cancel()

{

    if (is\_guest\_active) // 未登录拒绝

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    cout << "Please enter the id of the order to cancel: ";

    uint32\_t id = 0u;

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    auto it = order\_manager.find\_order(id);

    if (it.second == false) // 订单不存在

    {

        cout << "No such order." << endl;

        return;

    }

    if (it.first->cancel\_order()) // 取消成功

        cout << "Order cancelled successfully." << endl;

    else // 取消失败

        cout << "Failed to cancel order. Your order has shipped." << endl;

}

void System::order\_change()

{

    cout << "Please enter the id of the order to change: ";

    uint32\_t id = 0u;

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    auto it = order\_manager.find\_order(id);

    if (it.second == false) // 订单不存在

    {

        cout << "No such order." << endl;

        return;

    }

    elif (it.first->check\_status(true) >= OrderStatus::SHIPPING) // 订单已发货

    {

        cout << "Order has shipped." << endl;

        return;

    }

    string receiver, address, phone;

    cout << "Please enter the new receiver's name(press enter to skip): ";

    get\_input(\*this, receiver);

    cout << "Please enter the new receiver's address(press enter to skip): ";

    get\_input(\*this, address);

    cout << "Please enter the new receiver's phone number(press enter to skip): ";

    get\_input(\*this, phone);

    if (!receiver.empty())

        it.first->change\_receiver(receiver);

    if (!address.empty())

        it.first->change\_address(address);

    if (!phone.empty())

        it.first->change\_phone(phone);

    cout << "Order changed successfully." << endl;

}

void System::order\_check() const

{

    if (is\_guest\_active) // 未登录拒绝

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    order\_manager.view\_order(0);

}

void System::order\_find()

{

    if (is\_guest\_active) // 未登录拒绝

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    uint32\_t id = 0u;

    cout << "Please enter the id of the order to check: ";

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    order\_manager.view\_order(id);

}

void System::order\_remove()

{

    if (is\_guest\_active) // 未登录拒绝

    {

        cout << "You are not logged in. Please login first." << endl;

        return;

    }

    uint32\_t id = 0u;

    cout << "Please enter the id of the order to remove: ";

    if (!get\_input(\*this, id)) // 保证id输入有效

    {

        cout << "Invalid input. Please try again." << endl;

        return;

    }

    auto it = order\_manager.find\_order(id);

    if (it.second == false) // 订单不存在

    {

        cout << "No such order." << endl;

        return;

    }

    if (it.first->check\_status(true) != OrderStatus::DELIVERED &&

        it.first->check\_status(true) != OrderStatus::CANCELLED) // 订单未完成

        cout << "Order has not finished. Cannot remove it." << endl;

    else

    {

        order\_manager.remove\_order(id);

        cout << "Order removed successfully." << endl;

    }

}

**functions.h**

#include "system.h"

#include <string>

#include <ctime>

#include <sstream>

using std::string;

using std::stringstream;

string strip(const string &str);

float jaro\_winkler\_similarity(const string &s1, const string &s2);

float jaro\_similarity(const string &s1, const string &s2);

bool start\_with(const string &str, const string &prefix);

const string &strtime(time\_t t);

template <typename T> // 读取用户输入模板

bool get\_input(System &sys, T &input)

{

    string line;

    if (!get\_input(sys, line))

        return false;

    if (line.empty())

        return false;

    stringstream ss(line);

    T tmp;

    if (ss >> tmp)

        return true;

    return false;

}

template <> // 字符串特化

bool get\_input<string>(System &sys, string &input);

template <> // uint32\_t特化

bool get\_input<uint32\_t>(System &sys, uint32\_t &input);

template <> // time\_t特化

bool get\_input<time\_t>(System &sys, time\_t &input);

**functions.cpp**

#include "functions.h"

#include "system.h"

#include <string>

#include <vector>

#include <algorithm>

#include <ctime>

#include <stdexcept>

#include <iostream>

#include <sstream>

#include <iomanip>

using std::cerr;

using std::cin;

using std::equal;

using std::exception;

using std::getline;

using std::max;

using std::min;

using std::mktime;

using std::runtime\_error;

using std::string;

using std::stringstream;

using std::vector;

const char endl = '\n';

string strip(const string &str)

{

    auto start = str.find\_first\_not\_of(" \t\n\r\f\v"); // 找到第一个非空白位置

    if (start == string::npos)

        return "";                                  // 全空白字符串返回空串

    auto end = str.find\_last\_not\_of(" \t\n\r\f\v"); // 找到最后一个非空白位置

    return str.substr(start, end - start + 1);      // 截取有效部分

}

float jaro\_winkler\_similarity(const string &s1, const string &s2)

{

    int len1 = s1.length();

    int len2 = s2.length();

    // 处理空字符串情况

    if (len1 == 0 || len2 == 0)

        return 0.0f;

    // 完全相同的情况

    if (s1 == s2)

        return 1.0f;

    // 计算匹配窗口距离

    int match\_distance = (max(len1, len2) / 2) - 1;

    // 初始化匹配标记数组

    vector<bool> s1\_matches(len1, false);

    vector<bool> s2\_matches(len2, false);

    int matches = 0;

    // 遍历s1的每个字符寻找匹配

    for (int i = 0; i < len1; ++i)

    {

        int start = max(0, i - match\_distance);

        int end = min(len2 - 1, i + match\_distance);

        for (int j = start; j <= end; ++j)

            if (!s2\_matches[j] && s1[i] == s2[j])

            {

                s1\_matches[i] = true;

                s2\_matches[j] = true;

                matches++;

                break;

            }

    }

    // 无匹配则返回0

    if (matches == 0)

        return 0.0f;

    // 收集匹配的字符

    vector<char> s1\_chars, s2\_chars;

    for (int i = 0; i < len1; ++i)

        if (s1\_matches[i])

            s1\_chars.push\_back(s1[i]);

    for (int j = 0; j < len2; ++j)

        if (s2\_matches[j])

            s2\_chars.push\_back(s2[j]);

    // 计算换位数目

    float transpositions = 0.0f;

    for (int i = 0; i < s1\_chars.size(); ++i)

        if (s1\_chars[i] != s2\_chars[i])

            transpositions += 1.0f;

    transpositions /= 2.0f;

    // 计算Jaro相似度

    float m = static\_cast<float>(matches);

    float jaro = (m / len1 + m / len2 + (m - transpositions) / m) / 3.0f;

    // 计算前缀匹配长度（最多4个字符）

    int prefix\_len = 0;

    int max\_prefix = min(4, min(len1, len2));

    for (; prefix\_len < max\_prefix; ++prefix\_len)

        if (s1[prefix\_len] != s2[prefix\_len])

            break;

    // 应用Winkler调整

    float jaro\_winkler = jaro + prefix\_len \* 0.1 \* (1.0 - jaro);

    // 确保结果不超过1

    return min(jaro\_winkler, 1.0f);

}

float jaro\_similarity(const string &s1, const string &s2)

{

    int len1 = s1.length();

    int len2 = s2.length();

    // 处理空字符串情况

    if (len1 == 0 || len2 == 0)

        return 0.0f;

    // 完全相同的情况

    if (s1 == s2)

        return 1.0f;

    // 计算匹配窗口距离

    int match\_distance = (max(len1, len2) / 2) - 1;

    // 初始化匹配标记数组

    vector<bool> s1\_matches(len1, false);

    vector<bool> s2\_matches(len2, false);

    int matches = 0;

    // 遍历s1的每个字符寻找匹配

    for (int i = 0; i < len1; ++i)

    {

        int start = max(0, i - match\_distance);

        int end = min(len2 - 1, i + match\_distance);

        for (int j = start; j <= end; ++j)

            if (!s2\_matches[j] && s1[i] == s2[j])

            {

                s1\_matches[i] = true;

                s2\_matches[j] = true;

                matches++;

                break;

            }

    }

    // 无匹配则返回0

    if (matches == 0)

        return 0.0f;

    // 收集匹配的字符

    vector<char> s1\_chars, s2\_chars;

    for (int i = 0; i < len1; ++i)

        if (s1\_matches[i])

            s1\_chars.push\_back(s1[i]);

    for (int j = 0; j < len2; ++j)

        if (s2\_matches[j])

            s2\_chars.push\_back(s2[j]);

    // 计算换位数目

    float transpositions = 0.0f;

    for (int i = 0; i < s1\_chars.size(); ++i)

        if (s1\_chars[i] != s2\_chars[i])

            transpositions += 1.0f;

    transpositions /= 2.0f;

    // 计算Jaro相似度

    float m = static\_cast<float>(matches);

    float jaro = (m / len1 + m / len2 + (m - transpositions) / m) / 3.0f;

    return jaro;

}

bool start\_with(const string &str, const string &prefix)

{

    // 先检查长度防止越界访问

    if (prefix.size() > str.size())

        return false;

    // 使用标准算法比较字符序列

    return equal(prefix.begin(), prefix.end(), str.begin());

}

const string &strtime(time\_t t)

{

    static string format;

    format.clear();

    std::tm \*temp = std::localtime(&t);

    char buffer[32];

    std::strftime(buffer, 32, "%Y-%m-%d %H:%M:%S", temp);

    format = buffer;

    return format;

}

template <> // 字符串特化

bool get\_input<string>(System &sys, string &input)

{

    try // 尝试读取整行输入

    {

        if (!getline(cin, input))

        {

            input.clear();

            if (cin.eof())

                throw runtime\_error("EOF detected. Exiting...");

            else

                throw runtime\_error("Failed to read input.");

        }

        else

        {

            input = strip(input); // 去除前后空白

            return true;

        }

    }

    catch (const runtime\_error &err)

    {

        cerr << err.what() << endl

             << "The system has stopped unexpectedly." << endl;

        sys.stop();

        return false;

    }

    catch (const exception &err)

    {

        cerr << "Unknown error: " << err.what() << endl

             << "The system has stopped unexpectedly." << endl;

        sys.stop();

        return false;

    }

}

template <> // uint32\_t特化

bool get\_input<uint32\_t>(System &sys, uint32\_t &input)

{

    string line;

    if (!get\_input(sys, line))

        return false;

    if (line.empty())

        return false;

    long long tmp;

    if (sscanf(line.c\_str(), "%lld", &tmp) == 1 && tmp >= 0)

    {

        input = tmp;

        return true;

    }

    return false;

}

template <>

bool get\_input<time\_t>(System &sys, time\_t &input)

{

    string line;

    if (!get\_input(sys, line))

        return false;

    if (line.empty())

        return false;

    tm tmp = {};

    if (sscanf(line.c\_str(), "%d-%d-%d %d:%d:%d",

               &tmp.tm\_year, &tmp.tm\_mon, &tmp.tm\_mday,

               &tmp.tm\_hour, &tmp.tm\_min, &tmp.tm\_sec) != 6)

        return false;

    else

    {

        tmp.tm\_year -= 1900;

        tmp.tm\_mon--;

        input = mktime(&tmp);

        return true;

    }

}