网上超市管理系统——设计报告

# **系统总体设计**

我组在完美的完成小学期任务要求的基础上，为满足用户需要与界面友好，新增了许多功能，并在数据库上做出了重要的调整。

项目进度详情上载于Github.([https:/github.com/zolars/Supermarket-Management-System](https://github.com/zolars/Supermarket-Management-System))

## **系统总体功能描述**

## 

## 包含绝大部分顾客在使用网上超市时需要的功能。

## 购买时会判断存货以及余额判断是否下订单，并反馈给用户。

## 模拟购买支付功能。

## 在顾客购买后，可显示该超市的推荐畅销商品。

## **接口定义规范**

### 严格遵循Unix命名规范。

### 每个模块的头文件以及源文件顶端都含有调用该接口所需要的注释。

### 注释内容包含:

### 文件名

### 作者

### 完成日期

### 使用的编程语言

### 模块完成的功能

### 模块调用的头文件

### 自定义头文件

### 环境自带头文件

### 所需输入值（name, type）

### 返回值（name, type）

## **数据结构描述**

### 根目录/

### Admin\_information/

### {admin\_id}.txt

### Consumer\_information/

### {consumer}.txt

### Goods\_index/

### {goods\_id}.txt

### Name\_to\_id/

### {goods\_name.txt}

### Order\_admin/

### {admin\_id}\_all.txt

### {admin\_id}\_goods.txt

### {admin\_id}\_consumer.txt

### Order\_consumer/

### {consumer\_id}.txt

### Shop\_index/

### {shop\_name}.txt

### Shopping\_cart/

### {consumer}.txt

### All\_index.txt

### Login\_result.txt

## **系统开发环境**

### Mac

### 编译环境

### Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-gxx-include-dir=/usr/include/c++/4.2.1

### Apple LLVM version 8.1.0 (clang-802.0.42)

### Target: x86\_64-apple-darwin16.7.0

### Thread model: posix

### InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin

### 运行环境

### 系统：macOS 10.12.6 (16G29)

### 硬件：MacBook Pro (13-inch, Late 2016)

### Windows

### 编译环境

### Dev C++ 5.10

### Mingw 编译器

### 运行环境

### 系统：Windows10 ver1703

### 硬件：XPS 15 9550 i7 6700 HQ

# **database模块详细设计**

## **设计思路**

### 在需要调用database文件夹中的文件时，如果在每个源文件中都要编写读取写入的代码，不仅是无用功，很大程度上还会影响程序的完整性及可读性。

### 为了解决这个问题，特意编写了数据库接口代码database.cpp，这是我组项目的核心之一。其中包含结构体的定义以及调用文件的函数。

### 结构体为我们存储数据的方式：一个商品A有可能有许多条信息，例如售价、库存、折扣时间等，这时利用结构体数组来存储就会非常便利。

### 除此之外，由于数据库接口的作用，我组的数据库不需要在程序开始全部录入，而是按需录入，从而增大空间利用效率。

### 这个想法在项目开始时就成型，并已经着手搭建了。在之后的编写中，小组成员并不需要了解任何数据库具体结构与存储类型，只需要在自己的代码中添加#include “database.h”与包含函数名的短短几行代码即可。

### 在后期调试与维护时，数据库接口的作用更加明显。当我发现某些数据库并未按照预想工作时，只需要在数据库接口的返回值上做更改即可查出bug所在位置并改正。

### Notes:

### *使用头文件声明函数与全局变量，以完成在编译运行时只调用需要的函数，可以占用最小的内存。*

### *例如: database.h与database.cpp的配合运行*

### *在源文件中：*

### *STU\_consumer\_information consumer\_information;*

### *int database\_consumer\_information(char user\_id[30], int read\_type) { …*

### *};*

### *在头文件中：*

### *typedef struct {*

### *…*

### *} STU\_consumer\_information;*

### *extern STU\_consumer\_information consumer\_information;*

### *extern int database\_consumer\_information(char user\_id[30], int read\_type);*

### **注释**

#### Name:

#### database.cpp

#### Author:

#### 辛逸飞

#### Date:

#### 2017-09-13

#### Language:

#### C

#### Features:

#### 数据库函数: 文件读取, 转存为结构体或写入文件.

#### Modules:

##### stdio.h

##### stdlib.h

##### string.h

#### Notes:

##### 所有注释均写在 "database.h" 头文件中, 目的是便于隐藏具体数据布局, 仅仅暴露使用者需要调用的数据接口以及全局变量名.

##### 当我们需要遍历全部的结构体时, 找到结构体中的一个非零变量, 使用 while 语句检测其是否为零, 即可判断尽头.

##### 在头文件中定义结构体, 会使结构体中的具体数据布局暴露,但我暂时不知道有什么更好的解决方案.

# **shopping模块详细设计**

## **设计思路**

## 目的是完成购买时同步登入数据库。

## 在完成购买时，要先根据拥有的数据（超市ID，商品名，购买数量，用户ID）来推断是够可以购买。

### 减掉顾客余额

## 数据库: consumer\_information

#### 减掉商品清单中的库存, 销量加1

#### 数据库: goods\_index

#### 减掉商店清单中的库存, 销量加1

#### 数据库: shop\_index

#### 增加管理员订单\_all

#### 数据库: order\_admin\_all

#### 增加管理员订单\_goods

#### 数据库: order\_admin\_goods

#### 增加管理员订单\_consumer

#### 数据库: order\_admin\_consumer

#### 增加顾客订单

#### 数据库: order\_consumer

## **注释**

### Name:

### shopping.cpp

### Author:

### 辛逸飞

### Date:

### 2017-09-16

### Language:

### C

### Features:

### 完成购买入库的接口

### Modules:

### Customize:

### database.h

### shopping.h

### check\_time.h

### Native:

### stdio.h

### string.h

### stdlib.h

### Input:

### user\_id

### temp\_goods\_name

### temp\_shop\_id

### temp\_purchase\_num

### Output:

### 1: 购买成功

### 0: 存货不足

### (-1): 余额不足

# **add\_goods模块详细设计**

## **设计思路**

### 该模块完成的工作为：明确超市管理员身份并添加货物。再设计该模块时，我们需要建立商品名与商品ID一一对应的关系，以模拟真实的超市中存在的扫条形码上货的场景。 该模块为其余模块的基础，所以一定要严格按照格式来输入。在输入纠错方面我们下了很大的功夫：首先保证每一个输入都不会报错，其次在用户错误输入格式时，提醒其改正。

## **注释**

### Name:

### add\_goods.cpp

### Author:

### 王克淼

### 辛逸飞

### Date:

### 2017-09-15

### Language:

### C

### Features:

### 完成"商品添加"功能的模块.

### 管理员添加商品信息

### 添加信息时对每一个信息进行检测，判断是否不合法

### Modules

### add\_goods.h

### database.h

### ctype.h

### stdio.h

### stdlib.h

### string.h

# **buy\_search\_all模块详细设计**

## **设计思路**

### 该模块具体完成了购买时的搜索和针对搜索结果的筛选，重要性和整体性非常高。 我在实现搜索时曾经考虑过使用字符串匹配外加正则表达式的方案，但是由于这样的话，在调用“管理员管理超市信息”、“添加商品”、“订单录入”三个功能时，都要遍历并筛选，时间复杂度会很高。所以我们采取了两条索引标准的方案，使用“打开特定文件名”的方法实现搜索。虽然舍弃了部分搜索的便利性，但是在数据条数积累到很高的水平之后，这种方案的高效性就会被凸显出来，而快速和高效是超市管理中不容忽视的重要部分。

## 设计思路的延展：在项目即将结束时，我发现C语言是支持遍历文件夹提取文件名的，如果是这样，就可以借助正则表达式来实现模糊搜索。但由于时间有限，我没有把精力放在这方面。如果有可能的话，其实应当完善这一部分。这样的话，在系统需要精确的索引时，可以迅速找到文件并打开，节省时间；当系统需要模糊搜索时，也可以胜任。

## 注释

### Name:

### buy\_search\_all.cpp

### Author:

### 董泽元

### Date:

### 2017-09-15

### Language:

### C

### Features:

### 完成"商品查询"功能的模块.

### 商品查询分为按商品名查询及超市名查询

### 用户在搜索时可进行相应的筛选

### 该模块可直接将物品放入购物车或直接购买

### Modules:

### database.h

### stdio.h

### stdlib.h

### string.h

# **Search\_Order模块详细设计**

## **设计思路** 如何实现对结构体的排序曾经困扰了我很久，在利用搜索引擎进行查询后，我发现C++的sort函数可以按结构体的单个元素排序。这里主要使用了结构体的重构来实现，而且有一些细节一定要注意。

## **注释**

### Name:

### search\_for\_order.cpp

### Author:

### 辛逸飞

### Date:

### 2017-09-12

### Language:

### C

### Features:

### 完成"查询订单"功能的模块.

### Modules:

### Customize:

### database.h

### search\_order.h

### Native:

### stdio.h

### string.h

### stdlib.h

### Input:

### user\_id

### user\_type

### Output:

### 成功运行返回”1”

# **Cart模块详细设计**

## **设计思路** 该模块并不在老师的要求范围内，但我们考虑到用户的确需要购物车来记录欲购物品，便添加了这个模块。这也展示了模块化的重要优点：拓展性。 购物车功能包含：批量购买，单个购买，移除。

## **注释**

### Name:

### cart.cpp

### Author:

### 刘育成

### 王楠鑫

### Date:

### 2017-09-15

### Language:

### C

### Features:

### 购物车模块，存储顾客选择的商品的信息，验证库存余额能否购买

### Modules:

### cart.h

### check\_time.h

### database.h

### shopping.h

### stdio.h

### stdlib.h

### string.h

# **change\_information模块详细设计**

## **设计思路**

### 调取数据库，更改后储存。

### 

## **注释**

# **register & login模块详细设计**

## **设计思路** 存储信息入库。

## **login注释**

### Name:

### login.cpp

### Author:

### 刘育成

### Date:

### 2017-09-13

### Language:

### C

### Features:

### 登录模块，选择用户类型，用户名密码验证

### Modules:

### database.h

### login.h

### stdio.h

### stdlib.h

### string.h

## **register注释**

### Name:

### register.h

### Author:

### 王楠鑫

### Date:

### 2017-09-14

### Language:

### C

### Features:

### 用户输入新的用户名以后会检测该用户名是否已经被注册

### 自带电话号码及电子邮箱输入检测

### Modules:

### database.h

### register.h

### math.h

### stdio.h

### stdlib.h

### string.h

# **小组成员**

### 组长：

### 辛逸飞：数据库，购买入库，查询订单，购物车，系统整合，debug

### 组员：

### 王楠鑫：个人信息管理，注册

董泽元：商品查询

### 刘育成：登录

### 王克淼：商品添加

辛逸飞

2017.09.20