**《高级语言程序设计》实验大作业反思报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **实验大作业题目** | **简易交易管理系统** | | | **类 型** | | **信息管理系统** | |
| **学生姓名** | **王翰坤** | **班 号** | **11837101** | **学 号** | | **1183710106** | |
| **所在院系** | **计算机学院**  **软件工程专业** | **学 期** | **2018年秋季学期** | **任课教师** | | **苏小红** | |
| **实验类型** | **综合设计型** | | | | | | |
| **实验目的：** | | | | | | | |
| * 掌握程序设计的基本算法和简单数据结构基础，能够综合运用基本控制语句、算法和数据结构，以及自顶向下、逐步求精的模块化设计方法，能够设计具有一定规模的系统级C语言程序，提高系统编程能力； * 针对计算相关的复杂工程问题，能够使用恰当的算法和数据结构，完成计算、统计、排序、检索、匹配等相关的软件系统的构造、测试与实现； * 掌握常用的程序调试和测试方法。 | | | | | | | |
| **实验要求：** | | | | | | | |
| * 采用自顶向下、逐步求精的模块化设计思想设计一个小型信息库管理系统，或者闯关式游戏程序。 * 要求解释说明采用了什么数据结构和算法，为什么选择这种数据结构或算法，系统实现过程中遇到了哪些问题，这些问题是如何解决的，还有什么问题尚未解决，今后打算从哪几个方面进行改进，本设计的亮点和难点在哪里，实验结果如何，有哪些收获和学习体会； * 编写程序完成以下实验大作业内容并完成实验大作业反思报告。 | | | | | | | |
| **实验内容：** | | | | | | | |
| 设计一个简易交易管理系统，包括用户信息、商品信息、记录信息的增删改查等基本操作，编程实现如下将结构的简易交易管理系统：  **1 欢迎与注册/登录界面；**  **1.1 用户注册界面**  输入用户名及密码，程序分配用户编号，完成用户注册  **1.2 用户登录界面**  输入用户编号及密码以登录系统  **2 主菜单**  (1) 个人菜单  (2) 购物菜单  (3) 保存信息  **2.1 个人菜单**  (1) 查看个人购买记录、购买总数量与总金额  (2a) 申请商家用户资格（普通用户）  (2b) 进入商家菜单  **2.1.2 商家菜单**  (1) 查看个人商品列表  (2) 查看个人售卖记录、售卖总数量与总金额  (3) 增加商品  (4) 删除商品  (5) 修改指定商品库存量  **2.2 购物菜单**  (1)搜索（支持模糊搜索、带\*和?的通配符搜索）与购买  以“**欢迎与注册/登录界面”为例，**要求程序显示如下菜单，并提示用户输入选项：  =-- Welcome Menu --  0 Save and quit  1 Register  2 Login  >>  然后，根据用户输入的选项执行相应的操作。  用复杂的静态数据结构（结构体数组）保存用户、商品、记录等信息。以商品信息为例，定义如下的结构体：  typedef struct goods  {  int number;  int seller\_num;  int deleted;  int stock\_cnt;  int price;  char name[name\_max\_length];  } Goods;  使用模块化程序设计方法设计各个子模块，例如：注册模块、登录模块、各菜单的模块，添加商品模块，删除商品模块，搜索模块，链表操作模块、文件读写模块、MD5加密模块、自定义IO模块、用户输入检查模块等。能够对多种类型信息进行统计、分类和查找。例如，按所属用户查找商品信息、记录信息；按商品编号查找商品；按相似度对搜索结果进行排序等。 | | | | | | | |
| **实验环境：** | | | | | | | |
| 操作系统：macOS Mojave 10.14.2  集成开发环境：CLion 2018.3.2  编译器：Clang, Clang 1000.10.44.4 (Apple LLVM version 10.0.0)  CMake, GNU CMake 3.81  外部库：无 | | | | | | | |
| **输入输出设计：** | | | | | | | |
| **1 输入**：程序的输入分为两部分：文件输入与键盘输入  **1.1 文件输入**：共有5个.info文件。  - basic.info，包含三个整数：总用户数、总商品数、总记录数，数据类型为int  - links.info，保存了所有链表节点值，数据类型为int  这两个文件的保存，方便读取相对独立的整型数据。  - users.info，以数据块形式保存了所有用户信息，数据类型为struct user  - goods.info，以数据块形式保存了所有商品信息，数据类型为struct goods  - records.info，以数据块形式保存了所有记录信息，数据类型为struct record  这三个文件的保存，方便整体读写结构体类型的数据。  **1.2键盘输入**：均以字符（串）形式读入。  以字符（串）形式读入，提高了程序健壮性，方便后续处理  **2 输出**：程序的输出分为两部分：文件输出与屏幕输出  **2.1 文件输出**  与文件输入的方法与格式一致。  **2.2屏幕输出**  每次用户输入前，屏幕上都会显示相关提示或要求。  用户输入后，屏幕上会显示相应的反馈（列表信息、搜索结果、停顿确认、操作成功或失败等）  **3 异常处理**  - 初始化时，若有文件打开失败或损坏，系统会报错并自动重建所有文件。  - 所有的用户输入均会在函数入口处进行合法性检查。若不合格式，程序就会报错并要求用户重新输入。  - 若动态内存分配失败，系统会报错并要求用户尝试其他操作 | | | | | | | |
| **系统设计与实现：** | | | | | | | |
| 1. **系统功能模块划分**   对系统进行自顶向下的模块分解，画出系统各个功能模块之间的结构图如下（见下页）：  **/Users/yeawhk/Documents/Untitled.pngUntitled**   1. **函数功能和外部接口设计**   本系统总计设计了70个函数，每个函数的功能和接口设计如下表所示：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **序号** | **函数名** | **函数功能** | **参数** | **返回值** | | 1 | main | 主函数 | / | 0 | | 2 | run\_manager | 运行管理系统 | / | / | | 3 | initialize | 初始化 | / | / | | 4 | initialize\_normal | 读取存档 | / | / | | 5 | initialize\_file | 重建存档 | / | / | | 6 | initialize\_version | 初始化版本信息 | / | / | | 7 | recover\_all | 重置信息 | / | / | | 8 | check\_file\_write | 检查文件是否可写 | char\* file\_name | 0/1 | | 9 | check\_file\_read | 检查文件是否可读 | char\* file\_name | 0/1 | | 10 | write\_new\_user | 写入新用户信息 | User\* cur | / | | 11 | write\_new\_goods | 写入新商品信息 | Goods \*cur | / | | 12 | reset\_modify\_signs | 重置修改标记 | / | / | | 13 | save\_all | 文件存档 | / | / | | 14 | append\_linkNode | 非空链表尾添加节点 返回前驱节点地址 | lnk\* prev, int id | lnk\* | | 15 | display\_linkNode | 显示链表信息 返回链表信息汇总值 | lnk\* frst print\_info printer | int | | 16 | delete\_linkNode | 删除链表节点 返回头节点地址 | lnk\* head int node\_val | lnk\* | | 17 | get\_tail\_linkNode | 获取链表尾节点 | lnk\* head int node\_val | lnk\* | | 18 | table\_reconstruct | 从文件重建链表 | FILE\* fp lnk\*\* pHead lnk\*\* pHail int len | 0/1 | | 19 | table\_append | 链表尾添加节点 | int\* len lnk\*\* pHead lnk\*\* pTail int val | 0/1 | | 20 | table\_write | 向文件写入链表 | FILE\* fp lnk\* pHead | / | | 21 | add\_user | 添加用户 | const char \*user\_name const unsigned \*reg\_pw | / | | 22 | register\_seller | 将用户标记为商家 | / | / | | 23 | add\_goods | 添加商品 | const char \*goods\_name int storage int price | 0/1 | | 24 | del\_goods | 删除商品 | int goods\_id | / | | 25 | mod\_goods\_stock | 修改商品库存量 | int goods\_id int qnt | / | | 26 | add\_record | 添加记录 | int goods\_id int seller\_id int qnt | 0/1 | | 27 | buy\_goods | 购买商品 更新信息 | int goods\_id, int qnt | 0/1 | | 28 | get\_the\_line | 获取整行输入 | char \*str int need\_all | / | | 29 | print\_remind\_message | 输出提示信息 | int need | / | | 30 | wait\_continue | 等待用户回车确认 | / | / | | 31 | print\_waiting\_chars | 输出提示字符 | / | / | | 32 | print\_error | 输出错误信息 | int error\_id | / | | 33 | is\_number | 判断字符串是否为数 | char\* str | 0/1 | | 34 | get\_max | 返回较大值 | int x int y | int | | 35 | print\_buy\_records\_list | 输出购买记录列表 | / | / | | 36 | print\_my\_goods\_list | 输出商品列表 | / | / | | 37 | print\_sell\_records\_list | 输出售卖记录列表 | / | / | | 38 | print\_record | 输出记录 返回当前记录信息 | int order int record\_id | int | | 39 | print\_goods | 输出商品 返回当前商品信息 | int order int goods\_id | int | | 40 | print\_search\_item | 输出搜索条目 | int order int goods\_id | / | | 41 | input\_and\_check\_single | 单一参数输入与检查 | checker\_sgl legal\_check char \*str | / | | 42 | input\_and\_check\_num | 单一整数输入与检查 返回输入的整数值 | checker\_num legal\_check char\* str int lower int upper | int | | 43 | circulate\_confirm\_name | 确认用户名 | char \*str | / | | 44 | circulate\_confirm\_password | 确认密码 | unsigned \*dest | / | | 45 | legal\_check\_return\_num | 合法性检查（数值） 返回数值 | char \*str int lower int upper | int | | 46 | legal\_check\_name | 合法性检查（名） | char \*str | 0/1 | | 47 | legal\_check\_pw | 合法性检查（密码） | char\* str | 0/1 | | 48 | legal\_check\_keyword | 合法性检查（关键字） | char\* str | 0/1 | | 49 | legal\_check\_yon | 合法性检查（y/n） | char \*str | 0/1 | | 50 | check\_MD5 | 检查MD5 | const unsigned\* a const unsigned\* b | 0/1 | | 51 | left\_rotate | 循环左移n位 | unsigned val unsigned n | unsigned | | 52 | init\_value | 初始化MD5常量 | / | / | | 53 | convert\_to\_sexa\_str | 将数转为16进制字符串 | const char\* src char\* dest | / | | 54 | convert\_to\_sexa\_num | 将字符串转为16进制数 | const char\* src unsigned \*dest | / | | 55 | reverse\_sexa | 将16进制数逆序 | unsigned x | unsigned | | 56 | get\_MD5 | 获取字符串MD5 | const char\* inp unsigned\* dest | / | | 57 | cmp\_by\_sim | qsort比较函数  关键字为对应字符串相似度值 | const void\* a const void\* b | int | | 58 | convert\_to\_lower | 将字符串中大写字母改为小写 | char \*dest  const char \*src | / | | 59 | get\_fail | 计算KMP失配函数 | char \*P int \*f | / | | 60 | KMP | 单模式串匹配KMP 返回匹配点数量 | char \*T char \*P int \*f | int | | 61 | LCS | 最长公共子序列 | char\* a char\* b | int | | 62 | wild\_match | 通配符匹配 | char\* a char\* b | 0/1 | | 63 | main\_search | 关键字搜索 返回结果条数 | char\* inp int start\_ord int kw\_num int\* result | int | | 64 | show\_welcome | 显示欢迎界面 | / | / | | 65 | show\_register | 显示注册界面 | / | / | | 66 | show\_login | 显示登录界面 | / | / | | 67 | show\_main\_menu | 显示主菜单 | / | / | | 68 | show\_personal\_menu | 显示个人菜单 | / | / | | 69 | show\_seller\_menu | 显示商家菜单 | / | / | | 70 | show\_shopping\_menu | 显示购物菜单 | / | / |   各个函数之间的调用关系如下所示（见下页）：  /Users/yeawhk/Documents/Untitled0.pngUntitled0注：黄色箭头表示调用关系，它从被调用者出发，指向调用者。   1. **算法**   使用了**枚举**、**迭代**的算法  使用了**分类统计**和**递归**的方法。  使用了**快速排序**算法。  使用了**MD5算法**，加密用户信息。  使用了**KMP算法**，实现单模式串匹配。  使用了动态规划的方法，支持含有通配符\*和?的**模糊查找**。  使用了动态规划的方法，支持基于LCS评分的**模糊查找**。   1. **数据结构**   使用了**一维数组**  使用了**二维数组**  使用了**嵌套结构体**、**结构体数组**  使用了包括**指向指针的指针**、**结构体指针**和**函数指针**在内的多种指针  使用了**单向链表**   1. **程序主流程图**   系统总体流程图如下： | | | | | | | |
| **实验过程中遇到的问题及解决方法与思路：** | | | | | | | |
| **问题1**：用户信息中，无法用静态数组保存所属的商品与记录。  **原因**：若用静态数组保存，因无法得知每个用户究竟会下属多少商品和记录，所以只能按最大值声明数组大小，内存开销过大（每条用户信息占用数MB的空间）。  **解决方案**：用动态数据结构（单向链表）保存这些信息。保存时，为节约内存，链表元素的值域仅为一个整型int，表示对应的商品/记录编号。这样，每条用户信息的空间开销节约至136字节。  **问题2**：采用链表储存信息后，无法直接用文件存档与读取这些信息。  **原因**：虽然程序会自动存读档，但由于内存分配的随机性，上一次运行保存的地址不能被下一次利用。  **解决方案**：存档时，将链表的值保存起来；下一次运行读档时，读取这些值，重建链表。  **问题3：**用户信息中含有密码等私密信息，若明文保存在文件中，容易被不当利用。  **解决方案：**用MD5算法加密后保存。用户登录时，将其输入的密码计算MD5值后与正确密码的MD5值比对。若二者相等，则视为密码正确；否则视为密码错误。  **问题4：**写MD5模块时，测试用例的哈希结果与期望值相去甚远。  **原因：**对MD5算法理解不深入，代码存在较多错误。  **解决方案：**到某以b开头的视频网站上搜索了MD5算法演示动画，深入理解算法后重构了代码，取得了较好的效果。  **问题5：**程序重启后登录，正确地输入密码的情况下程序显示密码错误。  **原因：**MD5算法模块中，计算MD5值钱未彻底地对相关数组和变量进行初始化。  **解决方案：**将M数组、sex数组、str字符串、tmp字符串都进行了初始化。  **问题6**：代码中出现了较多的if (X == 0) { return 0; } 语句，削弱了代码的紧凑性和可读性。  **原因**：程序中存在较多的合法性检查函数，且函数中的判断条件多为“且”，只要一个条件不满足就可返回0。如果用一堆if (X == 0) { return 0; }，会占用三行空间，不优雅、不美观。  **解决方案**：定义了宏EXIT\_CHECK(X)和BREAK\_CHECK(X)，只需用一行代码即可完成，优雅又美观。  **问题7**：三个用数据块读写的文件，无法正常读入。  **原因**：以users.info为例，写/读文件时，我这样调用了fread和fwrite函数：  fread(user, sizeof(User), total\_user\_cnt, file\_users);  fwrite(user, sizeof(User), total\_user\_cnt, file\_users);  然而，我的结构体数组却是从下表为1的元素开始利用。这样就无法准确、完整地写入所需信息。  **解决方案**：将代码改为：  fread(user + 1, sizeof(User), total\_user\_cnt, file\_users);  fwrite(user + 1, sizeof(User), total\_user\_cnt, file\_users);  **问题8**：基于通配符\*和?的模糊匹配，对于某些数据无法输出正确结果。如：  文本串T：Goods1  模式串P：o\*s  **原因**：通配符匹配的递推数组d初始化错误。  **解决方案**：将d[0][0..len\_T]均初始化为1 | | | | | | | |
| **测试用例和系统测试结果：** | | | | | | | |
| **测试用例1**：对MD5加密模块的测试  **输入**：尝试加密字符串 ILoveU：    与使用在线查询MD5的工具给出的加密串比对：    二者是一致的。  **结论**：经过多组测试，可以说MD5加密模块表现良好。  **测试用例2**：对一般模糊搜索模块的测试  **输入**：    其中，第一行的字符串是模式串（即用户输入的搜索串），第三行至第十行的8个字符串是待匹配的文本串（即商品名）。  **输出**：    程序按相似度（第三列数值）从大到小的顺序输出了8个字符串。  例如：  - 第一个串strstrstr完全匹配了模式串str三次，相似度最高  - ssstr、strrr和srstrrtsrts均完全匹配了一次，但前两者长度与str更接近，故相似度稍高  - 对于没有完全匹配的串（st等），也以最长公共子序列为指标计算了相似度，并将其排在了搜索结果中的合适位置。  **结论**：一般模糊匹配模块运行良好。  **测试用例3**：对通配符搜索模块的测试  **输入**：    其中，第一行为用户输入的搜索串，含有通配符?和\*，且为大小写混杂。第三行至第十行为文本串。  **输出**：    输出列表左侧为字符串，右侧1/0为是否匹配（搜索过程忽略大小写）。例如strrr满足s?R\*形态，而ssstttrrr则不满足。  **结论**：通配符匹配模块运行良好。  **测试用例4**：对购物系统的测试  **输入**：登录系统并完成若干购物操作后，进入个人菜单检查个人购物记录。  **输出**：    可见系统正确地显示了购物记录，并给出了总消费额。  此外，我还设计并运行了大量其他测试用例，以测试增删改相关信息是否运行正常，菜单、提示等信息显示是否正确、合适。经过大量的调试、测试与修改，目前系统运行良好。 | | | | | | | |
| **程序的全部源代码：** | | | | | | | |
| 1 */\**  2 *TYPE C*  3 *NAME main.c*  4 *PATH ./main.c*  5 *\*/*  6  7 #include "run.h"  8  9 int main()  10 {  11 run\_manager();  12  13 return 0;  14 }  15  16  17  18 */\**  19 *TYPE HEADER*  20 *NAME run.h*  21 *PATH ./run.h*  22 *\*/*  23  24 #ifndef TRADE\_MANAGER\_SYSTEM\_RUN\_H  25 #define TRADE\_MANAGER\_SYSTEM\_RUN\_H  26  27 #include "lib.h"  28  29 void run\_manager();  30  31 void run\_manager()  32 {  33 initialize();  34  35 while (cur\_state != -1) {  36  37 switch (cur\_state) {  38  39 case 0:  40 show\_welcome();  41 break;  42  43 case 1:  44 show\_register();  45 break;  46  47 case 2:  48 show\_login();  49 break;  50  51 case 3:  52 show\_main\_menu();  53 break;  54  55 case 31:  56 show\_personal\_menu();  57 break;  58  59 case 312:  60 show\_seller\_menu();  61 break;  62  63 case 32:  64 show\_shopping\_menu();  65 break;  66  67 case 33:  68 save\_all();  69  70 case -1:  71 break;  72  73 default:  74 break;  75 }  76 *// break to here*  77 }  78  79 save\_all();  80  81 cur\_state = -11;  82 print\_remind\_message(0);  83  84 }  85  86 #endif *//TRADE\_MANAGER\_SYSTEM\_RUN\_H*  87  88  89  90 */\**  91 *TYPE HEADER*  92 *NAME lib.h*  93 *PATH ./lib.h*  94 *\*/*  95  96 #ifndef TRADE\_MANAGER\_SYSTEM\_LIB\_H  97 #define TRADE\_MANAGER\_SYSTEM\_LIB\_H  98  99 #include "lib/basic\_def.h"  100  101 #include "lib/basic\_io.h"  102 #include "lib/functions.h"  103 #include "lib/link\_op.h"  104 #include "lib/checker.h"  105  106 #include "lib/file\_op.h"  107 #include "lib/info\_op.h"  108 #include "lib/MD5.h"  109  110 #include "lib/type\_io.h"  111 #include "lib/search.h"  112  113 #include "lib/initialize.h"  114 #include "lib/wel\_log\_reg.h"  115 #include "lib/surface.h"  116  117 #endif *//TRADE\_MANAGER\_SYSTEM\_LIB\_H*  118  119  120  121 */\**  122 *TYPE HEADER*  123 *NAME basic\_def.h*  124 *PATH ./lib/basic\_def.h*  125 *\*/*  126  127 #ifndef TRADE\_MANAGER\_SYSTEM\_GLOBALS\_H  128 #define TRADE\_MANAGER\_SYSTEM\_GLOBALS\_H  129  130 #include <stdio.h>  131 #include <unistd.h>  132 #include <string.h>  133 #include <stdlib.h>  134 #include <ctype.h>  135 #include <math.h>  136  137 #define version\_length 20  138 #define MD5\_length 4 + 1  139  140 #define name\_max\_length 32  141 #define input\_max\_length 256  142 #define path\_max\_length 64  143  144 #define max\_goods\_num 10000  145 #define max\_record\_num 100000  146 #define max\_user\_num 10000  147  148 #define max\_storage 1000000  149 #define max\_price 1000000  150  151 #define ERROR (1<<30)-1 *// 错误值*  152 #define EXIT\_CHECK(X) if (X == 0) return 0 *// return宏*  153 #define BREAK\_CHECK(X) if (X == 0) break *// break宏*  154  155 int cur\_state; *// 当前命令行状态*  156 char input\_str[input\_max\_length]; *// 用于储存输入字符串*  157  158 int id\_num; *// 当前登录的用户编号*  159  160 int choice; *// 用户选择号*  161  162 char path\_basic[path\_max\_length] = "../info/basic.info";  163 char path\_users[path\_max\_length] = "../info/users.info";  164 char path\_goods[path\_max\_length] = "../info/goods.info";  165 char path\_records[path\_max\_length] = "../info/records.info";  166 char path\_links[path\_max\_length] = "../info/links.info";  167  168 int total\_users\_cnt; *// 总用户数*  169 int total\_goods\_cnt; *// 总商品数*  170 int total\_records\_cnt; *// 总记录数*  171  172 *// 修改标记*  173 int basic\_modified;  174 int users\_modified;  175 int goods\_modified;  176 int records\_modified;  177 int links\_modified;  178  179 *// 五个文件指针*  180 FILE\* file\_basic;  181 FILE\* file\_users;  182 FILE\* file\_goods;  183 FILE\* file\_records;  184 FILE\* file\_links;  185  186 *// 三个函数指针*  187 typedef int (\*checker\_sgl)(char\*);  188 typedef int (\*checker\_num)(char\*, int, int );  189 typedef int (\*print\_info)(int, int);  190  191 struct Version  192 {  193 char version\_number[version\_length]; *// 版本号*  194 char released\_time[version\_length]; *// 释出时间*  195 } current\_version;  196  197 *// 链表元素，值域为信息编号*  198 typedef struct linkNode  199 {  200 int val;  201 struct linkNode\* nxt;  202 } lnk;  203  204 *// 编号用number、num，数量用count、cnt*  205  206 typedef struct seller  207 {  208 int goods\_cnt; *// 商品数量*  209 lnk\* goods\_head; *// 商品链表头*  210 lnk\* goods\_tail; *// 商品链表尾*  211  212 int record\_cnt; *// 记录数量*  213 lnk\* record\_head; *// 记录链表头*  214 lnk\* record\_tail; *// 记录链表尾*  215 } Seller;  216  217 typedef struct user  218 {  219 int number; *// 用户编号*  220 unsigned encrypted[MD5\_length]; *// MD5加密后的4个16进制数*  221  222 char name[name\_max\_length]; *// 用户名*  223  224 int record\_cnt; *// 记录数量*  225 lnk\* record\_head; *// 记录链表头*  226 lnk\* record\_tail; *// 记录链表尾*  227  228 int is\_seller; *// 是否是商家用户*  229  230 Seller sell;  231 } User;  232 User user[max\_user\_num];  233  234 typedef struct goods  235 {  236 int number; *// 商品编号*  237 int seller\_num; *// 售卖者的用户编号*  238 int deleted; *// 是否已被删除*  239  240 int price; *// 价格*  241 int stock\_cnt; *// 库存量*  242  243 char name[name\_max\_length]; *// 商品名*  244  245 } Goods;  246 Goods goods[max\_goods\_num];  247  248 typedef struct record  249 {  250 int number; *// 记录编号*  251  252 int goods\_num; *// 交易商品编号*  253 int seller\_num; *// 售卖者的用户编号*  254 int buyer\_num; *// 购买者的用户编号*  255  256 int qnt; *// 交易数量*  257 } Record;  258 Record record[max\_record\_num];  259  260  261 #endif *//TRADE\_MANAGER\_SYSTEM\_GLOBALS\_H*  262  263  264  265 */\**  266 *TYPE HEADER*  267 *NAME basic\_io.h*  268 *PATH ./lib/basic\_io.h*  269 *\*/*  270  271 #ifndef TRADE\_MANAGER\_SYSTEM\_CUSTIO\_H  272 #define TRADE\_MANAGER\_SYSTEM\_CUSTIO\_H  273  274 #include "basic\_def.h"  275  276 void get\_the\_line(char \*str, int need\_all);  277  278 void print\_waiting\_chars();  279 void print\_error(int error\_id);  280 void print\_remind\_message(int need);  281  282 void get\_the\_line(char \*str, int need\_all)  283 *// need\_all = 1 表示不需要加工*  284 {  285 char c;  286 char line[input\_max\_length];  287  288 int cnt = 0;  289 do {  290 scanf("%c", &c);  291 line[cnt] = c;  292 cnt++;  293 } while (c != '\n');  294 line[--cnt] = '\0';  295  296 printf("\n");  297  298 if (need\_all) {  299 strcpy(str, line);  300 return;  301 }  302  303 *// 去空格，规范化用户输入*  304 int i = 0, j = 0;  305 while (line[i] == ' ') {  306 i++;  307 }  308  309 while (i < cnt) {  310 while (line[i] != ' ' && i < cnt) {  311 str[j++] = line[i++];  312 }  313  314 if (line[i] == ' ') {  315 str[j++] = line[i++];  316 }  317  318 while (line[i] == ' ') {  319 i++;  320 }  321 }  322  323 if (str[j - 1] == ' ') {  324 j--;  325 }  326 str[j] = '\0';  327 }  328  329 void print\_waiting\_chars()  330 {  331 printf(">> ");  332 }  333  334 void wait\_continue()  335 {  336 printf("Press Enter to continue: ");  337 getchar();  338 printf("\n");  339 }  340  341  342 void print\_error(int error\_id)  343 {  344 printf("Error %d: ", error\_id);  345  346 switch (error\_id) {  347  348 case 0:  349 printf("Illegal input.\n");  350 printf("Please try again.\n");  351 print\_waiting\_chars();  352 break;  353  354 case 20:  355 printf("Cannot load files!\n");  356 printf("The system will reconstruct new empty ones.\n");  357 break;  358  359 default:  360 printf("Fatal Error.\nAuto get back or exit.\n");  361 break;  362 }  363  364 }  365  366 void print\_remind\_message(int need)  367 {  368 switch (cur\_state) {  369  370 case 0: *// 欢迎界面*  371 system("clear"); *// Windows + cmd运行需要改为 system("cls")*  372 printf("\n ------------- Welcome Menu ------------- \n");  373 printf( "| |\n");  374 printf( "| |\n");  375 printf( "| 0 Save and quit |\n");  376 printf( "| |\n");  377 printf( "| 1 Register |\n");  378 printf( "| |\n");  379 printf( "| 2 Login |\n");  380 printf( "| |\n");  381 printf( "| |\n");  382 printf( " ---------------------------------------- \n\n");  383 break;  384  385 case 11: *// 注册，输入用户名*  386 printf("Please enter your user name.\nYour name should:\n");  387 printf("- Contain at least 3 characters and no more than %d characters.\n", name\_max\_length);  388 printf("- Only letters, digits and '\_' are allowed.\n");  389 break;  390  391 case 111:  392 printf("Valid user name.\nAccept it? [y/n]\n");  393 break;  394  395 case 112: *// 注册，输入密码*  396 printf("Please enter your password.\nYour password should:\n");  397 printf("- No less than 8 characters and no more than 16 characters.\n");  398 printf("- CanNOT only contain upper letters, lower letters, digits or special characters.\n");  399 break;  400  401 case 1121:  402 printf("Valid password.\nPlease enter it again to verify.\n");  403 break;  404  405 case 1120:  406 printf("InValid password!\nPlease try again.\n");  407  408 case 11201:  409 printf("Successfully Registered!\n");  410 printf("Your userID: %d\n", total\_users\_cnt);  411 break;  412  413 case 21:  414 printf("Please enter your userID, or enter '0' to get back.\n");  415 break;  416  417 case 22:  418 printf("Please enter your password.\n");  419 printf("You can try no more than 3 times.\n");  420 break;  421  422 case 2202:  423 printf("You can try no more than 2 times.\n");  424 break;  425  426 case 2201:  427 printf("You can try no more than 1 times.\n");  428 break;  429  430 case 2200:  431 printf("Login Failed!\nAuto get back to Welcome menu.\n");  432 break;  433  434 case 221:  435 printf("Successfully login!\n");  436 break;  437  438 case 3: *// 主菜单*  439 system("clear");  440 printf("\n ------------ Main Menu ------------ \n");  441 printf( "| |\n");  442 printf( "| |\n");  443 printf( "| 0 Logout |\n");  444 printf( "| |\n");  445 printf( "| 1 Personal |\n");  446 printf( "| |\n");  447 printf( "| 2 Shopping |\n");  448 printf( "| |\n");  449 printf( "| |\n");  450 printf( " ----------------------------------- \n\n");  451 break;  452  453 case 31: *// 个人菜单*  454 system("clear");  455 printf("\n ---------------- Personal Menu ---------------- \n");  456 printf( "| |\n");  457 printf( "| |\n");  458 printf( "| 0 Back to superior menu |\n");  459 printf( "| |\n");  460 printf( "| 1 Check my trade record (buy) |\n");  461 printf( "| |\n");  462 if (user[id\_num].is\_seller == 0) {  463 printf( "| 2 Apply for seller qualification |\n"); }  464 else {  465 printf( "| 2 Seller's Menu |\n"); }  466 printf( "| |\n");  467 printf( "| |\n");  468 printf( " ----------------------------------------------- \n\n");  469 break;  470  471 case 3121:  472 printf("Successfully applied!\n");  473 break;  474  475 case 312: *// 商家菜单*  476 system("clear");  477 printf("\n ------------------- Seller Menu ------------------- \n");  478 printf( "| |\n");  479 printf( "| |\n");  480 printf( "| 0 Back to superior menu |\n");  481 printf( "| |\n");  482 printf( "| 1 Check my goods list |\n");  483 printf( "| |\n");  484 printf( "| 2 Check my trade record (sell) |\n");  485 printf( "| |\n");  486 printf( "| 3 Add goods |\n");  487 printf( "| |\n");  488 printf( "| 4 Delete goods |\n");  489 printf( "| |\n");  490 printf( "| 5 Modify storage |\n");  491 printf( "| |\n");  492 printf( "| |\n");  493 printf( " -------------------------------------------------- \n\n");  494 break;  495  496 *// 添加商品*  497 case 31231:  498 printf("Please enter the goods name.\nThe name should:\n");  499 printf("- Contain at least 3 characters and no more than %d characters.\n", name\_max\_length);  500 printf("- Only letters, digits and '\_' are allowed.\n");  501 break;  502  503 case 31232:  504 printf("Please enter the storage.\n");  505 printf("The number should be no more than %d.\n", max\_storage);  506 break;  507  508 case 31233:  509 printf("Please enter the price.\n");  510 printf("The number should be no more than %d.\n", max\_price);  511 break;  512  513 case 31234:  514 printf("Successfully added!\n");  515 printf("The goodsID: %d\n", total\_goods\_cnt);  516 break;  517  518 *// 删除商品*  519 case 31241:  520 printf("Please enter the goodsID.\n");  521 break;  522  523 case 31242:  524 printf("Successfully deleted!\n");  525 break;  526  527 *// 更改商品库存量*  528 case 31251:  529 printf("Please enter the goodsID.\n");  530 break;  531  532 case 31252:  533 printf("Please enter the quantity you want to modify.\nThe number should:\n");  534 printf("- After modified, the storage should be no more than %d.\n", max\_storage);  535 printf("- After modified, the storage should be no less than 0.\n");  536 break;  537  538 case 31253:  539 printf("Successfully modified!\n");  540 break;  541  542 case 32: *// 购物菜单*  543 system("clear");  544 printf("\n ------------------ Shopping Menu -------------------- \n");  545 printf( "| |\n");  546 printf( "| |\n");  547 printf( "| 0 Back to superior menu |\n");  548 printf( "| |\n");  549 printf( "| 1 Search (support wildcard '\*' and '?') |\n");  550 printf( "| |\n");  551 printf( "| |\n");  552 printf( " ----------------------------------------------------- \n\n");  553 break;  554  555 *// 搜索界面*  556 case 321:  557 printf("Please input the keyword.\nThe keyword should:\n");  558 printf("- No more than %d characters.\n", name\_max\_length);  559 printf("- Only letters, digits, '\_', '?', '\*' are allowed.\n");  560 break;  561  562 case 3210:  563 printf("No results.\n");  564 break;  565  566 case 3211:  567 printf("Please enter the order of your intended goods.\n");  568 printf("Enter '0' to get back.\n");  569 break;  570  571 case 3212:  572 printf("Please enter the quantity you want to buy.\n");  573 printf("Enter '0' to get back.\n");  574 break;  575  576 case 3213:  577 printf("Successfully traded!\n");  578 break;  579  580 case 331:  581 printf("Saving...\n");  582 break;  583  584 case 332:  585 printf("Successfully saved!\n");  586 break;  587  588 case -10:  589 printf("Saving and quitting...\n");  590 break;  591  592 case -11:  593 printf("\nSee you.\n");  594 break;  595  596 default:  597 *// printf("state = %d: Unfinished.\n", cur\_state);*  598 break;  599  600 }  601 if (need == 1) {  602 print\_waiting\_chars();  603 }  604 }  605  606 #endif *//TRADE\_MANAGER\_SYSTEM\_CUSTIO\_H*  607  608  609  610 */\**  611 *TYPE HEADER*  612 *NAME functions.h*  613 *PATH ./lib/functions.h*  614 *\*/*  615  616 #ifndef TRADE\_MANAGER\_SYSTEM\_FUNCTIONS\_H  617 #define TRADE\_MANAGER\_SYSTEM\_FUNCTIONS\_H  618  619 #include "basic\_def.h"  620  621 int is\_number(char\* str);  622 int get\_max(int x, int y);  623  624 int is\_number(char \*str)  625 { *//判断当前字符串是否是纯数字*  626 int len = (int)strlen(str);  627 EXIT\_CHECK(len > 0);  628  629 int start\_pos;  630  631 start\_pos = ((str[0] == '-') | (str[0] == '+')) ? 1 : 0;  632 int is\_not\_char = !((start\_pos == 1) & (len == 1));  633 EXIT\_CHECK(is\_not\_char);  634  635 for (int i = start\_pos; i < len; i++) {  636 EXIT\_CHECK(isdigit(str[i]));  637 }  638 return 1;  639 }  640  641 int get\_max(int x, int y)  642 {  643 return x >= y ? x : y;  644 }  645  646 #endif *//TRADE\_MANAGER\_SYSTEM\_FUNCTIONS\_H*  647  648  649  650 */\**  651 *TYPE HEADER*  652 *NAME file\_op.h*  653 *PATH ./lib/file\_op.h*  654 *\*/*  655  656 #ifndef TRADE\_MANAGER\_SYSTEM\_FILE\_OP\_H  657 #define TRADE\_MANAGER\_SYSTEM\_FILE\_OP\_H  658  659 #include "basic\_def.h"  660 #include "basic\_io.h"  661 #include "link\_op.h"  662  663 int check\_file\_open(const char\* file\_name);  664 int check\_file\_write(const char\* file\_name);  665 void write\_new\_user(User \*cur);  666 void write\_new\_goods(Goods \*cur);  667 void reset\_modify\_signs();  668 void save\_all();  669  670 int check\_file\_open(const char\* file\_name)  671 {  672 return access(file\_name, F\_OK) == 0;  673 }  674  675 int check\_file\_write(const char\* file\_name)  676 {  677 int open\_ok = check\_file\_open(file\_name);  678 EXIT\_CHECK(open\_ok);  679 return (access(file\_name, W\_OK) == 0);  680 }  681  682 void write\_new\_user(User \*cur)  683 {  684 fwrite(cur, sizeof(User), 1, file\_users);  685 fflush(file\_users);  686 }  687  688 void write\_new\_goods(Goods \*cur)  689 {  690 fwrite(cur, sizeof(Goods), 1, file\_goods);  691 fflush(file\_goods);  692 }  693  694 void reset\_modify\_signs()  695 {  696 basic\_modified = 0;  697 users\_modified = 0;  698 goods\_modified = 0;  699 records\_modified = 0;  700 links\_modified = 0;  701 }  702  703 void save\_all()  704 *// cur\_state = 33*  705 {  706 cur\_state = 331;  707 print\_remind\_message(0);  708  709 if (basic\_modified == 1) {  710 file\_basic = fopen(path\_basic, "w");  711 fprintf(file\_basic, "%d %d %d\n", total\_users\_cnt, total\_goods\_cnt,total\_records\_cnt);  712 fflush(file\_basic); *// 将缓冲区中的内容及时写入文件*  713 }  714 if (users\_modified == 1) {  715 file\_users = fopen(path\_users, "w");  716 fwrite(user + 1, sizeof(User), (unsigned)total\_users\_cnt, file\_users);  717 fflush(file\_users);  718 }  719 if (goods\_modified == 1) {  720 file\_goods = fopen(path\_goods, "w");  721 fwrite(goods + 1, sizeof(Goods), (unsigned)total\_goods\_cnt, file\_goods);  722 fflush(file\_goods);  723 }  724 if (records\_modified == 1) {  725 file\_records = fopen(path\_records, "w");  726 fwrite(record + 1, sizeof(Record), (unsigned)total\_records\_cnt, file\_records);  727 fflush(file\_records);  728 }  729 if (links\_modified == 1) {  730 file\_links = fopen(path\_links, "w");  731 for (int i = 1; i <= total\_users\_cnt; i++) {  732 table\_write(file\_links, user[i].record\_head);  733 if (user[i].is\_seller == 1) {  734 table\_write(file\_links, user[i].sell.goods\_head);  735 table\_write(file\_links, user[i].sell.record\_head);  736 }  737 }  738 }  739  740 reset\_modify\_signs();  741  742 cur\_state = 332;  743 print\_remind\_message(0);  744  745 cur\_state = 3;  746 }  747  748 #endif *//TRADE\_MANAGER\_SYSTEM\_FILE\_OP\_H*  749  750  751  752 */\**  753 *TYPE HEADER*  754 *NAME info\_op.h*  755 *PATH ./lib/info\_op.h*  756 *\*/*  757  758 #ifndef TRADE\_MANAGER\_SYSTEM\_INFO\_OP\_H  759 #define TRADE\_MANAGER\_SYSTEM\_INFO\_OP\_H  760  761 #include "basic\_def.h"  762 #include "basic\_io.h"  763 #include "file\_op.h"  764 #include "link\_op.h"  765  766 int add\_ok;  767  768 void add\_user(const char \*user\_name, const unsigned \*reg\_pw); *// pw为password缩写*  769 void register\_seller();  770  771 int add\_goods(const char \*goods\_name, int storage, int price);  772 void del\_goods(int goods\_id);  773 void mod\_goods\_stock(int goods\_id, int qnt); *// qnt为quantity缩写*  774  775 int add\_record(int goods\_id, int seller\_id, int qnt);  776  777 int buy\_goods(int goods\_id, int qnt);  778  779  780 void add\_user(const char \*user\_name, const unsigned \*reg\_pw)  781 {  782 basic\_modified = 1;  783 links\_modified = 1;  784  785 total\_users\_cnt++;  786 User \*cur = &user[total\_users\_cnt];  787  788 cur->number = total\_users\_cnt;  789  790 strcpy(cur->name, user\_name);  791  792 for (int i = 1; i <= 4; i++)  793 cur->encrypted[i] = reg\_pw[i];  794  795 cur->is\_seller = 0;  796  797 cur->record\_cnt = 0;  798 cur->record\_head = NULL;  799 cur->record\_tail = NULL;  800  801 file\_users = fopen(path\_users, "a");  802 write\_new\_user(&user[total\_users\_cnt]);  803 }  804  805 void register\_seller()  806 {  807 users\_modified = 1;  808  809 User \*cur = &user[id\_num];  810 cur->is\_seller = 1;  811  812 cur->sell.goods\_cnt = 0;  813 cur->sell.record\_head = NULL;  814 cur->sell.record\_tail = NULL;  815  816 cur->sell.record\_cnt = 0;  817 cur->sell.goods\_head = NULL;  818 cur->sell.goods\_tail = NULL;  819 }  820  821 int add\_goods(const char \*goods\_name, int storage, int price)  822 {  823 basic\_modified = 1;  824 users\_modified = 1;  825 links\_modified = 1;  826  827 total\_goods\_cnt++;  828 Goods \*cur = &goods[total\_goods\_cnt];  829  830 *// 写入新商品信息*  831 cur->number = total\_goods\_cnt;  832 strcpy(cur->name, goods\_name);  833 cur->stock\_cnt = storage;  834 cur->price = price;  835 cur->seller\_num = id\_num;  836  837 *// 将新商品编号挂在当前用户商品链表上*  838 add\_ok = table\_append(&user[id\_num].sell.goods\_cnt,  839 &user[id\_num].sell.goods\_head, &user[id\_num].sell.goods\_tail, total\_goods\_cnt);  840  841 EXIT\_CHECK(add\_ok);  842  843 file\_goods = fopen(path\_goods, "a");  844 write\_new\_goods(&goods[total\_goods\_cnt]);  845  846 return 1;  847 }  848  849 void del\_goods(int goods\_id)  850 {  851 links\_modified = 1;  852 basic\_modified = 1;  853 goods\_modified = 1;  854 users\_modified = 1;  855  856 goods[goods\_id].deleted = 1;  857 user[id\_num].sell.goods\_head =  858 delete\_linkNode(user[id\_num].sell.goods\_head, goods[goods\_id].number);  859 user[id\_num].sell.goods\_tail =  860 get\_tail\_linkNode(user[id\_num].sell.goods\_head);  861  862 user[id\_num].sell.goods\_cnt--;  863 }  864  865 void mod\_goods\_stock(int goods\_id, int qnt)  866 {  867 goods[goods\_id].stock\_cnt += qnt;  868 }  869  870 int add\_record(int goods\_id, int seller\_id, int qnt)  871 {  872 basic\_modified = 1;  873 records\_modified = 1;  874 users\_modified = 1;  875  876 total\_records\_cnt++;  877 Record\* cur = &record[total\_records\_cnt];  878  879 cur->number = total\_records\_cnt;  880 cur->goods\_num = goods\_id;  881 cur->buyer\_num = id\_num;  882 cur->seller\_num = seller\_id;  883 cur->qnt = qnt;  884  885 return 1;  886 }  887  888 int buy\_goods(int goods\_id, int qnt)  889 {  890 basic\_modified = 1;  891 goods\_modified = 1;  892 users\_modified = 1;  893 records\_modified = 1;  894 links\_modified = 1;  895  896 Goods \*cur = &goods[goods\_id];  897 cur->stock\_cnt -= qnt;  898  899 add\_record(goods\_id, cur->seller\_num, qnt);  900  901 int buy\_id = id\_num;  902 int sel\_id = goods[goods\_id].seller\_num;  903  904 add\_ok = table\_append(&user[buy\_id].record\_cnt,  905 &user[buy\_id].record\_head, &user[buy\_id].record\_tail, total\_records\_cnt);  906  907 EXIT\_CHECK(add\_ok);  908  909 add\_ok = table\_append(&user[sel\_id].sell.record\_cnt,  910 &user[sel\_id].sell.record\_head, &user[sel\_id].sell.record\_tail, total\_records\_cnt);  911  912 EXIT\_CHECK(add\_ok);  913  914 return 1;  915 }  916  917 #endif *//TRADE\_MANAGER\_SYSTEM\_INFO\_OP\_H*  918  919  920  921 */\**  922 *TYPE HEADER*  923 *NAME link\_op.h*  924 *PATH ./lib/link\_op.h*  925 *\*/*  926  927 #ifndef TRADE\_MANAGER\_SYSTEM\_LINK\_OP\_H  928 #define TRADE\_MANAGER\_SYSTEM\_LINK\_OP\_H  929  930 #include "basic\_def.h"  931 #include "basic\_io.h"  932  933 lnk\* append\_linkNode(lnk\* prev, int id);  934 int display\_linkNode(lnk\* frst, print\_info printer);  935 lnk\* delete\_linkNode(lnk\* head, int node\_val);  936 lnk\* get\_tail\_linkNode(lnk\* head);  937  938 int table\_reconstruct(FILE\* fp, lnk\*\* pHead, lnk\*\* pHail, int len);  939 int table\_append(int\* len, lnk\*\* pHead, lnk\*\* pTail, int val);  940 void table\_write(FILE\* fp, lnk\* pHead);  941  942 lnk\* append\_linkNode(lnk\* prev, int id)  943 *// 向链表尾添加节点，这里是从指定节点prev而不是head开始往后找*  944 *// prev往往传入的是链表尾，这样可以提高查找效率*  945 *// 这个函数无法向空链表中加入头结点，外部调用时应先调用table\_append*  946 {  947 lnk\* p = NULL;  948 lnk\* pr = prev;  949  950 p = (lnk\*)malloc(sizeof(lnk));  951 if (p == NULL) {  952 print\_error(21);  953 return NULL;  954 }  955 if (prev == NULL) {  956 prev = p;  957 }  958 else {  959 while (pr->nxt != NULL) {  960 pr = pr->nxt;  961 }  962 pr->nxt = p;  963 }  964 p->val = id;  965 p->nxt = NULL;  966  967 return prev;  968 }  969  970 int display\_linkNode(lnk\* frst, print\_info printer)  971 {  972 lnk\* p = frst;  973 int j = 1;  974 int ret = 0;  975  976 while (p != NULL) {  977 ret += printer(j, p->val);  978 p = p->nxt;  979 j++;  980 }  981  982 return ret;  983 }  984  985 lnk\* delete\_linkNode(lnk\* head, int node\_val)  986 {  987 lnk\* p = head;  988 lnk\* pr = head;  989 if (head == NULL) {  990 print\_error(30);  991 return head;  992 }  993  994 while (node\_val != p->val && p->nxt != NULL) {  995 *// 找到待删除节点p的前驱节点pr*  996 pr = p;  997 p = p->nxt;  998 }  999  1000 if (node\_val == p->val) {  1001 if (p == head) {  1002 head = p->nxt;  1003 }  1004 else {  1005 pr->nxt = p->nxt;  1006 }  1007 free(p);  1008 }  1009 else {  1010 print\_error(31);  1011 return head;  1012 }  1013  1014 return head;  1015 }  1016  1017 lnk\* get\_tail\_linkNode(lnk\* head)  1018 {  1019 lnk\* p = head;  1020 if (p == NULL) {  1021 return NULL;  1022 }  1023  1024 while (p->nxt != NULL) {  1025 p = p->nxt;  1026 }  1027  1028 return p;  1029 }  1030  1031 int table\_reconstruct(FILE\* fp, lnk\*\* pHead, lnk\*\* pHail, int len)  1032 {  1033 lnk\* head = NULL;  1034 lnk\* tail = NULL;  1035  1036 int read\_ok;  1037 int cur\_len = 0;  1038 int val;  1039 for (int i = 1; i <= len; i++) {  1040 read\_ok = fscanf(fp, "%d", &val);  1041 EXIT\_CHECK(read\_ok);  1042 table\_append(&cur\_len, &head, &tail, val);  1043 }  1044  1045 (\*pHead) = head;  1046 (\*pHail) = tail;  1047  1048 return 1;  1049 }  1050  1051 int table\_append(int\* len, lnk\*\* pHead, lnk\*\* pTail, int val)  1052 *// 拥有完备接口的向链表尾添加元素的函数*  1053 {  1054 lnk\* head = \*pHead;  1055 lnk\* tail = \*pTail;  1056  1057 if (\*len == 0) { *// 若为空链表*  1058 lnk\* res = append\_linkNode(head, val);  1059 EXIT\_CHECK(res != NULL);  1060 head = res;  1061 tail = head;  1062 }  1063 else {  1064 lnk\* res = append\_linkNode(tail, val);  1065 EXIT\_CHECK(res != NULL);  1066 tail = res->nxt;  1067 }  1068  1069  1070 (\*len)++;  1071  1072 \*pHead = head;  1073 \*pTail = tail;  1074  1075 return 1;  1076 }  1077  1078 void table\_write(FILE\* fp, lnk\* head)  1079 {  1080 lnk\* p = head;  1081 while (p != NULL) {  1082 fprintf(fp, "%d ", p->val);  1083 p = p->nxt;  1084 }  1085 }  1086  1087 #endif *//TRADE\_MANAGER\_SYSTEM\_LINK\_OP\_H*  1088  1089  1090  1091 */\**  1092 *TYPE HEADER*  1093 *NAME checker.h*  1094 *PATH ./lib/checker.h*  1095 *\*/*  1096  1097 #ifndef TRADE\_MANAGER\_SYSTEM\_CHECKER\_H  1098 #define TRADE\_MANAGER\_SYSTEM\_CHECKER\_H  1099  1100 #include "basic\_def.h"  1101 #include "functions.h"  1102  1103 int legal\_check\_return\_num(char \*str, int lower, int upper);  1104 int legal\_check\_name(char \*str);  1105 int legal\_check\_pw(char\* str);  1106 int legal\_check\_keyword(char\* str);  1107 int legal\_check\_yon(char \*str);  1108 int check\_MD5(const unsigned\* a, const unsigned\* b);  1109  1110 int legal\_check\_return\_num(char \*str, int lower, int upper)  1111 {  1112 if (is\_number(str) == 0) {  1113 return ERROR;  1114 }  1115  1116 int ret = atoi(str);  1117 if (ret >= lower && ret <= upper) {  1118 return ret;  1119 }  1120  1121 return ERROR;  1122 }  1123  1124 int legal\_check\_name(char \*str)  1125 {  1126 int len = (int)strlen(str);  1127 if (len < 3 || len > name\_max\_length) return 0;  1128 for (int i = 0; i < len; i++) {  1129 char cur = str[i];  1130 int name\_ok = 0;  1131 name\_ok |= isdigit(cur);  1132 name\_ok |= isupper(cur);  1133 name\_ok |= islower(cur);  1134 name\_ok |= (cur == '\_');  1135 EXIT\_CHECK(name\_ok);  1136 }  1137 return 1;  1138 }  1139  1140 int legal\_check\_pw(char\* str)  1141 {  1142 int len = (int)strlen(str);  1143 if (len < 8 || len > 16) return 0;  1144  1145 int has\_dg = 0;  1146 int has\_lc = 0;  1147 int has\_uc = 0;  1148 int has\_sp = 0;  1149  1150 for (int i = 0; i < len; i++) {  1151 char cur = str[i];  1152 if (cur < ' ' || cur > 126) {  1153 return 0;  1154 }  1155 has\_dg |= isdigit(cur);  1156 has\_lc |= islower(cur);  1157 has\_uc |= isupper(cur);  1158 has\_sp |= (has\_dg | has\_lc | has\_uc | has\_sp == 0);  1159 }  1160  1161 if (has\_dg + has\_lc + has\_uc + has\_sp < 2) {  1162 return 0;  1163 }  1164  1165 return 1;  1166 }  1167  1168 int legal\_check\_keyword(char \*str)  1169 {  1170 int len = (int)strlen(str);  1171 if (len > name\_max\_length) return 0;  1172 for (int i = 0; i < len; i++) {  1173 char cur = str[i];  1174 int kw\_ok = 0;  1175 kw\_ok |= isdigit(cur);  1176 kw\_ok |= isupper(cur);  1177 kw\_ok |= islower(cur);  1178 kw\_ok |= (cur == '\_');  1179 kw\_ok |= (cur == '?');  1180 kw\_ok |= (cur == '\*');  1181 EXIT\_CHECK(kw\_ok);  1182 }  1183 return 1;  1184  1185 }  1186  1187 int legal\_check\_yon(char \*str)  1188 {  1189 int len = (int)strlen(str);  1190  1191 if (len < 1 || len > 3) {  1192 return 0;  1193 }  1194  1195 if (len == 1) {  1196 if (str[0] == 'y' || str[0] == 'Y') return 1;  1197 if (str[0] == 'n' || str[0] == 'N') return 1;  1198 }  1199 else if (len == 2) {  1200 if (str[0] != 'n' && str[0] != 'N') return 0;  1201 if (str[1] != 'o' && str[1] != 'O') return 0;  1202 return 1;  1203 }  1204 else {  1205 if (str[0] != 'y' && str[0] != 'Y') return 0;  1206 if (str[1] != 'e' && str[1] != 'E') return 0;  1207 if (str[2] != 's' && str[2] != 'S') return 0;  1208 return 1;  1209 }  1210  1211 return 0;  1212 }  1213  1214 int check\_MD5(const unsigned\* a, const unsigned\* b)  1215 {  1216 for (int i = 1; i <= 4; i++) {  1217 EXIT\_CHECK(a[i] == b[i]);  1218 }  1219 return 1;  1220 }  1221  1222 #endif *//TRADE\_MANAGER\_SYSTEM\_CHECKER\_H*  1223  1224  1225  1226 */\**  1227 *TYPE HEADER*  1228 *NAME type\_io.h*  1229 *PATH ./lib/type\_io.h*  1230 *\*/*  1231  1232 #ifndef TRADE\_MANAGER\_SYSTEM\_TYPE\_IO\_H  1233 #define TRADE\_MANAGER\_SYSTEM\_TYPE\_IO\_H  1234  1235 #include "basic\_def.h"  1236 #include "basic\_io.h"  1237 #include "checker.h"  1238 #include "MD5.h"  1239  1240 void print\_buy\_records\_list();  1241 void print\_my\_goods\_list();  1242  1243 void print\_sell\_records\_list();  1244  1245 int print\_record(int order, int record\_id);  1246 int print\_goods(int order, int goods\_id);  1247 void print\_search\_item(int order, int goods\_id);  1248  1249 void input\_and\_check\_single(checker\_sgl legal\_check, char \*str);  1250 int input\_and\_check\_num(checker\_num legal\_check, char\* str, int lower, int upper);  1251  1252 void circulate\_confirm\_name(char \*str);  1253 void circulate\_confirm\_password(unsigned \*dest);  1254  1255  1256 int print\_record(int order, int record\_id)  1257 *// 打印单条记录信息*  1258 {  1259 Record rec = record[record\_id];  1260 Goods gds = goods[rec.goods\_num];  1261  1262 printf("|%6d |%9d |%8d | %32s |%6d |%9d |%14d |%15d |\n", order,  1263 rec.number, rec.goods\_num, gds.name, gds.price, rec.qnt, rec.buyer\_num, rec.seller\_num);  1264  1265 return rec.qnt \* gds.price;  1266 }  1267  1268 void print\_buy\_records\_list()  1269 {  1270 printf(" ------------------------------------------------------------------------------------------------------------------- \n");  1271 printf("| order | recordID | goodsID | goods name | price | quantity | userID(buyer) | userID(seller) |\n");  1272 int fee = display\_linkNode(user[id\_num].record\_head, print\_record);  1273 printf(" ------------------------------------------------------------------------------------------------------------------- \n\n");  1274 printf(" Total consuming: %d\n\n", fee);  1275 }  1276  1277 void print\_sell\_records\_list()  1278 {  1279 printf(" ------------------------------------------------------------------------------------------------------------------- \n");  1280 printf("| order | recordID | goodsID | goods name | price | quantity | userID(buyer) | userID(seller) |\n");  1281 int fee = display\_linkNode(user[id\_num].sell.record\_head, print\_record);  1282 printf(" ------------------------------------------------------------------------------------------------------------------- \n\n");  1283 printf(" Total turnover: %d\n\n", fee);  1284 }  1285  1286 int print\_goods(int order, int goods\_id)  1287 *// 打印单条商品信息*  1288 {  1289 Goods cur = goods[goods\_id];  1290 printf("|%6d |%8d | %32s |%6d |%8d |\n", order, cur.number, cur.name, cur.price, cur.stock\_cnt);  1291 return 0;  1292 }  1293  1294 void print\_my\_goods\_list()  1295 {  1296 printf(" ---------------------------------------------------------------------- \n");  1297 printf("| order | goodsID | goods name | price | storage |\n");  1298 display\_linkNode(user[id\_num].sell.goods\_head, print\_goods);  1299 printf(" ---------------------------------------------------------------------- \n\n");  1300 }  1301  1302 void print\_search\_item(int order, int goods\_id)  1303 {  1304 Goods cur = goods[goods\_id];  1305 printf("|%6d |%8d | %32s |%6d |%8d |%15d |\n", order,  1306 cur.number, cur.name, cur.price, cur.stock\_cnt, cur.seller\_num);  1307 }  1308  1309 void input\_and\_check\_single(checker\_sgl legal\_check, char\* str)  1310 *// 用于单个参数输入和检查*  1311 {  1312 int input\_ok = 0;  1313 while (!input\_ok) {  1314 get\_the\_line(str, 1);  1315  1316 if (legal\_check(str)) {  1317 input\_ok = 1;  1318 }  1319 else {  1320 print\_error(0);  1321 }  1322 }  1323 }  1324  1325 int input\_and\_check\_num(checker\_num legal\_check, char\* str, int lower, int upper)  1326 *// 用于单个整数输入和检查，返回输入的整数*  1327 {  1328 int input\_ok = 0, ret = -1;  1329 while (!input\_ok) {  1330 get\_the\_line(str, 1);  1331 ret = legal\_check(str, lower, upper);  1332 if (ret != ERROR) {  1333 input\_ok = 1;  1334 }  1335 else {  1336 print\_error(0);  1337 }  1338 }  1339 return ret;  1340 }  1341  1342 void circulate\_confirm\_name(char \*str)  1343 *// cur\_state = 11*  1344 {  1345 int feel\_ok = 0;  1346  1347 while (!feel\_ok) {  1348 print\_remind\_message(1);  1349 input\_and\_check\_single(legal\_check\_name, str);  1350  1351 cur\_state = 111;  1352 print\_remind\_message(1);  1353  1354 char yes\_or\_no[input\_max\_length];  1355 input\_and\_check\_single(legal\_check\_yon, yes\_or\_no);  1356 if (yes\_or\_no[0] == 'y' || yes\_or\_no[0] == 'Y') {  1357 feel\_ok = 1;  1358 }  1359  1360 cur\_state = 11;  1361 }  1362 }  1363  1364 void circulate\_confirm\_password(unsigned\* dest)  1365 *// cur\_state = 112*  1366 {  1367 char fir\_pw[input\_max\_length];  1368 char cfm\_pw[input\_max\_length];  1369  1370 print\_remind\_message(1);  1371  1372 while (1) {  1373  1374 input\_and\_check\_single(legal\_check\_pw, fir\_pw);  1375  1376 cur\_state = 1121; *// 有效密码，要求再输一次确认*  1377 print\_remind\_message(1);  1378  1379 get\_the\_line(cfm\_pw, 1);  1380 if (strcmp(fir\_pw, cfm\_pw) == 0) {  1381 break;  1382 }  1383  1384 cur\_state = 1120; *// 两次输入密码不匹配*  1385 print\_remind\_message(1);  1386 }  1387  1388 get\_MD5(fir\_pw, dest);  1389 }  1390  1391 #endif *//TRADE\_MANAGER\_SYSTEM\_TYPE\_IO\_H*  1392  1393  1394  1395 */\**  1396 *TYPE HEADER*  1397 *NAME MD5.h*  1398 *PATH ./lib/MD5.h*  1399 *\*/*  1400  1401 #ifndef TRADE\_MANAGER\_SYSTEM\_MD5\_H  1402 #define TRADE\_MANAGER\_SYSTEM\_MD5\_H  1403  1404 #include "basic\_def.h"  1405  1406 #define max\_length 128 + 5  1407  1408 char str[max\_length \* 2];  1409 char tmp[max\_length];  1410  1411 unsigned sex[max\_length];  1412  1413 unsigned s[64] = { 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, \  1414 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, \  1415 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, \  1416 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21} ;  1417  1418 unsigned K[64] = { 0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee, \  1419 0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501, \  1420 0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be, \  1421 0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821, \  1422 0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa, \  1423 0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8, \  1424 0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed, \  1425 0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a, \  1426 0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c, \  1427 0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70, \  1428 0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05, \  1429 0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665, \  1430 0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039, \  1431 0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1, \  1432 0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1, \  1433 0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391} ;  1434  1435 unsigned a0; *//A*  1436 unsigned b0; *//B*  1437 unsigned c0; *//C*  1438 unsigned d0; *//D*  1439  1440 unsigned M[16], con\_sex[100];  1441  1442 unsigned left\_rotate(unsigned val, unsigned n);  1443 void init\_value();  1444 void convert\_to\_sexa\_str(const char\* src, char\* dest);  1445 void convert\_to\_sexa\_num(const char\* src, unsigned \*dest);  1446 unsigned reverse\_sexa(unsigned x);  1447 void get\_MD5(const char\* inp, unsigned\* dest);  1448  1449 unsigned left\_rotate(unsigned val, unsigned n)  1450 {  1451 return (val >> (32 - n)) | (val << n);  1452 }  1453  1454 void init\_value()  1455 {  1456 for (char i = '0'; i <= '9'; i++) {  1457 con\_sex[(int)i] = (unsigned)i - 48;  1458 }  1459 con\_sex[(int)'A'] = 10;  1460 con\_sex[(int)'B'] = 11;  1461 con\_sex[(int)'C'] = 12;  1462 con\_sex[(int)'D'] = 13;  1463 con\_sex[(int)'E'] = 14;  1464 con\_sex[(int)'F'] = 15;  1465  1466 a0 = 0x67452301; *//A*  1467 b0 = 0xefcdab89; *//B*  1468 c0 = 0x98badcfe; *//C*  1469 d0 = 0x10325476; *//D*  1470  1471 memset(M, 0, sizeof(M));  1472 memset(sex, 0, sizeof(sex));  1473 str[0] = '\0';  1474 tmp[0] = '\0';  1475 }  1476  1477 void convert\_to\_sexa\_str(const char\* src, char\* dest)  1478 {  1479 dest[0] = '\0';  1480 int len = (int)strlen(src);  1481 for (int i = 0; i < len; i++) {  1482 sprintf(tmp, "%X", (unsigned)src[i]);  1483 strcat(dest, tmp);  1484 }  1485  1486 }  1487  1488 void convert\_to\_sexa\_num(const char\* src, unsigned \*dest)  1489 {  1490 int len = (int)strlen(src);  1491 int j = 0;  1492 for (int i = 0; i < len; i += 2) {  1493 dest[j] = con\_sex[(int)src[i]] \* 16 + con\_sex[(int)src[i + 1]];  1494 j++;  1495 }  1496 }  1497  1498 unsigned reverse\_sexa(unsigned x)  1499 {  1500 unsigned k = (1 << 24), y = 0;  1501 while (x) {  1502 y += (x % 256) \* k;  1503 x >>= 8;  1504 k >>= 8;  1505 }  1506 return y;  1507 }  1508  1509 void get\_MD5(const char\* inp, unsigned\* dest)  1510 {  1511 init\_value();  1512  1513 convert\_to\_sexa\_str(inp, str);  1514  1515 strcat(str, "8");  1516 while (strlen(str) % 128 != 112) {  1517 strcat(str, "0");  1518 }  1519  1520 sprintf(tmp, "%X", (int)strlen(inp)\*8);  1521 strcat(str, tmp);  1522 while (strlen(str) % 128 != 0) {  1523 strcat(str, "0");  1524 }  1525  1526 convert\_to\_sexa\_num(str, sex);  1527  1528 int len = (int)strlen(str) / 2;  1529 for (int t = 0; t < len; t += 64) {  1530  1531 int k = t;  1532 for (int j = 0; j <= 15; j++) {  1533 M[j] = sex[k + 3] \* (1 << 24) + sex[k + 2] \* (1 << 16) \  1534 + sex[k + 1] \* (1 << 8) + sex[k];  1535 k += 4;  1536 }  1537  1538 unsigned A = a0, B = b0, C = c0, D = d0;  1539 for (int i = 0; i < 64; i++) {  1540 unsigned F;  1541 int g;  1542 if (i <= 15) {  1543 F = (B & C) | ((~B) & D);  1544 g = i;  1545 }  1546 else if (i <= 31) {  1547 F = (D & B) | ((~D) & C);  1548 g = (5 \* i + 1) % 16;  1549 }  1550 else if (i <= 47) {  1551 F = B ^ C ^ D;  1552 g = (3 \* i + 5) % 16;  1553 }  1554 else {  1555 F = C ^ (B | (~D));  1556 g = (7 \* i) % 16;  1557 }  1558 F = F + A + K[i] + M[g];  1559 A = D;  1560 D = C;  1561 C = B;  1562 B = B + left\_rotate(F, s[i]);  1563 }  1564 a0 += A;  1565 b0 += B;  1566 c0 += C;  1567 d0 += D;  1568 }  1569  1570 dest[1] = reverse\_sexa(a0);  1571 dest[2] = reverse\_sexa(b0);  1572 dest[3] = reverse\_sexa(c0);  1573 dest[4] = reverse\_sexa(d0);  1574 }  1575  1576 #endif *//TRADE\_MANAGER\_SYSTEM\_MD5\_H*  1577  1578  1579  1580 */\**  1581 *TYPE HEADER*  1582 *NAME search.h*  1583 *PATH ./lib/search.h*  1584 *\*/*  1585  1586 #ifndef TRADE\_MANAGER\_SYSTEM\_SEARCH\_H  1587 #define TRADE\_MANAGER\_SYSTEM\_SEARCH\_H  1588  1589 #include "basic\_def.h"  1590 #include "functions.h"  1591  1592 int has\_wildcard; *// 是否含有通配符*  1593 int ord[max\_goods\_num];  1594 char inp[name\_max\_length];  1595 char low\_name[name\_max\_length]; *// 全改为小写的商品名*  1596 int inp\_fail[name\_max\_length]; *// fail数组*  1597 int d[name\_max\_length][name\_max\_length]; *// 递推数组*  1598 double simity[max\_goods\_num]; *// similarity 相似度*  1599  1600 int cmp\_by\_sim(const void\* a, const void\* b);  1601 void convert\_to\_lower(char\*dest, const char\* src);  1602 void get\_fail(char \*P, int \*f);  1603 int KMP(char \*T, char \*P, int \*f);  1604 int LCS(char\* a, char\* b);  1605 int wild\_match(char\* a, char\* b);  1606 int main\_search(char\* ori\_inp, int start\_ord, int kw\_num, int\* result);  1607  1608 int cmp\_by\_sim(const void\* a, const void\* b)  1609 {  1610 double delta = (simity[\*(int\*)b] - simity[\*(int\*)a]);  1611 return delta > 0 ? 1 : -1;  1612 }  1613  1614 void convert\_to\_lower(char\*dest, const char\* src)  1615 {  1616 int len = (int)strlen(src);  1617 for (int i = 0; i < len; i++) {  1618 dest[i] = isupper(src[i]) ? (char)(src[i] - 'A' + 'a') : src[i];  1619 }  1620 dest[len] = '\0';  1621 }  1622  1623 void get\_fail(char \*P, int \*f)  1624 {  1625 int m = (int)strlen(P);  1626 f[0] = 0, f[1] = 0;  1627 for (int i = 1; i < m - 1; i++) {  1628 int j = f[i];  1629 while (j && P[i] != P[j]) {  1630 j = f[j];  1631 }  1632 f[i + 1] = P[i] == P[j] ? j + 1 : 0;  1633 }  1634 }  1635  1636 int KMP(char \*T, char \*P, int \*f)  1637 {  1638 int n = (int)strlen(T), m = (int)strlen(P);  1639 if (n < m) {  1640 return 0;  1641 }  1642  1643 get\_fail(P, f);  1644  1645 int j = 0, res = 0;  1646 for (int i = 0; i < n; i++) {  1647 while (j && P[j] != T[i]) {  1648 j = f[j];  1649 }  1650 if (P[j] == T[i]) {  1651 j++;  1652 }  1653 if (j == m) {  1654 res++;  1655 }  1656 }  1657  1658 return res;  1659 }  1660  1661 int LCS(char\* a, char\* b)  1662 {  1663 int len\_a = (int)strlen(a), len\_b = (int)strlen(b);  1664  1665 memset(d, 0, sizeof(d));  1666 if (a[0] == b[0]) {  1667 d[0][0] = 1;  1668 }  1669  1670 if (len\_a > 1 && a[1] == b[0]) {  1671 d[1][0] = 1;  1672 }  1673 if (len\_b > 1 && a[0] == b[1]) {  1674 d[0][1] = 1;  1675 }  1676  1677 for (int i = 1; i < len\_a; i++) {  1678 for (int j = 1; j < len\_b; j++) {  1679 d[i][j] = get\_max(d[i - 1][j], d[i][j - 1]);  1680 if (a[i] == b[j]) {  1681 d[i][j] = get\_max(d[i][j], d[i - 1][j - 1] + 1);  1682 }  1683 }  1684 }  1685  1686 return d[len\_a - 1][len\_b - 1];  1687 }  1688  1689 int wild\_match(char\* a, char\* b) { *// a为含有通配符\* ?的串*  1690 int len\_a = (int)strlen(a), len\_b = (int)strlen(b);  1691  1692 for (int i = 0; i <= len\_b; i++)  1693 d[0][i] = 1;  1694 for (int i = 0; a[i] == '\*'; i++)  1695 d[i + 1][0] = 1;  1696  1697 for (int i = 0; i < len\_a; i++)  1698 for (int j = 0; j < len\_b; j++)  1699 if (a[i] == '\*')  1700 d[i + 1][j + 1] = d[i][j + 1] | d[i][j] | d[i + 1][j];  1701 else if (a[i] == '?')  1702 d[i + 1][j + 1] = d[i][j];  1703 else  1704 d[i + 1][j + 1] = d[i][j] & (a[i] == b[j]);  1705  1706 for (int i = 1; i <= len\_b; i++) {  1707 if (d[len\_a][i])  1708 return 1;  1709 }  1710  1711 return 0;  1712 }  1713  1714 int main\_search(char\* ori\_inp, int start\_ord, int kw\_num, int\* result)  1715 {  1716 int len\_inp = (int)strlen(ori\_inp);  1717  1718 convert\_to\_lower(inp, ori\_inp);  1719  1720 has\_wildcard = 0;  1721 for (int i = 0; i < len\_inp; i++) {  1722 if (inp[i] == '\*' || inp[i] == '?') {  1723 has\_wildcard = 1;  1724 break;  1725 }  1726 }  1727  1728 if (has\_wildcard == 0) {  1729 get\_fail(inp, inp\_fail);  1730 }  1731  1732 for (int i = 1; i <= total\_goods\_cnt; i++) {  1733  1734 if (start\_ord == 1 && i == kw\_num) {  1735 continue;  1736 }  1737 if (goods[i].deleted == 1) {  1738 continue;  1739 }  1740  1741 convert\_to\_lower(low\_name, goods[i].name);  1742 int len\_str = (int)strlen(low\_name);  1743 simity[i] = 0;  1744 if (has\_wildcard == 0) { *// 不含有通配符*  1745 int kmp\_res = KMP(low\_name, inp, inp\_fail);  1746 if (kmp\_res == 0) {  1747 int match\_lcs = LCS(inp, low\_name);  1748 simity[i] = match\_lcs \* 0.8 / len\_inp;  1749 }  1750 else {  1751 simity[i] = 0.9;  1752 simity[i] += ((double)kmp\_res / 100);  1753 simity[i] -= (double)(len\_str - len\_inp) / (len\_str \* 500);  1754 simity[i] = (simity[i] > 1.0 ? 1.0 : simity[i]);  1755 }  1756 }  1757 else { *// 含有通配符*  1758 simity[i] = wild\_match(inp, low\_name);  1759 }  1760  1761 if (goods[i].stock\_cnt <= 0 && simity[i] >= 0.3) {  1762 simity[i] = 0.3;  1763 }  1764 }  1765  1766 for (int i = 1; i <= total\_goods\_cnt; i++) {  1767 ord[i] = i;  1768 }  1769  1770 qsort(ord + 1, (unsigned)total\_goods\_cnt, sizeof(int), cmp\_by\_sim);  1771  1772 int j = start\_ord + 1;  1773 int res\_cnt = 0;  1774  1775 for (int i = 1; i <= total\_goods\_cnt; i++, j++) {  1776  1777 int cur = ord[i];  1778 if (simity[cur] < 0.25) { *// 若相似度过低，则忽略*  1779 break;  1780 }  1781  1782 print\_search\_item(j, cur);  1783 res\_cnt++;  1784 }  1785  1786 for (int i = 1; i <= res\_cnt; i++) {  1787 result[i] = ord[i];  1788 }  1789  1790 return res\_cnt;  1791 }  1792  1793 #endif *//TRADE\_MANAGER\_SYSTEM\_SEARCH\_H*  1794  1795  1796  1797 */\**  1798 *TYPE HEADER*  1799 *NAME initialize.h*  1800 *PATH ./lib/initialize.h*  1801 *\*/*  1802  1803 #ifndef TRADE\_MANAGER\_SYSTEM\_INITIALIZE\_H  1804 #define TRADE\_MANAGER\_SYSTEM\_INITIALIZE\_H  1805  1806 #include "basic\_def.h"  1807 #include "basic\_io.h"  1808 #include "file\_op.h"  1809 #include "link\_op.h"  1810  1811 int read\_goods\_id[max\_goods\_num];  1812 int read\_record\_id[max\_record\_num];  1813 int read\_ok;  1814  1815 void initialize();  1816 int initialize\_normal();  1817 void initialize\_file();  1818 void initialize\_version();  1819 void recover\_all();  1820  1821 void initialize()  1822 {  1823 initialize\_version();  1824  1825 int load\_ok = initialize\_normal();  1826 if (load\_ok == 0) {  1827 print\_error(20);  1828 initialize\_file();  1829 recover\_all();  1830 }  1831  1832 reset\_modify\_signs();  1833  1834 printf("Successfully initialized!\n\n");  1835  1836 wait\_continue();  1837 }  1838  1839 void initialize\_version()  1840 {  1841 strcpy(current\_version.version\_number, "1.0.0");  1842 strcpy(current\_version.released\_time, "2018.11.11");  1843 }  1844  1845 int initialize\_normal()  1846 {  1847 printf("\n-- Welcome to Armeria Trade Manager System --\n");  1848 printf("\n>> Made by Wang Hankun, HIT\n\n");  1849  1850 printf("Loading Save files...\n");  1851  1852 cur\_state = 0;  1853  1854 *// 检查文件可读性*  1855 read\_ok = 1;  1856 read\_ok &= check\_file\_write(path\_basic);  1857 read\_ok &= check\_file\_write(path\_users);  1858 read\_ok &= check\_file\_write(path\_goods);  1859 read\_ok &= check\_file\_write(path\_records);  1860 read\_ok &= check\_file\_write(path\_links);  1861 EXIT\_CHECK(read\_ok);  1862  1863 *// 读取存档*  1864 file\_basic = fopen(path\_basic, "r");  1865 read\_ok = (fscanf(file\_basic, "%d%d%d",  1866 &total\_users\_cnt, &total\_goods\_cnt, &total\_records\_cnt) == 3);  1867 EXIT\_CHECK(read\_ok);  1868  1869 file\_users = fopen(path\_users, "r");  1870 read\_ok = (fread(user + 1, sizeof(User),  1871 (unsigned)total\_users\_cnt, file\_users) == total\_users\_cnt);  1872 EXIT\_CHECK(read\_ok);  1873  1874 file\_goods = fopen(path\_goods, "r");  1875 read\_ok = (fread(goods + 1, sizeof(Goods),  1876 (unsigned)total\_goods\_cnt, file\_goods) == total\_goods\_cnt);  1877 EXIT\_CHECK(read\_ok);  1878  1879 file\_records = fopen(path\_records, "r");  1880 read\_ok = (fread(record + 1, sizeof(Record),  1881 (unsigned)total\_records\_cnt, file\_records) == total\_records\_cnt);  1882 EXIT\_CHECK(read\_ok);  1883  1884 file\_links = fopen(path\_links, "r");  1885 for (int i = 1; i <= total\_users\_cnt; i++) {  1886 read\_ok = table\_reconstruct(file\_links,  1887 &user[i].record\_head, &user[i].record\_tail, user[i].record\_cnt);  1888 EXIT\_CHECK(read\_ok);  1889  1890 if (user[i].is\_seller == 1) {  1891 read\_ok = table\_reconstruct(file\_links,  1892 &user[i].sell.goods\_head, &user[i].sell.goods\_tail, user[i].sell.goods\_cnt);  1893 EXIT\_CHECK(read\_ok);  1894  1895 read\_ok = table\_reconstruct(file\_links,  1896 &user[i].sell.record\_head, &user[i].sell.record\_tail, user[i].sell.record\_cnt);  1897 EXIT\_CHECK(read\_ok);  1898 }  1899 }  1900  1901 printf("Successfully loaded savings!\n");  1902  1903 return 1;  1904 }  1905  1906 void initialize\_file()  1907 {  1908 file\_basic = fopen(path\_basic, "w");  1909 file\_users = fopen(path\_users, "w");  1910 file\_goods = fopen(path\_goods, "w");  1911 file\_records = fopen(path\_records, "w");  1912 file\_links = fopen(path\_links, "w");  1913  1914 fprintf(file\_basic, "%d %d %d\n", 0, 0, 0);  1915 fflush(file\_basic);  1916 }  1917  1918 void recover\_all()  1919 {  1920 total\_users\_cnt = 0;  1921 total\_goods\_cnt = 0;  1922 total\_records\_cnt = 0;  1923 }  1924  1925 #endif *//TRADE\_MANAGER\_SYSTEM\_INITIALIZE\_H*  1926  1927  1928  1929 */\**  1930 *TYPE HEADER*  1931 *NAME wel\_log\_reg.h*  1932 *PATH ./lib/wel\_log\_reg.h*  1933 *\*/*  1934  1935 #ifndef TRADE\_MANAGER\_SYSTEM\_WEL\_LOG\_REG\_H  1936 #define TRADE\_MANAGER\_SYSTEM\_WEL\_LOG\_REG\_H  1937  1938 #include "basic\_def.h"  1939 #include "basic\_io.h"  1940 #include "type\_io.h"  1941 #include "checker.h"  1942 #include "info\_op.h"  1943 #include "file\_op.h"  1944  1945 char reg\_user\_name[name\_max\_length];  1946  1947 void show\_welcome();  1948 void show\_register();  1949 void show\_login();  1950  1951 void show\_welcome()  1952 *// cur\_state = 0*  1953 {  1954 print\_remind\_message(1);  1955  1956 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, 2);  1957  1958 switch (choice) {  1959  1960 case 0:  1961 cur\_state = -1;  1962 return;  1963  1964 case 1:  1965 cur\_state = 1;  1966 break;  1967  1968 case 2:  1969 cur\_state = 2;  1970 break;  1971  1972 default:  1973 break;  1974 }  1975 }  1976  1977 void show\_register()  1978 *// cur\_state = 1*  1979 {  1980 if (total\_users\_cnt >= max\_user\_num) {  1981 print\_error(10);  1982 cur\_state = 1;  1983 return;  1984 }  1985  1986 cur\_state = 11;  1987 circulate\_confirm\_name(reg\_user\_name);  1988  1989 unsigned reg\_user\_pw\_MD5[5];  1990 cur\_state = 112;  1991 circulate\_confirm\_password(reg\_user\_pw\_MD5);  1992  1993 cur\_state = 11201;  1994 add\_user(reg\_user\_name, reg\_user\_pw\_MD5);  1995 print\_remind\_message(0);  1996  1997 cur\_state = 0;  1998 }  1999  2000 void show\_login()  2001 *// cur\_state = 2*  2002 {  2003 cur\_state = 21;  2004 print\_remind\_message(1);  2005 id\_num = input\_and\_check\_num(legal\_check\_return\_num, str, 0, total\_users\_cnt);  2006  2007 if (id\_num == 0) {  2008 cur\_state = 0;  2009 return;  2010 }  2011  2012 cur\_state = 22;  2013 int rest\_try = 3; *// 剩余密码尝试次数*  2014 char inp\_pw[input\_max\_length];  2015 unsigned inp\_pw\_MD5[MD5\_length];  2016  2017 while (rest\_try--) {  2018 print\_remind\_message(1);  2019 get\_the\_line(inp\_pw, 1);  2020 get\_MD5(inp\_pw, inp\_pw\_MD5);  2021  2022 if (check\_MD5(inp\_pw\_MD5, user[id\_num].encrypted)) {  2023 cur\_state = 221;  2024 print\_remind\_message(0);  2025 break;  2026 }  2027 else {  2028 cur\_state = 220 \* 10 + rest\_try;  2029 if (rest\_try) {  2030 print\_remind\_message(1);  2031 }  2032 else {  2033 print\_remind\_message(0);  2034 cur\_state = 0;  2035 return;  2036 }  2037 }  2038 }  2039 printf("\nWelcome, %s!\n", user[id\_num].name);  2040 wait\_continue();  2041  2042 cur\_state = 3;  2043 }  2044  2045 #endif *//TRADE\_MANAGER\_SYSTEM\_WEL\_LOG\_REG\_H*  2046  2047  2048  2049 */\**  2050 *TYPE HEADER*  2051 *NAME surface.h*  2052 *PATH ./lib/surface.h*  2053 *\*/*  2054  2055 #ifndef TRADE\_MANAGER\_SYSTEM\_SHOW\_MAIN\_SURFACE\_H  2056 #define TRADE\_MANAGER\_SYSTEM\_SHOW\_MAIN\_SURFACE\_H  2057  2058 #include "basic\_def.h"  2059 #include "checker.h"  2060 #include "basic\_io.h"  2061 #include "type\_io.h"  2062 #include "info\_op.h"  2063 #include "search.h"  2064 #include "functions.h"  2065  2066 char tmp\_name[name\_max\_length];  2067 int tmp\_storage;  2068 int tmp\_price;  2069 int id;  2070 int mod\_qnt; *// 修改量*  2071 int upper; *// 修改量上界*  2072 int lower; *// 修改量下节*  2073  2074 char kw[name\_max\_length]; *// kw为keyword缩写*  2075 int kw\_num;  2076 int result[max\_goods\_num]; *// 搜索结果编号*  2077 int buy\_qnt;  2078 int result\_cnt;  2079 int start\_ord; *// start\_ord = 1 时，表示存在goodsID与用户输入符合的商品*  2080 int chose\_id;  2081  2082 int op\_ok;  2083  2084 void show\_main\_menu();  2085 void show\_personal\_menu();  2086 void show\_seller\_menu();  2087 void show\_shopping\_menu();  2088  2089 void show\_main\_menu()  2090 *// cur\_state = 3*  2091 {  2092 print\_remind\_message(1);  2093  2094 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, 3);  2095  2096 switch (choice) {  2097  2098 case 0:  2099 cur\_state = 0;  2100 id\_num = 0;  2101 return;  2102  2103 case 1:  2104 cur\_state = 31;  2105 break;  2106  2107 case 2:  2108 cur\_state = 32;  2109 break;  2110  2111 case 3:  2112 cur\_state = 33;  2113 break;  2114  2115 default:  2116 break;  2117 }  2118 }  2119  2120 void show\_personal\_menu()  2121 *// cur\_state = 31*  2122 {  2123 print\_remind\_message(1);  2124  2125 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, 2);  2126  2127 switch (choice) {  2128  2129 *// 返回上级菜单*  2130 case 0:  2131 cur\_state = 3;  2132 return;  2133  2134 *// 显示购买记录*  2135 case 1:  2136 print\_buy\_records\_list();  2137 wait\_continue();  2138 break;  2139  2140 *// 注册商家用户，或进入商家菜单*  2141 case 2:  2142 if (user[id\_num].is\_seller == 0) {  2143 register\_seller();  2144 cur\_state = 3121;  2145 print\_remind\_message(0);  2146 break;  2147 }  2148 else {  2149 cur\_state = 312;  2150 return;  2151 }  2152  2153 default:  2154 break;  2155 }  2156  2157 cur\_state = 31;  2158 }  2159  2160 void show\_seller\_menu()  2161 *// cur\_state = 312*  2162 {  2163 print\_remind\_message(1);  2164  2165 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, 5);  2166  2167 switch (choice) {  2168  2169 *// 返回上级菜单*  2170 case 0:  2171 cur\_state = 31;  2172 return;  2173  2174 *// 显示自己的商品列表*  2175 case 1:  2176 print\_my\_goods\_list();  2177 wait\_continue();  2178 break;  2179  2180 *// 显示售卖记录*  2181 case 2:  2182 print\_sell\_records\_list();  2183 wait\_continue();  2184 break;  2185  2186 *// 增加商品*  2187 case 3:  2188 if (total\_goods\_cnt >= max\_goods\_num) {  2189 print\_error(10);  2190 break;  2191 }  2192  2193 cur\_state = 31231;  2194 print\_remind\_message(1);  2195 input\_and\_check\_single(legal\_check\_name, input\_str);  2196 strncpy(tmp\_name, input\_str, name\_max\_length);  2197  2198 cur\_state = 31232;  2199 print\_remind\_message(1);  2200 tmp\_storage = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, max\_storage);  2201  2202 cur\_state = 31233;  2203 print\_remind\_message(1);  2204 tmp\_price = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, max\_price);  2205  2206 op\_ok = add\_goods(tmp\_name, tmp\_storage, tmp\_price);  2207 BREAK\_CHECK(op\_ok);  2208  2209 cur\_state = 31234;  2210 print\_remind\_message(0);  2211  2212 break;  2213  2214 *// 删除商品*  2215 case 4:  2216 print\_my\_goods\_list();  2217  2218 cur\_state = 31241;  2219 print\_remind\_message(1);  2220 id = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 1, total\_goods\_cnt);  2221 del\_goods(id);  2222  2223 cur\_state = 31242;  2224 print\_remind\_message(0);  2225  2226 break;  2227  2228 *// 更改库存量*  2229 case 5:  2230 print\_my\_goods\_list();  2231  2232 cur\_state = 31251;  2233 print\_remind\_message(1);  2234 id = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 1, total\_goods\_cnt);  2235 if (goods[id].seller\_num != id\_num || goods[id].deleted == 1) {  2236 print\_error(11);  2237 break;  2238 }  2239  2240 cur\_state = 31252;  2241 print\_remind\_message(1);  2242 upper = max\_storage - goods[id].stock\_cnt;  2243 lower = -goods[id].stock\_cnt;  2244 mod\_qnt = input\_and\_check\_num(legal\_check\_return\_num, input\_str, lower, upper);  2245  2246 mod\_goods\_stock(id, mod\_qnt);  2247  2248 cur\_state = 31253;  2249 print\_remind\_message(0);  2250  2251 break;  2252  2253 default:  2254 break;  2255 }  2256 cur\_state = 312;  2257 }  2258  2259 void show\_shopping\_menu()  2260 *// cur\_state = 32*  2261 {  2262 print\_remind\_message(1);  2263  2264 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, 5);  2265  2266 switch (choice) {  2267  2268 *// 返回上级菜单*  2269 case 0:  2270 cur\_state = 3;  2271 return;  2272  2273 *// 搜索*  2274 case 1:  2275 cur\_state = 321;  2276 print\_remind\_message(1);  2277  2278 start\_ord = 0;  2279 input\_and\_check\_single(legal\_check\_keyword, input\_str);  2280 strncpy(kw, input\_str, name\_max\_length);  2281  2282 system("clear");  2283 printf(" --------------------------------------------------------------------------------------- \n");  2284 printf("| order | goodsID | goods name | price | storage | userID(seller) |\n");  2285  2286 if (is\_number(kw)) {  2287 kw\_num = atoi(kw);  2288 if (kw\_num > 0 && kw\_num <= total\_goods\_cnt) {  2289 print\_search\_item(1, kw\_num);  2290 start\_ord = 1;  2291 }  2292 }  2293  2294 memset(result, 0, sizeof(result));  2295 result\_cnt = main\_search(kw, start\_ord, kw\_num, result);  2296  2297 printf(" --------------------------------------------------------------------------------------- \n\n");  2298  2299 result\_cnt += start\_ord;  2300 if (result\_cnt == 0) {  2301 printf("No results.\n");  2302 wait\_continue();  2303 break;  2304 }  2305  2306 *// 读取用户选择的商品序号*  2307 cur\_state = 3211;  2308 print\_remind\_message(1);  2309 choice = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, result\_cnt);  2310 BREAK\_CHECK(choice);  2311  2312 if (start\_ord == 1 && choice == 1) {  2313 chose\_id = kw\_num;  2314 }  2315 else {  2316 chose\_id = ord[choice - start\_ord];  2317 }  2318  2319 *// 读取用户购买量*  2320 cur\_state = 3212;  2321 print\_remind\_message(1);  2322 buy\_qnt = input\_and\_check\_num(legal\_check\_return\_num, input\_str, 0, goods[chose\_id].stock\_cnt);  2323 BREAK\_CHECK(buy\_qnt);  2324  2325 op\_ok = buy\_goods(chose\_id, buy\_qnt);  2326 BREAK\_CHECK(op\_ok);  2327  2328 cur\_state = 3213;  2329 print\_remind\_message(0);  2330 wait\_continue();  2331  2332 default:  2333 break;  2334 }  2335  2336 cur\_state = 32;  2337 }  2338  2339 #endif *//TRADE\_MANAGER\_SYSTEM\_SHOW\_MAIN\_SURFACE\_H*  2340 | | | | | | | |
| **分析总结、收获和体会:** | | | | | | | |
| **优点：**  数据结构设计合理，静态存储结构与动态存储结构搭配较好。  使用了分级菜单，界面清晰，逻辑性强。  搜索接口简洁，无需让用户思考应该用商品名搜索还是编号搜索，也无需思考应该用准确搜索、模糊搜索还是通配符搜索，程序会自动判断用户需求并生成搜索结果。  使用了MD5算法加密用户信息，提高了数据安全性。  **创新之处：**  使用状态码cur\_state保存当前状态信息（表示用户进入到了哪一级菜单、在进行哪一步操作），可以高效地调用相关功能函数。  将输出较多、面向用户较多的函数与偏向底层构建的函数分开，并按照功能放在不同的头文件中，使得项目逻辑清晰。  **不足之处与需要改进的地方：**  - 文件读写较为低效。只要有一条用户信息被修改，就得重新存档整个users.info，记录信息、商品信息也是这样。若能将每条信息都各自保存在一个单独的文件中，就可较好地改善这个问题。  - 功能尚不完全。虽然主体框架已完成，但部分功能细节还需要完善。  **收获与体会**：  这次的实验大作业是我第一次独立完成两千行级别的迷你型工程。中学参加算法竞赛时写的程序，大多较短，解决的问题较为单一，且对交互设计、功能模块化与代码规范的要求不高：因为算法竞赛对程序的评价标准往往仅为正确性和运行效率，而并不关心程序内部的实现方式与细节。这与本次实验大作业的评价标准存在较大差异。为了满足本次实验大作业的要求，我对自己的代码风格进行了一定的调整，在确保程序正确性与性能的同时兼顾了程序的可读性。  我还熟悉了不少以前较为生疏的语言特性，尤其是对指针与链表的运用。在本次实验中，我大量运用了整型指针、字符型指针、结构体指针和函数指针，大大提高了程序的灵活性和泛化能力。  此外，我的系统设计能力与大局观也得到了较好的锻炼。虽然实现的只是一个迷你系统，但麻雀虽小，五脏俱全。这为我今后从事大型软件系统的框架设计和开发积累了宝贵经验。 | | | | | | | |
| **自我评价：** | | | | | 是 | | 否 |
| 程序是否能通过编译并正常运行，没有bug？ | | | | | √ | |  |
| 是否在撰写报告之前观看了spoc里的代码风格视频？ | | | | | √ | |  |
| 程序代码是否符合代码规范(模块化，变量和函数命名规范，对齐与缩进，有必要的注释)？ | | | | | √ | |  |
| 是否按模块化要求进行了程序设计，系统功能是否完善？ | | | | | √ | |  |
| 是否是独立完成，未参考其他人的设计或代码？ | | | | | √ | |  |
| **自我评语：**  实验成品符合预期，收获颇丰，反思总结深入。我认为自己较为圆满地完成了本次实验大作业。  **报告完成日期：2018年12月27日** | | | | | | | |