电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2016220201024

姓 名 王籽涵

（实验） 课程名称 超市商品信息管理系统

理论教师 张翔

实验教师 张翔

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：王籽涵 学号：2016220201024 指导教师：张翔**

**实验地点：基础实验楼 实验时间：2016.12.28**

**一、 实验名称：超市商品信息管理系统**

**二、 实验学时：8学时**

**三、 实验目的：**

1) 掌握结构体、指针数组、指针、字符串、函数等概念和使用方法，能够熟练的操作。

2) 养成应用型程序必有交互界面设计的思维，并形成自己的易于大众使用的交互风格。

3) 深刻贯彻“分而治之”的思想，让程序尽可能的模块化、简介化、高效化。

4）深思熟虑相应的边界条件，增加程序的健壮性。

5）考虑并完成程序的高效复用问题。

**四、 实验原理：**

1. 结构体：结构的元素可能具有不同的类型。而且每个结构成员都有名字，所以**为了选择特定的结构成员需要指明结构成员的名字而不是它的位置。**
2. 结构变量的声明：

struct{

int number;

char name[NAME\_LEN+1];

int on\_hand;

}part1,part2;

**结构的成员在内存中是按照声明的顺序存储的**。

1. 结构变量的初始化：

struct{

int number;

char name[NAME\_LEN+1];

int on\_hand;

}part1 = {528,”Disk drive”,10},

part2 = {914,”Printer cable”,5};

初始化式中的值必须按照结构成员的顺序进行显示。

1. 对结构的操作：了访问结构内的成员，**首先写出结构的名字，然后写一个据点，再写出成员的名字**。
2. **结构标记**：结构标记是用于标识某种特定结构的名字。下面的例子声明了名为part的结构标记：

struct part{

int number;

char name[NAME\_LEN+1];

int on\_hand;

}；

一旦创建了标记part，就可以用它来声明变量了：

struct part part1，part2；

**但是，不能通过漏掉单词struct来缩写这个声明：**

1. **结构类型的定义**：除了声明结构标记，还可以用typedef来定义真是的类型名。例如，可以按照如下方式定义名为Part的类型：

typedef struct {

int number;

char name[NAME\_LEN+1];

int on\_hand;

}Part；

**注意，类型Part的名字必须出现在定义的末尾，而不是在单词struct的后边。**

**五、 实验内容：**

用C语言实现一个小型的超市商品管理系统，该系统需要具备商品信息录入、商品信息修改、商品信息删除、商品信息查找、商品信息的插入这几个功能。具体实现步骤如下：

1. 软件界面控制：实现一个数字选项式的启动界面，其中包含、商品信息修改、商品信息删除、商品信息查找、退出系统并保存5个选项。
2. 商品信息的初始化：定义一个商品信息的结构体，并且声明一个指针数组，数组中的内容为指向结构体的指针。实现一个函数，从已有的商品信息文件中读入商品信息，并且分配内存保存至指针数组中。
3. 商品信息的修改：实现一个函数完成商品信息的修改功能，实现可以根据商品的名称修改商品信息。其中用字符串比较的方式来查找待修改商品。
4. 商品信息的删除：实现一个函数，实现根据商品的名称来删除对应的商品信息的功能，商品查找通过字符串比较的方式，查找到后释放对应指针指向的内存区域，完成删除。
5. 商品信息的查找：实现一个函数，函数的功能是根据输入的商品名称来查找对应的商品信息，商品名称的判断用字符串比较的方式来实现，然后调用格式化输出子函数显示查找到的商品信息。
6. 退出系统，并保存：实现一个文件写入函数将所有信息的改动写入到商品信息文件，然后清理系统运行过程中已分配的内存。

**六、 实验器材（设备、元器件）：**

Macbook Pro 16

**七、 实验步骤：**

**一、实现基本功能与创新。**

该实验要求程序实现**7基本种功能：**

1、商品信息的初始化；

2、商品信息的修改；

3、商品信息的删除；

4、商品信息的查找；

5、商品信息的插入；

6、查看所有商品信息；

7、退出系统，并保存。

本人在此基础上更新了如下功能：

**1.基于各种特征的升序的商品信息排列；**

**2.可通过不同特征的搜索找到相应的商品；**

**3.可修改某一商品的部分或全部信息；**

**4.更加人性化的清屏功能（windows）；**

**5.交互背景修改为白底绿字（windows），减轻用户的审美疲劳；**

**6.更加人性化的输入提示；**

**7.增加了商品信息格式化功能；**

**8.基于操作的自动存储功能，让用户不在手动保存。**

**二、对于全局和函数的构思**

**1.覆盖全局的宏与结构指针数组：**

#define NAME\_LEN 50

#define MAX\_ITEMS 200

#define STRUCT\_LEN sizeof(struct info\_items)

//宏定义了相应名字长度和可以储存的最大商品数量以及为要使用到的动态分配做准备

struct info\_items{ //Use struct to define the items

int item\_id;

char item\_name[NAME\_LEN];

char item\_cate[NAME\_LEN];

float item\_price;

int item\_stock;

};

struct info\_items \*items[MAX\_ITEMS];

//结构指针数组，数组元素为指向 储存某一商品的结构 的地址

int num\_items = 0;//全局变量用于记录数组中有记录的数量

**2.函数接口：**

1.从文件初始化所有商品信息需要的函数：

**void info\_init();**  该函数通过循环使用readline函数来初始化所有商品信息。

**GoodInfo\* read\_line(FILE\* fp) 用来读取一行（一个）商品信息。**

void main\_menu (void); //主界面函数

void edit (void); //信息修改操作

void delete (void); //信息删除操作

void insert (void); //信息增加操作

void sort (void); //信息排序操作

void print (void); //信息打印操作

void format (void); //信息格式化操作

void search (void); //信息检索操作

struct info\_items\* read\_line (FILE\* fp);

//文件读行操作

三、**编写主函数：**

**采用了for的死循环形式，且内部用switch判断所进行的操作，知道用户用指定指令推出系统，并且相应的操作字符如下**：

1.检索;

2.修改;

3.删除;

4.插入;

5.排序;

6.格式化;

7.打印;

9.退出系统;

int main (void) //The main body of the system

{

system("color F2");

char operation,confirm,start\_code;

S: printf("System will be started based on file Items\_Information.txt \n\n");

printf("START or QUIT?<S/Q>: ");

start\_code = getchar();

while(getchar() != '\n');

switch (start\_code) {

case 's':

case 'S':

printf("Start successfully!!!\n\nPress any key to coutinue.");

getchar();

break;

case 'q':

case 'Q':

printf("Confirm to QUIT<Y/N>: ");

confirm = getchar();

switch (confirm) {

case 'y':

case 'Y':

printf("\nYou quited the system successfully!~\n");

return 0;

default:

while(getchar() != '\n');

goto S;

}

default:

printf("Wrong code!!!");

goto S;

break;

}

for ( ; ; ){

main\_menu();

printf("Enter operation code: ");

operation = getchar();

while (getchar() != '\n')

;

switch (operation){

case '1': search();

break;

case '2': edit();

break;

case '3': delete();

break;

case '4': insert();

break;

case '5': sort();

break;

case '6': format();

break;

case '7': print();

break;

case '9':

printf("Confirm to QUIT<Y/N>: ");

if ((confirm = getchar()) == 'y' || confirm == 'Y' ){

printf("\nYou quited the system successfully!~\n");

return 0;

}

else if (confirm == 'n' || confirm == 'N'){

getchar();

break;

}

default:

printf("Wrong operation code!!!");

break;

}

}

return 0;

}

由于本系统的使用前提是必须要有相应文件才能打开（本人构思后觉得加入创建文件功能会使系统变得冗杂且用户不易操作），所以一启动便会要求用户是否基于文件使用此系统，若同意，则继续，若不同意则退出。

进入系统后相应的操作就基于提示和用户输入，比较简洁易懂。

**四、编写子函数：**

高板块将于实验分析中详细解释。

**五、调试程序，检查功能是否实现。**

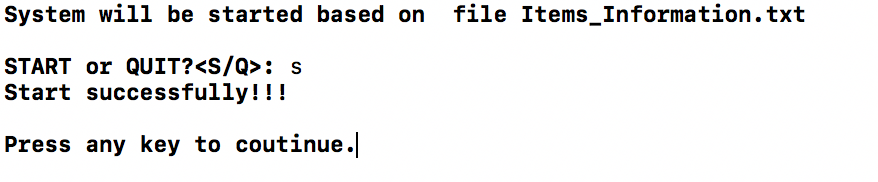
**六、反思总结及边界条件完善。**

**八、 实验结果与分析（含重要数据结果分析或核心代码流程分析）**

**1.写在分析之前。**

基于本人的构思，本实验代码可能会有一些难以理解，但我会尽量讲述的更加详细。

由于私认为创建新文件会使操作冗杂，所以在此版本代码中我略去了创建文件的步骤，直接将整个系统基于一个可读写的文本文件，这样应该能更方便用户的操作。



由于各个函数的打开文件和写入文件有不同差异，所以我将原本要写成单独函数的打开和写入文件没有单独的做成函数，而是在每个功能函数中重复出现，可能会有一些小复杂，但总的来说应该还是能够理解。

由于此程序是在Mac上写的，所以其中的格式在windows上不是很对称，我在写此报告时将它调成了mac上能够对其的格式，但在windows上需要再次调整。

由于检索功能和编辑功能有部分重复，所以在一个版本中我将它们写成了一个函数，但在此版本中不会出现。

由于函数路径是基于我的私人电脑的，所以届时老师想仔细检查，请将宏定义中的F\_PATH（文件路径）修改一下即可。

**由于要实现每个操作后都要自动存储，所以我将每个函数的基本形式都写为**

【1】打开文件—读取文件—检查文件—功能操作—写入文件—关闭文件

所以每个函数都将文件打开了一遍，只是有些没有写入。

目前正在考虑更高效的复用，但由于还在进行其他紧张的学业复习，所以暂时没有进行具体实现。

**2.读行函数与写行函数**

struct info\_items\* read\_line (FILE\* fp)

{

struct info\_items\* items = malloc(STRUCT\_LEN);

fscanf(fp, "%d", &items->item\_id);

fscanf(fp, "%s", items->item\_name);

fscanf(fp, "%s", items->item\_cate);

fscanf(fp, "%f", &items->item\_price);

fscanf(fp, "%d\n", &items->item\_stock);

return items;

};

从文件中读取一行的信息。

void write\_line(int i)

{

printf("%-10d\t",items[i]->item\_id);

printf("%-15s\t",items[i]->item\_name);

printf("%-15s\t",items[i]->item\_cate);

printf("%-10.2f\t",items[i]->item\_price);

printf("%-10d\n",items[i]->item\_stock);

return;

}

将一行商品信息打印至界面。

**3.主界面函数**

void main\_menu (void)

{

printf("Welcome to the supermarket commodity information management system\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* \*\n");

printf("\* 1 - Search the item \*\n");

printf("\* \*\n");

printf("\* 2 - Edit the item \*\n");

printf("\* \*\n");

printf("\* 3 - Delete the item \*\n");

printf("\* \*\n");

printf("\* 4 - Insert the item \*\n");

printf("\* \*\n");

printf("\* 5 - Sort the items \*\n");

printf("\* \*\n");

printf("\* 6 - Format the items \*\n");

printf("\* \*\n");

printf("\* 7 - Print the items' table \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

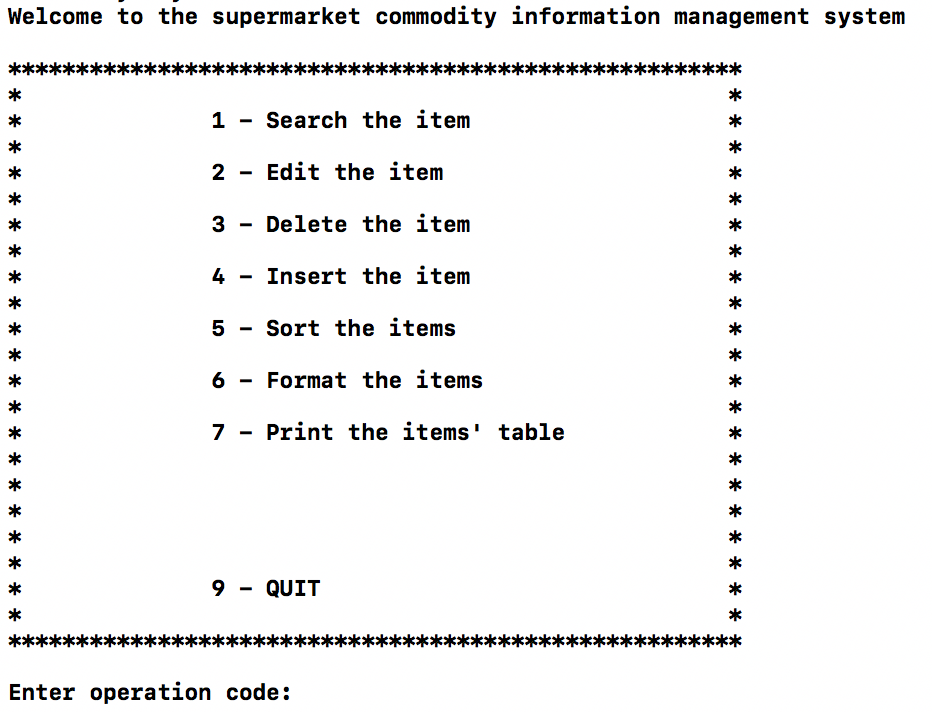
printf("\* 9 - QUIT \*\n");

printf("\* \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

}

即主界面，由于其只相当于视图，所以并未将它包含在主函数内。



**4.检索函数**

void search (void)

{

int id;

char name[NAME\_LEN];

char search\_way;

int i,j = 0;

char tem[999];

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

if ((fp = fopen(F\_PATH,"r")) == NULL){

fprintf(stderr,"Can't open it, please check it carefully!!!");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

num\_items = j;

if (j == 0){

printf("Nothing in the file!");

return;

}

fclose(fp);

SERCH:printf("Use ID/NAME to search the item <I/N>: ");

search\_way = getchar();

switch (search\_way) {

case 'i':

case 'I':

while(getchar() != '\n');

printf("\nPlease enter the ID of item: \n");

scanf("%d", &id);

for(i = 0; i < j; i++){

if (items[i]->item\_id == id){

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

break;

}

}

if (i == j){

printf("Didn't find the item, please check the ID!");

}

break;

case 'n':

case 'N':

while(getchar() != '\n');

printf("\nPlease enter the NAME of item: ");

scanf("%s",name);

for(i = 0; i < j; i++){

if (strcmp(items[i]->item\_name, name) == 0){

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

break;

}

}

if (i ==j){

printf("Didn't find the item, please check the ID!");

}

break;

default:

printf("\nIllegle instruction code!\nPlease check carefully!!\n");

while(getchar() != '\n');

goto SERCH;

}

while(getchar() != '\n');

printf("\nPress any key to contimue.\n");

getchar();

system ("cls");

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

}

return;

}

在基于【1】的整体格式下，函数还加入了goto函数，使用户在输入错误的指令下能够继续进行想要的操作。

特性如下：

1. 允许用户通过两种方式（名字或ID）来搜索相应的商品。
2. 在搜索到相应商品时，打印其相关信息。
3. 在用户指令出错时能够继续进行其功能。

**4.编辑（修改）函数**

void edit (void)

{

int i,j = 0;

char search\_way;

int id;

char name[NAME\_LEN];

char tem[999];

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

fp = fopen(F\_PATH, "r");

if (fp == NULL){

printf("Open failed , please check carefully.\n");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

fclose(fp);

if (j == 0){

printf("Nothing in the file!");

return;

}

EDIT:printf("Use ID/NAME to search the item <I/N>: ");

search\_way = getchar();

switch (search\_way){

case 'i' :

case 'I' :

printf("Please enter the ID of item : ");

scanf("%d", &id);

for (i = 0; i < j; i++){

if (items[i]->item\_id == id){

printf("The item you want to edit below: \n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

if (edit\_menu\_and\_function(i)){

printf("Edit the item successfully!~\n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

break;

}

else{

goto EDIT;

}

}

}

if (i == j){

printf("Didn't find the item!!!\n");

getchar();

printf("\nPress any key to contimue.\n");

getchar();

system ("cls");

return;

}

break;

case 'n' :

case 'N' :

printf("Please enter the NAME of item : ");

scanf("%s", name);

for (i = 0; i < j; i++){

if (strcmp(name, items[i]->item\_name) == 0){

printf("The item you want to edit below: \n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

if (edit\_menu\_and\_function(i)){

printf("Edit the item successfully!~\n");

printf("Now the information of item you changed is shown below\n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

break;

}

else{

goto EDIT;

}

}

}

if (i == j){

printf("Didn't find the item!!!\n");

getchar();

printf("\nPress any key to contimue.\n");

getchar();

system ("cls");

return;

}

break;

default:

printf("\nIllegle instruction code!\nPlease check carefully!!\n");

getchar();

goto EDIT;

}

fp = fopen(F\_PATH, "w");

fputs("ID NAME CATEGORY PRICE STOCK\n",fp);

for (i = 0; i < j; i++){

fprintf(fp, "%-10d", items[i]->item\_id);

fprintf(fp, "%-15s", items[i]->item\_name);

fprintf(fp, "%-15s", items[i]->item\_cate);

fprintf(fp, "%-20.2f", items[i]->item\_price);

fprintf(fp, "%-10d\n", items[i]->item\_stock);

}

fclose(fp);

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

}

getchar();

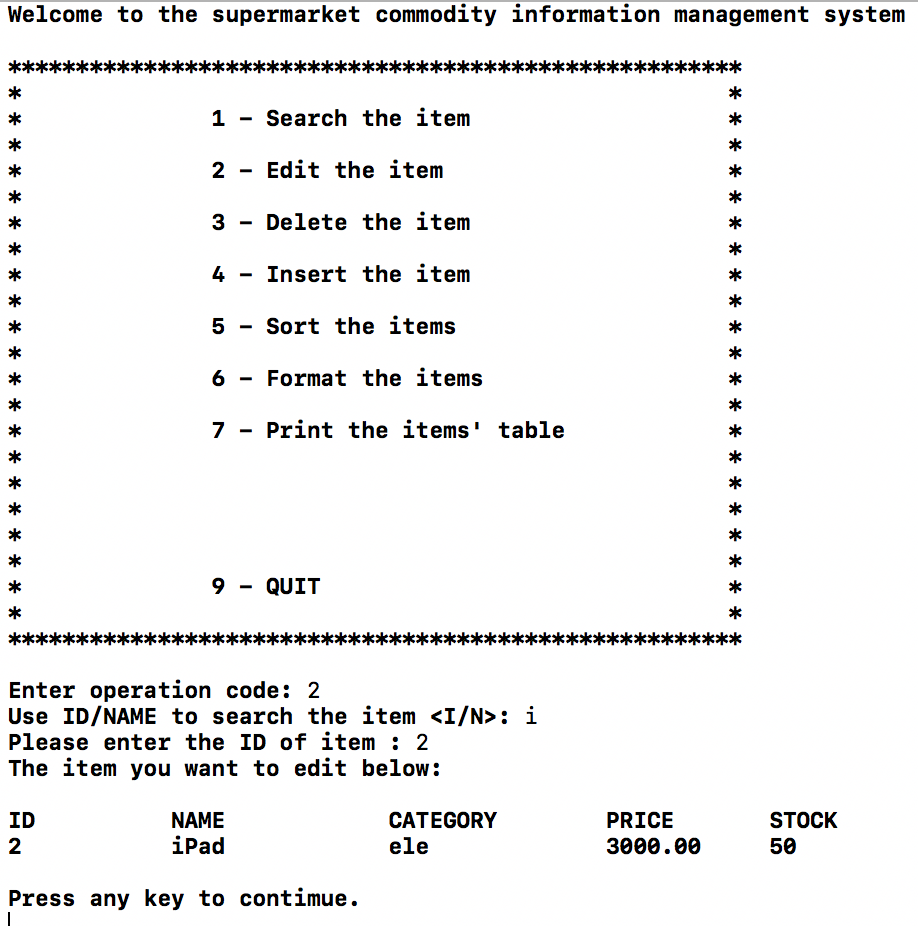
printf("\nPress any key to contimue.\n");

getchar();

system ("cls");

return;

}



由于此版本的编辑函数比较复杂，所以我将其写为了两个函数，避免看起来十分混乱。上面给出的edit函数功能是找到相应想要修改的商品，而下面给出的函数则是对其进行相应操作。

int edit\_menu\_and\_function (int i)

{

int j;;

char operation;

int id;

char name[NAME\_LEN];

char category[NAME\_LEN];

float price;

int stock;

printf("Welcome to the supermarket commodity information management system\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* \*\n");

printf("\* 1 - Edit the ID \*\n");

printf("\* \*\n");

printf("\* 2 - Edit the name \*\n");

printf("\* \*\n");

printf("\* 3 - Edit the category \*\n");

printf("\* \*\n");

printf("\* 4 - Edit the price \*\n");

printf("\* \*\n");

printf("\* 5 - Edit the stock \*\n");

printf("\* \*\n");

printf("\* 6 - Edit all the features \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\* 0 - BACK \*\n");

printf("\* \*\n");

printf("\* \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("Enter the operation code to edit: ");

O:

operation = getchar();

while(getchar() != '\n');

switch (operation) {

case '1':

printf("Enter the ID you want to change: ");

ID: scanf("%d", &id);

for (j = 0; j < num\_items; j++){

if (items[j]->item\_id == id){

printf ("\nThis ID has already been existed!!!, please chang another one: ");

goto ID;

}

}

items[i]->item\_id = id;

break;

case '2':

printf("Enter the NAME you want to change: ");

NAME: scanf("%s", name);

for (j = 0; j < num\_items; j++){

if (strcmp(items[j]->item\_name, name) == 0){

printf ("\nThis NAME has already been existed!!!, please chang another one: ");

goto NAME;

}

}

strcpy(items[i]->item\_name, name);

break;

case '3':

printf("Enter the CATEGORY you want to change: ");

scanf("%s", category);

strcpy(items[i] -> item\_cate, category);

break;

case '4':

printf("Enter the PRICE you want to change: ");

scanf("%f", &price);

items[i]->item\_price = price;

break;

case '5':

printf("Enter the STOCK you want to change: ");

scanf("%d", &stock);

items[i]->item\_stock = stock;

break;

case '6':

printf("Enter the ID you want to change: ");

ID\_A: scanf("%d", &id);

for (j = 0; j < num\_items; j++){

if (items[j]->item\_id == id){

printf ("This ID has already been existed!!!, please chang another one: ");

goto ID\_A;

}

}

items[i]->item\_id = id;

printf("Enter the NAME you want to change: ");

NAME\_A: scanf("%s", name);

for (j = 0; j < num\_items; j++){

if (strcmp(items[j]->item\_name, name) == 0){

printf ("\nThis NAME has already been existed!!!, please chang another one: ");

goto NAME\_A;

}

}

strcpy(items[i]->item\_name, name);

printf("Enter the PRICE you want to change: ");

scanf("%f", &price);

items[i]->item\_price = price;

printf("Enter the STOCK you want to change: ");

scanf("%d", &stock);

items[i]->item\_stock = stock;

break;

case '0':

UI\_clear();

return 0;

default:

printf("Wrong operation code, please enter agian: ");

goto O;

}

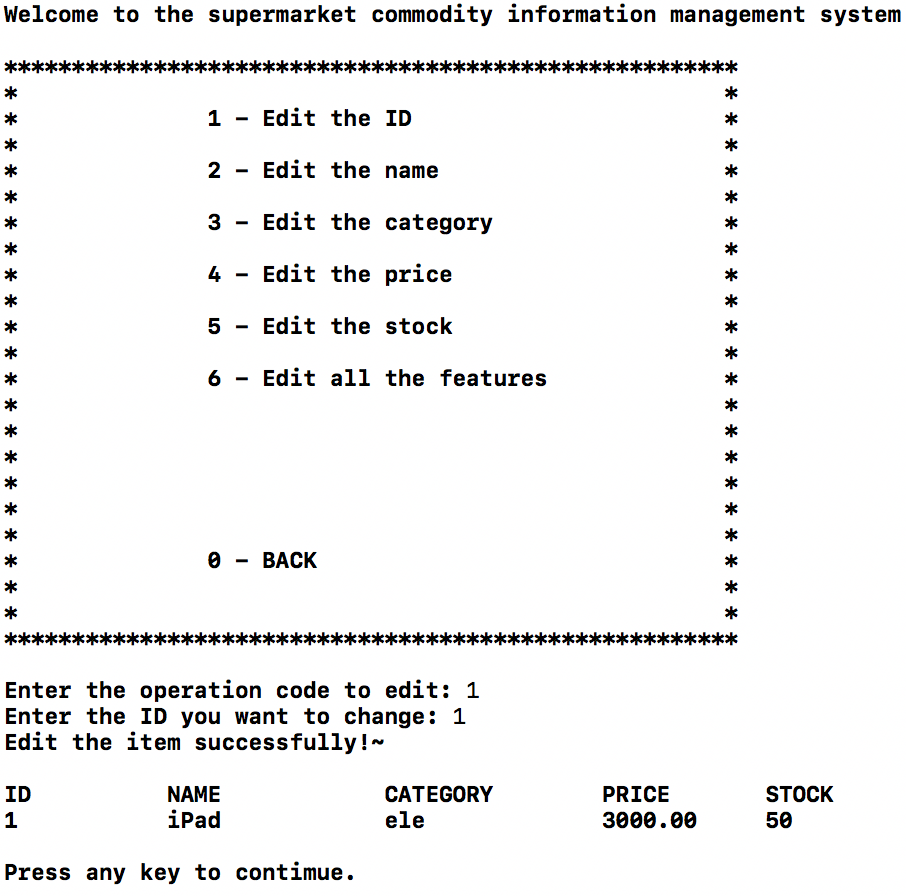
return 1;

}

不难看出，此函数包含了具体的操作界面和相应的功能。

整个“编辑”的功能是此版本中所有函数中最复杂的，因为其能实现的功能如下：

1. 通过名字或ID找到相应商品
2. 拥有相应的操作界面
3. **可以进行修改单一特征或全部特征**
4. 在误搜到其他商品信息时可以返回重新搜索
5. 一定规格的清屏使之易于交互



**5.删除函数**

void delete (void)

{

int i,j = 0;

char name[NAME\_LEN];

char search\_way;

int id;

char tem[999];

char confirm;

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

fp = fopen(F\_PATH, "r");

if (fp == NULL){

printf("Open failed , please check carefully.\n");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

fclose(fp);

if (j == 0){

printf("The file has already been formated!!!");

return;

}

DELETE:printf("Use ID/NAME to search the item <I/N>: ");

search\_way = getchar();

switch (search\_way){

case 'i' :

case 'I' :

printf("Please enter the ID of item : ");

scanf("%d", &id);

for (i = 0; i < j; i++){

if (items[i]->item\_id == id){

printf("The item you want to delete below: \n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

printf("Confirm to DELETE? <Y/N>: ");

confirm = getchar();

switch (confirm) {

case 'y':

case 'Y':

printf("\nYou quited the system successfully!~\n");

UI\_clear();

return;

default:

while(getchar() != '\n');

break;

}

for (; i <= j; i++){

\*items[i] = \*items[i + 1];

}

break;

}

}

if (i == j){

printf("Didn't find the item!!!\n");

UI\_clear();

return;

}

break;

case 'n' :

case 'N' :

printf("Please enter the NAME of item : ");

scanf("%s", name);

for (i = 0; i < j; i++){

if (strcmp(name, items[i]->item\_name) == 0){

printf("The item you want to delete below: \n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

printf("Confirm to DELETE? <Y/N>: ");

confirm = getchar();

switch (confirm) {

case 'y':

case 'Y':

printf("\nYou quited the system successfully!~\n");

UI\_clear();

return;

default:

while(getchar() != '\n');

break;

}

for (; i <= j; i++){

\*items[i] = \*items[i + 1];

}

break;

}

}

if (i == j){

printf("Didn't find the item!!!\n");

UI\_clear();

return;

}

break;

default:

printf("\nIlleagle instruction code!\nPlease check carefully!!\n");

while(getchar() != '\n');

goto DELETE;

}

j--;

fp = fopen(F\_PATH, "w");

fputs("ID NAME CATEGORY PRICE STOCK\n",fp);

for (i = 0; i < j; i++){

fprintf(fp, "%-10d", items[i]->item\_id);

fprintf(fp, "%-15s", items[i]->item\_name);

fprintf(fp, "%-15s", items[i]->item\_cate);

fprintf(fp, "%-20.2f", items[i]->item\_price);

fprintf(fp, "%-10d\n", items[i]->item\_stock);

}

fclose(fp);

printf("\nDelete successfully!~\n");

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

}

UI\_clear();

return;

}

同样在【1】的结构基础之下，删除函数的基本原理就是将想要删除的商品后面的商品依次向前移一位。

其能实现的具体功能如下：

1. 通过名字或ID找到相应的商品
2. 打印所想要删除的商品信息
3. 确认是否删除商品

**6.添加函数**

void insert (void)

{

int i,j = 0;

int id;

char name[NAME\_LEN];

char tem[999];

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

fp = fopen(F\_PATH, "r");

if (fp == NULL){

printf("Open failed , please check carefully.\n");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

fclose(fp);

if (j == 0){

printf("Nothing in the file!!!");

return;

}

if(j == MAX\_ITEMS){

printf("The space dosen't exists!\n");

return;

}

printf("Please enter the ID of the new item: ");

scanf("%d", &id);

for(i=0;i<j;i++){

if(id == items[i]->item\_id){

printf("The item has already existed!!\n");

return;

}

items[j] -> item\_id = id;

}

printf("Please enter the NAME of the new item: ");

scanf("%s", name);

for(i=0;i<j;i++){

if(items[i]==NULL){

printf("The space dosen't exists!\n");

}

if(strcmp(name,items[i]->item\_name)==0){

printf("The item has already existed!!\n");

return;

}

strcpy(items[j]->item\_name, name);

}

printf("Please enter the CATEGORY of the new item: ");

scanf("%s", items[j]->item\_cate);

printf("Please enter the PRICE of the new item: ");

scanf("%f", &items[j]->item\_price);

printf("Please enter the STOCK of the new item: ");

scanf("%d", &items[j]->item\_stock);

printf("\nInsert the new item successfully!~\n\n");

printf("Now the information of item you changed is shown below\n");

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

write\_line(i);

fp = fopen(F\_PATH, "w");

fputs("ID NAME CATEGORY PRICE STOCK\n",fp);

for (i = 0; i <= j; i++){

fprintf(fp, "%-10d", items[i]->item\_id);

fprintf(fp, "%-15s", items[i]->item\_name);

fprintf(fp, "%-15s", items[i]->item\_cate);

fprintf(fp, "%-20.2f", items[i]->item\_price);

fprintf(fp, "%-10d\n", items[i]->item\_stock);

}

fclose(fp);

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

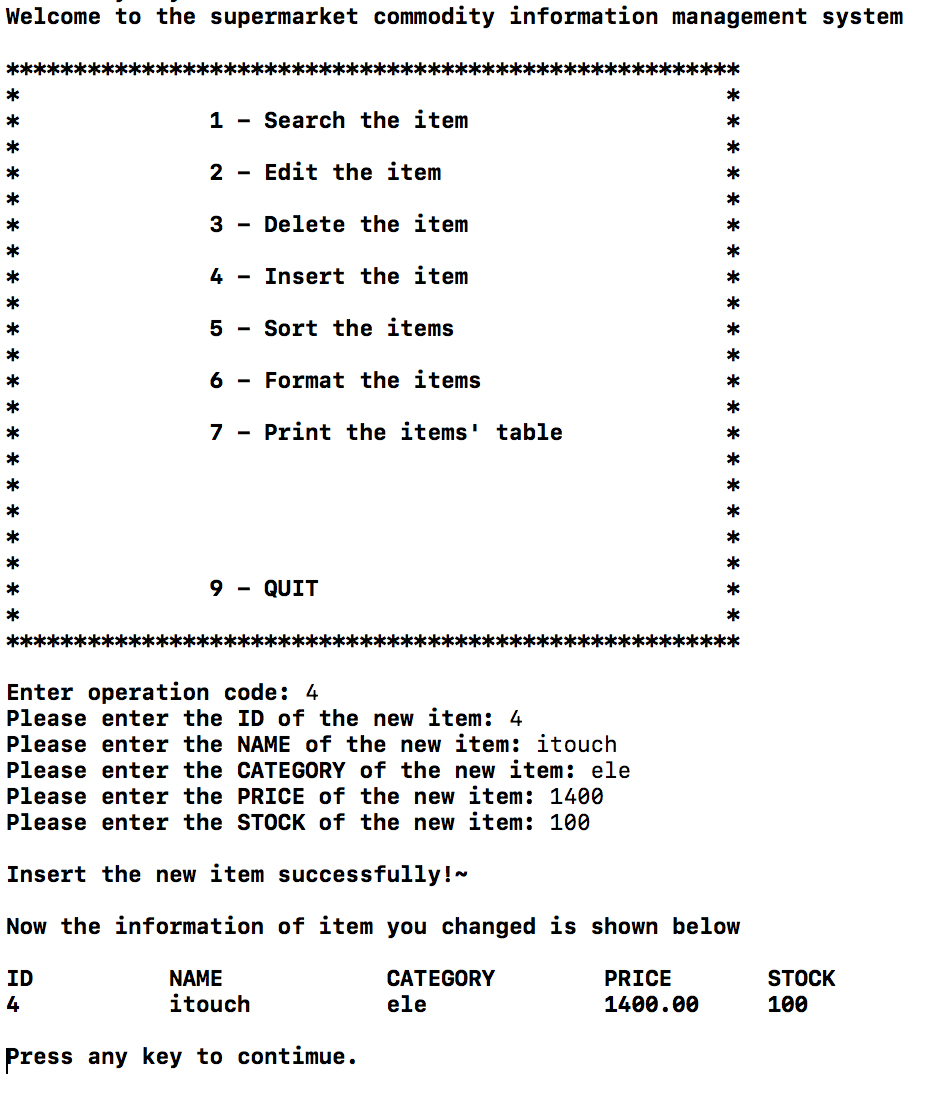
}

UI\_clear();

return;

}

添加（插入）函数具体实现方式和删除函数相差不大，思路一模一样，一个是减一行，一个是加一行。



其具体功能如下：

1. 添加相关物品信息
2. 打印所添加的物品

**7.排序函数**

void sort (void)

{

int i,j = 0;

char sort\_way;

int m,n;

char tem[999];

struct info\_items trans\_item;

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

if ((fp = fopen(F\_PATH,"r")) == NULL){

fprintf(stderr,"Can't open it, please check it carefully!!!");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

num\_items = j;

if (j == 0){

printf("Nothing in the file!");

return;

}

fclose(fp);

SORT:printf("Use ID/NAME/PRICE/STOCK to sort <I/N/P/S>: ");

sort\_way = getchar();

switch (sort\_way) {

case 'i':

case 'I':

for (m = 0; m < j - 1; m++){

for (n = 0; n < j - m - 1; n++){

if (items[n]->item\_id > items[n + 1]->item\_id){

trans\_item.item\_id = items[n]->item\_id;

strcpy(trans\_item.item\_name, items[n]->item\_name);

strcpy(trans\_item.item\_cate, items[n]->item\_cate);

trans\_item.item\_price = items[n]->item\_price;

trans\_item.item\_stock = items[n]->item\_stock;

\*items[n] = \*items[n + 1];

\*items[n + 1] = trans\_item;

}

}

}

break;

case 'n':

case 'N':

for (m = 0; m < j - 1; m++){

for (n = 0; n < j - m - 1; n++){

if (strcmp(items[n]->item\_name, items[n + 1]->item\_name)){

trans\_item.item\_id = items[n]->item\_id;

strcpy(trans\_item.item\_name, items[n]->item\_name);

strcpy(trans\_item.item\_cate, items[n]->item\_cate);

trans\_item.item\_price = items[n]->item\_price;

trans\_item.item\_stock = items[n]->item\_stock;

\*items[n] = \*items[n + 1];

\*items[n + 1] = trans\_item;

}

}

}

break;

case 'p':

case 'P':

for (m = 0; m < j - 1; m++){

for (n = 0; n < j - m - 1; n++){

if (items[n]->item\_price > items[n + 1]->item\_price){

trans\_item.item\_id = items[n]->item\_id;

strcpy(trans\_item.item\_name, items[n]->item\_name);

strcpy(trans\_item.item\_cate, items[n]->item\_cate);

trans\_item.item\_price = items[n]->item\_price;

trans\_item.item\_stock = items[n]->item\_stock;

\*items[n] = \*items[n + 1];

\*items[n + 1] = trans\_item;

}

}

}

break;

case 's':

case 'S':

for (m = 0; m < j - 1; m++){

for (n = 0; n < j - m - 1; n++){

if (items[n]->item\_stock > items[n + 1]->item\_stock){

trans\_item.item\_id = items[n]->item\_id;

strcpy(trans\_item.item\_name, items[n]->item\_name);

strcpy(trans\_item.item\_cate, items[n]->item\_cate);

trans\_item.item\_price = items[n]->item\_price;

trans\_item.item\_stock = items[n]->item\_stock;

\*items[n] = \*items[n + 1];

\*items[n + 1] = trans\_item;

}

}

}

break;

default:

printf("\nIlleagle instruction code!\nPlease check carefully!!\n");

goto SORT;

}

fp = fopen(F\_PATH, "w");

fputs("ID NAME CATEGORY PRICE STOCK\n",fp);

for (i = 0; i < num\_items; i++){

fprintf(fp, "%-10d", items[i]->item\_id);

fprintf(fp, "%-15s", items[i]->item\_name);

fprintf(fp, "%-15s", items[i]->item\_cate);

fprintf(fp, "%-20.2f", items[i]->item\_price);

fprintf(fp, "%-10d\n", items[i]->item\_stock);

}

fclose(fp);

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

}

while (getchar() != '\n');

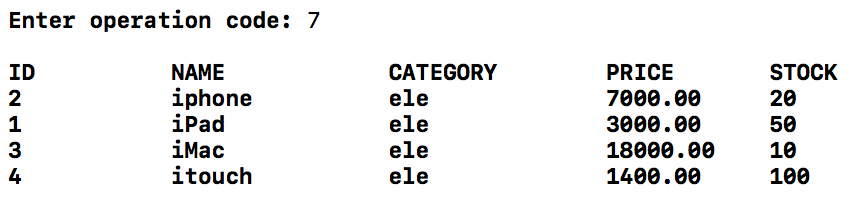
printf("Sort successfully!~");

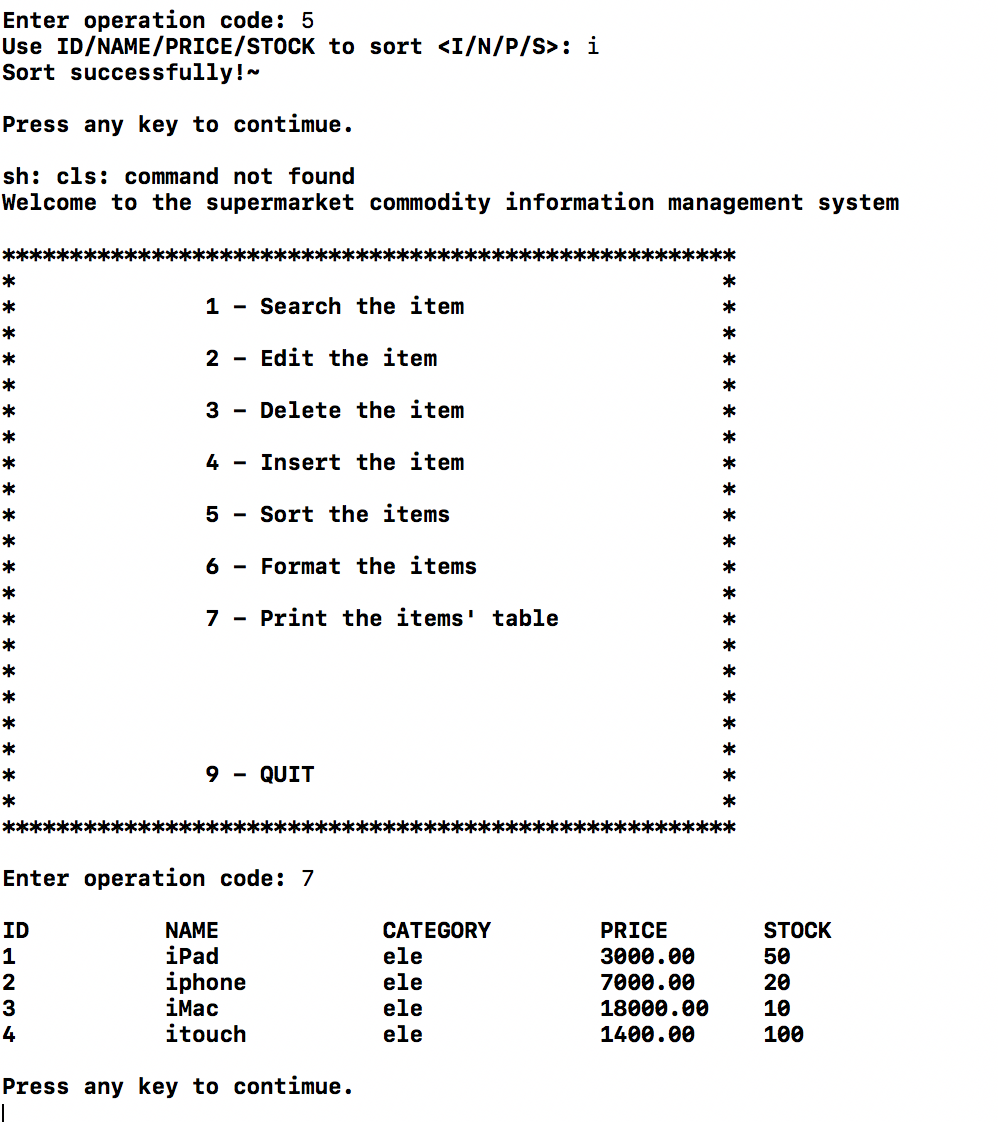
UI\_clear();

return;

}

在基于【1】结构下，通过冒泡排序，使得函数达成相应功能。





其具体功能为：通过名字／ID／价格／库存升序排序。

**8.格式化函数**

void format (void)

{

int i,j = 0;

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

fp = fopen(F\_PATH, "r");

if (fp == NULL){

printf("Open failed , please check carefully.\n");

return;

}

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

fclose(fp);

if (j == 0){

printf("The file has already been formated!!!");

return;

}

char confirm;

printf("\nConfirm to format the file <Y/N>: ");

if ((confirm = getchar()) == 'y' || confirm == 'Y' ){

fp = fopen(F\_PATH, "w");

fputs("ID NAME CATEGORY PRICE STOCK\n",fp);

fclose(fp);

printf("\nFormated successfully!~\nNow nothing in the file\n");

}

else{

while (getchar() != '\n');

printf("\nNow back to the menu.\n");

}

for (i = 0; i < MAX\_ITEMS; i++){

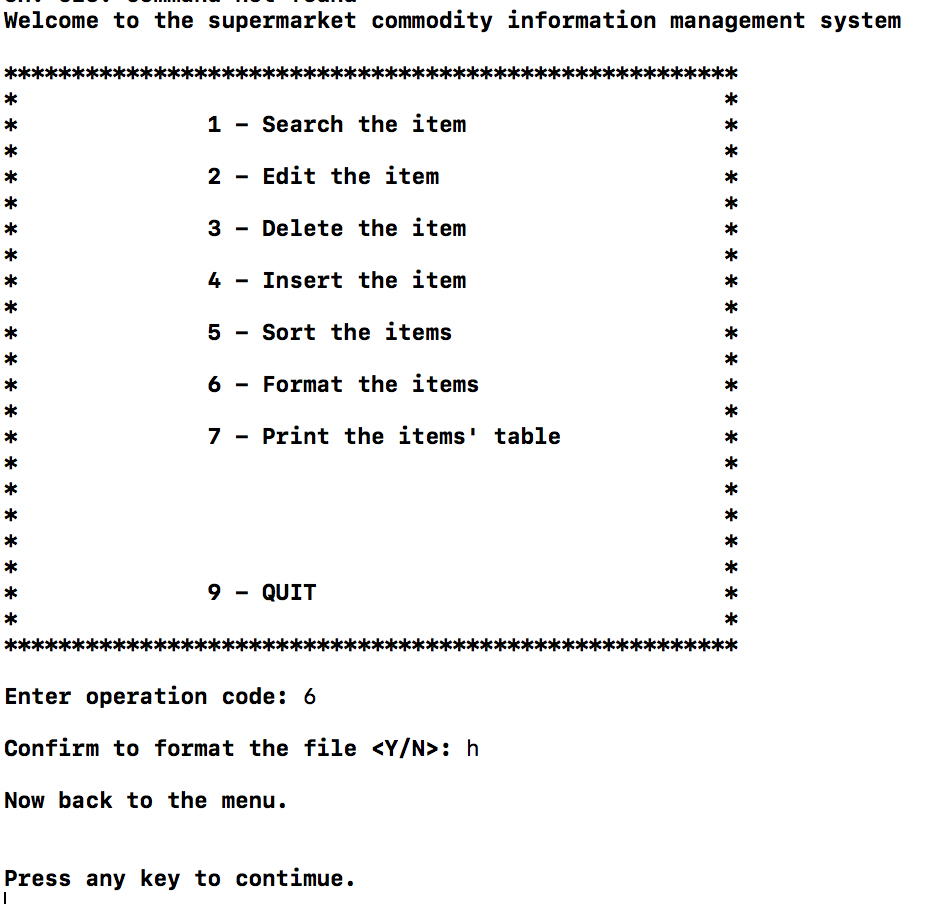
free(items[i]);

}

UI\_clear();

return;

}



此函数功能较简单，实现方式就是在写入的时候用“w”形式打开而不写入。

**9.打印函数**

void print (void)

{

int i,j = 0;

char tem[999];

for (i = 0; i < MAX\_ITEMS; i++){

items[i] = (struct info\_items\*)malloc(STRUCT\_LEN);

}

FILE \*fp = NULL;

if ((fp = fopen(F\_PATH,"r")) == NULL){

fprintf(stderr,"Can't open it, please check it carefully!!!");

return;

}

fgets(tem, 999, fp);

while(!feof(fp)){

items[j] = read\_line(fp);

j++;

}

num\_items = j;

if (j == 0){

printf("Nothing in the file!");

return;

}

fclose(fp);

printf("\nID\t\t\tNAME\t\t\tCATEGORY\t\tPRICE\t\tSTOCK\n");

for (i = 0; i < j; i++){

write\_line(i);

}

printf("\nPress any key to contimue.\n");

getchar();

system ("cls");

for (i = 0; i < MAX\_ITEMS; i++){

free(items[i]);

}

return;

}

在此就不给出其相应的图片解释了，前面有大量调用到此函数的地方。

也是基于【1】的结构而书写的函数，功能十分简单，就是打印。

**10.清屏函数**

void UI\_clear (void)

{

while(getchar() != '\n');

printf("\nPress any key to contimue.\n");

while(getchar() != '\n');

system ("cls");

}

**也是使用率极高，所以就不给出相应的使用了。**

**九、 总结及心得体会：**

本实验是绝对的综合能力实验，需要通过不断的调试修改才能完成，但由于时间有限，我还有很多地方没有改到，就我现在知道的就有输入类型错误的边界条件，希望以后能找时间写上去。

这个实验我也是写了很久，写了4个版本，最长的有1400多行，单后来还是选择了最后的版本。等寒假有空我还会给它加上模糊搜索和大小写转换。

综合型的试验就需要我们拥有综合型的思维，不能一开始就想具体实现，而是先构思整体，然后定义接口，最后才是相应的实现与边界条件的完善。

**十、 对本实验过程及方法、手段的改进建议：**

更多地去考虑高效复用的问题，这个版本中海油很多重复出现的代码，希望下次能够好好的把它精简化。

**报告评分：**

**指导教师签字：**