

河南工业大学

课 程 报 告

课 程 名 称： 程序设计实践

专 业 班 级： 软件 1601

学 生 姓 名： 高天

学 号： 201616030213

任 课 教 师： 徐振强

学 期： 2017-2018 学年第 2 学期

课程报告任务书

题目	小型超市库存与销售管理系统
主要内容	<p>用 C 语言开发一个小型超市库存与销售管理系统。</p> <p>【数据结构】</p> <p>1、库存数据文件。每个商品的信息包含：商品编号，商品名称，进价，建议售价，生产厂商，库存数量；</p> <p>2、销售数据文件。每个商品的信息包含：商品编号，商品名称，售价，销售数量，销售日期；</p> <p>3、用户数据文件。每个用户的信息包括：用户账号，密码，权限级别（区别系统管理员用户和普通用户）。</p> <p>【系统实现功能】</p> <p>1、用户管理（操作用户数据文件）。（1）<u>高级用户</u>（超市管理者）：具有所有功能，同时可以添加用户、删除用户。创建新用户后，将普通用户信息存储到文件“user.dat”中。（2）<u>普通用户</u>（售货员）：可执行销售业务处理和查询功能。</p> <p>2、进货&库存业务管理（操作库存数据文件）。根据商品编号查找相应商品，如果不存在，执行添加操作；如果商品已存在，根据进货数量修改商品数量。</p> <p>3、销售业务处理（操作库存数据文件和销售数据文件）。用户在销售数据文件中添加销售记录，同时更新库存数据文件的相应商品的信息。</p> <p>4、查询功能。支持以下几种查询方式：（1）<u>按商品名称</u>（精确查找，或模糊查找）查询商品基本信息（商品名、售价、库存）（2）<u>按商品名称和生产厂商</u>（多条件查询）查询商品基本信息（商品名、售价、库存）；（3）<u>按销售日期</u>查询某一天销售的情况。</p> <p>5、功能扩展。可以按照自己对商品库存与销售系统的理解和解决问题的程度对系统进行扩展。比如：（1）支持商品批量入库（从文件中导入）。（2）按照固定的日期范围统计所销售的货物信息（商品名称，销售总量，收益率），像，月销售情况统计，季度销售情况统计，年度销售情况统计。（3）按照销售量、销售利润等条件，统计指定日期范围内商品的销售情况。</p>
任务要求	<p>一、提交材料应包括：（1）系统源代码 （2）课程报告</p> <p>二、整个设计过程具体要求</p> <p>（1）需求分析 要求学生对案例系统进行分析，设计出需要完成的功能，完善各个模块的调用关系；</p> <p>（2）设计过程 要求学生进一步明确各模块调用关系，进一步完善模块函数细节（函数名、参数、返回值等）</p> <p>（3）实现过程 要求学生养成良好的编码习惯、完成各个模块并进行测试，最终完成系统整体测试；</p> <p>（4）总结阶段 按照要求完成系统设计和实现报告，并进行总结、答辩。</p>

1 需求分析

1) 登陆

管理员和售货员可通过各自的账号、密码分别进入管理员和售货员的子系统。对于输入不在系统所存储的账号或输入的账号密码不匹配时，要求用户重新输入。

2) 用户管理

管理员用户可浏览系统内所有的用户的账号、密码、权限类别，可添加用户，可删除用户。

3) 库存管理

管理员可手动添加商品，也可从文件中批量导入商品，可查看库存内的全部商品信息，对于库存内商品数为 0 的商品可进行批量清理。

4) 查询商品

管理员和售货员可通过商品名称、商品生产商、名称和生产商的方式查询商品信息。管理员可获取全部商品信息（商品 ID、商品名称、进价、售价、生产厂商、余量），售货员可获取出进价以外的商品信息。查询可支持模糊查找、仅输入前缀。

5) 销售商品

管理员和售货员可对库存内商品进行销售，对销售请求进行检查，销售后对库存相应商品的余量进行更新，同时记录销售的商品信息、销售时间，更新销售记录数据文件。

6) 销售统计

管理员可浏览某天的或日期区间内的所有销售记录，可对指定日期区间内的销售记录进行综合统计，统计每种商品的销量、收入，统计总收入，可通过销量、销售额筛选统计结果。

2 概要设计

1) 数据结构

单个商品数据用 Goods 结构体存储，多个商品用链表存储。

```
typedef struct
{
    int id;
    char name[MAXGOODSNAME];
    double buying_price;
```

```

    double selling_price;

    char manufacturer[MAXMANUFACTURERNAME];

    int quantity;
} Goods;

```

```

typedef struct GoodsListNode *GoodsList;

struct GoodsListNode
{
    Goods goods;

    GoodsList next;
};

```

单个销售数据用 SoldGoodsRecord 结构体存储，多个销售数据用链表存储。

```

typedef struct
{
    int id;

    char name[MAXGOODSNAME];

    double buying_price;

    double selling_price;

    int sold_quantity;

    SoldDate date;
}SoldGoodsRecord;

typedef struct RecordsListNode *RecordsList;

struct RecordsListNode
{
    SoldGoodsRecord record;

    RecordsList next;
};

```

2) 模块划分

a. 管理商品模块 `manage_goods.c`

对商品进行操作的函数集。

初始化商品链表: `GoodsList InitGoodsList();`

销毁商品链表: `void DeleteGoodsList(GoodsList head);`

向链表添加商品: `int AddGoodsToList(GoodsList head, Goods goods);`

遍历商品链表: `void TraverseGoodsList(GoodsList head, void(*Fun)(Goods *));`

显示商品信息: `void DisplayGoodsInfo(Goods *goods);`

显示商品基本信息: `void DisplayBasicGoodsInfo(Goods *goods);`

添加某商品数量:

`int IncreaseGoodsQuantity(GoodsList head, int id, int quantity);`

减少某商品数量:

`int ReduceGoodsQuantity(GoodsList head, int id, int quantity);`

从文件中导入商品: `void ImportGoodsFromFile(GoodsList head, FILE *fp);`

导出商品至文件: `void ExportGoodsToFile(GoodsList head, FILE *fp);`

清空数量为 0 商品: `void RemoveZeroQuantityGoods(GoodsList head);`

根据 ID 查找商品: `GoodsList FindGoodsByID(GoodsList head, int id);`

打开商品文件: `FILE* OpenGoodsFile(char *mod);`

b. 管理销售记录模块 `manage_records.c`

初始化销售记录链表: `RecordsList InitRecordsList();`

销毁销售记录链表: `void DeleteRecordsList(RecordsList head);`

遍历销售记录链表:

`void TraverseRecordsList(RecordsList head, void(*Fun)(SoldGoodsRecord *));`

显示一条销售记录信息: `void DisplayARecordInfo(SoldGoodsRecord *record);`

向文件追加一条销售记录:

`void AppendARecordToFile(SoldGoodsRecord record, FILE *fp);`

向链表条件一条销售记录:

`void AddRecordToList(RecordsList head, SoldGoodsRecord record);`

从文件中导入销售记录数据至链表:

```
void ImportRecordsFromFile(RecordsList head, FILE *fp);
```

获取当前销售时间: SoldDate GetNowDate();

打开销售记录文件: FILE* OpenRecordsFile(char *mod);

c. 查询模块 query.c

按名字在链表中查找商品:

```
GoodsList QueryGoodsByName(GoodsList head, char *name);
```

按生产商在链表中查找商品:

```
GoodsList QueryGoodsByManufacturer(GoodsList head, char *manufacturer)
```

按名字和生产商在链表中查找商品:

```
GoodsList QueryGoodsByNameAndManufacturer(GoodsList head, char  
*goods_name, char *manufacturer);
```

按日期区间查找销售记录:

```
void QuerySoldRecordsByDate(RecordsList head, SoldDate start, SoldDate  
end);
```

d. 统计模块 statistics.c

按日期区间对销售商品进行统计, 通过销量、销售额筛选统计结果:

```
void SoldStatisticsByDate(RecordsList head, SoldDate start, SoldDate end,  
int min_sold_cnt, int min_earnings);
```

e. 售货员模块 salesman.c

判断是否为销售员用户:

```
int IsSalesmanAccount(char *account, char *password);
```

显示待售商品: void DisplaySoldGoods(GoodsList head);

销售商品: void SoldGoods(GoodsList head);

销售员初始化菜单: void SalesmanInitMenu();

查询商品菜单: void SalesmanLookUpGoods(GoodsList head);

按名称查询商品: void SalesmanLookUpGoodsByName(GoodsList head);

按生产商查询商品: void SalesmanLookUpGoodsByManufacturer(GoodsList head);

按名称和生产商查询商品：

```
void SalesmanLookUpGoodsByNameAndManufacturer(GoodsList head);
```

f. 管理员模块 admin_user.c

判断是否为管理员用户：int IsAdminAccount(char *account, char *password);

添加用户：int AddAccount(UserAccount* user);

删除用户：int DeleteAccount(char *account);

显示所有用户信息：int DisplayAccountInfo();

管理员初始菜单：void AdminInitMenu();

库存管理菜单：void StockManagement();

添加商品：void AddGoodsToStock(GoodsList head);

商品批量入库：void BatchedStock(GoodsList head);

查看库存商品：void LookOverStock(GoodsList head);

查询商品菜单：void LookUpGoods(GoodsList head);

按名称查询商品：void LookUpGoodsByName(GoodsList head);

按生产商查询商品：void LookUpGoodsByManufacturer(GoodsList head);

按名称和生产商查询商品：

```
void LookUpGoodsByNameAndManufacturer(GoodsList head);
```

清理库存：void ClearStock(GoodsList head);

用户管理菜单：void UserManagement();

查看用户：void LookOverUser();

添加用户：void AddUser();

删除用户：void DeleteUser();

销售商品：void SoldManagement();

销售统计菜单：void SoldStatistics();

浏览单天销售记录：void SingleDaySoldRecords(RecordsList head);

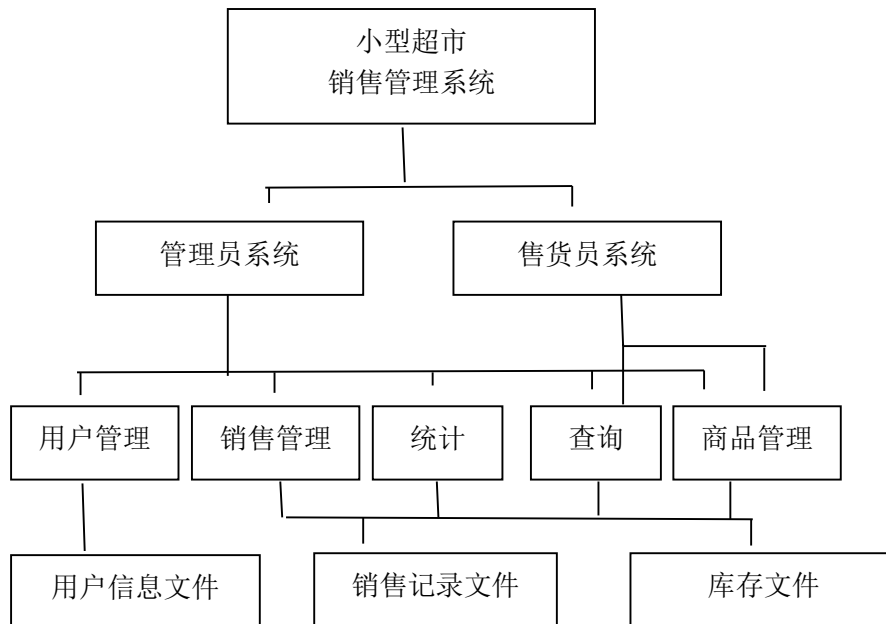
浏览多天销售记录：void DaysSoldRecords(RecordsList head);

按日期区间浏览销售统计：void DaysSoldStatistics(RecordsList head);

g. 外部 Hash 模块 uthash.c

开源 Hash 模块，实现商品按 id 哈希处理，进行销售统计。

3) 程序总体框架



3 详细设计

部分重要底层函数：

a) 管理商品模块 `manage_goods.c`

//初始化商品链表 创建带头结点的链表

```
GoodsList InitGoodsList()
```

```
{
```

```
    GoodsList head = (GoodsList)malloc(sizeof(struct GoodsListNode));
```

```
    head->next = NULL;
```

```
    return head;
```

```
}
```

//销毁链表

```
void DeleteGoodsList(GoodsList head)
```

```
{
```

```
    GoodsList next;
```

```
    while (head) {
```



```

        next = head->next;
        free(head);
        head = next;
    }
}

```

//在商品链表中添加一条商品信息，原有此商品数量合并，返回 1
 //原没有，在链表末尾添加，返回 0

```

int AddGoodsToList(GoodsList head, Goods goods)
{
    int id = goods.id;
    GoodsList p = head->next;
    while (p) {
        if (p->goods.id == id) {
            p->goods.quantity += goods.quantity;
            return 1;
        }
        p = p->next;
    }
}

```

```

GoodsList newNode = (GoodsList)malloc(sizeof(struct GoodsListNode));
newNode->goods = goods;
newNode->next = head->next;
head->next = newNode;

return 0;
}

```

//从文件中导入商品数据

```

void ImportGoodsFromFile(GoodsList head, FILE *fp)

```

```

{
    Goods goods;
    while (!feof(fp))
    {
        fscanf(fp, "%d%s%lf%lf%s%d\n", &goods.id, goods.name, &goods.buying_price, &goods.selling_price, goods.manufacturer, &goods.quantity);
        AddGoodsToList(head, goods);
    }
    fclose(fp);
}

```

b) 管理销售记录模块 `manage_records.c`

```

const char GOODS_SALES_RECORD_PATH[50] = "Data\\sold_goods_list.txt";

//初始化销售记录链表，创建带头结点的链表
RecordsList InitRecordsList()
{
    RecordsList head = (RecordsList)malloc(sizeof(struct RecordsListNode));
    head->next = NULL;
    return head;
}

//销毁销售记录链表
void DeleteRecordsList(RecordsList head)
{
    GoodsList next;

    while (head) {
        next = head->next;
        free(head);
    }
}

```

```

        head = next;
    }
}

//在销售记录链表尾部增加一条销售记录信息
void AddRecordToList(RecordsList head, SoldGoodsRecord record)
{
    RecordsList newNode =
        (RecordsList)malloc(sizeof(struct RecordsListNode));
    newNode->record = record;
    newNode->next = NULL;

    RecordsList p = head;
    while (p->next) p = p->next;

    p->next = newNode;
}

//向文件中新增加一条销售记录数据
void AppendARecordToFile(SoldGoodsRecord record, FILE *fp)
{
    fprintf(fp, "%d %s %.2f %.2f %d %d-%d-%d-%d:%d:%d\n", record.id,
        record.name, record.buying_price, record.selling_price, record.sold_quantity,
        record.date.year, record.date.month, record.date.day, record.date.hour,
        record.date.min, record.date.second);

    fclose(fp);
}

```

//从文件中导出销售记录数据

```
void ImportRecordsFromFile(RecordsList head, FILE *fp)
{
    SoldGoodsRecord record;
    while (!feof(fp))
    {
        fscanf(fp, "%d %s %lf %lf %d %d-%d-%d:%d:%d\n",
            &record.id, record.name, &record.buying_price,
            &record.selling_price, &record.sold_quantity, &record.date.year,
            &record.date.month, &record.date.day, &record.date.hour,
            &record.date.min, &record.date.second);
        AddRecordToList(head, record);
    }

    fclose(fp);
}
```

c) 查询模块 query.c

//按照商品名和生产厂商前缀查询商品，输出商品基本信息

```
GoodsList QueryGoodsByNameAndManufacturer(GoodsList head, char *goods_name,
char *manufacturer)
{
    if (head->next == NULL)
        return NULL;

    GoodsList queried_goods = InitGoodsList();
    GoodsList p = head->next;
    while (p) {
        char goods_name_prefix[MAXGOODSNAME] = { 0 };

```

```

char manufacturer_prefix[MAXMANUFACTURERNAME] = { 0 };
strncpy(goods_name_prefix, p->goods.name, strlen(goods_name));
strncpy(manufacturer_prefix,
        p->goods.manufacturer, strlen(manufacturer));
if(strcmp(goods_name_prefix, goods_name) == 0 &&
    strcmp(manufacturer_prefix, manufacturer) == 0) {
    GoodsList newNode =
        (GoodsList)malloc(sizeof(struct GoodsListNode));
    newNode->goods = p->goods;
    newNode->next = queried_goods->next;
    queried_goods->next = newNode;
}
p = p->next;
}

return queried_goods;
}

```

d) 统计模块 statistics.c

```

void SoldStatisticsByDate(RecordsList head, SoldDate start, SoldDate end,
int min_sold_cnt, int min_earnings)
{
    CountStatistics *s, *goods = NULL;
    RecordsList p = head->next;

    while (p) {
        if (CompareDate(p->record.date, start) >= 0 &&
            CompareDate(p->record.date, end) <= 0) {
            HASH_FIND_INT(goods, &p->record.id, s);
            if (s) {

```

```

        s->cnt += p->record.sold_quantity;
        s->earnings += (p->record.selling_price -
        p->record.buying_price) * p->record.sold_quantity;
    }
    else {
        s = (CountStatistics*)malloc(sizeof(CountStatistics));
        s->id = p->record.id;
        strcpy(s->name, p->record.name);
        s->cnt = p->record.sold_quantity;
        s->earnings = (p->record.selling_price-
        p->record.buying_price) * p->record.sold_quantity;
        HASH_ADD_INT(goods, id, s);
    }
}

p = p->next;
}

double total_earnings = 0;
int total_goods_cnt = 0;

printf("-----\n");
printf("%-5s %-12s %-6s %-6s\n", "ID", "名称", "销量", "收入");
for (s = goods; s != NULL; s = (CountStatistics*)(s->hh.next)) {
    if (s->cnt >= min_sold_cnt && s->earnings >= min_earnings) {
        total_earnings += s->earnings;
        total_goods_cnt++;
        printf("%-5d %-12s %-6d %-6.2f\n", s->id, s->name,
                s->cnt, s->earnings);
    }
}

```

```

    }

    printf("\n 总商品数:%d   总收入:%.2f 元\n",
           total_goods_cnt, total_earnings);

    printf("-----\n");
}

```

e) 售货员模块 salesman.c

```

//购买商品，并更新商品，销售记录
void SoldGoods(GoodsList head)
{
    int id, cnt;
    GoodsList found_goods;
    while (1) {
        system("cls");
        DisplaySoldGoods(head);
        printf("\n 输入待销售的商品 ID (输入-1 退出)\n>");
        scanf("%d", &id);

        if (id < 0) break;

        found_goods = FindGoodsByID(head, id);
        if (found_goods) {
            DisplayBasicGoodsInfo(&found_goods->goods);
            printf("输入销售数量\n>");
            scanf("%d", &cnt);
            if (cnt <= 0) {
                printf("商品数量有误\n");
                system("pause");
                continue;
            }
        }
    }
}

```

```

}
else {
    FILE *goods_fp = OpenGoodsFile("w");
    FILE *records_fp = OpenRecordsFile("a");
    if (goods_fp && records_fp) {
        if (!ReduceGoodsQuantity(head, id, cnt)) {
            printf("该商品库存不足\n");
            system("pause");
            continue;
        }
        ExportGoodsToFile(head, goods_fp);

        SoldGoodsRecord record;
        record.id = found_goods->goods.id;
        strcpy(record.name, found_goods->goods.name);
        record.selling_price = found_goods->goods.selling_price;
        record.sold_quantity = cnt;
        record.buying_price = found_goods->goods.buying_price;
        record.date = GetNowDate();

        AppendARecordToFile(record, records_fp);
        printf("销售成功\n");
        system("pause");
        continue;
    }
    else {
        if (goods_fp) fclose(goods_fp);
        if (records_fp) fclose(records_fp);
        printf("连接系统数据失败\n");
        system("pause");
    }
}

```



```

        break;
    }
}
else {
    printf("无此商品\n");
    system("pause");
    continue;
}
}
}

```

f) 管理员模块 admin_user.c

```

//判断是否为管理员账户，无法打开文件返回-1，是返回 1，否返回 0
int IsAdminAccount(char *account, char *password)
{
    FILE *fp = fopen("Data\\user.dat", "rb");

    if (fp == NULL)
        return -1;

    fseek(fp, 0, SEEK_END);
    int size = ftell(fp)/sizeof(UserAccount);
    fseek(fp, 0, SEEK_SET);

    UserAccount* user = (UserAccount*)malloc(size * sizeof(UserAccount));

    for (int i = 0; i < size; i++) {
        fread(user + i, sizeof(UserAccount), 1, fp);
        if (strcmp(account, user[i].account) == 0 &&

```

```

        strcmp(password, user[i].password) == 0 &&
        user[i].permission_level == 1)
        return 1;
    }

    free(user);
    fclose(fp);

    return 0;
}

//增加账户信息，无法打开返回-1，成功返回 1，账户已存在返回 0（更改密码）
int AddAccount(UserAccount *newInfo)
{
    FILE *fp = fopen("Data\\user.dat", "rb+");

    if (fp == NULL)
        return -1;

    fseek(fp, 0, SEEK_END);
    int size = ftell(fp) / sizeof(UserAccount);
    fseek(fp, 0, SEEK_SET);

    int index = -1;
    UserAccount *user = (UserAccount*)malloc((size + 1)
                                                * sizeof(UserAccount));
    for (int i = 0; i < size; i++) {
        fread(user + i, sizeof(UserAccount), 1, fp);
        if (strcmp(user[i].account, newInfo->account) == 0) {
            return 0;
        }
    }
}

```

```

    }
}

user[size] = *newInfo;
fseek(fp, 0, SEEK_SET);

for (int i = 0; i <= size; i++) {
    fwrite(user + i, sizeof(UserAccount), 1, fp);
}

free(user);
fclose(fp);
return 1;
}

```

4 调试分析

问题 1: 链表操作易出错

解决: 为链表增加空的头结点, 便于操作。销毁函数用于销毁链表, 避免内存泄漏。对于同一级均要操作的链表的, 在该级初始化链表, 在返回上级时销毁。

问题 2: 需要读取多个文件

解决: 将文件读取封装为函数调用, 对于售货功能, 需要同时更新商品文件和销售记录文件, 两个文件需要同时可打开时才可写入, 采用类似原子操作思想。

问题 3: 交互逻辑使用不便。

解决: 改进交互逻辑, 提供跳转功能、选择错误处理。

问题 4: 销售记录是分散记录, 难于以商品为单位整合统计

解决: 采用开源的 Hash 库, 通过 Hash 算法以商品 id 为 key 进行整合, 算出该 id 商品的总销量和总收入。

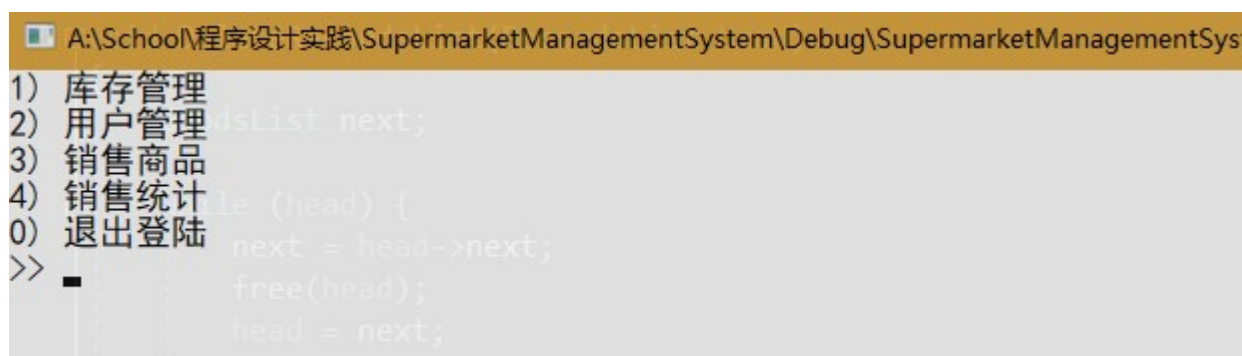
问题 5: 指定日期区间查询或统计时不便

解决: 构造日期结构体, 设置日期比较规则。

程序改进：

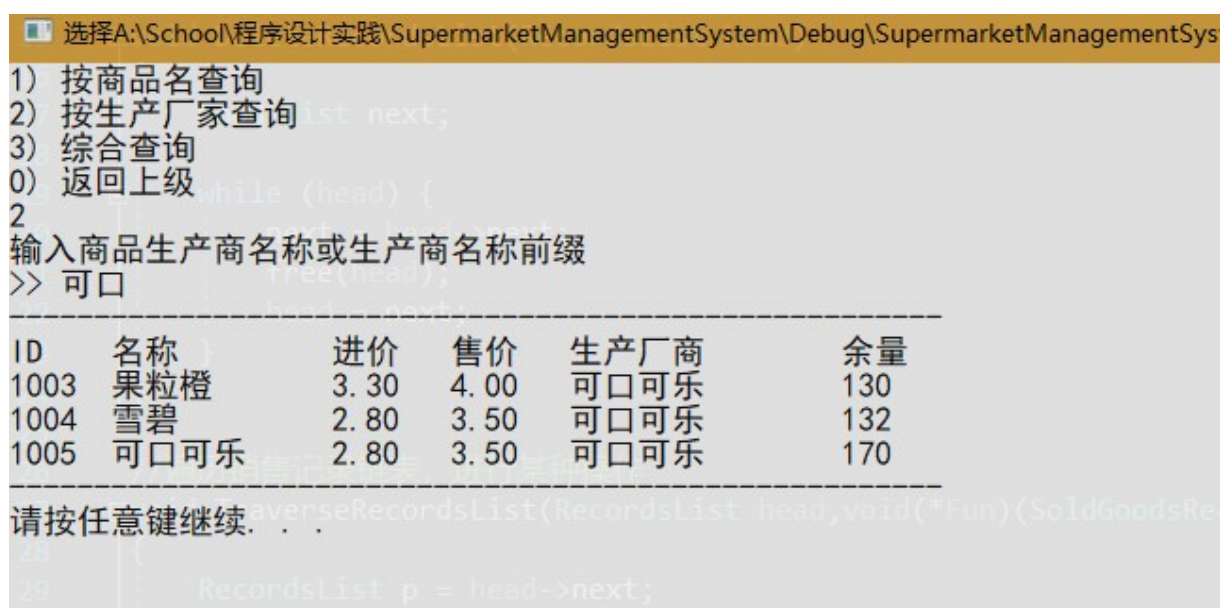
1. 改进操作逻辑，使得操作更加方便。
2. 增强大量数据时的处理能力
3. 采用数据库管理信息。
4. 考虑多用户并发问题。
5. 增加程序鲁棒性。
6. 改进交互。

5 测试结果



```
A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSys
1) 库存管理
2) 用户管理
3) 销售商品
4) 销售统计
0) 退出登陆
>> █
```

图 1. 管理员功能列表



```
选择A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSys
1) 按商品名查询
2) 按生产厂家查询
3) 综合查询
0) 返回上级
2
输入商品生产商名称或生产商名称前缀
>> 可口

ID    名称      进价  售价  生产厂商  余量
1003  果粒橙    3.30  4.00  可口可乐  130
1004  雪碧      2.80  3.50  可口可乐  132
1005  可口可乐  2.80  3.50  可口可乐  170

请按任意键继续. . .
```

图 2. 查询功能

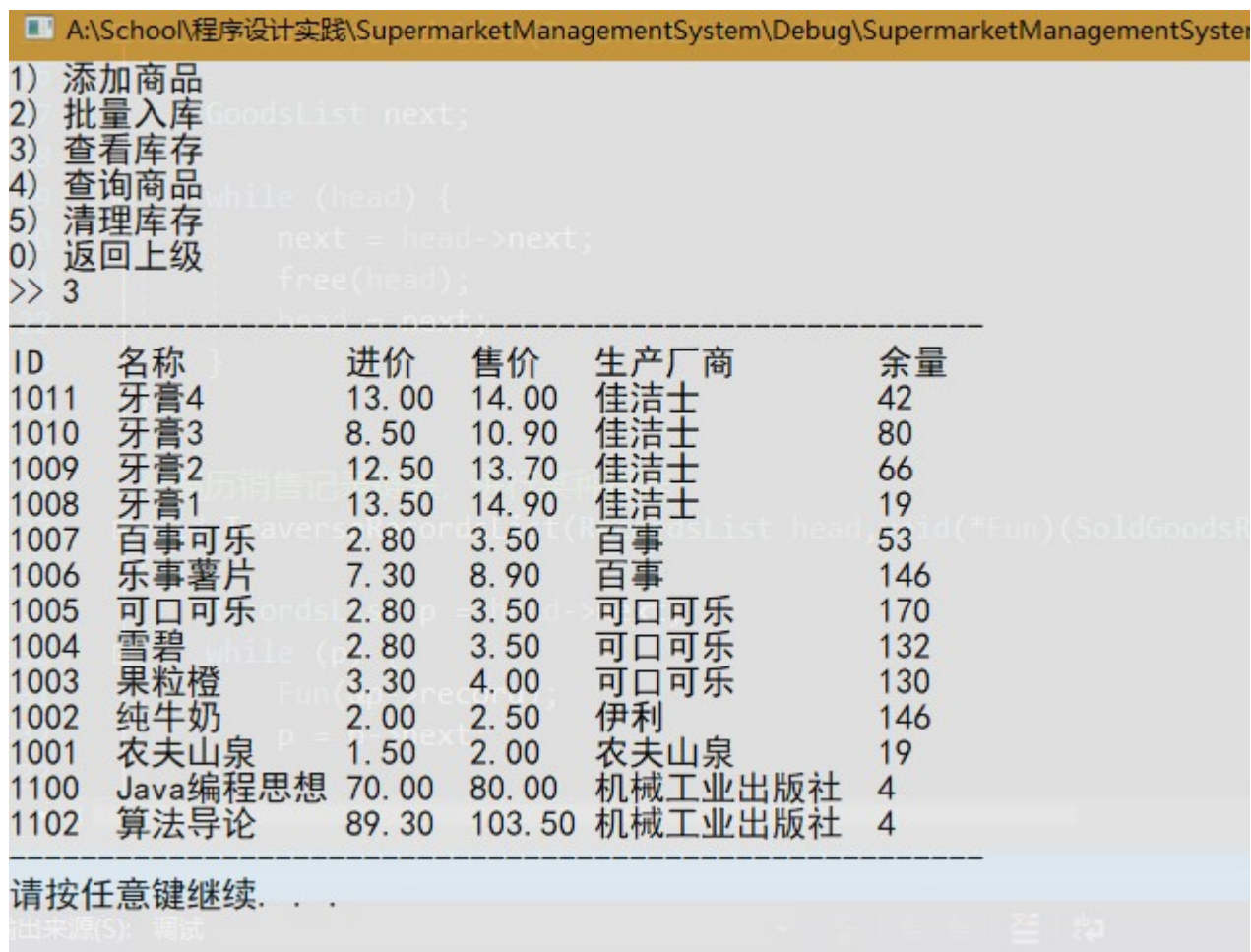


图 3. 查看库存功能

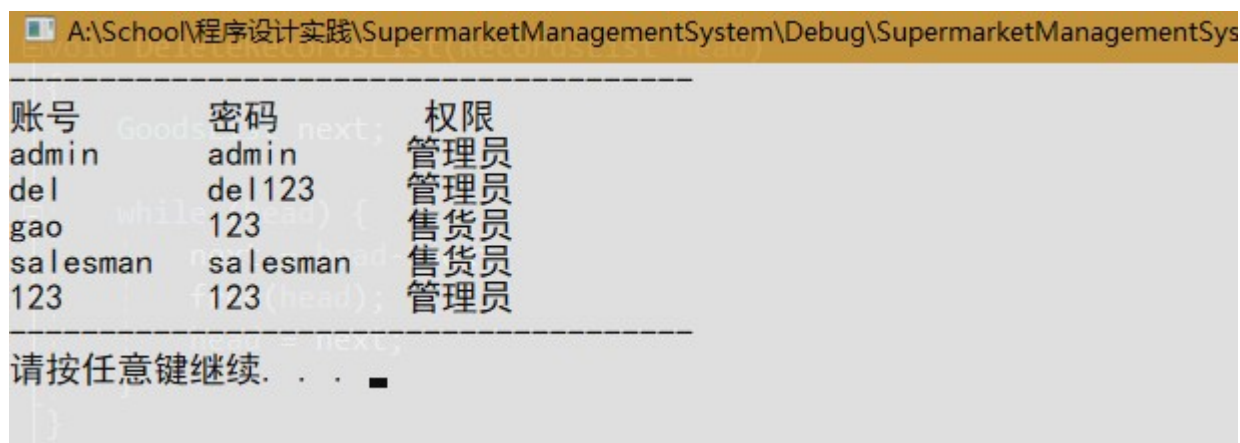


图 4. 浏览用户功能

```

A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSystem.exe
ID      名称      售价      生产厂商      余量
1011    牙膏4      14.00      佳洁士        42
1010    牙膏3      10.90      佳洁士        80
1009    牙膏2      13.70      佳洁士        66
1008    牙膏1      14.90      佳洁士        19
1007    百事可乐    3.50      百事          53
1006    乐事薯片    8.90      百事          146
1005    可口可乐    3.50      可口可乐      170
1004    雪碧        3.50      可口可乐      132
1003    果粒橙      4.00      可口可乐      130
1002    纯牛奶      2.50      伊利          146
1001    农夫山泉    2.00      农夫山泉      19
1100    Java编程思想 80.00      机械工业出版社 4
1102    算法导论    103.50     机械工业出版社 4

输入待销售的商品ID (输入-1退出)
>1102
1102  算法导论    103.50  机械工业出版社  4
输入销售数量
>1
销售成功
请按任意键继续. . .

```

图 5. 销售商品功能(商品 1102 销售中)

```

A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSystem.exe
ID      名称      售价      生产厂商      余量
1011    牙膏4      14.00      佳洁士        42
1010    牙膏3      10.90      佳洁士        80
1009    牙膏2      13.70      佳洁士        66
1008    牙膏1      14.90      佳洁士        19
1007    百事可乐    3.50      百事          53
1006    乐事薯片    8.90      百事          146
1005    可口可乐    3.50      可口可乐      170
1004    雪碧        3.50      可口可乐      132
1003    果粒橙      4.00      可口可乐      130
1002    纯牛奶      2.50      伊利          146
1001    农夫山泉    2.00      农夫山泉      19
1100    Java编程思想 80.00      机械工业出版社 4
1102    算法导论    103.50     机械工业出版社 3

输入待销售的商品ID (输入-1退出)
>

```

图 6. 销售商品功能(商品 1102 销售后)


```

A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSystem
1) 单天记录
2) 多天记录
3) 多天统计
0) 返回上级
>> 1
依次输入年 月 日
>> 2018 5 1
-----
ID    名称          进价    售价    购买量    购买时间
1100  Java编程思想  70.00   80.00    1    2018\05\01 11:14:29
1102  算法导论      89.30  103.50    2    2018\05\01 11:32:41
1010  牙膏3         8.50   10.90    7    2018\05\01 11:34:01
1012  牙膏5        13.30   14.10    3    2018\05\01 11:34:19
1102  算法导论      89.30  103.50    1    2018\05\01 20:40:08
-----
请按任意键继续. . .

```

图 7. 单天销售记录(最后一条为 1102 销售记录)

```

输入查询开始日期 年 月 日
>> 2018 4 29
输入查询结束日期 年 月 日
>> 2018 5 1
-----
ID    名称          进价    售价    购买量    购买时间
1008  牙膏1        13.50   14.90    2    2018\04\29 20:48:14
1011  牙膏4        13.00   14.00    3    2018\04\30 15:16:20
1008  牙膏1        13.50   14.90    2    2018\04\30 16:31:03
1005  可口可乐      2.80    3.50   10    2018\04\30 23:07:30
1002  纯牛奶        2.00    2.50    3    2018\04\30 23:37:55
1011  牙膏4        13.00   14.00    2    2018\04\30 23:40:35
1100  Java编程思想  70.00   80.00    1    2018\05\01 11:14:29
1102  算法导论      89.30  103.50    2    2018\05\01 11:32:41
1010  牙膏3         8.50   10.90    7    2018\05\01 11:34:01
1012  牙膏5        13.30   14.10    3    2018\05\01 11:34:19
1102  算法导论      89.30  103.50    1    2018\05\01 20:40:08
-----
请按任意键继续. . .

```

图 8. 多天销售记录

```
A:\School\程序设计实践\SupermarketManagementSystem\Debug\SupermarketManagementSys
1) 单天记录
2) 多天记录
3) 多天统计
0) 返回上级
>> 3
输入查询开始日期 年 月 日
>> 2018 4 1
输入查询结束日期 年 月 日
>> 2018 6 30
输入最小销量
>> 0
输入最小销售额
>> 0

-----
ID    名称    销量    收入
1001  农夫山泉    140    70.00
1002  纯牛奶      66    33.00
1003  果粒橙      56    39.20
1008  牙膏1        5     7.00
1007  百事可乐     5     3.50
1006  乐事薯片     6     9.60
1005  可口可乐    13     9.10
1004  雪碧         4     2.80
1009  牙膏2        5     6.00
1011  牙膏4        5     5.00
1100  Java编程思想  1    10.00
1102  算法导论     3    42.60
1010  牙膏3        7    16.80
1012  牙膏5        3     2.40

总商品数:14  总收入:257.00元
-----
请按任意键继续. . .
```

图 9. 2018 第二季度销售统计

6 课程心得总结

经过一周的编写完成了此次大作业，基本上实现了全部功能和扩展功能。由于自己是补修，总体来说不太困难，只是思想的角度方面有了较大的提升。能从更为全局的角度去设计系统。此系统采用分层的结构，先实现底层的接口函数，再以此为基础实现上层面向用户的功能。由于功能较多、结构较为复杂，采用模块化的编程，将程序分为若干独立的模块去实现相应的功能。通过分层和模块化设计，极大的提高了程序的逻辑与编码效率。

考虑到程序会涉及较多的插入删除操作，以及进一步练习链表的使用，程序商品数据和销售数据均采用链表存储。经过实践，发现链表还是很好用的，是个非常有用的数据结构。在实现销售数据统计功能，相同 id 商品要统计在一起时，用链表不能很方便解决这个问题，采用 Hash 会较为简单，于是上网搜索 C 语言的 Hash 库，搜到了 `uthash` 开源 Hash 库，通过查看文档，很简单的就实现了合并功能。

总之，通过这次大作业，对 C 语言有了更进一步的理解，对 List 和 Hash 的应用有了更清晰的认识，也通过这次实践提高了自己的编码能力，认识到只有将理论和实践结合起来我们才能真正学好编程。

7 参考文献

- [1] 甘勇, 李晔, 卢冰. C 语言程序设计[M]. 北京: 中国铁道出版社, 2014.
- [2] 谭浩强. C 程序设计[M]. 北京: 高等教育出版社, 2010.
- [3] 苏小红, 王宇颖, 孙志岗. C 语言程序设计[M]. 北京: 高等教育出版社, 2011.
- [4] 王新, 孙雷. c 语言课程设计[M]. 北京: 高等教育出版社, 2009.
- [5] www.csdn.net
- [6] <https://stackoverflow.com/>
- [7] <http://troydhanson.github.io/uthash/userguide.html>