电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2018091619034

姓 名 王省

（实验） 课程名称 程序设计与算法基础

理论教师 白忠建

实验教师 白忠建

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：王省 学号：2018091610934 指导教师：白忠建**

**实验地点：电子科技大学清水河校区 实验时间：2019.01.09**

1. **实验名称：超市商品管理系统链表实现**
2. **实验学时：4学时**
3. **实验目的：**
4. 掌握单链表的定义和使用方法
5. 掌握单链表的建立方法
6. 掌握单链表中节点的查找与删除
7. 掌握输出单链表节点的方法
8. 掌握链表节点排序的一种方法
9. 掌握C语言创建菜单的方法
10. 掌握结构体的定义和使用方法
11. **实验原理：**
12. 结构体：结构体是由不同数据类型组成的，结构是可能具有相同或不同类型的值(成员)的集合。组成结构的每一个数据称为该结构的成员项。
13. 结构体的定义：结构体的定义是宣布该结构由哪几个成员项组成。用以下方式定义一个结构：

struct 结构名{

数据类型 成员名1；

数据类型 成员名2

…

}；

例如：在书这个结构中，包括了标题，作者，价格三个成员。

struct book{

char title[MAXTITLE];

char author[MAXAUTL];

float value;

} ;

1. 结构体变量的声明：以以下形式声明一个结构体变量：

<存储类型> struct 结构名 结构变量

例如struct book library。

也可以将定义和声明同时进行。例如：

struct book{

char title[MAXTITLE];

char author[MAXAUTL];

float value;

} library1, library2;

library1中的成员不会与library2中的成员冲突。

每个结构变量都有三个成员:title(书的名称)、auther(书的作者) 和 value(书的价格)。这里的声明格式和C语言中其他变量的声明格式一样。struct{...}指明了类型。而library1和library2则是具有这种类型的变量。结构的成员在内存中是按照声明的顺序存储的。

1. 计算结构变量的内存大小：通常使用sizeof运算求出。
2. 对结构的操作:既然最常见的数组操作是取下标，那么结构最常用的操作是选择成员也就无需惊讶了。但是结构成员是通过名字而不是通过位置访问的。为了访问结构内的成员，首先写出结构的名字，然后写一个句点，再写出成员的名字。访问一个结构体中的成员使用以下形式表示：

结构变量名.成员

例如，下列语句显示结构library1的成员的值:  
printf(“Part number:%d\n”**, library1.title**);  
printf(“Part name:%s\n”, **library1.auther**);  
printf(“Quantity on hand:%d\n”, **library1.value**);

结构的成员是左值，所以它们可以出现在赋值运算的左侧，也可以作为自增或自减表达式的操作数:

library1.book = “to Kill a Mocking Bird” ;  
library1.value++;

用于访问结构成员的句点实际上就是一个C语言的运算符，句点运算符的优先级几乎高于所有其他运算符。  
结构的另一种主要操作是赋值运算:

library1 = library2;

这一句的效果是把library1.book复制到 library2.book,把library2.auther复制到 library1.auther,依次类推。

1. 结构体变量的初始化：一般形式为：

struct 结构名 结构变量 = {初始数据}

例如：

struct book library = {“to Kill a Mocking Bird”; “Harper Lee”;43.5}

其实现的效果和对其成员赋值的效果一样。

library .title = “to Kill a Mocking Bird”;

library .author = “Harper Lee”;

library .value = 43.5

1. 结构指针：指向结构的指针称为结构指针。其定义格式为：

<存储类型> struct 结构名 \*结构指针名

例如 struct library \*p;

使用指针对结构成员进行引用的时候，一般有两种形式：

(\*结构指针名) .成员名

结构指针名->成员名

例如

(\*p). book; p->book；

这两种等价

1. 结构类型的定义：形式为typedef <存储类型> 新命名。

例如：

typedef p\* nodeptr;

1. 链表存储结构：链表是一种很常见重要的结构。它是存储分配的一种结构。链表有一个头指针的变量，一般以head表示。它没有数据域，只有指针域。存放一个地址，指向第一个元素。链表中每一个元素称为“节点”，每一个节点包含两个部分：(1)用户所需要的实际数据，即数据域；(2)下一个节点指向下一个指针。
2. 建立动态链表：步骤：(1) 使用malloc（）函数为一个结构分配足够的空间。(2)．存储这个结构的地址。(3)．把正确的信息复制到这个结构中。
3. 文件操作实例

FILE \*fp; //定义文件操作的指针

//fopen用于打开文件，接收两个参数，一个是文件的路径，一个是文件打开的方式。例如xxxxxx.txt和该项目的可执行文件在同一目录下，则此处只需要所读取内容文件名，r代表以只读方式打开文件

fp = fopen("xxxxxxx.txt", "r");

//如果以w方式代表打开只写文件，若文件存在则长度清为0，即该文件内容消失，若不存在则创建该文件，其余打开方式请自行查阅文档

fp = fopen("xxxxxxx.txt", "w");

//fscanf用于从txt文件中读取内容，下例以字符串形式读入一段字符并存放在tempstring中

fscanf(fp, "%s", tempstring);

//或者以格式化的方式读入字符串

fscanf(fp, "\t%s\n", tempstring);

//fprintf以格式化的方式向txt文件中写入内容

fprintf(fp, "%s\t", tempstring);

//检查文件内容是否已经读到文件结束符了

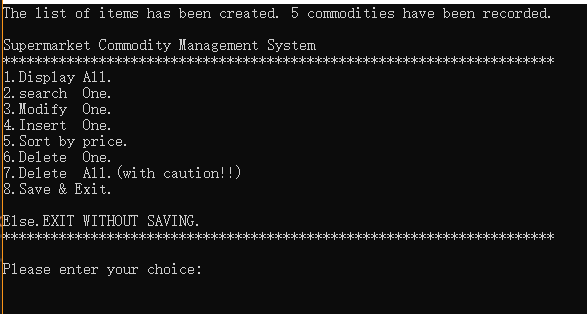
while ( !feof(fp)){…………}

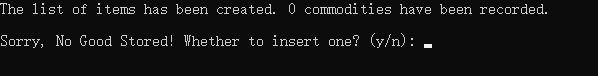
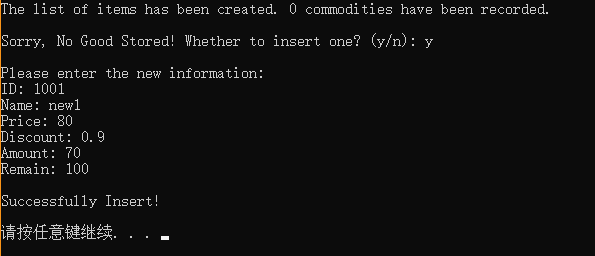
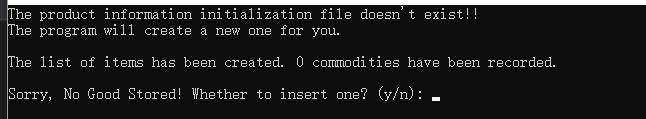
//最后需要使用fclose关闭文件指针

fclose(fp);

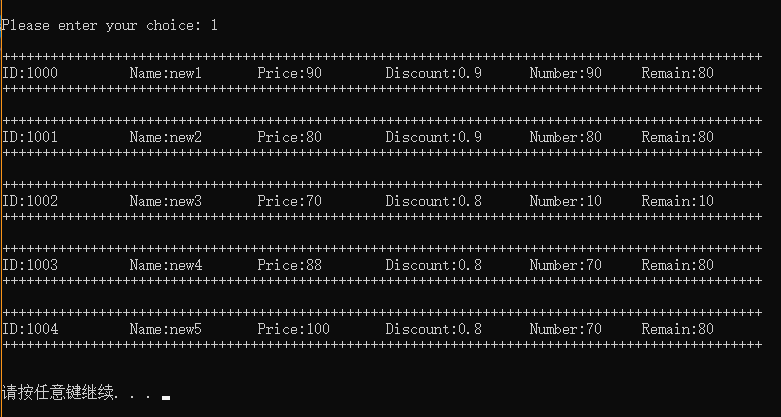
1. **实验内容：**

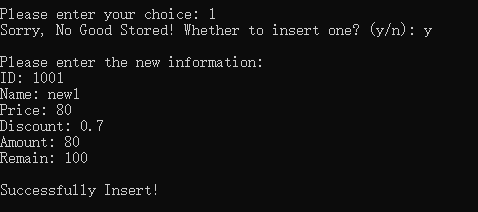
用C语言+单链表数据结构实现一个小型的超市商品管理系统，该系统需要具备商品信息录入、商品信息修改、商品信息删除、商品信息查找、商品信息的插入这几个功能。具体实现步骤如下:

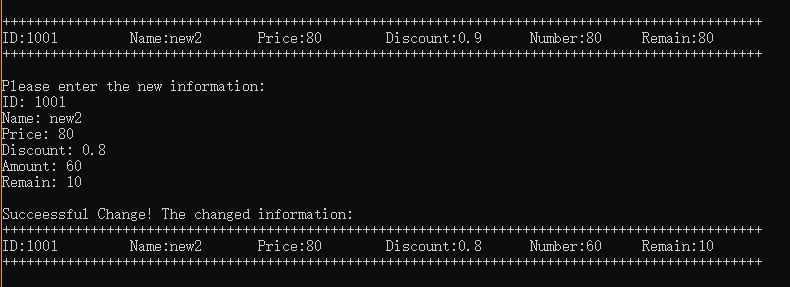
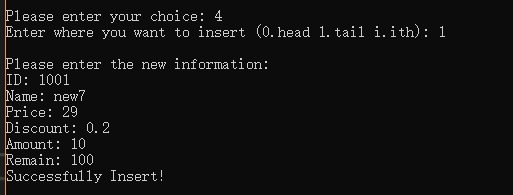
* 1. 软件界面控制:实现一个数字选项式的启动界面，包含显示所有商品信息、商品信息插入、商品信息修改、商品信息删除、商品信息查找、退出系统并保存、商品价格排序、删除所有的数据8个选项。并且这些功能可以循环调用。
  2. 商品信息的初始化：定义链表并初始化。实现从已有的商品信息文件中读入商品信息，并且分配内存保存至链表中。如1）里的图示所示，从文件中读取了5个商品记录。
  3. 特别的，当文件不存在或文件为空时，询问用户是否输入文件，若确定，则提示输入商品信息：



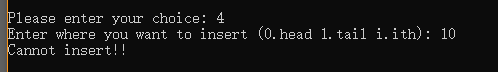
上图分别为商品文件不存在和文件为空时的情景。

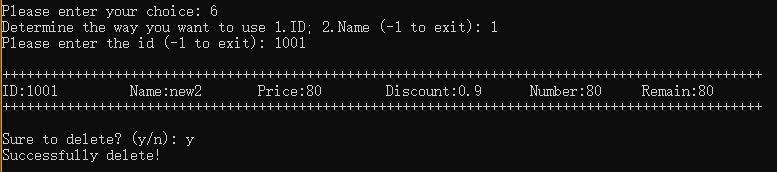
* 1. 所有商品的展示： 定义一个函数，将链表中所有商品信息以格式化的方式打印到屏幕上，如图所示。

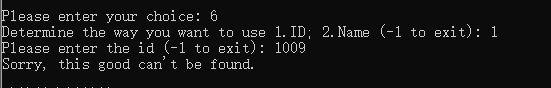
当无商品时会询问用户是否录入商品，若确定，输入信息。如图所示：

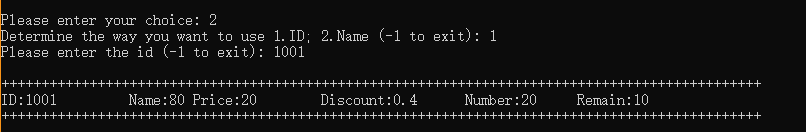
* 1. 修改商品的信息：定义一个函数，实现商品信息的修改。用户可通过选择id或者名字查询到商品进而进行修改信息。如图所示：
  2. 插入商品的信息：定义一个函数，实现商品信息的插入。读取用户输入的信息，保存至链表节点。用户可通过选择头插、尾插、中间插三种插法进行插入。其中中间插法用i表示。如图所示：

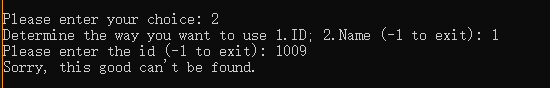
特别的，当使用中间插，输入的数大于商品总数时会提示错误：

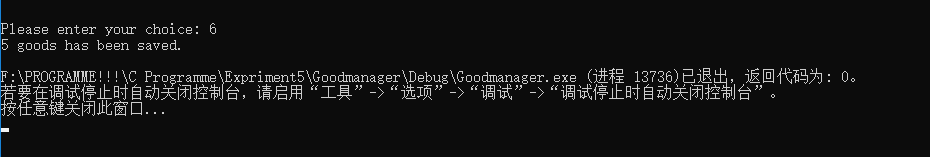
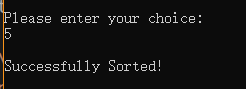
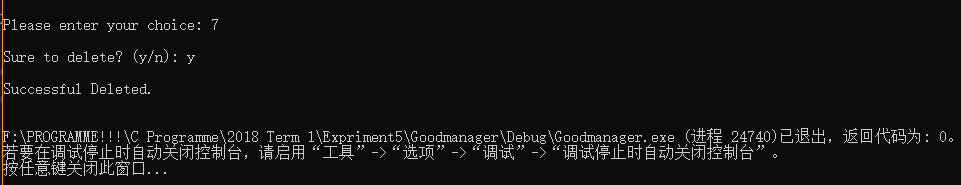


* 1. 删除商品信息：定义一个函数，能够删除一个商品的信息。用户可通过输入id或名称查找删除该商品，并销毁该商品的节点。如图所示：

若查找不到该商品时，会提示无此商品并返回主菜单，如图所示：

* 1. 查找商品的信息：定义一个函数，能够查找一个商品的信息。用户可通过输入id或名称查找。查找功能用字符串的比较来实现。如图所示：

若查找不到该商品时，会提示无此商品并返回主菜单，如图所示：

* 1. 存档并退出程序：定义一个函数，实现商品的存档。使用fprintf函数将商品链表中的信息保存至文件中，销毁链表，并退出程序。如图所示：
  2. 根据商品价格进行排序：定义一个函数，将商品按照价格高低进行排序。使用冒泡排序法，当前者的价格高于后者时，交换两者数据域。如图所示：
  3. 删除所有信息：定义一个函数，实现所有信息的删除。销毁链表节点，删除文件，并退出程序。如图所示：

1. **实验器材（设备、元器件）：**

个人电脑一台

1. **实验步骤：**
2. 分别完成模板中注释为//TO DO YOUR WORK的函数内容
3. 定义并实现函数void info\_init(GoodsList \*\*head)。head表示头指针。用该函数初始化一个链表，从txt文档中读取商品的信息并保存到链表中。录入的时候每一条信息用malloc函数分配内存给一个节点，将该节点的节点指针指向该内存单元。最后打印商品的数目。当商品数目为零时，head为空。
4. 定义并完成函数 void info\_output\_all(GoodsList \*L)，\*L是链表头节点指针。用格式化的方式将商品链表中的每条信息都打印到屏幕上。
5. 定义并完成函数void info\_change(GoodList \*\*L)，完成商品的信息修改。\*\*L表示链表头节点指针的地址，用户输入需要修改的id或名称，然后对名称或ID 进行查找，找到则继提示用户修改信息；未找到则提示无此商品。
6. 定义并完成函数void info\_insert(GoodsList \*\*L)，完成某个商品信息的插入。\*\*L表示链表头节点指针的地址，用户可选择头插，尾插，中间插三种插发。先插入分配的动态内存存储信息，再将该内存的指针加入到链表中。
7. 定义并实现函数void info\_dele(GoodsList \*\*L)，完成某个商品信息的删除。\*\*L代表链表头节点指针的地址。用户输入要删除的商品id或名称，如果找到该商品则删除该商品的信息，询问用户是否确定删除，若确定，提示删除成功；如果未找到则提示该商品不存在。
8. 定义并实现函数 void info\_search(GoodsList \*L)，完成某个商品信息的查找。\*L表示头节点指针。用户可通过输入id或名称来查找商品信息。如果存在则打印该商品信息，否则则提示该商品不存在。
9. 定义并实现函数void info\_flush(GoodsList \*\*L)，完成对商品信息的存档和退出程序。将链表重新写入文件中，并销毁商品链表。
10. 定义并实现函数 void info\_bubble\_sort(GoodsList \*\*L)，完成对商品价格的从低到高排序。使用冒泡排序法进行排序。交换链表指针的数据域。
11. 定义并实现函数 void info\_del\_all(GoodsList \*\*L)，完成对文件和对链表的销毁。询问用户是否确定删除，若确定，则先将文件移除，再销毁链表。
12. 实现程序的入口函数即main函数，通过一个条件为1的while循环完成以上功能的循环调用，直至选择正常退出。
13. 编译、调试程序直至达到实验要求。
14. **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. 打印单个商品链表节点的信息

printf("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");

printf("ID:%s\t\tName:%s\tPrice:%d\tDiscount:%s\t Number:%d\tRemain:%d\n",(p->data).id,(p->data).name,(p->data).price, (p->data).discount, (p->data).amount, (p->data).remain);

printf("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");

|  |
| --- |
| 代码功能：打印单个商品的节点信息。  算法正确性：格式化打印节点信息，其中用（p->data）.成员名 表示商品的id、名称、价格、折扣、数量、剩余信息。  结果：打印出表示某个商品的id、名称、价格、折扣、数量、剩余信息。 |

2. 释放链表内存

while (p)

{

q = p;

p = q->next;

free(q);

q = NULL;

}

\*L = NULL;

CurrentCnt = 0;

|  |
| --- |
| 代码功能：实现对商品所有链表的销毁。  算法正确性：让q等于p，再让p指向下一个节点，释放q的内存，让q为空。循环上述结构，直到p为空。最后使头指针为空，并令商品数量为0。  结果：能够销毁链表。 |

3. 输入某个节点数据

GoodsList\* info\_goodchange(GoodsList \*p)

{

printf("\nPlease enter the new information: ");

printf("\nID: ");

scanf("%s", (p->data).GoodsInfos\_id);

printf("Name: ");

scanf("%s", (p->data).GoodsInfos\_name);

printf("Price: ");

scanf("%d", &((p->data).GoodsInfos\_price));

printf("Discount: ");

scanf("%s", (p->data).GoodsInfos\_discount);

printf("Amount: ");

scanf("%d", &((p->data).GoodsInfos\_amount));

printf("Remain: ");

scanf("%d", &((p->data).GoodsInfos\_remain));

return p;

}

|  |
| --- |
| 代码功能：实现节点数据的录入。  算法正确性：p表示某个节点的指针。用格式化的方式输入p->data.成员名 的信息，并返回p。  结果：能够将用户输入的信息保存到链表中。 |

4. 查找某个商品的节点

while (p)

{

if (strcmp(GoodsInfo\_search, (p->data).GoodsInfos\_id) == 0)

{

GoodsInfosprint(p); //打印商品节点的信息

break;

}

p = p->next;

}

|  |
| --- |
| 代码功能：查找某个商品的节点。  算法正确性：从头指针开始查找，每查找一个向p指向一个节点。使用字符串的比较来确定是否查找到信息。  结果：能够查找到用户输入的信息在链表中的位置。 |

5. 对商品价格进行排序

GoodsList \*p, \*pre;

pre = (GoodsList \*)malloc(sizeof(GoodsList)); //交换的媒介

for (int i = 1; i < CurrrentCnt; ++i)

{

p = \*L;

for (int j = 1; j < CurrrentCnt - i + 1; ++j)

{

if ((p->data). price > (p->next->data). price)

{

pre->data = p->data;

p->data = p->next->data;

p->next->data = pre->data;

}

p = p->next;

}

}

|  |
| --- |
| 代码功能：将商品价格进行排序。  算法正确性：内层循环结束后又从头开始进行排序，价格高的往后调整。  结果：能够达到对商品价格进行排序的功能。 |

6. 无商品时提醒输入商品信息：

If(\*head == NULL)

{

GoodsList \*L;

L = \*head;

L = (GoodsList \*)malloc(sizeof(GoodsList));

L = GoodsInfoChange(L); //输入信息的函数

L->next = NULL;

CurrentCnt++;

\*head = L;

}

|  |
| --- |
| 代码功能：当无商品时提醒输入信息。  算法正确性：令L等于头指针，为L分配内存，输入信息，再令L->next为空指针，增加商品的数量，并使头指针head=L。  结果：能够达到增加一条商品信息的功能。 |

7.询问用户是否要确定删除/插入商品：

char sure;

scanf(" %c", &sure);

while (getchar() != '\n') /\* skips to end of line \*/

;

if (sure == 'n' || sure == 'N')

return false; //输入n或N时退出

else if (sure == 'y' || sure == 'Y')

return true;

else

{

printf("Unvalidated Input!");

return false;

}

|  |
| --- |
| 代码功能：判断是否要确认删除/插入商品。  算法正确性：提示用户输入y或n，如果输入N或n则返回false，输入y或Y返回true。  结果：能够完成对用户的询问并返回值。 |

8. 读入文件并初始化链表：

while(!feof(fp))

{

p = (GoodsList \*)malloc(sizeof(GoodsList));

if (\*head == NULL) //头节点

\*head = p;

else //后续结构

pre->next = p;

p->next = NULL;

pre = p;

fscanf(fp,"%s%s%d%s%d%d\t",(p->data).GoodsInfos\_id,(p->data).GoodsInfos\_name,&(p->data).GoodsInfos\_price,&(p->data).GoodsInfos\_discount,&(p->data).GoodsInfos\_amount,&(p->data).GoodsInfos\_remain);

++CurrentCnt;

if (CurrentCnt > MAX)

{

printf("Too Many Commedies!Cannot load!");//商品信息过多

exit(-1);

}

}

|  |
| --- |
| 代码功能：初始化链表结构。  算法正确性：为p分配内存，当头指针为空的时候，使头指针=p，否则使pre->next = p。使pre = p，将文件中的信息以行读取到p中，商品数量增加。重复上述操作，当读取到文件尾的时候结束循环。当商品数量大于最大储存数量时，结束程序。  结果：能够完成对链表的初始化。 |

9. 插入节点:

if (code == 0) //头插法

{

p->next = \*L;

\*L = p;

}

|  |
| --- |
| 代码功能：插入头节点。  算法正确性：使插入的节点p->next = 以前的头指针。再另新的头指针=p。如图所示：  L->next  L  P  结果：能够将用户输入的数据存储到头节点。 |

else if (code == 1) //尾插法

{

while (pre->next)

pre = pre->next;

pre->next = p;

p->next = NULL;

}

|  |
| --- |
| 代码功能：插入尾节点。  算法正确性：使链表最后一个节点的->next指向插入的节点p，再让p的下一个节点为空。如图所示：  NULL  Pre  P  结果：能够将用户输入的数据存储到尾节点。 |

else //插入到中间的位置

{

int i = 1;

while (i < code - 1)

{

pre = pre->next;

++i;

}

p->next = pre->next;

pre->next = p;

}

|  |
| --- |
| 代码功能：将节点插入中间。  算法正确性：使链表最后一个节点的->next指向插入的节点p，再让p的下一个节点为空。  如图所示：    p  p->next    p  p->next  q  结果：能够将某个节点插入中间 |

10. 应用结果：

分析：该程序能够将商品的文件存入链表，通过对链表的修改来改变商品的信息，并将链表存入商品的文件。

正确与否：能够实现一个小型的超市商品管理系统，成功运行。

是否完成功能：1.该系统具备商品信息录入、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（包括头插尾插中间插）、商品价格排序、删除商品文件、保存商品文件这几个功能。2.根据需要加入了情感化的功能:询问用户是否确定删除，是否确定插入。

1. **总结及心得体会：**
2. 总结： 本次实验以动态单链表为基础，从单链表的定义和使用方法出发，掌握单链表的建立方法 ，节点的查找与删除 ，输出单链表节点的方法，链表节点排序的一种方法，文件的读写操作。
3. 心得体会：

1).通过此次实验，学到了很多知识。

a.加深了对单链表的理解，使用方法。

b.加深了对冒泡排序法的理解。巩固了数组中冒泡排序法的使用，也拓展了冒泡排序法在链表中的使用；

c.知道了一些细枝末节的小知识，例如c = getchar()中为什么c被自动赋值了，有可能是因为前一个回车被c吃了。

2).通过此次实验，明白了一些道理。

a.在绝望的情况下不放弃。第一次写10kb的代码，极大地锻炼了我的心理承受能力，即使是第一次运行报错40多个，错误越改越复杂，改了错误再改bug，即使是改到麻木，也要相信胜利就在眼前；

b.拿着模板的时候先去分析整个架构和逻辑，而不是先关注该怎么写。刚开始看模板我就开始定义变量的，后来越来越不懂到底怎么回事，后来又重新去看了链表的知识，再去分析模板，就迎刃而解了。

c.培养自己解决问题的习惯。当遇到自己不知道该如何修改的错误或bug，首先自己思考该如何解决。这样才能加深对代码程序的理解。

d.利用好身边的资源。在自己解决不了的问题上，先去找搜索引擎，搜索引擎找不到，再去问老师和资深的同学。

1. **对本实验过程及方法、手段的改进建议：**

1.实现商品排序这个功能中，可以让用户来选择排序的方式，例如按照id，名称，折扣排序。

2.希望能和这学期学到过的更多的知识串联起来。例如枚举、快排、数组。

3.可以增加对操作者的记录，用户访问的记录。

**报告评分：**

**指导教师签字：**