电子科技大学信息与软件工程学院

实 验 报 告

学　　号 20180916180008

姓　　名 袁昊男

(实验)　课程名称 C 语言程序设计

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：袁昊男　　学号：2018091618008　指导教师：白忠建

实验地点：基础实验大楼 实验时间：2019.1.1

一、 实验名称：超市商品管理系统链表实现

二、 实验学时：4 学时

三、 实验目的：

1. 掌握单链表的定义和使用方法
2. 掌握单链表的建立方法
3. 掌握单链表中结点的查找与删除
4. 掌握输出单链表结点的方法
5. 掌握链表结点排序的一种方法
6. 掌握 C 语言创建菜单的方法
7. 掌握结构体的定义和使用方法

四、 实验原理：

1. **结构体：**结构是可能具有不同类型的值(成员)的集合，结构的元素(在 C 语言中的说法是结构的成员)可能具有不同的类型。而且每个结构成员都有名字，所以在选择特定的结构成员时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：**当需要存储相关数据项的集合时，结构是一种合乎逻辑的选择。例如：假设需要记录存储在仓库中的零件，用来存储每种零件的信息可能包括零件的编号(整数)、零件的名称(字符串)以及现有零件的数量。为了产生一个可以存储全部三种数据项的变量，可以使用类似下面这样的声明：结构 part1 和 part2 中的成员 number 和成员 name 不会与结构 employee1 和 employee2 中的成员 number 和成员 name 冲突。

```
1. struct{
2. int number;
3. char name[NAME_LEN+1];
4. int on_hand;
5. } part1, part2;
```

每个结构变量都有三个成员：number(零件的编号)、name(零件的名称)和 on_hand(现有数量)。注意这里的声明格式和 C 语言中其他变量的声明格式

一样，struct{...}指明了类型，而 part1 和 part2 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。

3. **每个结构代表一种新的作用域：**任何声明中此作用域内的名字都不会和程序中的其他名字冲突。(用 C 语言的术语可以表述为：每个结构都为它的成员设置了独立的名字空间(name space))例如，下来声明可以出现在同一程序中：

```
struct{
int number;
char name[NAME_LEN+1];
int on_hand;
}part1, part2;
struct{
int number;
char name[NAME_LEN+1];
char sex;
} employee1, employee2;
```

4. **结构变量的初始化：**和数组一样，结构变量也可以在声明的同时进行初始化。为了对结构进行初始化，要把待存储到结构中的值的列表准备好，并用花括号把它括起来：

```
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。结构初始化式遵循的原则类似于数组初始化式的原则，用于结构初始化式的表达式必须是常量。也就是说，不能用变量来初始化结构 part1 的成员 on_hand。

5. **对结构的操作：**既然最常用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。例如，下列语句显示结构 part1 的成员的值：

```
printf("Part number: %d\n", part1.number);
printf("Part name: %s\n", part1.name);
printf("Quantity on hand: %d\n", part1.on_hand);
```

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：

```
Part1.number = 258;
Part1.on_hand++;
```

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：

```
part2 = part1;
```

这一句的效果是把 part1.number 复制到 part2.number，把 part1.name 复制到 part2.name，依次类推。

6. **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：

```
part  part1, part2; //错误声明
```

因为 part 不是类型名，如果没有单词 struct 的话，它没有任何意义。

7. **结构类型的定义：**除了声明结构标记，还可以用 typedef 来定义真实的类型名。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 后。可以像内置类型那样使用 Part。例如，可以用它声明变量：

```
Part  part1, part2;
```

8. **链表存储结构：**在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系则是由指针域来确定的。链式存储方式既可以用于表示线性结构，也可用于表示非线性结构。一般来说，在线性表的链式存储结构中，各数据结点的存储符号是不连续的，并且各结点在存储空间中的位置关系与逻辑关系也不一致。对于线性链表，可以从头指针开始，沿各结点的指针遍历链表中的所有结点。
9. **链表结点的类型：**链表结点的类型应该包括存储元素的数据域，一般用 data 表示，它的类型可以是前期定义的元素结构体，以及包括存储后续结点位置

的指针域，一般用 next 表示，最后一个结点的 next 指针域为空。例如下面定义一个商品的链表结点：

```
typedef struct node
{
    GoodsInfo data;
    struct node *next;
} GoodsList;
```

10. **单链表的头结点：**线性链表的链接存储中，为了方便在表头插入和删除结点的操作，经常在表头结点(*存储第一个元素的结点*)的前面增加一个结点，称之为头结点或表头附加结点。这样原来的表头指针由指向第一个元素的结点改为指向头结点，头结点的数据域为空，头结点的指针域指向第一个元素的结点。

11. **链表的基本操作函数：**
    a) `InitList(&L)`：链表初始化，构造一个空的链表 L，主要工作为 L 头结点分配空间。例如：

    ```
    GoodsList* pGoodsList = (GoodsList*)malloc(sizeof(GoodsList));
    ```

    b) `DestroyList(&L)`：销毁链表 L，需要链表 L 存在的情况下执行。主要工作为释放 L 每个结点的空间。
    c) `ListEmpty(L)`：判断 L 是否是空表，返回 TRUE 或者 FALSE。
    d) `ListLength(L)`：在 L 存在的情况下，计算 L 的结点数。
    e) `GetElem(L,i)`：在 L 存在的情况下，且 i 值在链表结点数范围内，返回链表第 i 个元素的值。
    f) `ClearList(&L)`：在链表 L 存在的情况下将 L 重置为空表。
    g) `ListInsert(&L,i,e)`:在 L 存在的情况下，且 `1<=i<=ListLength(L)`，在 L 第 i 个元素之前插入新的元素 e。
    h) `ListDelete(&L,i)`：在 L 存在的情况下，且 `1<=i<=ListLength(L)`，删除 L 的第 i 个元素。

12. **系统选择界面：**可以通过一个 while 条件为 1 的循环来完成程序功能的循环调用，直至输入退出系统的命令。例如：

```
while(1)
{
switch(choice) {
case 1: Fucntion1;
break; ...
} }
```

13. **文件操作示例：**

```
char tempstring[30];
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，另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的文件名；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);
```

五、 实验内容：

用 C 语言下的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、商品信息的插入、商品信息删除、商品信息查找、商品价格排序等功能。具体实现步骤如下：

1) **软件界面控制：**实现一个带密码登录功能(默认为 123)的数字选项式的主菜单启动界面，其中包含显示所有商品信息、修改商品信息、插入商品信息、删除商品信息、查找商品信息、商品价格排序、删除所有内容、不保存退出系统 8 个功能，并且这些功能可以循环调用。

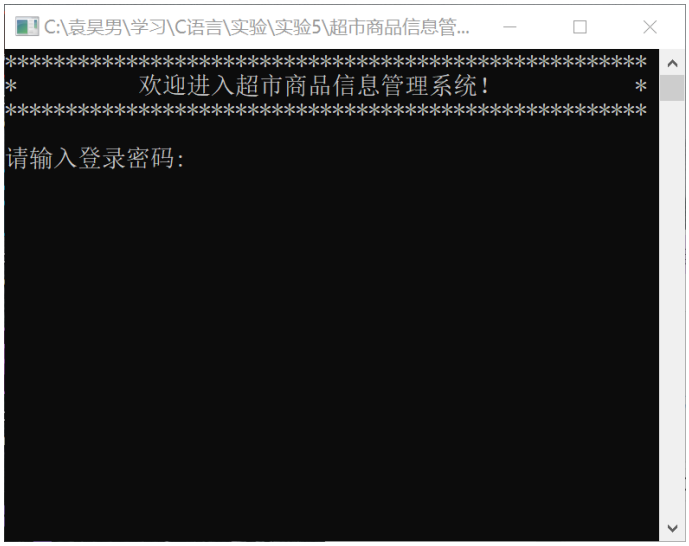

图 5.1.1 密码登录界面

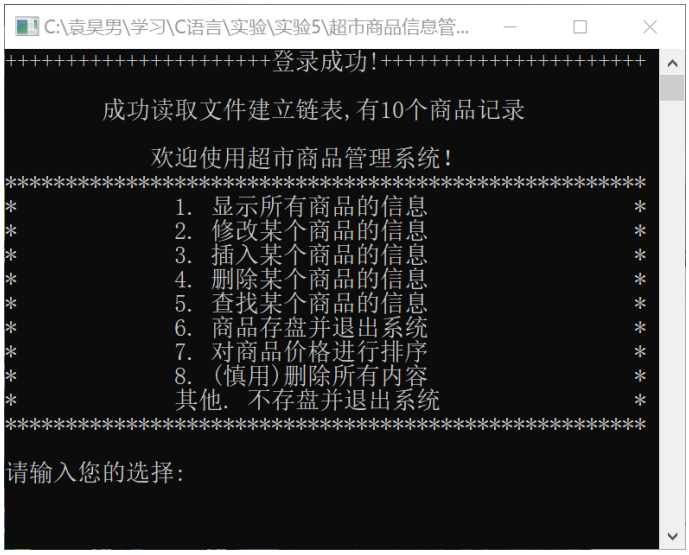

图 5.1.2　主菜单

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；若商品信息文件不存在则新建空白商品信息文件并打开。如图 5.1.2 所示，从文件中读取了 10 个商品记录。选择数字【1】可显示所有的商品信息。

图 5.2.1　显示所有商品的信息

3) **商品信息的修改：**实现一个函数，完成商品信息的修改。可以根据商品的 ID 修改商品信息(按商品的名称来修改更符合逻辑，此为编写代码时的疏忽)，并可选择是否将修改保存至文件。其中用字符串比较函数来查找待修改商品。

图 5.3.1　修改某个商品的信息

4) **商品信息的增加：**实现一个函数，完成单个商品信息的增加。可接受用户输入新增商品的各项信息，然后保存至链表结点。录入完成后自动显示更新后的商品目录，并可选择是否将修改保存至文件。目前只实现在指定商品之后插入新的商品信息，而在指定商品之前插入新的商品信息功能还未实现。

图 5.4.1　插入某个商品的信息

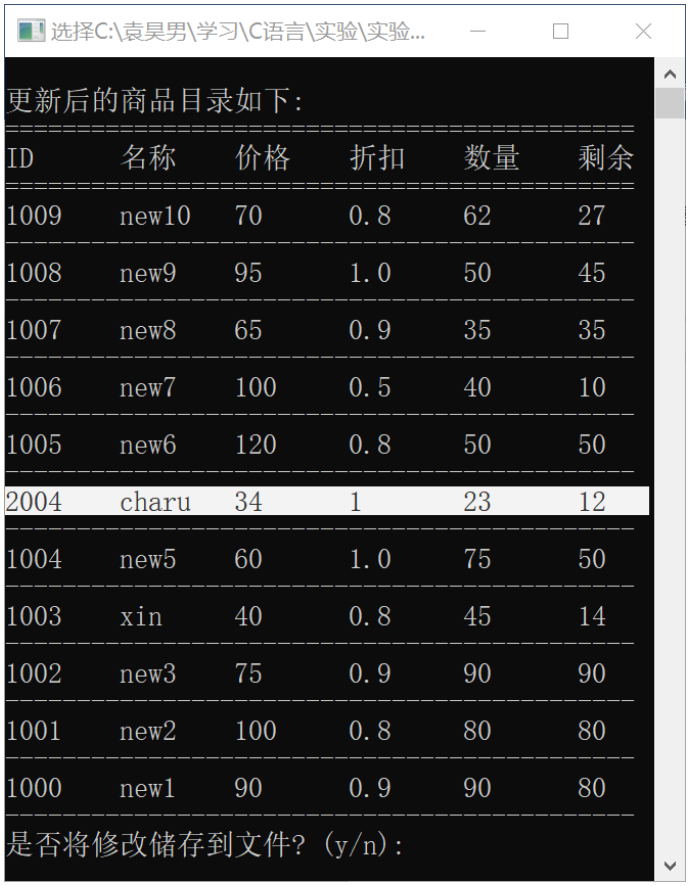

图 5.4.2　更新后的商品目录(图中高亮为新增信息)

5) **商品信息的删除：**实现一个函数，根据商品的 ID 来删除对应的商品信息(按商品的名称来查找更符合逻辑，此为编写代码时的疏忽)。商品查找通过调用字符串比较函数实现，查找到目的商品后释放对应指针指向的内存区域，完成删除，并可选择是否将修改保存至文件。

图 5.5.1　删除某个商品的信息

6) **商品信息的查找：**实现一个函数，根据输入的商品 ID 来查找对应的商品信息(按商品的名称来查找更符合逻辑，此为编写代码时的疏忽)。商品 ID 的判断调用字符串比较函数来实现，然后调用 `Output()`函数格式化打印出查找到的商品信息。

图 5.6.1　查找某个商品的信息

7) **对商品价格排序：**实现一个函数，可以根据链表中的商品的价格，对商品进行排序(升序)。排序算法采用冒泡排序实现，最后将排序后的链表打印至屏幕。

图 5.7.1　对商品价格进行排序

8) **删除所有内容：**实现一个函数，可以清空链表、删除所有结点并释放内存。调用 `free()`函数实现。

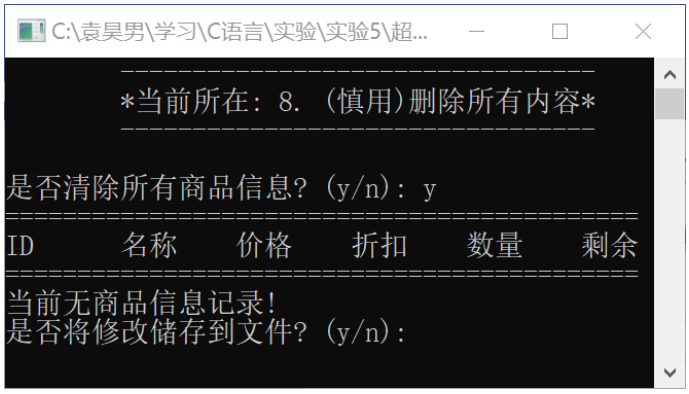

图 5.8.1　删除所有内容

9)　不存盘并退出系统：不将商品信息的修改保存到文件并退出系统。

六、　实验器材(设备、元器件)：

个人电脑一台

七、　实验步骤：

1) 分析所要编写的超市商品管理系统的各功能需求，确定系统的基本框架和编写思路；
2) 参照代码模板，完成结构体及头结点的定义、登录函数 Login()、操作界面函数 operaion_interface()、空文件检查函数 check_nullfile()、文件读入函数 Read()、写文件函数 Write()、打印链表函数 Output()、搜索结点函数 Search()、修改结点函数 Change()、插入结点函数 Cut_in()、删除结点函数 Delete()、结点冒泡排序函数 Bubble()、清空链表函数 Clear()、主菜单函数 Menu()；
3) 在 Menu()函数中调用各功能函数，在 main()函数中调用 Login()、Read()、Menu()函数以达到各功能循环调用的目的；
4) 调试软件，修改错误，完善功能以达到实验目的要求；
5) 测试软件各功能是否正常；
6) 实验结果分析；

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.　从文件读入商品信息并建立链表【Read()函数】

```
1. void Read(Headptr l) {
2.     check_nullfile();
3.     l->head.next = NULL;
4.     l->Goodsn = 0;
5.     FILE *fp;
```

```
    fp = fopen("goodsinfo.txt", "r");
    if(fp == NULL) {
        fp == fopen("goodsinfo.txt", "w");
        printf("商品信息初始化文件不存在！程序将为您新建一个.\n");
    }
    else {
        GoodsListptr p;
        p = (GoodsListptr)malloc(sizeof(GoodsList));

        while(!feof(fp)) {
            fscanf(fp, "%s", p->data.id);
            fscanf(fp, "%s", p->data.name);
            fscanf(fp, "%d", &p->data.price);
            fscanf(fp, "%s", p->data.discount);
            fscanf(fp, "%d", &p->data.amount);
            fscanf(fp, "%d", &p->data.remain);
            p->next = l->head.next;
            l->head.next = p;
            l->Goodsn++;
            p = (GoodsListptr)malloc(sizeof(GoodsList));
        }
        n = l->Goodsn;
    }
}
```

代码功能分析：本段代码编写的 Read()函数实现的功能是从文件中读入商品信息并将其按照顺序保存至新建的链表结点中。首先调用了模板中的函数 check_nullfile()来检查商品文件是否存在；形式参数 l 的类型是 Head-ptr(一个指向带有头结点及商品数量 Goodsn 结构体变量的指针)，即 l 是头结点，并将头结点的下一个结点定义为 NULL，商品数量 Goodsn 初始化为 0；然后调用文件操作函数打开商品信息文件，定义一个 GoodsListptr 类型的指针 p，使用 malloc()动态内存分配函数为 p 在内存中开辟一个空间，接着使用 while 语句，在文件没有读完时执行循环体：以 fscanf()格式化读入商品数据，并保存至相应的结构中；然后新建结点，通过 l->head.next = p; l->head.next = p; 两条语句使得上一结点和下一结点链接上，完成后 Goodsn 自加一次，并再次为新的结点分配内存、循环，直至文件读完(粗略过程如图 8.1.1 所示)。

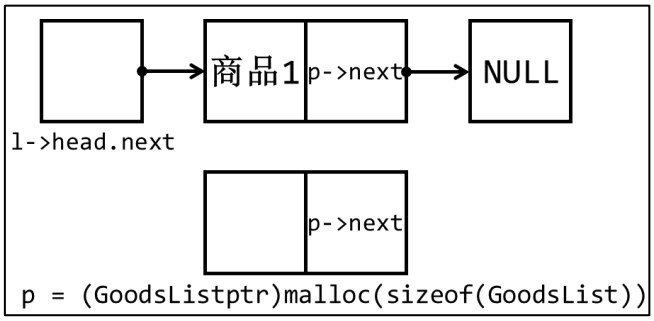

图 8.1.1　链接结点

算法正确性分析：该算法经检查、测试在本身功能上正确。

结果分析：该函数能够完成目的功能。当商品信息文件存在时能够正确读取商品文件中的内容；当商品信息文件不存在时可以打印出错提示信息并新建立商品信息文件。

2. 打印单个商品链表结点的信息【Output()函数】

```
1. void Output(Headptr l) {
2.  Goo
3. dsListptr prt = l->head.next;
4.  printf("============================================\n");
5.  printf("ID\t 名称\t 价格\t 折扣\t 数量\t 剩余\n");
6.  printf("============================================\n");
7.     while (prt) {
8.     printf("%s\t%s\t%d\t%s\t%d\t%d\n", prt->data.id, prt->da
   ta.name, prt->data.price, prt->data.dis-
   count, prt->data.amount, prt->data.remain);
9.   printf("--------------------------------------\n");
10.  prt = prt->next;
11.    }
12.}
```

代码功能分析：本段代码编写的 Output()函数实现的功能是将商品信息链表的每一个结点打印到屏幕上。首先定义了一个指针类型为 GoodsListptr 的变量 prt(print 缩写)，并使它指向商品链表的头结点的下一结点(因为头结点没有储存商品信息)；prt 存在的意义在于作为一个“第三指针”来遍历原链表中的所有结点而不改动其中的数据信息；之后使用 while 语句，当 prt 所指向的对象不为 NULL 时执行循环体，依次打印出商品结点的 id、name、price、discount、amount 及 remain 六条信息，并通过语句 prt = prt->next；使得“第三指针”prt 后移一个结点，遍历链表直至末尾，此时链表中存储的所有结点信息便打印到了屏幕上。

算法正确性分析：该算法经检查、测试在本身功能上正确。

结果分析：该函数能够完成目的功能，但在实际测试时发现打印的商品信息与实际商品信息文件中的商品顺序恰好相反。经检查，发现该处错误出现在文件读入函数 Read()上：在读入信息、新建结点、将信息存入结点中时，我

采用的是“头插法”插入结点，这就导致了第二个结点插入到了第一个结点之前(**以头结点的下一结点为第一个结点，下同**)，即第二个商品信息插入到了第一个商品信息之前；在调用 Output()函数打印结点时，prt 指针从第一个结点开始遍历，而经过 Read()函数的第一个结点实际上储存的是最后一个商品的信息。要解决此问题，有两种解决办法：一是在插入结点时采用“尾插法”；二是在打印商品信息时逆序打印。

3. 商品链表结点信息写入文件【Write()函数】

```
1. void Write(Headptr l) {
2.     FILE *fp;
3.     fp = fopen("goodsinfo.txt", "w");
4.     if(fp == NULL) {
5.         fp == fopen("goodsinfo.txt", "w");
6.         printf("商品信息初始化文件不存在！程序将为您新建一
   个.\n");
7.     }
8.     else {
9.         GoodsListptr p;
10.        p = l->head.next;
11.        while (p) {
12.            fprintf(fp, "%s\t", p->data.id);
13.            fprintf(fp, "%s\t", p->data.name);
14.            fprintf(fp, "%d\t", p->data.price);
15.            fprintf(fp, "%s\t", p->data.discount);
16.            fprintf(fp, "%d\t", p->data.amount);
17.            fprintf(fp, "%d", p->data.remain);
18.            p = p->next;
19.            if (p != NULL) fprintf(fp, "\n");
20.        }
21.    }
22.    fclose(fp);
23.    return;
24.}
```

代码功能分析：本段代码编写的 Write()函数实现的功能是将商品链表结点信息写入文件。同 Read()函数，使用文件操作函数打开商品信息文件，准备对其进行更改；定义一个 GoodsListptr 类型的指针 p，并让其指向商品链表的第一个结点；使用 while 语句，当 p 所指向的结点不为 NULL 时，执行循环体：使用 fprintf()函数格式化写入商品信息，注意每写完同一个商品的一项信息后加注\t 转义序列符使其与原文件中的商品信息格式相同；最后一项信息不必加注\t，而是先将指针 p 指向下一个结点后判断下一个结点是否为 NULL。若不为 NULL，说明还没有读到最后一个商品信息结点，则写入一个\n 换行符；若 p 指向的下一个结点为 NULL，则不必加注\n 换行符(若

加了换行符则会导致商品信息文件末尾出现一个空行，导致再次读入数据时出现一条乱码的商品数据)。

算法正确性分析：该算法经检查、测试在本身功能上正确。

结果分析：该函数能够完成目的功能。在第一次测试时，没有判断是否为最后一个结点并添加\n 这一步，导致了再次测试 Read()函数时出现空白行。

4. 搜索商品信息【Search()函数】

```
1. GoodsListptr Search(Headptr l) {
2.     GoodsListptr sch = l->head.next;
3.     printf("请输入要查找的商品 ID: ");
4.     char chaid[10];
5.     scanf("%s", &chaid);
6.     while(sch) {
7.     if(strcmp(sch->data.id, chaid) == 0) {
8.     printf("\n");
9.    printf("查询到 ID 为%s 的商品信息: \n", chaid);
10.   printf("=============================================\n");
11.   printf("ID\t 名称\t 价格\t 折扣\t 数量\t 剩余\n");
12.   printf("=============================================\n");
13.   printf("%s\t%s\t%d\t%s\t%d\t%d\n", sch->data.id, sch->dat
   a.name, sch->data.price, sch->data.dis-
   count, sch->data.amount, sch->data.remain);
14.   printf("---------------------------------------\n");
15.           return sch;
16.       }
17.       sch = sch->next;
18.    }
19.    printf("你要查询的商品不存在!\n");
20.    return NULL;
21.}
```

代码功能分析：本段代码编写的 Search()函数实现的功能是在链表中搜索结点。首先定义一个 GoodsListptr 类型的指针 sch(search 缩写)，并使其指向第一个结点；定义一个字符串数组 chaid，用 scanf()函数读入用户输入待查询的商品 ID；接着使用 while 循环语句，当 sch 指针不为 NULL 时，执行循环体：由于储存的商品 ID 数据类型是字符串，因此想要比较两个字符串需要用到字符串比较函数 strcmp()；若输入的 ID 和 sch 指针指向的某一结点的 id 项相同时(strcmp()函数会返回 0 值)显示提示信息并打印该商品信息；若比较不符，则 sch = sch->next; 使 sch 指针指向后一个结点，直至遍历整个链表。若未找到相符的商品 ID，显示提示信息。

算法正确性分析：该算法经检查、测试在本身功能上正确。但是其查找策略有些不符合常理：当用户想要查找一个商品信息应是按商品名称来查找，而不是该商品在数据库中的编号 ID。想要改进此问题也十分容易：将代码中所有涉及到 id 的地方全部替换为 name 即可。

结果分析：该函数能够完成目的功能。

5. **修改商品信息【Change()函数】**

```
1. void Change(Headptr l) {
2.     GoodsListptr sch = l->head.next;
3.     GoodsListptr found;
4.     //省略和 Search()函数相同部分代码
5.         if (found == NULL)
6.             printf("你要查询的商品不存在!\n");
7.             printf("*** 现在正在对 ID 为%s 的商品信息进行修
   改 ***\n", chaid);
8.             printf("\n");
9.             printf("ID: %s\n", chaid);
10.            printf("名称: ");
11.            scanf("%s", found->data.name);
12.            printf("价格: ");
13.            scanf("%d", &found->data.price);
14.            printf("折扣: ");
15.            scanf("%s", found->data.discount);
16.            printf("数量: ");
17.            scanf("%d", &found->data.amount);
18.            printf("剩余: ");
19.            scanf("%d", &found->data.remain);
20.            printf("\n");
21.  printf("商品信息修改完成! 修改后的商品信息如下: \n");
22.  printf("=============================================\n");
23.  printf("ID\t 名称\t 价格\t 折扣\t 数量\t 剩余\n");
24.  printf("=============================================\n");
25.  printf("%s\t%s\t%d\t%s\t%d\t%d\n", found->data.id, found->
   data.name, found->data.price, found->data.dis-
   count, found->data.amount, found->data.remain);
26.  printf("---------------------------------------------\n");
27.            sch = found;
28.            printf("是否将修改储存到文件? (y/n): ");
29.            char yn;
30.            scanf("%s", &yn);
31.            switch (yn) {
32.            case 'y': Write(l); printf("\n"); printf("信息已
   保存! \n"); return;
33.                case 'n': exit(1);
34.            }
35.}
```

代码功能分析：本段代码编写的 Change()函数实现的功能是在链表中查找到指定商品信息后对该商品信息进行修改。首先定义一个 GoodsListptr 类型的指针 sch(search 缩写)和 found，使得 found 指向查找到的结点 sch；如果 found 指向 NULL，说明要查找的商品信息不存在；否则，通过 printf()和 scanf()函数对 found 所指结点的商品信息进行修改，修改完成后将

found 新结点再重新链回原来的 sch 中，并提示用户是否将改动保存到文件；若用户选择 y，则调用 Write()函数写入新信息到文件；反之退出程序；

算法正确性分析：该算法经检查、测试在本身功能上正确。查找商品中存在的逻辑问题同上。

结果分析：该函数能够完成目的功能。

6. 插入商品信息【Cut_in()函数】

```
1. void Cut_in(Headptr l) {
2.     GoodsListptr sch = l->head.next;
3.     GoodsListptr temp;
4.     //省略和 Search()函数相同部分代码
5.     GoodsListptr pNew;
6.     pNew = (GoodsListptr)malloc(sizeof(GoodsList));
7.     pNew->next = temp->next;
8.     temp->next = pNew;
9.     printf("\n");
10.    printf("* 现在正在 ID 为%s 的商品之后插入一个新商
   品 *\n", chaid);
11.    printf("ID: ");
12.    scanf("%s", pNew->data.id);
13.    //省略部分信息录入代码
14.    printf("\n");
15.    system("cls");
16.    printf("商品信息录入完成! 录入的商品信息如下: \n");
17.    printf("=============================================\n")
   ;
18.    printf("ID\t 名称\t 价格\t 折扣\t 数量\t 剩余\n");
19.    printf("=============================================\n")
   ;
20.    printf("%s\t%s\t%d\t%s\t%d\t%d\n", pNew->data.id, pNew->
   data.name, pNew->data.price, pNew->data.dis-
   count, pNew->data.amount, pNew->data.remain);
21.    printf("---------------------------------------------
   \n");
22.    printf("\n");
23.    printf("更新后的商品目录如下: \n");
24.    Output(l);
25.    printf("是否将修改储存到文件? (y/n): ");
26.    //省略保存到文件部分代码
27.}
```

代码功能分析：本段代码编写的 Cut_in()函数实现的功能是在链表中查找到指定商品信息后在此商品后插入一个新的商品信息。首先定义一个 GoodsListptr 类型的指针 sch(search 缩写)和 temp，使得 temp 指向查找到的结点 sch；再定义一个 GoodsListptr 类型的指针 pNew；然后新建结点并将结点链接到链表中：pNew->next = temp->next; temp->next =

pNew;(粗略过程如图 8.6.1 所示)，之后再用 printf()和 scanf()函数录入新增商品信息保存至新增结点中。

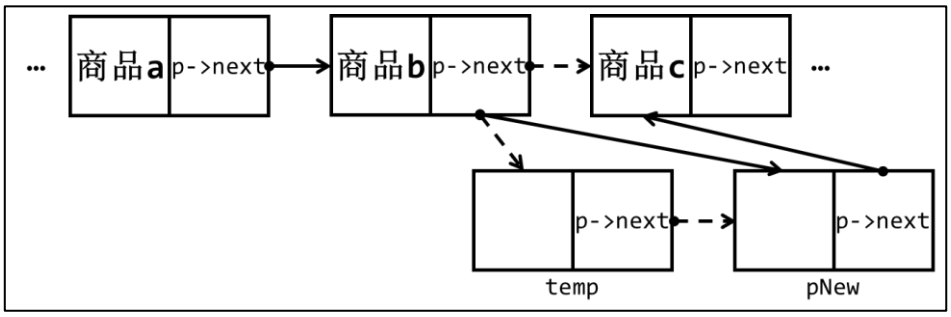

图 8.6.1　插入新结点

算法正确性分析：该算法经检查、测试在本身功能上大体完整，但目前只实现在指定商品之后插入新的商品信息，而在指定商品之前插入新的商品信息功能还未完善。查找商品中存在的逻辑问题同上。

结果分析：该函数能够完成目的功能(后插新商品信息)。

7. 删除商品信息【Delete()函数】

```
1. void Delete(Headptr l) {
2.     GoodsListptr sch = l->head.next;
3.     GoodsListptr temp;
4.     //省略和 Search()函数相同部分
5.     printf("确认删除 ID 为%s 的商品? (y/n): ", chaid);
6.     char yn;
7.     scanf("%s", &yn);
8.     switch (yn) {
9.     case 'y': {
10.    system("cls");
11.    GoodsListptr pNew;
12.    pNew = (GoodsListptr)malloc(sizeof(GoodsList));
13.    pNew = temp->next;
14.    temp->data = pNew->data;
15.    temp->next = pNew->next;
16.    printf("商品信息已删除！");
17.    printf("更新后的商品目录如下: \n");
18.    Output(l);
19.    //省略保存到文件部分代码
20.}
```

代码功能分析：本段代码编写的 Delete()函数实现的功能是在链表中查找到指定商品信息后删除该商品信息。首先定义一个 GoodsListptr 类型的指针 sch(search 缩写)和 temp，使得 temp 指向查找到的结点 sch；再定义一个 GoodsListptr 类型的指针 pNew；使用 malloc()动态内存分配函数为 pNew 在内存中开辟一个空间，然后执行 pNew = temp->next; temp->data = pNew->data; temp->next = pNew->next; 三条语句，使得选定结点的商品数据被后一个结点的数据所覆盖，然后将选定结点的下一个结点链接

到再下一个结点(粗略过程如图 8.7.1 所示)，从而达到依次“删除结点”的效果；之后打印出修改后的商品列表并保存到文件。

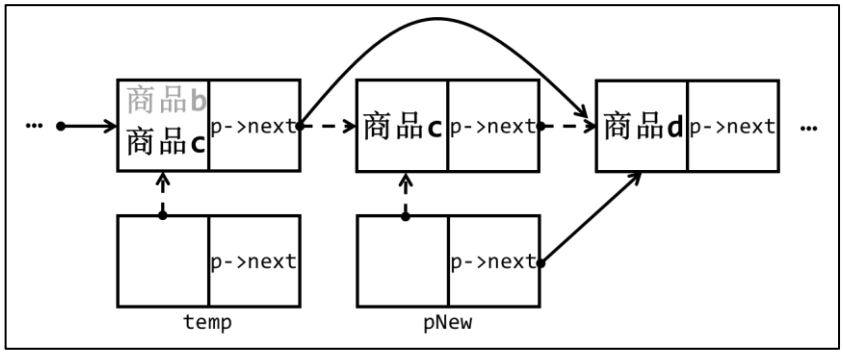

图 8.7.1　删除结点

算法正确性分析：该算法经检查、测试在本身功能上大体完整。但对于头结点和尾结点的删除还没有更深入的讨论分析下去。查找商品中存在的逻辑问题同上。

结果分析：该函数能够完成目的功能(删除除头结点、尾结点外的其他任意结点)。

8.　对商品按价格排序【Bubble()函数】

```
1. GoodsListptr Bubble(Headptr l) {
2. GoodsListptr p, q;
3. int num = 0;
4. int x = 0;
5. int i = 0;
6. q = l->head.next;
7. while (q != NULL) {
8.       q = q->next;
9.       num++;
10.     }
11.for (i = 0; i < num - 1; i++) {
12.      p = q = l->head.next;
13.      x = num - i - 1;
14.     while (p->next != NULL && x != 0) {
15.         x--;
16.         if ((p->data.price) > (p->next->data.price)) {
17.             if (p == l->head.next) l->head.next = p->next;
18.             else q->next = p->next;
19.             q->next = p->next;
20.             q = q->next;
21.             p->next = q->next;
22.             q->next = p;
23.             p = q;
24.             }
25.             q = p;
26.             p = p->next;
27.         }
```

```
    }
    printf("对所有商品按价格升序排序如下: \n");
    Output(l->head.next);
    printf("是否将修改储存到文件? (y/n): ");
            char yn;
            scanf("%s", &yn);
            switch (yn) {
                case 'y': Write(l); printf("\n"); printf("信息已保存! \n"); return;
                case 'n': exit(1);
            }
}
```

代码功能分析：本段代码编写的 Bubble()函数实现的功能是对链表中的所有商品按价格进行升序排序。此函数实质上是对价格套用冒泡排序算法。首先定义了两个 GoodsListptr 的变量 p 和 q，令 q 指向链表的第一个结点，使用 while 语句使 q 遍历链表结点，从而达到计数的目的，商品信息个数储存在 int 型变量 num 中；然后通过嵌套的内外两层循环完成相邻结点 price 数据的比较、交换，最终实现通过冒泡排序对商品价格进行升序排序。这里“交换”指的是交换两个结点的商品信息，而不是真正意义上的交换两个结点。

算法正确性分析：该算法经检查、测试在本身功能上大体完整。但目前仅实现了对商品价格按升序排序，不能选择降序排序；其次，该段代码某些地方比较冗杂，有改进的地方。

结果分析：该函数能够完成目的功能。

9. 删除所有内容【Clear()函数】

```
void Clear(Headptr l) {
    printf("是否清除所有商品信息? (y/n): ");
    char yn;
    scanf("%s", &yn);
    switch (yn) {
    case 'y': {
    Headptr temp = NULL;
    if (l->head.next = NULL) {
    printf("数据已清空! ");
        return ;
    }
    while (l->head.next != NULL) {
    temp = l->head.next;
    free(l->head.next);
    l->head.next = temp;
    }
    if (l->head.next != NULL) {
    free(l->head.next);
    l->head.next = NULL;
```

```
    }
    Output(l);
    printf("当前无商品信息记录! \n");
    printf("是否将修改储存到文件? (y/n): ");
    //省略保存到文件部分代码
}
```

代码功能分析：本段代码编写的 Clear()函数实现的功能是清空链表的所有内容。在用户选择 y 进入清空功能后，定义了一个 Headptr 类型的变量 temp，初始化为 NULL；如果第一个结点已经为 NULL，则提示“数据已清空”；否则，使用 while 语句，当第一个结点不为 NULL 时，执行循环体：让 temp 指向第一个结点，紧接着调用 free()函数释放第一个结点，再执行 l->head.next = temp; 语句，则现在的第一个结点实质上是原来的第二个结点，直至最后一个结点被释放；最后释放仍存在的头结点和第一个结点，完成整个链表结点的删除。

算法正确性分析：该算法经检查、测试在本身功能上正确。

结果分析：该函数能够完成目的功能。

10. 附：结构体及头结点定义、Menu()函数、main()函数部分代码：

- 结构体及头结点定义：

```
typedef struct node {
    GoodsInfo data;
    struct node *next;
} GoodsList;

typedef GoodsList *GoodsListptr;

typedef struct {
    GoodsList head;
    int Goodsn;
} Head;

typedef Head *Headptr;
```

- Menu()函数：

```
void Menu() {
    Headptr l;
    Read(l);
    printf("        成功读取文件建立链表,有%d 个商品记录        \n", n);
    operation_interface();
    int choice;
    printf("\n");
    printf("请输入您的选择: ");
    scanf("%d", &choice);
    switch (choice) {
        case 1: {
```

```
12.         system("cls");
13.         printf("\t--------------------------------
    \n");
14.         printf("\t*当前所在：1. 显示所有商品的信息*\n");
15.         printf("\t--------------------------------
    \n");
16.         printf("\n");
17.         Output(l);
18.         printf("回到主菜单？(y/n)：");
19.         char yn;
20.         scanf("%s", &yn);
21.         switch (yn) {
22.         case 'y': system("cls"); Head-
    ptr l; Read(l); Menu();
23.         case 'n': exit(1);
24.      }
25.     }
26.         case 2: //略
27.         case 3: //略
28.         case 4: //略
29.         case 5: //略
30.         case 6: //略
31.         case 7: //略
32.         case 8: //略
33.         default: exit(1);
34.     }
35.}
```

九、 总结及心得体会：

这是我学习 C 语言一学期以来所编写的第一个比较大型的程序，也可以算作对一学期所学知识的复习、应用和总结。由于对指针、结构体、链表和文件读写操作学习时间较短，对相关知识及应用不熟悉，一开始就卡在从文件中读入数据保存至链表结点这个初始步骤上。毫不夸张地说，那几晚做的梦都是关于链表。从开始分析系统需求到软件编写完成，我推翻重写了三次、改了五个版本，请教了老师、同学，在 CSDN 论坛上查阅了很多相关资料，最终才得以踉踉跄跄地编写出一个功能大体上完善的链表版超市商品管理系统。这六百多行的代码中，必然还存在许多错误和值得优化的地方，但限于能力和精力，未能一一完善，还请多多包涵。

通过这个软件的编写，能快速、深入地让 C 语言初学者掌握有关指针、结构体、链表和文件读写操作的应用，了解相关数据结构的知识和指针高级应用的特点；其次，还能让 C 语言初学者了解、熟悉编写一个完整程序的方法和步骤，为以后大型程序的编写打下基础，实践性非常强，有重要的学习指导意义；同时在

编写的过程中，我深切地体会到链表和指针的抽象性、编程对思维的训练以及成功时的喜悦。

在此，再次感谢在编程过程中给予过我帮助的白老师和几位同学！

十、 对本实验过程及方法、手段的改进建议：

本实验在给出的资源包中的模板源文件编写得十分规范到位，给出了所有需要用到的函数，只需要将空白处补充完整即可。这个模板文件有利也有弊：好处在于能帮同学理清编写的思路，有了准确的切入点，易于下手；不太好的地方在于其固定死了头结点和结构体的定义以及大体的框架。我的前两版文件就是直接补充源文件的空白内容，但在没有彻底弄清楚模板的编写思路和框架的情况下，一开始有点懵，反而不知道怎么下手。在推翻重写时，我没有沿用模板中的头结点和结构体定义以及所需函数，而是将模板内容作为参照自己编写所有的内容。

综上所述，我的改进建议是可以不给出模板文件，而是将编写思路或者系统框架以流程图等形式以 PPT 或实验指导书的形式给出；其次，老师可以适当讲解编写思路及流程。以上是我的个人看法，不代表所有人的观点，适当参考即可。

报告评分：

指导教师签字：