

电子科技大学

实验报告

学生姓名: Lolipop 学号: 2018091202000 指导教师: XXXXX

实验地点: XXXXXX 实验时间: XX.XX.XX

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. 掌握单链表中节点的查找与删除
4. 掌握输出单链表节点的方法
5. 掌握链表节点排序的一种方法
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: 结构体(struct)是由一系列具有相同类型或不同类型的数据构成的数据集合, 叫做结构。结构的成员(member)可以具有不同的类型, 而且每个结构成员都应有名字。成员一般用名字访问。
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。比如, 对于仓库的零件管理系统而言, 存储每种零件的信息可能包括零件的编号(int, 整型数)、零件的名称(char, 字符串)以及现有零件的数量(int, 整型数)等。
3. 每个结构代表一种新的作用域。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;
}STORE_1,STORE_2;
struct{
```

```

        int number;
        char name[NAME_LEN+1];
        char sex;
    }KILLER_1,KILLER_2;

```

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};

```

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。

5. 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，首先写出结构的名称，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 的成员的值得。结构的成员是左值，所以它们可以出现在复制运算的左侧，也可以作为自增或自减表达式的操作数。

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称。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` 来定义真正的类型名。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。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：链表（Linked list）是一种常见的基础数据结构，是一种线性表，但是并不会按线性的顺序存储数据，而是在每一个节点里存到下一个节点的指针（Pointer）。链表最明显的好处就是，常规数组排列关联项目的方式可能不同于这些数据项目在记忆体或磁盘上顺序，数据的访问往往要在不同的排列顺序中转换。而链表是一种自我指示数据类型，因为它包含指向另一个相同类型的数据的指针（链接）。链表允许插入和移除表上任意位置上的节点，但是不允许随机存取。链表有很多种不同的类型：单向链表，双向链表以及循环链表。本实验采用单向链表。
一个单向链表的节点被分成两个部分。第一个部分保存或者显示关于节点的信息，第二个部分存储下一个节点的地址。单向链表只可向一个方向遍历。
链表最基本的结构是在每个节点保存数据和到下一个节点的地址，在最后一个节点保存一个特殊的结束标记，另外在一个固定的位置保存指向第一个节点的指针，有的时候也会同时储存指向最后一个节点的指针。一般查找一个节点的时候需要从第一个节点开始每次访问下一个节点，一直访问到需要的位置。但是也可以提前把一个节点的位置另外保存起来，然后直接访问。当然如果只是访问数据就没必要了，不如在链表上储存指向实际数据的指针。这样一般是为了访问链表中的下一个或者前

一个节点。

9. 链表节点的类型：链表节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，以及包括存储后续节点位置的指针域，此处用 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节点：

```
struct goods {  
    int number;  
    int price;  
    int on_hand;  
    struct goods *next;  
};
```

10. 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：

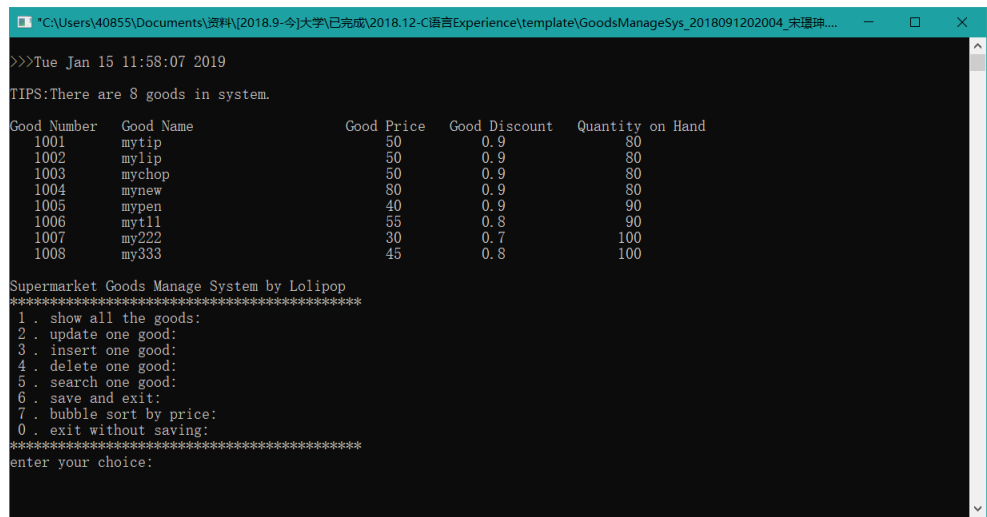
```
while(1)  
{  
    switch(choice){  
        case 1:Fucntion1;  
        break; ...  
    }  
}
```

当然，也可以使用 for(;;)来实现循环功能。

五、 实验内容：

用 C 语言和单链表数据结构实现一个简单的超市商品管理系统，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、商品信息的插入等功能。具体实现步骤如下：

- 1) 系统界面控制：实现一个数字选项式的启动界面，其中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、商品信息查询、按照商品价格排序、退出超市管理系统并保存以及退出系统但不保存这 8 个选项。并且这些功能可以循环调用。如图 5-1.



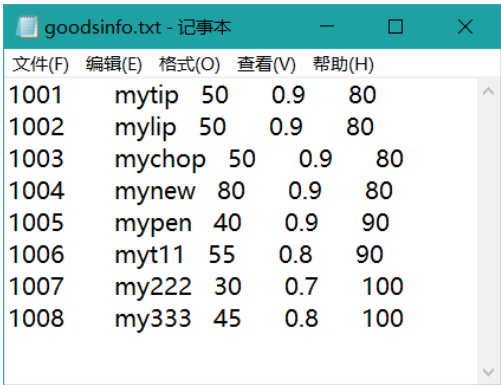
```
>>>Tue Jan 15 11:58:07 2019
TIPS:There are 8 goods in system.

Good Number   Good Name      Good Price   Good Discount  Quantity on Hand
1001          mytip          50           0.9            80
1002          mylip          50           0.9            80
1003          mychop         50           0.9            80
1004          mynew          80           0.9            80
1005          mypen          40           0.9            90
1006          myt11          55           0.8            90
1007          my222          30           0.7            100
1008          my333          45           0.8            100

Supermarket Goods Manage System by Lolipop
*****
1. show all the goods:
2. update one good:
3. insert one good:
4. delete one good:
5. search one good:
6. save and exit:
7. bubble sort by price:
0. exit without saving:
*****
enter your choice:
```

图 5-1

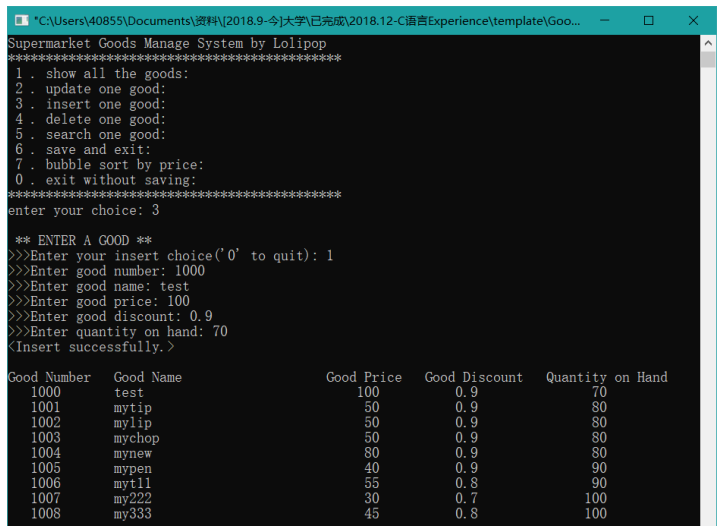
- 2) 商品信息的初始化：读取并初始化 txt 文件中的链表。实现从已有的商品信息文件中读入商品信息，并且分配内存保存至链表中。如 1) 里的图示所示，从文件中读取了 7 个商品记录。读取的文件可参见图 5-2.



```
goodsinfo.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1001    mytip    50    0.9    80
1002    mylip    50    0.9    80
1003    mychop   50    0.9    80
1004    mynew    80    0.9    80
1005    mypen    40    0.9    90
1006    myt11    55    0.8    90
1007    my222    30    0.7    100
1008    my333    45    0.8    100
```

图 5-2

- 3) 商品信息的增加：完成单个商品信息的增加，接受用户的输入的各项信息，然后保存至链表结点。同时实现可以根据用户的输入，将该结点插入到列表的任意位置——如输入“1”，则插入到链表的第一项。如图 5-3，在链表的首端添加了名为“test”的商品。



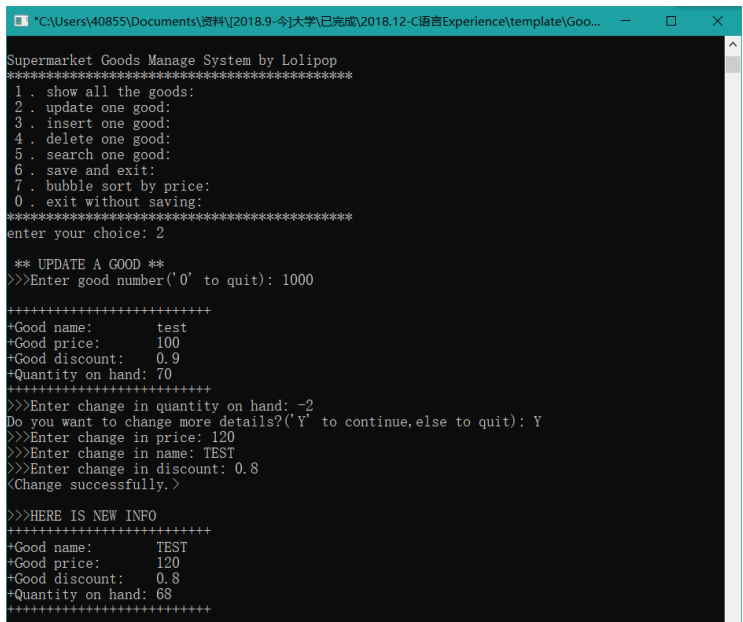
```
Supermarket Goods Manage System by Lolipop
*****
1. show all the goods:
2. update one good:
3. insert one good:
4. delete one good:
5. search one good:
6. save and exit:
7. bubble sort by price:
0. exit without saving:
*****
enter your choice: 3

** ENTER A GOOD **
>>>Enter your insert choice('0' to quit): 1
>>>Enter good number: 1000
>>>Enter good name: test
>>>Enter good price: 100
>>>Enter good discount: 0.9
>>>Enter quantity on hand: 70
<Insert successfully.>

Good Number    Good Name          Good Price    Good Discount    Quantity on Hand
1000           test                100           0.9              70
1001           mytip              50            0.9              80
1002           mytip              50            0.9              80
1003           mychop             50            0.9              80
1004           mynew              80            0.9              80
1005           mypen              40            0.9              90
1006           mytll              55            0.8              90
1007           my222              30            0.7              100
1008           my333              45            0.8              100
```

图 5-3

- 4) 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 ID（即 **number**，下同）修改商品信息。基本的商品信息修改为数量的修改，“-”和“+”后面加上数量来完成修改。进一步可选择将所有内容都进行修改并显示最后修改的结果。如图 5-4.



```
Supermarket Goods Manage System by Lolipop
*****
1. show all the goods:
2. update one good:
3. insert one good:
4. delete one good:
5. search one good:
6. save and exit:
7. bubble sort by price:
0. exit without saving:
*****
enter your choice: 2

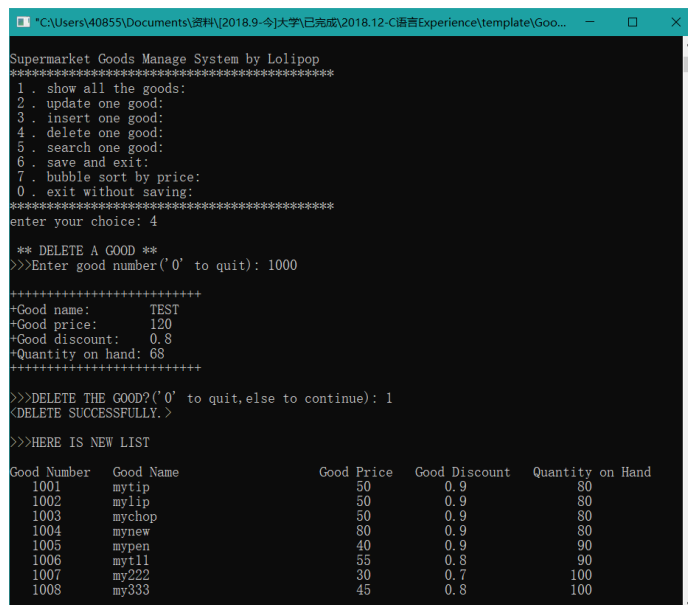
** UPDATE A GOOD **
>>>Enter good number('0' to quit): 1000

+-----+
+Good name:      test
+Good price:     100
+Good discount:  0.9
+Quantity on hand: 70
+-----+
>>>Enter change in quantity on hand: -2
Do you want to change more details?('Y' to continue, else to quit): Y
>>>Enter change in price: 120
>>>Enter change in name: TEST
>>>Enter change in discount: 0.8
<Change successfully.>

>>>HERE IS NEW INFO
+-----+
+Good name:      TEST
+Good price:     120
+Good discount:  0.8
+Quantity on hand: 68
+-----+
```

图 5-4

- 5) 商品信息的删除：实现根据商品的 ID 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。如图 5-5。



```
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 4

** DELETE A GOOD **
>>>Enter good number('0' to quit): 1000

*****
+Good name:      TEST
+Good price:     120
+Good discount:  0.8
+Quantity on hand: 68
*****

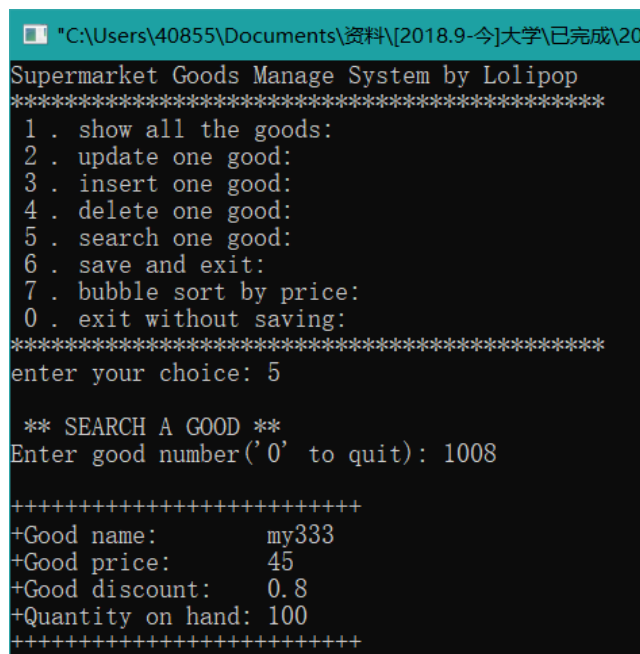
>>>DELETE THE GOOD?('0' to quit,else to continue): 1
<DELETE SUCCESSFULLY.>

>>>HERE IS NEW LIST

Good Number   Good Name      Good Price   Good Discount   Quantity on Hand
-----
1001          mytip          50           0.9             80
1002          mylip          50           0.9             80
1003          mychop         50           0.9             80
1004          mynew          80           0.9             80
1005          mypen          40           0.9             90
1006          myt11          55           0.8             90
1007          my222          30           0.7             100
1008          my333          45           0.8             100
```

图 5-5

- 6) 商品信息的查找：实现一个函数，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，然后显示查找到的商品信息。如图 5-6。



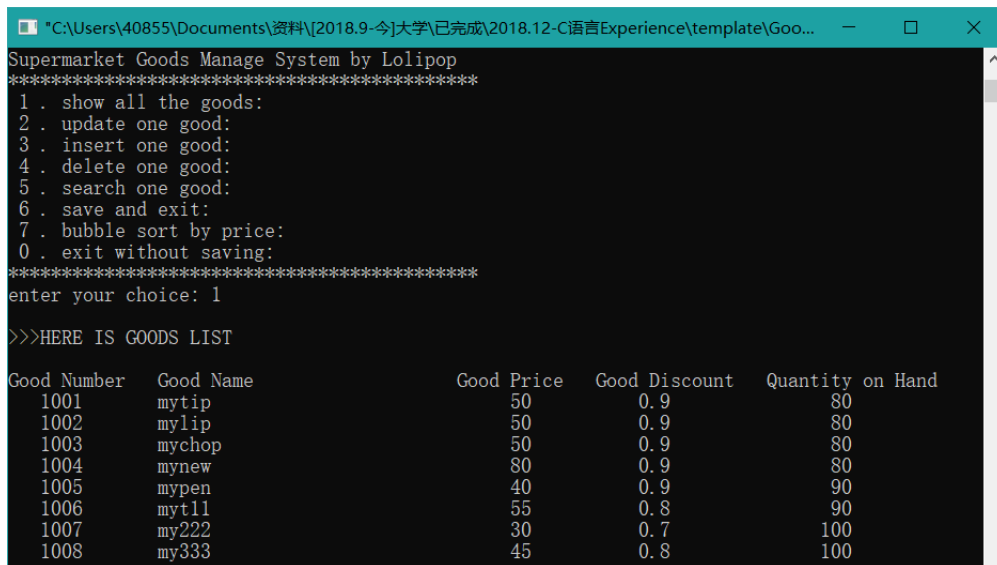
```
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 5

** SEARCH A GOOD **
Enter good number('0' to quit): 1008

*****
+Good name:      my333
+Good price:     45
+Good discount:  0.8
+Quantity on hand: 100
*****
```

图 5-6

- 7) 所有商品信息的显示：实现一个函数，该函数的功能是将链表中所有的商品信息以格式化的方式打印输出到屏幕上。如图 5-7.



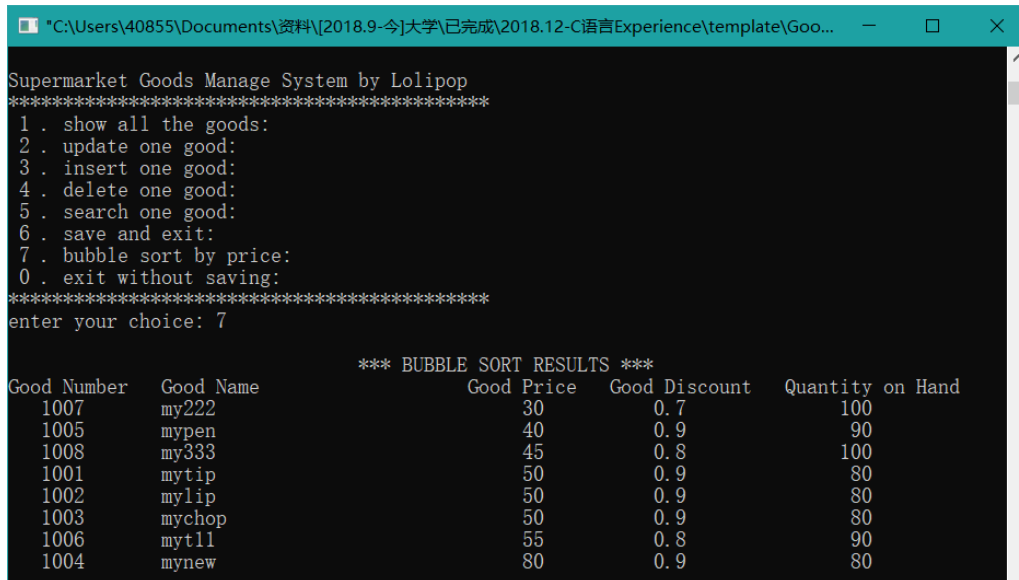
```
"C:\Users\40855\Documents\资料\[2018.9-今]大学\已完成\2018.12-C语言Experience\template\Goo...
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 1

>>>HERE IS GOODS LIST

Good Number    Good Name          Good Price    Good Discount    Quantity on Hand
1001           mytip              50            0.9              80
1002           mylip              50            0.9              80
1003           mychop             50            0.9              80
1004           mynew              80            0.9              80
1005           mypen              40            0.9              90
1006           myt11              55            0.8              90
1007           my222              30            0.7              100
1008           my333              45            0.8              100
```

图 5-7

- 8) 对商品按价格排序：实现一个函数，可以根据链表中的商品的价格，对商品进行冒泡排序，然后将排序后的结果（如图 5-8）打印至屏幕。



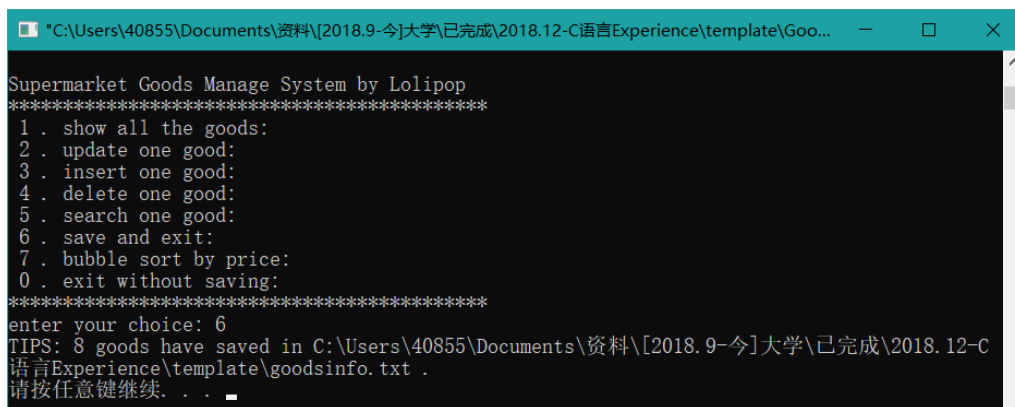
```
"C:\Users\40855\Documents\资料\[2018.9-今]大学\已完成\2018.12-C语言Experience\template\Goo...
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 7

*** BUBBLE SORT RESULTS ***

Good Number    Good Name          Good Price    Good Discount    Quantity on Hand
1007           my222              30            0.7              100
1005           mypen              40            0.9              90
1008           my333              45            0.8              100
1001           mytip              50            0.9              80
1002           mylip              50            0.9              80
1003           mychop             50            0.9              80
1006           myt11              55            0.8              90
1004           mynew              80            0.9              80
```

图 5-8

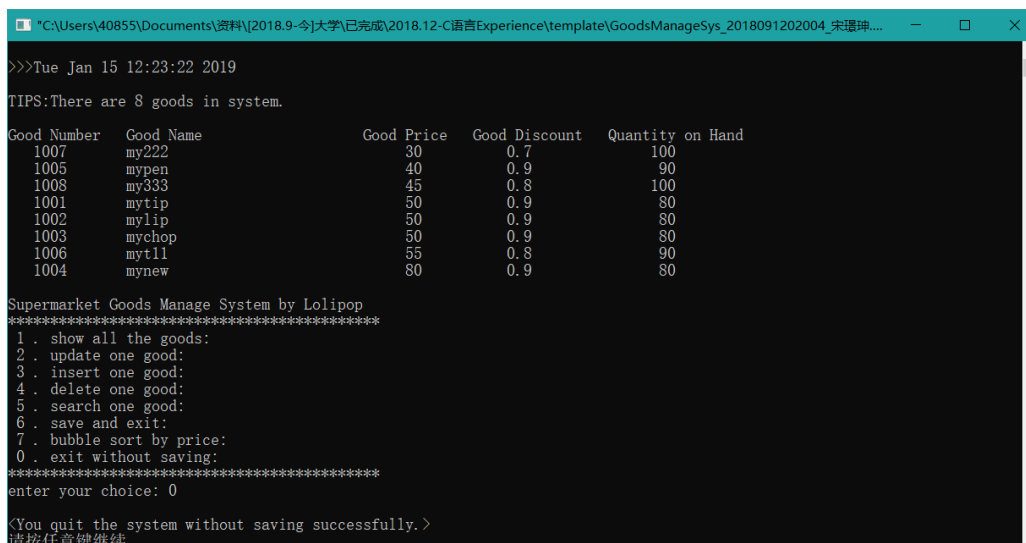
- 9) 退出系统并保存：实现一个函数首先读取信息，并将新的信息写入到储存商品信息的文件中，然后清理系统运行过程中已分配的内存。最后退出系统，完成交互。如图 5-9。



```
"C:\Users\40855\Documents\资料\[2018.9-今]大学\已完成\2018.12-C语言Experience\template\Goo...
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 6
TIPS: 8 goods have saved in C:\Users\40855\Documents\资料\[2018.9-今]大学\已完成\2018.12-C
语言Experience\template\goodsinfo.txt .
请按任意键继续. . .
```

图 5-9

- 10) 退出系统但不保存：直接退出系统，完成交互。适用于商品系统的调试或者避免错误改动。如图 5-10。



```
"C:\Users\40855\Documents\资料\[2018.9-今]大学\已完成\2018.12-C语言Experience\template\GoodsManageSys_2018091202004_宋璟坤....
>>>Tue Jan 15 12:23:22 2019
TIPS:There are 8 goods in system.
Good Number   Good Name           Good Price   Good Discount   Quantity on Hand
1007          my222                30           0.7             100
1005          mypen                40           0.9             90
1008          my333                45           0.8             100
1001          mytip                50           0.9             80
1002          mylip                50           0.9             80
1003          mychop               50           0.9             80
1006          mytll                55           0.8             90
1004          mynew                80           0.9             80
Supermarket Goods Manage System by Lolipop
*****
1 . show all the goods:
2 . update one good:
3 . insert one good:
4 . delete one good:
5 . search one good:
6 . save and exit:
7 . bubble sort by price:
0 . exit without saving:
*****
enter your choice: 0
<You quit the system without saving successfully.>
请按任意键继续. . .
```

图 5-10

六、 实验器材（设备、元器件）：

电脑一台

七、 实验步骤：

- 1) 学习《C 语言程序设计现代方法》的相关章节内容。
- 2) 分别完成模板中注释为//TO DO YOUR WORK 的函数。

- 3) 定义用于表示某种商品的所有信息的结构体，并且定义链表结构用来组织所有的商品信息库中的商品信息。
- 4) 利用宏定义，定义 FILE_PLACE 为存放 goodsinfo.txt 的根目录，之后可以进行的随时修改。
- 5) 定义并实现一个函数：void readfile(char * filename)。该函数初始化一个链表，然后从指向的 txt 文件读入商品链表库并使用相应的商品信息来初始化超市的商品库，在读入的时候每读到一条商品信息就实时的动态分配内存来把信息放到分配得到的链表结点指针指向的内存单元中，然后把链表节点加入到商品链表中。
- 6) 定义并实现函数：void info_flush(char * filename)。该函数完成将系统运行期间改动过的商品信息库写回到存放商品信息的 txt 文件中，然后销毁商品链表。
- 7) 定义并实现函数：void print(void)。以格式化的方式，完成将读取的链表中的每项信息打印到屏幕上。
- 8) 定义并实现函数：int output(m)，格式化输出某一项商品信息，其中 m 是该商品的 ID，在 main 函数处由用户输入。
- 9) 定义并实现函数：struct goods *find_good(int number)。该函数根据输入的 ID 查找目标商品在链表中的位置，并返回该指针。是实现众多功能的核心。
- 10) 定义并实现函数：void update(void)。该函数完成商品信息的修改功能，用户通过输入需要修改的某项商品的 ID，然后对 ID 进行查找，找到则显示该商品信息，提示用户继续输入该商品的信息修改内容，完成后显示商品修改的结果并提示修改成功；如果没有找到则提示对应商品未找到并返回到初始菜单。
- 11) 定义并实现函数：int dele(int num)。其中 m 是该商品的 ID，在 main 函数处由用户输入。该函数实现删除链表中某一个商品的功能，通过输入的某项商品的 ID 删除对应的商品，如果在商品库中找到对应的商品，则显示商品的具体信息，并提示用户进行确认操作。如果确认删除，则删除该商品信息(即释放指针所指向的内存，并把该指针赋值为

NULL，同时完成链表的前后重新衔接)，并提示删除成功。如果不删除，则退出返回到系统主界面;如果没有找到该商品提示没有找到该商品信息，并返回主界面。

- 12) 定义并实现函数：void search(void)。该函数完成商品信息的查找功能，然后通过输入商品的 ID 来检索商品信息库，查找到则显示该商品的详细信息；没有查找到则提示没有该商品并返回主界面。
- 13) 定义并实现函数：int insert(int n)。该函数完成商品信息的插入，其中 n 是该商品插入的位置（如“1”为头部），在 main 函数处由用户输入。在插入之前动态的分配内存用来存储插入的商品信息，然后把指向该内存的指针加入到链表中。
- 14) 定义并实现函数：void bubblesort(void)。该函数利用冒泡排序，将链表中的所有商品按照商品价格从低到高排序。
- 15) 实现程序的入口函数即 main 函数，然后通过一个 for(;;) 循环完成以上功能的循环调用，直至选择退出（保存或不保存）。
- 16) 编译、调试程序直至达到实验要求。
- 17) 适当美化显示结果的内容，使其符合人的基本审美要求。

八、实验结果与分析

➤ 读取商品文件信息并建立链表

```
/*读取商品文件信息*/
void readfile(char * filename)
{
    FILE *fp;
    int count=0;
    struct goods *cur, *prev, *new_node;
    bool res = check_nullfile(filename);
    if ((fp = fopen(filename, "r")) == NULL)
    {
        if ((fp = fopen(filename, "w")) == NULL)
            printf("TIPS:Failed to create database file.\n");
    }
    else {

        while (res && !feof(fp))
        {
```

```

        new_node = malloc(sizeof(struct goods));
        if (new_node == NULL) {
            printf("TIPS:The system database is full and cannot continue
to add good.\n");
            return;
        }
        for (cur = inventory, prev = NULL;
            cur != NULL && new_node->number > cur->number;
            prev = cur, cur = cur->next)
            ;
        if (cur != NULL && new_node->number == cur->number) {
            printf("Good already exists.\n");
            free(new_node);
            return;
        }

        fscanf(fp, "%d", &new_node->number);
        fscanf(fp, "    %s", new_node->name);
        fscanf(fp, "    %d", &new_node->price);
        fscanf(fp, "    %s", new_node->goods_discount);
        fscanf(fp, "    %d\n", &new_node->on_hand);
        new_node->next = cur;
        if (prev == NULL)
            inventory = new_node;
        else
            prev->next = new_node;
        count++;
    }
    printf("TIPS:There are %d goods in system.\n",count);
}

fclose(fp);
}

/*检查空文件*/
bool check_nullfile(char* filename)
{
    FILE *fp = fopen(filename, "r");
    if (!fp) {
        printf("TIPS:Document does not exist. A new document has been
created for you!\n");
        FILE *fp = fopen(filename, "w");
        fclose(fp);
    }
}

```

```

        return false;
    }
    else {
        int temp;
        int res = fscanf(fp, "%d", &temp);
        fclose(fp);
        if (res <= 0)
            return false;
        else
            return true;
    }
}

/*打印链表*/
void goodslist_init(void)
{
    time_t t;
    time(&t);
    printf("\n>>>%s\n", ctime(&t));
    //调用时间函数，显示打开系统的时间（可选）

    readfile(FILE_PLACE);
    printf("\n");
    print();
}

```

➤ 将当前商品链表中的内容存入商品文件 **goodsinfo.txt**，保存后销毁链表

```

void save_to_file_exit(void)
{
    info_flush(FILE_PLACE);
    exit(0);
}

/*保存到文件中*/
Aavoid info_flush(char * filename){
    struct goods *cur, *prev;
    int savecount = 0;
    FILE *fp;
    if ((fp = fopen("goodsinfo.txt", "w")) == NULL)

```

```

{
    printf("TIPS:Reading data failed.\n");
    return;
}
for (cur = inventory, prev = NULL;
     cur != NULL;
     prev = cur, cur = cur->next)
{

    fprintf(fp, "%d", cur->number);
    fprintf(fp, "      %s", cur->name);
    fprintf(fp, "    %d", cur->price);
    fprintf(fp, "      %s", cur->goods_discount);
    fprintf(fp, "    %d\n", cur->on_hand);
    savecount++;

}
fclose(fp);
DestroyGoods(inventory);
printf("TIPS: %d goods have saved in %s .\n", savecount,filename);
}

```

➤ 输出所有商品信息

```

void print(void)
{
    struct goods *p;

    printf("Good Number    Good Name                                Good Price    Good
Discount    "
          "Quantity on Hand\n");
    for (p = inventory; p != NULL; p = p->next)
        printf("%7d        %-25s    %7d        %7s        %11d\n", p->number,
p->name,p->price,p->goods_discount,
            p->on_hand);
}

```

➤ 添加一条商品信息

```

int insert(int n)
{

```

```

struct goods *cur, *prev, *new_code;
int number, choice, i;

new_code = malloc(sizeof(struct goods));
if(new_code == NULL){
    printf("\n<Database is full; can't add more goods.>\n");
    free(new_code);
    return;
}

printf(">>>Enter good number: ");
scanf("%d", &new_code->number);

for (cur = inventory, prev = NULL;
     cur != NULL && new_code->number > cur->number;
     prev = cur, cur = cur->next)
    ;
if (cur != NULL && new_code->number == cur->number){
    printf("\n<Good already exists.>\n");
    free(new_code);
    return;
}

printf(">>>Enter good name: ");
read_line(new_code->name, MAX_NAME_LEN);
printf(">>>Enter good price: ");
scanf("%d", &new_code->price);
printf(">>>Enter good discount: ");
read_line(new_code->goods_discount, MAX_DISCOUNT_LEN);
printf(">>>Enter quantity on hand: ");
scanf("%d", &new_code->on_hand);

for (cur = inventory, prev = NULL, i = 1;
     cur != NULL && i <= n-1;
     prev = cur, cur = cur->next, i++)
    ;
new_code->next = cur;
if (n == 1){
    inventory = new_code;
}
else {
    prev->next = new_code;
}

```

```
    printf("<Insert successfully.>\n\n");
    print();
}
```

➤ 查找一条商品记录

```
void search(void)
{
    int number;
    struct goods *p;

    printf("\n ** SEARCH A GOOD ** \nEnter good number('0' to quit):
");
    scanf("%d",&number);
    if (number == 0){
        printf("\n<You quit successfully.>\n");
        return;
    }

    p = find_good(number);
    if (p != NULL){
        printf("\n+++++");
        printf("\n+Good name:      %s\n",p->name);
        printf("+Good price:      %d\n",p->price);
        printf("+Good discount:    %s\n",p->goods_discount);
        printf("+Quantity on hand: %d\n",p->on_hand);
        printf("+++++\n");
    }
    else {
        printf("\n<Good not found>\n");
    }
}
```

➤ 删除一条商品信息

```
int dele(int num)
{
    int i;
    struct goods *cur, *prev;
```



```

    if ((find_good(num)) == NULL){
        printf("\n<Good not found.>\n");
        return;
    }

    output(num);

    printf("\n>>>DELETE THE GOOD?('0' to quit,else to continue): ");
    scanf("%d",&i);
    if (i == 0){
        return;
    }

    for(cur = inventory, prev = NULL;
        cur != NULL;
        prev = cur, cur = cur->next){
        if (cur->number == num){
            break;
        }
    };

    if (prev == NULL){
        inventory = inventory->next;
        goto loop;
    }
    else
    prev->next = cur->next;

    loop:free(cur);
    printf("<DELETE SUCCESSFULLY.>\n\n>>>HERE IS NEW LIST\n\n");
    print();
    return;
}

```

➤ 修改一条商品信息

```

void update(void)
{
    int number, change_p, change_q;
    char change_n[MAX_NAME_LEN+1], change_d[MAX_DISCOUNT_LEN+1];
    char ch_1[2]="Y",ch_2[10]={NULL};
    struct goods *p;

```

```

    printf("\n ** UPDATE A GOOD ** \n>>>Enter good number('0' to quit):
");
    scanf("%d",&number);
    if (number == 0){
        printf("\n<You quit successfully.>\n");
        return;
    }

    p = find_good(number);
    if (p != NULL){
        output(number);
        printf(">>>Enter change in quantity on hand: ");
        scanf("%d",&change_q);
        p->on_hand += change_q;

        printf("Do you want to change more details?('Y' to continue,else
to quit): ");
        scanf("%s",ch_2);
        if (strcmp(ch_1,ch_2) == 0){
            goto loop_1;
        }
        else {
            goto loop_2;
        }

loop_1:printf(">>>Enter change in price: ");
scanf("%d",&change_p);
p->price = change_p;

printf(">>>Enter change in name: ");
scanf("%s", change_n);
strcpy(p->name,change_n);

printf(">>>Enter change in discount: ");
scanf("%s", change_d);
strcpy(p->goods_discount,change_d);

loop_2:
printf("<Change successfully.>\n\n>>>HERE IS NEW INFO");
output(number);
}
else {
    printf("\n<Good not found.>\n");

```

```
}  
}
```

➤ 释放链表内存

```
void DestroyGoods(struct goods* inventory) {  
    if (!inventory) return;  
  
    struct goods *cur,*prev;  
    for (cur = inventory, prev = NULL;  
        cur != NULL;  
        prev = cur, cur = cur->next,free(cur))  
        ;  
}
```

➤ 依照价格从小到大的冒泡排序

```
void bubblesort(void)  
{  
    bubblesort_before();  
  
    //冒泡排序  
    struct goods *p, *q, *tail, *h, *new_code;  
  
    tail = NULL;  
    h = inventory;  
    while(h->next != NULL)  
    {  
        p = inventory;  
        q = p->next;  
        while(q->next != NULL)  
        {  
            if(p->next->price > q->next->price)  
            {  
                p->next = q->next;  
                q->next = q->next->next;  
                p->next->next = q;  
                p = p->next;  
            }  
            else  
            {  
                ;  
            }  
        }  
    }  
}
```

```

        p = p->next;
        q = q->next;
    }
}
tail = q;
}

bubblesort_after();

printf("\n\t\t\t\t\t*** BUBBLE SORT RESULTS *** \n");
print();
}

/*在开头插入一个任意头*/
void bubblesort_before(void)    //辅助完成冒泡排序的函数
{
    //在开头创建一个空链表头
    struct goods *newcode;

    newcode = malloc(sizeof(struct goods));
    newcode->number = 2019;
    newcode->price = 0;
    newcode->on_hand = 18;
    strcpy(newcode->name, "Lolipop");
    strcpy(newcode->goods_discount, "0.9");

    newcode->next = inventory;
    inventory = newcode;
}

/*删除插入的头*/
void bubblesort_after(void)    //辅助完成冒泡排序的函数
{
    struct goods *cur_0;
    cur_0 = inventory;
    inventory = inventory->next;
    free(cur_0);
    //删除开头创建的链表头
}

```

➤ 其它相关代码

```
/*查找商品并返回指向该商品的指针*/
struct goods *find_good(int number)
{
    struct goods *p;

    for (p = inventory;
         p != NULL && number != p->number;
         p = p->next)
        ;
    if (p != NULL && number == p->number)
        return p;
    return NULL;
}

/*在进行删除等操作的时候，显现出目标商品信息（可选设计）*/
int output(m)
{
    struct goods *p;

    p = find_good(m);

    printf("\n++++++");
    printf("\n+Good name:      %s\n",p->name);
    printf("+Good price:      %d\n",p->price);
    printf("+Good discount:    %s\n",p->goods_discount);
    printf("+Quantity on hand: %d\n",p->on_hand);
    printf("++++++\n");
}

/*插入或修改结构体元素字符串字面量时调用*/
int read_line(char str[], int n)
{
    int ch, i = 0;

    while (isspace(ch = getchar()))
        ;
    while (ch != '\n' && ch != EOF) {
        if (i < n)
            str[i++] = ch;
        ch = getchar();
    }
}
```

```
    str[i] = '\\0';  
    return i;  
}
```

➤ 主界面实现

```
int main(void)  
{  
    int choice,number,cho;  
    goodslist_init();  
    printf("\\n");  
    for (;;) {  
        printf("Supermarket Goods Manage System by Lolipop\\n");  
        printf("*****\\n");  
        printf(" 1 . show all the goods:\\n");  
        printf(" 2 . update one good:\\n");  
        printf(" 3 . insert one good:\\n");  
        printf(" 4 . delete one good:\\n");  
        printf(" 5 . search one good:\\n");  
        printf(" 6 . save and exit:\\n");  
        printf(" 7 . bubble sort by price:\\n");  
        printf(" 0 . exit without saving:\\n");  
        printf("*****\\n");  
        printf("enter your choice: ");  
        scanf("%d", &choice);  
        while (getchar() != '\\n')  
            ;  
        switch (choice){  
            case 1: printf("\\n>>>HERE IS GOODS LIST\\n\\n");print(); break;  
            case 2: update(); break;  
            case 3:  
                printf("\\n ** ENTER A GOOD ** \\n>>>Enter your insert  
choice('\\0' to quit): ");  
                scanf("%d",&cho);  
                if (cho == 0){  
                    printf("\\n<You quit successfully.>\\n");  
                    break;}  
                insert(cho); break;  
            case 4:  
                printf("\\n ** DELETE A GOOD ** \\n>>>Enter good number('\\0' to  
quit): ");  
                scanf("%d",&number);
```

```
        if (number == 0){
            printf("\n<You quit successfully.>\n");
            break;
        }
        dele(number); break;
    case 5: search(); break;
    case 6: save_to_file_exit(); break;
    case 7: bubblesort(); break;
    case 0: printf("\n<You quit the system without saving
successfully.>\n");return 0;
        default: printf(">>>Wrong code. Please try again.\n"); break;
    }
    printf("\n");
}
}
```

报告评分：

指导教师签字：