**电子科技大学信息与软件工程学院**

**实 验 指 导 书**

**（实验）课程名称 C语言程序设计**

**电子科技大学教务处制表**

# 实验五 超市商品信息管理新系统（链表）

## 一、实验目的

1. 掌握单链表的定义和使用方法
2. 掌握链表的建立方法
3. 掌握链表中节点的查找与删除
4. 掌握输出链表节点的方法
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
8. 掌握字符串的定义和使用方法

## 二、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:当需要存储相关数据项的集合时，结构是一种呵护逻辑的选择。例如假设需要记录存储在仓库中的零件。用才存储每种零件的信息可能包括零件的编号(证书)、零件的名称(字符串)以及现在有零件的数量。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。

struct{  
int number;

char name[NAME\_LEN+1];

int on\_hand;

} 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。

1. 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;

1. 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。例如，不能用变量来初始化结构part1的成员on\_hand。

1. 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。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复制运算的左侧，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。

1. 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。

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。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;

1. 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是由指针域来确定的。链式存储方式即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](http://baike.soso.com/v105830.htm?ch=ch.bk.innerlink)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，沿各结点的指针扫描到链表中的所有结点。
2. 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：

typedef struct node

{

GoodInfo data;

struct node \*next;

}GoodList;

1. 单链表的头结点：线性表的链接存储中，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。
2. 链表的基本操作函数：
3. 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：

GoodList\* pGoodList = (GoodList\*)malloc(sizeof(GoodList));

1. 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的每个节点的空间。
2. ListEmpty(L):判断L是否是空表，返回TRUE或者FALSE
3. ListLength(L):在L存在的情况下，计算L的结点数。
4. 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。
5. ClearList(&L):在链表L存在的情况下将L重置为空表。
6. 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.
7. 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
8. 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; ...

} }

1. 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，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);

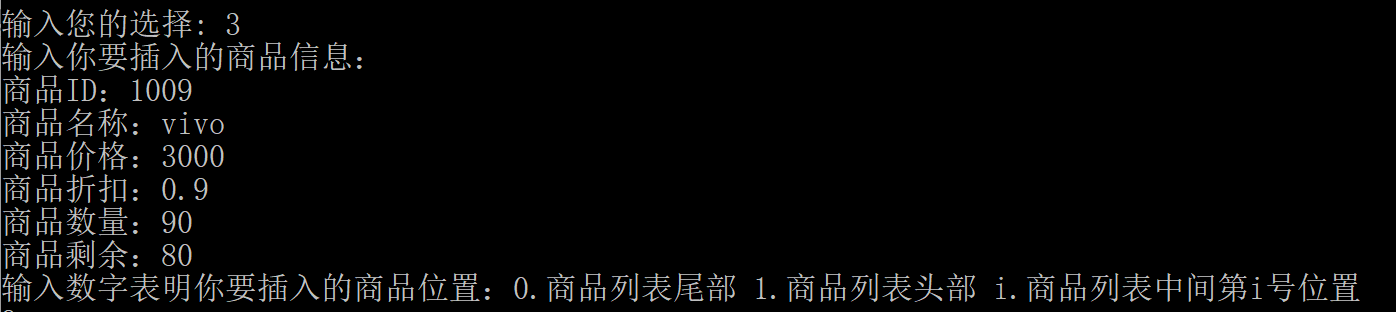
## 三、实验内容

用C语言+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、商品信息的插入等这几个功能。具体实现步骤如下（注：图示为建议显示内容和格式，可自行增加显示内容）:

* 1. 软件界面控制:实现一个数字选项式的启动界面，其中至少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、商品价格排序、退出系统并保存7个选项。并且这些功能可以循环调用。



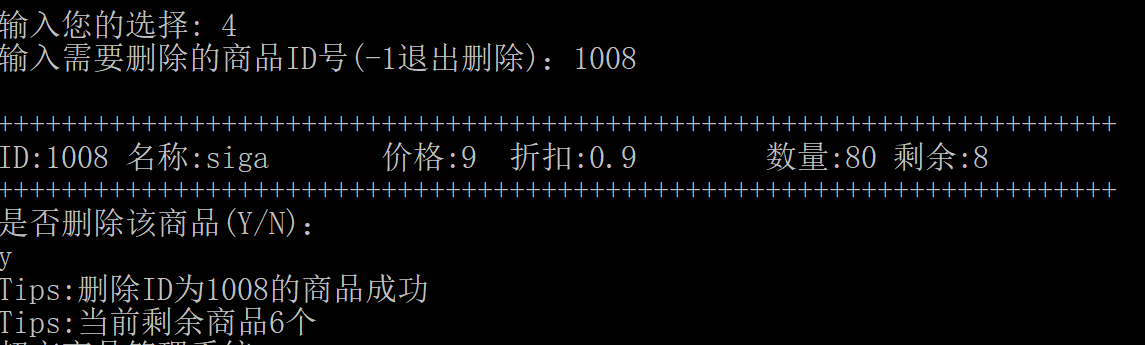
* 1. 商品信息的初始化: 定义链表并初始化。实现从已有的商品信息文件中读入商品信息，并且分配内存保存至链表中。如1）里的图示所示，从文件中读取了7个商品记录。
  2. 商品信息的增加：实现一个函数完成单个商品信息的增加，接受用户的输入的各项信息，然后保存至链表结点。同时实现可以根据用户的输入，将该结点插入到列表的头部，尾部，或者中间i号位置。



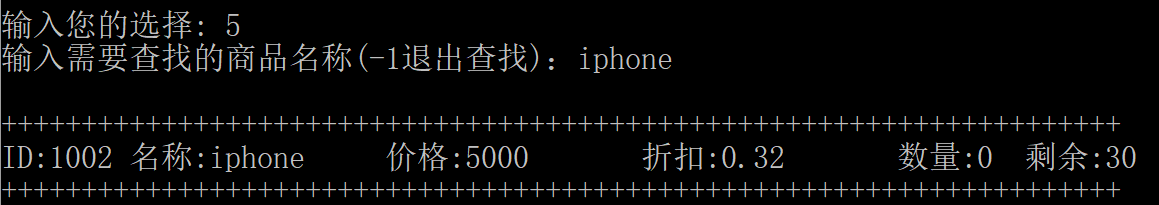
* 1. 商品信息的修改: 实现一个函数完成商品信息的修改功能，实现可以根据商品的名称或者ID修改商品信息。其中用字符串比较的方式来查找待修改商品。



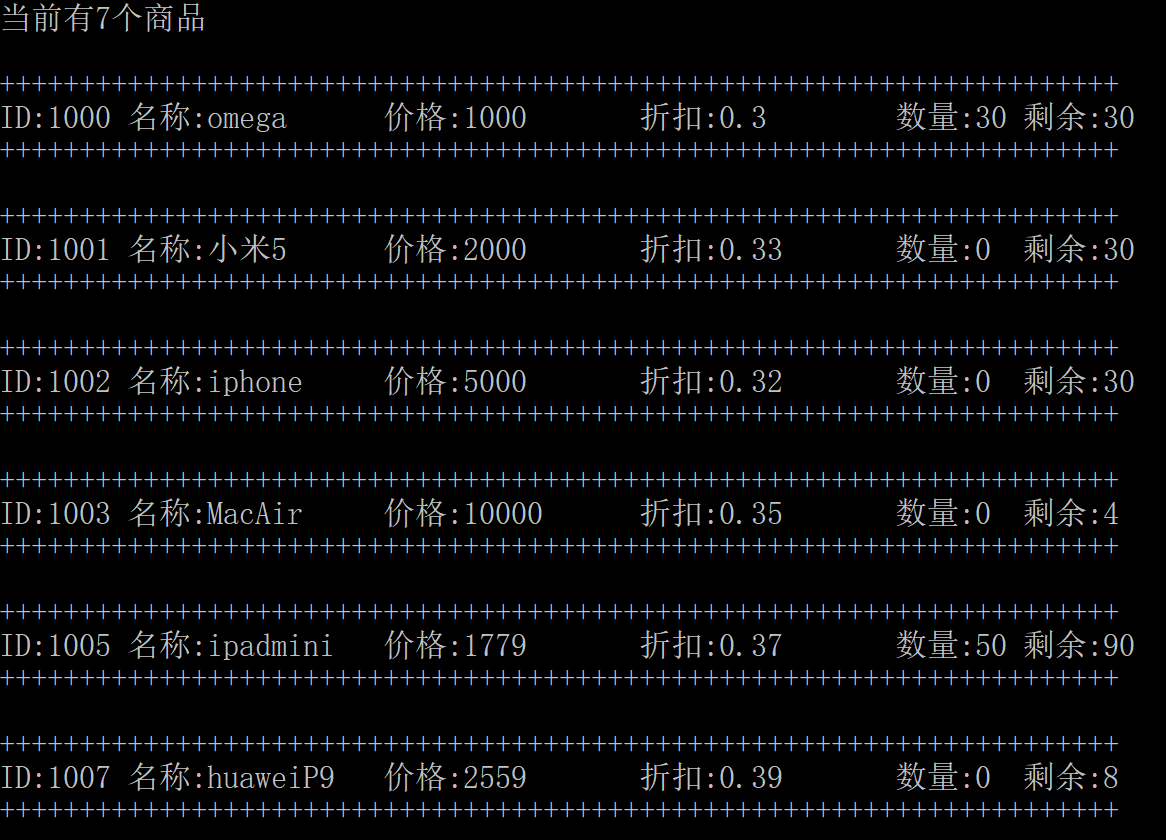
* 1. 商品信息的删除: 实现一个函数，实现根据商品的名称来删除对应的商品信息的功能，商品查找通过字符串比较的方式，查找到后释放对应指针指向的内存区域，完成删除。



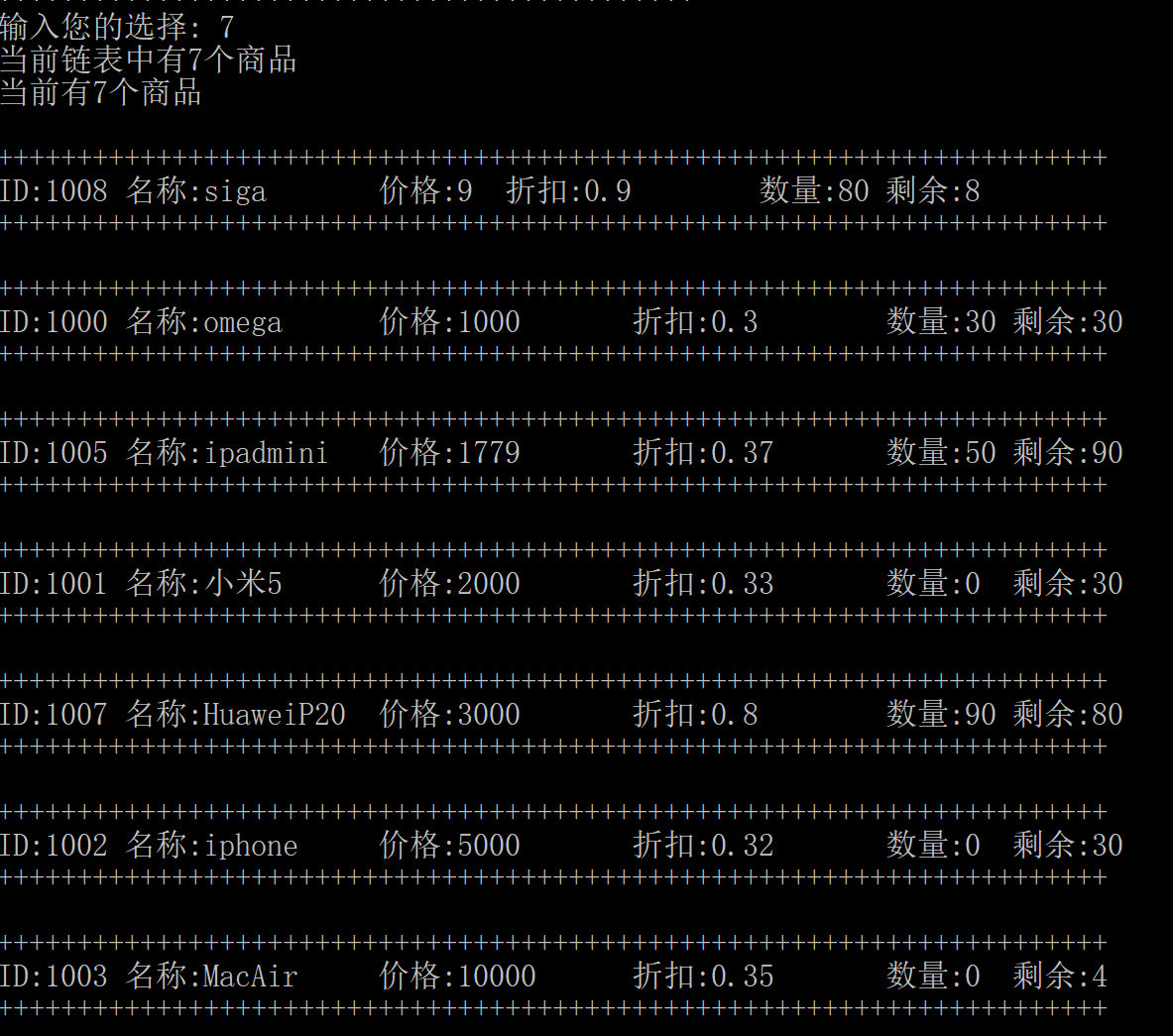
* 1. 商品信息的查找: 实现一个函数，函数的功能是根据输入的商品名称来查找对应的商品信息，商品名称的判断用字符串比较的方式来实现，然后调用格式化显示查找到的商品信息。



* 1. 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。如下图所示的例子



* 1. 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，对商品进行排序，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。



* 1. 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，0然后清理系统运行过程中已分配的内存。

## 四、实验步骤

* 1. 分别完成模板中注释为//TO DO YOUR WORK的函数内容
  2. 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，
  3. 定义并实现一个函数: void info\_init(GoodList \*\*L)，\*\*L是链表的头节点指针的地址。该函数初始化一个链表，然后完成从一个txt文件读入商品链表库并使用相应的商品信息来初始化商品库（即初始化链表），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。
  4. 定义并实现函数:void info\_flush(GoodList \*\*L)，\*\*L是链表的头节点指针的地址。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。
  5. 定义并实现函数:void OutputAll(GoodList \*L)，\*L是链表头节点指针。以格式化的方式，完成将商品链表中的每项信息打印到标准输出(即屏幕上)。
  6. 定义并实现函数:void info\_output(GoodList \*p)，格式化输出某一项商品信息，其中p是该商品结点指针。
  7. 定义并实现函数:void info\_change(GoodList \*\*L)，\*\*L是链表的头节点指针地址。该函数完成商品信息的修改功能，其中要求用户输入需要修改的某项商品的名称或者ID，然后对名称或者ID进行查找，找到则继续输入该商品的各项信息，并提示修改成功;如果没有找到则提示对应商品未找到在两种选择后都返回到初始的选择菜单
  8. 定义并实现函数:void info\_dele(GoodList \*\*L)，\*\*L是链表的头节点指针地址。该函数完成删除某条商品库中信息的功能， 通过输入的某项商品的名称或者ID删除对应的信息，如果在商品库中找到对应的商品便删除该商品信息(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;如果没有找到该商品要提示没有找到该商品信息，在两种选择下都回到上一步选择界面
  9. 定义并实现函数:void info\_search(GoodList \*L)，\*L是链表头节点指针。该函数完成商品信息的查找功能，然后通过输入某种商品信息的名称来检索商品信息库，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
  10. 定义并实现函数:void info\_insert(GoodList \*\*L)，\*\*L是链表的头节点指针地址。该函数完成商品信息的插入，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，加入链表时需要选择在头部插入或者在尾部插入，或者在链表范围内的i号位置插入;在插入之前必须考虑整个信息库的限定容量，如果超过上限要给用户以提示
  11. 定义并实现函数：bubble\_sort(GoodList \*\*L)，\*\*L是链表的头节点指针地址。该函数实现将链表中的所有商品按照商品价格从低到高排序。
  12. 实现程序的入口函数即 main 函数，然后通过一个条件为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