电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2018091602006

姓 名 吴昌博

（实验） 课程名称 程序设计与算法基础I

理论教师 罗绪成

实验教师 罗绪成、朱相印

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：吴昌博 学号：2018091602006指导教师：罗绪成、朱相印**

**实验地点：基础实验大楼535 实验时间：2018.12.25**

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**

【描述本实验中要用到的知识点或者算法】

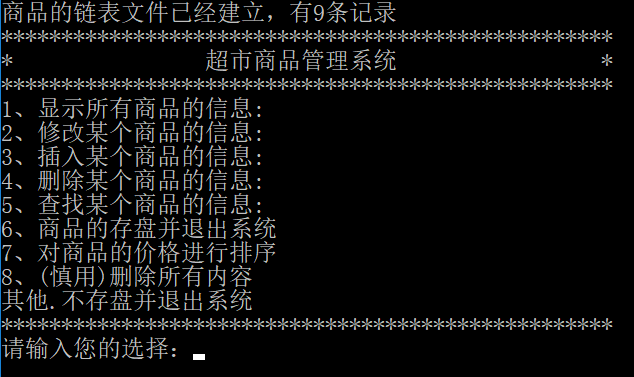
基本知识：顺序、分支、循环三大结构程序设计，数组、字符串，函数，指针（包括结构指针、二级指针、函数指针等高级指针操作），结构类型变量，链表，文件操作，程序设计（文件包含、全局变量、预处理与宏定义、设计原则）

基本算法：优化版的冒泡排序、链表操作。

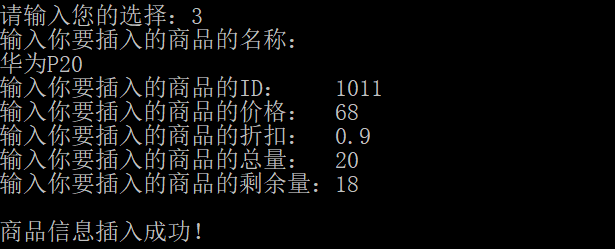
1. **实验内容**

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（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。



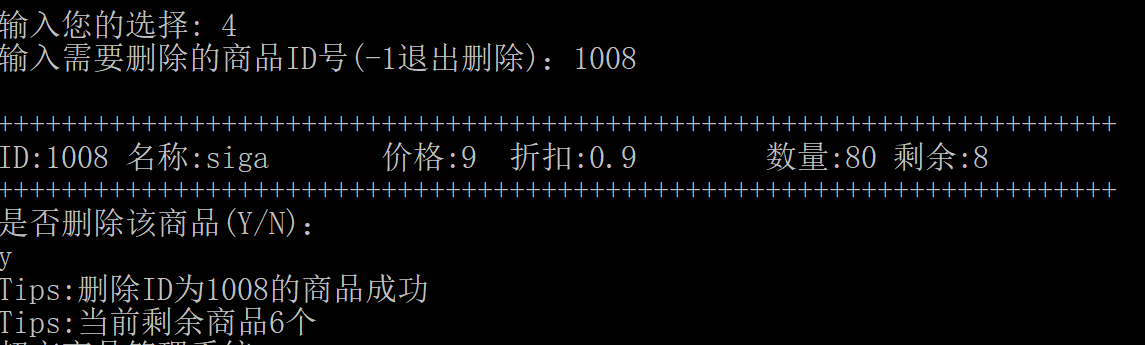
* 1. 商品信息的初始化: 定义链表并初始化。实现从已有的商品信息文件中读入商品信息，并且分配内存保存至链表中。如1）里的图示所示，从文件中读取了7个商品记录。
  2. 商品信息的增加：实现一个函数完成单个商品信息的增加，接受用户的输入的各项信息，然后保存至链表结点。



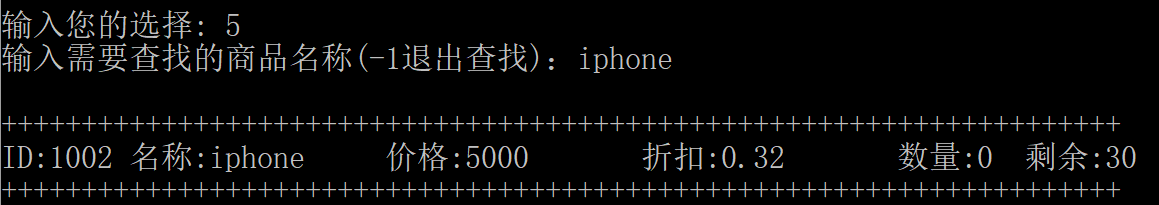
* 1. 商品信息的修改: 实现一个函数完成商品信息的修改功能，实现可以根据商品的名称或者ID修改商品信息。其中用字符串比较的方式来查找待修改商品。



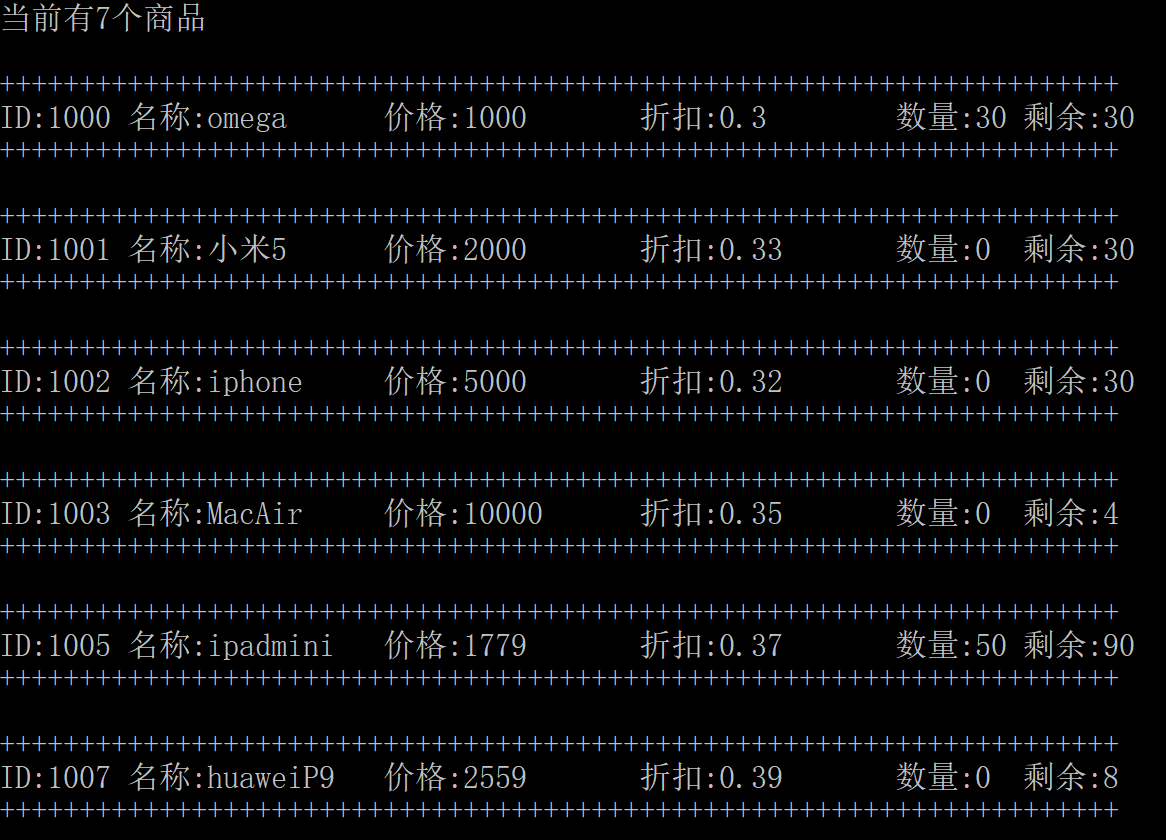
* 1. 商品信息的删除: 实现一个函数，实现根据商品的名称来删除对应的商品信息的功能，商品查找通过字符串比较的方式，查找到后释放对应指针指向的内存区域，完成删除。



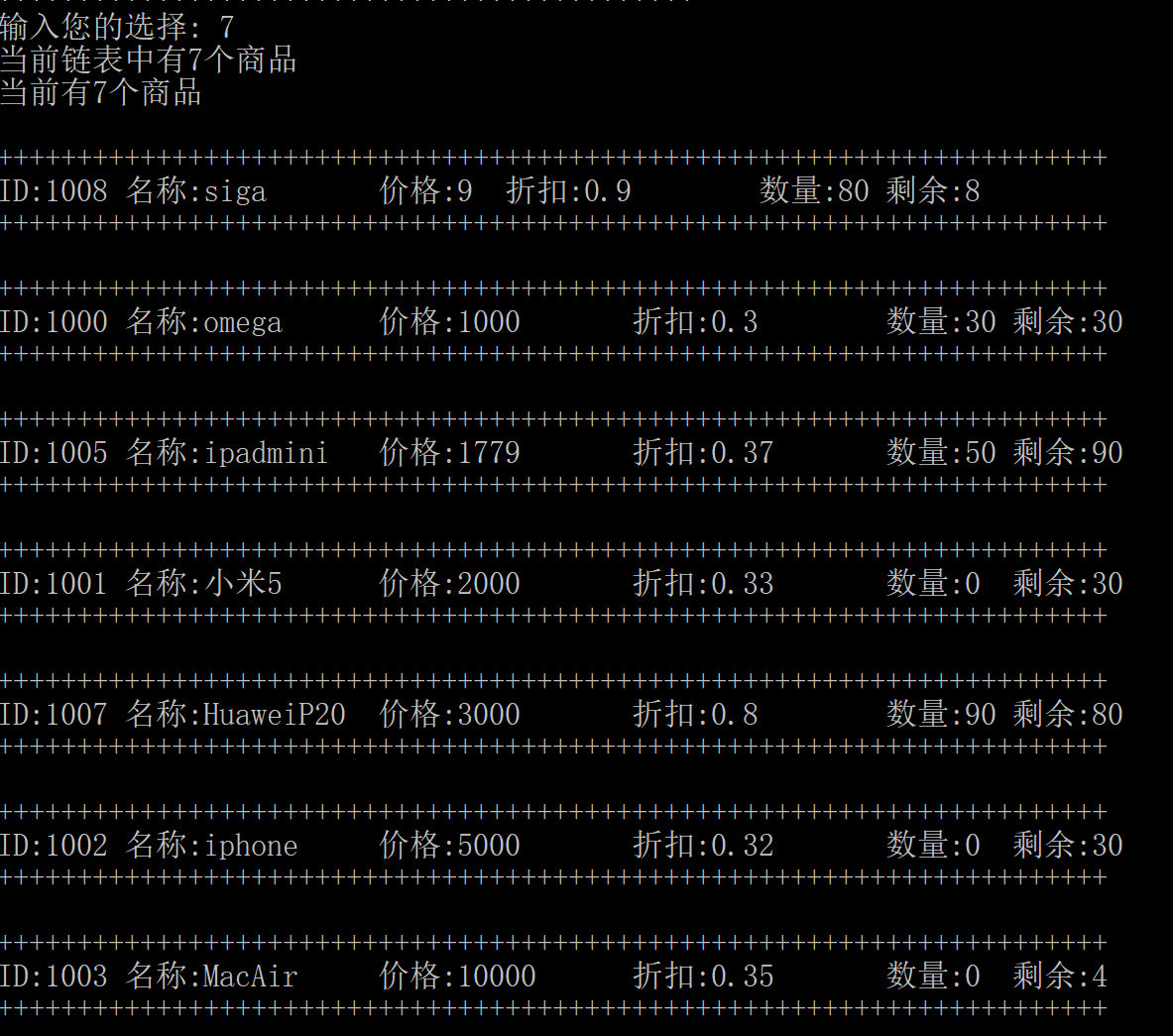
* 1. 商品信息的查找: 实现一个函数，函数的功能是根据输入的商品名称来查找对应的商品信息，商品名称的判断用字符串比较的方式来实现，然后调用格式化显示查找到的商品信息。



* 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。

1. **实验器材（设备、元器件）：**

个人电脑一台

1. **实验步骤**
   1. 根据需求，构建程序的框架：  
      声明必要的头文件、宏定义、全局变量、文件包含。  
      设计主函数：面向用户的界面(数字选项式的启动界面)，操作方式。为满足程序单一出口的原则，决定使用函数指针，在结束分支之后统一调用功能函数。  
      声明八个功能模块的函数原型。遵循避免重复原则(DRY - Don’t repeat yourself)，加入了各个函数中可能会用到的通用的子函数，比如void Goodsprint(GoodsList \*itemdb);
   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。  
      经过试验，在做内容比较复杂的程序时，可以先把链表的数值部分做的简单一点，在最后在进行补充。这样做一方面能高效地构建程序的框架、不必在数据的处理细节上花太多功夫、影响项目进程，另一方面能减少过程错误对整个进程的影响。  
      在链表的数值存储设置上决定将链表的中记录数据的部分再封装成一个结构，可以使结构更加清晰、程序设计更加有条不紊。
   3. 定义并实现一个函数: void init\_itemdb(GoodsList \*\*itemdb)。  
      \*\* itemdb是链表的头节点指针的地址。该函数初始化一个链表，然后用fread(pitem, sizeof(GoodsList), 1, fp)块输入函数，完成从一个二进制文件读入商品链表库，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。  
      其中，为了避免程序因为文件缺失而难以运行，特地定义check\_nullfile()函数检查文件是否存在或者是否为空，如果商品信息初始化文件不存在，程序将为用户新建一个空文件。返回值表示是否文件为空。
   4. 定义并实现函数: void OutputAll(GoodsList \*\*itemdb);  
      \* itemdb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)。
   5. 定义并实现函数: void Goodsprint(GoodsList \*itemdb)，格式化输出某一项商品信息，其中itemdb是该商品结点指针。
   6. 定义并实现函数: void info\_change(GoodsList \*\*itemdb)，\*\* itemdb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，然后对名称进行查找，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  
      考虑到用户的实际使用，决定只用名称查找，用户记得每一个商品的ID是不现实的。  
      其次为了使程序更加灵活，控制副作用，加入了退出修改的选择。
   7. 定义并实现函数: void info\_dele(GoodsList \*\*itemdb)，\*\* itemdb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 。  
      同样地了使程序更加灵活，避免误删信息，加入了确定删除的的提示。
   8. 定义并实现函数:void info\_search(GoodsList \*\*itemdb)，\*\* itemdb是链表的头节点指针地址。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
   9. 定义并实现函数:void info\_insert(GoodsList \*\*itemdb)，\*\* itemdb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，默认加入链表时需要选择在头部插入；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 。  
      让用户来选择插入的位置是画蛇添足的行为，用户不必了解技术的细节，代码应该尽可能减少让读者惊喜。编写的代码只需按照项目、用户的要求来编写。其他华丽的功能就不必了，以免弄巧成拙。
   10. 定义并实现函数：bubble\_sort(GoodsList \*\*itemdb)，\*\* itemdb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。  
       同时定义了一个bool型变量用来检查一轮循环后有没有交换，如果没有就说明已经有序，函数返回。以此来优化冒泡排序。
   11. 编译、调试程序直至达到实验要求
2. **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

|  |
| --- |
| void Goodsprint(GoodsList \*list)  {  printf("商品ID：%4s ", list->data.goods\_id);  printf("商品名称：%6s ", list->data.goods\_name);  printf("商品价格：%3d ", list->data.goods\_price);  printf("商品折扣：%4s ", list->data.goods\_discount);  printf("商品数量：%4d ", list->data.goods\_amount);  printf("商品余数：%4d\n", list->data.goods\_remain);  } |

* 1. 释放链表内存

|  |
| --- |
|  |

* 1. 读取商品文件信息并建立链表

|  |
| --- |
| bool check\_nullfile(void)  {  FILE \*fp = fopen(DATAFILE, "r");  int temp; //temp用来试读一个文件中的数据  int res; //res接收fscanf的返回值，如果是true，表示文件有内容不空，反之为空  //如果文件不存在  if(!fp) {  printf("商品信息初始化文件不存在！程序将为您新建一个.\n");  FILE \*fp = fopen(DATAFILE, "w");  fclose(fp);  }  //如果文件存在，就试读一个数据，看文件是不是空的  res = fscanf(fp, "%d", &temp);  fclose(fp);  if (res <= 0)  return false;  else  return true;  } |

* 1. 将当前商品链表中的内容存入商品文件

|  |
| --- |
| void save\_and\_exit(GoodsList \*\* itemdb)  {  FILE \*fp;  if (itemdb == NULL)  {  printf("the linked list is empty!\n");  exit(EXIT\_FAILURE);  }  if ((fp = fopen(DATAFILE, "wb")) == NULL)  {  perror("File open error:");  exit(EXIT\_FAILURE);  }  while (\*itemdb != NULL)  {  if (fwrite(\*itemdb, sizeof(GoodsList), 1, fp) == 1)  {  \*itemdb = (\*itemdb)->next;  }  }  fclose(fp);  printf("\n|存盘成功！\n");  exit(EXIT\_SUCCESS);  } |

* 1. 输出所有商品信息

|  |
| --- |
|  |
| void OutputAll(GoodsList \*\*itemdb)  {  if (\*itemdb == NULL)  {  printf("目前没有商品信息！\n");  return ;  }  GoodsList \*list;  printf("++++++++++++++++++++++++++++++++++++++++++++++++"  "+++++++++++++++++++++++++++++++++++++++++++++++\n");  for (list = \*itemdb;list != NULL; list = list->next)  {  Goodsprint(list);  printf("++++++++++++++++++++++++++++++++++++++++++++++++"  "+++++++++++++++++++++++++++++++++++++++++++++++\n");  }  } |

* 1. 添加一条商品信息

|  |
| --- |
|  |
| void info\_insert(GoodsList \*\*itemdb)  {  GoodsList \*cur = NULL, \*prev = NULL;  GoodsList \*new\_node;  char insert\_name[MAX\_NAME\_LEN+1];//只是申明为指针出现问题（野指针)  //规定数据上限  if (CurrentCnt == MAX)  {  printf("Database is full!\n");  exit(EXIT\_FAILURE);  }  else if ((new\_node = (GoodsList \*)malloc(sizeof(GoodsList))) == NULL)  {  printf("Error in mallocating!\n");  exit(EXIT\_FAILURE);  }  printf("输入你要插入的商品的名称：\n");  scanf("%s", insert\_name);  if (\*itemdb == NULL)  {  goto insert;  }  for (prev = NULL, cur = \*itemdb;  cur != NULL && strcmp(cur->data.goods\_name, insert\_name) != 0; //strcmp 写成 strcpy  prev = cur, cur = cur->next)  ;  insert:  if (cur == NULL)  {  strcpy(new\_node->data.goods\_name, insert\_name);  printf("输入你要插入的商品的ID： "); scanf("%s", new\_node->data.goods\_id);  printf("输入你要插入的商品的价格： "); scanf("%d", &new\_node->data.goods\_price);  printf("输入你要插入的商品的折扣： "); scanf("%s", new\_node->data.goods\_discount);  printf("输入你要插入的商品的总量： "); scanf("%d", &new\_node->data.goods\_amount);  printf("输入你要插入的商品的剩余量："); scanf("%d", &new\_node->data.goods\_remain);  }  else if (strcpy(cur->data.goods\_name, insert\_name) == 0)  {  printf("该商品已经存在！\n");  free(new\_node);  return;  }  new\_node->next = \*itemdb;  \*itemdb = new\_node;  printf("\n商品信息插入成功！\n");  CurrentCnt++;  //new\_node = cur;  //if (prev == NULL)  // \*itemdb = new\_node;  //else  // prev->next = new\_node;  } |

* 1. 删除一条商品信息

|  |
| --- |
|  |
| void info\_dele(GoodsList \*\*itemdb)  {  GoodsList \*cur, \*prev;  char insert\_name[MAX\_NAME\_LEN+1];  char judge;  if (CurrentCnt == 0)  {  printf("Database is empty!\n");  exit(EXIT\_FAILURE);  }  printf("输入你要删除的商品的名称：\n");  scanf\_s("%s", insert\_name, MAX\_NAME\_LEN);  for (prev = NULL, cur = \*itemdb;  cur != NULL && strcmp(cur->data.goods\_name, insert\_name) != 0;  prev = cur, cur = cur->next)  ;  if (strcmp(cur->data.goods\_name, insert\_name) != 0)  {  printf("该商品不存在！\n");  return;  }  Goodsprint(cur);  printf("确定要删除此商品信息吗？(Y/N)\n");  scanf\_s(" %c", &judge, 1);  judge = toupper(judge);  while (judge != 'N' && judge != 'Y')  {  printf("请输入正确地指令！\n");  printf("你确定要删除所有数据吗？(Y/N)\n");  scanf\_s(" %c", &judge, 1);  judge = toupper(judge);  }  if (judge == 'N')  {  return;  }  else if (judge == 'Y')  {  if (prev == NULL)  {  \*itemdb = cur->next;  }  else  {  prev->next = cur->next;  }  free(cur);  printf("\n商品信息删除成功！\n");  CurrentCnt--;  }  } |

* 1. 查找一条商品记录

|  |
| --- |
|  |
| void info\_search(GoodsList \*\*itemdb)  {  GoodsList \*p;  char insert\_name[MAX\_NAME\_LEN+1];  printf("输入你要查找的商品的名称：\n");  scanf("%s", insert\_name);  for (p = \*itemdb;  p != NULL && strcmp(p->data.goods\_name, insert\_name) != 0;  p = p->next)  ;  if (p == NULL)  {  printf("该商品不存在！\n");  return;  }  Goodsprint(p);  } |

* 1. 修改一条商品信息

|  |
| --- |
|  |
| void info\_change(GoodsList \*\*itemdb)  {  GoodsList \*p;  char insert\_name[MAX\_NAME\_LEN+1];  printf("输入你要修改的商品的名称：(输入\"back\"退出修改)\n");  scanf("%s", insert\_name);  if (strcmp("back", insert\_name) == 0)  {  return;  }  for (p = \*itemdb;  p != NULL && strcmp(p->data.goods\_name, insert\_name) != 0;  p = p->next)  ;  if (p == NULL)  {  printf("该商品不存在！\n");  return;  }  printf("输入修改后的商品的价格： "); scanf("%d", &p->data.goods\_price);  printf("输入修改后的商品的折扣： "); scanf("%s", p->data.goods\_discount);  printf("输入修改后的商品的总量： "); scanf("%d", &p->data.goods\_amount);  printf("输入修改后的商品的剩余量："); scanf("%d", &p->data.goods\_remain);  printf("\n商品信息修改成功！\n");  } |

* 1. 链表排序

|  |
| --- |
|  |
| void bubble\_sort(GoodsList \*\*itemdb)  {  bool flag = false;  int count = 0;  GoodsList \*first, \*second;  GoodsInfo temp\_value;  first = \*itemdb;  while (first != NULL)  {  count++;  first = first->next;  }  for (int i = 1; i < count; i++)  {  first = \*itemdb;  second = (\*itemdb)->next;//优先级  for (int j = 0; j < count - i; j++)  {  if (first->data.goods\_price > second->data.goods\_price)  {  temp\_value = second->data;  second->data = first->data;  first->data = temp\_value;  flag = true;  }  first = first->next;  second = second->next;  }  if (flag)  flag = false;  else  break;  }  printf("排序完成！\n");  } |

* 1. 主界面实现

|  |
| --- |
|  |
| int main(void)  {  GoodsList \* itemdb = NULL;  int choice = 0;  //if ((itemdb =(GoodsList \*) malloc(sizeof(GoodsList))) == NULL)  //{  // printf("Error in setting up a list!\n");  // exit(EXIT\_FAILURE);  //}  init\_itemdb(&itemdb);  while (1)  {  main\_interface();  printf("请输入您的选择：");  scanf("%d", &choice);  while (getchar() != '\n')//get√  ;  switch (choice)  {  case 1:funprt = OutputAll;  break;  case 2:funprt = info\_change;  break;  case 3:funprt = info\_insert;  break;  case 4:funprt = info\_dele;  break;  case 5:funprt = info\_search;  break;  case 6:funprt = save\_and\_exit;  break;  case 7:funprt = bubble\_sort;  break;  case 8:funprt = DelAll;  break;  default: goto End;  break;  }  (\*funprt)(&itemdb); //使用函数指针，满足程序单一出口的原则  printf("\n");//get√  }  End:;  return 0;  }  //------------------------------------------------------------------------------  //主界面 输出  void main\_interface()//怎么做成流动式的？  {  printf(  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"  "\* 超市商品管理系统 \*\n"  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"  "1、显示所有商品的信息:\n"  "2、修改某个商品的信息:\n"  "3、插入某个商品的信息:\n"  "4、删除某个商品的信息:\n"  "5、查找某个商品的信息:\n"  "6、商品的存盘并退出系统\n"  "7、对商品的价格进行排序\n"  "8、(慎用)删除所有内容\n"  "其他.不存盘并退出系统\n"  "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"  );  } |

1. **总结及心得体会：**

写一个项目和完成一道编程题目是有很大差别的。

首先是目标不同。编程题是为了AC，尽可能降低时间和空间的复杂度，有时候对程序的可读性没有太大的要求，尽可能地简单。而项目最终目标是为了服务用户，如果立脚点不是这个，作品就不容易商业化，服务大众。所以会在程序设计过程中考虑很多实际的问题。比如会考虑为了避免程序因为文件缺失而难以运行，特地定义check\_nullfile()函数检查文件是否存在或者是否为空，如果商品信息初始化文件不存在，程序将为用户新建一个空文件。比如在查找时，用户记得每一个商品的ID是不现实的。考虑到用户的实际使用，决定只用名称查找；比如其次为了使程序更加灵活，控制副作用，加入了退出修改的选择、确定删除的的提示。再比如让用户来选择插入的位置是画蛇添足的行为，用户不必了解技术的细节。编写的代码只需按照项目、用户的要求来编写。

其次是规模不同。一道题连续的一段时间即可解决。但是完成一个项目可能要花费一段不连续的时间，所以要保证程序的可读性。另一方面，涉及大规模的程序设计时有可能需要和别人合作，那么就更需要和他人协作，所以要减少全局变量的使用、注意书写的风格规范性等等。要考虑函数间的关系、函数对整个程序的影响等等。

规模不同源于实现功能的多少。一个项目的功能是比较多的。但依然有一些程序设计的原则：程序单一出口的原则、避免重复原则等等。所以决定使用函数指针，在结束分支之后统一调用功能函数；加入了各个函数中可能会用到的通用的子函数。在实验时注重框架感。为了进程的保证，最开始把链表的数值部分做的简单一点。一方面能高效地构建程序的框架、不必在数据的处理细节上花太多功夫，影响项目进程，另一方面能减少过程错误对整个进程的影响。

1. **对本实验过程及方法、手段的改进建议：**

**报告评分：**

**指导教师签字：**