电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2018091201001

姓 名 杨泽东

（实验） 课程名称 程序设计基础

理论教师 吴劲

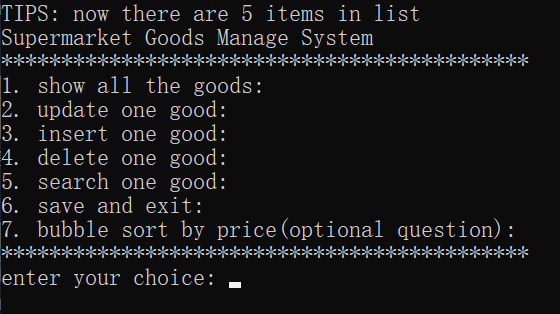
实验教师 杨珊

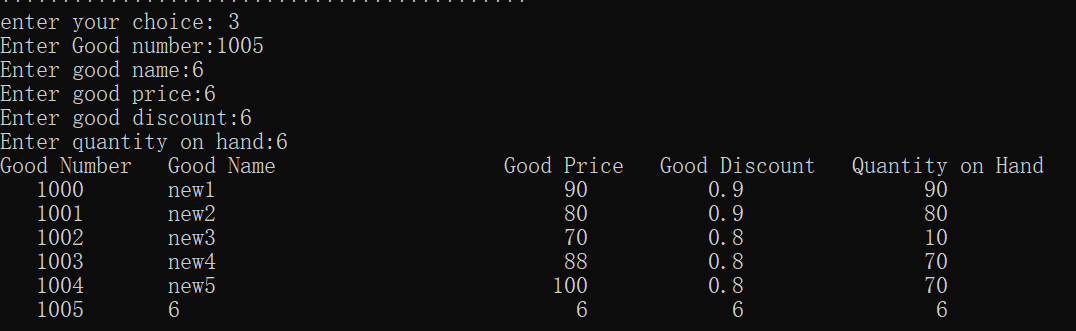
**电 子 科 技 大 学**

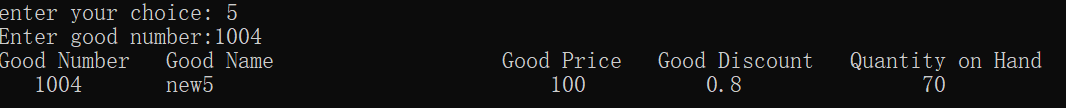
**实 验 报 告**

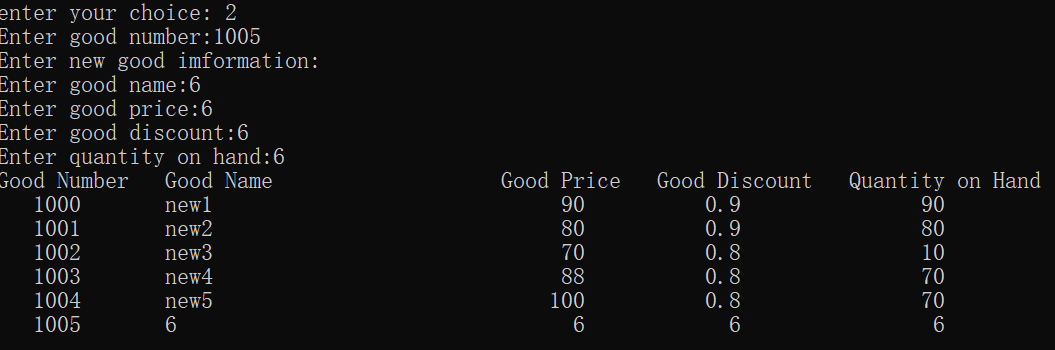
**学生姓名：杨泽东 学号：2018091201001 指导教师：吴劲**

**实验地点：品学楼C519 实验时间：2018.12.28-2019.1.4**

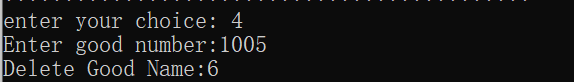
1. **实验名称：超市商品管理系统链表实现**
2. **实验学时：4学时**
3. **实验目的：**
4. 掌握单链表的定义和使用方法
5. 掌握单链表的建立方法
6. 掌握单链表中节点的查找与删除
7. 掌握输出单链表节点的方法
8. 掌握链表节点排序的一种方法
9. 掌握C语言创建菜单的方法
10. 掌握结构体的定义和使用方法
11. **实验原理：**
12. 单链表的数据操作
13. 基本语法的使用
14. C语言文件操作函数的理解和使用
15. Visual Studio开发环境
16. **实验内容：**
17. 创建单链表并将其定义为新类型 goodlist ，同时定义inventory为链表头节点。
18. 使用c语言文件操作函数fopen , fclose , fprintf , fscanf与fprintf\_s,fscanf\_s等打开指定txt文件，并读取txt文件内容。
19. 将读取到的内容录入到链表。
20. 输出超市系统封面（main函数）。
21. 完善链表操作函数

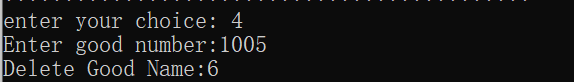
void insert(void);查重，确认无重复后，将新商品插入。

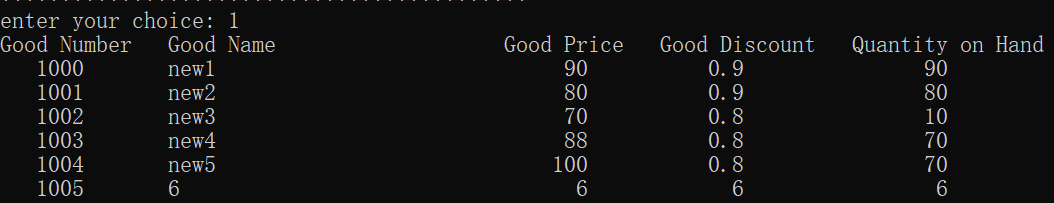
void search(void);查找所需函数，若没有则打印错误信息。

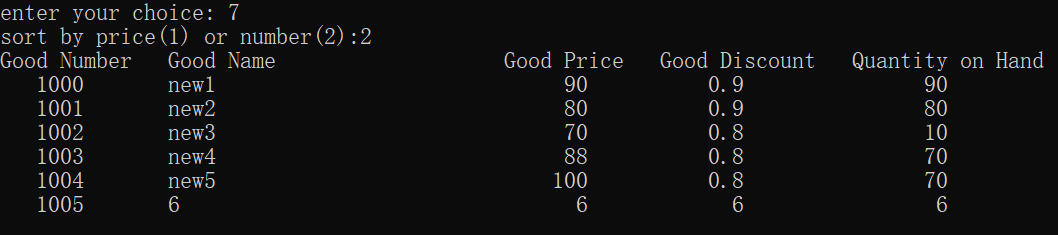
void update(void);寻找目标节点，提示用户输入新信息，若没有，打印错误信息

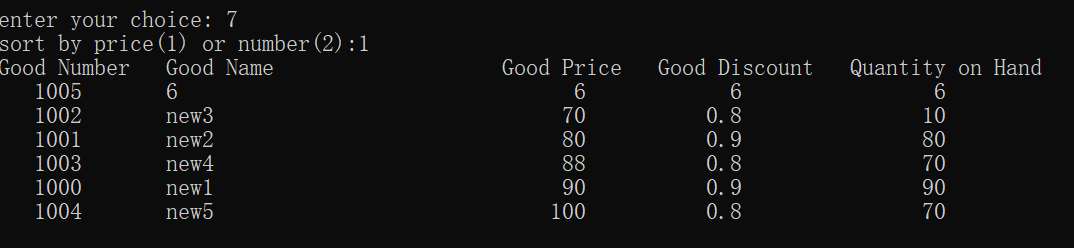
void deletes(void);提示用户输入信息，寻找目标信息，若存在，则删除，若不在则打印错误信息。





void print(void)；将商品信息从头开始打印。

void bubblesort(void);冒泡排序。按照价格或者序号进行排序。



1. **实验器材（设备、元器件）：**

个人电脑一台

1. **实验步骤：**
2. 阅读实验要求，明确实验目的
3. 查阅相关资料，为编写程序做好准备
4. 编写程序
5. 运行测试
6. 提交老师审查
7. **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）**

代码1-1创建单链表并定义头节点

|  |
| --- |
| typedef struct goods {  int number;  int price;  char name[MAX\_NAME\_LEN + 1];  char goods\_discount[MAX\_DISCOUNT\_LEN + 1];  int on\_hand;  struct goods \*next;  }goodslist;  goodslist \*inventory; |

创建单链表并将其定义为新声明goodslist方便以后的使用，然后定义inventory为头节点。

代码1-2定义系统需要调用的函数

|  |
| --- |
| //系统主体功能（单链表的操作）  void insert(void);  void search(void);  void update(void);  void deletes(void);  void print(void);  void bubblesort(void);  //文件操作函数及退出函数  void goodslist\_init(void);  void save\_to\_file\_exit(void);  void DestroyGoods(goodslist\* inventory); |

将系统分割为几大功能块，并将其定义为各个函数，精简main函数的同时，明确目标及操作流程。

代码1-3输出系统封面（即完善main函数）

|  |
| --- |
| int main(void)  {  int choice;  goodslist\_init();  while(1)  {  printf("Supermarket Goods Manage System\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(optional question):\n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("enter your choice: ");  scanf\_s("%d", &choice,20);  switch (choice)  {  case 1:print(); break;  case 2:update(); break;  case 3:insert(); break;  case 4:deletes(); break;  case 5:search(); break;  case 6:save\_to\_file\_exit(); break;  case 7:bubblesort(); break;  default:printf("Please enter a right choise!"); break;  }  printf("\n");  }  } |

使用根据系统特点，选择使用while(1)进行外圈大循环，swich函数进行内圈小循环，选择需要执行的函数。

代码2-1链表初始化

|  |
| --- |
| void goodslist\_init(void)  {  FILE \*fp;  int count = 0;  goodslist \*cur, \*prev;  inventory = NULL;  prev = inventory;  cur = NULL;    if ((err = fopen\_s(&fp,"goodsinfo.txt", "r")) != 0)  {  if ((err = fopen\_s(&fp, "goodsinfo.txt", "w")) != 0)  {  printf("TIPS：CAN NOT CREAT GOODS DATABASE FILE \n");  return;  }  else  {  printf("TIPS: now there are 0 items in list\n");  printf("Please insert new one!\n");  }//若文件不存在，新建空文件并且让用户输入新的信息  }  else {  inventory = (goodslist\*)malloc(sizeof(goodslist));  fscanf\_s(fp, "%d", &inventory->number,20);  fscanf\_s(fp, " %s", inventory->name,20);  fscanf\_s(fp, " %d", &inventory->price,20);  fscanf\_s(fp, " %s", inventory->goods\_discount,20);  fscanf\_s(fp, " %d\n", &inventory->on\_hand,20);  inventory->next=NULL;  while (!feof(fp))  {  cur = (goodslist\*)malloc(sizeof(goodslist));  for (prev = inventory;  prev->next != NULL;  prev = prev->next  );  if (cur == NULL) {  printf("Database is full; can't add more goods.\n");  return;  }  prev->next = cur;  cur->next = NULL;  fscanf\_s(fp, "%d", &cur->number,20);  fscanf\_s(fp, " %s",cur->name,20);  fscanf\_s(fp, " %d", &cur->price,20);  fscanf\_s(fp, " %s", cur->goods\_discount,20);  fscanf\_s(fp, " %d\n", &cur->on\_hand,20);  prev = cur;    count++;  }    printf("TIPS: now there are %d items in list\n", count+1);  }  cur = NULL;  prev = NULL;  fclose(fp);  fp = NULL;  } |

打开同目录下指定文件，先初始化首节点inventory，之后逐行读取文本内容并依次存入cur节点，使用prev指针寻找到链表尾，然后将cur节点接入到链表尾，每接入一次，count+1，最后打印时加上头节点

2-2-1保存链表内容到文件并清空链表内存

|  |
| --- |
| void save\_to\_file\_exit(void)  {  goodslist \*cur, \*prev;  int savecount = 0;//计数  FILE \*fp;  if ((err = fopen\_s(&fp, "newgoodsinfo.txt", "w")) != 0)  {  printf("TIPS：can not open goods file database!\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);  fp = NULL;  DestroyGoods(inventory);  cur = NULL;  prev = NULL;  printf("TIPS：%d items save into newgoodsinfo.txt \n", savecount );  } |

在相对目录下创建空txt文件，使用for循环，将链表内容依次打印到txt文件中，打印结束后使用DestroyGoods函数回收内存。

2-2-2清空链表所占用内存

|  |
| --- |
| void DestroyGoods(goodslist\* inventory) {  if (!inventory) return;  goodslist \*current, \*prev;  current = inventory;  while (current->next!=NULL)  {  prev = current;  current = current->next;  free(prev);  }  prev = NULL;  free(current);  current = NULL;  } |

依次清空链表占用的内存空间

代码2-3链表插入新节点

|  |
| --- |
| void insert(void)  {  goodslist \*current, \*pre,\*check;    pre = inventory;      current = (goodslist\*)malloc(sizeof(goodslist));  if (pre == NULL)  {  inventory = current;    }//若链表没有初始化，则从首节点开始插入  else  {  while (pre->next != NULL)  pre = pre->next;  pre->next = current;  }  current->next = NULL;  printf("Enter Good number:");  scanf\_s("%d", &current->number,20);    for (check = inventory;  check->next != NULL ;  check = check->next)  {  if (check->number == current->number)  {  printf("Good already exists.\n");  check = NULL;  free(current);  current = NULL;  pre->next = NULL;  return;  }  }//查重  check = NULL;    printf("Enter good name:");  scanf\_s("%s", current->name,20);  printf("Enter good price:");  scanf\_s("%d", &current->price,20);  printf("Enter good discount:");  scanf\_s("%s", current->goods\_discount,20);  printf("Enter quantity on hand:");  scanf\_s("%d", &current->on\_hand,20);  print();  current = NULL;  pre = NULL;  } |

先判断首节点是否已经初始化，既判断上一步中链表是否已经初始化成功（若txt文件无法读取或者不存在情况），若没有初始化，则从首节点inventory进行插入操作，否则从尾端开始插入，之后还需加入查重工序，确认无重复后照常录入即可。

代码2-4寻找到目标节点并打印

|  |
| --- |
| void search(void)  {  goodslist \*prev;  printf("Enter good number:");  int i;  scanf\_s("%d", &i,20);  for (prev=inventory;  prev->number != i&&prev->next!=NULL;  prev = prev->next);  if (prev->next == NULL)  if (prev->number != i)  {  printf("This good isn't isist!");  return;  }//寻找到目标节点  printf("Good Number Good Name Good Price Good Discount "  "Quantity on Hand\n");  printf("%7d %-25s %7d %7s %11d\n", prev->number, prev->name, prev->price, prev->goods\_discount,  prev->on\_hand);  prev = NULL;  } |

使用prev指针，遍历链表，寻找目标节点，若节点存在，则将其内容打印。

代码2-5节点内容的修改

|  |
| --- |
| void update(void)  {  goodslist \*p;  p = inventory;  printf("Enter good number:");  int i;  scanf\_s("%d", &i);    for (; p->number != i&&p->next!=NULL; p = p->next);  if (p->next == NULL)  if (p->number != i)  {  printf("This good is not exist!");  return;  }  printf("Enter new good imformation:\n");  printf("Enter good name:");  scanf\_s("%s", p->name,20);  printf("Enter good price:");  scanf\_s("%d", &p->price,20);  printf("Enter good discount:");  scanf\_s("%s", p->goods\_discount,20);  printf("Enter quantity on hand:");  scanf\_s("%d",&p->on\_hand,20);  p = NULL;  } |

使用p指针，遍历链表，寻找目标节点，若节点存在，则提示用户输入该商品的新信息，将旧信息修改。

代码2-6节点的删除

|  |
| --- |
| void deletes(void)  {  goodslist \*p,\*f;  printf("Enter good number:");  int i;  scanf\_s("%d", &i,20);  f = inventory;  for (p = inventory;  p->number != i&&p->next!=NULL;  f=p, p = p->next);//找到目标节点，同时使用f标记前一节点，方便衔接  if (p->next == NULL)  if(p->number!=i)  {  printf("This good is not exist!");  return;  }  printf("Delete Good Name:%s", p->name);  if (p == inventory)  inventory = p->next;  else  {  f->next = p->next;  }  free(p);  f = NULL;  p = NULL;  } |

遍历链表，找到目标节点，若节点存在，且节点是头节点，则将头节点重新设置，若是普通节点，则连接好前后节点后，将其占用的内存空间清空，并使其指向NULL。

代码2-7冒泡排序法

|  |
| --- |
| void bubblesort(void)  {  printf("sort by price(1) or number(2):");  int i;  scanf\_s("%d", &i, 20);  goodslist \*pre, \*cur,\*next,\*end, \*temp,\*head;  head = (goodslist\*)malloc(sizeof(goodslist));  head->next = inventory;  end = NULL;  if (i == 1)  {  while (head->next->next != end)  {  pre = head;  cur = pre->next;  next = cur->next;  while (next != end)  {  if (cur->price > next->price)  {  cur->next = next->next;  pre->next = next;  next->next = cur;  temp = next;  next = cur;  cur = temp;//前中后节点的交换  }  pre = pre->next;  cur = cur->next;  next = next->next;  }  end = cur;  }  inventory = head->next;//使inventory重新成为头指针  }  else {  while (head->next->next != end)  {  pre = head;  cur = pre->next;  next = cur->next;  while (next != end)  {  if (cur->number > next->number)  {  cur->next = next->next;  pre->next = next;  next->next = cur;  temp = next;  next = cur;  cur = temp;  }  pre = pre->next;  cur = cur->next;  next = next->next;  }  end = cur;  }  inventory = head->next;  }  print();  pre = NULL;  cur = NULL; next = NULL; end = NULL; temp = NULL; head = NULL;  } |

根据系统需要，采用if作为外层大结构，之后使用冒泡排序法对链表内容进行排序。若两节点内容符合要求，则两节点交换，注意衔接处理，内圈小循环没循环一次，就可以把最大的下沉到最后，并将其设置为新的end指针，这样使得每循环一次，最大的都可以排到最后；外圈大循环则使循环一直进行到全部元素（除辅助head外）都参与排序为止。由于这里在inventory前创建多一个head指针方便之后的操作，所以在操作的最后需要把参与排序而移位了的inventory重新指向head->next，使其重新成为头节点指针。

inventory

head

1000

inventory

1000

head

代码2-8打印全部链表信息

|  |
| --- |
| void print(void)  {  goodslist \*p;  p = inventory;  printf("Good Number Good Name Good Price Good Discount "  "Quantity on Hand\n");  for (; p->next != NULL; p = p->next)  printf("%7d %-25s %7d %7s %11d\n", p->number, p->name, p->price, p->goods\_discount,  p->on\_hand);  if(p->next==NULL)  printf("%7d %-25s %7d %7s %11d\n", p->number, p->name, p->price, p->goods\_discount,  p->on\_hand);  p = NULL;  } |

使用指针p，逐个打印链表全部内容，最后把指针赋值为NULL。

1. **总结及心得体会：**

总结：很有挑战性的一次实验，需要查阅较多资料，虽然开始时误入歧途，但最终还是有所收获。

心得体会：本实验初次上手，就觉得不适应，毕竟需要完成多个功能，但是将主要功能体捋一遍后，脉络逐渐清晰，开始逐步进行。

由于visual studio的特殊性，编译过程出现了较多的麻烦，比如scanf\_s的内存空间设置（缓冲区大小），fopen\_s的使用等，由于没有设置内存空间，稍不注意就报错，导致浪费了大量时间，不过也因此学习到了很多关于visual studio的使用技巧，尤其是在visual studio中链表的指针指向NULL后的种种问题，这都是在其他ide上没有遇到的，虽然解决耗费了大量时间，但是收益颇丰。期间由于没有看懂老师给的函数如何使用，便自己仿照写了一个文件打开以及操作的函数，期间也遇到了EOF，feof的使用，fscanf\_s的使用的问题，查阅了许多资料，最终得以解决，虽然现在还不完善（没有达到预期要求），当打开文件为空时任然会出现错误(将EOF作为inventory)。目前还想不到较好的解决方案，只能做到当文件不存在时，提示用户插入新的商品。

随后便是链表操作函数的实现，在弄清楚链表原理后，麻烦并不大，偶尔出现指针指向错误空间的问题都得到了一一解决。

当写到最后阶段的冒泡排序函数时，通过上网，查到了许多资料和做法，慢慢参照，最后终于得以理解，同时也对链表的操作有了更深的认识，但中途由于网上方法的局限性，导致我首节点无法参与到比较中去，或者打印的时候只能打印两个链表，后来经过老师的指点，在inventory前创建head作为辅助指针，这样就可以将inventory纳入比较中去，最后将进行比较排序了的inventory指向head->next，重新成为头节点，解决了打印问题，不得不夸一下老师，太强了！

现在回过头来看，本次实验，主要是考察我们对c语言高级数据操作——链表的掌握程度以及文件操作函数的简单应用，虽然一开始听的很懵，但是真正多次上手，查阅较多资料后，链表逐渐变得清晰，最后实战，链表的使用更加深入人心。此次实验，对我个人而言还使我更好的掌握了visual studio的使用技巧，对冒泡排序也有了更深的理解。同时，也让我认识到学习软件工程这一专业，单凭老师的教学是远远不够的，这更需要我们善于利用各种手段查阅各种资料，认真学习，化为己用，学会主动学习，让兴趣成为自己真正的“老师”；同时也要虚心向周围同学交流，思想的碰撞才能点亮智慧的火花，给别人看代码的同时，对自己也是一种提高。

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：**