

**课程设计报告**

|  |  |  |
| --- | --- | --- |
| 课程设计题目 | ： | 链表的维护与文件形式的保存 |
| 专 业 | ： | 计算机科学与技术 |
| 班 级 | ： | 2220704 |
| 学 号 | ： | 2022212080 |
| 学生姓名 | ： | 刘纪彤 |
| 指导教师 | ： | 王同罕 |
| 日 期 | ： | 2023年 11月 30日 |

目录

[一、 设计题目 3](#_Toc152494296)

[二、 设计内容 3](#_Toc152494297)

[三、 概要设计 3](#_Toc152494298)

[三、 算法描述 4](#_Toc152494299)

[三、 程序测试 13](#_Toc152494300)

[五、 算法分析 19](#_Toc152494301)

[五、 心得体会 19](#_Toc152494302)

[六、 参考文献 20](#_Toc152494303)

1. 设计题目

**链表的维护与文件形式的保存**

用链表结构的有序表表示某商场家电的库存模型。当有提货或进货时需要对该链表进行维护。每个工作日结束之后，将该链表中的数据以文件形式保存，每日开始营业之前，需将以文件形式保存的数据恢复成链表结构的有序表。

链表结点的数据域包括家电名称、品牌、单价和数量，以单价的升序体现链表的有序性。程序功能包括：创建表、营业开始（读入文件恢复链表数据）、进货（插入）、提货（更新或删除）、查询信息、更新信息、营业结束（链表数据存入文件）等。

1. 设计内容

主要想要以链表的形式存储某商场的货物，可以根据我们的需要，进行增删改查，数据保存和数据的导入。

1. 概要设计

程序主要有以下几种模块设计：

1、主体，调用所有的各个分模块，将各个模块结合起来，成为一个程序

2、链表操作模块，这个模块主要负责的是关于链表处理相关的模块设计。

3、文件操作模块，此模块则主要负责的对于文件的读写操作，提供文件内容，写入文件内容等。

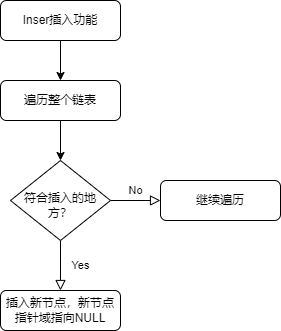
4、其他辅助模块，包括输入输出、界面显示的处理。

5、GUI的实现模块

通过GUI的Mainwindow模块链接各个子模块，子模块分Inset,Search,Fileio,update,change,update\_datas,这几个子模块，分别实现了题目要求的插入，查找，更新，更新数据，完成整个程序的功能。

1. 算法描述

1、insert插入功能主要的目的是插入数据，并按照升序插入，包含fileio里面从文件读取功能，也要调用这个insert功能。程序流程如下：



该模块实现的代码如下:

**void insert(goods \*headaq, const string& name, const string& brand, int num, float price)**

**{**

**goods \*p,\*q;**

**p=headaq;**

**while(p->next!=NULL)**

**{**

**if(p->next->price>price)**

**{**

**break;**

**}**

**p=p->next;**

**}**

**q=new goods;**

**q->name = name;**

**q->brand = brand;**

**q->num=num;**

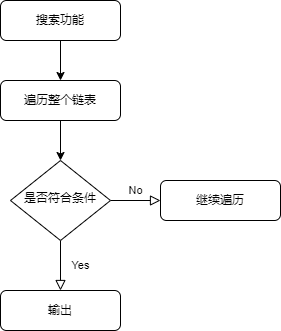
**q->price=price;**

**q->next=p->next;**

**p->next=q;**

**}**

2、搜索功能，搜索分为了按照品牌搜索和按照名称搜索，但是他们的基本逻辑都是相同的，都是按照以下流程工作的:



该模块的代码：

**void serach\_table::search\_name(goods \*head, const string& name)**

**{**

**goods \*p;**

**p=head->next;**

**while(p!=NULL)**

**{**

**if(p->name==name)**

**{**

**ui->out->setText("查询结果");**

**ui->out->append(QString("商品名称:")+QString(QString::fromStdString(p->name)));**

**ui->out->append(QString("商品品牌:")+QString(QString::fromStdString(p->brand)));**

**ui->out->append(QString("商品数量:")+QString(QString::number(p->num)));**

**ui->out->append(QString("商品价格:")+QString(QString::number(p->price)));**

**break;**

**}**

**p=p->next;**

**}**

**}**

**// 按品牌查找商品信息**

**void serach\_table::search\_brand(goods \*head, const string& type)**

**{**

**goods \*p;**

**p = head->next;**

**while (p != NULL)**

**{**

**if (p->brand == type)**

**{**

**ui->out->setText("查询结果");**

**ui->out->append(QString("商品名称:")+QString(QString::fromStdString(p->name)));**

**ui->out->append(QString("商品品牌:")+QString(QString::fromStdString(p->brand)));**

**ui->out->append(QString("商品价格:")+QString(QString::number(p->price)));**

**ui->out->append(QString("商品数量:")+QString(QString::number(p->num)));**

**break;**

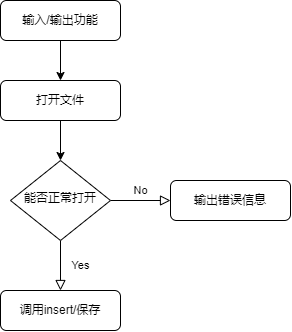
**}**

**p = p->next;**

**}**

**}**

3、文件操作模块：我们导入一个文本文件，从文件中读取我们输出的文本文件的数值，恢复数据。在结束营业时也可以利用存储进行文件的保存，其流程图如下:



代码如下:

**// 每日开始营业之前，需将以文件形式保存的数据恢复成链表结构的有序表。**

**void create\_head(goods \*headq,QString fn) // 创建链表**

**{**

**ifstream fin(fn.toStdString());**

**if(!fin)**

**{**

**QMessageBox::critical(NULL,QMessageBox::tr("错误"),QMessageBox::tr("文件打开失败"));**

**exit(0);**

**}**

**char name[20];**

**char brand[20];**

**int num;**

**float price;**

**while(fin>>name>>brand>>num>>price)**

**{**

**insert(headq,name,brand,num,price);**

**}**

**fin.close();**

**}**

**// 结束营业，将链表中的数据写入文件**

**void end(goods \*headq,QString fn)**

**{**

**ofstream fout(fn.toStdString());**

**if(!fout)**

**{**

**QMessageBox::critical(NULL,QMessageBox::tr("错误"),QMessageBox::tr("文件打开失败"));**

**exit(0);**

**}**

**goods \*p;**

**p=headq->next;**

**while(p!=NULL)**

**{**

**fout<<p->name<<" "<<p->brand<<" "<<p->num<<" "<<p->price<<endl;**

**p=p->next;**

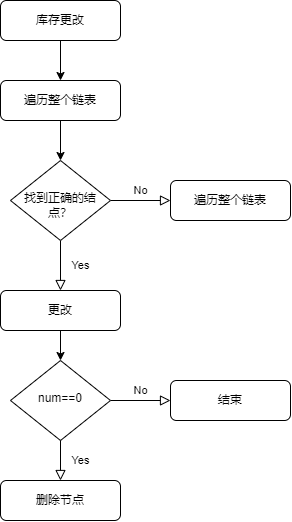
**}**

**fout.close();**

**}**

4、进货/提货：整个功能我都利用了update函数来实现库存的更新，当进货、提货一个商品的时候，将库存改为指定的值，如果提货之后，库存的值为0，商品就可以删除了。

流程图如下:



代码如下:

**void updating(goods \*head, const string& name, int num)**

**{**

**goods \*p;**

**p=head->next;**

**while(p!=NULL)**

**{**

**if(p->name==name)**

**{**

**p->num=num;**

**//如果数量为0，删除该节点**

**if(p->num==0)**

**{**

**goods \*q;**

**q=head;**

**while(q->next!=p)**

**{**

**q=q->next;**

**}**

**q->next=p->next;**

**delete p;**

**}**

**break;**

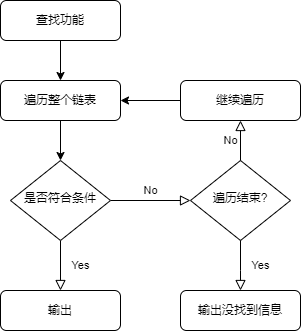
**}**

**p=p->next;**

**}**

**}**

5、更新信息：利用数据域中的name来对品牌或者价格，或者两者都更新，通过调用update\_brand和update\_price进行更新。流程图如下:



代码如下:

**void update\_goods\_data::update\_brand(goods \*headp, const string& name, const string& brand)03**

**{**

**goods \*p;**

**p=headp->next;**

**while(p!=NULL)**

**{**

**if(p->name==name)**

**{**

**p->brand=brand;**

**QMessageBox::information(NULL,QMessageBox::tr("更新结果"),QMessageBox::tr("品牌更新成功"));**

**break;**

**}**

**p=p->next;**

**}**

**if(p==NULL) QMessageBox::information(NULL,QMessageBox::tr("未更改"),QMessageBox::tr("没有找到商品"));**

**}**

**void update\_goods\_data::update\_price(goods \*headp, const string& name, float price)**

**{**

**goods \*p;**

**p=headp->next;**

**while(p!=NULL)**

**{**

**if(p->name==name)**

**{**

**p->price=price;**

**QMessageBox::information(NULL,QMessageBox::tr("更新结果"),QMessageBox::tr("价格更新成功"));**

**break;**

**}**

**p=p->next;**

**}**

**if(p==NULL) QMessageBox::information(NULL,QMessageBox::tr("未更改"),QMessageBox::tr("没有找到商品"));**

**}**

**update\_goods\_data::update\_goods\_data(QWidget \*parent) :**

**QDialog(parent),**

**ui(new Ui::update\_goods\_data)**

**{**

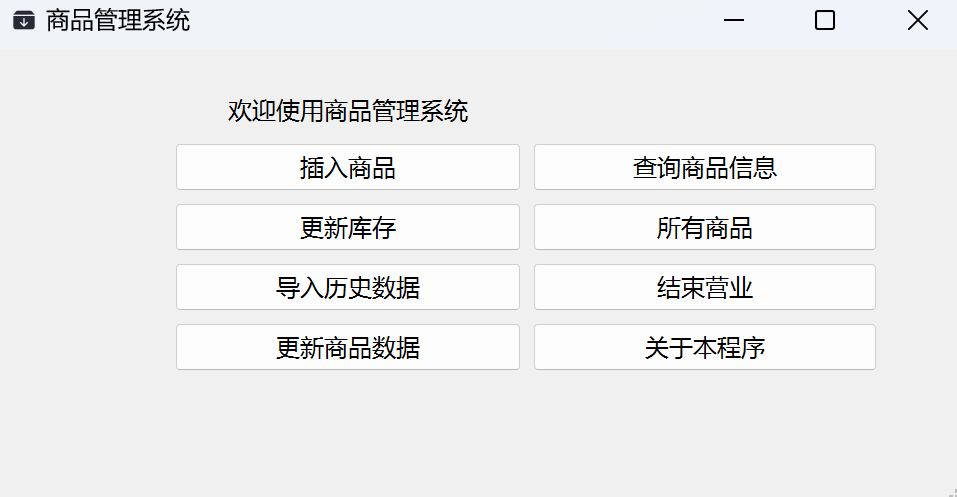
**ui->setupUi(this);**

**ui->brand->setEnabled(false);**

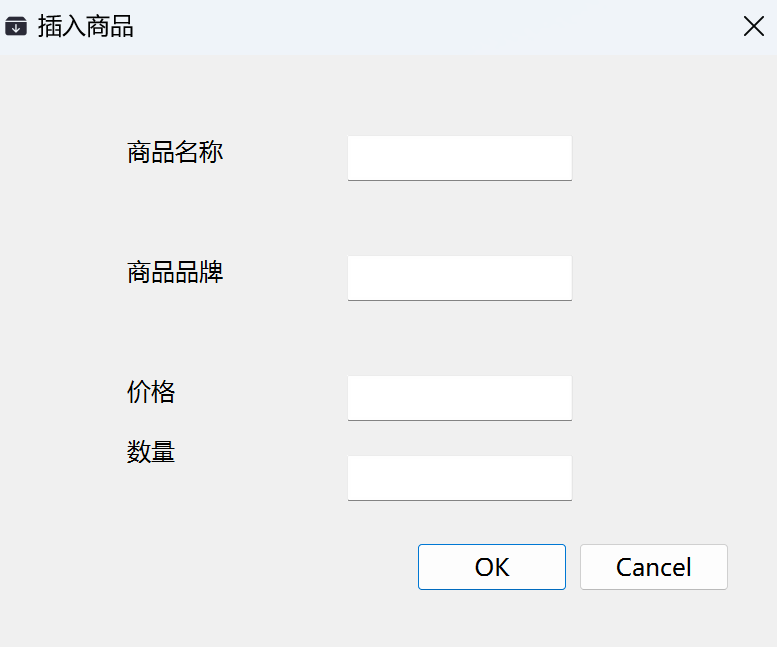
**ui->price->setEnabled(false);**

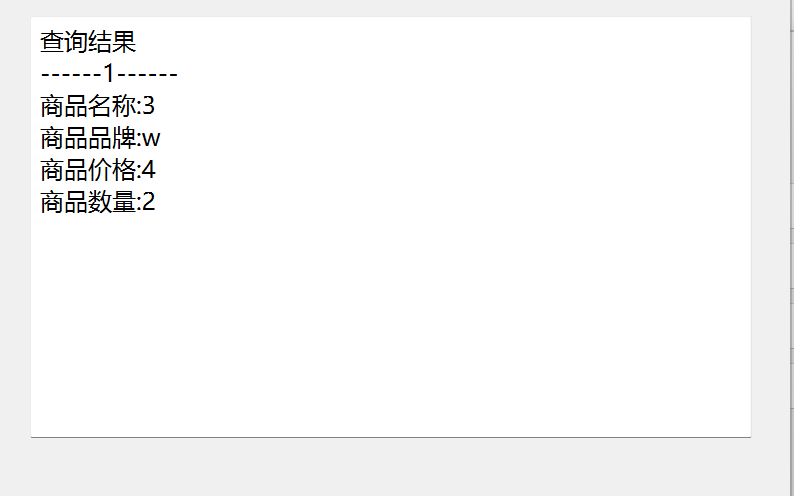
**}**

1. 程序测试

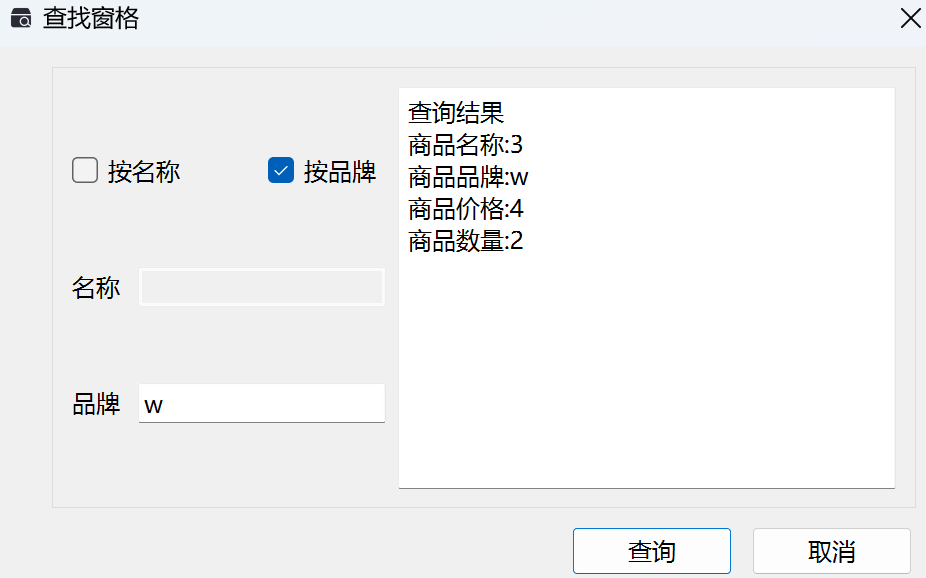


具有windows风格 的主界面，可以通过插入商品对商品进行插入

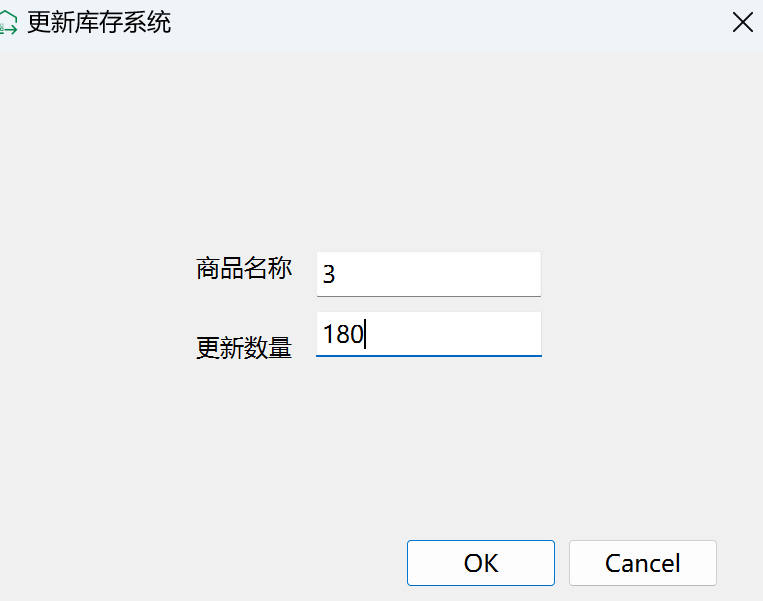




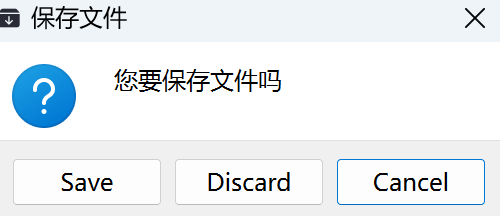
可以通过查询所有对所有值进行观察，发现可以插入正确。



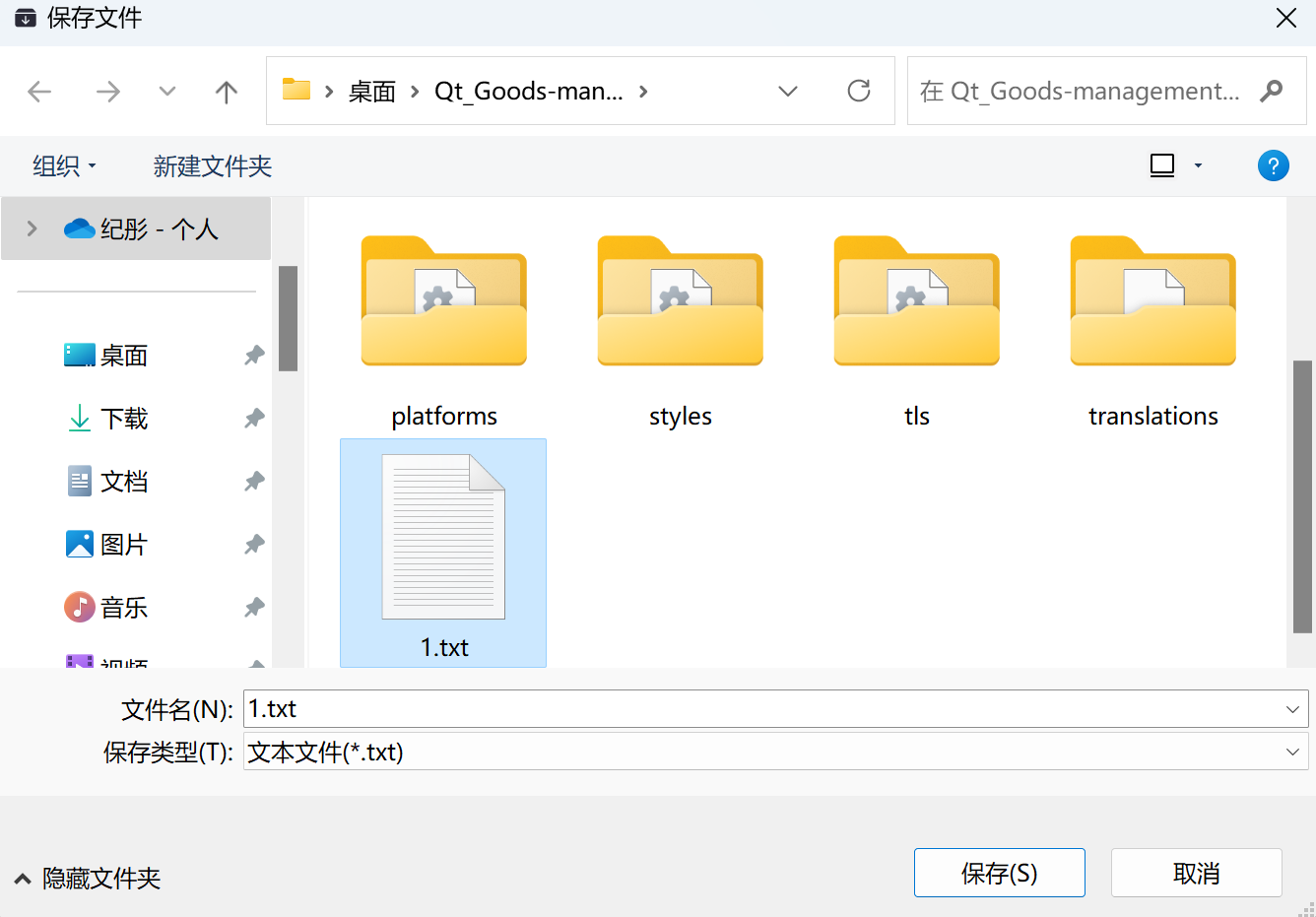
查询窗口提供两种查询方式，分别是按照品牌和名称查询商品



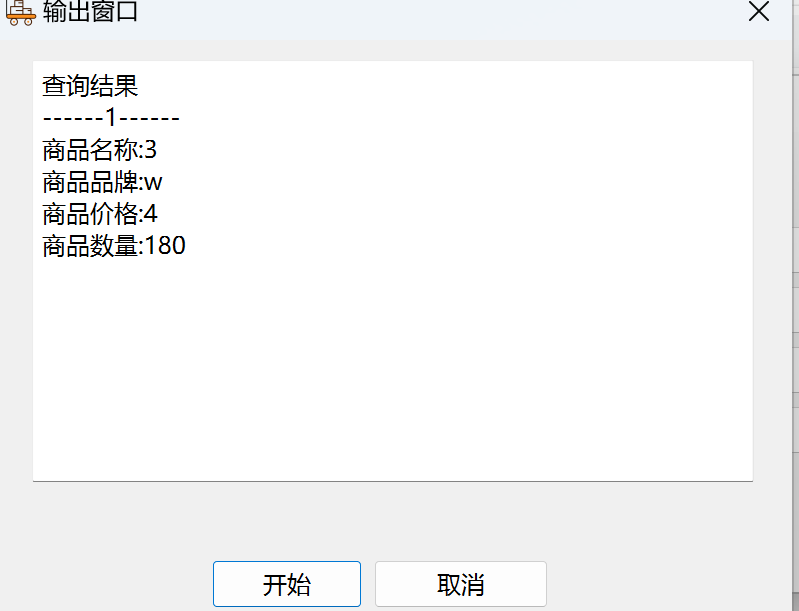
更新库存提供了一个界面，用于更新商品数量



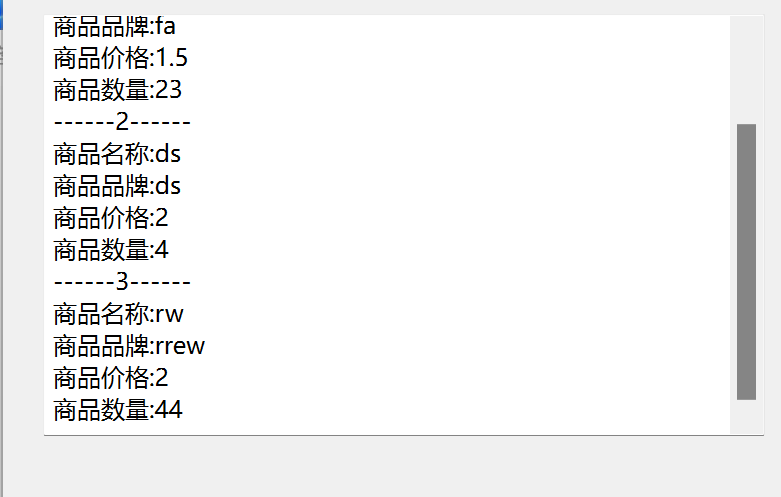
结束营业的话，可以save文件于路径位置，存放文件



打开文件并通过导入历史数据可以恢复原有数据。

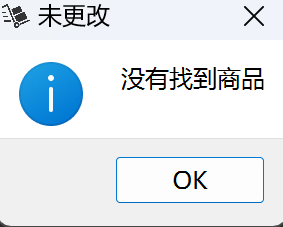


据此可以看出，文件的操作没有问题。

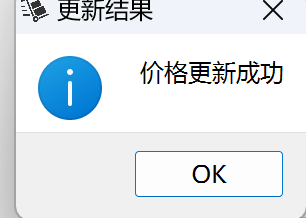


价格按照升序排列整个链表，体现了一种有序性



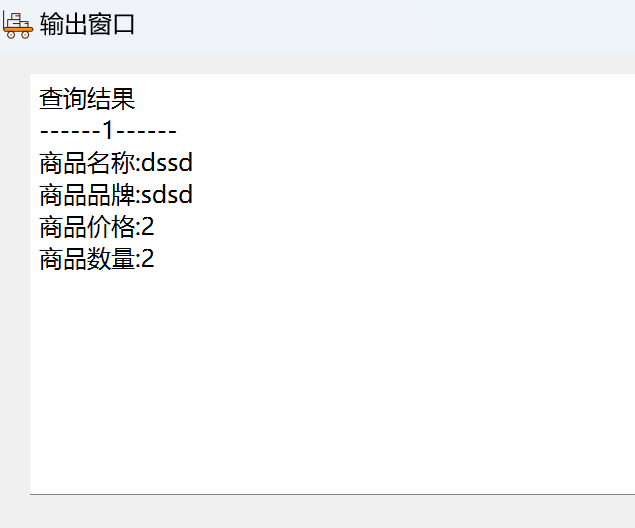


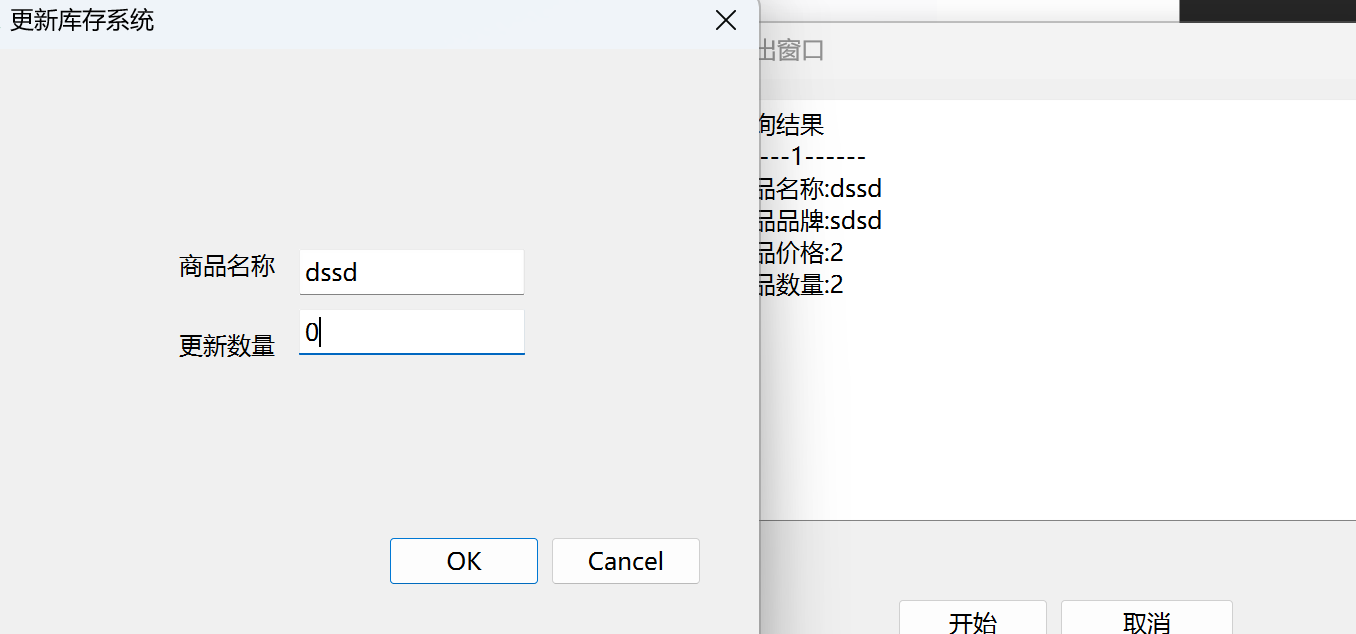
更新商品信息提供了几个窗格，通过名称查询物品，更新品牌和价格。没找到会提示

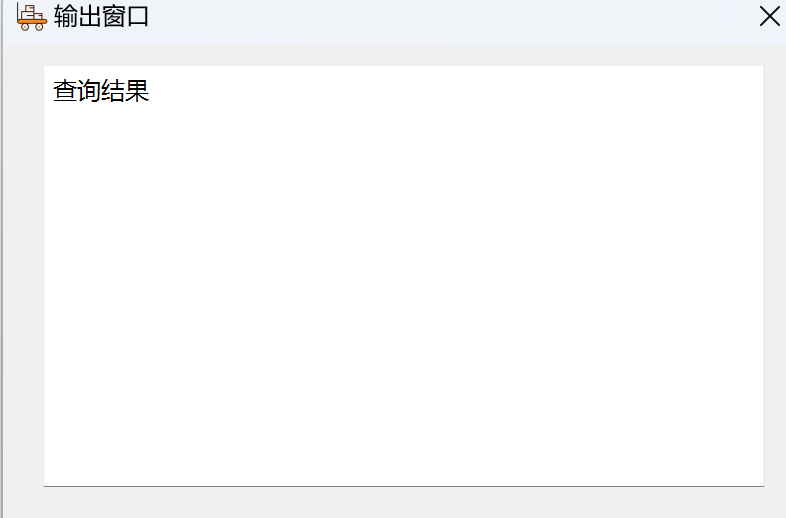


更新成功也会给提示，经测试没有问题

对于库存为0的删除：







查询结果为空，说明程序功能正确。

以上内容是本软件的操作和其操作是否运行的测试案例。

1. 算法分析

本程序使用链表的数据结构进行存储。对于各算法：

插入算法：

在插入数据时，我们首先通过链表操作模块找到插入位置，并按照商品价格升序插入新的结点。该算法的时间复杂度为O(n)，其中n为链表长度。

查询算法：

在查询操作中，我们根据用户输入的品牌或商品名称进行匹配查找。该算法的时间复杂度为O(n)，其中n为链表长度。

更新算法：

在更新操作中，我们需要先找到要更新的结点，然后根据用户输入的新数据进行更新。该算法的时间复杂度为O(n)，其中n为链表长度。

删除算法：

在删除操作中，我们需要先找到要删除的结点，然后将其从链表中删除。该算法的时间复杂度为O(n)，其中n为链表长度。

1. 心得体会

通过本次课程设计，我对链表数据结构有了更深入的了解，掌握了链表的基本操作方法，如创建、插入、删除、查询等。同时，我也学会了如何将链表数据保存到文件中，并从文件中恢复数据。此外，我还发现我对问题理解不够深入、编程技能不够熟练等。这些不足之处需要我在今后的学习和实践中加以改进和提高。

总之，本次课程设计让我收获颇丰，不仅提高了我的编程技能，还让我更加深入地理解了链表数据结构的原理和应用。同时，我也意识到自己在编程方面还有很多需要提升的地方，需要不断学习和实践才能不断提高自己的编程能力。

1. 参考文献

[1] 严蔚敏，李冬梅，吴伟民. 数据结构（C语言版 第2版）[M]. 北京：人民邮电出版社，2011.

[2] 王维波，栗宝鹃，侯春望. Qt6 C++开发指南 [M]. 北京:人民邮电出版社，2023.