西北农林科技大学信息工程学院
面向对象程序设计实习报告

题　目：超市进销管理系统设计与实现

学　　号	2019013121
姓　　名	林炅
专业班级	计算机科学与技术 1903 班
指导教师	王泽鹏
实践日期	2020 年 8 月 24 日—9 月 5 日

目　　录

一、 综合训练目的与要求

(1) 目的

1) 通过一学期对于面向对象程序课程的学习，理解面向对象程序设计的思想，了解面向对象程序设计的几大特点，抽象性，封装性，继承性，和多态性。并运用所学的思想，进行软件的开发实现，并能够完成解决一些实际问题

2) 通过团队协作，采用面向对象程序设计的思想(类的设计，抽象，封装，包含与继承，多态性，UML 图)进行项目开发。

3) 通过C/C++编程，面向对象分析与设计，增加软件的健壮性，使程序的重用性、可维护性等等更强。结合Qt 中的一些类，进行界面上的设计，充分利用Qt 信号与槽的机制实现界面与槽函数直接的链接，实现界面与类之间的使用。

(2) 要求

采用面向对象程序设计思想设计Good 类，并在类中申请商店的各个私有属性，并设计成员函数用于调取和设置类中的私有成员。基于Good 类，实现超市进销管理系统。通过将商品名称与Good *关联，实现在系统中完成对超市商品数据信息的新增、删除、查找、修改等等操作，并实现打印库存，打印当日流水清单，打印购物清单等等操作.

二、 综合训练任务

(1) 综合任务

1) 采用面向对象程序设计思想设计Good 类，并在类中申请商店的各个私有属性，并设计成员函数用于调取和设置类中的私有成员。基于Good 类，实现超市进销管理系统.

2) 实现登录界面，对输入正确用户名及密码的用户实行通过进入超市进销管理系统。

3) 通过QMap 完成对超市进销系统商品的管理，实现能够新增商品，并对其进行保存。

4) 实现对超市库存中的商品进行库存商品管理，实现对库存的展示，对库存中商品的查找，删除，修改，以及反馈库存中缺货商品名称。

5) 实现超市销售管理，记录并保存一天的流水清单，并能够计算当日的营业额。

6) 实现用户购买界面，允许用户将商品加入购物车，并完成结算，打印小票购物清单操作

(2) 个人任务

1) 通过与队友讨论，两人一起通过功能与需求的分析得到软件基本上所需要的功能和界面的大致布局设计，并以此作为参考，辅助队友进行主界面的搭建。

2) 采用面向对象思想设计，运用STL 库中的QMap 键值对用于存储商品信息Good*，以便于管理商品库存。

3) 设计槽函数，负责与队友设计的ui 界面连接，用QMap 的功能实现键值对的查询，删除，新增，查找功能,以及对商品流水情况和收入情况进行输出。

4) 设计文件处理系统，实现能够对库存商品总信息，购物清单信息，当日流水清单的保存。

5) 设计登录界面，将登录界面QWidget 与QMainWindow 主界面进行连接，实现登录完成后跳转至主界面。

三、 总体设计

主要模块分为四块：

1) 文件处理模块：主要负责文件的写入，以及界面上相应文件功能的实现（保存），分别完成对用户购物清单，库存商品信息，和当日流水情况的打印保存。

2) 价格计算模块：用槽函数与QPushButton 连接实现能够计算出超市当日营业额计算以及每个用户购物车商品的价格计算。

3) 商品管理模块：完成新增商品，删除商品，修改商品，查找商品。

4) 商品信息展示模块：展示库存商品信息，展示本次商品进销信息，展示用户购物车商品信息。

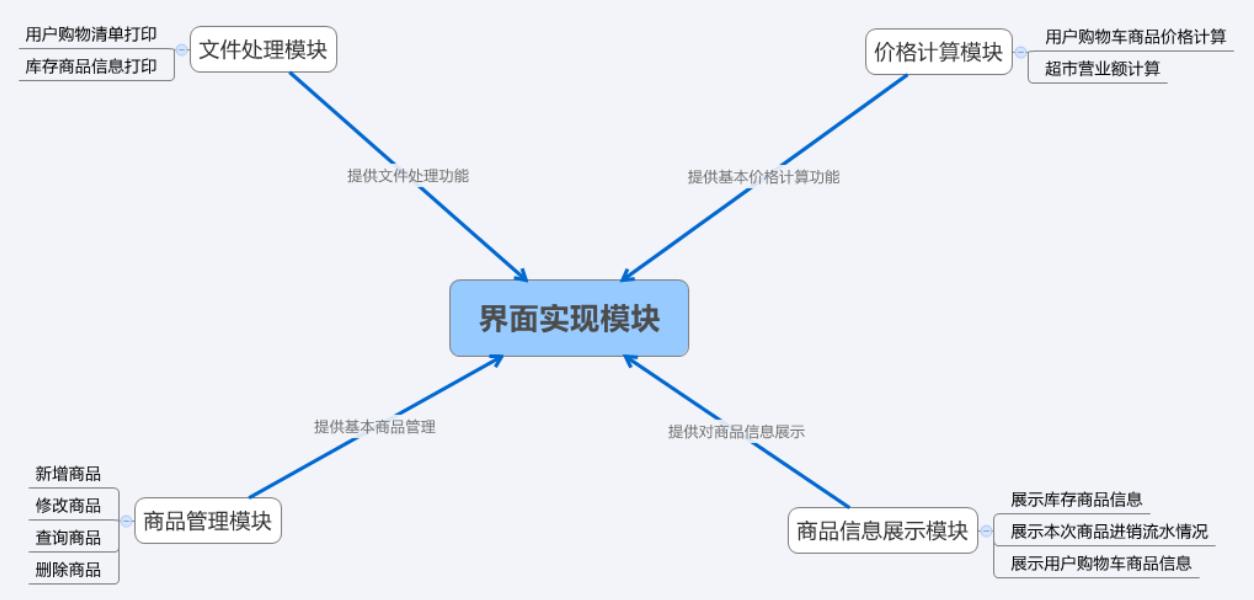

图1 模块设计图

负责的模块：

1) 商品的库存管理功能的设计和实现，通过STL库中的QMap 来实现对库存中的商品的查看，查找，修改以及删除

2) 商品的销售管理功能的设计和实现，通过对新增商品和卖出商品时的操作进行保存记录，并查看当日超市营业的总收入

3) 超市的登录界面的设计和连接，设计登录界面，并将其与MainWindow 主窗口进行连接，输入正确后转入主窗口。

4) 用户购物清单，库存商品信息以及超市进销库存流水情况打印的功能的实现。

① 商品的库存管理功能的设计和实现

对商品的库存管理，我分为了QMenu菜单栏中的一个库存管理选项，在其子类的QAction中，我们增加了库存查看和信息修改两个功能。这个功能的设定，就是为了方便完成对在库存中的商品信息的管理，能够实时查看当前商品的内容，以及对这些内容进行修改。

总体的设计思路为，运用QMap的成员函数QMap.find、QMap.erase、QMap.value实现查询QMap内容，删除键值对，获取商品数据，通过与LineEdit结合，通过按钮连接槽函数实现，完成对商品数据的管理。运用QMap的迭代器，实现对QMap的遍历，并逐个在QTableView中输出展示。同时，为了防止出现缺货现象，在我们的超市进销管理系统中，新增了一个缺货查询功能，当使用QMap中的查询功能，查询到某件商品的数量减为0时，系统提示该商品已经缺货，建议补货操作。

② 商品的销售管理功能的设计和实现

对于超市的销售管理功能，我分为了QMenu菜单栏中的一个销售管理选项，在其子类的QAction中，新增加账目功能。功能的设定，方便用户能够查看到超市库存的进销情况，能够看到进货，卖货的各项数据。以及对商品总计的购物价格的统计总和，计算出营业总额。

总体实现的设计思路如下，库存进销情况展示分为了入库展示和出库展示，分别在入库时，将入库信息打印到本界面的QTextEdit 中，同理在购物车点击结算后，输出出库商品以及其信息到本界面的QTextEdit 中。这样的操作需要创建一个全局变量，在两个不同的槽函数中保存库存的进销情况。

③ 超市的登录界面的设计和连接

对于超市的登录界面功能的实现，在界面中创建了用户名与密码以及对应的输入框QTextEdit，分别再创建关闭和登陆两个按钮完成界面的设计。修改一些界面QTextEdit的属性，使之更加符合真实登录环境。

界面的总体的实现和连接如下，创建的是一个新的ui界面，并在界面中放入想要的组件，并将按钮与槽函数连接，点击关闭按钮时，槽函数使创建的ui窗口界面关闭。点击登录便判断条形框中输入是否与程序预设的用户名和密码是否相同。相同则关闭当前登录界面弹出登录成功提醒并且打开主窗口，否则要清空QTextEdit，并报出失败登录提示，等待重新输入直到正确。

④ 文件处理打印功能的设计与实现

对于文件处理打印功能的设计，我们分别使用在了三个方面，对库存管理中的库存商品情况的清单打印，对销售管理的商品进销情况的清单打印，以及对结算界面的所购买的商品进行清单打印

文件处理打印功能的总体实现情况如下，使用QT中的QFileDialog类来完成点击按钮跳出另存为界面，用其中的getSaveFileName(this,”保存”,QDir::currentPath(),”文本文件(*.txt)”

);成员函数实现特定保存文件位置，调出设置文件文本框，期间还要判断存放的商品名字不能为空。接下来，使用QFile来保存文本框中的内容至所规定的文件中，创建文本输出流QTextStream，分别调取库存管理中的库存商品情况、销售管理的商品进销情况以及结算界面的所购买的商品购物车至所创建的输出流中。由于回车指令有些不同，需要在文本中做些替换操作。结束读入后，关闭文件清除QFile，释放内存，提示输出成功即可。

四、 详细设计说明

Qt界面的设计与实现模块的详细说明：

(1) 第一个部分：商品库存管理功能的设计和实现

通过对项目的需求分析，然后通过进一步的功能的调整与确认，分析出所需要的功能为：库存查看，和信息修改，其中商品信息修改中包含对给定商品的查找，修改以及删除。在队友实现的商品入库后，完成对QMap中所保存的内容进行各种操作。

① 信息修改

信息修改中包括对给定的商品名称，用QMap中的find函数查询到商品的所有保存信息，并可以选择对已保存的商品信息展示在lineedit中进行修改，并确认修改，或者直接在QMap 中删除该项商品所存在信息。分别使用map.find、map.erase以及直接对map对应点Good*类中的成员用set进行修改具体实现如下：代码清单1、2、3

代码清单1：查询代码

```
void MainWindow::on_findbtn_2_clicked()
{
  if(shop.find(ui->goodname_2->text())!=shop.end())
  {
      ui->barcode_2->setText(shop.find(ui->goodname_2->text()).
         value()->getcode());
      ui->amount_2->setText(number);
      ui->inprice->setText(inprice);
      ui->outprice->setText(outprice);
      ui->sup_2->setText(shop.find(ui->goodname_2->text()).value()
         ->getsupplier());
      QMessageBox::information(this,"查询提示","已为你查询到商品全
         部信息");
  }
  else
  {
      QMessageBox::information(this,"查询提示","库存中没有该商品");
  }
}
```

代码清单2：删除代码

```
void MainWindow::on_deletebtn_clicked()
{
    shop.erase(shop.find(ui->goodname_2->text()));
    ui->goodname_2->clear();
    ui->barcode_2->clear();
    ui->amount_2->clear();
    ui->inprice->clear();
    ui->outprice->clear();
    ui->sup_2->clear();
    QMessageBox::information(this,"修改提示",ui->goodname_2->text()
        +"商品已被删除");
}
```

代码清单3：修改代码

```
void MainWindow::on_changebtn_clicked()
{
    shop[ui->goodname_2->text()]->setcode(ui->barcode_2->text());
    shop[ui->goodname_2->text()]->setnumber(ui->amount_2->text().
        toInt());
    shop[ui->goodname_2->text()]->setinprice(ui->inprice->text().
        toDouble());
    shop[ui->goodname_2->text()]->setoutprice(ui->outprice->text().
        toDouble());
    shop[ui->goodname_2->text()]->setsupplier(ui->sup_2->text());
    QMessageBox::information(this,"修改提示",ui->goodname_2->text()
        +"商品信息已经修改");
}
```

② 库存查看

库存查看允许用户查看目前在QMap中保存的所有商品信息，用迭代器it遍历QMap便能得到全部商品信息输出，呈现商品信息使用的QT部件QTableView以表格形式呈现。并当某件商品数量为0时，输出展示当前缺货商品，方便管理员进货。

创建TableView表格，在头文件创建QTableView*表格视角和QStandardItemModel*标准模型项指针。并声明inittable函数用于生成连接表格。在函数中，添加表头，准备数据模型，利用setModel()方法将数据模型与QTableView绑定。随后修改表格的部分属性以满足产品需求。见代码清单4

代码清单4：表格创建

```
void MainWindow::inittable()
{
    table_view = ui->tableView;
    item_model = new QStandardItemModel(8,4);   // 8行6列
```

```
    table_view->setModel(item_model);//两者关联
    QStringList column, row;
    column << "商品名称" << "条形码" << "商品数量" << "商品进价" <<
        "商品售价"  << "供应商" ;
    row << "1" << "2" << "3" << "4";
    item_model->setHorizontalHeaderLabels(column);
    item_model->setVerticalHeaderLabels(row);
    table_view->setEditTriggers(QTreeView::NoEditTriggers);
}
```

将QMap中存放的键值对用QMap迭代器进行遍历，从map.begin()遍历到map.end()将查询到的每个商品信息按行存入标准数据模型，并与表格相连接，按行展示信息。每次展示完当前库存后，想要下次继续完成展示时，将上次所放入表格中的所有信息清除后重新用QMap迭代器进行遍历查询存入标准数据模型，否则将会出现两次展示的商品库存信息重叠，出现表格数据异常的情况。见代码清单5

代码清单5：库存信息录入表格

```
void MainWindow::on_showbtn_clicked()
{
    QMap<QString ,Good*>::Iterator itt;
    int i=0;
    for(itt=shop.begin();itt!=shop.end();itt++)
    {
        item_model->setItem(i,0,new QStandardItem(QString("%1").arg
            (itt.value()->getname())));
        item_model->setItem(i,1,new QStandardItem(QString("%1").arg
            (itt.value()->getcode())));
        item_model->setItem(i,2,new QStandardItem(QString("%1").arg
            (itt.value()->getnumber())));
        item_model->setItem(i,3,new QStandardItem(QString("%1").arg
            (itt.value()->getinprice())));
        item_model->setItem(i,4,new QStandardItem(QString("%1").arg
            (itt.value()->getoutprice())));
        item_model->setItem(i,5,new QStandardItem(QString("%1").arg
            (itt.value()->getsupplier())));
    i++;
    }
    QMessageBox::information(this,"提示","已为你显示全部库存信息");
}
```

(2) 第二个部分：商品销售功能的设计和实现

通过对项目的需求分析，然后通过进一步的功能的调整与确认，分析出所需要的功能为：库存进销流水情况展示，和超市当前总营业额展示。库存进销流水情况展示，让用户能在textedit中得到所有的进销操作，包括商品进货和商品卖出，实现方法即是，在队友完成的商品入库槽函数中新增函数，在入库时向textedit中加入本次操作信息。同理，在购物车

结算时，完成购物后，向textedit中加入本次出货商品以及操作信息。设置全局变量 QString water和int count=1 用来记录存放当前的商品流水情况，以及当前的操作次数统计代码见代码清单6

代码清单 6： 库存进销流水情况展示

```
QString waterl;
int count=1;
void MainWindow::on_input_clicked()
{
    QString str;
    str=QString("第%1次操作 ， 进货%2 ， 数量%3 ， 付款%4元").arg(
       count).arg(ui->goodname->text()).arg(ui->amount->text().
       toInt()).arg(ui->inPrice->text().toDouble()*ui->amount->text
       ().toInt());
    water+=str+"\n";
    ui->list_2->setText(water);
    count++;
    QMessageBox::information(this,"入库提示",ui->goodname->text()+"
        已成功新增");
}
void MainWindow::on_paytotal_clicked()
{
    QString sb;
    sb=QString("第%1次操作 , ").arg(count);
    water+=sb;
    QString spr;
    QMap<QString ,Good*>::Iterator it;
    for(it=car.begin();it!=car.end();it++)
    {
        spr=QString("售货%1 ， 数量%2 ， 收款%3元").arg(it.value()->
           getname()).arg(it.value()->getcarnumber()).arg(it.value
           ()->getcarnumber()*it.value()->getoutprice());
        water=water+spr+"\n";
    }
    ui->list_2->setText(water);
}
```

超市当前总营业额展示，实现展示从程序开始到点击按钮的这个时间段，用户购买系统所收到的钱，也即是超市总营业额。想要完成这个功能，要在MainWindow全局中创建double 变量money 用于储存当前的收入总量。在购物车完成结算的函数中新增将结算的钱加入到money中，最终实现点击按钮输出money变量的数值的操作。

(3) 第三个部分：超市的登录界面的设计和连接

出于对超市进销管理系统安全性的保证，我们选择了新增一个超市用户管理系统来实现输入密码完成对超市进销管理系统的登录操作。

实现超市的登录界面的设计，首先新建了一个ui界面，对ui界面进行加工完善，设置

用户名和密码的LineEdit的placeholdertext，使得用户能被更好的提示到输入用户名和位置。与此同时，在密码的LineEdit中，将他的echoMode属性改为password密码，模拟正式的登录系统。完成ui界面的设计处理之后，完成槽函数的设计以及按钮和槽函数的连接，点击取消按钮使用this->close()用于关闭窗口。想要成功登陆应输入正确用户名以及密码，关闭原有的login小窗口，并打开MainWindow类型的大窗口，实现界面的切换。具体代码实现如代码清单7

代码清单7：登录界面实现与连接

```
void login::on_pushButton_clicked()
{
    QString username = ui->useredit->text();
    QString password = ui->passedit->text();
    if(username=="admin" && password=="123456")//用户名以及密码设置
    {
        QMessageBox::information(this,"登录提示","登录成功");
        this->close();
        MainWindow * t=new MainWindow;
        t->setWindowTitle("商店管理系统");
        t->show();
    }
    else
    {
        QMessageBox::information(this,"登录提示","用户名或密码错误"
            );
        ui->useredit->clear();
        ui->passedit->clear();
    }
}
```

(4) 第四个部分：文件处理打印功能的设计与实现

通过对项目的需求分析，我们认为，我们仍然需要一个文件处理功能，用于实现对库存商品信息，用户购物清单，以及对超市进销库存流水情况的信息进行打印和保存。

针对这个项目，我选择了使用QT中的QFileDialog类，用来完成选择文件夹，用类中的成员函数getSaveFileName(this,”保存”,QDir::currentPath(),”文本文件(*.txt)”);实现调出设置文件对话框，并用判断防止文件名字为空。随后便可以向文件中保存数据，使用QFile类申请新的成员，并做修改成员文件名并将其使用只写方式打开。创建文本输出流QTextStream out(file)，分别取库存商品信息，用户购物清单，以及超市进销库存流水信息的文本内容，放入out输出流中。由于在win系统中回车指令不再是”\n”,而变为了“\r\n”，因此使用替换代码，将文本框中的回车全部替换为win系统记事本中所能识别的回车操作。结束读入后，完成关闭文件，并清除申请的QFile,来释放内存，并用QMessageBox输出保存成功提示。具体实现代码如代码清单8

代码清单8：查询代码

```
void MainWindow::on_printbtn_clicked()
{
    QString filename = QFileDialog::getSaveFileName(this,"保存",
       QDir::currentPath(),"文本文件(*.txt)");
    if(filename.isEmpty())
    {
        QMessageBox::information(this,"提示","请输入正确的文件名");
    }
    QFile *file =new QFile;
    file->setFileName(filename);
    bool ok =file->open(QIODevice::WriteOnly);
    if(ok)
    {
        QTextStream out(file);
        QString text = ui->information->toPlainText();
        text.replace(QString ("\n"),QString("\r\n"));
        out<<text;
        file->close();
        delete file;
        QMessageBox::information(this,"提示","保存成功");
    }
    else
    {
        QMessageBox::information(this,"提示","保存失败");
    }
}
```

五、 测试与调试

登录界面，运行程序进入登录界面，输入正确的用户名和密码，切换至主界面。如图2

图2 登录界面

库存浏览界面，点击查看库存便能显示所有库存商品信息，点击打印库存清单实现保存库存商品信息至指定目录。点击查找按钮，查询所有缺货商品。如图3

图3 库存查看

信息修改界面，通过输入想要查找商品名称，查找到商品所有信息，对LineEdit修改后点击修改按钮进行商品信息修改，若想要删除商品则点击删除商品。如图4

图4 信息修改

进入销售情况界面就能在TextEdit中查看到近期的商品进出库情况展示，点击打印销

售清单便能成功保存打印。点击统计后便能查看当前商品的总营业额。如图5

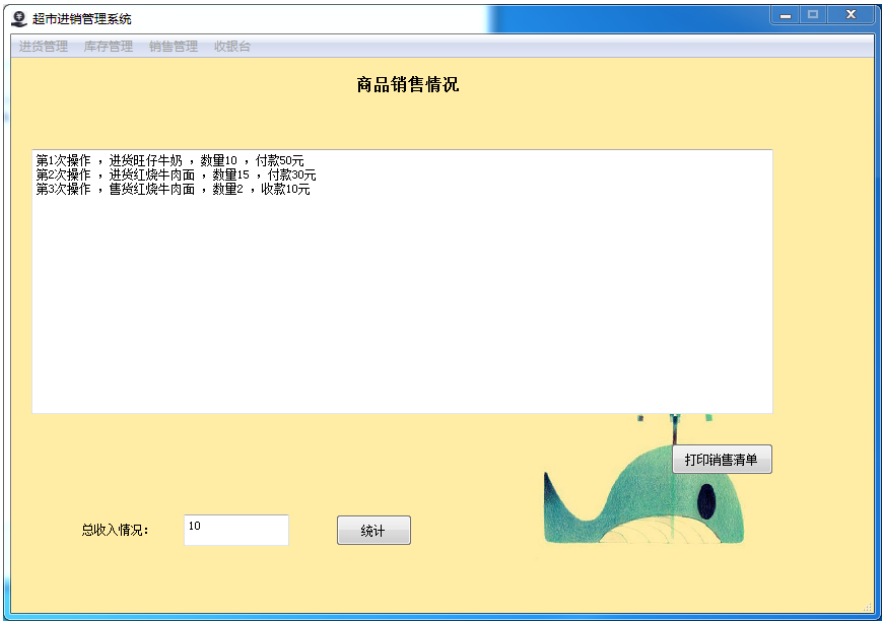

图5 销售情况

六、 实习日志

(1) 8月24日

主要任务：确定项目总体设计与模块划分

主要进行了如下工作：

1) 需求分析，分析超市进销管理系统所需要的功能

2) 对系统的界面进行一个大致的框架设计

3) 所使用的工具进行确认，使用c++语言下的QT，使用gitee代为管理代码，并用LaTeX代为编写实习报告.

4) 类图的分析与设计

我们选择的题目是超市进销管理系统，基于上学期期末选择的书店管理系统，我们这次选择这超市进销管理系统，准备在上次的基础上进行一些提高，让功能的实现更加立体化，并还原真正的超市管理系统。第一天我们选择不着急开始敲写代码，毕竟磨刀不误砍柴工，我们选择先开始对QT进行一些学习和了解，并在老师的指导下学习用git来完成分工协作，将自己的代码上传，更新远端的代码，将团队成员所做的贡献获取下来。最后我们进行了对我们管理系统类图的分析和设计。

(2) 8月25日

主要任务：完成类的设计以及界面的框架实现

主要进行了如下工作：

1) 对类的基本分析以及类的设计和实现

2) 继续对超市进销管理系统的功能和实现进行讨论并确定方案

3) 设计和确定ui主界面，设计stacked widget的各个界面的总体框架布局规划

今天我们就开始了正式的项目启动了，首要的是对商品类的设计，确定了商品的几大必要的基本属性以及成员函数，并予以实现。接着继续对系统的功能进行了讨论，最终确定了有进货管理，库存管理，销售管理，收银台，四个功能区域，并初步设计了使用stacked widget分开对每个模块进行设计，并完成两人的分工。

(3) 8月26日

主要任务：创建和完善用户登录界面

今天的主要任务就是完成用户登录界面，在网络上搜索，有许多个版本的登录界面的设计方案，在其中进行有效的挑选后。第一次的选择出现错误，在信号与槽的发射方面还是不太精通，因此选择了一个更加有效和易懂的登录界面，参考还原真正的管理系统，我们创建了这样一个用户登录界面，并修改了其部分自带属性，来使其更加真实的模拟真实登录界面。

(4) 8月27日

主要任务：完成槽函数的设计(主要是对商品库存管理界面的功能实现)

今天的主要任务是完成商品库存管理界面的实现，由于使用了QMap的STL 库，我边上网查找了许多关于c++的STL库中的map的资料进行学习，使用了QMap中的许多成员函数进行操作，结合队友完成的这个模块的ui界面进行连接操作，经过多方面的比对后，最后选择用QTableView这个组件来代替原本的文本框来盛放保存库存商品的信息，这样使得我们的系统在展示库存商品情况方面更加立体化和形象化。于是在下午又对QTableView进行研究，研究他的使用方法，参考网上的代码，理解了很长时间。虽然十分复杂，但是不断梳理还是明白了用QTableView 和QStandardItemModel 进行连接，用QStandardItemModel盛放数据，QTableView盛放表格的基本形式。

(5) 8月28日

主要任务：完成槽函数的设计(主要是对商品销售管理功能的设计和实现)

今天的主要任务是完成销售管理系统界面的设计，经过讨论，将当日的流水清单进行展示还是需要在各个功能实现的情况下，进行新增，于是我在git上下载了队友的成果，并在进货管理和购物车的结算环节加入了一定的成员函数，使得当日的流水清单能够正常的打印。并设计了一个计算当日营业额的小函数插入其中。

由于程序基本的框架已经浮现，我便进行了一次大整合，将我们两个所完成的部分的代码在git上进行比对和查看。完成整合后，我们对管理系统进行了小小的调试和讨论，对其中发现的一些小小的不合理的bug进行了修复，如进货名称不能为空，总计金额的LineEdit要设置为不可写入，等等。除此之外，我们也对下一步的代码编写做了一些总结，新增了

一个文件处理功能由我来实现，一个ui界面的美化由队友来实现

(6) 8月31日

主要任务：完成文件处理的功能模块

今天最主要的任务是完成文件的处理功能，使我们的系统能够将库存商品信息，销售商品流水情况以及购物清单保存文件操作。开始的时候使用的是QFile 以及文本输出流QTextStream 分别取库存商品信息，用户购物清单，以及超市进销库存流水信息的文本内容，放入out输出流中输出。可是由于这样的操作导致存放的文件的位置固定，无法特定保存，因此又上网查阅资料寻找了解决的方法，使用了QT中的的QFileDialog 类，用来完成选择文件夹，用类中的成员函数getSaveFileName实现调出设置文件对话框。结果又出现了输出内容保存进文件后出现所有内容保存在一行之中，输出不了换行。经过长时间的检查和上网查问，发现win系统的回车符号有所不同。在修改后就完成了文件打印功能的实现。

(7) 9月1日

主要任务：程序的大测试以及课程报告的LaTeX初步编写

今天的任务就是对我们合作完成的超市进销管理系统进行一次大复查，检查系统中不合理的，出现问题的部分。对产生问题的部分进行修改。对ui界面的背景等等进行调整。对一些临界点，极值点，进行测试和调试，保证代码程序不会出现漏洞bug。

与此同时，展开对课程实习报告的LaTeX编写，让自己重新梳理对这个系统的框架，整理出框架便于自己有条不紊的完成对课程论文报告中自己完成的部分进行写作

(8) 9月2日

主要任务：程序的借鉴以及可测和那个报告的正式编写

今天的任务是借鉴网络上和同学们程序代码中用的好的部分进行学习和借鉴，丰富自己的代码，和一些小小的贴心的小细节。

同时正式开始对课程报告进行书写，完成论文的大部分自己负责的部分，并开始对自己这两周的实习进行总结和阐述。

(9) 9月3日

主要任务：准备答辩PPT，以及准备答辩的内容

今天，针对所完成的超市进销管理系统进行重新回顾，和队友确认负责部分以及分别讲对这次实习所完成部分的细节。接下来就是对自己答辩部分进行熟悉，确定自己答辩的方向

(10) 9月4日

主要任务：进行实习答辩，继续完善论文部分

今天开始完成答辩，向老师介绍自己的项目以及自己所负责的部分的实现。和老师进行良好的沟通，老师也给我们提供了许多改进的思路和方法，并分析了我们目前出现的错误和原因，为我们未来完善这个软件提供了帮助。

七、 实习总结

本次为期两个星期的实习是我感触比较深的实习，由于春季学期正值疫情，我们在家中完成了面向对象程序设计这门课程的学习，由于在家中学习，并没有很高的学习效率。因此C/C++编程水平仍有欠缺，在上学期结课时选择了书店管理系统，勉勉强强完成了管理系统的设计。这次夏季学期选择的超市进销管理系统，相比于书店管理系统，类的成分会减少些许，但是在功能方面却有所增加。我和我的队友，在上学期的书店管理系统设计的背景下，学习了我当时运用的较好的Qmap 键值对用来保存商品信息，方便数据的遍历。也总结了当时书店管理系统所存在的问题，在长达两个星期的调试中，一次一次的改进代码，优化程序，清除bug.

还有与上次大作业不同的是，这次我们采取了两人合作的形式完成超市进销管理系统的设计。虽然说人多力量大，但是在两个的协调分工合作方面，我们也确实产生了一点问题。由于不停的产生代码的修改，两个人的代码更新就较为困难。在老师的建议下，我们选择使用了gitee这个平台进行协作合同，并早在实习前将其弄懂。每个人通过pull，fetch 等等操作将自己的代码上传，更新远端的代码，用这个平台管理我们自己的代码。我也负责在其中将两人的代码进行整合管理。

对于QT这个C/C++用户开发框架，刚上手是确实不适应。在一段时间熟悉后，我们发现，Qt中信号与槽，是他脱颖而出的最大关键，通过这个特性，我们可以实现一系列的连接操作。除此之外，Qt中部分内置的类也有许多强大的功能，通过使用这些类或者继承这些类，然后完成我们自己类的设定，方便了我们程序的开发以及设计。

在这两周的实习过程中，必然的，我们也遇到了许多问题。如何掌握一个全新不认识的类，通过向学长，向同学，老师请教，我逐渐开始明白了怎么剖析一个类，查询它的父类子类，查询它的共有属性和专属的特殊属性，函数，信号槽。通过这次实习中，我们所克服的这些困难，我们也积累了许多的经验，我收获了许多，不仅仅是知识层面上的，比如说本次实习在代码上采用的是Qt设计，报告上采用的是LaTeX，在代码管理方面采用的是Git。更多的是一种经验，让自己知道，再大的困难也能克服。

虽然本次实习选择的课题难度并不是很大，但是我认为，能把简单的程序做到最优化，做到最精致，不停地反复调试，难道不应该也是一件有挑战的任务吗？只有不断反问自己，还能更优化代码了吗？还能再新增功能了吗？做到精益求精，我认为未尝不是一个挑战。

八、 附录：核心代码清单

本项目的代码上传至Gitee网站管理，项目地址：https://gitee.com/CH3OCH3/c-internship

接下来展示负责部分的核心代码。

由于部分代码已在上文介绍功能实现中展示，不做过多赘述如代码清单12345678

代码清单9: login.h/login.cpp

```
login.h
#ifndef LOGIN_H
#define LOGIN_H

#include <QWidget>

namespace Ui {
    class login;
}

class login : public QWidget
{
    Q_OBJECT^^I
    public:
    explicit login(QWidget *parent = nullptr);
    ~login();
    void paintEvent(QPaintEvent *);
    private slots:
    void on_pushButton_2_clicked();
    void on_pushButton_clicked();
    private:
    Ui::login *ui;
};
#endif // LOGIN_H


login.cpp
#include "login.h"
#include "ui_login.h"
#include "mainwindow.h"
#include <QMessageBox>
#include<QPainter>
login::login(QWidget *parent) :
QWidget(parent),
ui(new Ui::login)
{
    ui->setupUi(this);
}

login::~login()
{
    delete ui;
}

void login::on_pushButton_2_clicked()
{
    this->close();
}
```

```
49
50 void login::on_pushButton_clicked()
51 {
52     QString username = ui->useredit->text();
53     QString password = ui->passedit->text();
54     if(username=="admin" && password=="123456")
55     {
56         QMessageBox::information(this,"登录提示","登录成功");
57         this->close();
58         MainWindow * t=new MainWindow;
59         t->setWindowTitle("超市进销管理系统");
60         t->setFixedSize(850,570);
61         t->setWindowIcon(QIcon(":/photo/shop.jpg"));
62         t->show();
63     }
64     else
65     {
66         QMessageBox::information(this,"登录提示","用户名或密码错误"
               );
67         ui->useredit->clear();
68         ui->passedit->clear();
69     }
70 }
71 void login:: paintEvent(QPaintEvent *)
72 {
73     QPainter painter(this);
74     QPixmap pix;
75     pix.load(":/photo/u=2413354430,3009708332&fm=26&gp=0.jpg");
76     painter.drawPixmap(0,0,this->width(),this->height(),pix);
77     //  画背景上图标
78     //  pix.load(":/photo/ziti.jpg_w800");
79     //  缩放
80     //  pix = pix.scaled(pix.width()*0.5,pix.height()*0.5);
81     //  painter.drawPixmap(10,30,pix);
82 }
```

此处放出MainWindow.h/MainWindow.cpp中部分未展示代码

代码清单10: MainWindow.h/MainWindow.cpp

```
1 MainWindow.h
2 #ifndef MAINWINDOW_H
3 #define MAINWINDOW_H
4
5 #include <QMainWindow>
6 #include <QMap>
7 #include <good.h>
8 #include <QMessageBox>
9 #include <QSpinBox>
10 #include <QDebug>
11 #include <QFile>
```

```
#include <QFileDialog>
#include<QPainter>
#include <QTableView>
#include <QListView>
#include <QTreeWidget>
#include <QStandardItemModel>
#include <QStandardItem>
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

    public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void paintEvent(QPaintEvent *);
    QTableView * table_view;
    QStandardItemModel * item_model;
    QTableView * table_view_2;
    QStandardItemModel * item_model_2;
    void inittable();
    private:
    Ui::MainWindow *ui;
    QMap<QString ,Good *> shop;
    QMap<QString ,Good *> car;
    private slots:
    void in_();
    void stock_();
    void income_();
    void sale_();
    void change_();
    void on_input_clicked();
    void on_findbtn_clicked();
    void on_pushButton_clicked();
    void on_incar_clicked();
    void on_paytotal_clicked();
    void on_findbtn_2_clicked();
    void on_changebtn_clicked();
    void on_deletebtn_clicked();
    void on_totalbtn_clicked();
    void on_printbtn_clicked();
    void on_printbtn_2_clicked();
    void on_printbtn_3_clicked();
};

MainWindow.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

```cpp
#include <QMessageBox>
#include <QSpinBox>
#include <QDebug>
#include <QFile>
#include <QFileDialog>
#include<QPainter>
#include <QTableView>
#include <QListView>
#include <QTreeWidget>
#include <QStandardItemModel>
#include <QStandardItem>
QString water;
int count=1;
double money=0;

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui->incontrol,&QMenu::triggered,this,&MainWindow::in_);
    connect(ui->stock,&QAction::triggered,this,&MainWindow::stock_)
        ;
    connect(ui->incomecontrol,&QMenu::triggered,this,&MainWindow::
        income_);
    connect(ui->salecontrol,&QMenu::triggered,this,&MainWindow::
        sale_);
    connect(ui->change,&QAction::triggered,this,&MainWindow::
        change_);
    inittable();
}

MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::inittable()
{
    table_view = ui->tableView;
    item_model = new QStandardItemModel(8,4);   // 4行4列
    table_view->setModel(item_model);//两者关联
    QStringList column, row;//行列表头
    column << "商品名称" << "条形码" << "商品数量" << "商品进价" <<
        "商品售价"  << "供应商" ;
    row << "1" << "2" << "3" << "4";
    item_model->setHorizontalHeaderLabels(column);
        // 设置水平表头标签
    item_model->setVerticalHeaderLabels(row);
    table_view->setEditTriggers(QTreeView::NoEditTriggers);


```

```
    table_view_2 = ui->tableView_2;
    item_model_2 = new QStandardItemModel(8,4);  // 4行4列
    table_view_2->setModel(item_model_2);//两者关联
    QStringList column2, row2;//行列表头
    column2 << "商品名称" << "条形码" << "购买数量"  << "商品价格"
        << "供应商" << "总计";
    row2 << "1" << "2" << "3" << "4";
    item_model_2->setHorizontalHeaderLabels(column2);
                        // 设置水平表头标签
    item_model_2->setVerticalHeaderLabels(row2);
    table_view_2->setEditTriggers(QTreeView::NoEditTriggers);
}
void MainWindow::on_findbtn_clicked()
{
    QString list;
    int s=0;
    QMap<QString ,Good*>::Iterator it;
    for(it=shop.begin();it!=shop.end();it++)
    {
        QString str ;
        if(it.value()->getnumber()==0)
        {
            str =QString("%1已经缺货，请及时补货").arg(it.value()->
               getname());
            str=str+"\n";
            s++;
        }
        list=list+str;
    }
    if(s==0) QMessageBox::information(this,"提示","库存中未存在缺货
        商品");
    else QMessageBox::information(this,"提示","以为你展示全部缺货商
        品");

    ui->textEdit->setText(list);
}
void MainWindow::on_pushButton_clicked()
{
    item_model->clear();
    item_model = new QStandardItemModel(8,4);  // 4行4列
    table_view->setModel(item_model);//两者关联
    //table_view->setSortingEnabled(true);
    QStringList column, row;//行列表头
    column << "商品名称" << "条形码" << "商品数量" << "商品进价" <<
        "商品售价"  << "供应商" ;
    row << "1" << "2" << "3" << "4";
    item_model->setHorizontalHeaderLabels(column);
       // 设置水平表头标签
    item_model->setVerticalHeaderLabels(row);
    table_view->setEditTriggers(QTreeView::NoEditTriggers);

```

```
152     QMap<QString ,Good*>::Iterator itt;
153     int i=0;
154     for(itt=shop.begin();itt!=shop.end();itt++)
155     {
156         item_model->setItem(i,0,new QStandardItem(QString("%1").arg
                (itt.value()->getname())));
157         item_model->setItem(i,1,new QStandardItem(QString("%1").arg
                (itt.value()->getcode())));
158         item_model->setItem(i,2,new QStandardItem(QString("%1").arg
                (itt.value()->getnumber())));
159         item_model->setItem(i,3,new QStandardItem(QString("%1").arg
                (itt.value()->getinprice())));
160         item_model->setItem(i,4,new QStandardItem(QString("%1").arg
                (itt.value()->getoutprice())));
161         item_model->setItem(i,5,new QStandardItem(QString("%1").arg
                (itt.value()->getsupplier())));
162         i++;
163     }
164     QMessageBox::information(this,"提示","已为你显示全部库存信息");
165     QString list;
166     QMap<QString ,Good*>::Iterator it;
167     for(it=shop.begin();it!=shop.end();it++)
168     {
169         QString str;
170         str=QString("商品名称：%1  条形码：%2  数量：%3  进价：%4
                售价：%5  供应商：%6").arg(it.value()->getname()).arg(it
                .value()->getcode()).arg(it.value()->getnumber()).arg(it
                .value()->getinprice()).arg(it.value()->getoutprice()).
                arg(it.value()->getsupplier());
171         str=str+"\n";
172         list=list+str;
173     }
174     ui->information->setText(list);
175 }
176 void MainWindow:: paintEvent(QPaintEvent *)
177 {
178     QPainter painter(this);
179     QPixmap pix;
180     pix.load(":/photo/timg.jpg");
181     painter.drawPixmap(0,0,this->width(),this->height(),pix);
182
183     //  画背景上图标
184     //  pix.load(":/photo/ziti.jpg_w800");
185     //  缩放
186     //  pix = pix.scaled(pix.width()*0.5,pix.height()*0.5);
187     //  painter.drawPixmap(10,30,pix);
188 }
```