# ­­­­­­­­­­Project 1技术报告

姓名：欧穗新 学号：16340173 班级：教务三班

## 需求分析：

小喵要求的仓库管理系统的功能如下：

1. 查看仓库库存列表功能
2. 查询功能：可以随时查看当前仓库的库存中的某些商品的一系列信息，包括商品名，剩余量
3. 进货功能：（以下两个功能是根据输入自动选择的）

（1）在仓库进货时，如果仓库中没有此商品，则为仓库增添新的商品项目

（2）在仓库进货时，如果仓库中已有此商品，则增加此商品的库存量

1. 出货功能：

（1）在仓库出货时，减少对应商品的库存量

（2）在仓库出货时，如果这是货物是此商品的最后一批货（库存量为0），则删除仓库中此商品项目

5. 退出程序时存储仓库信息

6. 记录仓库管理日志

## 实际功能： 我的仓库管理系统的功能如下（分为管理员和普通用户）：

## \*\*其中2,3,4,5功能都可以一次性输入几个商品，而不是每次都只能输入一个商品信息\*

一、管理员可以执行的功能：

1. 退出程序
2. 查看仓库库存列表功能
3. 查询功能：可以随时查看当前仓库的库存中的某些商品的一系列信息，包括商品名，剩余量，售价（管理员还可以查看成本价）
4. 进货功能：（以下两个功能是根据输入自动选择的）

（1）在仓库进货时，如果仓库中没有此商品，则为仓库增添新的商品项目

（2）在仓库进货时，如果仓库中已有此商品，则增加此商品的库存量

4. 出货功能：

（1）在仓库出货时，减少对应商品的库存量

（2）在仓库出货时，如果这是货物是此商品的最后一批货（库存量为0），则删除仓库中此商品项目

5. 更改产品价格功能

6. 管理员可以更改获取管理权限的密码

7．管理员可以查看所有用户的操作历史记录

二、普通用户可以执行的功能：

1. 退出程序
2. 查看仓库库存列表
3. 查询功能：可以随时查看当前仓库的库存中的某些商品的一系列信息，包括商品名，剩余量，售价

三、仓库资料库功能（有一个固有文件txt1，存储管理密码和用户数目；一个固有文件存储仓库信息txt2）：

1. 退出程序时存储仓库信息 （每次运行程序时，将txt2中的仓库信息写入程序，同时将txt1里面的密码；程序结束前，把仓库信息保存到txt2里面，同时也保存密码信息进入txt1）

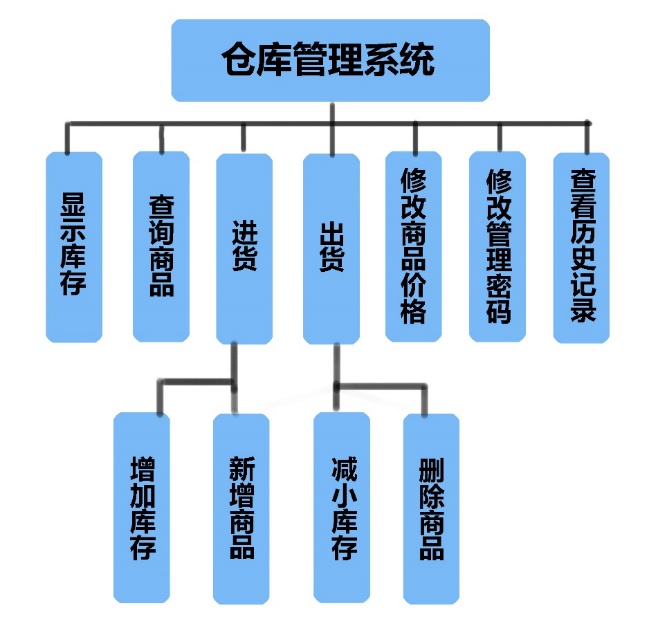
2．记录仓库管理日志（每个注册的用户都会在注册时产生一个记录其操作历史的文件，文件名就是ID号，由txt1里面的用户总数产生，每次注册 “用户总数”++，产生该文件。）

**四、输入异常处理功能**

每次用户输入都会检测用户输入是否正确（一个check（）函数）

## 实现思路:

功能结构图：·



本仓库管理系统主要分为出货、进货、查询、列表库存、修改商品价格、修改管理密码、查看历史记录。七大模块，分别对出货、进货、查询、修改商品价格、修改管理密码、查看历史记录等的操作进行管理。

进货模块中又细分为增加库存和新增商品子功能，当进货时，若此商品在仓库中没有库存，则在仓库库存条目中新增此商品项目，若已有此类商品，则根据进货量增加对应的库存量。

出货模块中又细分为减少库存和删除商品子功能，当出货时，减少对应商品的数目，注意到当库存不足时，出货失败，且若出货成功并且库存为0时，删除仓库目录中此商品项目。

具体的实现方法是：

设计一个链表，通过遍历链表打印商品信息来 实现显示商品库存的函数；

通过遍历链表，比较用户输入的name查找商品来 实现查找商品的函数；

然后在进货时，通过查找函数判断库存中是否有该商品，从而决定执行新增商品/增加库存；

同样在出货时，通过查找函数判断库存中是否有该商品，如果有，再进一步判断商品的数目与用户想要的数目，从而选择delete商品/decrease商品；

修改价格时 通过查找函数得到指向该商品的指针，从而更改价格；

修改管理密码则是直接更改全局变量Password，最后把改完的结果输入txt1；

查看历史记录是根据用户（管理员）输入的ID打开相应的用户文件，把文件内容写到终端；

通过以上方法，最终实现了：七个主要功能（对普通用户而言只有两个功能，即1,2功能）、一个简单加密管理系统、生成文件并存储用户操作记录、存档仓库信息。

## 数据设计：

struct Goods {

**string** name;

**int** count = 0;

**double** prime\_costs = 0;

**double** selling\_price = 0;

**Goods** \*next = NULL;

}

name表示名称，count表示数量，prime\_cost表示进价，selling\_cost表示售价,count表 示货物数量

其中，管理员可以访问所有变量，并赋予所有变量初始值，并且可以在之后更改售价

用户只能读取除了进价以外的数据，不能进行写操作

## 函数设计：

一、关于交互信息的函数:

（1）询问用户是不是管理员并审核，如果是manager，返回真且ret全局变量置为1，表示表示manager，不是manager返回真且ret=0，否则输入错误，返回假，要求重新输入

**bool** is\_manager(**void**);

（2）打印该仓库系统的功能:

**int** print\_function();

（3）根据用户所选数字确定对应的用户想要的功能，如果用户选择退出程序，返回0；否则正常执行，返回1

**int** choose\_function();

-

二、基本函数声明

（1）进货，对应进货模块，表示当前进货一批数量为count的name商品。

**void** add\_goods(**string** name, **int** count, **double** prime\_costs, **double** selling\_price);

它有两个子函数：

（1.1）更新库存信息，对应增加库存子功能，对name商品新增count数量

**void** increase\_count (**string** name, **int** count);

（1.2）更新库存列表，对应新增商品子功能，新增name商品且初始数量为count

**void** add\_to\_list(**string** name, **int** count, **double** prime\_costs, **double** selling\_price);

（2）出货，对应出货模块，表示当前出货一批数量为count的name商品，如果该货物 不存在，返回0，否则，返回1，表示成功取出。

**int** delete\_goods(string name, int count);

它有两个子函数：

（2.1）更新库存信息，对应减少库存子功能，对name商品减少count数量

**bool** decrease\_count(**string** name, **int** count);

（2.2）更新库存列表，对应删除商品子功能，删除商品列表中name商品，

**bool** delete\_from\_list**(string** name);

（3）显示当前库存列表，包括商品名及其库存量，成本价，售价；普通成员不可以看 到成本价

**void** show\_goods();

（4）查看仓库中的name商品的一系列参数，普通成员不可以看到成本价，函数返回的是指向查找得到的商品的指针；如果仓库没有这个商品，返回一个默认构造的goods变量（这个变量的count是0，以此判断该商品不存在）

**Goods**\* find\_goods(**string** name);

三、新增功能函数：

新增的功能,总共有管理者函数和用户函数两部分:

（1）管理者函数:

（1.1）首先应该包括所有的basicfunction里面的函数:

即 add\_goods, increase\_count, add\_to\_list;

和delete\_goods, decrease\_count, delete\_from\_list

以及show\_goods, find\_goods

然后是新增的函数:

（1.2）修改仓库访问密码:

**void** change\_password(**string** password);

（1.3）修改售价:根据商品名称进行修改:（如果存在这个商品，返回1表示修改成功，否则返回0表示修改）

**int** change\_price(**string** name, **double** price);

（1.4）管理员查看所有用户操作（包括普通用户和管理员）如果存在该用户id，返回1表示访问成功，否则返回0表示访问失败

**int** open\_file(**string** id);

（2）用户函数:

包括:**void** show\_goods(), **Goods**\* find\_goods(**string** name);

（注意这个功能不同于管理者的查询函数,不可以看到进货价格）

四、存档所有信息函数，包括密码信息，仓库信息，用户操作信息等等

（1）每次运行程序将之前保存的仓库信息，密码信息，用户数目写入程序

**void** write\_back(**void**);

（2）这个函数用来保存关闭程序时仓库的存储情况，以免退出程序后数据丢失

而对应的文件“txt2”，就是专门用来保存仓库信息的.

**void** Place\_on\_file(**void**);

（3）这个函数可以向存储用户信息的文件写入时间信息,从而记录用户进行某一个操作的时间，返回操作具体时间

**string** get\_time(**void**);

//本来应该有一个专门根据用户的选择，向文件输出，记录用户操作的函数

//但是用户每个不同的操作，有不同的输入，而每个操作用户只输出一次，所以不适合把‘写向屏幕’和‘写向文件’的操作分成两个函数

//如果非要这么做，要么（1）要么需要传递很多变量，多分很多情况，要么（2）需要定义很多全局变量，

//所以干脆统一写在choose\_function()里面了

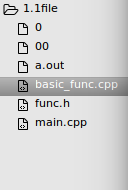
//总之，，水平有限，，，，，，时间精力有限，，，，，，

**五、异常处理函数（2.0新增）**

一个**void** check（**int** num），用于检测用户输入是否正确。

## 输入与输出（以下附上图片和说明，从中可以看到输入输出）：

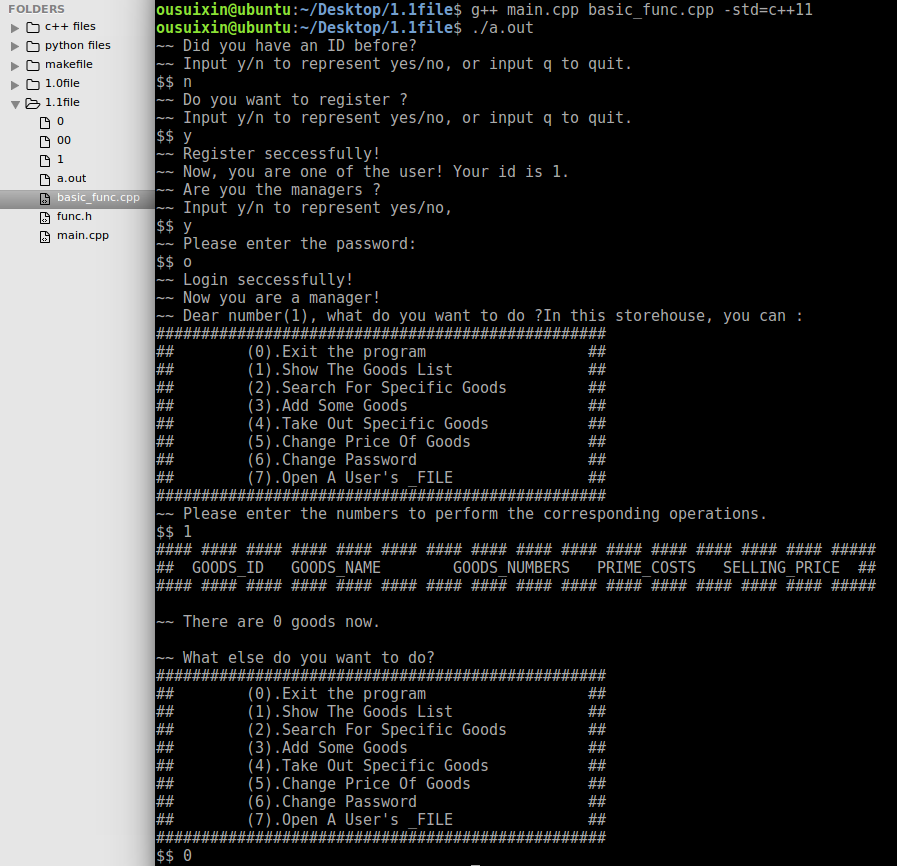
这是一开始的文件状况，0文件即Word说明文档中的txt1,00文件对应txt2：



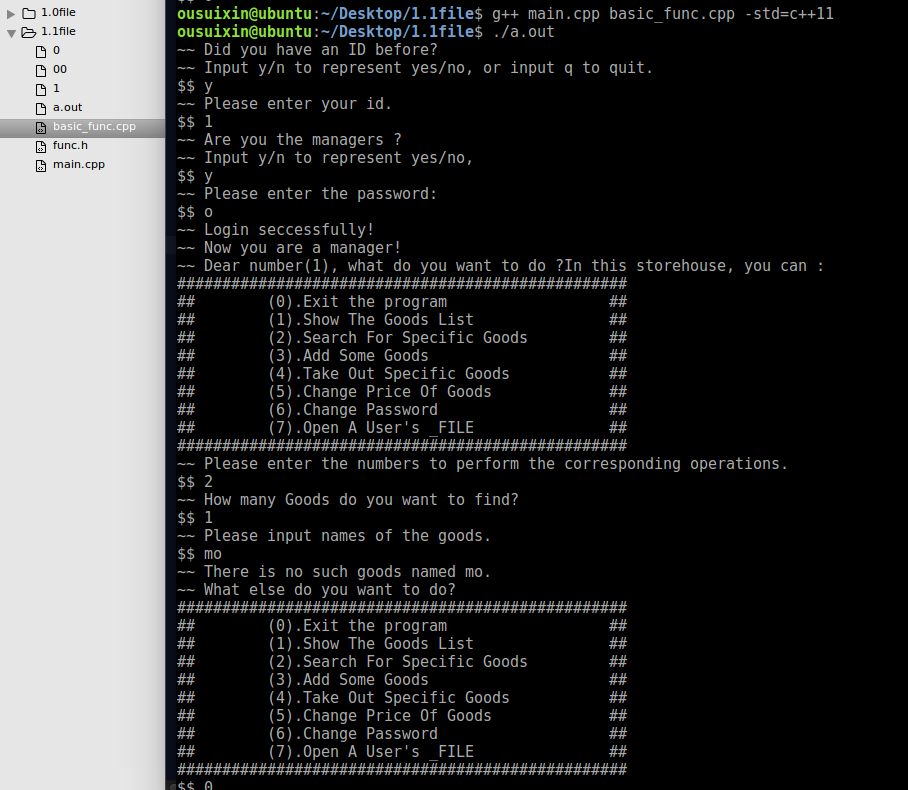
这是第一次注册后，文件状况（左侧框）和输入输出情况（右侧终端界面）

可以看到（1）文件筐多了“1”文件，这是保存用户ID为“1”的用户的操作记录文件

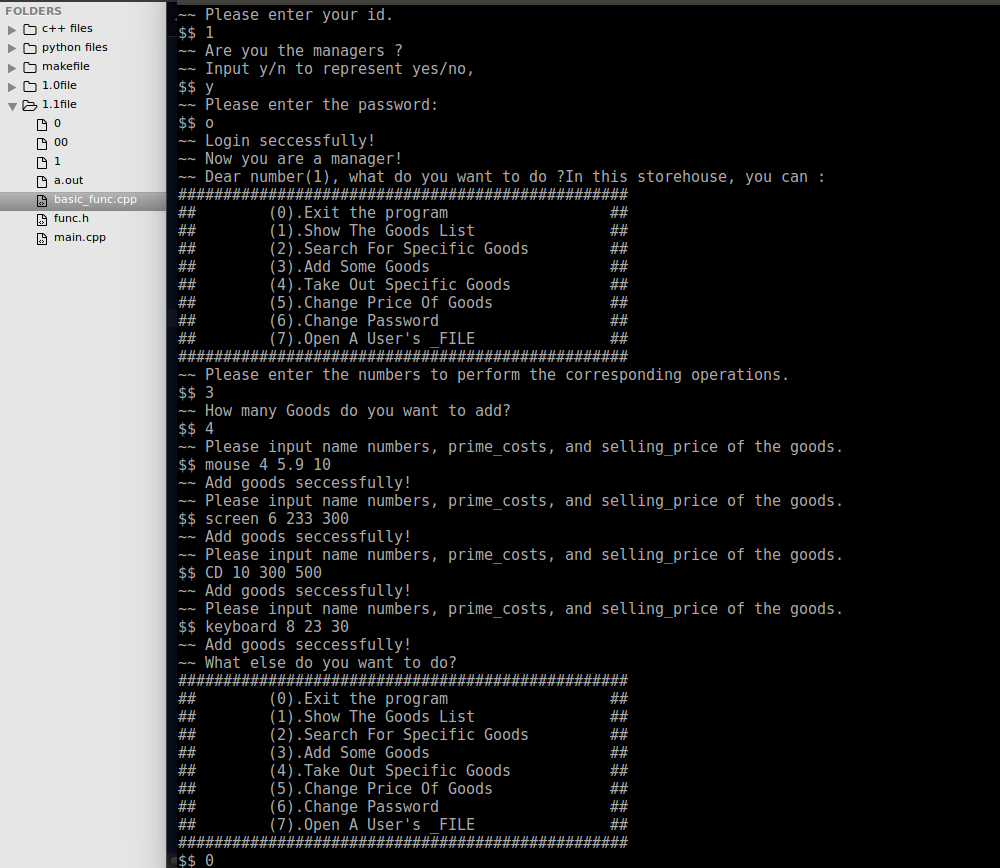
（2）右侧终端中，每一行以“**~~**”开头的表示程序的**输出**语句（用于提示用户操作），以“**$$**”开头的表示用户的**输入**。以此便于区分（图中还进行了第一个功能操作 ）



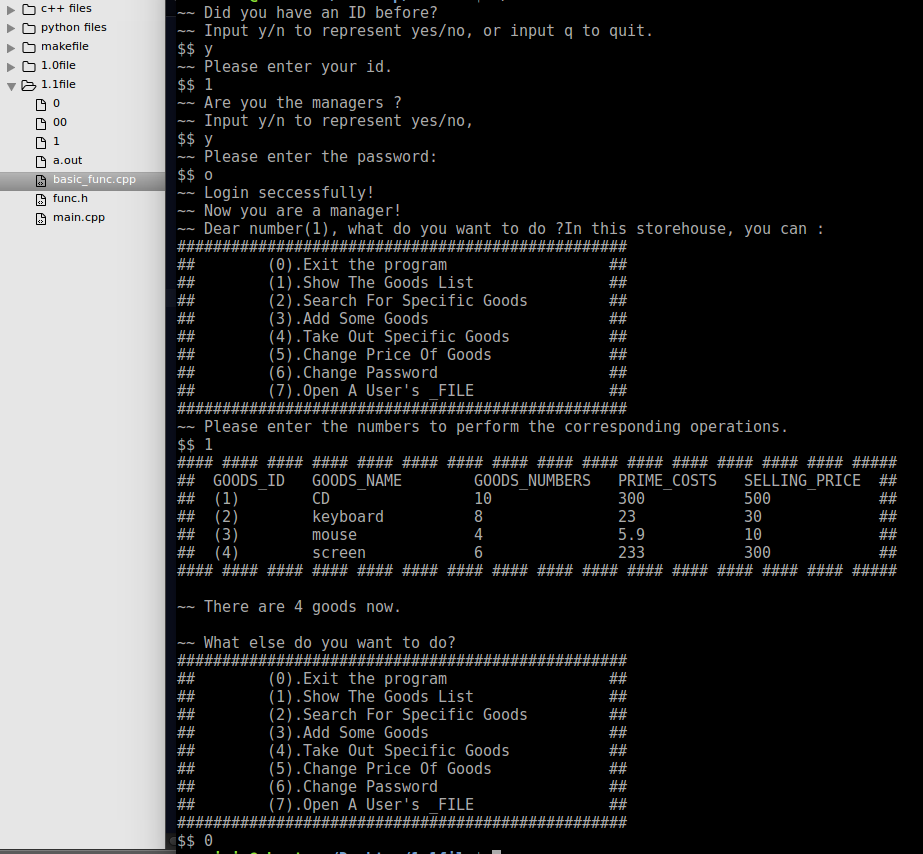
这里执行了第二个操作 （由于仓库是空的，拒绝搜索）：



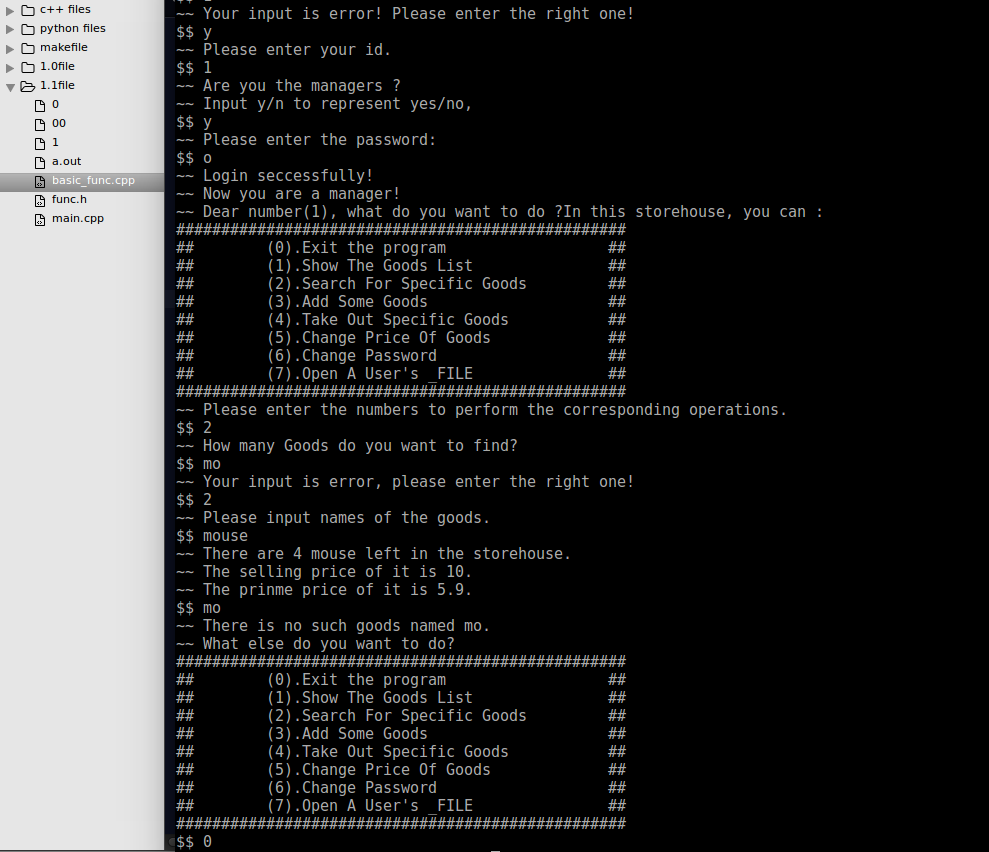
执行第三功能，连续添加四种商品：



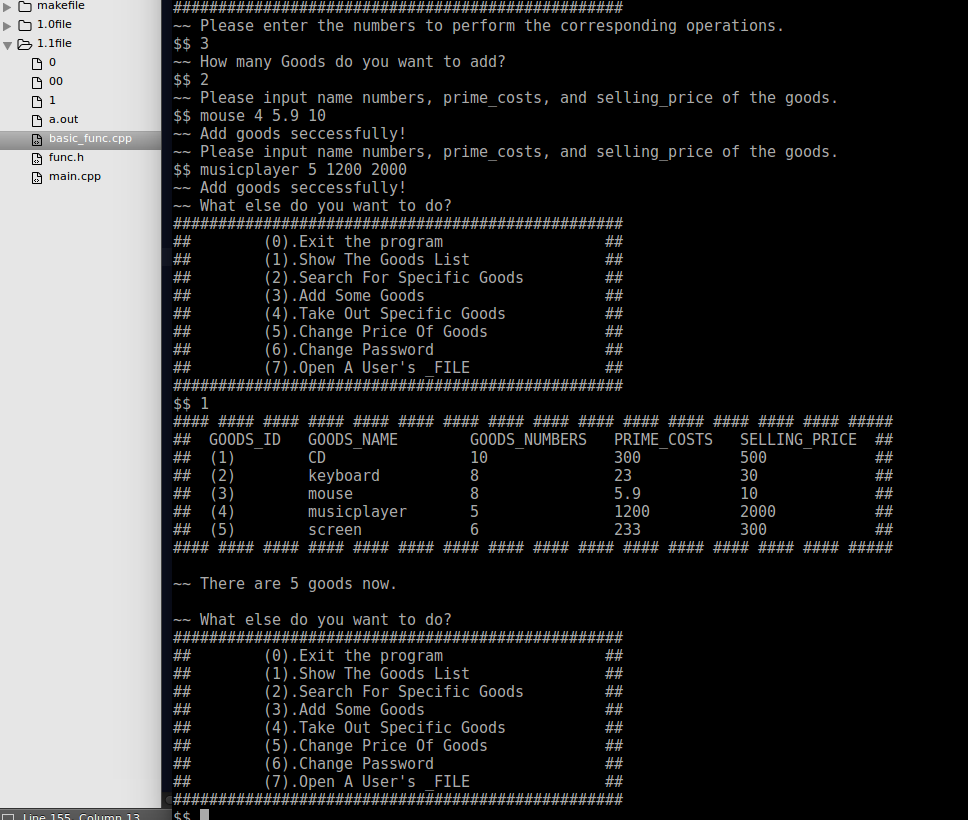
这时候再使用第一个函数，就有商品列表了：



这时候执行第二个功能，就可以进行（1）查找，并且（2）如果没有该商品，会指出没有商品：

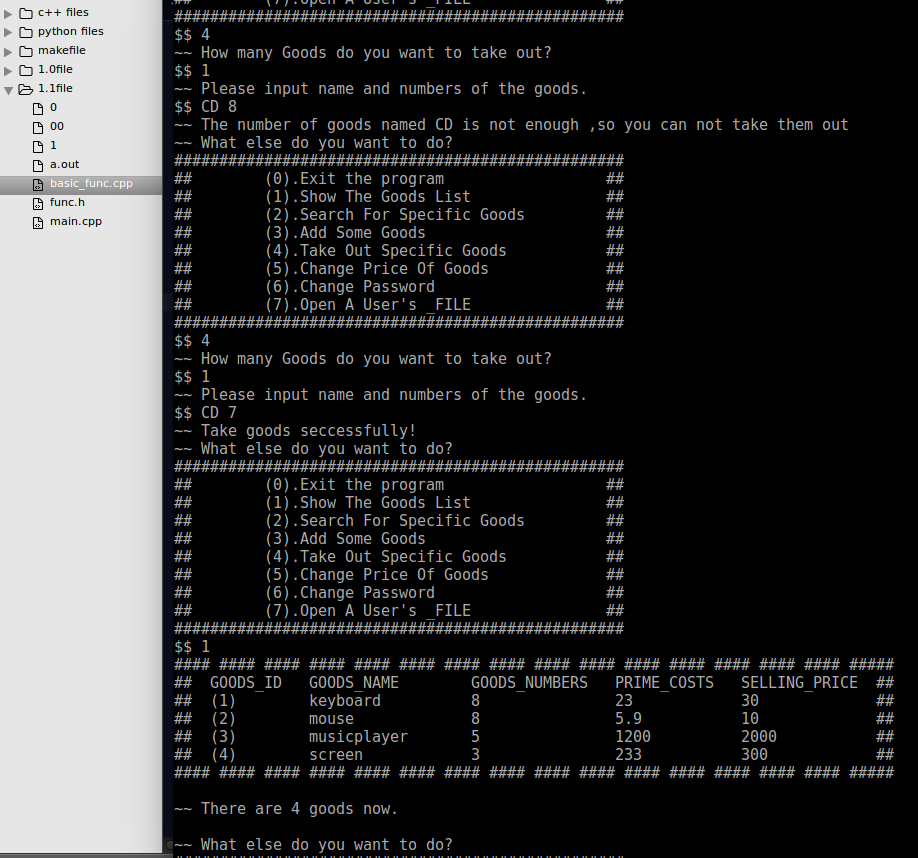


此时再次进行add，可以看到仓库中（1）mouse增加了4件（2）新增了musicplayer 5件：

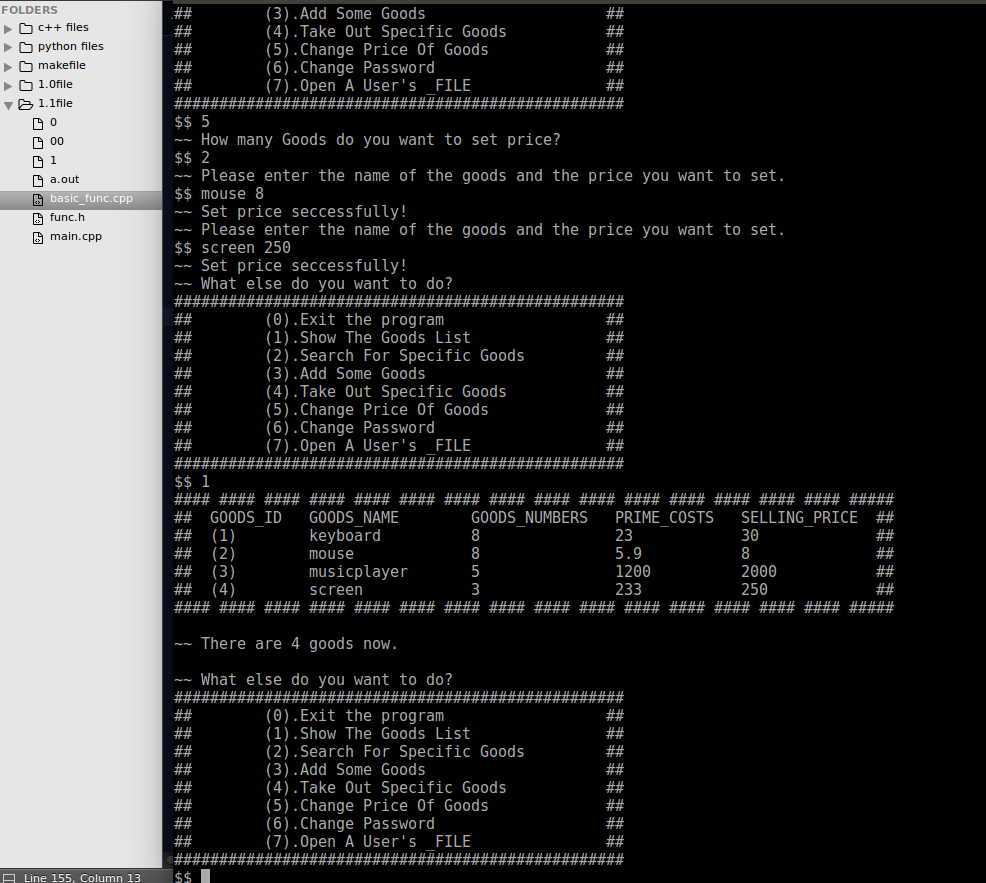


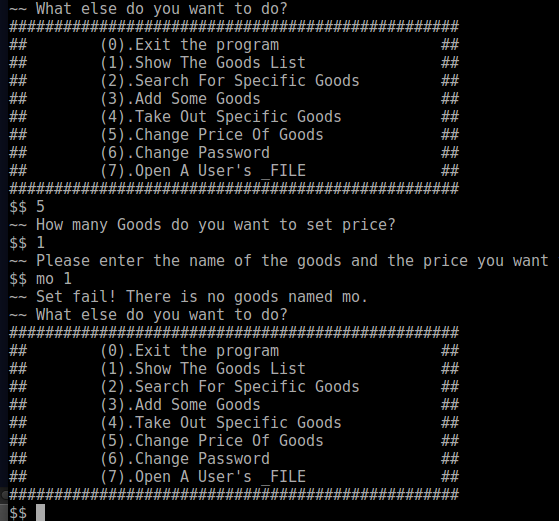
执行功能四，可以看到（1）取出CD件数为8，超过库存时，拒绝了操作，（2）取出件数与库存相等时，删除了该商品，所以之后库存没了CD：

（还有一张图，关于功能四我搞丢了：（3）可以看到 没有商品时拒绝取出（4）商品数目大于要求数目，该商品数目decrease了）

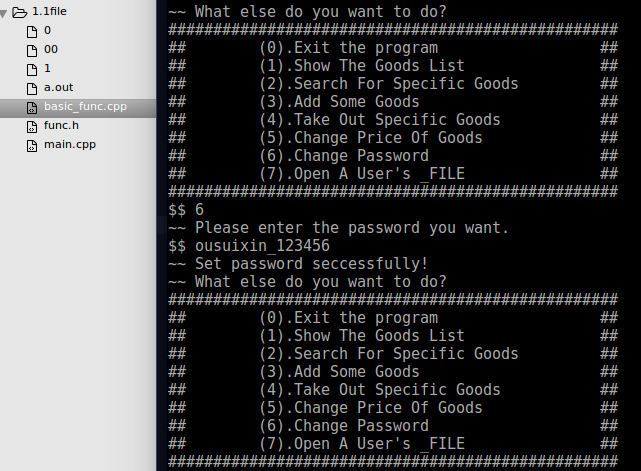


我们执行了5，连续更改两个商品价格，看到了商品mouse，screen价格都改变成输入的数字，再次执行5，看到：如果没有这个商品，则拒绝更改商品价格 ：

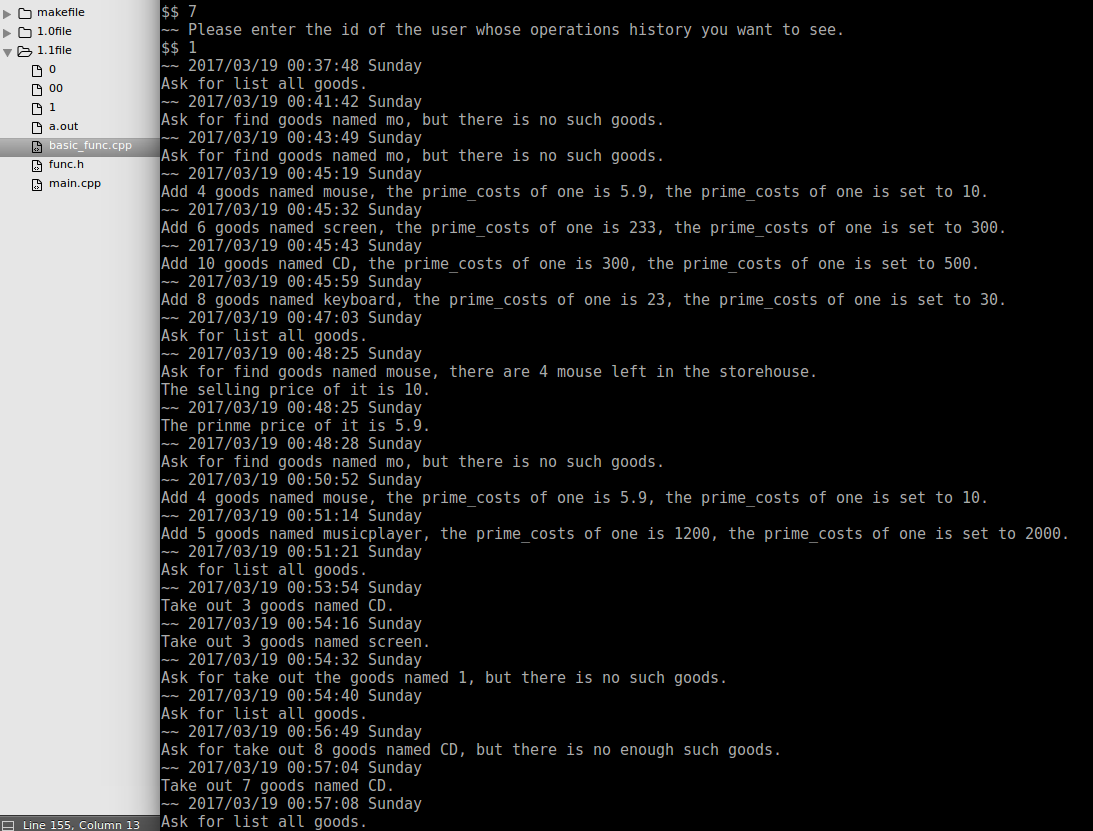




我们执行了功能6，之后关闭程序，会看到文件0里面的密码数据由初始的“o”变成了“ousuixin\_123456 ：



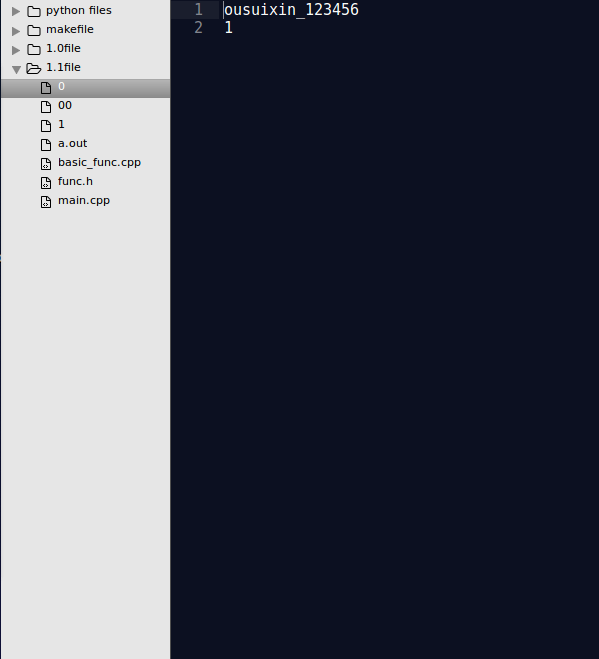
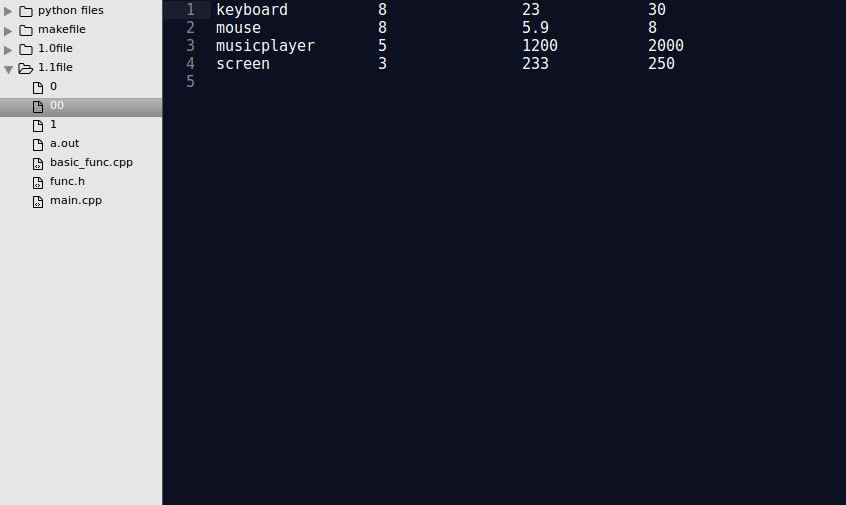
执行了功能7，通过输入用户自己的ID，看到了自己的操作记录 ：



退出程序之后看到，（1）原来的0文件保存的初始密码“0”；

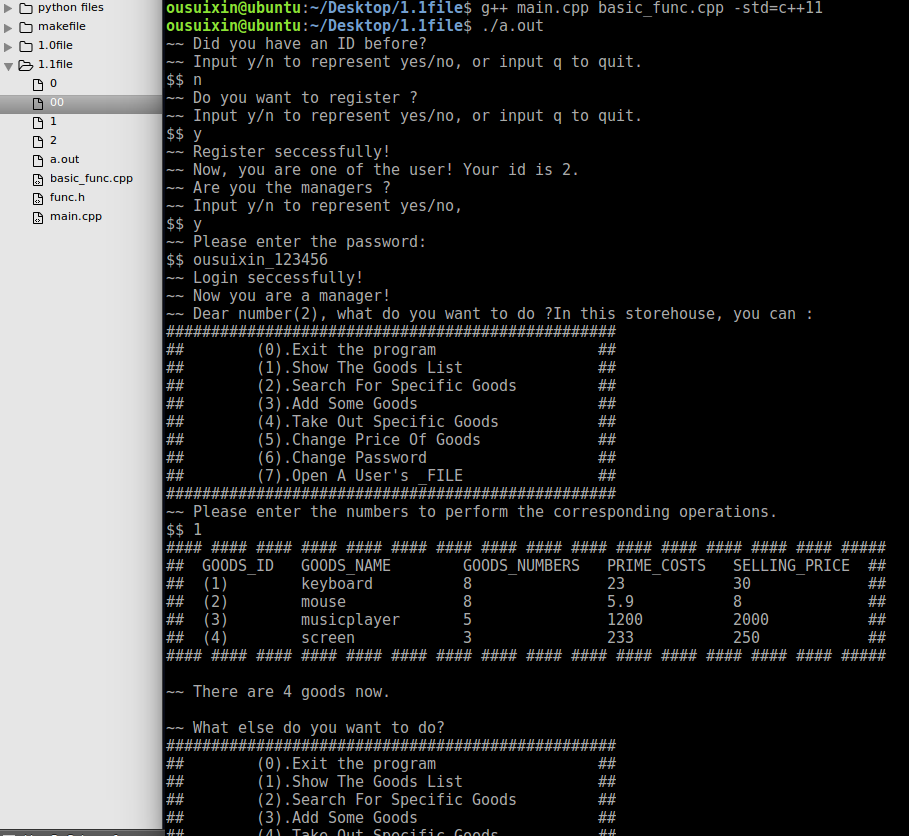
（2）变成了“ousuixin\_123456”，用户数由初始的0，变成了1；

（3）仓库库存由空变成了四件：

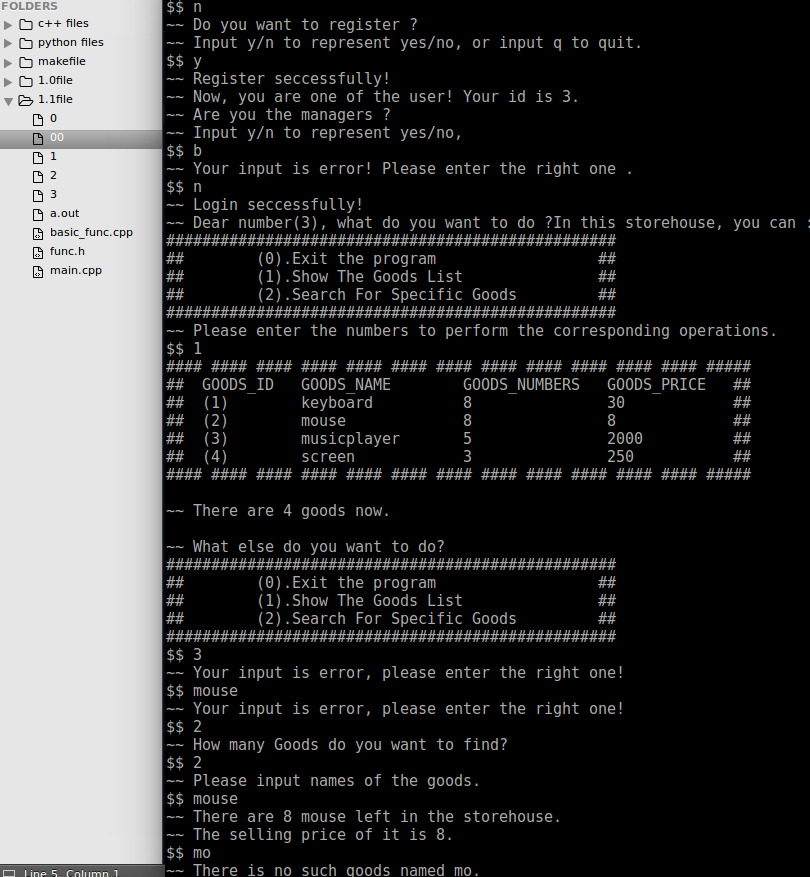
 

我们又注册了一个账户，发现文件夹里面又多了“2”文件

即保存新用户历史记录的文件：



我们又注册了新用户，（1）文件夹又多了一个名为“3”的文件，保存新用户操作记录，但是（2）这个用户不是管理员，所以只有两个功能，两个功能均与管理员功能相似，但是看不到成本价。（3）同时可以看到，要求输入数字却非法输入字符串，其他数字之类都会报错 ，并要求重新输入



**总结：**

大概功能，实现方法，输入输出…上面已经列出

（2.0版本修复若干bug：没有内存泄漏，用valgrind试过了；异常处理也做了。）

最后讲一下缺点吧

1. 本来还想实现每个用户都有一个密码（包括管理员，普通用户，但是管理员还有管理权限密码，即管理员有两个密码），另外有一个管理员才知道的管理权限密码，并且一旦管理员注册过后，系统会知道这个ID的用户是管理员，不要再输入管理权限密码就可以进行管理员操作

但是精力，时有限，无法完成了

1. （还有，自己的代码重复度很高，查找算法也不好，还可以优化，但是时间，，，精力，，，，不足，就没有继续优化某些代码了）

**最后辛苦老师，ta们看我的破代码orz，（又长又low，，，没时间改进了orz）**

**O(∩\_∩)O谢谢你们！**