项目总结

一些解释：

本人使用vs2022进行编程，由于其特性，使得scanf函数使用、变量未初始化等正常操作变为警告，但本人考虑到这些警告不影响程序运行，并在自己当前认知中认为没有出现重大语法错误，于是没有将其修改，因此这些程序或多或少包含一些警告，希望可以得到谅解，万分感谢【双手合十】

1-1

关键点：

1.注意不同变量的类型（如int和char）

2.使用循环

1-2

关键点：

1.使用循环

1-3

思路框架：

一个完整过程分为摆货、买货、付钱、找零这四个模块

由于摆货只需要一次，因此可以使用scanf函数分别扫描商品种类（%c）、通道号（%d）、单价（%d）、个数（%d）这四个数据的输入。之后自动进入买货环节。买货环节可以继续使用scanf函数分别扫描商品种类（%c）、通道号（%c）、个数（%c）这三个数据，然后根据买货数量以及摆货时输入的单价计算总价。之后进入付钱环节，用scanf函数扫描输入的金额（%d），累加输入总金额。当输入金额大于总价时进入找零环节，直接计算零钱数量并利用printf函数进行显示，完成整个过程。

细节：

1.对一些异常数据报错：

①摆货时：摆货数量大于50、通道号不是1-5、单价不是1-9

对于这种问题，有两种解决方法。第一种：可以直接使用if else语句判断，如果有输入错误，直接goto到输入环节，重新输入商品种类、通道号、单价、个数，用新数据覆盖原有错误数据。第二种，将摆货数量、通道号、单价赋值为0，这样就无法进行购买。之后在整个摆货环节构建for循环，只要通道号（或者单价）为0就继续循环，这样就不会携带错误数据进入买货环节

②买货时：商品种类与摆货时不对应、通道号与摆货时不对应、买货数量大于摆货数量

解决方法类似

③付款时：付款金额不在1、2、5之中

直接将这次付款金额赋值为0

关键点：

1.注意不同变量的类型（如int和char）

2.使用循环

3.面对买货、付款输入错误数据的情况，若已将这次数据累加进卖货数量、买货总价、付款金额这三个变量中，则需要进行扣除（不能让错误数据进入）

4.清空缓冲区（防止scanf（“%c”）吸收到回车字符）

2-1

思路框架：

这道题与1-3相比有本质区别，在于需要识别当前输入是否为END，若为END就要进入下一环节，若不为END且数据正确就需要将这次数据录入当前环节。可见一次输入承担两个功能，那对应的变量也要具备储存两种数据的能力。由于END是字符串，A空格3空格7空格5回车也可以视为一种字符串，因此我选择gets函数将数据赋值给字符串str，并利用strcmp函数判断字符串str与END是否相等，若相等就跳出当前循环，进入下一个循环，否则就将str的数据分别录入到商品种类、通道号、单价、摆放个数这四个变量中。商品种类对应str【0】，通道号对应str【2】，单价对应【4】，摆放个数是个位数就对应str【6】，否则就对应str【6】与str【7】。值得注意的是，字符形式的数字0-9与整型数字0-9大小不同，ASCII码差了48。因此需要判断，如果摆放个数是个位数，则str【7】存放’\0’，ASCII码为0，如果摆放个数是两位数，则str【7】存放字符型数字，ASCII码为48-57。可以进行判断，如果str【7】==0则输入个数为个位数，则真实摆放个数为str【6】-48。如果str【7】！=0，则输入个数为两位数（还可能是三位数及以上，因此需要设置报错进行排除），此时真实摆放个数（如果是两位数就是真实摆放个数）为（str【6】-48）\*10+str【7】-48。同样的通道号与单价也可以分别为str【2】-48与str【4】-48，这样就实现了摆放环节数据的全部录入。

进入购买环节、付钱环节就和1-3没有本质区别了，唯一的差异就是用gets函数替换scanf函数，需要通过ASCII码进行字符数字与整型数字替换。以及购买环节要判断字符串是否为END（付款时如果钱没付完就不能主动结束）。考虑真实情况，在找零后可以循环买货环节，也可以输入END结束整个流程，这时候就需要再次使用gets函数吸收输入的字符串，如果字符串不为END，那就用strcpy将这组字符串复制给新一轮买货环节的第一组数据。若字符串为END，那就结束这一个完整流程，并开始新一轮摆货环节（标志着一个完整流程的结束，同时开始了新一轮流程）

细节：

①摆货时：摆货数量大于50、通道号不是1-5、单价不是1-9

通道号和单价的解决方案跟1-3类似

摆货数量要考虑到三位数及以上的输入，例如输入100，实际上超过50，应该报错，但是由（str【6】-48）\*10+str【7】-48的计算方法就得到10，小于50，不会报错。因此需要判断str【8】的ASCII是否在48-57之间，若在，就说明卖货数量至少为三位数，肯定超过50了。

②摆货时：同种商品价格不统一

例如先输入A空格3空格7空格5回车，再输入A空格2空格6空格5回车，对于这种问题，就需要数据遍历，判断如果商品种类相同时单价是否相同。

③摆货时：通道号被重复使用

例如先输入A空格3空格7空格5回车，再输入A空格3空格7空格20回车，这时候再进行数据遍历，存放通道号的数组中有相同元素，则证明通道号重复使用，进行报错。

④买货时:商品种类与通道号不对应

这种可能是输入了不存在的商品种类、输入了不存在的通道号、输入了存在的商品种类和通道号但是不匹配（例如摆货环节输入A空格3空格7空格5回车，再输入B空格2空格7空格20回车），最后买货环节输入A空格2空格7空格5回车。

对于这种问题，应当进行商品种类与通道号数据遍历，判断卖货数据是否与摆货数据匹配。

⑤付钱时：金额大小错误

解决方法和1-3类似

关键点：

1.找到用来存放数据的变量（字符串）

2.字符串数字与整型数字的差异（ASCII码）

3.善用循环

4.清空缓冲区

5.每次找零后要清空剩余零钱，给零钱变量赋值0

补充：

考虑到真实情况，在一轮流程彻底结束后会自动开始第二轮流程（开始第二次摆货）

如果不结束第一轮，继续买货时，当所有通道货物剩余量为0时将自动结束这一轮完整流程，并提示售货机没有货物了，自动开始第二轮流程（开始第二次摆货）。

改善与提升：我认为我的这个程序还有提升空间，比如我的程序不允许摆货时重复使用同种通道，但是现实情况或许不是。如果先输入A空格3空格7空格5回车，再输入B空格3空格7空格20回车，那同一个通道放两种货物确实不合适，应当报错。但如果先输入A空格3空格7空格5回车，再输入A空格3空格7空格20回车，那是不是应该等效为A空格3空格7空格25回车，而不是报错呢？

我预想的改进方向是在识别到输入两次相同通道号时进行判断，判断是否时相同货物，如果是相同货物且加和小于等于50，就将这两次摆放数量相加，并计入数据。而不是报错。

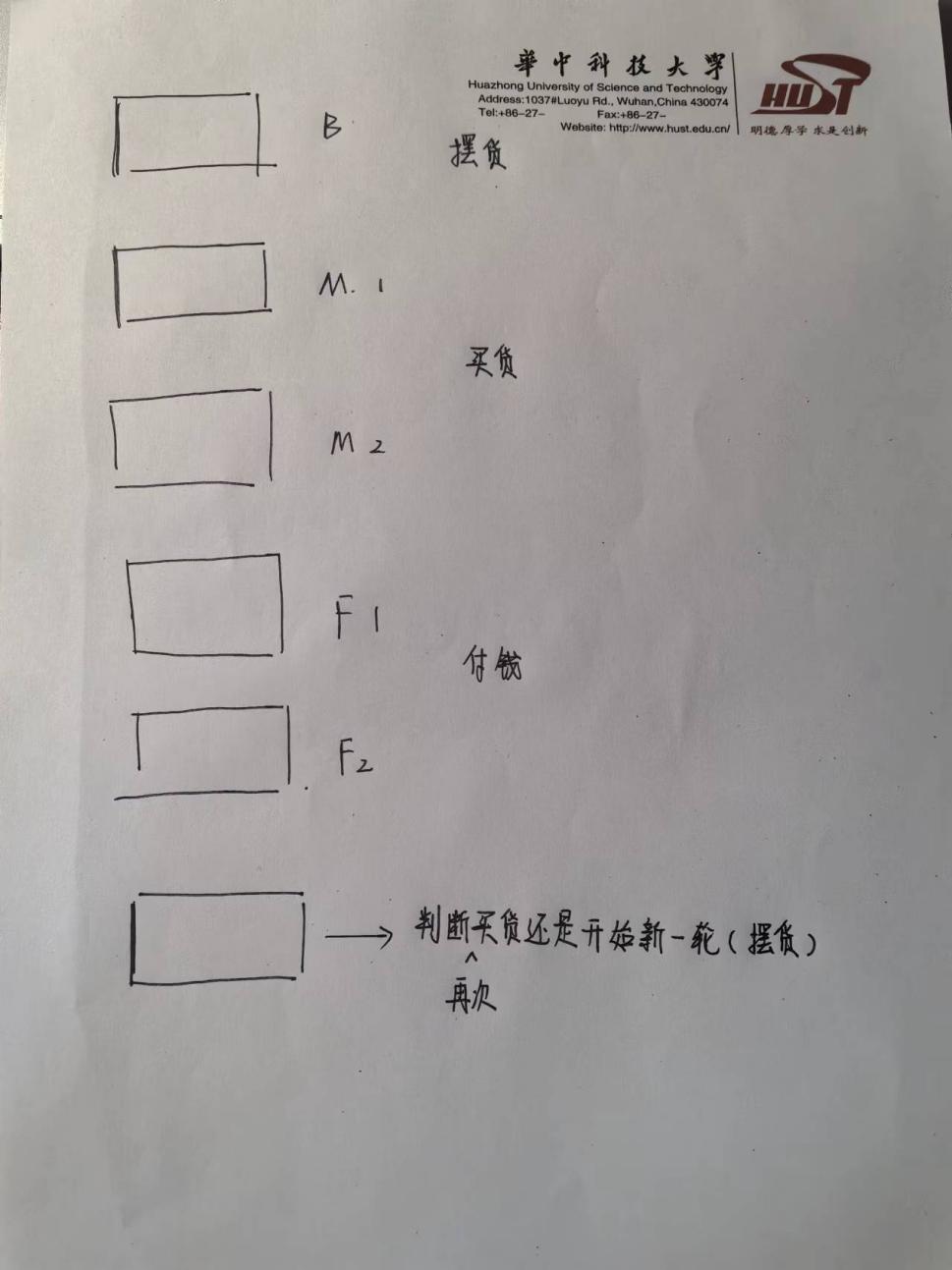
2-2

我认为2-2是2-1的再次提升

BACK不仅是能在单一模块消除数据（比如分别在摆货、买货、付钱环节消除摆货、买货、付钱的输入数据），还可以实现不同模块的跳转。例如摆完货物输入END，之后输入一组买货数据，此时若连续输入两次BACK，意味着消除了一组卖货数据，并消除END，再次进入摆货环节。

考虑到这种功能，我想到使用goto来实现不同模块的跳转。

（下一页有图片）



由于要实现上述功能，我设置了如图的结构。其中B、M、F分别代表摆货、买货、付钱环节。M1和F1是连接不同模块的桥梁。在B到F2中每一小块都会判断字符串是END还是BACK还是数据。如果在M1和F1这两个桥梁上识别到BACK，那就会跳转到上一个模块（M1跳转到B）（F1跳转到M2），如果在M2与F2中识别到BACK，只会清除该模块的数据，不会跳转到上一个模块。例如输入END进入买货环节后，输入两组数据，此时程序进入M2，如果只输入1次BACK，只会删除第二组买货数据（删除最近输入的一组），如果输入2次BACK，会先删除第二组买货数据，再删除第一组买货数据，并通过goto进入M1，如果此时再输入一次BACK，就跳转到B，即摆货模块。

细节：关于错误数据的报错和2-1大体类似，不再赘述了

关键点：

包含前面提到的所有点，但我认为，建构处图片所示的结构最为关键。至于判断何时处于M1，何时处于M2，就需要设一个变量（例如i），通过i的大小判断。还有一点就是不能连续输入三次以上的BACK，但是输入BACK后下次不输入BACK，次数就会刷新清零。针对这个功能，我采取每一次输入都判断是否为BACK，若不是就是BACK已使用次数赋值为0。

改善与提升：

程序太复杂了，一方面是滥用goto导致的，另一方面我的大脑运转比较慢，第一次写这种程序，有时候不能清晰理出逻辑关系。我的代码是经过自己反复数据输入进行调试修改得来的，这意味着我的代码还不完美，也许面对特殊数据（错误数据）时或出现BUG。但是可以在不输入错误数据的前提下实现功能。

希望以后能慢慢改进优化这些程序。