

//预先学习内容 1, .c 文件如何转化成可执行文件

预处理: 包含头文件, 删除注释, 宏的替换和删除

编译: 语法分析, 词法分析 (检查关键字), 语义分析, 符号汇总

汇编: 转换成计算机可识别的二进制代码 (?)

链接: 合并段表, 符号表的合并和重定位

//预先学习内容 2, 字符串

## 1-1:

1, 摆放货物功能:

思考:

a, 首先, 需要定义一个合适的数据结构来存储货物信息。我打算是用一个结构体, 包含货物种类、通道标号、单价和个数等字段;

b, 然后需要一个数组存储多个货物信息。使用数组, 需要确定一个合适的大小并且初始化存储结构, 从控制台;

c, 并且需要从控制台 scanf 读取用户输入的货物信息, 并将其存储在定义的数据结构中, 并分类存放到各自变量中;

d, 需要无限循环 while(1) 来实现重复多次的货物存放操作。在循环中, 可以不断提示用户输入新的货物信息, 直到用户输入退出 (END) 命令, 用 break 来退出存放模式;

e, 最后要求的输出格式需要用字符串 for 循环反复叠加货物;

2, 显示已存放货物的信息给用户 (+主菜单):

即输出货物信息, 使用 for 循环遍历存储结构, 将每个货物的信息格式化输出。Printf 所有储存信息, 考虑用 switch case 语句在 printf 之后选择模式 (store/buy)

## 学习+尝试→问题一:

我以为字符串可以直接相加减或者乘法运算

就例如: "a"+"a"="aa", "a"\*2="aa"

经过学习, 改正为引入数组

## 学习+尝试→问题二:

输入字符串, 如何用 if 语句判断?

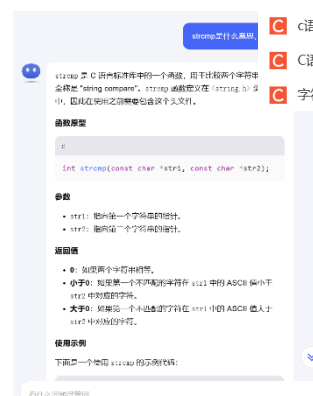
请教了朋友, 得知需要 strcmp 语句

【 C 语言 \_058\_ 字符串比较函数 strcmp 】

[https://www.bilibili.com/video/BV1Ki4y1w7gJ?vd\\_source=17f72e33628e5456079163e322a59cd6](https://www.bilibili.com/video/BV1Ki4y1w7gJ?vd_source=17f72e33628e5456079163e322a59cd6)

看视频发现还挺好理解的, 然后舍弃了使用 switch case 的方案, 直接用 if else 进行售货机状态的调节。

- C 语言实现字符串连接\_C语言字符串连接-CS... 11:55
- C 语言字符串连接- CSDN搜索 11:55
- C 语言: 字符串-带括号的加减乘除运算\_C语... 11:54
- C 语言字符串怎么加减- CSDN文库 11:54
- C 语言字符串加减- CSDN搜索 11:54
- 字符串加减- CSDN搜索 11:54



### 学习+尝试→问题三:

发现 for 语句循环输出 AAAA 失败, 各种报错或者变量消失术, 抓耳挠腮一个小时, 无果, 毅然改写成 Ax4 的写法, 准备之后重启再试一次。

然后越搞越乱, vs 里全是报错, exe 根本跑不起来———————  
——》我换成了 Dev-C++

### 学习+尝试→问题四:

现在的程序只能存放一种东西, 我打算用二维数组, 但是二维数组里怎么存放不同类型的变量? 又怎么把一个货架的商品信息检索显示出来? 好像不行。

然后我就去找我哥请教学习了结构体, 真是个好东西!  
顺手学了个 strcpy 函数用于存储商品名字。

在学习结构体并且引用后, 我发现我“buy”时出现的商品展示中有许多并未输入的货架摆了“乱码商品”——》发现这是没有初始化的后果

### 学习+尝试→问题五:

这个其实是之前的问题, 没有把字符串的输入学到位, 先是写了 scanf ("%c",choice ), 改成 scanf ("%c",&choice ), 再改成 scanf ("%s",&choice ), 最后改成 scanf ("%s",choice)。基础不牢, 地动山摇!

```
1 #define _CRT_SECURE_NO_WARNINGS
2 #include<stdio.h>
3
4 int main()
5 {
6     char object;
7
8     int num, location, price = 0;
9     printf("位置, 物品, 数量, 单价\r\n");
10    scanf("%d\r\n", &location);
11    scanf("%c\r\n", &object);
12    scanf("%d\r\n", &num);
13    scanf("%d", &price);
14
15    char result[128] = { 0 };
16    for (int i = 0; i < num; i++)
17    {
18        result[i] = object;
19    }
20
21    printf("%d:%c %d", location, result, price);
22
23    return 0;
24 }
```

```
printf("位置, 物品, 数量, 单价 (数据间需要空格分开) \n");
scanf("%d %c %d %d",&location,&name,&num,&price);

char result[128];
result[0] = '\0';
for (int i = 0; i < num; i++)
{
    result[i] = name;
}
result[num] = '\0';
// 重复名字显示个数
printf("%d:%s %d", location, result, price);

return 0;
```

- C 如何理解左操作数必须为左值\_左操作数必... 1
- C id returned 1 exit status- CSDN搜索 14:04
- C 字符串 c左操作数必须为左值- CSDN搜索 1
- C 问题: warning: assignment makes integ... 13:44
- C gcc\_warning:assignment makes integer ... 13:44
- C assignment makes integer from pointer... 13:44
- C 左操作数必须为左值怎么解决\_编程语言-CS... 1
- Edge Image Viewer 13:44
- C 错误 c1075 "(": 未找到匹配令牌- CSDN文库 12:57
- C fatal error C1075: "(": 未找到匹配令牌\_严... 12:56
- C c1075未找到匹配令牌- CSDN搜索 12:56

```
D:\>gcc 1.c -o 1.exe
1.c: 在函数 main 中:
18:     {
19:         result[i] = object;
20:     }
21:     result[num] = '\0';
22:     printf("%d:%c %d", location, result, price);
23:
24:
25:     return 0;
26: }
27:
28: struct good
29: {
30:     char name;
31:     int price;
32:     int cut;
33: };
34:
35: good channel[3];
36:
37: int a[3];
38:
39: // channel[i] --> {a,i}
40:
41: good tmpchannel[1];
42: tmp_name="a";
43: char_name="a";
44:
45: int _cdecl strcpy_s(char _dstch[1024], rchar _srcch[1024],
46: _rsize_t _rsize)
47: {
48:     _strcpy_s(dstch, srcch, rsize);
49:     return 0;
50: }
```

**使用示例**

在你的代码中, strcpy(channel[location].name, name); 该行代码的作用是将 name 字符串复制到 channel 数组中指定位置 (由 location 索引指定) 的 name 成员中。

**代码解释**

- channel 是一个结构体数组, 每个元素都是 good 类型。
- channel[location].name 表示结构体数组中索引为 location 的元素的 name 成员。
- name 是一个字符串, 存储了用户输入的商品名称。

**详细步骤**

- 包含头文件: 包含 <string.h> 以使用 strcpy 函数。
- 定义结构体: 定义一个包含字符串的 good 结构体。
- 读取输入: 使用 scanf 读取用户输入的商品名称到 name 数组。
- 复制字符串: 使用 strcpy 将 name 数组的内容复制到 channel[location].name。

```
16
17
18
19
20
21
22
23
24
25
26
27
28
```

```
char name, choice[8]; //物品名称, buy/store选择

int num, location, price = 0;

while (1)
{
    printf("buy or store");
    scanf("%s",&choice);
    // switch(choice)
    printf("%s", choice);
}
```

buy or store  
buy

## 1-2:

这段代码是一个简单的模拟商店购物和存储商品的程序。其中，投币和找零功能是购物流程的一部分，设计思路如下：

### 1. 投币逻辑：

- 用户在购买商品后，程序会提示用户投币。
- 用户输入投币金额，程序将这个金额从总金额中扣除。
- 这个过程在一个`while`循环中进行，直到总金额减到 0 或者变成负数。

### 2. 找零逻辑：

- 当用户投币总额等于或超过商品总价时，程序会停止投币循环。
- 如果投币总额超过了商品总价，程序会计算出找零金额。
- 程序会显示找零金额，然后结束购物流程。

在代码中，这部分逻辑体现在以下几行：

```
while(total>0)
{
    printf("请投币:");
    scanf("%d",&input);
    total=total-input;
}
```

这段代码是一个循环，它会一直执行直到`total`（总金额）小于或等于 0。每次循环，程序都会提示用户投币，然后从总金额中减去用户输入的金额。

接下来的代码块处理找零：

```
if(total==0)
{
    system("cls");
    printf("谢谢惠顾，欢迎下次光临！ ");
    sleep(1);
}
else
{
    system("cls");
    printf("找零%d 元\n",change);
    sleep(1);
    printf("谢谢惠顾，欢迎下次光临！ ");
}
...
```

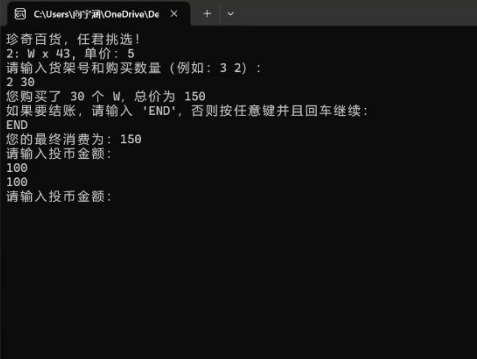
如果`total`等于 0，说明用户投币金额正好等于商品总价，程序会感谢用户并结束购物流

程。如果`total`小于 0，说明用户投币金额超过了商品总价，程序会计算找零金额（`change`），显示给用户，然后结束购物流程。

### 学习+尝试→问题一：

投币功能一直出错，如图：

```
printf("如果要结账，请输入 'END'，否则按任意键并且回车继续：\n");
scanf("%s", choice);
if (strcmp(choice, "END") == 0)
{
    printf("您的最终消费为： %d\n", total);
    int change=0,input=0;//找零和单次付费额度
    while(total>0)
    {
        printf("请输入投币金额： \n");
        scanf("%d\n",&input);
        total=total-input;
    }
    if(total==0)
    {
        printf("谢谢惠顾，欢迎下次光临！");
    }
    else
    {
        printf("找零%d元",change);
    }
    break;
}
```



苦思冥想不得其解，于是向 kimi 求助，发现是 scanf 处不应该有\n

投币输入处理：scanf("%d\n", &input); 中的 \n 可能会导致输入问题，通常不需要在格式字符串中包含换行符。什么输入问题



在 C 语言中使用 scanf 函数时，格式字符串中的 \n 并不会按预期工作来匹配输入中的换行符。实际上，scanf 函数会忽略输入中的空白字符，包括空格、制表符和换行符。因此，将 \n 放在 scanf 的格式字符串中通常不会对输入处理产生任何效果，也不会导致输入问题。

然而，如果你的意图是确保用户在输入后按下回车键，那么 \n 是不必要的，因为 scanf 在等待输入时，用户按下回车键是自然的行为。如果用户在输入数字后没有按下回车键，scanf 会等待直到回车键被按下。

```
return 0;
}投币功能出问题
```



在您提供的代码中，投币和找零的逻辑确实存在一些问题。以下是代码中相关部分的分析和改进方法：

#### 问题分析

1. **找零逻辑未实现**：变量 change 用于存储找零金额，但在代码中没有被正确计算。
2. **投币输入处理**：scanf("%d\n", &input); 中的 \n 可能会导致输入问题，通常不需要在格式字符串中包含换行符。

原来如此！

### 学习+尝试→问题二：

我的找零语句和“谢谢惠顾”和“输入错误”全部消失了，kimi 也没看出问题在哪。

直到我电脑卡了一次，他们一闪而过，我才发现，全是清屏 `system("cls");`搞的鬼

```
}  
if(total==0)  
{  
    printf("谢谢惠顾，欢迎下次光临！");  
}  
else  
{  
    printf("找零%d元",change);  
}  
break;
```

### 1-3:

题目要求 1:

在完成 Level1-1 和 Level1-2 之后,你已经可以搭建一个简易的自动售货机的工作流程,所以请将前两个 Level 的功能串联起来实现一个简单的售货机,要求除了正常实现前两个 Level 的功能,还需要允许用户在货物卖空之前重复购买。

关于重复购买:

我的整个程序是通过一个无限循环 (while (1)) 来实现的,这意味着程序会不断地执行,直到用户选择退出(例如,通过输入特定的命令)。在每次循环中,程序都会根据用户的选择来执行购买或存储商品的操作。这种设计允许用户在一个会话中多次购买或存储商品,直到他们选择退出程序。

```
while (1)  
{  
    system("cls"); // 清屏操作  
    printf("BUY or STORE\n");  
    scanf("%s", choice);  
  
    if (strcmp(choice, "BUY") == 0)  
    {  
        int total = 0;  
        system("cls"); // 清屏操作  
        printf("珍奇百货，任君挑选！\n");
```

题目要求 2:

考虑到实际情况,用户肯定会出现误操作的情况,会输入一些不合理的内容,我们希望你对这些不合理的内容进行处理。

我排除不合理输入的方法主要通过以下几个步骤实现:

1. 检查用户选择：
  - 程序首先询问用户是选择“BUY”还是“STORE”。如果输入既不是“BUY”也不是“STORE”，则提示用户输入无效，并要求重新输入。
2. 检查货架号范围：
  - 在购买商品的逻辑中，程序检查用户输入的货架号是否在 0 到 15 之间 (`location >= 0 && location < 16`)。这是因为数组 `channel` 的大小为 16，任何超出这个范围的输入都是不合理的。
3. 检查商品库存：
  - 程序还检查用户想要购买的数量是否小于或等于货架上的实际库存 (`channel[location].num >= num`)。如果用户尝试购买的商品数量超过了库存，程序会提示商品不足或货架号输入错误，并要求用户重试。
4. 检查投币金额：
  - 在结账时，程序要求用户投币，并检查投币金额是否为 1、2 或 5 (`if(input==1||input==2||input==5)`)。这是因为程序只接受这三种面额的硬币。如果用户输入了其他金额，程序会提示使用了无法识别的货币，并要求重新输入。
5. 检查存储输入：
  - 在存储商品的逻辑中，程序要求用户输入商品的位置、名称、数量和单价。虽然程序没有显式地检查这些输入的合理性，但它假设用户会按照提示正确输入。在实际应用中，可能还需要检查商品名称是否为空、数量和单价是否为正数等。
6. 使用循环和条件语句：
  - 程序使用 `while` 循环和 `if` 条件语句来不断检查用户的输入，并在发现不合理输入时提示用户重试。这种循环检查的方法确保了只有合理的输入才会被接受并处理。

通过这些方法，程序能够有效地排除不合理的输入，确保了用户操作的正确性和程序的健壮性。

题目要求 3:

**可以输出一些提示信息，让你的工作流程更加一目了然**

我认为，提示语句是用来与用户交互的，它们指导用户如何操作程序，以及在用户输入不符合预期时提供反馈。以下是我程序中一些主要的提示语句及其作用：

#### 1.主菜单提示：

```
printf("BUY or STORE\n");
```

这条语句提示用户他们可以选择“BUY”来购买商品，或者选择“STORE”来存储商品。

#### 2.购买模式下的提示：

```
system("cls");// 清屏操作
```

```
printf("珍奇百货，任君挑选！\n");
```

当用户选择“BUY”时，这条语句清空屏幕并告诉用户他们现在可以开始挑选商品。

#### 3.显示库存商品：

```

for (int i = 0; i < 16; i++) {
    if (channel[i].num > 0) {
        printf("%d: %s x %d, 单 价 : %d\n", i, channel[i].obj, channel[i].num,
channel[i].price);
    }
}

```

这个循环显示所有有库存的商品，包括货架号、商品名称、数量和单价。

#### 4. 购买商品输入提示：

```
printf("请输入货架号和购买数量（例如：3 2）：\n");
```

这条语句告诉用户他们需要输入货架号和想要购买的商品数量。

#### 5. 购买成功和结账提示：

```
printf("您购买了 %d 个 %s，总价为 %d\n", num, channel[location].obj,
channel[location].price * num);
```

```
printf("如果要结账，请输入 'END'，否则按任意键并且回车继续：\n");
```

当用户成功购买商品后，程序会显示购买的商品数量、名称和总价，并询问用户是否要结账。

#### 6. 投币提示：

```
printf("请投币(1 或 2 或 5):");
```

在结账过程中，程序提示用户投币，并指明接受的硬币面额。

#### 7. 投币错误提示：

```
printf("使用了无法识别的 money，请用 1 或 2 或 5 来输入！\n");
```

如果用户输入了不接受的硬币面额，程序会显示这条错误提示，并要求用户重新输入。

#### 8. 存储模式下的提示：

```
system("cls"); // 清屏操作
```

```
printf("货架空空，请君入瓮！\n");
```

当用户选择“STORE”时，这条语句清空屏幕并告诉用户他们现在可以开始存储商品。

#### 9. 存储商品输入提示：

```
printf("位置，物品，数量，单价（数据间需要空格分开）\n");
```

这条语句指导用户输入存储商品所需的信息，包括货架位置、商品名称、数量和单价。

#### 10. 结束输入提示：

```
printf("如果要结束输入，请输入 'END'，否则按任意键并且回车继续：\n");
```

在购买和存储模式下，这条语句告诉用户如果他们想要结束当前的操作，应该输入“END”。

#### 11. 无效输入提示：



```
printf("无法识别该输入，请输入 BUY 或者 STORE! \n");
```

如果用户输入了除了“BUY”或“STORE”之外的其他内容，程序会显示这条提示，并要求用户重新输入。

这些提示语句对于用户来说非常重要，因为它们提供了清晰的指导和反馈，确保用户能够正确地与程序交互。

## 2-1:

题目要求 1:

面对同学们日益增长的购物需求，售货机增加了货物的种类，并且将通道数增加到了 5 个（编号 1~5），你需要升级你的售货机以适应新功能：摆放货物功能需要支持不同种类的货物分别摆放在不同的通道中，并且不同的通道可以放相同种类的货物，先摆放完所有货物，再选择要购买的货物，当选择完所有需要购买的货物时，再进行投币。（可以学习状态机相关内容）

我发现用结构体储存数据就完全没这个烦恼，已经是 plus 升级版啦！

## 2-2:

题目要求 1:

为了让售货机的操作更人性化，学校对所有售货机进行了升级，新增了回退功能。每当用户对自己的操作感到后悔的时候就可以选择回退功能，撤销自己之前的操作，回退操作可以重复使用，但最多撤销三步操作。

我认为以我的水平是写不出这个代码的，扔到 chatgpt 里面搞出来的东西我也看不懂。但是，我认为可以更加人性化的增加“购物车”功能，这样用户就会有更好的购物体验。（顺便谈谈我对“撤销 3 步”的看法，如果一个用户在付了三次钱后，突然不想买他第一次选的商品了，怎么办？）

```
shopcar[location].num += num; // 更新购物车上的数量
```

但当我加上这个 shopcar 之后，灵光闪现

**既然我的购买全是输入整形作为数量，那输入为什么不能是负数呢？**

**如果再输入负数，岂不可以实现取消购买？**

**所以我只需要在结账前再进行一次提示，想要取消则可以回到购买界面，输入负数即可！**

反思不足：

槽点 1：没有真正做到撤回

槽点 2：输入负数没有限制，会出现初次结算时 total<0，售货机直接开始爆金币

所以我在程序之中又加入了一个 if 语句用于最基础的判断用户行为

1. 用户在购买商品后，程序会计算总价 total。



2. 如果在计算总价的过程中发现 total 小于 0 (这通常意味着用户尝试退货的数量超过了他们实际购买的数量, 或者输入了负数), 程序会执行以下操作:

```
if (total<0)
{
    printf("\n你的退货数量异常, 退货失败! (如果你想让本售货机爆金币那还是省省吧! )");
    channel[location].num += num; // 撤回操作
    total -= channel[location].price * num; // 总价复原
    sleep(2);
}
```

3. 在这段代码中, 首先打印一条消息告知用户退货数量异常, 退货失败。
4. 然后, 程序通过 channel[location].num += num; 将用户尝试退货的数量重新加回到货架上, 这样货架上的商品数量就恢复到了用户尝试退货之前的状态。
5. 接着, 程序通过 total -= channel[location].price \* num; 将总价调整回用户尝试退货之前的状态。
6. 最后, 程序使用 sleep(2); 暂停 2 秒钟, 可能是为了给用户一个短暂的停顿, 让他们意识到操作已经被撤销。

这段代码的目的是为了防止用户通过输入负数来获取额外的金钱或者使售货机出现错误。这是一种简单的错误处理机制, 确保了程序在面对不合理的输入时能够保持稳定运行。

但是它也并不完美, 假设用户购买了 3 个一元钱的 A 和 3 个两元钱的 B, 他可以选择退掉四个 B 而不引起程序报错!

**更改设想是把每一个相关数字变量再设置一组, 选择货物时不刷新, 支付成功后刷新数值, 并用他们来判断是否退货异常(已更改)**

设置 initial\_num 的思路

initial\_num 字段用于记录商品的初始数量, 即商品在被购买之前的数量。这样, 在用户购买商品后, 如果他们想要退货, 程序可以通过比较当前数量和 initial\_num 来判断用户是否尝试退货超过他们购买的数量, 从而避免异常退货。

选择货物时不刷新, 支付成功后刷新数值

在 BUY 操作中, 用户可以多次购买商品, 每次购买后, 程序不会立即更新 initial\_num。只有当用户输入 END 选择结账时, 程序才会更新 initial\_num 为当前的数量, 这样用户在结账前可以更改他们的购买决策。

判断是否退货异常

如果用户在购买后尝试退货, 程序会检查当前数量是否大于 initial\_num。如果是, 说明用户尝试退货的数量超过了他们购买的数量, 这是不允许的。程序会撤销这次购买操作, 并提示用户退货失败。

```
if (channel[location].num>channel[location].initial_num)
{
    printf("\n你的退货数量异常, 退货失败!");
    channel[location].num += num; // 撤回操作
    total -= channel[location].price * num; // 总价复原
    sleep(2);
}
```

还有一个问题，就是储存的时候，假如先在 2 存了 3 个 A，后在 2 存了 3 个 B，最后就显示存了 6 个 B，用 B 的价格出售。

更改设想是添加一个判断，必须是空的位置才能储存(已更改)

必须是空的位置才能储存的实现思路

在 STORE 操作中，程序首先检查用户选择的位置是否为空（`channel[location].num == 0`）或者该位置存储的商品与用户尝试存储的商品相同（`strcmp(channel[location].obj, &name) == 0`）。如果位置不为空且商品不同，则不允许存储，提示用户该位置已有不同商品。

```
if (channel[location].num == 0 || strcmp(channel[location].obj, &name) == 0) // 检查位置是否为空
{
    strcpy(channel[location].obj, &name); // 处理字符串需要strcpy
    channel[location].price = price;
    channel[location].num += num;
    channel[location].initial_num = channel[location].num; // 更新initial_num的数据
    printf("商品已存储在位置 %d\n", location);
}
else
{
    printf("位置 %d 已有不同商品，无法存储! \n", location);
}
```

# OVER