一、输入、输出缓冲区

1. 输入缓冲区

---》在不同速度的设备之间传递数据的时候需要使用缓冲区临时储存数据

---》scanf函数工作时会利用一个输入缓冲区把用户在键盘上输入的字符临时存储起来，程序实际上从这个输入缓冲区里获得数字

---》先进入输入缓冲区的数据必须优先处理

---》如果用户输入的内容和scanf函数要求的格式不一致则它们会一直无法被处理,这会导致后面输入的内容也无法处理

---》可以使用如下两条语句把输入缓冲区里可能存在的错误数据丢弃(见代码输入输出缓冲区)

scanf("%\*[^\n]"); //把输入缓冲区里第一个换行字符前的所有内容丢弃

scanf("%\*c"); //把输入缓冲区里第一个换行字符丢弃

2. 输出缓冲区

---》printf函数使用输出缓冲区临时存储要打印的内容

---》只有满足如下四个条件中的某一个，输出缓冲区里的内容才会显示在屏幕上

（1）'\n'换行字符前面的内容会被打印在屏幕上

（2）当输出缓冲区被充满的时候里面的内容会被打印在屏幕上

（3）当主函数结束后程序打印的内容出现在屏幕上

（4）可以使用fflush(stdout)语句把输出缓冲区里的内容强制显示在屏幕上

=============================================

二、数组

1. 数组名称不可以被赋值

---》计算机把数组当指针处理的(相关地址的数据访问)

1. 数组名既可以表示数组中第一个存储区的地址，也可以表示数组整体

---》计算机处理方式：arr[3] <=> \*(arr + 3)

3. 数组：表示内存中一组连续的同类型存储区

---》数组需要先声明然后才能使用

---》数组声明的时需要提供：数组名称、数据类型和存储区的个数

---》数组一旦被创建成功则里面所包含的存储区个数就固定了

---》数组通常不能作为整体使用，一般每次只使用其中的某一个存储区

---》数组中的每个存储区有一个编号，这个编号叫做这个存储区的下标（0~n-1）

---》超过范围的下标不可以使用，否则可能导致严重错误

---》下标是用来表示数组中某个存储区的唯一方法

4. 通常使用for循环依次处理数组中每个存储区

---》在这个循环里让循环变量依次代表每个存储区的下标

5. 声明数组时也应该初始化（数组初始化要求）

（1）应该把所有初始化数据写在一对大括号里，不同数据之间使用逗号分开

（2）所有初始化数据按照从前向后的顺序依次对每个存储区进行赋值

（3）如果初始化数据个数多于存储区个数则丢弃多余的数据

（4）如果初始化数据个数少于存储区个数则后面的存储区自动被初始化成0

（5）如果初始化初始化数据个数和存储区个数一样则可以省略存储区个数

6. 可以对数组名称进行sizeof计算，得到的结果是整个数组中包含的字节个数

---》可以利用sizeof关键字在程序中计算出数组中包含的存储区个数

---》sizeof(a)/a[0]

7. 变长数组

---》可以在声明数组时使用变量表示数组中存储区的个数，这种数组叫做变长数组

---》只有C99规范中才允许使用变长数组

---》变长数组不可以初始化，且赋值长度后，长度不可更改

---》变长数组在长度给定后，才可以声明

8. 多维数组

---》多维数组不但可以表示内存中一组连续的同类型存储区，同时还可以表示这些存储区的分组情况

---》多维数组中存储区可以多次进行分组

---》声明二维数组时需要指定一共分了几组，还需要指定每组中包含的存储区个数

---》二维数组中使用一个下标表示分组（组下标）还使用另外一个下标表示分组中的存储区（组内下标）

---》组下标的范围是从0到组个数减一

---》组内下标的范围是从0到组内存储区个数减一

---》可以采用一维数组的初始化语法对二维数组进行初始化

---》也可以在对二维数组进行初始化的时候把初始化数据进行分组，每组用来初始化二维数组中某一组存储区

---》如果可以根据初始化数据计算出分组个数则可以省略数组声明中的分组个数

一维数组名称可以代表数组中第一个存储区的地址

二维数组加一个下标可以代表这组中第一个存储区的地址

多维数组：首地址相同，类型决定视野

用gdb调试打印01arr.c/04arr.c中内容，以明确多维数组各种写法的含义(arr, &arr, &arr[0]等)