一. 字符串

1. C语言没有专门用于存储字符串的变量类型，字符串都被储存在char类型的数组中。

2. 声明：char name[40];（这里的40代表该数组中的元素数量）

3. 打印：打印所要用到的conversion specification是%s

4. 空字符（null character）:这是非打印字符，字符串末尾一定是以空字符结束的，好在它对其他的函数（如string.h中的strlen()函数）不可见。注意由于它的存在，设置字符串数组长度时一定要预留一个空间。

5. string.h中的strlen():string.h头文件中包含多个与字符串相关的函数原型，包括strlen()，它是用来统计字符串的**长度**。

6. sizeof关键字拓展:①当运算对象是类型时，圆括号必不可少，如sizeof(double)；当运算对象是特定量时，可有可无；如sizeof name;sizeof 1.1；②与strlen()的区别：对于已开辟一定空间的数组，sizeof的返回值是该空间的大小，而strlen()返回的是字符串的长度。e.g. char name[40];　　sizeof name == 40, 而strlen(name) == 0; 至于那些字符串宏，由于不知道事先开辟了多少空间，所以sizeof会把最后的空字符也算在内（空字符唯一的影响），所以sizeof的返回值要比strlen多一。

二.常量和C预处理器

1. 在C中，我们的常量一般都是用大写字母表示。

2. 三种特殊常量：

①C预处理器定义的常量：格式#define NAME value，他会在编译时替换文本。

②const关键字：格式const type NAME = value,用于限定一个**变量**为**只读**（只能作用于一定类型的变量）

③limits.h和float.h头文件中的特殊常量：分别提供了整型和浮点型大小限制的相关信息。

三.printf()函数

1. 格式：printf("格式字符串", 代打印列表)；

2. 返回值：打印的**字符**个数（包括\n）

3. conversion specification：重要的转换说明：%e（E）、%f、%g (G)用来打印相应浮点，%p用来打印指针，%%用来打印百分号。

4. 转换说明修饰符：

①最小字段宽度：设置字段宽度，如果打印值宽度更大将使用更宽的字段，e.g.%4d

②精度：格式：.数字；对于%f,%e,%E转换，表示小数点右边数字的位数。对

于%g，%G表示有效数字的最大位数。对于**整型**，表示带打印**数字的最小位数**（不足且有必要的情况下补0）；对于**字符串**，表示待打印字符的最大数量。

③l：和整型转换说明一起使用，表示long int或unsigned long int 类型

④L：和浮点转换说明一起使用，表示long double类型

⑤t：和整型转换说明一起使用，表示ptrdiff_t类型的值

⑥j：和整型转换说明一起使用，表示intmax_t或uintmax_t类型的值

⑦z：和整型转换说明一起使用，表示size_t类型的值

5. printf()中的标记

①-:表示左对齐；②+：正的显示+号，负的显示-号；③　空格：正的显示空格，负的显示负号；④#：转换为另一种格式；⑤0：用前导0填充字段宽度。

6. 关于参数的传递：程序把传入的值放入称为栈的内存区域，然后根据**变量类型**将值**放入**栈中，最后根据**转换说明**从栈中**读取**值。

7. 关于浮点float的转换说明：为了向下兼容，float一般都会自动转化为double类型的值。

8. 技巧：处理格式字符串太长的情况：①多用printf(); ②用\和Enter来分隔格式字符串（注意一个字符串不能直接用Enter分行，而参数之间是可以用多行来表示，编译器会忽略空白）③用字符串的连接，不能用逗号加号，只能用两个简单的字符串相邻来表示，如"a""b"等价于“ab”。

三.scanf()函数

1. 格式：scanf(格式字符串，变量指针列表)；（现在都使用scanf_s函数，使用方法相同）

2. 返回值：①一般返回读取成功的项数②读取错误或未设置读取返回0③检测到文件结尾返回EOF

3. 关于&：读取基本类型，要加&；读取数组类型，不用加&。

4. 关于空白：scanf()使用空白（如空格，制表符和换行符）将输入分为多个字段（唯一例外：%c，他也会读取空格）

5. conversion specification:与printf()大致相同，**注意这里引入了double的转换说明：%lf。**

6. 修饰符：与printf()大致相同，注意其中的数字代表输入达到**最大字段宽度**或第一次遇到空白字符时停止。e.g.%2d

7. scanf()是怎么读取输入的？①不断地跳过空白读取（详见书本）②**读到不符合的，就读取完毕，下一次读取从这个不符合的数开始。（与C++的cin不同，后者是直接卡死）**③一开始就无法读取的，将会一直停在那里。④对于字符串，上面的结论

依然适用，可见这里的字符串是读取单词，别忘了最后scanf会在字符串的末尾加一个\0。

8. 关于格式字符串：①要求除空格外的普通字符必须与输入字符串严格匹配。②格式字符串中的空白意味着跳过下一个输入项前面的所有空白。

四. scanf_s（）函数

1. 格式：scanf_s(格式字符串，转换说明1，**1所需长度**，转换说明2，**2所需长度**……)；

2. 存在的意义：（主要与scanf相比）

①scanf()在读取时不检查边界，所以可能会造成内存访问越界，例如分配了5字节的空间但是读入了10字节。

②**vs**默认scanf()不安全，在源代码中使用scanf甚至被视为错误，无法通过编译；现在你可以理解为什么要使用scanf了吧。

五.关于*修饰符

1. 在printf()中，可以用*来代替字段宽度，把一个输入的变量作为字段宽度来使
e.g.printf("%*d", width, number);

2. 在scanf()中，*代表抑制输出，简单地来说，就是跳过相应输出项，
e.g.scanf("%*d, %d", num);前两个输出将不会被读取。这在读取文件的特定列时特别有用。