给 C 语言初学者　使用 printf 格式化输出

printf 函数(还有它的同类 fprintf 和 sprintf)提供了一种打印信息的方式，这种方式对格式化细节有相当大的控制能力。第一个参数是格式串(format string)，而其余的参数都是要打印的值。在格式串里，每个以“%”开始的字符序列都表示如何格式化下一个参数。典型的示例包括：‘%d’是输出一个十进制整数，‘%f’是输出一个浮点数，而‘%c’是输出一个字符，其编码由参数给出。

指定确定大小数据类型的格式，如 int32_t，要更复杂一些，相关内容参见 2.2.3 节的旁注。

可以观察到，尽管浮点型和整型数据都是对数值 12 345 编码，但是它们有截然不同的字节模式：整型为 0x00003039，而浮点数为 0x4640E400。一般而言，这两种格式使用不同的编码方法。如果我们将这些十六进制模式扩展为二进制形式，并且适当地将它们移位，就会发现一个有 13 个相匹配的位的序列，用一串星号标识出来：

```
   0   0   0   0   3   0   3   9
00000000000000000011000000111001
                ************
     4   6   4   0   E   4   0   0
01000110010000001110010000000000
```

这并不是巧合。当我们研究浮点数格式时，还将再回到这个例子。

给 C 语言初学者　指针和数组

在函数 show_bytes(图 2-4)中，我们看到指针和数组之间紧密的联系，这将在 3.8 节中详细描述。这个函数有一个类型为 byte_pointer(被定义为一个指向 unsigned char 的指针)的参数 start，但是我们在第 8 行上看到数组引用 start[i]。在 C 语言中，我们能够用数组表示法来引用指针，同时我们也能用指针表示法来引用数组元素。在这个例子中，引用 start[i]表示我们想要读取以 start 指向的位置为起始的第 i 个位置处的字节。

给 C 语言初学者　指针的创建和间接引用

在图 2-4 的第 13、17 和 21 行，我们看到对 C 和 C++ 中两种独有操作的使用。C 的“取地址”运算符 & 创建一个指针。在这三行中，表达式 &x 创建了一个指向保存变量 x 的位置的指针。这个指针的类型取决于 x 的类型，因此这三个指针的类型分别为 int*、float*和 void **。(数据类型 void *是一种特殊类型的指针，没有相关联的类型信息。)

强制类型转换运算符可以将一种数据类型转换为另一种。因此，强制类型转换(byte_pointer)&x 表明无论指针 &x 以前是什么类型，它现在就是一个指向数据类型为 unsigned char 的指针。这里给出的这些强制类型转换不会改变真实的指针，它们只是告诉编译器以新的数据类型来看待被指向的数据。

旁注　生成一张 ASCII 表

可以通过执行命令 man ascii 来得到一张 ASCII 字符码的表。

练习题 2.5　思考下面对 show_bytes 的三次调用：

```
int val = 0x87654321;
byte_pointer valp = (byte_pointer) &val;
```