7.5 函数和 C-风格字符串

C-风格字符串由一系列字符组成，以空值字符结尾。前面介绍的大部分有关设计数组函数的知识也适用于字符串函数。

例如，将字符串作为参数时意味着传递的是地址，但可以使用 const 来禁止对字符串参数进行修改。下面首先介绍一些有关字符串的特殊知识。

7.5.1 将 C-风格字符串作为参数的函数

假设要将字符串作为参数传递给函数，则表示字符串的方式有 3 种：

- char 数组；
- 用引号括起的字符串常量（也称字符串字面值）；
- 被设置为字符串的地址的 char 指针。

但上述 3 种选择的类型都是 char 指针（准确地说是 char*），因此可以将其作为字符串处理函数的参数：

```
char ghost[15] = "galloping";
char * str = "galumphing";
int n1 = strlen(ghost);        // ghost is &ghost[0]
int n2 = strlen(str);          // pointer to char
int n3 = strlen("gamboling"); // address of string
```

可以说是将字符串作为参数来传递，但实际传递的是字符串第一个字符的地址。这意味着字符串函数原型应将其表示字符串的形参声明为 char *类型。

C-风格字符串与常规 char 数组之间的一个重要区别是，字符串有内置的结束字符（前面讲过，包含字符，但不以空值字符结尾的 char 数组只是数组，而不是字符串）。这意味着不必将字符串长度作为参数传递给函数，而函数可以使用循环依次检查字符串中的每个字符，直到遇到结尾的空值字符为止。程序清单 7.9 演示了这种方法，使用一个函数来计算特定的字符在字符串中出现的次数。由于该程序不需要处理负数，因此它将计数变量的类型声明为 unsigned int。

程序清单 7.9 strgfun.cpp

```
// strgfun.cpp -- functions with a string argument
#include <iostream>
unsigned int c_in_str(const char * str, char ch);
int main()
{
    using namespace std;
    char mmm[15] = "minimum"; // string in an array
// some systems require preceding char with static to
// enable array initialization

    char *wail = "ululate"; // wail points to string

    unsigned int ms = c_in_str(mmm, 'm');
    unsigned int us = c_in_str(wail, 'u');
    cout << ms << " m characters in " << mmm << endl;
    cout << us << " u characters in " << wail << endl;
    return 0;
}

// this function counts the number of ch characters
// in the string str
unsigned int c_in_str(const char * str, char ch)
{
    unsigned int count = 0;

    while (*str) // quit when *str is '\0'
    {
        if (*str == ch)
            count++;
        str++; // move pointer to next char
    }
    return count;
}
```