• An array of char

• A quoted string constant (also called a string literal)

• A pointer-to-char set to the address of a string

All three choices, however, are type pointer-to-char (more concisely, type char *), so you can use all three as arguments to string-processing functions:

```c
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
```

Informally, you can say you're passing a string as an argument, but you're really passing the address of the first character in the string. This implies that a string function prototype should use type char * as the type for the formal parameter representing a string.

One important difference between a string and a regular array is that the string has a built-in terminating character. (Recall that a char array containing characters but no null character is just an array and not a string.) That means you don't have to pass the size of the string as an argument. Instead, the function can use a loop to examine each character in the string in turn until the loop reaches the terminating null character. Listing 7.9 illustrates that approach with a function that counts the number of times a given character appears in a string.

```
Listing 7.9 strgfun.cpp
```

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