The special relationship between arrays and pointers extends to strings. Consider the following code:

```cpp
char flower[10] = "rose";
cout << flower << "s are red\n";
```

The name of an array is the address of its first element, so `flower` in the `cout` statement is the address of the `char` element containing the character `r`. The `cout` object assumes that the address of a `char` is the address of a string, so it prints the character at that address and then continues printing characters until it runs into the null character (`\0`). In short, if you give `cout` the address of a character, it prints everything from that character to the first null character that follows it.

The crucial element here is not that `flower` is an array name but that `flower` acts as the address of a `char`. This implies that you can use a pointer-to-`char` variable as an argument to `cout`, also, because it, too, is the address of a `char`. Of course, that pointer should point to the beginning of a string. We'll check that out in a moment.

But first, what about the final part of the preceding `cout` statement? If `flower` actually is the address of the first character of a string, what is the expression "s are red\n"? To be consistent with `cout`'s handling of string output, this quoted string also should be an address. And it is, for in C++ a quoted string, like an array name, serves as the address of its first element. The preceding code doesn't really send a whole string to `cout`, it just sends the string address. This means strings in an array, quoted string constants, and strings described by pointers all are handled equivalently. Each really is passed along as an address. That's certainly less work than passing each and every character in a string.

**Remember**

With `cout` and with most C++ expressions, the name of an array of `char`, a pointer-to-`char`, and a quoted string constant all are interpreted as the address of the first character of a string.

Listing 4.15 illustrates using the different forms of strings. It uses two functions from the string library. The `strlen()` function, which we've used before, returns the length of a string. The `strcpy()` function copies a string from one location to another. Both have function...