library function `strcpy()` to copy the string from `temp` to the new block. The function doesn’t check to see if the string fits or not, but `getname()` covers that by requesting the right number of bytes with `new`. Finally, the function returns `pn`, the address of the string copy.

In `main()`, the return value (the address) is assigned to the pointer `name`. This pointer is defined in `main()`, but it points to the block of memory allocated in the `getname()` function. The program then prints the string and the address of the string.

Next, after it frees the block pointed to by `name`, `main()` calls `getname()` a second time. C++ doesn’t guarantee that newly freed memory is the first to be chosen the next time `new` is used, and in this sample run, it isn’t.

Note in this example that `getname()` allocates memory and `main()` frees it. It’s usually not a good idea to put `new` and `delete` in separate functions because that makes it easier to forget to use `delete`. But this example does separate `new` from `delete` just to show that it is possible.

To appreciate some of the more subtle aspects of this program, you should know a little more about how C++ handles memory. So let’s preview some material that’s covered more fully in Chapter 9.

**Automatic Storage, Static Storage, and Dynamic Storage**

C++ has three ways of managing memory for data, depending on the method used to allocate memory: *automatic storage*, *static storage*, and *dynamic storage*, sometimes called the *free store* or *heap*. Data objects allocated in these three ways differ from each other in how long they remain in existence. We’ll take a quick look at each type.

**Automatic Variables**

Ordinary variables defined inside a function are called *automatic variables*. They come into existence automatically when the function containing them is invoked, and they expire when the function terminates. For example, the `temp` array in Listing 4.17 exists only while the `getname()` function is active. When program control returns to `main()`, the memory used for `temp` is freed automatically. If `getname()` had returned the address of `temp`, the `name` pointer in `main()` would have been left pointing to a memory location that soon