food[19] = '\0';

This copies up to 19 characters into the array and then sets the last element to the null character. If the string is shorter than 19 characters, strncpy() adds a null character earlier to mark the true end of the string.

**Remember**

Use strcpy() or strncpy(), not the assignment operator, to assign a string to an array.

**Using new to Create Dynamic Structures**

You've seen how it can be advantageous to create arrays during runtime rather than compile time. The same holds true for structures. You need to allocate space for only as many structures as a program needs during a particular run. Again, the new operator is the tool to use. With it, you can create dynamic structures. Again, "dynamic" means the memory is allocated during runtime, not during compilation. Incidentally, because classes are much like structures, you are able to use the techniques you learn for structures with classes, too.

Using new with structures has two parts: creating the structure and accessing its members. To create a structure, use the structure type with new. For example, to create an unnamed structure of the inflatable type and assign its address to a suitable pointer, you can do the following:

```c
inflatable * ps = new inflatable;
```

This assigns to ps the address of a chunk of free memory large enough to hold a structure of the inflatable type. Note that the syntax is exactly the same as it is for C++'s built-in types.

The tricky part is accessing members. When you create a dynamic structure, you can't use the dot membership operator with the structure name, because the structure has no name. All you have is its address. C++ provides an operator just for this situation: the arrow membership operator (->). This operator, formed by typing a hyphen and then a