

```
memcpy(a, b, sizeof(a));
```

Many programmers prefer `memcpy`, especially for large arrays, because it's potentially faster than an ordinary loop.

***Q:** Section 6.4 mentioned that C99 doesn't allow a `goto` statement to bypass the declaration of a variable-length array. What's the reason for this restriction?

A: The memory used to store a variable-length array is usually allocated when the declaration of the array is reached during program execution. Bypassing the declaration using a `goto` statement could result in a program accessing the elements of an array that was never allocated.

Exercises

Section 8.1

- 1. We discussed using the expression `sizeof(a) / sizeof(a[0])` to calculate the number of elements in an array. The expression `sizeof(a) / sizeof(t)`, where `t` is the type of `a`'s elements, would also work, but it's considered an inferior technique. Why?
- 2. The Q&A section shows how to use a *letter* as an array subscript. Describe how to use a *digit* (in character form) as a subscript.
- 3. Write a declaration of an array named `weekend` containing seven `bool` values. Include an initializer that makes the first and last values `true`; all other values should be `false`.
- 4. (C99) Repeat Exercise 3, but this time use a designated initializer. Make the initializer as short as possible.
- 5. The Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, ..., where each number is the sum of the two preceding numbers. Write a program fragment that declares an array named `fib_numbers` of length 40 and fills the array with the first 40 Fibonacci numbers. *Hint:* Fill in the first two numbers individually, then use a loop to compute the remaining numbers.

Section 8.2

- 6. Calculators, watches, and other electronic devices often rely on seven-segment displays for numerical output. To form a digit, such devices "turn on" some of the seven segments while leaving others "off":

```

  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐  ┌─┐
  │  │  │  │  │  │  │  │  │  │  │  │  │  │
  └─┘  └─┘  └─┘  └─┘  └─┘  └─┘  └─┘  └─┘  └─┘

```

Suppose that we want to set up an array that remembers which segments should be "on" for each digit. Let's number the segments as follows:

```

      0
    ┌─┐
    │  │
    │ 6 │
    │ 3 │
    └─┘

```

Here's what the array might look like, with each row representing one digit:

```
const int segments[10][7] = {{1, 1, 1, 1, 1, 1, 0}, ...};
```

I've given you the first row of the initializer; fill in the rest.