

8 Arrays

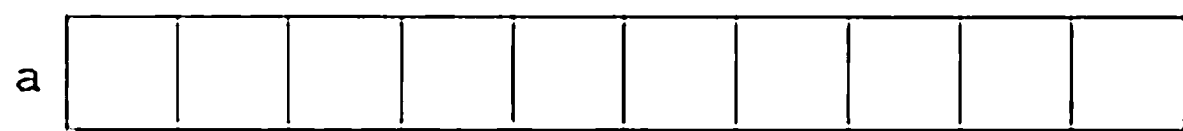
*If a program manipulates a large amount of data,
it does so in a small number of ways.*

So far, the only variables we've seen are *scalar*: capable of holding a single data item. C also supports *aggregate* variables, which can store collections of values. There are two kinds of aggregates in C: arrays and structures. This chapter shows how to declare and use arrays, both one-dimensional (Section 8.1) and multidimensional (Section 8.2). Section 8.3 covers C99's variable-length arrays. The focus of the chapter is on one-dimensional arrays, which play a much bigger role in C than do multidimensional arrays. Later chapters (Chapter 12 in particular) provide additional information about arrays; Chapter 16 covers structures.

8.1 One-Dimensional Arrays

An *array* is a data structure containing a number of data values, all of which have the same type. These values, known as *elements*, can be individually selected by their position within the array.

The simplest kind of array has just one dimension. The elements of a one-dimensional array are conceptually arranged one after another in a single row (or column, if you prefer). Here's how we might visualize a one-dimensional array named *a*:



To declare an array, we must specify the *type* of the array's elements and the *number* of elements. For example, to declare that the array *a* has 10 elements of type `int`, we would write

```
int a[10];
```