If the initializer is *shorter* than the array, the remaining elements of the array are given the value 0:

```
int a[10] = {1, 2, 3, 4, 5, 6};
  /* initial value of a is {1, 2, 3, 4, 5, 6, 0, 0, 0, 0} */
```

Using this feature, we can easily initialize an array to all zeros:

```
int a[10] = {0};
  /* initial value of a is {0, 0, 0, 0, 0, 0, 0, 0, 0, 0} */
```

It's illegal for an initializer to be completely empty, so we've put a single 0 inside the braces. It's also illegal for an initializer to be *longer* than the array it initializes.

If an initializer is present, the length of the array may be omitted:

```
int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

The compiler uses the length of the initializer to determine how long the array is. The array still has a fixed number of elements (10, in this example), just as if we had specified the length explicitly.

C99 Designated Initializers

It's often the case that relatively few elements of an array need to be initialized explicitly; the other elements can be given default values. Consider the following example:

```
int a[15] = {0, 0, 29, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 48};
```

We want element 2 of the array to be 29, element 9 to be 7, and element 14 to be 48, but the other values are just zero. For a large array, writing an initializer in this fashion is tedious and error-prone (what if there were 200 zeros between two of the nonzero values?).

C99's ***designated initializers*** can be used to solve this problem. Here's how we could redo the previous example using a designated initializer:

```
int a[15] = {[2] = 29, [9] = 7, [14] = 48};
```

Each number in brackets is said to be a ***designator***.

Besides being shorter and easier to read (at least for some arrays), designated initializers have another advantage: the order in which the elements are listed no longer matters. Thus, our previous example could also be written in the following way:

```
int a[15] = {[14] = 48, [9] = 7, [2] = 29};
```

Designators must be integer constant expressions. If the array being initialized has length n, each designator must be between 0 and $n - 1$. However, if the length of the array is omitted, a designator can be any nonnegative integer. In the latter case, the compiler will deduce the length of the array from the largest designator.