

type `int` (a signed type), whereas `sizeof` produces a value of type `size_t` (an unsigned type). We know from Section 7.4 that comparing a signed integer with an unsigned integer is a dangerous practice, although in this case it's safe because both `i` and `sizeof(a) / sizeof(a[0])` have nonnegative values. To avoid a warning, we can add a cast that converts `sizeof(a) / sizeof(a[0])` to a signed integer:

```
for (i = 0; i < (int) (sizeof(a) / sizeof(a[0])); i++)
    a[i] = 0;
```

Writing `(int) (sizeof(a) / sizeof(a[0]))` is a bit unwieldy; defining a macro that represents it is often helpful:

```
#define SIZE ((int) (sizeof(a) / sizeof(a[0])))

for (i = 0; i < SIZE; i++)
    a[i] = 0;
```

If we're back to using a macro, though, what's the advantage of `sizeof`? We'll answer that question in a later chapter (the trick is to add a parameter to the macro).

parameterized macros ► 14.3

PROGRAM Computing Interest

Our next program prints a table showing the value of \$100 invested at different rates of interest over a period of years. The user will enter an interest rate and the number of years the money will be invested. The table will show the value of the money at one-year intervals—at that interest rate and the next four higher rates—assuming that interest is compounded once a year. Here's what a session with the program will look like:

```
Enter interest rate: 6
Enter number of years: 5
```

Years	6%	7%	8%	9%	10%
1	106.00	107.00	108.00	109.00	110.00
2	112.36	114.49	116.64	118.81	121.00
3	119.10	122.50	125.97	129.50	133.10
4	126.25	131.08	136.05	141.16	146.41
5	133.82	140.26	146.93	153.86	161.05

Clearly, we can use a `for` statement to print the first row. The second row is a little trickier, since its values depend on the numbers in the first row. Our solution is to store the first row in an array as it's computed, then use the values in the array to compute the second row. Of course, this process can be repeated for the third and later rows. We'll end up with two `for` statements, one nested inside the other. The outer loop will count from 1 to the number of years requested by the user. The inner loop will increment the interest rate from its lowest value to its highest value.