Although we can use the `sizeof` operator to help determine the length of an array *variable*, it doesn't give the correct answer for an array *parameter*:

```
int f(int a[])
{
  int len = sizeof(a) / sizeof(a[0]);
    /*** WRONG: not the number of elements in a ***/
  ...
}
```

Section 12.3 explains why.

The following function illustrates the use of one-dimensional array arguments. When given an array `a` of `int` values, `sum_array` returns the sum of the elements in `a`. Since `sum_array` needs to know the length of `a`, we must supply it as a second argument.

```
int sum_array(int a[], int n)
{
  int i, sum = 0;

  for (i = 0; i < n; i++)
    sum += a[i];

  return sum;
}
```

The prototype for `sum_array` has the following appearance:

```
int sum_array(int a[], int n);
```

As usual, we can omit the parameter names if we wish:

```
int sum_array(int [], int);
```

When `sum_array` is called, the first argument will be the name of an array, and the second will be its length. For example:

```
#define LEN 100

int main(void)
{
  int b[LEN], total;
  ...
  total = sum_array(b, LEN);
  ...
}
```

Notice that we don't put brackets after an array name when passing it to a function:

```
total = sum_array(b[], LEN);    /*** WRONG ***/
```