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

  return sum;
}
```

Not being able to pass multidimensional arrays with an arbitrary number of columns can be a nuisance. Fortunately, we can often work around this difficulty by using arrays of pointers. C99's variable-length array parameters provide an even better solution to the problem.

arrays of pointers ➤13.7

C99 Variable-Length Array Parameters

C99 adds several new twists to array arguments. The first has to do with variable-length arrays (VLAs), a feature of C99 that allows the length of an array to be specified using a non-constant expression. Variable-length arrays can also be parameters, as it turns out.

variable-length arrays ➤8.3

Consider the sum_array function discussed earlier in this section. Here's the definition of sum_array, with the body omitted:

```
int sum_array(int a[], int n)
{
  ...
}
```

As it stands now, there's no direct link between n and the length of the array a. Although the function body treats n as a's length, the actual length of the array could in fact be larger than n (or smaller, in which case the function won't work correctly).

Using a variable-length array parameter, we can explicitly state that a's length is n:

```
int sum_array(int n, int a[n])
{
  ...
}
```

The value of the first parameter (n) specifies the length of the second parameter (a). Note that the order of the parameters has been switched; order is important when variable-length array parameters are used.

⚠ The following version of sum_array is illegal:

```
int sum_array(int a[n], int n)    /*** WRONG ***/
{
  ...
}
```

The compiler will issue an error message at int a[n], because it hasn't yet seen n.
