

```

int sum_two_dimensional_array(int n, int m, int a[n][m])
{
    int i, j, sum = 0;

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

    return sum;
}

```

Prototypes for this function include the following:

```

int sum_two_dimensional_array(int n, int m, int a[n][m]);
int sum_two_dimensional_array(int n, int m, int a[*][*]);
int sum_two_dimensional_array(int n, int m, int a[][m]);
int sum_two_dimensional_array(int n, int m, int a[][*]);

```

C99 Using `static` in Array Parameter Declarations

C99 allows the use of the keyword `static` in the declaration of array parameters. (The keyword itself existed before C99. Section 18.2 discusses its traditional uses.)

In the following example, putting `static` in front of the number 3 indicates that the length of `a` is guaranteed to be at least 3:

```

int sum_array(int a[static 3], int n)
{
    ...
}

```

Using `static` in this way has no effect on the behavior of the program. The presence of `static` is merely a “hint” that may allow a C compiler to generate faster instructions for accessing the array. (If the compiler knows that an array will always have a certain minimum length, it can arrange to “prefetch” these elements from memory when the function is called, before the elements are actually needed by statements within the function.)

One last note about `static`: If an array parameter has more than one dimension, `static` can be used only in the first dimension (for example, when specifying the number of rows in a two-dimensional array).

C99 Compound Literals

Let’s return to the original `sum_array` function one last time. When `sum_array` is called, the first argument is usually the name of an array (the one whose elements are to be summed). For example, we might call `sum_array` in the following way:

```

int b[] = {3, 0, 3, 4, 1};
total = sum_array(b, 5);

```