An Address Does Not Define an Array Length

```
int32_t min_value
    (int32_t const values[]);
```

same as `int32_t const* values`

Look good?

How can `min_value` know the array size?

As shown, it cannot.

So …?

Add a second parameter for the length.

Finding Minimum Value with a C Function

```
int32_t min_value
    (int32_t const values[], int32_t n_values)
{
    int32_t min = values[0];
    int32_t check;
    for (check=1; n_values > check; check++) {
        if (min > values[check]) {
            min = values[check];
        }
    }
    return min;
}
```

Assume first value is smallest.

In loop body, **`check`** goes from **`1`** to **`n_values - 1`**.

Finding Minimum Value with a C Function

```
int32_t min_value
    (int32_t const values[], int32_t n_values)
{
    int32_t min = values[0];
    int32_t check;
    for (check=1; n_values > check; check++) {
        if (min > values[check]) {
            min = values[check];
        }
    }
    return min;
}
```

If smaller value found, copy value to **`min`**.

Return smallest value found.

Using Our Minimum Value Function

How do we use the function?

```
int32_t my_nums[4] =
    {93, 100, 79, 42};
int32_t least;

least = min_value (my_nums, 4);
```

Initializes array to values shown.

pointer to **`my_nums[0]`**

Holds 42 after assignment.

length of my_nums

