The Allocated Size

Suppose that you have an array containing the names of all U.S. cities with populations of over 1,000,000. Taking data from the 2010 census, you would write:

```
const string cities[] = {
    "New York", "Los Angeles", "Chicago",
    "Houston", "Philadelphia", "Phoenix",
    "San Antonio", "San Diego", "Dallas",
};
```

However, the size of the cities **changes over time**. In 2020, both San Jose and Austin Texas joined the list. Fortunately, you **may** simply **add new names to the list**, or delete them, and then **let the compiler count how many there are**. This is so common, **you are allowed to leave a trailing comma** (such as the one after Dallas) and **it doesn't create a syntax error**.

So, **how do you know how many cities there are?** You don't want **to have to count** them! After all, **the compiler knows** how many there are. C++ has a **standard idiom** for determining the **allocated size** of an array **at compile time**, provided **the array definition is in scope**.

```
constexpr size_t kCitiesSize = sizeof(cities) / sizeof(cities[0]);
```

This compile-time expression takes the size of the entire array (returned from the **`sizeof`** operator, and thus of type **`size_t`**), and divides it by the size of the initial element in the array. Because all elements of an array are the same size, the result is the number of elements in the array, regardless of the element type.