Decaying Arrays

When you pass an array to a function, we say that the array "decays to a pointer". This is similar to what happens with primitive types in this case:

```
int n = 3.14;
```

The `int` variable `n` cannot store the fractional portion of `3.14`, so it **truncates the number** and stores `3`.

When you pass an array name to a function, and it is **converted into a pointer**, it also **loses certain information**; specifically, it **loses the ability to determine the allocated size of the array** inside the function.

When you **declare** the array, the compiler "knows" the allocated size of the array:

```
int array[] = {...};
size_t kLen = sizeof(array) / sizeof(array[0]);   // OK;
```

However, you pass that array to a function, you **cannot** use the same code.

```
void f(const int a[]) {
  size_t kBug = sizeof(a) / sizeof(a[0]);   // ERROR
}
```

That means we **must calculate an array size when the array is created**, and then **supply it when calling the function**.

This course content is offered under a CC Attribution Non-Commercial license. Content in this course can be considered under this license unless otherwise noted.