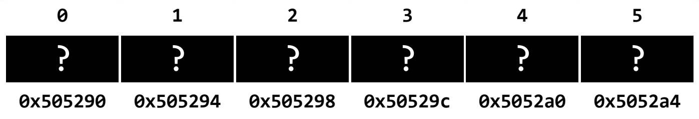
Defining Arrays

An array must be defined before it is used:

```
base-type name[size];
```

The definition requires a **base type**, **name**, and **allocated size**; **size** is a **positive integer constant expression** indicating the number of elements for the compiler to allocated. For example:

```
const size_t kSize = 6;
int a[kSize];
```



This creates an array named `a`, of `6` elements, each of which is an **uninitialized** `int`.

- A good practice is to **specify the size as a symbolic constant** instead of a literal.
- The size **must be positive**; zero or negative are illegal.
- The size must be **constant**; a **regular, non-const variable should not work**, although some compilers may permit it.
- If defined inside a function, the **elements are on the stack**; if defined outside of a function, the elements are allocated in the **static storage area**.

Index numbers begin with `0` and run up to the **array size minus one**.

C++ arrays are different than those in Java where the array variable and the allocated actual array are different. In C++ there is no array variable equivalent. Instead, the array name (`a` in the example) is an alias for the ***address of the first element****,* `0x505290` *here.*