The elements of an array may be of any type; the length of the array can be specified by any (integer) constant expression. Since array lengths may need to be adjusted when the program is later changed, using a macro to define the length of an array is an excellent practice:

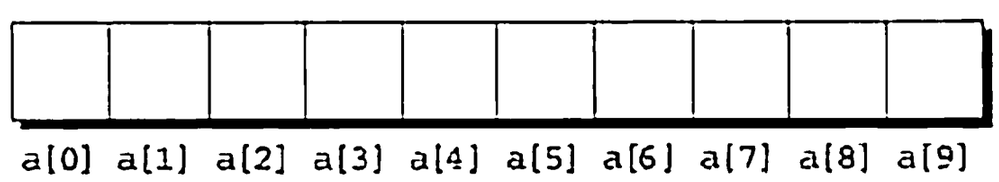
constant expressions ➤5.3

```
#define N 10
...
int a[N];
```

Array Subscripting

To access a particular element of an array, we write the array name followed by an integer value in square brackets (this is referred to as *subscripting* or *indexing* the array). Array elements are always numbered starting from 0, so the elements of an array of length *n* are indexed from 0 to *n* – 1. For example, if `a` is an array with 10 elements, they're designated by `a[0]`, `a[1]`, ..., `a[9]`, as the following figure shows:

Q&A



Expressions of the form `a[i]` are lvalues, so they can be used in the same way as ordinary variables:

lvalues ➤4.2

```
a[0] = 1;
printf("%d\n", a[5]);
++a[i];
```

In general, if an array contains elements of type *T*, then each element of the array is treated as if it were a variable of type *T*. In this example, the elements `a[0]`, `a[5]`, and `a[i]` behave like `int` variables.

Arrays and `for` loops go hand-in-hand. Many programs contain `for` loops whose job is to perform some operation on every element in an array. Here are a few examples of typical operations on an array `a` of length `N`:

idiom
```
for (i = 0; i < N; i++)
  a[i] = 0;                    /* clears a */
```

idiom
```
for (i = 0; i < N; i++)
  scanf("%d", &a[i]);          /* reads data into a */
```

idiom
```
for (i = 0; i < N; i++)
  sum += a[i];                 /* sums the elements of a */
```

Notice that we must use the & symbol when calling `scanf` to read an array element, just as we would with an ordinary variable.