```
double ident[2][2] = {[0][0] = 1.0, [1][1] = 1.0};
```

As usual, all elements for which no value is specified will default to zero.

Constant Arrays

Any array, whether one-dimensional or multidimensional, can be made "constant" by starting its declaration with the word `const`:

```
const char hex_chars[] =
  {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
   'A', 'B', 'C', 'D', 'E', 'F'};
```

An array that's been declared `const` should not be modified by the program; the compiler will detect direct attempts to modify an element.

Declaring an array to be `const` has a couple of primary advantages. It documents that the program won't change the array, which can be valuable information for someone reading the code later. It also helps the compiler catch errors, by informing it that we don't intend to modify the array.

`const` type qualifier ➤ *18.3*

`const` isn't limited to arrays; it works with any variable, as we'll see later. However, `const` is particularly useful in array declarations, because arrays may contain reference information that won't change during program execution.

PROGRAM

Dealing a Hand of Cards

Our next program illustrates both two-dimensional arrays and constant arrays. The program deals a random hand from a standard deck of playing cards. (In case you haven't had time to play games recently, each card in a standard deck has a *suit*—clubs, diamonds, hearts, or spades—and a *rank*—two, three, four, five, six, seven, eight, nine, ten, jack, queen, king, or ace.) We'll have the user specify how many cards should be in the hand:

```
Enter number of cards in hand: 5
Your hand: 7c 2s 5d as 2h
```

It's not immediately obvious how we'd write such a program. How do we pick cards randomly from the deck? And how do we avoid picking the same card twice? Let's tackle these problems separately.

`time` function ➤ *26.3*

To pick cards randomly, we'll use several C library functions. The `time` function (from `<time.h>`) returns the current time, encoded in a single number. The

`srand` function ➤ *26.2*

`srand` function (from `<stdlib.h>`) initializes C's random number generator. Passing the return value of `time` to `srand` prevents the program from dealing the

`rand` function ➤ *26.2*

same cards every time we run it. The `rand` function (also from `<stdlib.h>`) produces an apparently random number each time it's called. By using the `%` operator, we can scale the return value from `rand` so that it falls between 0 and 3 (for suits) or between 0 and 12 (for ranks).

To avoid picking the same card twice, we'll need to keep track of which cards have already been chosen. For that purpose, we'll use an array named `in_hand`