

Notice the initializer for the `in_hand` array:

```
bool in_hand[NUM_SUITS][NUM_RANKS] = {false};
```

Even though `in_hand` is a two-dimensional array, we can use a single pair of braces (at the risk of possibly incurring a warning from the compiler). Also, we've supplied only one value in the initializer, knowing that the compiler will fill in 0 (false) for the other elements.

8.3 Variable-Length Arrays (C99)

Section 8.1 stated that the length of an array variable must be specified by a constant expression. In C99, however, it's sometimes possible to use an expression that's *not* constant. The following modification of the `reverse.c` program (Section 8.1) illustrates this ability:

```
reverse2.c /* Reverses a series of numbers using a variable-length
              array - C99 only */

#include <stdio.h>

int main(void)
{
    int i, n;

    printf("How many numbers do you want to reverse? ");
    scanf("%d", &n);

    int a[n];    /* C99 only - length of array depends on n */

    printf("Enter %d numbers: ", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    printf("In reverse order:");
    for (i = n - 1; i >= 0; i--)
        printf(" %d", a[i]);
    printf("\n");

    return 0;
}
```

The array `a` in this program is an example of a *variable-length array* (or *VLA* for short). The length of a VLA is computed when the program is executed, not when the program is compiled. The chief advantage of a VLA is that the programmer doesn't have to pick an arbitrary length when declaring an array; instead, the program itself can calculate exactly how many elements are needed. If the programmer makes the choice, it's likely that the array will be too long (wasting memory) or too short (causing the program to fail). In the `reverse2.c` program, the num-