

PROGRAM Reversing a Series of Numbers

Our first array program prompts the user to enter a series of numbers, then writes the numbers in reverse order:

```
Enter 10 numbers: 34 82 49 102 7 94 23 11 50 31
In reverse order: 31 50 11 23 94 7 102 49 82 34
```

Our strategy will be to store the numbers in an array as they're read, then go through the array backwards, printing the elements one by one. In other words, we won't actually reverse the elements in the array, but we'll make the user think we did.

```
reverse.c /* Reverses a series of numbers */

#include <stdio.h>

#define N 10

int main(void)
{
    int a[N], i;

    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;
}
```

This program shows just how useful macros can be in conjunction with arrays. The macro `N` is used four times in the program: in the declaration of `a`, in the `printf` that displays a prompt, and in both `for` loops. Should we later decide to change the size of the array, we need only edit the definition of `N` and recompile the program. Nothing else will need to be altered; even the prompt will still be correct.

Array Initialization

An array, like any other variable, can be given an initial value at the time it's declared. The rules are somewhat tricky, though, so we'll cover some of them now and save others until later.

Initializers ► 18.5

The most common form of *array initializer* is a list of constant expressions enclosed in braces and separated by commas:

```
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```