To simplify the loop, we can replace &a[0] by a and &a[N] by a + N:

idiom
```
for (p = a; p < a + N; p++)
  sum += *p;
```

⚠ Although an array name can be used as a pointer, it's not possible to assign it a new value. Attempting to make it point elsewhere is an error:

```
while (*a != 0)
  a++;              /*** WRONG ***/
```

This is no great loss: we can always copy a into a pointer variable, then change the pointer variable:

```
p = a;
while (*p != 0)
  p++;
```

PROGRAM Reversing a Series of Numbers (Revisited)

The `reverse.c` program of Section 8.1 reads 10 numbers, then writes the numbers in reverse order. As the program reads the numbers, it stores them in an array. Once all the numbers are read, the program steps through the array backwards as it prints the numbers.

The original program used subscripting to access elements of the array. Here's a new version in which I've replaced subscripting with pointer arithmetic.

reverse3.c
```
/* Reverses a series of numbers (pointer version) */

#include <stdio.h>

#define N 10

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

  printf("Enter %d numbers: ", N);
  for (p = a; p < a + N; p++)
    scanf("%d", p);

  printf("In reverse order:");
  for (p = a + N - 1; p >= a; p--)
    printf(" %d", *p);
  printf("\n");

  return 0;
}
```

In the original program, an integer variable i kept track of the current position within the array. The new version replaces i with p, a pointer variable. The num-