```
current = va_arg(ap, int);
```

fetches max_int's remaining arguments, one by one, as it is executed inside a loop.

Don't forget that va_arg always advances to the next argument after fetching the current one. Because of this property, we couldn't have written max_int's loop in the following way:

```
for (i = 1; i < n; i++)
  if (va_arg(ap, int) > largest)    /*** WRONG ***/
    largest = va_arg(ap, int);
```

---

va_end

The statement

```
va_end(ap);
```

is required to "clean up" before the function returns. (Or, instead of returning, the function might call va_start and traverse the argument list again.)

va_copy

The va_copy macro copies src (a va_list value) into dest (also a va_list). The usefulness of va_copy lies in the fact that multiple calls of va_arg may have been made using src before it's copied into dest, thus processing some of the arguments. Calling va_copy allows a function to remember where it is in the argument list so that it can later return to the same point to reexamine an argument (and possibly the arguments that follow it).

Each call of va_start or va_copy must be paired with a call of va_end, and the calls must appear in the same function. All calls of va_arg must appear between the call of va_start (or va_copy) and the matching call of va_end.

default argument promotions ➤*9.3*

When a function with a variable argument list is called, the compiler performs the default argument promotions on all arguments that match the ellipsis. In particular, char and short arguments are promoted to int, and float values are promoted to double. Consequently, it doesn't make sense to pass types such as char, short, or float to va_arg, since arguments—after promotion—will never have one of those types.

Calling a Function with a Variable Argument List

Calling a function with a variable argument list is an inherently risky proposition. As far back as Chapter 3, we saw how dangerous it can be to pass the wrong arguments to printf and scanf. Other functions with variable argument lists are equally sensitive. The primary difficulty is that a function with a variable argument list has no way to determine the number of arguments or their types. This information must be passed into the function and/or assumed by the function. max_int relies on the first argument to specify how many additional arguments follow; it