```
((voltage <= max_voltage)?
  printf("Passed test: %s\n", "voltage <= max_voltage"):
  printf("Voltage %d exceeds %d\n", voltage, max_voltage));
```

When the program is executed, the program will display the message

```
Passed test: voltage <= max_voltage
```

if `voltage` is no more than `max_voltage`. Otherwise, it will display the values of `voltage` and `max_voltage`:

```
Voltage 125 exceeds 120
```

C99 The `__func__` Identifier

Another new feature of C99 is the `__func__` identifier. `__func__` has nothing to do with the preprocessor, so it actually doesn't belong in this chapter. However, like many preprocessor features, it's useful for debugging, so I've chosen to discuss it here.

Every function has access to the `__func__` identifier, which behaves like a string variable that stores the name of the currently executing function. The effect is the same as if each function contains the following declaration at the beginning of its body:

```
static const char __func__[] = "function-name";
```

where *function-name* is the name of the function. The existence of this identifier makes it possible to write debugging macros such as the following:

```
#define FUNCTION_CALLED() printf("%s called\n", __func__);
#define FUNCTION_RETURNS() printf("%s returns\n", __func__);
```

Calls of these macros can then be placed inside functions to trace their calls:

```
void f(void)
{
  FUNCTION_CALLED();   /* displays "f called" */
  ...
  FUNCTION_RETURNS();  /* displays "f returns" */
}
```

Another use of `__func__`: it can be passed to a function to let it know the name of the function that called it.

14.4 Conditional Compilation

The C preprocessor recognizes a number of directives that support ***conditional compilation***—the inclusion or exclusion of a section of program text depending on the outcome of a test performed by the preprocessor.