C99

- If the return type is omitted in C89, the function is presumed to return a value of type `int`. In C99, it's illegal to omit the return type of a function.

As a matter of style, some programmers put the return type *above* the function name:

```
double
average(double a, double b)
{
  return (a + b) / 2;
}
```

Putting the return type on a separate line is especially useful if the return type is lengthy, like `unsigned long int`.

Q&A

After the function name comes a list of parameters. Each parameter is preceded by a specification of its type; parameters are separated by commas. If the function has no parameters, the word `void` should appear between the parentheses. *Note:* A separate type must be specified for each parameter, even when several parameters have the same type:

```
double average(double a, b)    /*** WRONG ***/
{
  return (a + b) / 2;
}
```

The body of a function may include both declarations and statements. For example, the `average` function could be written

```
double average(double a, double b)
{
  double sum;          /* declaration */

  sum = a + b;         /* statement */
  return sum / 2;      /* statement */
}
```

C99

Variables declared in the body of a function belong exclusively to that function: they can't be examined or modified by other functions. In C89, variable declarations must come first, before all statements in the body of a function. In C99, variable declarations and statements can be mixed, as long as each variable is declared prior to the first statement that uses the variable. (Some pre-C99 compilers also allow mixing of declarations and statements.)

The body of a function whose return type is `void` (which I'll call a "`void` function") can be empty:

```
void print_pun(void)
{
}
```

Leaving the body empty may make sense during program development: we can leave room for the function without taking the time to complete it, then come back later and write the body.