Q: Some programming languages allow procedures and functions to be nested within each other. Does C allow function definitions to be nested?

A: No. C does not permit the definition of one function to appear in the body of another. Among other things, this restriction simplifies the compiler.

***Q: Why does the compiler allow the use of function names that aren't followed by parentheses? [p. 189]**

A: We'll see in a later chapter that the compiler treats a function name not followed by parentheses as a *pointer* to the function. Pointers to functions have legitimate uses, so the compiler can't automatically assume that a function name without parentheses is an error. The statement

pointers to functions ➤ 17.7

```
print_pun;
```

is legal because the compiler treats `print_pun` as a pointer and therefore an expression, making this a valid (although pointless) expression statement.

expression statements ➤ 4.5

***Q: In the function call `f(a, b)`, how does the compiler know whether the comma is punctuation or whether it's an operator?**

A: It turns out that the arguments in a function call can't be arbitrary expressions. Instead, they must be "assignment expressions," which can't contain commas used as operators unless they're enclosed in parentheses. In other words, in the call `f(a, b)` the comma is punctuation; in the call `f((a, b))` it's an operator.

Q: Do the names of parameters in a function prototype have to match the names given later in the function's definition? [p. 192]

A: No. Some programmers take advantage of this fact by giving long names to parameters in the prototype, then using shorter names in the actual definition. Or a French-speaking programmer might use English names in prototypes, then switch to more familiar French names in function definitions.

Q: I still don't understand why we bother with function prototypes. If we just put definitions of all the functions before `main`, we're covered, right?

A: Wrong. First, you're assuming that only `main` calls the other functions, which is unrealistic. In practice, some of the functions will call each other. If we put all function definitions above `main`, we'll have to watch their order carefully. Calling a function that hasn't been defined yet can lead to big problems.

But that's not all. Suppose that two functions call each other (which isn't as far-fetched as it may sound). No matter which function we define first, it will end up calling a function that hasn't been defined yet.

But there's still more! Once programs reach a certain size, it won't be feasible to put all the functions in one file anymore. When we reach that point, we'll need prototypes to tell the compiler about functions in other files.

Q: I've seen function declarations that omit all information about parameters: