declaration or definition required prior to function call
C99 requires that either a declaration or a definition of a function be present prior to any call of the function. C89 doesn't have this requirement; if a function is called without a prior declaration or definition, the compiler assumes that the function returns an `int` value.

variable-length array parameters
C99 allows variable-length array parameters. In a function declaration, the * symbol may appear inside brackets to indicate a variable-length array parameter.

`static` *array parameters*
C99 allows the use of the word `static` in the declaration of an array parameter, indicating a minimum length for the first dimension of the array.

compound literals
C99 supports the use of compound literals, which allow the creation of unnamed array and structure values.

declaration of `main`
C99 allows `main` to be declared in an implementation-defined manner, with a return type other than `int` and/or parameters other than those specified by the standard.

`return` *statement without expression*
In C89, executing a `return` statement without an expression in a non-`void` function causes undefined behavior (but only if the program attempts to use the value returned by the function). In C99, such a statement is illegal.

14 The Preprocessor

additional predefined macros
C99 provides several new predefined macros.

empty macro arguments
C99 allows any or all of the arguments in a macro call to be empty, provided that the call contains the correct number of commas.

macros with a variable number of arguments
In C89, a macro must have a fixed number of arguments, if it has any at all. C99 allows macros that take an unlimited number of arguments.

`__func__` *identifier*
In C99, the `__func__` identifier behaves like a string variable that stores the name of the currently executing function.

standard pragmas
In C89, there are no standard pragmas. C99 has three: `CX_LIMITED_RANGE`, `FENV_ACCESS`, and `FP_CONTRACT`.

`_Pragma` *operator*
C99 provides the `_Pragma` operator, which is used in conjunction with the `#pragma` directive.

16 Structures, Unions, and Enumerations

structure type compatibility
In C89, structures defined in different files are compatible if their members have the same names and appear in the same order, with corresponding members having