A K&R C function declaration omits all information about parameters:

```
double square();
int rand();
```

function calls When a K&R C definition or declaration is used, the compiler doesn't check that the function is called with arguments of the proper number and type. Furthermore, the arguments aren't automatically converted to the types of the corresponding parameters. Instead, the integral promotions are performed, and `float` arguments are converted to `double`.

`void` K&R C doesn't support the `void` type.

12 Pointers and Arrays

pointer subtraction Subtracting two pointers produces an `int` value in K&R C but a `ptrdiff_t` value in C89.

13 Strings

string literals In K&R C, adjacent string literals aren't concatenated. Also, K&R C doesn't prohibit the modification of string literals.

string initialization In K&R C, an initializer for a character array of length n is limited to $n - 1$ characters (leaving room for a null character at the end). C89 allows the initializer to have length n.

14 The Preprocessor

`#elif`, `#error`, `#pragma` K&R C doesn't support the `#elif`, `#error`, and `#pragma` directives.

`#`, `##`, `defined` K&R C doesn't support the `#`, `##`, and `defined` operators.

16 Structures, Unions, and Enumerations

structure and union members and tags In C89, each structure and union has its own name space for members; structure and union tags are kept in a separate name space. K&R C uses a single name space for members and tags, so members can't have the same name (with some exceptions), and members and tags can't overlap.

whole-structure operations K&R C doesn't allow structures to be assigned, passed as arguments, or returned by functions.

enumerations K&R C doesn't support enumerations.