18.3 Type Qualifiers

C99

restricted pointers ➤ 17.8

There are two type qualifiers: `const` and `volatile`. (C99 has a third type qualifier, `restrict`, which is used only with pointers.) Since the use of `volatile` is limited to low-level programming, I'll postpone discussing it until Section 20.3. `const` is used to declare objects that resemble variables but are "read-only": a program may access the value of a `const` object, but can't change it. For example, the declaration

```
const int n = 10;
```

creates a `const` object named `n` whose value is 10. The declaration

```
const int tax_brackets[] = {750, 2250, 3750, 5250, 7000};
```

creates a `const` array named `tax_brackets`.

Declaring an object to be `const` has several advantages:

- It's a form of documentation: it alerts anyone reading the program to the read-only nature of the object.
- The compiler can check that the program doesn't inadvertently attempt to change the value of the object.
- When programs are written for certain types of applications (embedded systems, in particular), the compiler can use the word `const` to identify data to be stored in ROM (read-only memory).

At first glance, it might appear that `const` serves the same role as the `#define` directive, which we've used in previous chapters to create names for constants. There are significant differences between `#define` and `const`, however:

- We can use `#define` to create a name for a numerical, character, or string constant. `const` can be used to create read-only objects of *any* type, including arrays, pointers, structures, and unions.
- `const` objects are subject to the same scope rules as variables; constants created using `#define` aren't. In particular, we can't use `#define` to create a constant with block scope.
- The value of a `const` object, unlike the value of a macro, can be viewed in a debugger.

Q&A

- Unlike macros, `const` objects can't be used in constant expressions. For example, we can't write

```
const int n = 10;
int a[n];               /*** WRONG ***/
```

C99

since array bounds must be constant expressions. (In C99, this example would