0111111111111111

which has the value 32,767 ($2^{15} - 1$). The largest 32-bit integer is

01111111111111111111111111111111

which has the value 2,147,483,647 ($2^{31} - 1$). An integer with no sign bit (the leftmost bit is considered part of the number's magnitude) is said to be ***unsigned***. The largest 16-bit unsigned integer is 65,535 ($2^{16} - 1$), and the largest 32-bit unsigned integer is 4,294,967,295 ($2^{32} - 1$).

By default, integer variables are signed in C—the leftmost bit is reserved for the sign. To tell the compiler that a variable has no sign bit, we declare it to be `unsigned`. Unsigned numbers are primarily useful for systems programming and low-level, machine-dependent applications. We'll discuss typical applications for unsigned numbers in Chapter 20; until then, we'll generally avoid them.

C's integer types come in different sizes. The `int` type is usually 32 bits, but may be 16 bits on older CPUs. Since some programs require numbers that are too large to store in `int` form, C also provides ***long*** integers. At times, we may need to conserve memory by instructing the compiler to store a number in less space than normal; such a number is called a ***short*** integer.

To construct an integer type that exactly meets our needs, we can specify that a variable is `long` or `short`, `signed` or `unsigned`. We can even combine specifiers (e.g., `long unsigned int`). However, only the following six combinations actually produce different types:

```
short int
unsigned short int

int
unsigned int

long int
unsigned long int
```

Other combinations are synonyms for one of these six types. (For example, `long signed int` is the same as `long int`, since integers are always signed unless otherwise specified.) Incidentally, the order of the specifiers doesn't matter; `unsigned short int` is the same as `short unsigned int`.

C allows us to abbreviate the names of integer types by dropping the word `int`. For example, `unsigned short int` may be abbreviated to `unsigned short`, and `long int` may be abbreviated to just `long`. Omitting `int` is a widespread practice among C programmers, and some newer C-based languages (including Java) actually require the programmer to write `short` or `long` rather than `short int` or `long int`. For these reasons, I'll often omit the word `int` when it's not strictly necessary.