second, returning an `imaxdiv_t` value. `imaxdiv_t` is a structure that contains both a quotient member (named `quot`) and a remainder member (`rem`); both members have type `intmax_t`.

strtoimax
strtoumax

The `strtoimax` and `strtoumax` functions are greatest-width versions of the numeric conversion functions of `<stdlib.h>`. The `strtoimax` function is the same as `strtol` and `strtoll`, except that it returns a value of type `intmax_t`. The `strtoumax` function is equivalent to `strtoul` and `strtoull`, except that it returns a value of type `uintmax_t`. Both `strtoimax` and `strtoumax` return zero if no conversion could be performed. Both functions store `ERANGE` in `errno` if a conversion produces a value that's outside the range of the function's return type. In addition, `strtoimax` returns the smallest or largest `intmax_t` value (`INTMAX_MIN` or `INTMAX_MAX`); `strtoumax` returns the largest `uintmax_t` value, `UINTMAX_MAX`.

wcstoimax
wcstoumax

<wchar.h> header ➤25.5

The `wcstoimax` and `wcstoumax` functions are greatest-width versions of the wide-string numeric conversion functions of `<wchar.h>`. The `wcstoimax` function is the same as `wcstol` and `wcstoll`, except that it returns a value of type `intmax_t`. The `wcstoumax` function is equivalent to `wcstoul` and `wcstoull`, except that it returns a value of type `uintmax_t`. Both `wcstoimax` and `wcstoumax` return zero if no conversion could be performed. Both functions store `ERANGE` in `errno` if a conversion produces a value that's outside the range of the function's return type. In addition, `wcstoimax` returns the smallest or largest `intmax_t` value (`INTMAX_MIN` or `INTMAX_MAX`); `wcstoumax` returns the largest `uintmax_t` value, `UINTMAX_MAX`.

27.3 Complex Numbers (C99)

Complex numbers are used in scientific and engineering applications as well as in mathematics. C99 provides several complex types, allows operators to have complex operands, and adds a header named `<complex.h>` to the standard library. There's a catch, though: complex numbers aren't supported by all implementations of C99. Section 14.3 discussed the difference between a *hosted* C99 implementation and a *freestanding* implementation. A hosted implementation must accept any program that conforms to the C99 standard, whereas a freestanding implementation doesn't have to compile programs that use complex types or standard headers other than `<float.h>`, `<iso646.h>`, `<limits.h>`, `<stdarg.h>`, `<stdbool.h>`, `<stddef.h>`, and `<stdint.h>`. Thus, a freestanding implementation may lack both complex types and the `<complex.h>` header.

We'll start with a review of the mathematical definition of complex numbers and complex arithmetic. We'll then look at C99's complex types and the operations that can be performed on values of these types. Coverage of complex numbers continues in Section 27.4, which describes the `<complex.h>` header.