27.4 The <complex.h> Header (C99): Complex Arithmetic

As we saw in Section 27.3, C99 has significant built-in support for complex numbers. The <complex.h> header provides additional support in the form of mathematical functions on complex numbers, as well as some very useful macros and a pragma. Let's look at the macros first.

<complex.h> Macros

The <complex.h> header defines the macros shown in Table 27.4.

Table 27.4 <complex.h> Macros

Name	*Value*
complex	_Complex
_Complex_I	Imaginary unit; has type const float _Complex
I	_Complex_I

complex serves as an alternative name for the awkward _Complex keyword. We've seen a situation like this before with the Boolean type: the C99 committee chose a new keyword (_Bool) that shouldn't break existing programs, but provided a better name (bool) as a macro defined in the <stdbool.h> header. Programs that include <complex.h> may use complex instead of _Complex, just as programs that include <stdbool.h> may use bool rather than _Bool.

<stdbool.h> header ➤21.5

The I macro plays an important role in C99. There's no special language feature for creating a complex number from its real part and imaginary part. Instead, a complex number can be constructed by multiplying the imaginary part by I and adding the real part:

```
double complex dc = 2.0 + 3.5 * I;
```

The value of the variable dc is $2 + 3.5i$.

Note that both _Complex_I and I represent the imaginary unit *i*. Presumably most programmers will use I rather than _Complex_I. However, since I might already be used in existing code for some other purpose, _Complex_I is available as a backup. If the name I causes a conflict, it can always be undefined:

```
#include <complex.h>
#undef I
```

The programmer might then define a different—but still short—name for *i*, such as J:

```
#define J _Complex_I
```