C99 provides three complex types, which were first introduced in Section 7.2: `float _Complex`, `double _Complex`, and `long double _Complex`. These types can be used in the same way as other types in C: to declare variables, parameters, return types, array elements, members of structures and unions, and so forth. For example, we could declare three variables as follows:

```
float _Complex x;
double _Complex y;
long double _Complex z;
```

Each of these variables is stored just like an array of two ordinary floating-point numbers. Thus, `y` is stored as two adjacent `double` values, with the first value containing the real part of `y` and the second containing the imaginary part.

C99 also allows implementations to provide imaginary types (the keyword `_Imaginary` is reserved for this purpose) but doesn't make this a requirement.

Operations on Complex Numbers

Complex numbers may be used in expressions, although only the following operators allow complex operands:

- Unary + and -
- Logical negation (`!`)
- `sizeof`
- Cast
- Multiplicative (`*` and `/` only)
- Additive (`+` and `-`)
- Equality (`==` and `!=`)
- Logical *and* (`&&`)
- Logical *or* (`||`)
- Conditional (`? :`)
- Simple assignment (`=`)
- Compound assignment (`*=`, `/=`, `+=`, and `-=` only)
- Comma (`,`)

Some notable omissions from the list include the relational operators (`<`, `<=`, `>`, and `>=`), along with the increment (`++`) and decrement (`--`) operators.

Conversion Rules for Complex Types

Section 7.4 described the C99 rules for type conversion, but without covering the complex types. It's now time to rectify that situation. Before we get to the conversion rules, though, we'll need some new terminology. For each floating type there is a ***corresponding real type.*** In the case of the real floating types (`float`, `double`, and `long double`), the corresponding real type is the same as the original type.