The macros in the final group (Table 27.7) correspond only to functions in <complex.h>.

Table 27.7
Type-Generic Macros in <tgmath.h> (Group 3)

carg	conj	creal
cimag	cproj	

Q&A Between the three tables, all functions in <math.h> and <complex.h> that have multiple versions are accounted for, with the exception of modf.

Invoking a Type-Generic Macro

To understand what happens when a type-generic macro is invoked, we first need the concept of a *generic parameter*. Consider the prototypes for the three versions of the nextafter function (from <math.h>):

```
double nextafter(double x, double y);
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);
```

The types of both x and y change depending on the version of nextafter, so both parameters are generic. Now consider the prototypes for the three versions of the nexttoward function:

```
double nexttoward(double x, long double y);
float nexttowardf(float x, long double y);
long double nexttowardl(long double x, long double y);
```

The first parameter is generic, but the second is not (it always has type long double). Generic parameters always have type double (or double complex) in the unsuffixed version of a function.

When a type-generic macro is invoked, the first step is to determine whether it should be replaced by a <math.h> function or a <complex.h> function. (This step doesn't apply to the macros in Table 27.6, which are always replaced by a <math.h> function, or the macros in Table 27.7, which are always replaced by a <complex.h> function.) The rule is simple: if any argument corresponding to a generic parameter is complex, then a <complex.h> function is chosen; otherwise, a <math.h> function is selected.

The next step is to deduce which version of the <math.h> function or <complex.h> function is being called. Let's assume that the function being called belongs to <math.h>. (The rules for the <complex.h> case are analogous.) The following rules are used, in the order listed:

1. If any argument corresponding to a generic parameter has type long double, the long double version of the function is called.
2. If any argument corresponding to a generic parameter has type double or any integer type, the double version of the function is called.
3. Otherwise, the float version of the function is called.

Q&A Rule 2 is a little unusual: it states that an integer argument causes the double version of a function to be called, not the float version, which you might expect.