One advantage of using <tgmath.h> is that calls of math functions become easier to write (and read!). More importantly, a call of a type-generic macro won't have to be modified in the future should the type of its argument(s) change.

The <tgmath.h> header includes both <math.h> and <complex.h>, by the way, so including <tgmath.h> provides access to the functions in both headers.

Type-Generic Macros

The type-generic macros defined in the <tgmath.h> header fall into three groups, depending on whether they correspond to functions in <math.h>, <complex.h>, or both headers.

Table 27.5 lists the type-generic macros that correspond to functions in both <math.h> and <complex.h>. Note that the name of each type-generic macro matches the name of the "unsuffixed" <math.h> function (acos as opposed to acosf or acosl, for example).

Table 27.5
Type-Generic Macros in <tgmath.h> (Group 1)

<math.h> *Function*	<complex.h> *Function*	*Type-Generic Macro*
acos	cacos	acos
asin	casin	asin
atan	catan	atan
acosh	cacosh	acosh
asinh	casinh	asinh
atanh	catanh	atanh
cos	ccos	cos
sin	csin	sin
tan	ctan	tan
cosh	ccosh	cosh
sinh	csinh	sinh
tanh	ctanh	tanh
exp	cexp	exp
log	clog	log
pow	cpow	pow
sqrt	csqrt	sqrt
fabs	cabs	fabs

The macros in the second group (Table 27.6) correspond only to functions in <math.h>. Each macro has the same name as the unsuffixed <math.h> function. Passing a complex argument to any of these macros causes undefined behavior.

Table 27.6
Type-Generic Macros in <tgmath.h> (Group 2)

atan2	fma	llround	remainder
cbrt	fmax	log10	remquo
ceil	fmin	log1p	rint
copysign	fmod	log2	round
erf	frexp	logb	scalbn
erfc	hypot	lrint	scalbln
exp2	ilogb	lround	tgamma
expm1	ldexp	nearbyint	trunc
fdim	lgamma	nextafter	
floor	llrint	nexttoward	