Complex arithmetic <complex.h>

WG14/N471 X3J11/95-072 (Draft 8/24/95)

Jim Thomas
Taligent, Inc.
10201 N. DeAnza Blvd.
Cupertino, CA 95014-2233
`jim_thomas@taligent.com`

7.x Complex arithmetic <complex.h>

The header `<complex.h>` defines macros and declares functions that support complex arithmetic.

The macro

```
_Imaginary_I
```

expands to an expression with the const-qualified imaginary type that corresponds to the narrowest real floating type used for expression evaluation[1] and with the value of the imaginary unit.

The macro

```
I
```

is defined to be `_Imaginary_I`.

[Positing such a constant is a natural analog to the mathematical notion of augmenting the reals with the imaginary unit. It allows writing imaginary and complex expressions in common mathematical style, for example `x + y*I`. Note that the multiplication here affects translated code, but does not necessitate an actual floating-point multiply, nor does the addition necessitate a floating-point add.

The choice of `I` instead of `i` concedes to the widespread use of the identifier `i` for other purposes.

The scheme for defining `I` and `_Imaginary_I` establishes a degree of uniformity in the designation of the imaginary unit as `I`, but offers flexibility to use a different identifier for the imaginary unit and to use `I` for other purposes. For example, one might follow the inclusion of `<complex.h>` with

```
#define j _Imaginary_I
#undef I
```

An `i` suffix to designate imaginary constants is not required. Multiplication by `I` provides a sufficiently convenient and more generally useful notation for imaginary terms.

Lacking an imaginary type, other proposals required macros in order to create certain special values. For example, an "imaginary" infinity could be created by `CMPLX(0.0,INFINITY)`. With the imaginary type, imaginary infinity is simply the value of `INFINITY*I`. And, in general, the values of *y*I* and *x + y*I*, where *x* and *y* are real floating

---

[1] If `FLT_EVAL_METHOD` equals 0, 1, or 2, then `_Imaginary_I` has type `float imaginary`, `double imaginary`, or `long double imaginary`, respectively.

595

values, cover all values of the imaginary and complex types, hence eliminating this need for the complex macros.]

7.x.1 Overloading

Some overloading macros accept imaginary and complex arguments. A floating type is characterized by its kind (real, imaginary, complex) and by its corresponding real type (`float`, `double`, `long double`). For each overloading parameter, the kind matches the argument kind, and the corresponding real type (which is the same for all overloading parameters of a given macro) is the wider of

— the corresponding real types for all floating arguments to the overloading parameters
— the narrowest real floating format used for expression evaluation

For the return value, the kind is determined by the function and the argument kind(s), and the corresponding real type matches the corresponding real type of the overloading parameters.

Examples

1. The `cos` macro which has the form

```
floating-type cos(floating-type x);
```

accepts imaginary and complex arguments. The following declare imaginary and complex variables:

```
float  imaginary fi;
double imaginary di;
long double imaginary ldi;
float  complex fc;
double complex dc;
long double complex ldc;
```

With these declarations, the following illustrate the determination of return types (`cos` of an imaginary is real):

If `FLT_EVAL_METHOD` equals 0, then `cos(fi)`, `cos(di)`, and `cos(ldi)` have types `float`, `double`, and `long double`, respectively, and `cos(fc)`, `cos(dc)`, and `cos(ldc)` have types `float complex`, `double complex`, and `long double complex`, respectively.

If `FLT_EVAL_METHOD` equals 1, then `cos(fi)`, `cos(di)`, and `cos(ldi)` have types `double`, `double`, and `long double`, respectively, and `cos(fc)` `cos(dc)`, and `cos(ldc)` have types `double complex`, `double complex`, and `long double complex`, respectively.

If `FLT_EVAL_METHOD` equals 3, then `cos(fi)`, `cos(di)`, and `cos(ldi)` all have type `long double`, and `cos(fc)`, `cos(dc)`, and `cos(ldc)` all have type `long double complex`.

Use a table for example 1. Add an example with 2 arguments to illustrate parameter types.

7.x.2 `<fp.h>` macros

When the header `<complex.h>` is included, these overloading macros from `<fp.h>` accept imaginary and complex arguments:

```
acos     cos     acosh     cosh     exp     fabs
asin     sin     asinh     sinh     log
atan     tan     atanh     tanh     sqrt
```

The return type for **fabs** is always real. Otherwise, for all these macros, if the argument is complex then so is the return type. If the argument is imaginary then the return type is real, imaginary, or complex, as appropriate for the particular macro: if the argument is imaginary, then the return types of **cos** and **cosh** are real, the return types of **sin**, **tan**, **sinh**, **tanh**, **asin**, and **atanh** are imaginary, and the return types of the others are complex.

7.x.2.1 Branch cuts

Some of the functions below have branch cuts, across which the function is discontinuous. For implementations with a signed zero (including all IEEE implementations), the sign of zero distinguishes one side of a cut from another so the function is continuous (except for format limitations) as the cut is approached from either side. For example, for the square root function, which has a branch cut along the negative real axis, the top of the cut, with imaginary part +0, maps to the positive imaginary axis, and the bottom of the cut, with imaginary part -0, maps to the negative imaginary axis.

Implementations with an unsigned zero cannot distinguish the sides of branch cuts. They map a cut so the function is continuous as the cut is approached coming around the finite endpoint of the cut in a counter clockwise direction. (Branch cuts for the functions specified here have just one finite endpoint.) For example, for the square root function, coming counter clockwise around the finite endpoint of the cut along the negative real axis approaches the cut from above, so the cut maps to the positive imaginary axis.

7.x.3 Trigonometric macros

7.x.3.1 The acos macro

Synopsis

```
#include <complex.h>
complex-type acos(complex-or-imaginary-type z);
```

Description

The **acos** macro computes the complex arc cosine of **z**, with branch cuts outside the interval [-1, 1] along the real axis.

597

Returns

The `acos` macros returns the complex arc cosine of `z`, in the range of a strip mathematically unbounded along the imaginary axis and in the interval $[0, \pi]$ along the real axis.

7.x.3.2 The `asin` macro

Synopsis

```
#include <complex.h>
imaginary-type asin(imaginary-type z);
complex-type asin(complex-type z);
```

Description

The `asin` macro computes the complex arc sine of `z`, with branch cuts outside the interval [-1, 1] along the real axis.

Returns

The `asin` macro returns the complex arc sine of `z`, in the range of a strip mathematically unbounded along the imaginary axis and in the interval $[-\pi/2, \pi/2]$ along the real axis.

7.x.3.3 The atan macro

Synopsis

```
#include <complex.h>
complex-type atan(complex-or-imaginary-type z);
```

Description

The `atan` macro computes the complex arc tangent of `z`, with branch cuts outside the interval $[-i, i]$ along the imaginary axis.

Returns

The `atan` macro returns the complex arc tangent of `z`, in the range of a strip mathematically unbounded along the imaginary axis and in the interval $[-\pi/2, \pi/2]$ along the real axis.

7.x.3.4 The cos macro

Synopsis

```
#include <complex.h>
real-type cos(imaginary-type z);
complex-type cos(complex-type z);
```

Description

The `cos` macro computes the complex cosine of `z`.

Returns

The `cos` macro returns the complex cosine of `z`.

7.x.3.5 The sin macro

Synopsis

```
#include <complex.h>
imaginary-type sin(imaginary-type z);
complex-type sin(complex-type z);
```

Description

The `sin` macro computes the complex sine of `z`.

Returns

The `sin` macro returns the complex sine of `z`.

7.x.3.6 The tan macro

Synopsis

```
#include <complex.h>
imaginary-type tan(imaginary-type z);
complex-type tan(complex-type z);
```

Description

The `tan` macro computes the complex tangent of `z`.

Returns

The `tan` macro returns the complex tangent of `z`.

7.x.4 Hyperbolic functions

7.x.4.1 The acosh macro

Synopsis

```
#include <complex.h>
complex-type acosh(complex-or-imaginary-type z);
```

Description

The `acosh` macro computes the complex arc hyperbolic cosine of `z`, with a branch cut at values less than 1 along the real axis.

599

Returns

The **acosh** macro returns the complex arc hyperbolic cosine of **z**, in the range of a half-strip of non-negative values along the real axis and in the interval [$-i\pi$, $i\pi$] along the imaginary axis.

7.x.4.2 The asinh macro

Synopsis

```
#include <complex.h>
complex-type asinh(complex-type z);
```

Description

The **asinh** macro computes the complex arc hyperbolic sine of **z**, with branch cuts outside the interval [$-i$, i] along the imaginary axis.

Returns

The **asinh** macro returns the complex arc hyperbolic sine of **z**, in the range of a strip mathematically unbounded along the real axis and in the interval [$-i\pi/2$, $i\pi/2$] along the imaginary axis.

7.x.4.3 The atanh macro

Synopsis

```
#include <complex.h>
imaginary-type atanh(imaginary-type z);
complex-type atanh(complex-type z);
```

Description

The **atanh** macro computes the complex arc hyperbolic tangent of **z**, with branch cuts outside the interval [-1, 1] along the real axis.

Returns

The **atanh** macro returns the complex arc hyperbolic tangent of **z**, in the range of a strip mathematically unbounded along the real axis and in the interval [$-i\pi/2$, $i\pi/2$] along the imaginary axis.

7.x.4.4 The cosh macro

Synopsis

```
#include <complex.h>
real-type cosh(imaginary-type z);
complex-type cosh(complex-type z);
```

Description

The **cosh** macro computes the complex hyperbolic cosine of **z**.

Returns

The `cosh` macro returns the complex hyperbolic cosine of `z`.

7.x.4.5 The sinh macro

Synopsis

```
#include <complex.h>
imaginary-type sinh(imaginary-type z);
complex-type sinh(complex-type z);
```

Description

The `sinh` macro computes the complex hyperbolic sine of `z`.

Returns

The `sinh` macro returns the complex hyperbolic sine of `z`.

7.x.4.6 The tanh macro

Synopsis

```
#include <complex.h>
imaginary-type tanh(imaginary-type z);
complex-type tanh(complex-type z);
```

Description

The `tanh` macro computes the complex hyperbolic tangent of `z`.

Returns

The `tanh` macro returns the complex hyperbolic tangent of `z`.

7.x.5 Exponential and logarithmic macros

7.x.5.1 The exp macro

Synopsis

```
#include <complex.h>
complex-type exp(complex-or-imaginary-type z);
```

Description

The `exp` macro computes the complex base-e exponential of `z`.

Returns

The `exp` macro returns the complex base-e exponential of `z`.

601

7.x.5.2 The log macro

Synopsis

```
#include <complex.h>
complex-type log(complex-or-imaginary-type z);
```

Description

The `log` macro computes the complex natural (base-e) logarithm of `z`, with a branch cut along the negative real axis.

Returns

The `log` macro returns the complex natural logarithm of `z`, in the range of a strip mathematically unbounded along the real axis and in the interval $[-i\pi, i\pi]$ along the imaginary axis.

7.x.5.3 The sqrt macro

Synopsis

```
#include <complex.h>
complex-type sqrt(complex-or-imaginary-type z);
```

Description

The `sqrt` macro computes the complex square root of `z`, with a branch cut along the negative real axis.

Returns

The `sqrt` macro returns the complex square root of `z`, in the range of the right half-plane.

7.x.6 Power and absolute value macros

7.x.6.1 The fabs macro

Synopsis

```
#include <complex.h>
real-type fabs(complex-or-imaginary-type z);
```

Description

The `fabs` macro computes the absolute value (also called norm, modulus, or magnitude) of `z`.

Returns

The `fabs` macro returns the absolute value of `z`.

602

7.x.6.2 The pow macro

Synopsis

```
#include <complex.h>
complex-type pow(floating-type x, floating-type y)
```

Description

The `pow` macro computes the complex power function x^y, with a branch cut for the first parameter along the negative real axis.

Returns

The `pow` macro returns the complex power function x^y.

7.x.7 Complex-specific macros

The header `<complex.h>` declares overloading macros pertaining specifically to complex arithmetic. They accept arguments of real, complex, or imaginary type.

7.x.7.1 The arg macro

Synopsis

```
#include <complex.h>
real-type arg(floating-type z);
```

Type determination follows the same pattern as for `fabs`.

Description

The `arg` macro computes the argument or phase angle of `z`, with a branch cut along the negative real axis.

Returns

The `arg` macro returns the argument or phase angle of `z`, in the range $[-\pi, \pi]$.

7.x.7.2 The conj macro

Synopsis

```
#include <complex.h>
floating-type conj(floating-type z);
```

Return types match parameter types.

Description

The `conj` macro computes the complex conjugate of `z`, by reversing the sign of its imaginary part, if any.

[Note that conj(3.0) yields 3.0, not 3.0 - 0.0i.]

Returns

The `conj` macro returns the complex conjugate of `z`.

7.x.7.3 The imag macro

Synopsis

```
#include <complex.h>
real-type imag(floating-type z);
```

Type determination follows the same pattern as for `fabs`.

Description

The `imag` macro computes the imaginary part of `z`.

Returns

The `imag` macro returns the imaginary part of `z`.

7.x.7.4 The proj function

Synopsis

```
#include <complex.h>
floating-type proj(floating-type z);
```

The return type is real if the argument is real; the return type is complex if the argument is complex or imaginary.

Description

The `proj` macro computes a projection of `z` onto the Riemann sphere: `z` projects to `z` except that all infinities, even ones with one infinite part and one NaN part, project to positive infinity on the real axis.

Returns

The `proj` macro returns a projection of `z` onto the Riemann sphere.

[Two topologies are commonly used in complex mathematics: the complex plane with its continuum of infinities and the Riemann sphere with its single infinity. The complex plane is better suited for transcendental functions, the Riemann sphere for algebraic functions. The complex types with their multiplicity of infinities provide a useful (though imperfect) model for the complex plane. The `proj` function helps model the Riemann sphere by mapping all infinities to one, and should be used just before any operation, especially comparisons, that might give spurious results for any of the other infinities.

Note that a complex value with one infinite part and one NaN part is regarded as an infinity, not a NaN, because if one part is infinite, the complex value is infinite independent of the value of the other part. For the same reason, `fabs` returns an infinity if its argument has an infinite part and a NaN part.]

7.x.7.5 The real macro

Synopsis

```
#include <complex.h>
real-type real(floating-type z);
```

Type determination follows the same pattern as for **imag** (7.x.7.3).

Description

The **real** macro computes the real part of **z**.

Returns

The **real** macro returns the real part of **z**.

605