C mathematical functions

C mathematical operations are a group of functions in the standard library of the C programming language implementing basic mathematical functions.[1][2] All functions use floating point numbers in one manner or another. Different C standards provide different, albeit backwards-compatible, sets of functions. Most of these functions are also available in the C++ standard library, though in different headers (the C headers are included as well, but only as a deprecated compatibility feature).

Contents

Overview of functions

Most of the mathematical functions are defined in `math.h` (`cmath` header in C++). The functions that operate on integers, such as `abs`, `labs`, `div`, and `ldiv`, are instead defined in the `stdlib.h` header (`cstdlib` header in C++).

Any functions that operate on angles use radians as the unit of angle.[1]

Not all of these functions are available in the C89 version of the standard. For those that are, the functions accept only type `double` for the floating-point arguments, leading to expensive type conversions in code that otherwise used single-precision `float` values. In C99, this shortcoming was fixed by introducing new sets of functions that work on `float` and `long double` arguments. Those functions are identified by `f` and `l` suffixes respectively.[3]

	Function	Description
Exponential functions	`abs` (http://en.cppreference.com/w/c/numeric/math/abs) `labs` (http://en.cppreference.com/w/c/numeric/math/labs) `llabs` (http://en.cppreference.com/w/c/numeric/math/llabs)	computes absolute value of an integer value
	`fabs` (http://en.cppreference.com/w/c/numeric/math/fabs)	computes absolute value of a floating point value
	`div` (http://en.cppreference.com/w/c/numeric/math/div) `ldiv` (http://en.cppreference.com/w/c/numeric/math/ldiv) `lldiv` (http://en.cppreference.com/w/c/numeric/math/lldiv)	computes the quotient and remainder of integer division
	`fmod` (http://en.cppreference.com/w/c/numeric/math/fmod)	remainder of the floating point division operation
	`remainder` (http://en.cppreference.com/w/c/numeric/math/remainder)	signed remainder of the division operation
	`remquo` (http://en.cppreference.com/w/c/numeric/math/remquo)	signed remainder as well as the three last bits of the division operation
	`fma` (http://en.cppreference.com/w/c/numeric/math/fma)	fused multiply-add operation
	`fmax` (http://en.cppreference.com/w/c/numeric/math/fmax)	larger of two floating point values
	`fmin` (http://en.cppreference.com/w/c/numeric/math/fmin)	smaller of two floating point values
	`fdim` (http://en.cppreference.com/w/c/numeric/math/fdim)	positive difference of two floating point values
	`nan` (http://en.cppreference.com/w/c/numeric/math/nan) `nanf` (http://en.cppreference.com/w/c/numeric/math/nanf) `nanl` (http://en.cppreference.com/w/c/numeric/math/nanl)	returns a not-a-number (NaN)
	`exp` (http://en.cppreference.com/w/c/numeric/math/exp)	returns e raised to the given power
	`exp2` (http://en.cppreference.com/w/c/numeric/math/exp2)	returns 2 raised to the given power
	`expm1` (http://en.cppreference.com/w/c/numeric/math/expm1)	returns e raised to the given power, minus one
	`log` (http://en.cppreference.com/w/c/numeric/math/log)	computes natural logarithm (to base e)
	`log2` (http://en.cppreference.com/w/c/numeric/math/log2)	computes binary logarithm (to base 2)
	`log10` (http://en.cppreference.com/w/c/numeric/math/log10)	computes common logarithm (to base 10)
	`log1p` (http://en.cppreference.com/w/c/numeric/math/log1p)	computes natural logarithm (to base e) of 1 plus the given number
	`ilogb` (http://en.cppreference.com/w/c/numeric/math/ilogb)	extracts exponent of the number
	`logb` (http://en.cppreference.com/w/c/numeric/math/logb)	extracts exponent of the number
Power functions	`sqrt` (http://en.cppreference.com/w/c/numeric/math/sqrt)	computes square root
	`cbrt` (http://en.cppreference.com/w/c/numeric/math/cbrt)	computes cubic root
	`hypot` (http://en.cppreference.com/w/c/numeric/math/hypot)	computes square root of the sum of the squares of two given numbers
	`pow` (http://en.cppreference.com/w/c/numeric/math/pow)	raises a number to the given power[4]
Trigonometric functions	`sin` (http://en.cppreference.com/w/c/numeric/math/sin)	computes sine
	`cos` (http://en.cppreference.com/w/c/numeric/math/cos)	computes cosine
	`tan` (http://en.cppreference.com/w/c/numeric/math/tan)	computes tangent
	`asin` (http://en.cppreference.com/w/c/numeric/math/asin)	computes arc sine
	`acos` (http://en.cppreference.com/w/c/numeric/math/acos)	computes arc cosine
	`atan` (http://en.cppreference.com/w/c/numeric/math/atan)	computes arc tangent
	`atan2` (http://en.cppreference.com/w/c/numeric/math/atan2)	computes arc tangent, using signs to determine quadrants
Hyperbolic functions	`sinh` (http://en.cppreference.com/w/c/numeric/math/sinh)	computes hyperbolic sine
	`cosh` (http://en.cppreference.com/w/c/numeric/math/cosh)	computes hyperbolic cosine
	`tanh` (http://en.cppreference.com/w/c/numeric/math/tanh)	computes hyperbolic tangent
	`asinh` (http://en.cppreference.com/w/c/numeric/math/asinh)	computes hyperbolic arc sine
	`acosh` (http://en.cppreference.com/w/c/numeric/math/acosh)	computes hyperbolic arc cosine
	`atanh` (http://en.cppreference.com/w/c/numeric/math/atanh)	computes hyperbolic arc tangent
Error and gamma functions	`erf` (http://en.cppreference.com/w/c/numeric/math/erf)	computes error function
	`erfc` (http://en.cppreference.com/w/c/numeric/math/erfc)	computes complementary error function
	`lgamma` (http://en.cppreference.com/w/c/numeric/math/lgamma)	computes natural logarithm of the absolute value of the gamma function
	`tgamma` (http://en.cppreference.com/w/c/numeric/math/tgamma)	computes gamma function
Nearest integer floating point operations	`ceil` (http://en.cppreference.com/w/c/numeric/math/ceil)	returns the nearest integer not less than the given value
	`floor` (http://en.cppreference.com/w/c/numeric/math/floor)	returns the nearest integer not greater than the given value
	`trunc` (http://en.cppreference.com/w/c/numeric/math/trunc)	returns the nearest integer not greater in magnitude than the given value
	`round` (http://en.cppreference.com/w/c/numeric/math/round) `lround` (http://en.cppreference.com/w/c/numeric/math/lround) `llround` (http://en.cppreference.com/w/c/numeric/math/llround)	returns the nearest integer, rounding away from zero in halfway cases
	`nearbyint` (http://en.cppreference.com/w/c/numeric/math/nearbyint)	returns the nearest integer using current rounding mode
	`rint` (http://en.cppreference.com/w/c/numeric/math/rint) `lrint` (http://en.cppreference.com/w/c/numeric/math/rint) `llrint` (http://en.cppreference.com/w/c/numeric/math/llrint)	returns the nearest integer using current rounding mode with exception if the result differs
Floating point manipulation functions	`frexp` (http://en.cppreference.com/w/c/numeric/math/frexp)	decomposes a number into significand and a power of 2
	`ldexp` (http://en.cppreference.com/w/c/numeric/math/ldexp)	multiplies a number by 2 raised to a power
	`modf` (http://en.cppreference.com/w/c/numeric/math/modf)	decomposes a number into integer and fractional parts
	`scalbn` (http://en.cppreference.com/w/c/numeric/math/scalbn) `scalbln` (http://en.cppreference.com/w/c/numeric/math/scalbln)	multiplies a number by FLT_RADIX raised to a power
	`nextafter` (http://en.cppreference.com/w/c/numeric/math/nextafter) `nexttoward` (http://en.cppreference.com/w/c/numeric/math/nexttoward)	returns next representable floating point value towards the given value
	`copysign` (http://en.cppreference.com/w/c/numeric/math/copysign)	copies the sign of a floating point value
Classification	`fpclassify` (http://en.cppreference.com/w/c/numeric/math/fpclassify)	categorizes the given floating point value
	`isfinite` (http://en.cppreference.com/w/c/numeric/math/isfinite)	checks if the given number has finite value
	`isinf` (http://en.cppreference.com/w/c/numeric/math/isinf)	checks if the given number is infinite
	`isnan` (http://en.cppreference.com/w/c/numeric/math/isnan)	checks if the given number is NaN
	`isnormal` (http://en.cppreference.com/w/c/numeric/math/isnormal)	checks if the given number is normal
	`signbit` (http://en.cppreference.com/w/c/numeric/math/signbit)	checks if the given number is negative

Floating point environment

C99 adds several functions and types for fine-grained control of floating point environment.[3] These functions can be used to control a variety of settings that affect floating-point computations, for example, the rounding mode, on what conditions exceptions occur, when numbers are flushed to zero, etc. The floating point environment functions and types are defined in `fenv.h` header (`cfenv` in C++).

Function	Description
`feclearexcept` (http://en.cppreference.com/w/c/numeric/fenv/feclearexcept)	clears exceptions (C99)
`fegetenv` (http://en.cppreference.com/w/c/numeric/fenv/feenv)	stores current floating-point environment (C99)
`fegetexceptflag` (http://en.cppreference.com/w/c/numeric/fenv/feexceptflag)	stores current status flags (C99)
`fegetround` (http://en.cppreference.com/w/c/numeric/fenv/feround)	retrieves current rounding direction (C99)
`feholdexcept` (http://en.cppreference.com/w/c/numeric/fenv/feholdexcept)	saves current floating-point environment and clears all exceptions (C99)
`feraiseexcept` (http://en.cppreference.com/w/c/numeric/fenv/feraiseexcept)	raises a floating-point exception (C99)
`fesetenv` (http://en.cppreference.com/w/c/numeric/fenv/feenv)	sets current floating-point environment (C99)
`fesetexceptflag` (http://en.cppreference.com/w/c/numeric/fenv/feexceptflag)	sets current status flags (C99)
`fesetround` (http://en.cppreference.com/w/c/numeric/fenv/feround)	sets current rounding direction (C99)
`fetestexcept` (http://en.cppreference.com/w/c/numeric/fenv/fetestexcept)	tests whether certain exceptions have been raised (C99)
`feupdateenv` (http://en.cppreference.com/w/c/numeric/fenv/feupdateenv)	restores floating-point environment, but keeps current exceptions (C99)

Complex numbers

C99 adds a new `_Complex` keyword (and `complex` convenience macro) that provides support for complex numbers. Any floating point type can be modified with `complex`, and is then defined as a pair of floating point numbers. Note that C99 and C++ do not implement complex numbers in a code-compatible way - the latter instead provides the class `std::complex`.

All operations on complex numbers are defined in `complex.h` header. As with the real-valued functions, an `f` or `l` suffix denotes the `float complex` or `long double complex` variant of the function.

	Function	Description
Basic operations	`cabs` (http://en.cppreference.com/w/c/numeric/complex/cabs)	computes absolute value (C99)
	`carg` (http://en.cppreference.com/w/c/numeric/complex/carg)	computes argument of a complex number (C99)
	`cimag` (http://en.cppreference.com/w/c/numeric/complex/cimag)	computes imaginary part of a complex number (C99)
	`creal` (http://en.cppreference.com/w/c/numeric/complex/creal)	computes real part of a complex number (C99)
	`conj` (http://www.opengroup.org/onlinepubs/9699919799/functions/conj.html)	computes complex conjugate (C99)
	`cproj` (http://en.cppreference.com/w/c/numeric/complex/cproj)	computes complex projection into the Riemann sphere (C99)
Exponentiation operations	`cexp` (http://en.cppreference.com/w/c/numeric/complex/cexp)	computes complex exponential (C99)
	`clog` (http://en.cppreference.com/w/c/numeric/complex/clog)	computes complex logarithm (C99)
	`csqrt` (http://en.cppreference.com/w/c/numeric/complex/csqrt)	computes complex square root (C99)
	`cpow` (http://en.cppreference.com/w/c/numeric/complex/cpow)	computes complex power (C99)
Trigonometric operations	`csin` (http://en.cppreference.com/w/c/numeric/complex/csin)	computes complex sine (C99)
	`ccos` (http://en.cppreference.com/w/c/numeric/complex/ccos)	computes complex cosine (C99)
	`ctan` (http://en.cppreference.com/w/c/numeric/complex/ctan)	computes complex tangent (C99)
	`casin` (http://en.cppreference.com/w/c/numeric/complex/casin)	computes complex arc sine (C99)
	`cacos` (http://en.cppreference.com/w/c/numeric/complex/cacos)	computes complex arc cosine (C99)
	`catan` (http://en.cppreference.com/w/c/numeric/complex/catan)	computes complex arc tangent (C99)
Hyperbolic operations	`csinh` (http://en.cppreference.com/w/c/numeric/complex/csinh)	computes complex hyperbolic sine (C99)
	`ccosh` (http://en.cppreference.com/w/c/numeric/complex/ccosh)	computes complex hyperbolic cosine (C99)
	`ctanh` (http://en.cppreference.com/w/c/numeric/complex/ctanh)	computes complex hyperbolic tangent (C99)
	`casinh` (http://en.cppreference.com/w/c/numeric/complex/casinh)	computes complex hyperbolic arc sine (C99)
	`cacosh` (http://en.cppreference.com/w/c/numeric/complex/cacosh)	computes complex hyperbolic arc cosine (C99)
	`catanh` (http://en.cppreference.com/w/c/numeric/complex/catanh)	computes complex hyperbolic arc tangent (C99)

A few more complex functions are "reserved for future use in C99".[5] Implementations are provided by open-source projects that are not part of the standard library.

	Function	Description
Error functions	`cerf` (http://apps.jcns.fz-juelich.de/libcerf)	computes the complex error function (C99)
	`cerfc` (http://apps.jcns.fz-juelich.de/libcerf)	computes the complex complementary error function (C99)

Type-generic functions

The header `tgmath.h` defines a type-generic macro for each mathematical function defined in `math.h` and `complex.h`. This adds a limited support for function overloading of the mathematical functions: the same function name can be used with different types of parameters; the actual function will be selected at compile-time according to the types of the parameters.

Each type-generic macro that corresponds to a function that is defined for both real and complex numbers encapsulates a total of 6 different functions: `float`, `double` and `long double`, and their `complex` variants. The type-generic macros that correspond to a function that is defined for only real numbers encapsulates a total of 3 different functions: `float`, `double` and `long double` variants of the function.

The C++ language includes native support for function overloading and thus does not provide the `tgmath.h` header even as a compatibility feature.

Random number generation

The header `stdlib.h` (`cstdlib` in C++) defines several functions that can be used for statistically random number generation.[6]

Function	Description
`rand` (http://en.cppreference.com/w/c/numeric/random/rand)	generates a pseudo-random number between 0 and `RAND_MAX`.
`srand` (http://en.cppreference.com/w/c/numeric/random/srand)	initializes a pseudo-random number generator
`arc4random`	generates a pseudo-random number between 0 and `UINT32_MAX`, usually using a better algorithm than `rand`
`arc4random_uniform`	generates a pseudo-random number between 0 and a maximum value.
`arc4random_buf`	fill a buffer with a pseudo-random bitstream.
`arc4random_stir`	initializes a pseudo-random number generator.

The `arc4random` family of random number functions are not defined in POSIX standard, but is found in some common `libc` implementations. It used to refer to the keystream generator of a leaked version of RC4 cipher (hence "alleged **RC4**"), but different algorithms, usually from other ciphers like ChaCha20, have been implemented since using the same name.

The quality of randomness from `rand` are usually too weak to be even considered statistically random, and it requires explicit seeding. It is usually advised to use `arc4random` instead of `rand` when possible. Some C libraries implement `rand` using `arc4random_uniform` internally.

libm

Under Linux and BSD, the mathematical functions (as declared in `math.h`) are bundled separately in the mathematical library `libm`. Therefore, if any of those functions are used, the linker must be given the directive `-lm`.

There are various `libm` implementations, including:

- GNU libc's libm (https://www.gnu.org/software/libc/manual/html_node/Mathematics.html)
- AMD's libm (http://developer.amd.com/amd-cpu-libraries/amd-math-library-libm/)
- Red Hat's libm (https://sourceware.org/newlib/libm.html)
- Sun's FDLIBM (http://www.netlib.org/fdlibm/readme), which was used as the basis for FreeBSD's msun (http://svnweb.freebsd.org/base/head/lib/msun/) and OpenBSD's libm (http://cvsweb.openbsd.org/cgi-bin/cvsweb/src/lib/libm/), both of which in turn were the basis of Julia's OpenLibm (http://openlibm.org/)
- Arénaire project's CRlibm (https://web.archive.org/web/20161027224938/http://lipforge.ens-lyon.fr/www/crlibm) (correctly rounded libm), and its successor MetaLibm (http://metalibm.org/)

See also

- C99 floating point support

References

1. *ISO/IEC 9899:1999 specification* (http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf) (PDF), p. 212, § 7.12.
2. Prata, Stephen (2004). *C primer plus*. Sams Publishing. Appendix B. Section V. The Standard ANSI C Library with C99 Additions. ISBN 0-672-32696-5.
3. Prata, Stephen (2004). *C primer plus*. Sams Publishing. Appendix B. Section VIII. C99 Numeric Computational Enhancements. ISBN 0-672-32696-5.
4. Notationally, it may seem convenient to use pow(x,2) or pow(x,3) to compute squares or cubes. However, this is not advisable in time-critical code. Unless an implementation takes special care of these cases at compile time, x*x or x*x*x will execute much faster. Also, sqrt(x) and cbrt(x) should be preferred over pow(x,.5) or pow(x,1./3).
5. man cerf(3), man cerfc(3), see e.g. http://linux.die.net/man/3/cerf
6. "GNU C Library -- Mathematics" (https://web.archive.org/web/20110820024510/http://www.cs.utah.edu/dept/old/texinfo/glibc-manual-0.02/library_17.html). Archived from the original (https://www.cs.utah.edu/dept/old/texinfo/glibc-manual-0.02/library_17.html) on 20 August 2011. Retrieved 2 November 2011.

External links

- `math.h` (http://www.opengroup.org/onlinepubs/9699919799/basedefs/math.h.html): mathematical declarations – Base Definitions Reference, The Single UNIX Specification, Issue 7 from The Open Group
- C reference for math functions (http://en.cppreference.com/w/c/numeric/math)