Fondamenti di Informatica II e Lab.

La libreria <math.h>

Pagine estratte dal sito www.cplusplus.com

Ultimo aggiornamento 02/11/2016

constant

NAN

```
float
```

Not-A-Number

Macro constant that expands to an expression of type `float` that represents a NaN if the implementation supports quiet NaNs (otherwise, it is not defined).

constant

INFINITY

Infinity

Macro constant that expands to an expression of type `float`.

If the implementation supports *infinity* values, this is defined as the value that represents a positive or unsigned infinity. Otherwise, it is a positive constant that overflows at translation time.

This may be returned by a function that signals a *range error* by setting `errno` to `ERANGE`.

function

cos

```
     double cos  (double x);
      float cosf (float x);
long double cosl (long double x);
```

Compute cosine
Returns the cosine of an angle of *x* radians.

Parameters

x

Value representing an angle expressed in radians.
One *radian* is equivalent to `180/PI` *degrees*.

Return Value

Cosine of *x* radians.

function

sin

```
     double sin  (double x);
      float sinf (float x);
long double sinl (long double x);
```

Compute sine
Returns the sine of an angle of *x* radians.

Parameters

x

Value representing an angle expressed in radians.
One *radian* is equivalent to `180/PI` *degrees*.

Return Value

Sine of *x* radians.

function

tan

```
     double tan  (double x);
      float tanf (float x);
long double tanl (long double x);
```

Compute tangent
Returns the tangent of an angle of *x* radians.

Parameters

x

Value representing an angle, expressed in radians.
One *radian* is equivalent to `180/PI` *degrees*.

Return Value

Tangent of *x* radians.

function

acos

```
     double acos  (double x);
      float acosf (float x);
long double acosl (long double x);
```

Compute arc cosine
Returns the principal value of the arc cosine of *x*, expressed in radians.

In trigonometrics, *arc cosine* is the inverse operation of *cosine*.

Parameters

x

Value whose arc cosine is computed, in the interval `[-1,+1]`.
If the argument is out of this interval, a *domain error* occurs.

Return Value

Principal arc cosine of *x*, in the interval [0,pi] radians.
One *radian* is equivalent to `180/PI` *degrees*.

function

asin

```
     double asin  (double x);
      float asinf (float x);
long double asinl (long double x);
```

Compute arc sine

Returns the principal value of the arc sine of *x*, expressed in radians.

In trigonometrics, *arc sine* is the inverse operation of *sine*.

Parameters

x

Value whose arc sine is computed, in the interval `[-1,+1]`.
If the argument is out of this interval, a *domain error* occurs.

Return Value

Principal arc sine of *x*, in the interval [-pi/2,+pi/2] radians.
One *radian* is equivalent to `180/PI` *degrees*.

function

atan

```
     double atan  (double x);
      float atanf (float x);
long double atanl (long double x);
```

Compute arc tangent

Returns the principal value of the arc tangent of *x*, expressed in radians.

In trigonometrics, *arc tangent* is the inverse operation of *tangent*.

Notice that because of the sign ambiguity, the function cannot determine with certainty in which quadrant the angle falls only by its tangent value. See `atan2` for an alternative that takes a fractional argument instead.

Parameters

x

Value whose arc tangent is computed.

Return Value

Principal arc tangent of *x*, in the interval [-pi/2,+pi/2] radians.
One *radian* is equivalent to `180/PI` *degrees*.

function

exp

```
     double exp  (double x);
      float expf (float x);
long double expl (long double x);
```

Compute exponential function

Returns the *base-e* exponential function of *x*, which is *e* raised to the power *x*: e^x.

Parameters

x

Value of the exponent.

Return Value

Exponential value of *x*.
If the magnitude of the result is too large to be represented by a value of the return type, the function returns HUGE_VAL(or HUGE_VALF or HUGE_VALL) with the proper sign, and an overflow *range error* occurs.

function

log

```
     double log  (     double x);
      float logf (      float x);
long double logl (long double x);
```

Compute natural logarithm

Returns the *natural logarithm* of *x*.

The *natural logarithm* is the base-e logarithm: the inverse of the natural exponential function (exp). For common (base-10) logarithms, see log10.

Parameters

x

Value whose logarithm is calculated.
If the argument is negative, a *domain error* occurs.

Return Value

Natural logarithm of *x*.
If *x* is negative, it causes a *domain error*.
If *x* is zero, it may cause a *pole error* (depending on the library implementation).

function

log10

```
     double log10  (double x);
      float log10f (float x);
long double log10l (long double x);
```

Compute common logarithm

Returns the *common* (base-10) *logarithm* of *x*.

Parameters

x

Value whose logarithm is calculated.
If the argument is negative, a *domain error* occurs.

Return Value

Common logarithm of *x*.
If *x* is negative, it causes a *domain error*.
If *x* is zero, it may cause a *pole error* (depending on the library implementation).

function

log2

```
     double log2  (double x);
      float log2f (float x);
long double log2l (long double x);
```

Compute binary logarithm

Returns the *binary* (base-2) *logarithm* of *x*.

Parameters

x

Value whose logarithm is calculated.
If the argument is negative, a *domain error* occurs.

Return Value

The *binary logarithm* of *x*: $\log_2 x$.
If *x* is negative, it causes a *domain error*:
If *x* is zero, it may cause a *pole error* (depending on the library implementation).

function

pow

```
     double pow  (double base     , double exponent);
      float powf (float base      , float exponent);
long double powl (long double base, long double exponent);
```

Raise to power
Returns *base* raised to the power *exponent*:

$base^{exponent}$

Parameters

`base`
: Base value.

`exponent`
: Exponent value.

Return Value

The result of raising *base* to the power *exponent*.

If the *base* is finite negative and the *exponent* is finite but not an integer value, it causes a *domain error*.
If both *base* and *exponent* are zero, it may also cause a *domain error* on certain implementations.
If *base* is zero and *exponent* is negative, it may cause a *domain error* or a *pole error* (or none, depending on the library implementation).
The function may also cause a *range error* if the result is too great or too small to be represented by a value of the return type.

function

sqrt

```
     double sqrt  (double x);
      float sqrtf (float x);
long double sqrtl (long double x);
```

Compute square root
Returns the *square root* of *x*.

Parameters

x
: Value whose square root is computed.
If the argument is negative, a *domain error* occurs.

Return Value

Square root of *x*.
If *x* is negative, a *domain error* occurs:

function

ceil

```
     double ceil  (double x);
      float ceilf (float x);
long double ceill (long double x);
```

Round up value

Rounds *x* upward, returning the smallest integral value that is not less than *x*.

Parameters

x

Value to round up.

Return Value

The smallest integral value that is not less than *x* (as a floating-point value).

function

floor

```
     double floor  (double x);
      float floorf (float x);
long double floorl (long double x);
```

Round down value

Rounds *x* downward, returning the largest integral value that is not greater than *x*.

Parameters

x

Value to round down.

Return Value

The value of *x* rounded downward (as a floating-point value).

function

round

```
     double round  (double x);
      float roundf (float x);
long double roundl (long double x);
```

Round to nearest

Returns the integral value that is nearest to *x*, with halfway cases rounded away from zero.

Parameters

x

Value to round.

Return Value

The value of *x* rounded to the nearest integral (as a floating-point value).

function

fabs

```
     double fabs  (double x);
      float fabsf (float x);
long double fabsl (long double x);
```

Compute absolute value

Returns the *absolute value* of *x*: |*x*|.

Parameters

x

Value whose absolute value is returned.

Return Value

The absolute value of *x*.

macro

isfinite

```
   macro isfinite(x)
```

Is finite value

Returns whether *x* is a *finite value*.

A *finite value* is any floating-point value that is neither *infinite* nor *NaN* (*Not-A-Number*).

In C, this is implemented as a macro that returns an `int` value. The type of *x* shall be `float`, `double` or `long double`.

Parameters

x

A floating-point value.

Return value

A non-zero value (`true`) if *x* is finite; and zero (`false`) otherwise.

macro/function

isinf

```
macro isinf(x)
```

Is infinity

Returns whether *x* is an *infinity value* (either *positive infinity* or *negative infinity*).

In C, this is implemented as a macro that returns an `int` value. The type of *x* shall be `float`, `double` or `long double`.

Parameters

x

A floating-point value.

Return value

A non-zero value (`true`) if *x* is an infinity; and zero (`false`) otherwise.

macro/function

isnan

```
macro isnan(x)
```

Is Not-A-Number

Returns whether *x* is a *NaN* (*Not-A-Number*) value.

The *NaN* values are used to identify undefined or non-representable values for floating-point elements, such as the square root of negative numbers or the result of 0/0.

In C, this is implemented as a macro that returns an `int` value. The type of *x* shall be `float`, `double` or `long double`.

Parameters

x

A floating-point value.

Return value

A non-zero value (`true`) if *x* is a *NaN* value; and zero (`false`) otherwise.