Полный справочник по

Глава 15

Математические функции

В версии C99 математическая библиотека была значительно пополнена; при этом число ее функций увеличилось более чем в три раза (стандарт C89 определял всего лишь 22 математические функции). Одной из основных целей комитета по версии C99 было повышение применимости языка C для численных расчетов. Теперь с уверенностью можно сказать, что эта цель достигнута!

Для использования математических функций в программу необходимо включить заголовок <math.h>. Помимо объявления математических функций, этот заголовок определяет один или несколько макросов. В версии C89 заголовком <math.h> определяется только макрос HUGE_VAL, который представляет собой значение типа double, сигнализирующее о возникшем переполнении. В версии C99 кроме него определены следующие макросы.

HUGE_VALF	версия макроса HUGE_VAL с типом float
HUGE_VALL	версия макроса HUGE_VAL с типом long double
INFINITY	Значение, представляющее бесконечность
math_errhandling	Содержит макросы MATH_ERRNO и/или MATH_ERREXCEPT
MATH_ERRNO	Встроенная глобальная переменная errno, используемая для вывода сообщений об ошибках
MATH_ERREXCEPT	Исключение, возбуждаемое при выполнении операций над вещественными числами, с целью вывода сообщения об ошибках
NAN	Не число

В версии C99 определены следующие макросы (подобные функциям), классифицирующие значение.

int fpclassify(*fpval*)	В зависимости от значения аргумента *fpval* возвращает FP_INFINITY, FP_NAN, FP_NORMAL, FP_SUBNORMAL или FP_ZERO. Эти макросы определяются заголовком <math.h>
int isfinite(*fpval*)	Возвращает ненулевое значение, если *fpval* конечное
int isinf(*fpval*)	Возвращает ненулевое значение, если *fpval* представляет бесконечность
int isnan(*fpval*)	Возвращает ненулевое значение, если *fpval* — не является числом
int isnormal(*fpval*)	Возвращает ненулевое значение, если *fpval* представляет собой нормализованное число
int signbit(*fpval*)	Возвращает ненулевое значение, если *fpval* отрицательно (т.е. установлен его знаковый разряд)

В версии C99 определены следующие макросы сравнения, аргументы которых (*a* и *b*) должны иметь числовые значения в формате с плавающей точкой.

int isgreater(*a, b*)	Возвращает ненулевое значение, если *a* больше *b*
int isgreaterequal(*a, b*)	Возвращает ненулевое значение, если *a* больше или равно *b*
int isless(*a, b*)	Возвращает ненулевое значение, если *a* меньше *b*
int islessequal(*a, b*)	Возвращает ненулевое значение, если *a* меньше или равно *b*
int islessgreater(*a, b*)	Возвращает ненулевое значение, если *a* больше или меньше *b*
int isunordered(*a, b*)	Возвращает 1, если *a* и *b* не упорядочены одно по отношению к другому Возвращает 0, если *a* и *b* упорядочены

Эти макросы введены, так как они прекрасно обрабатывают значения, которые не являются числами, не вызывая при этом исключений вещественного типа.

Макросы EDOM и ERANGE также используются математическими функциями. Эти макросы определены в заголовке <errno.h>.

Ошибки в версиях C89 и C99 обрабатываются по-разному. Так, в версии C89, если аргумент математической функции не попадает в область определения, возвращается

некоторое значение, зависящее от конкретной реализации, а встроенная глобальная целая переменная errno устанавливается равной значению EDOM. В версии C99 нарушение области определения также приводит к возврату значения, зависящего от конкретной реализации. Однако по значению math_errhandling можно судить о выполнении других действий. Если math_errhandling содержит значение MATH_ERRNO, то встроенная глобальная целая переменная errno устанавливается равной значению EDOM. Если же math_errhandling содержит значение MATH_ERREXCEPT, возбуждается исключение вещественного типа.

В версии C89, если функция генерирует результат, который слишком велик и потому не может быть представлен в машинном формате, то происходит переполнение. В этом случае функция возвращает значение HUGE_VAL, а переменная errno устанавливается равной значению ERANGE, сигнализирующему о выходе за пределы диапазона. При потере значимости функция возвращает нуль и устанавливает переменную errno равной значению ERANGE. В версии C99 ошибка переполнения также приводит к тому, что функция возвращает значение HUGE_VAL, а при потере значимости — нуль. Если math_errhandling содержит значение MATH_ERRNO, глобальная переменная errno устанавливается равной значению ERANGE, свидетельствующему об ошибке диапазона. Если же math_errhandling содержит значение MATH_ERREXCEPT, возбуждается исключение вещественного типа.

В версии C89 аргументами математических функций должны быть значения типа double, причем значения, возвращаемые функциями, тоже имеют тип double. В версии C99 добавлены варианты этих функций, работающие с типами float и long double. В этих функциях используются суффиксы f и l соответственно. Например, в версии C89 функция sin() определена следующим образом.

```
double sin(double arg);
```

Версия C99 поддерживает приведенное выше определение функции sin(), но в ней добавлены еще две ее модификации — sinf() и sinl().

```
float sinf(float arg);
long double sinl(long double arg);
```

Операции, выполняемые всеми тремя функциями, одинаковы; различаются лишь типы данных, над которыми выполняются эти операции. Добавление модификаций f и l математических функций позволяет использовать версию, которая наиболее точно соответствует данным, с которыми работают функции.

Поскольку в версии C99 добавлено так много новых функций, стоит отдельно перечислить те из них, которые поддерживаются версией C89. Это следующие функции.

acos	cos	fmod	modf	tan
asin	cosh	frexp	pow	tanh
atan	exp	ldexp	sin	
atan2	fabs	log	sinh	
ceil	floor	log10	sqrt	

И последнее замечание: все углы задаются в радианах.

Семейство функций acos

```
#include <math.h>
float acosf(float arg);
double acos(double arg);
long double acosl(long double arg);
```

Функции `acosf()` и `acosl()` добавлены в версии C99.

Каждая функция семейства `acos()` возвращает значение арккосинуса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данная программа выводит значения арккосинусов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. программа составляет таблицу арккосинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арккосинус %f равен %f.\n", val, acos(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `atan()`, `atan2()`, `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций acosh

```
#include <math.h>
float acoshf(float arg);
double acosh(double arg);
long double acoshl(long double arg);
```

Функции `acosh()`, `acoshf()` и `acoshl()` добавлены в версии C99.

Каждая функция семейства `acosh()` возвращает значение гиперболического арккосинуса от аргумента *arg*. Значение аргумента должно быть больше или равно нулю; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Зависимые функции

`asinh()`, `atanh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций asin

```
#include <math.h>
float asinf(float arg);
double asin(double arg);
long double asinl(long double arg);
```

Функции `asinf()` и `asinl()` добавлены в версии C99.

Каждая функция семейства `asin()` возвращает значение арксинуса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данная программа выводит значения арксинусов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арксинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арксинус %f равен %f.\n", val, asin(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`acos()`, `atan()`, `atan2()`, `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций asinh

```
#include <math.h>
float asinhf(float arg);
double asinh(double arg);
long double asinhl(long double arg);
```

Функции `asinh()`, `asinhf()` и `asinhl()` добавлены в версии C99.

Каждая функция семейства `asinh()` возвращает значение гиперболического арксинуса от аргумента *arg*.

Зависимые функции

`acosh()`, `atanh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atan

```
#include <math.h>
float atanf(float arg);
double atan(double arg);
long double atanl(long double arg);
```

Функции `atanf()` и `atanl()` добавлены в версии C99.

Каждая функция семейства `atan()` возвращает значение арктангенса от аргумента *arg*.

Пример

Данная программа выводит значения арктангенсов последовательности аргументов, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арктангенса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арктангенс %f равен %f.\n", val, atan(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `tan()`, `cos()`, `sin()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atanh

```
#include <math.h>
float atanhf(float arg);
double atanh(double arg);
long double atanhl(long double arg);
```

Функции `atanh()`, `atanhf()` и `atanhl()` добавлены в версии C99.

Каждая функция семейства `atanh()` возвращает значение гиперболического арктангенса от аргумента *arg*. Значение аргумента должно находиться в диапазоне от -1 до 1 (не включая границы); в противном случае возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если *arg* равен 1 или -1, возможен выход за пределы допустимого диапазона.

Зависимые функции

`acosh()`, `asinh()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций atan2

```
#include <math.h>
float atan2f(float a, float b);
double atan2(double a, double b);
long double atan2l(long double a, long double b);
```

Функции `atan2f()` и `atan2l()` добавлены в версии C99.

Каждая функция семейства `atan2()` возвращает значение арктангенса отношения *a/b*. Для вычисления квадранта возвращаемого значения используются знаки аргументов функции.

Пример

Данная программа выводит значения арктангенсов последовательности аргументов *y*, лежащих в интервале от -1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу арктангенса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Арктангенс %f равен %f.\n", val, atan2(val, 1.0));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan()`, `tan()`, `cos()`, `sin()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций cbrt

```
#include <math.h>
float cbrtf(float num);
double cbrt(double num);
long double cbrtl(long double num);
```

Функции `cbrt()`, `cbrtf()` и `cbrtl()` добавлены в версии C99.

Каждая функция семейства `cbrt()` возвращает значение кубического корня от аргумента *num*.

Пример

Данный фрагмент программы выводит на экран число `2`.

```
printf("%f", cbrt(8));
```

Зависимые функции

`sqrt()`

Семейство функций ceil

```
#include <math.h>
float ceilf(float num);
```

```
double ceil(double num);
long double ceill(long double num);
```

Функции `ceilf()` и `ceill()` добавлены в версии C99.

Каждая функция семейства `ceil()` возвращает наименьшее целое (представленное в виде значения с плавающей точкой), которое больше значения аргумента *num* или равно ему. Например, если *num* равно 1.02, функция `ceil()` вернет значение 2.0, а при *num*, равном -1.02, — значение -1.

Пример

Данный фрагмент программы выводит на экран число `10`.

```
printf("%f", ceil(9.9));
```

Зависимые функции

`floor()` и `fmod()`

Семейство функций copysign

```
#include <math.h>
float copysignf(float val, float signval);
double copysign(double val, double signval);
long double copysignl(long double val, long double signval);
```

Функции `copysign()`, `copysignf()` и `copysignl()` добавлены в версии C99.

Каждая функция семейства `copysign()` наделяет аргумент *val* знаком, который имеет аргумент *signval*, и возвращает полученный результат. Таким образом, возвращаемое значение имеет величину, равную величине аргумента *val*, а его знак совпадает со знаком аргумента *signval*.

Зависимые функции

`fabs()`

Семейство функций cos

```
#include <math.h>
float cosf(float arg);
double cos(double arg);
long double cosl(long double arg);
```

Функции `cosf()` и `cosl()` добавлены в версии C99.

Каждая функция семейства `cos()` возвращает значение косинуса аргумента *arg*. Значение аргумента должно быть выражено в радианах.

Пример

Данная программа выводит значения косинусов последовательности аргументов, лежащих в интервале от −1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу косинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Косинус %f равен %f.\n", val, cos(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

asin(), acos(), atan2(), atan(), tan(), sin(), sinh(), cos() и tanh()

Семейство функций cosh

```
#include <math.h>
float coshf(float arg);
double cosh(double arg);
long double coshl(long double arg);
```

Функции coshf() и coshl() добавлены в версии C99.

Каждая функция семейства cosh() возвращает значение гиперболического косинуса аргумента *arg*.

Пример

Данная программа выводит значения гиперболических косинусов последовательности аргументов, лежащих в интервале от −1 до 1 и увеличивающихся с шагом одна десятая, т.е. составляет таблицу гиперболического косинуса.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический косинус %f равен %f.\n", val, cosh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

asin(), acos(), atan2(), atan(), tan(), sin() и tanh()

Семейство функций erf

```
#include <math.h>
float erff(float arg);
double erf(double arg);
long double erfl(long double arg);
```

Функции `erf()`, `erff()` и `erfl()` добавлены в версии C99.

Каждая функция семейства `erf()` возвращает значение функции ошибок[1] от аргумента *arg*.

Зависимые функции

`erfc()`

Семейство функций erfc

```
#include <math.h>
float erfcf(float arg);
double erfc(double arg);
long double erfcl(long double arg);
```

Функции `erfc()`, `erfcf()` и `erfcl()` добавлены в версии C99.

Каждая функция семейства `erfc()` возвращает функцию ошибок дополнительную[2] от аргумента *arg*.

Зависимые функции

`erf()`

Семейство функций exp

```
#include <math.h>
float expf(float arg);
double exp(double arg);
long double expl(long double arg);
```

Функции `expf()` и `expl()` добавлены в версии C99.

---

[1] Интеграл (вероятности) ошибок:

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt\ .$$

Иногда называется просто *интегралом ошибок* или *интегралом вероятности*. В теории вероятности чаще используется не интеграл вероятности, а *интеграл вероятности Гаусса*, или *функция нормального распределения*

$$\Phi(x)=\frac{1}{2}[1+erf(\frac{x}{\sqrt{2}})]\ .$$ — *Прим. ред.*

[2] *Дополнительный интеграл вероятности*:

$$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt=1\text{-}\mathrm{erf}(x).$$ — *Прим. ред.*

Каждая функция семейства `exp()` возвращает значение экспоненты от аргумента *arg* (число *e*, возведенное в степень, которая равна значению аргумента *arg*).

Пример

Данный фрагмент программы выводит число e, округленное до значения 2.718282.

```
printf("e, возведенное в первую степень, приблизительно равно: %f.",
exp(1.0));
```

Зависимые функции

`exp2()` и `log()`

Семейство функций exp2

```
#include <math.h>
float exp2f(float arg);
double exp2(double arg);
long double exp2l(long double arg);
```

Функции `exp2()`, `exp2f()` и `exp2l()` добавлены в версии C99.

Каждая функция семейства `exp2()` возвращает число 2, возведенное в степень *arg*.

Зависимые функции

`exp()` и `log()`

Семейство функций expm1

```
#include <math.h>
float expm1f(float arg);
double expm1(double arg);
long double expm1l(long double arg);
```

Функции `expm1()`, `expm1f()` и `expm1l()` добавлены в версии C99.

Каждая функция семейства `expm1()` возвращает уменьшенное на единицу значение числа *e*, возведенного в степень *arg* (т.е. возвращаемое значение равно $e^{arg}-1$).

Зависимые функции

`exp()` и `log()`

Семейство функций fabs

```
#include <math.h>
float fabsf(float num);
double fabs(double num);
long double fabsl(long double num);
```

Функции `fabsf()` и `fabsl()` добавлены в версии C99.

Каждая функция семейства `fabs()` возвращает абсолютное значение аргумента *num*.

Пример

Данная программа дважды выводит на экран число `1.0`.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%1.1f %1.1f", fabs(1.0), fabs(-1.0));

  return 0;
}
```

Зависимые функции

`abs()`

Семейство функций fdim

```
#include <math.h>
float fdimf(float a, float b);
double fdim(double a, double b);
long double fdiml(long double a, long double b);
```

Функции `fdim()`, `fdimf()` и `fdiml()` добавлены в версии C99.

Каждая функция семейства `fdim()` возвращает нуль, если значение аргумента *a* меньше значения аргумента *b* или равно ему. В противном случае возвращается результат вычисления разности *a-b*.

Зависимые функции

`remainder()` и `remquo()`

Семейство функций floor

```
#include <math.h>
float floorf(float num);
double floor(double num);
long double floorl(long double num);
```

Функции `floorf()` и `floorl()` добавлены в версии C99.

Каждая функция семейства `floor()` возвращает наибольшее целое (представленное в виде значения с плавающей точкой), которое меньше значения аргумента *num* или равно ему. Например, при *num*, равном 1.02, функция `floor()` вернет значение 1.0, а при *num*, равном −1.02, — значение −2.0.

Пример

Данный фрагмент программы выводит на экран число `10`.

```
printf("%f", floor(10.9));
```

Зависимые функции

`ceil()` и `fmod()`

Семейство функций fma

```
#include <math.h>
float fmaf(float a, float b, float c);
double fma(double a, double b, double c);
long double fmal(long double a, long double b, long double c);
```

Функции `fma()`, `fmaf()` и `fmal()` определены в версии C99.

Каждая функция семейства `fma()` возвращает значение выражения *a*b+c*. Округление выполняется только один раз, после завершения всей операции.

Зависимые функции

`round()`, `lround()` и `llround()`

Семейство функций fmax

```
#include <math.h>
float fmaxf(float a, float b);
double fmax(double a, double b);
long double fmaxl(long double a, long double b);
```

Функции `fmax()`, `fmaxf()` и `fmaxl()` определены в версии C99.

Каждая функция семейства `fmax()` возвращает больший из аргументов *a* и *b*.

Зависимые функции

`fmin()`

Семейство функций fmin

```
#include <math.h>
float fminf(float a, float b);
double fmin(double a, double b);
long double fminl(long double a, long double b);
```

Функции `fmin()`, `fminf()` и `fminl()` определены в версии C99.

Каждая функция семейства `fmin()` возвращает меньший из аргументов *a* и *b*.

Зависимые функции

`fmax()`

Семейство функций fmod

```
#include <math.h>
float fmodf(float a, float b);
double fmod(double a, double b);
long double fmodl(long double a, long double b);
```

Функции `fmodf()` и `fmodl()` определены в версии C99.

Каждая функция семейства `fmod()` возвращает остаток от деления аргументов *a/b*.

Пример

Данная программа выводит на экран число `1.0`, являющееся остатком деления 10/3.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%1.1f", fmod(10.0,3.0));

  return 0;
}
```

Зависимые функции

`ceil()`, `floor()` и `fabs()`

Семейство функций frexp

```
#include <math.h>
float frexpf(float num, int *exp);
double frexp(double num, int *exp);
long double frexpl(long double num, int *exp);
```

Функции `frexpf()` и `frexpl()` добавлены в версии C99.

Каждая функция семейства `frexp()` разбивает число *num* на мантиссу *mantissa*, значение которой удовлетворяет неравенствам $0.5 \le mantissa < 1$, и целый показатель степени числа 2 (он обозначен через *exp*), притом числа *mantissa* и *exp* выбираются так, чтобы выполнялось равенство $num = mantissa * 2^{exp}$. Значение мантиссы возвращается функцией, а значение показателя[1] присваивается переменной, адресуемой указателем *exp*.

[1] Напомним, что представление числа *num* в виде $num = mantissa * b^{exp}$ (здесь *b* — основание системы счисления) называется *представлением с плавающей точкой* (*запятой*) или *полулогарифмическим представлением*, и что целая часть логарифма называется характеристикой. Так что $exp = \chi_2(num)+1$, где $\chi_2(num) = \lfloor \log_2(num) \rfloor$ — характеристика двоичного логарифма. Число *exp* часто называется *порядком* числа *num* (в нормализованном представлении). Заметим также, что терминам *мантисса* и *характеристика* часто придается и иной смысл. Так, по историческим причинам под *мантиссой* часто подразумевают *дробную часть логарифма*; иногда ее называют также *мантиссой логарифма*. Что же касается *характеристики*, то под ней иногда понимают просто *число, которое представляет порядок в представлении с плавающей запятой*. (В этом смысле в большинстве машин характеристика равна порядку, если он положительный; отличия между ними, как правило, обусловлены тем, что представление порядка, который может быть также и неположительным числом, при реализации операций над числами в полулогарифмическом представлении рассматривают как представление неотрицательного числа.) Так что можно сказать, что характеристика в этом смысле — машинное представление порядка числа. *Порядок* в этом контексте называется также иногда *экспонентой*. (Не путайте с экспонентой-функцией!) — *Прим. ред.*

Пример

Данный фрагмент программы выводит число `0.625` в качестве мантиссы и число `4` — в качестве показателя степени.

```
int e;
double f;

f = frexp(10.0, &e);
printf("%f %d", f, e);
```

Зависимые функции

`ldexp()`

Семейство функций hypot

```
#include <math.h>
float hypotf(float side1, float side2);
double hypot(double side1, double side2);
long double hypotl(long double side1, long double side2);
```

Функции `hypot()`, `hypotf()` и `hypotl()` определены в версии C99.

Каждая функция семейства `hypot()` возвращает длину гипотенузы при заданных длинах двух катетов (т.е. функция возвращает значение квадратного корня из суммы квадратов значений аргументов *side1* и *side2*)[1].

Зависимые функции

`sqrt()`

Семейство функций ilogb

```
#include <math.h>
int ilogbf(float num);
int ilogb(double num);
int ilogbl(long double num);
```

Функции `ilogb()`, `ilogbf()` и `ilogbl()` добавлены в версии C99.

Каждая функция семейства `ilogb()` возвращает порядок аргумента *num*. Возвращаемое значение имеет тип `int`.

Зависимые функции

`logb()`

Семейство функций ldexp

```
#include <math.h>
float ldexpf(float num, int exp);
```

[1] Или расстояние точки с координатами (*side1; side2*) от начала координат. — *Прим. ред.*

```
double ldexp(double num, int exp);
long double ldexpl(long double num, int exp);
```

Функции `ldexpf()` и `ldexpl()` добавлены в версии C99.

Каждая функция семейства `ldexp()` возвращает значение выражения *num* * 2^{exp}.

Пример

Данная программа выводит число `4`.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  printf("%f", ldexp(1, 2));

  return 0;
}
```

Зависимые функции

`frexp()` и `modf()`

Семейство функций lgamma

```
#include <math.h>
float lgammaf(float arg);
double lgamma(double arg);
long double lgammal(long double arg);
```

Функции `lgamma()`, `lgammaf()` и `lgammal()` добавлены в версии C99.

Каждая функция семейства `lgamma()` вычисляет абсолютное значение *гамма-функции*[1] от аргумента *arg* и возвращает ее натуральный логарифм.

Зависимые функции

`tgamma()`

Семейство функций llrint

```
#include <math.h>
long long int llrintf(float arg);
long long int llrint(double arg);
long long int llrintl(long double arg);
```

Функции `llrint()`, `llrintf()` и `llrintl()` добавлены в версии C99.

Каждая функция семейства `llrint()` возвращает значение аргумента *arg*, округленного до ближайшего целого, которое имеет тип `long long int`.

Зависимые функции

`lrint()` и `rint()`

[1] Другие названия: *Г-функция*, *Г-функция Эйлера*, *эйлеров интеграл второго рода*. — *Прим. ред.*

Семейство функций llround

```
#include <math.h>
long long int llroundf(float arg);
long long int llround(double arg);
long long int llroundl(long double arg);
```

Функции `llround()`, `llroundf()` и `llroundl()` добавлены в версии C99.

Каждая функция семейства `llround()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long long int`. Значения, отстоящие от большего и меньшего целых на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`lround()` и `round()`

Семейство функций log

```
#include <math.h>
float logf(float num);
double log(double num);
long double logl(long double num);
```

Функции `logf()` и `logl()` добавлены в версии C99.

Каждая функция семейства `log()` возвращает значение натурального логарифма от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Следующая программа выводит на экран значения натуральных логарифмов чисел от 1 до 10 (с шагом 1), т.е. составляет таблицу натуральных логарифмов целых чисел от 1 до 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = 1.0;

  do {
    printf("%f %f\n", val, log(val));
    val++;
  } while (val<11.0);

  return 0;
}
```

Зависимые функции

`log10()` и `log2()`

Семейство функций log1p

```
#include <math.h>
float log1pf(float num);
double log1p(double num);
long double log1pl(long double num);
```

Функции `log1p()`, `log1pf()` и `log1pl()` добавлены в версии C99.

Каждая функция семейства `log1p()` возвращает значение натурального логарифма от аргумента *num+1*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно −1, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Зависимые функции

`log()`

Семейство функций log10

```
#include <math.h>
float log10f(float num);
double log10(double num);
long double log10l(long double num);
```

Функции `log10f()` и `log10l()` добавлены в версии C99.

Каждая функция семейства `log10()` возвращает значение логарифма по основанию 10 от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Данная программа выводит значение десятичных логарифмов чисел, изменяющихся от 1 до 10 с шагом 1, т.е. составляет таблицу десятичных логарифмов целых чисел от 1 до 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = 1.0;

  do {
    printf("%f %f\n", val, log10(val));
    val++;
  } while (val<11.0);

  return 0;
}
```

Зависимые функции

`log()` и `log2()`

Семейство функций log2

```
#include <math.h>
float log2f(float num);
double log2(double num);
long double log2l(long double num);
```

Функции `log2()`, `log2f()` и `log2l()` добавлены в версии C99.

Каждая функция семейства `log2()` возвращает значение логарифма по основанию 2 от аргумента *num*. Если значение аргумента *num* отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений[1].

Зависимые функции

`log()` и `log10()`

Семейство функций logb

```
#include <math.h>
float logbf(float num);
double logb(double num);
long double logbl(long double num);
```

Функции `logb()`, `logbf()` и `logbl()` добавлены в версии C99.

Каждая функция семейства `logb()` возвращает показатель аргумента *num*. Возвращаемое значение является числом с плавающей точкой. Если значение аргумента *num* равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

Зависимые функции

`ilogb()`

Семейство функций lrint

```
#include <math.h>
long int lrintf(float arg);
long int lrint(double arg);
long int lrintl(long double arg);
```

Функции `lrint()`, `lrintf()` и `lrintl()` добавлены в версии C99.

Каждая функция семейства `lrint()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long int`.

Зависимые функции

`llrint()` и `rint()`

[1] Как известно, в нуле логарифм не определен, но из-за трудностей представления близких к нулю положительных чисел автор придерживается столь осторожных формулировок. — *Прим. ред.*

Семейство функций lround

```
#include <math.h>
long int lroundf(float arg);
long int lround(double arg);
long int lroundl(long double arg);
```

Функции `lround()`, `lroundf()` и `lroundl()` добавлены в версии C99.

Каждая функция семейства `lround()` возвращает значение аргумента *arg*, округленное до ближайшего целого, которое имеет тип `long int`. Значения, отстоящие от большего и меньшего целых на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`llround()` и `round()`

Семейство функций modf

```
#include <math.h>
float modff(float num, float *i);
double modf(double num, double *i);
long double modfl(long double num, long double *i);
```

Функции `modff()` и `modfl()` добавлены в версии C99.

Каждая функция семейства `modf()` разбивает аргумент *num* на целую и дробную части. Функция возвращает дробную часть и размещает целую часть в переменной, адресуемой параметром *i*.

Пример

Данный фрагмент программы выводит на экран числа `10` и `0.123`.

```
double i;
double f;

f = modf(10.123, &i);
printf("%f %f", i, f);
```

Зависимые функции

`frexp()` и `ldexp()`

Семейство функций nan

```
#include <math.h>
float nanf(const char *content);
double nan(const char *content);
long double nanl(const char *content);
```

Функции `nan()`, `nanf()` и `nanl()` добавлены в версии C99.

Каждая функция семейства `nan()` возвращает значение, которое не является числом и которое содержит строку, адресуемую параметром *content*.

Зависимые функции

`isnan()`

Семейство функций nearbyint

```
#include <math.h>
float nearbyintf(float arg);
double nearbyint(double arg);
long double nearbyintl(long double arg);
```

Функции `nearbyint()`, `nearbyintf()` и `nearbyintl()` добавлены в версии C99.

Каждая функция семейства `nearbyint()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой.

Зависимые функции

`rint()` и `round()`

Семейство функций nextafter

```
#include <math.h>
float nextafterf(float from, float towards);
double nextafter(double from, double towards);
long double nextafterl(long double from, long double towards);
```

Функции `nextafter()`, `nextafterf()` и `nextafterl()` добавлены в версии C99.

Каждая функция семейства `nextafter()` возвращает значение, следующее после аргумента *from*, причем выбор следующего значения осуществляется в направлении, задаваемом аргументом *towards*.

Зависимые функции

`nexttoward()`

Семейство функций nexttoward

```
#include <math.h>
float nexttowardf(float from, long double towards);
double nexttoward(double from, long double towards);
long double nexttowardl(long double from, long double towards);
```

Функции `nexttoward()`, `nexttowardf()` и `nexttowardl()` добавлены в версии C99.

Каждая функция семейства `nexttoward()` возвращает значение, следующее после аргумента *from*, причем выбор следующего значения осуществляется в направлении, задаваемом аргументом *towards*. Действие этих функций аналогично действию функций семейства `nextafter()` за исключением того, что параметр всех трех функций *towards* имеет тип `long double`.

Зависимые функции

`nextafter()`

Семейство функций pow

```
#include <math.h>
float powf(float base, float exp);
double pow(double base, double exp);
long double powl(long double base, long double exp);
```

Функции `powf()` и `powl()` добавлены в версии C99.

Каждая функция семейства `pow()` возвращает значение аргумента *base*, возведенное в степень *exp*, т.е. в результате получается $base^{exp}$. Если значение аргумента *base* равно нулю, а *exp* меньше или равно нулю, возможна ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Она произойдет также в том случае, если *base* отрицательно, а *exp* не является целым числом. При этом также может возникнуть ошибка из-за выхода за пределы диапазона представимых значений.

Пример

Следующая программа выводит первые десять степеней числа 10, т.е. составляет таблицу степеней числа 10.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double x = 10.0, y = 0.0;

  do {
    printf("%f\n", pow(x, y));
    y++;
  } while(y<11.0);

  return 0;
}
```

Зависимые функции

`exp()`, `log()` и `sqrt()`

Семейство функций remainder

```
#include <math.h>
float remainderf(float a, float b);
double remainder(double a, double b);
long double remainderl(long double a, long double b);
```

Функции `remainder()`, `remainderf()` и `remainderl()` определены в версии C99.

Каждая функция семейства `remainder()` возвращает остаток от деления значений аргументов *a/b*.

Зависимые функции

`remquo()`

Семейство функций remquo

```
#include <math.h>
float remquof(float a, float b, int *quo);
double remquo(double a, double b, int *quo);
long double remquol(long double a, long double b, int *quo);
```

Функции `remquo()`, `remquof()` и `remquol()` определены в версии C99.

Каждая функция семейства `remquo()` возвращает остаток от деления значений аргументов *a/b*. При этом целое, адресуемое параметром *quo*, будет содержать частное.

Зависимые функции

`remainder()`

Семейство функций rint

```
#include <math.h>
float rintf(float arg);
double rint(double arg);
long double rintl(long double arg);
```

Функции `rint()`, `rintf()` и `rintl()` добавлены в версии C99.

Каждая функция семейства `rint()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой. Может возникнуть исключение вещественного типа.

Зависимые функции

`nearbyint()` и `round()`

Семейство функций round

```
#include <math.h>
float roundf(float arg);
double round(double arg);
long double roundl(long double arg);
```

Функции `round()`, `roundf()` и `roundl()` добавлены в версии C99.

Каждая функция семейства `round()` возвращает значение аргумента *arg*, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой. Значения, отстоящие от большего и меньшего целого на одинаковую величину (например, число 3.5), округляются в сторону большего целого.

Зависимые функции

`lround()` и `llround()`

Семейство функций scalbln

```
#include <math.h>
float scalblnf(float val, long int exp);
double scalbln(double val, long int exp);
long double scalblnl(long double val, long int exp);
```

Функции `scalbln()`, `scalblnf()` и `scalblnl()` добавлены в версии C99.

Каждая функция семейства `scalbln()` возвращает произведение параметра *val* и значения `FLT_RADIX`, возведенного в степень, которая равна значению параметра *exp*, т.е. в результате получается *val* * FLT_RADIX^{exp}.

Макрос `FLT_RADIX` определен в заголовке `<float.h>`, и его значение равно основанию системы счисления, используемой для представления вещественных чисел.

Зависимые функции

`scalbn()`

Семейство функций scalbn

```
#include <math.h>
float scalbnf(float val, int exp);
double scalbn(double val, int exp);
long double scalbnl(long double val, int exp);
```

Функции `scalbn()`, `scalbnf()` и `scalbnl()` добавлены в версии C99.

Каждая функция семейства `scalbn()` возвращает произведение параметра *val* и значения `FLT_RADIX`, возведенного в степень *exp*, т.е. в результате получается *val* * FLT_RADIX^{exp}.

Макрос `FLT_RADIX` определен в заголовке `<float.h>`, и его значение равно основанию системы счисления, используемой для представления вещественных чисел.

Зависимые функции

`scalbln()`

Семейство функций sin

```
#include <math.h>
float sinf(float arg);
double sin(double arg);
long double sinl(long double arg);
```

Функции `sinf()` и `sinl()` добавлены в версии C99.

Каждая функция семейства `sin()` возвращает значение синуса от аргумента *arg*. Значение аргумента должно быть задано в радианах.

Пример

Данная программа выводит синусы последовательности значений, лежащих в пределах от −1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу синусов чисел от −1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Синус %f равен %f.\n", val, sin(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `atan()`, `tan()`, `cos()`, `sinh()`, `cosh()` и `tanh()`

Семейство функций sinh

```
#include <math.h>
float sinhf(float arg);
double sinh(double arg);
long double sinhl(long double arg);
```

Функции `sinhf()` и `sinhl()` добавлены в версии C99.

Каждая функция семейства `sinh()` возвращает значение гиперболического синуса от аргумента *arg*.

Пример

Данная программа выводит гиперболические синусы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу гиперболических синусов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический синус %f равен %f.\n", val, sinh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

`asin()`, `acos()`, `atan2()`, `atan()`, `tan()`, `cos()`, `cosh()` и `sin()`

Семейство функций sqrt

```
#include <math.h>
float sqrtf(float num);
double sqrt(double num);
long double sqrtl(long double num);
```

Функции `sqrtf()` и `sqrtl()` добавлены в версии C99.

Каждая функция семейства `sqrt()` возвращает значение квадратного корня от аргумента *num*. Если значение аргумента отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения).

Пример

Данный фрагмент программы выводит на экран число `4`.

```
printf("%f", sqrt(16.0));
```

Зависимые функции

`exp()`, `log()` и `pow()`

Семейство функций tan

```
#include <math.h>
float tanf(float arg);
double tan(double arg);
long double tanl(long double arg);
```

Функции `tanf()` и `tanl()` добавлены в версии C99.

Каждая функция семейства `tan()` возвращает значение тангенса от аргумента *arg*. Значение аргумента должно быть задано в радианах.

Пример

Данная программа выводит тангенсы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу тангенсов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Тангенс %f равен %f.\n", val, tan(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

acos(), asin(), atan(), atan2(), cos(), sin(), sinh(), cosh() и tanh()

Семейство функций tanh

```
#include <math.h>
float tanhf(float arg);
double tanh(double arg);
long double tanhl(long double arg);
```

Функции tanhf() и tanhl() добавлены в версии C99.

Каждая функция семейства tanh() возвращает значение гиперболического тангенса от аргумента *arg*.

Пример

Данная программа выводит гиперболические тангенсы последовательности значений, лежащих в пределах от -1 до 1 и изменяющихся с шагом одна десятая, т.е. составляет таблицу гиперболических тангенсов чисел от -1 до 1.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
  double val = -1.0;

  do {
    printf("Гиперболический тангенс %f равен %f.\n", val, tanh(val));
    val += 0.1;
  } while(val<=1.0);

  return 0;
}
```

Зависимые функции

acos(), asin(), atan(), atan2(), cos(), sin(), cosh(), sinh() и tan ()

Семейство функций tgamma

```
#include <math.h>
float tgammaf(float arg);
double tgamma(double arg);
long double tgammal(long double arg);
```

Функции tgamma(), tgammaf() и tgammal() добавлены в версии C99.

Каждая функция семейства tgamma() возвращает значение гамма-функции от аргумента *arg*.

Зависимые функции

lgamma()

Семейство функций trunc

```
#include <math.h>
float truncf(float arg);
double trunc(double arg);
long double truncl(long double arg);
```

Функции `trunc()`, `truncf()` и `truncl()` добавлены в версии C99.

Каждая функция семейства `trunc()` возвращает усеченное значение аргумента *arg*, т.е. значение, в котором отброшена дробная часть[1].

Зависимые функции

`nearbyint()`

[1] Иногда говорят, что это округленное значение аргумента *arg*, причем округление в данном случае выполняется отбрасыванием дробной части. — *Прим. ред.*