API библиотеки арифметики произвольной точности MPL

К.В. Никулов, <*knikulov@yandex.ru*>

Г.А. Ситкарев, <*sitkarev@unixkomi.ru*>

Сыктывкарский Государственный Университет
Лаборатория Прикладной Математики и Программирования
http://amplab.syktsu.ru

Содержание

mpl 1
mpl_abs_cmp 2
mpl_add 3
mpl_and 4
mpl_clear 5
mpl_clearv 6
mpl_cmp 7
mpl_copy 8
mpl_dbg 9
mpl_div 10
mpl_gcd 11
mpl_init 12
mpl_initv 13
mpl_iseven 14
mpl_isneg 15
mpl_isodd 16
mpl_isone 17
mpl_iszero 18
mpl_mod_exp 19
mpl_mod_inv 20
mpl_mul_dig 21
mpl_mul 22
mpl_mul_ndig 23
mpl_nr_bits 24
mpl_or 25
mpl_primality_miller_rabin 26
mpl_random 27
mpl_reduce_barrett 28
mpl_reduce_barrett_setup 29
mpl_set_one 30
mpl_set_sint 31
mpl_set_str 32
mpl_set_uchar 33
mpl_set_uint 34
mpl_shl 35
mpl_shr 36
mpl_sqr 37
mpl_sub 38
mpl_swap 39
mpl_to_str 40
mpl_to_uchar 41
mpl_xor 42
mpl_zero 43

ИМЯ

mpl – библиотека арифметики произвольной точности

ОПИСАНИЕ

Данная библиотека содержит функции языков Си и C++ для работы с числами произвольной точности, то есть такими числами, длина которых, будучи выражена в битах, сильно превышает длину регистра ЦПУ. Изначально библиотека писалась как база для написания криптографических алгоритмов, в коих активно используются числа произвольной точности.

Для представления числа произвольной точности библиотека использует структуру **mpl_int**. Для использования функций библиотеки необходимо заводить переменные типа **mpl_int**, а затем инициализировать их с помощью функции **mpl_init**(3) или **mpl_initv**(3). Неинициализированные переменные использовать нельзя.

После того, как переменная выполнила свою функцию, необходимо очистить и освободить память, которую она использовала. Это делается с помощью функций **mpl_clear**(3) и **mpl_clearv**(3).

В каждую функцию библиотеки следует передавать не сами переменные **mpl_int**, а указатели на эти переменные. В библиотеке принято следующее соглашение: переменная, в которую будет записан результат, идёт первой в списке аргументов функции. Это сделано для того, чтобы сохранить естественный порядок аргументов, как в речи (C равно A плюс B).

Библиотека определяет несколько констант, настраивающих поведение её функций:

MPL_INT_BITS
: количество битов, задействованных для хранения одного разряда числа: 28 для 32-х разрядных процессоров x86;

MPL_INT_BASE
: основание системы счисления для арифметики произвольной точности. Для 32-х разрядных процессоров x86: 2^MPL_INT_BITS = 268435456;

MPL_INT_MASK
: битовая маска, выделяющая биты в слове, которые задействованы для хранения одного разряда числа;

MPL_INT_ALLOC_DEFAULT
: количество разрядов, выделяемое по умолчанию из кучи для хранения числа;

MPL_INT_APPEND
: количество разрядов, дополнительно выделяемых из кучи при необходимости.

Большая часть функций возвращает значение, по которому можно определить, как завершилась работа функции. Это значение может быть:

MPL_OK
: функция успешно завершила свою работу;

MPL_ERR
: в функцию был передан неверный аргумент, либо произошла какая-то ошибка;

MPL_NOMEM
: функции не удалось получить количество памяти, необходимое для работы;

MPL_COMPOSITE
: число не является простым (это значение может возвратить только функция **mpl_primality_miller_rabin**(3)).

СМОТРИ ТАКЖЕ

mpl_init(3), **mpl_initv**(3), **mpl_clear**(3), **mpl_clearv**(3), **mpl_zero**(3), **mpl_set_one**(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3), **mpl_iszero**(3), **mpl_isone**(3), **mpl_iseven**(3), **mpl_isodd**(3), **mpl_isneg**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_dbg**(3), **mpl_nr_bits**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_mul**(3), **mpl_mul_dig**(3), **mpl_mul_ndig**(3), **mpl_sqr**(3), **mpl_div**(3), **mpl_gcd**(3), **mpl_random**(3), **mpl_primality_miller_rabin**(3), **mpl_reduce_barrett_setup**(3), **mpl_reduce_barrett**(3), **mpl_mod_inv**(3), **mpl_mop_exp**(3), **mpl_cmp**(3), **mpl_abs_cmp**(3), **mpl_copy**(3), **mpl_swap**(3), **mpl_and**(3), **mpl_or**(3), **mpl_xor**(3)

ИМЯ

mpl_cmp – сравнить переменные mpl_int

СИНТАКСИС

#include <mpl.h>

int mpl_cmp(const mpl_int *_a_, const mpl_int *_b_);
int mpl_abs_cmp(const mpl_int *_a_, const mpl_int *_b_);

ОПИСАНИЕ

Функция *mpl_cmp* сравнивает числа *a* и *b*.

Функция *mpl_abs_cmp* сравнивает числа *a* и *b* по модулю (их абсолютные значения).

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функции возвращают:

MPL_CMP_GT если *a* больше *b*.

MPL_CMP_LT если *a* меньше *b*.

MPL_CMP_GT если *a* и *b* равны.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_add, mpl_sub, mpl_mul, mpl_div – простейшие арифметические действия

СИНТАКСИС

#include <mpl.h>

int mpl_add(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_sub(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_mul(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_div(mpl_int *_q_, mpl_int *_r_, const mpl_int *_y_, conts mpl_int *_x_);

ОПИСАНИЕ

Функция **mpl_add** складывает *a* и *b* и записывает сумму в *c*.

c = a + b

Функция **mpl_sub** вычитает из *a* число *b* и записывает разность в *c*.

c = a - b

Функция **mpl_mul** умножает *a* на *b* и записывает произведение в *c*.

*c = a * b*

Функция **mpl_div** делит с остатком *y* на *x* и записывает частное в *q*, а остаток в *r*.

q = y / x
r = y % x

Если остаток не нужен, то на место *r* можно подставить NULL:

```
mpl_int a, b, c;

...

mpl_div(&c, NULL, &a, &b);

...
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все функции возвращают **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

Функция **mpl_div** возвращает **MPL_ERR**, если делитель равен нулю.

ЗАМЕЧАНИЯ

Все функции могут использовать переменные-операнды как переменные для записи результата. К примеру:

```
mpl_int a, b;

...

mpl_mul(&b, &b, &b);
mpl_add(&a, &a, &b);
mpl_div(&b, &a, &a, &b);

...
```

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3), **mpl_cmp**(3), **mpl_abs_cmp**(3), **mpl_mul_dig**(3), **mpl_mul_ndig**(3), **mpl_sqr**(3), **mpl_gcd**(3), **mpl_primality_miller_rabin**(3), **mpl_reduce_barrett** **mpl_mod_exp**(3), **mpl_mod_inv**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_and**(3), **mpl_or**(3), **mpl_xor**(3), **mpl_copy**(3), **mpl_swap**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_and, mpl_or, mpl_xor – выполнить побитовую операцию

СИНТАКСИС

#include <mpl.h>

int mpl_and(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_or(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_xor(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**

ОПИСАНИЕ

Функция **mpl_and** записывает в *c* результат побитового И чисел *a* и *b*.

c = *a* & *b*

Функция **mpl_or** записывает в *c* результат побитового ИЛИ чисел *a* и *b*.

c = *a* | *b*

Функция **mpl_xor** записывает в *c* результат побитового ИСКЛЮЧАЮЩЕГО ИЛИ чисел *a* и *b*.

c = *a* ^ *b*

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все три функции возвращают **MPL_OK** в случае успеха, либо **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_shr**(3), **mpl_shl**(3)

ИМЯ

mpl_clear, mpl_clearv – инициализирует переменные типа mpl_int

СИНТАКСИС

#include <mpl.h>

void mpl_clear(mpl_int * *a*);
void mpl_clearv(mpl_int * *a*, ...);

ОПИСАНИЕ

Функции **mpl_clear** и **mpl_clearv** освобождают ресурсы, занимаемые переменными типа mpl_int.

Функция **mpl_clear** освобождает ресурсы переменной *a*.

Функция **mpl_clearv** освобождает ресурсы нескольких переменных. Список указателей, передаваемых в функцию, должен заканчиваться значением NULL.

ЗАМЕЧАНИЯ

Функции применимы только к инициализированным переменным.

СМОТРИ ТАКЖЕ

mpl_init(3), **mpl_initv**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_dbg**(3)

ИМЯ

mpl_clear, mpl_clearv – инициализирует переменные типа mpl_int

СИНТАКСИС

#include <mpl.h>

void mpl_clear(mpl_int * *a*);
void mpl_clearv(mpl_int * *a*, ...);

ОПИСАНИЕ

Функции **mpl_clear** и **mpl_clearv** освобождают ресурсы, занимаемые переменными типа mpl_int.

Функция **mpl_clear** освобождает ресурсы переменной *a*.

Функция **mpl_clearv** освобождает ресурсы нескольких переменных. Список указателей, передаваемых в функцию, должен заканчиваться значением NULL.

ЗАМЕЧАНИЯ

Функции применимы только к инициализированным переменным.

СМОТРИ ТАКЖЕ

mpl_init(3), **mpl_initv**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_dbg**(3)

ИМЯ

mpl_cmp – сравнить переменные mpl_int

СИНТАКСИС

#include <mpl.h>

int mpl_cmp(const mpl_int *_a_, const mpl_int *_b_);
int mpl_abs_cmp(const mpl_int *_a_, const mpl_int *_b_);

ОПИСАНИЕ

Функция *mpl_cmp* сравнивает числа *a* и *b*.

Функция *mpl_abs_cmp* сравнивает числа *a* и *b* по модулю (их абсолютные значения).

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функции возвращают:

MPL_CMP_GT если *a* больше *b*.

MPL_CMP_LT если *a* меньше *b*.

MPL_CMP_GT если *a* и *b* равны.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_copy – скопировать mpl_int

СИНТАКСИС

#include <mpl.h>

int mpl_copy(mpl_int ****dst*, const mpl_int ****src***);**

ОПИСАНИЕ

Функция **mpl_copy** копирует содержимое переменной *src* в переменную *dst*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция возвращает **MPL_OK** в случае успеха, либо **MPL_NOMEM** в случае ошибки.

СМОТРИ ТАКЖЕ

mpl_swap(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3),

ИМЯ

mpl_dbg – вывести отладочную информацию

СИНТАКСИС

#include <mpl.h>

void mpl_dbg(const mpl_int **a*, FILE ** *fp*);

ОПИСАНИЕ

Функция **mpl_dbg** записывает в файл *fp* следующиую отладочную информацию о переменной *a*: указатель на переменную *a*, номер вершины в числе, знак числа, указатель на массив с собственно числом, а так же содержание этого массива.

СМОТРИ ТАКЖЕ

mpl_clear(3), **mpl_clearv**(3)

ИМЯ

mpl_add, mpl_sub, mpl_mul, mpl_div – простейшие арифметические действия

СИНТАКСИС

#include <mpl.h>

int mpl_add(mpl_int ****c*, const mpl_int ****a***, conts mpl_int ****b***);**
int mpl_sub(mpl_int ****c*, const mpl_int ****a***, conts mpl_int ****b***);**
int mpl_mul(mpl_int ****c*, const mpl_int ****a***, conts mpl_int ****b***);**
int mpl_div(mpl_int ****q*, mpl_int ****r***, const mpl_int ****y***, conts mpl_int ****x***);**

ОПИСАНИЕ

Функция **mpl_add** складывает *a* и *b* и записывает сумму в *c*.

c = *a* + *b*

Функция **mpl_sub** вычитает из *a* число *b* и записывает разность в *c*.

c = *a* - *b*

Функция **mpl_mul** умножает *a* на *b* и записывает произведение в *c*.

c = *a* * *b*

Функция **mpl_div** делит с остатком *y* на *x* и записывает частное в *q*, а остаток в *r*.

q = *y* / *x*
r = *y* % *x*

Если остаток не нужен, то на место *r* можно подставить NULL:

```
mpl_int a, b, c;

...

mpl_div(&c, NULL, &a, &b);

...
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все функции возвращают **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

Функция **mpl_div** возвращает **MPL_ERR**, если делитель равен нулю.

ЗАМЕЧАНИЯ

Все функции могут использовать переменные-операнды как переменные для записи результата. К примеру:

```
mpl_int a, b;

...

mpl_mul(&b, &b, &b);
mpl_add(&a, &a, &b);
mpl_div(&b, &a, &a, &b);

...
```

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3), **mpl_cmp**(3), **mpl_abs_cmp**(3), **mpl_mul_dig**(3), **mpl_mul_ndig**(3), **mpl_sqr**(3), **mpl_gcd**(3), **mpl_primality_miller_rabin**(3), **mpl_reduce_barrett** **mpl_mod_exp**(3), **mpl_mod_inv**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_and**(3), **mpl_or**(3), **mpl_xor**(3), **mpl_copy**(3), **mpl_swap**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_gcd – найти наибольший общий делитель

СИНТАКСИС

#include <mpl.h>

int mpl_gcd(mpl_int *_c_, const mpl_int *_a_, const mpl_int *_b_);

ОПИСАНИЕ

Функция **mpl_gcd** записывает в *c* наибольший общий делитель чисел *a* и *b*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция возвращает **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_primality_miller_rabin**(3)

ИМЯ

mpl_init, mpl_initv – инициализирует переменные типа mpl_int

СИНТАКСИС

#include <mpl.h>

**int mpl_init(mpl_int ****a);*
int mpl_initv(mpl_int ****a*, ...);**

ОПИСАНИЕ

Функции **mpl_init** и **mpl_initv** инициализируют переменные типа mpl_int. Неинициализированные mpl_int использовать нельзя.

Функция **mpl_init** инициализирует переменную *a*.

Функция **mpl_initv** принимает последовательность указателей на переменные mpl_int, терминированную значением NULL. Инициализирует все переменные, указатели на которые передаются в функцию.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

В случае успеха возвращается **MPL_OK**. В случае ошибки возвращается **MPL_NOMEM**.

ЗАМЕЧАНИЯ

Повторная инициализация переменной типа mp_int возможна только после освобождения ресурсов этой переменной с помощью функции **mpl_clear**(3) или **mpl_clearv**(3).

СМОТРИ ТАКЖЕ

mpl_clear(3), **mpl_clearv**(3), **mpl_zero**(3), **mpl_set_one**(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_init, mpl_initv – инициализирует переменные типа mpl_int

СИНТАКСИС

#include <mpl.h>

**int mpl_init(mpl_int ****a);*
int mpl_initv(mpl_int ****a*, ...);**

ОПИСАНИЕ

Функции **mpl_init** и **mpl_initv** инициализируют переменные типа mpl_int. Неинициализированные mpl_int использовать нельзя.

Функция **mpl_init** инициализирует переменную *a*.

Функция **mpl_initv** принимает последовательность указателей на переменные mpl_int, терминированную значением NULL. Инициализирует все переменные, указатели на которые передаются в функцию.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

В случае успеха возвращается **MPL_OK**. В случае ошибки возвращается **MPL_NOMEM**.

ЗАМЕЧАНИЯ

Повторная инициализация переменной типа mp_int возможна только после освобождения ресурсов этой переменной с помощью функции **mpl_clear**(3) или **mpl_clearv**(3).

СМОТРИ ТАКЖЕ

mpl_clear(3), **mpl_clearv**(3), **mpl_zero**(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_iszero, mpl_isone, mpl_iseven, mpl_isodd, mpl_isneg – проверяет свойства переменной типа mp_int

СИНТАКСИС

#include <mpl.h>

int mpl_iszero(mpl_int ****a*);**
int mpl_isone(mpl_int ****a*);**
int mpl_iseven(mpl_int ****a*);**
int mpl_isodd(mpl_int ****a*);**
int mpl_isneg(mpl_int ****a*);**

ОПИСАНИЕ

Эти функции проверяют свойства переменной *a*.

mpl_iszero()
проверяет, является ли *a* нулём.

mpl_isone()
проверяет, является ли *a* единицей.

mpl_iseven()
проверяет, является ли *a* чётным числом.

mpl_isodd()
проверяет, является ли *a* нечётным числом.

mpl_isneg()
проверяет, является ли *a* отрицательным числом.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает 1, если переменная *a* обладает интересующим свойством, иначе возвращает 0.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_iszero, mpl_isone, mpl_iseven, mpl_isodd, mpl_isneg – проверяет свойства переменной типа mp_int

СИНТАКСИС

#include <mpl.h>

int mpl_iszero(mpl_int ****a*);**
int mpl_isone(mpl_int ****a*);**
int mpl_iseven(mpl_int ****a*);**
int mpl_isodd(mpl_int ****a*);**
int mpl_isneg(mpl_int ****a*);**

ОПИСАНИЕ

Эти функции проверяют свойства переменной *a*.

mpl_iszero()
проверяет, является ли *a* нулём.

mpl_isone()
проверяет, является ли *a* единицей.

mpl_iseven()
проверяет, является ли *a* чётным числом.

mpl_isodd()
проверяет, является ли *a* нечётным числом.

mpl_isneg()
проверяет, является ли *a* отрицательным числом.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает 1, если переменная *a* обладает интересующим свойством, иначе возвращает 0.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_iszero, mpl_isone, mpl_iseven, mpl_isodd, mpl_isneg – проверяет свойства переменной типа mp_int

СИНТАКСИС

#include <mpl.h>

int mpl_iszero(mpl_int ****a*);**
int mpl_isone(mpl_int ****a*);**
int mpl_iseven(mpl_int ****a*);**
int mpl_isodd(mpl_int ****a*);**
int mpl_isneg(mpl_int ****a*);**

ОПИСАНИЕ

Эти функции проверяют свойства переменной *a*.

mpl_iszero()
проверяет, является ли *a* нулём.

mpl_isone()
проверяет, является ли *a* единицей.

mpl_iseven()
проверяет, является ли *a* чётным числом.

mpl_isodd()
проверяет, является ли *a* нечётным числом.

mpl_isneg()
проверяет, является ли *a* отрицательным числом.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает 1, если переменная *a* обладает интересующим свойством, иначе возвращает 0.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_iszero, mpl_isone, mpl_iseven, mpl_isodd, mpl_isneg – проверяет свойства переменной типа mp_int

СИНТАКСИС

#include <mpl.h>

int mpl_iszero(mpl_int ****a*);**
int mpl_isone(mpl_int ****a*);**
int mpl_iseven(mpl_int ****a*);**
int mpl_isodd(mpl_int ****a*);**
int mpl_isneg(mpl_int ****a*);**

ОПИСАНИЕ

Эти функции проверяют свойства переменной *a*.

mpl_iszero()
проверяет, является ли *a* нулём.

mpl_isone()
проверяет, является ли *a* единицей.

mpl_iseven()
проверяет, является ли *a* чётным числом.

mpl_isodd()
проверяет, является ли *a* нечётным числом.

mpl_isneg()
проверяет, является ли *a* отрицательным числом.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает 1, если переменная *a* обладает интересующим свойством, иначе возвращает 0.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_iszero, mpl_isone, mpl_iseven, mpl_isodd, mpl_isneg – проверяет свойства переменной типа mp_int

СИНТАКСИС

#include <mpl.h>

int mpl_iszero(mpl_int ***a);**
int mpl_isone(mpl_int ***a);**
int mpl_iseven(mpl_int ***a);**
int mpl_isodd(mpl_int ***a);**
int mpl_isneg(mpl_int ***a);**

ОПИСАНИЕ

Эти функции проверяют свойства переменной *a*.

mpl_iszero()
проверяет, является ли *a* нулём.

mpl_isone()
проверяет, является ли *a* единицей.

mpl_iseven()
проверяет, является ли *a* чётным числом.

mpl_isodd()
проверяет, является ли *a* нечётным числом.

mpl_isneg()
проверяет, является ли *a* отрицательным числом.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает 1, если переменная *a* обладает интересующим свойством, иначе возвращает 0.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)

ИМЯ

mpl_mod_exp – возвести в степень по модулю

СИНТАКСИС

#include <mpl.h>

int mpl_mod_exp(mpl_int ****res*, const mpl_int ****a***, const mpl_int ****y***, const mpl_int ****b***);**

ОПИСАНИЕ

Функция **mpl_mod_exp** записывает в *res* результат возведения числа *a* в степнь *y* по модулю *b*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция **mpl_mod_exp** возвращает **MPL_OK** в случае успеха, либо сообщение об ошибки, если произошла ошибка.

ОШИБКИ

MPL_ERR функции были переданы неверные входные параметры.

MPL_NOMEM для работы функции недостаточно памяти.

ЗАМЕЧАНИЯ

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_mod_inv**(3)

ИМЯ

mpl_mod_inv – найти обратное по модулю число

СИНТАКСИС

#include <mpl.h>

int mpl_mod_inv(mpl_int *_c_, const mpl_int *_a_, const mpl_int *_b_);

ОПИСАНИЕ

Функция **mpl_mod_inv** записывает в *c* число, обратное по модулю *b* числу *a*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция **mpl_mod_inv** возвращает **MPL_OK** в случае успеха, либо сообщение об ошибке, если произошли ошибка.

ОШИБКИ

MPL_ERR функции были переданы неверные входные параметры.

MPL_NOMEM для работы функции недостаточно памяти.

ЗАМЕЧАНИЯ

Математическая запись того, что делает функция:

$$c = x : ax = 1 \pmod{b};$$

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_mod_exp**(3)

ИМЯ

mpl_mul_dig - умножить mpl_int на число

СИНТАКСИС

#include <mpl.h>

int mpl_mul_dig(mpl_int ***_c_, const mpl_int ***_a_**, _mpl_int_t** _b_**);**

ОПИСАНИЕ

Функция **mpl_mul_dig** умножает *a* на положительное целое число, записанное в *b*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция возвращает **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_mul(3), **mpl_mul_ndig**(3), **mpl**(7)

ИМЯ

mpl_add, mpl_sub, mpl_mul, mpl_div – простейшие арифметические действия

СИНТАКСИС

#include <mpl.h>

int mpl_add(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_sub(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_mul(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_div(mpl_int *_q_, mpl_int *_r_, const mpl_int *_y_, conts mpl_int *_x_);

ОПИСАНИЕ

Функция **mpl_add** складывает *a* и *b* и записывает сумму в *c*.

c = a + b

Функция **mpl_sub** вычитает из *a* число *b* и записывает разность в *c*.

c = a - b

Функция **mpl_mul** умножает *a* на *b* и записывает произведение в *c*.

*c = a * b*

Функция **mpl_div** делит с остатком *y* на *x* и записывает частное в *q*, а остаток в *r*.

q = y / x
r = y % x

Если остаток не нужен, то на место *r* можно подставить NULL:

```
mpl_int a, b, c;

...

mpl_div(&c, NULL, &a, &b);

...
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все функции возвращают **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

Функция **mpl_div** возвращает **MPL_ERR**, если делитель равен нулю.

ЗАМЕЧАНИЯ

Все функции могут использовать переменные-операнды как переменные для записи результата. К примеру:

```
mpl_int a, b;

...

mpl_mul(&b, &b, &b);
mpl_add(&a, &a, &b);
mpl_div(&b, &a, &a, &b);

...
```

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3), **mpl_cmp**(3), **mpl_abs_cmp**(3), **mpl_mul_dig**(3), **mpl_mul_ndig**(3), **mpl_sqr**(3), **mpl_gcd**(3), **mpl_primality_miller_rabin**(3), **mpl_reduce_barrett** **mpl_mod_exp**(3), **mpl_mod_inv**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_and**(3), **mpl_or**(3), **mpl_xor**(3), **mpl_copy**(3), **mpl_swap**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_mul_ndig - умножение нескольких разрядов двух чисел

СИНТАКСИС

#include <mpl.h>

int mpl_mul_ndig(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***, int** *ndig***);**

ОПИСАНИЕ

Функция **mpl_mul_ndig** умножает *a* на *b* и сохраняет в *c* первые *ndig* **MP_INT_BASE**-ичных разрядов произведения.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция возвращает **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_mul(3), **mpl_mul_dig**(3), **mpl**(7)

ИМЯ

mpl_nr_bits – длина числа в битах

СИНТАКСИС

#include <mpl.h>

long int mpl_nr_bits(const mpl_int *a***);**

ОПИСАНИЕ

Функция **mpl_nr_bits** вычисляет количество значащих бит числа *a*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает количество значащих бит числа *a*.

СМОТРИ ТАКЖЕ

mpl_random(3), **mpl_shr**(3), **mpl_shl**(3)

ИМЯ

mpl_and, mpl_or, mpl_xor – выполнить побитовую операцию

СИНТАКСИС

#include <mpl.h>

int mpl_and(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_or(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_xor(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**

ОПИСАНИЕ

Функция **mpl_and** записывает в *c* результат побитового И чисел *a* и *b*.

c = *a* & *b*

Функция **mpl_or** записывает в *c* результат побитового ИЛИ чисел *a* и *b*.

c = *a* | *b*

Функция **mpl_xor** записывает в *c* результат побитового ИСКЛЮЧАЮЩЕГО ИЛИ чисел *a* и *b*.

c = *a* ^ *b*

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все три функции возвращают **MPL_OK** в случае успеха, либо **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_shr**(3), **mpl_shl**(3)

ИМЯ

mpl_primality_miller_rabin - проверить число на простоту

СИНТАКСИС

#include <mpl.h>

mpl_primality_miller_rabin(const mpl_int *_a_, int _r_
int (*_rnd_)(void *_buf_, size_t _len_, void *_rndctx_), void *_rndctx_);

ОПИСАНИЕ

Функция **mpl_primality_miller_rabin** проверяет число *a* на простоту методом Миллера-Рабина, пытаясь найти *r* свидетелей простоты.

Функция **rnd** служит источником энтропии, и записывает в буфер *buf* длины *len* случайные байты, возможно, используя контекст *rndctx*.

Функция **rnd** должна возвращать количество байтов, записанных в *buf*, либо -1 в случае ошибки.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция **mpl_miller_rabin** возвращает **MPL_OK**, если число *a* - простое, **MPL_COMPOSITE**, если *a* - составное, либо сообщение об ошибке, если произошла ошибка.

ОШИБКИ

MPL_ERR один из переданных аргументов имеет неверное значение. **MPL_NOMEM** для работы функции недостаточно памяти.

ЗАМЕЧАНИЯ

Функция **mpl_primality_miller_rabin** не гарантирует верного определения простоты числа со стопроцентной вероятностью.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_gcd**(3), **mpl_random**(3)

ИМЯ

mpl_random – сгенерировать случайное число

СИНТАКСИС

#include <mpl.h>

int mpl_random(mpl_int ****a*, int** *size***,**
int (**rnd***)(void ****buf***, size_t** *len***, void ****rndctx***), void ****rndctx***);**

ОПИСАНИЕ

Функция **mpl_random** записывает в *a* случайное число размера *size* байт.

Источником энтропии служит функция **rnd**, которая записывает в буфер *buf* длины *len* случайные байты, возможно, используя контекст *rndctx*.

Функция **rnd** должна возвращать количество байтов, записанных в *buf*, либо -1 в случае ошибки.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция **mpl_random** возвращает **MPL_OK** в случае успеха, или сообщение об ошибке в случае ошибки.

ОШИБКИ

MPL_NOMEM для работы функции не хватило памяти.

MPL_ERR функция **rnd** вернула -1.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_nr_bits**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_to_uchar**(3), **mpl_primality_miller_rabin**(3)

ИМЯ

mpl_reduce_barrett_setup, mpl_reduce_barrett – редукция по модулю методом Барретта

СИНТАКСИС

#include <mpl.h>

int mpl_reduce_barrett_setup(mpl_int ****mu*, const mpl_int ****b***);**

int mpl_reduce_barrett(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***, const mpl_int ****mu***);**

ОПИСАНИЕ

Функция **mpl_reduce_barrett_setup** вычисляет *mu*, необходимое для редукции по модулю *b*.

Функция **mpl_reduce_barrett** выполняет редукцию числа *a* по модулю *b*, записывая результат редукции в *c*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Обе функции возвращают **MPL_OK** в случае успеха, либо сообщение об ошибке в случае ошибки.

ОШИБКИ

MPL_NOMEM для работы функции недостаточно памяти.

MPL_ERR число *a* не нуждается в редукции, т.к. оно меньше, чем модуль *b*.

ЗАМЕЧАНИЯ

Редукция по модулю, которую выполняет функция **mpl_reduce_barrett**, математически записывается так:

$$c = a \pmod{b}$$

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_reduce_barrett_setup, mpl_reduce_barrett – редукция по модулю методом Барретта

СИНТАКСИС

#include <mpl.h>

int mpl_reduce_barrett_setup(mpl_int *_mu_, const mpl_int *_b_);

int mpl_reduce_barrett(mpl_int *_c_, const mpl_int *_a_, const mpl_int *_b_, const mpl_int *_mu_);

ОПИСАНИЕ

Функция **mpl_reduce_barrett_setup** вычисляет *mu*, необходимое для редукции по модулю *b*.

Функция **mpl_reduce_barrett** выполняет редукцию числа *a* по модулю *b*, записывая результат редукции в *c*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Обе функции возвращают **MPL_OK** в случае успеха, либо сообщение об ошибке в случае ошибки.

ОШИБКИ

MPL_NOMEM для работы функции недостаточно памяти.

MPL_ERR число *a* не нуждается в редукции, т.к. оно меньше, чем модуль *b*.

ЗАМЕЧАНИЯ

Редукция по модулю, которую выполняет функция **mpl_reduce_barrett**, математически записывается так:

$$c = a \pmod{b}$$

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_set_one, mpl_set_sint, mpl_set_uint – устанавливают переменные типа mp_int в небольшие значения

СИНТАКСИС

#include <mpl.h>

void mpl_set_one(mpl_int ****a*);**
void mpl_set_sint(mpl_int ****a*, int** *value***);**
void mpl_set_uint(mpl_int ****a*, unsigned int** *value***);**

ОПИСАНИЕ

Эти функции предназначены для относительно быстрого присваивания небольших значений переменным типа mpl_int.

Функция **mpl_set_one** устанавливает значение переменной *a* в 1.

Функция **mpl_set_sint** записывает в переменную *a* число, содержащееся в младших **MPL_INT_BITS** (см. **mpl**(7)) переменной *value*, сохраняя у переменной *a* знак переменной *value*.

Функция **mpl_set_uint** записывает в переменную *a* положительное число, содержащееся в младших **MP_INT_BITS** битах числа *value*.

СМОТРИ ТАКЖЕ

mpl_set_uchar(3), **mpl_set_str**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_set_one, mpl_set_sint, mpl_set_uint – устанавливают переменные типа mp_int в небольшие значения

СИНТАКСИС

#include <mpl.h>

void mpl_set_one(mpl_int ****a*);**
void mpl_set_sint(mpl_int ****a*, int** *value***);**
void mpl_set_uint(mpl_int ****a*, unsigned int** *value***);**

ОПИСАНИЕ

Эти функции предназначены для относительно быстрого присваивания небольших значений переменным типа mpl_int.

Функция **mpl_set_one** устанавливает значение переменной *a* в 1.

Функция **mpl_set_sint** записывает в переменную *a* число, содержащееся в младших **MPL_INT_BITS** (см. **mpl**(7)) переменной *value*, сохраняя у переменной *a* знак переменной *value*.

Функция **mpl_set_uint** записывает в переменную *a* положительное число, содержащееся в младших **MP_INT_BITS** битах числа *value*.

СМОТРИ ТАКЖЕ

mpl_set_uchar(3), **mpl_set_str**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_set_str, mpl_to_str - запись/чтение значения из строки

СИНТАКСИС

#include <mpl.h>

int mpl_set_str(mpl_int *_a_, const char *_str_, int _base_);
int mpl_to_str(const mpl_int *_a_, const char *_str_, int _base_, int _len_);

ОПИСАНИЕ

Функция **mpl_set_str** записывает в переменную *a* число в системе счисления с основанием *base*, хранящееся в строке *str*.

Функция **mpl_to_str** считывает число в *base*-ичной системе счисления из переменной *a* в строку *str* длиной *len*.

Основание системы счёта *base* должно быть целым числом в промежутке от 2 до 36 включительно.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает **MPL_OK** в случае успеха, либо сообщение об ошибке.

ОШИБКИ

MPL_NOMEM недостаточно памяти для записи числа.

MPL_ERR значение *base* вне допустимого интервала, либо в *str* встретился недопустимый символ, либо строки длины *len* недостаточно, чтобы записать туда число в данной системе счисления.

ЗАМЕЧАНИЯ

Число в *str* может начинаться со знака '–'. В таком случае считается, что ему соответствует отрицательное число в mpl_int.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_set_uchar, mpl_to_uchar – запись/чтение значения mpl_int

СИНТАКСИС

#include <mpl.h>

int mpl_set_uchar(mpl_int ****a*, const unsigned char ****buf***, int** *len***);**
int mpl_to_uchar(const mpl_int ****a*, unsigned char ****buf***, int** *len***);**

ОПИСАНИЕ

Функция **mpl_set_uchar** записывает положительное число в переменную *a* из буфера *buf* размера *len* байт.

Функция **mpl_to_uchar** считывает положительное число в буфер *buf* размера *len* байт из перменной *a*. Если число длиннее, чем *len*, то считываются только первые *len* байт.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

В случае успеха возвращается **MPL_OK**, или **MPL_NOMEM** в случае ошибки.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_str**(3), **mpl_to_str**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_set_one, mpl_set_sint, mpl_set_uint – устанавливают переменные типа mp_int в небольшие значения

СИНТАКСИС

#include <mpl.h>

void mpl_set_one(mpl_int ****a*);**
void mpl_set_sint(mpl_int ****a*, int** *value***);**
void mpl_set_uint(mpl_int ****a*, unsigned int** *value***);**

ОПИСАНИЕ

Эти функции предназначены для относительно быстрого присваивания небольших значений переменным типа mpl_int.

Функция **mpl_set_one** устанавливает значение переменной *a* в 1.

Функция **mpl_set_sint** записывает в переменную *a* число, содержащееся в младших **MPL_INT_BITS** (см. **mpl**(7)) переменной *value*, сохраняя у переменной *a* знак переменной *value*.

Функция **mpl_set_uint** записывает в переменную *a* положительное число, содержащееся в младших **MP_INT_BITS** битах числа *value*.

СМОТРИ ТАКЖЕ

mpl_set_uchar(3), **mpl_set_str**(3), **mpl_to_uchar**(3), **mpl_to_str**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_shr, mpl_shl – битовые сдвиги

СИНТАКСИС

#include <mpl.h>

int mpl_shr(mpl_int ***a, unsigned int** nr**);**
int mpl_shl(mpl_int ***a, unsigned int** nr**);**

ОПИСАНИЕ

Функции **mpl_shr** и **mpl_shl** выполняют битовый сдвиг (вправо и влево соответственно) числа *a* на *nr* битов.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функции возвращают **MPL_OK** в случае успеха, либо **MPL_NOMEM** в случае ошибки.

СМОТРИ ТАКЖЕ

mpl_and(3), **mpl_or**(3), **mpl_xor**(3), **mpl_nr_bits**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_shr, mpl_shl – битовые сдвиги

СИНТАКСИС

#include <mpl.h>

int mpl_shr(mpl_int ****a*, unsigned int** *nr***);**
int mpl_shl(mpl_int ****a*, unsigned int** *nr***);**

ОПИСАНИЕ

Функции **mpl_shr** и **mpl_shl** выполняют битовый сдвиг (вправо и влево соответственно) числа *a* на *nr* битов.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функции возвращают **MPL_OK** в случае успеха, либо **MPL_NOMEM** в случае ошибки.

СМОТРИ ТАКЖЕ

mpl_and(3), **mpl_or**(3), **mpl_xor**(3), **mpl_nr_bits**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_sqr – возвести число в квадрат

СИНТАКСИС

#include <mpl.h>

int mpl_sqr(mpl_int * *c*, **const mpl_int *** *x*);

ОПИСАНИЕ

Функция **mpl_sqr** записывает в *c* результат возведения в квадрат числа *x*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Функция возвращает **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_add, mpl_sub, mpl_mul, mpl_div – простейшие арифметические действия

СИНТАКСИС

#include <mpl.h>

int mpl_add(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_sub(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_mul(mpl_int *_c_, const mpl_int *_a_, conts mpl_int *_b_);
int mpl_div(mpl_int *_q_, mpl_int *_r_, const mpl_int *_y_, conts mpl_int *_x_);

ОПИСАНИЕ

Функция **mpl_add** складывает *a* и *b* и записывает сумму в *c*.

c = a + b

Функция **mpl_sub** вычитает из *a* число *b* и записывает разность в *c*.

c = a - b

Функция **mpl_mul** умножает *a* на *b* и записывает произведение в *c*.

*c = a * b*

Функция **mpl_div** делит с остатком *y* на *x* и записывает частное в *q*, а остаток в *r*.

q = y / x
r = y % x

Если остаток не нужен, то на место *r* можно подставить NULL:

```
mpl_int a, b, c;

...

mpl_div(&c, NULL, &a, &b);

...
```

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все функции возвращают **MPL_OK** в случае успеха, или **MPL_NOMEM**, если произошла ошибка.

Функция **mpl_div** возвращает **MPL_ERR**, если делитель равен нулю.

ЗАМЕЧАНИЯ

Все функции могут использовать переменные-операнды как переменные для записи результата. К примеру:

```
mpl_int a, b;

...

mpl_mul(&b, &b, &b);
mpl_add(&a, &a, &b);
mpl_div(&b, &a, &a, &b);

...
```

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3), **mpl_cmp**(3), **mpl_abs_cmp**(3), **mpl_mul_dig**(3), **mpl_mul_ndig**(3), **mpl_sqr**(3), **mpl_gcd**(3), **mpl_primality_miller_rabin**(3), **mpl_reduce_barrett** **mpl_mod_exp**(3), **mpl_mod_inv**(3), **mpl_shr**(3), **mpl_shl**(3), **mpl_and**(3), **mpl_or**(3), **mpl_xor**(3), **mpl_copy**(3), **mpl_swap**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_swap – поменять местами

СИНТАКСИС

#include <mpl.h>

int mpl_swap(mpl_int ****a*, mpl_int ****b***);**

ОПИСАНИЕ

Функция **mpl_swap** меняет местами значения в переменных *a* и *b*.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает **MPL_OK** в случае успеха.

СМОТРИ ТАКЖЕ

mpl_copy(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3)

ИМЯ

mpl_set_str, mpl_to_str - запись/чтение значения из строки

СИНТАКСИС

#include <mpl.h>

int mpl_set_str(mpl_int ****a*, const char ****str***, int** *base***);**
int mpl_to_str(const mpl_int ****a*, const char ****str***, int** *base***, int** *len***);**

ОПИСАНИЕ

Функция **mpl_set_str** записывает в переменную *a* число в системе счисления с основанием *base*, хранящееся в строке *str*.

Функция **mpl_to_str** считывает число в *base*-ичной системе счисления из переменной *a* в строку *str* длиной *len*.

Основание системы счёта *base* должно быть целым числом в промежутке от 2 до 36 включительно.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Возвращает **MPL_OK** в случае успеха, либо сообщение об ошибке.

ОШИБКИ

MPL_NOMEM недостаточно памяти для записи числа.

MPL_ERR значение *base* вне допустимого интервала, либо в *str* встретился недопустимый символ, либо строки длины *len* недостаточно, чтобы записать туда число в данной системе счисления.

ЗАМЕЧАНИЯ

Число в *str* может начинаться со знака '−'. В таком случае считается, что ему соответствует отрицательное число в mpl_int.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_to_uchar**(3), **mpl_to_str**(3)

ИМЯ

mpl_set_uchar, mpl_to_uchar – запись/чтение значения mpl_int

СИНТАКСИС

#include <mpl.h>

int mpl_set_uchar(mpl_int ****a*, const unsigned char ****buf***, int** *len***);**
int mpl_to_uchar(const mpl_int ****a*, unsigned char ****buf***, int** *len***);**

ОПИСАНИЕ

Функция **mpl_set_uchar** записывает положительное число в переменную *a* из буфера *buf* размера *len* байт.

Функция **mpl_to_uchar** считывает положительное число в буфер *buf* размера *len* байт из перменной *a*. Если число длиннее, чем *len*, то считываются только первые *len* байт.

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

В случае успеха возвращается **MPL_OK**, или **MPL_NOMEM** в случае ошибки.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_str**(3), **mpl_to_str**(3), **mpl_add**(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_random**(3)

ИМЯ

mpl_and, mpl_or, mpl_xor – выполнить побитовую операцию

СИНТАКСИС

#include <mpl.h>

int mpl_and(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_or(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**
int mpl_xor(mpl_int ****c*, const mpl_int ****a***, const mpl_int ****b***);**

ОПИСАНИЕ

Функция **mpl_and** записывает в *c* результат побитового И чисел *a* и *b*.

c = *a* & *b*

Функция **mpl_or** записывает в *c* результат побитового ИЛИ чисел *a* и *b*.

c = *a* | *b*

Функция **mpl_xor** записывает в *c* результат побитового ИСКЛЮЧАЮЩЕГО ИЛИ чисел *a* и *b*.

c = *a* ^ *b*

ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ

Все три функции возвращают **MPL_OK** в случае успеха, либо **MPL_NOMEM**, если произошла ошибка.

СМОТРИ ТАКЖЕ

mpl_add(3), **mpl_sub**(3), **mpl_div**(3), **mpl_mul**(3), **mpl_shr**(3), **mpl_shl**(3)

ИМЯ

mpl_zero – зануляет переменную типа mpl_int

СИНТАКСИС

#include <mpl.h>

void mpl_zero(mpl_int ****a*);**

ОПИСАНИЕ

Функция **mpl_zero** устанавливает значение переменной *a* в 0 с положительным знаком.

СМОТРИ ТАКЖЕ

mpl_set_one(3), **mpl_set_sint**(3), **mpl_set_uint**(3), **mpl_set_uchar**(3), **mpl_set_str**(3)