

Целые	Целые (cont)	Целые (cont)
Установка $n^{\text{го}}$ бита	$x \mid (1 \ll n)$	Different rightmost bit of numbers m & n $(n^m) \& - (n^m) // \text{returns } 2^x \text{ where } x \text{ is the position of the different bit (0 based)}$
Выключение $n^{\text{го}}$ бита	$x \& \sim (1 \ll n)$	Common rightmost bit of numbers m & n $\sim (n^m) \& (n^m) + 1 // \text{returns } 2^x \text{ where } x \text{ is the position of the common bit (0 based)}$
инверсия $n^{\text{го}}$ бита	$x \wedge (1 \ll n)$	
Округление вниз	$n \gg 0$ $5.7812 \gg 0 // \text{вернёт } 5$	
Умножение на 2	$1n \ll 1;$	
Деление на 2	$n \gg 1;$	
Умножение на $2^{\text{ую}}$ степень двойки	$n \ll m;$	
Деление на $2^{\text{ую}}$ степень двойки	$n \gg m;$	
Проверка на чётность (кратность 2)	$(n \& 1) == 1;$	
Максимум из двух	$b \& ((a-b) \gg 31) \mid a \& (\sim (a-b) \gg 31);$	
Минимум из двух	$a \& ((a-b) \gg 31) \mid b \& (\sim (a-b) \gg 31);$	
Проверка на одинаковый знак	$(x \wedge y) \geq 0;$	
Смена знака	$i = \sim i + 1; // \text{or}$ $i = (i \wedge -1) + 1; // i = -i$	
Вернёт 2^n	$1 \ll n;$	
Является ли число степенью 2	$n > 0 \&\& !(n \& (n - 1));$	
Остаток от деления на 2^n на m	$m \& ((1 \ll n) - 1);$	
Среднее арифметическое	$(x + y) \gg 1;$ $((x \wedge y) \gg 1) + (x \oplus y) \gg 1;$	
	Получить $n^{\text{ый}}$ бит из n (от младшего к старшему)	
	Получить $n^{\text{ый}}$ бит из n (от старшего к младшему)	
	Проверить включен ли $n^{\text{ый}}$ бит	
	Выделение самого правого включенного бита	
	Выделение самого правого выключенного бита	
	Выделение $x \mid (x+1)$	
	Выделение $x \& (x-1)$	
	Выделение $x \mid (x+1)$	
	Выделение $x \& (x-1)$	
	Получение отрицательного значения	
	if $(x == a)$ $x = b;$	
	if $(x == b)$ $x = a;$	
	Поменять смежные биты	

Округление до ближайшей степени двойки

unsigned int v; // работает только с 32 битными числами

```
v--;  
v |= v >> 1;  
v |= v >> 2;  
v |= v >> 4;  
v |= v >> 8;  
v |= v >> 16;  
v++;
```

Получение максимального целого

```
int maxInt = ~(1 << 31);  
int maxInt = (1 << 31) - 1;  
int maxInt = (1 << -1) - 1;  
int maxInt = -1u >> 1;
```

Получение минимального целого

```
int minInt = 1 << 31;  
int minInt = 1 << -1;
```

Получение максимального long

```
long maxLong = ((long)1 << 127) -
```

Остаток от деления

```
n & 0b1; // на 2  
n & 0b11; // на 4  
n & 0b111; // на 8
```

И так далее

Проверка равенства

```
(a^b) == 0; // a == b  
!(a^b) // использовать внутри if()
```

Обмен значениями

```
//version 1  
a ^= b;  
b ^= a;  
a ^= b;
```

```
//version 2  
a = a ^ b ^ (b = a)
```

Получение абсолютного значения

```
//version 1  
x < 0 ? -x : x;  
  
//version 2  
(x ^ (x >> 31)) - (x >> 31);
```



By **Arest** (arest)
cheatography.com/arest/

Published 24th November, 2021.
Last updated 24th November, 2021.
Page 1 of 3.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Строки	Десятичные дроби (cont)	Приоритет операций (cont)
Конвертировать в нижний регистр` (x '') <i>Пример: ('a' ' ') => 'a' ; ('A' ' ') => 'a'</i>	float root(float x, int n) { #DEFINE MAN_MASK 0x7fffff #DEFINE EXP_MASK 0x7f800000 #DEFINE EXP_BIAS 0x3f800000 uint32_t bits = f2i(x); uint32_t man = bits & MAN_MASK; uint32_t exp = (bits & EXP_MASK) - EXP_BIAS; return i2f((man + man / n) ((EXP_BIAS + exp / n) & EXP_MASK)); }	++ -- + - ! ~ (t & sizeof new, delete, .* - * new[] delete[] >*
Конвертировать в верхний регистр (x & '_') <i>Пример: ('a' & ' _') => 'A' ; ('A' & ' _') => 'A'</i>		% + - << >> < < >= == != & ^ & ?: = += -= *= /= % >>= &= ^= =
Инvertировать регистр (x ^ '') <i>Пример: ('a' ^ ' ') => 'A' ; ('A' ^ ' ') => 'a'</i>	Быстрая степень return i2f((1 - exp) (0x3f800000 - 0x5c416) + f2i(x) exp)	Чтобы не плодить скобки, нужно знать приоритет операций
Позиция буквы в алфавите (англ) (x & "\x1F") <i>Пример: ('a' & " \x1 F") => 1 ; ('B' & " \x1 F") => 2</i>	Быстрый натуральный логарифм #DEFINE EPSILON 1.1920928955078-125e-07 #DEFINE LOG2 0.6931471805599453 return (f2i(x) - (0x3f800000 - 0x66774)) EPSILON LOG2 => 2	Приоритет операций Чтобы не плодить скобки, нужно знать приоритет операций. В C++ он такой:
Позиция большой буквы в алфавите (англ) (x & '?') или (x ^ '@') <i>Пример: ('C' & '?') => 3 ; ('Z' ^ '@') => 6</i>	Быстрая экспонента return i2f(0x3f800000 + (uint32_t)(x * (0x800000 + 0x38aa22)))	Приоритет операций :: ++ -- () [] . -> ++ -- + - ! (type) * & sizeof new, new[] delete, delete[] .* -> ** / % + - << >> < <= > >= == != & ^ && ?: = += -= *= /= %= <<= >>= &= ^= =
Позиция строчной буквы в алфавите (англ) (x ^ '') <i>Пример: ('d' ^ ' ') => 4 ; ('x' ^ ' ') => 24</i>	Примечание: хаки с float могут не работать на Ардуино! Разбить float в массив бит (unsigned uint32_t)	
Десятичные дроби Примечание: хаки с float могут не работать на Ардуино! Разбить float в массив бит (unsigned uint32_t) #include <stdint.h> typedef union {float flt; uint32_t bits} lens_t; uint32_t f2i(float x) { return ((lens_t) {.flt = x}).bits; }	Другое Быстрая конвертация цвета R5G5B5 в R8G8B8 R8 = (R5 << 3) (R5 >> 2) G8 = (R5 << 3) (R5 >> 2) B8 = (R5 << 3) (R5 >> 2)	
Вернуть массив бит обратно в float float i2f(uint32_t x) { return ((lens_t) { .bits = x }).flt; }	Приоритет операций Чтобы не плодить скобки, нужно знать приоритет операций	
Быстрый обратный квадратный корень return i2f(0x 5f3 759df - f2i(x) / 2) ;		
Быстрый n ^{ый} корень из целого числа		



By **Arest** (arest)
cheatography.com/arest/

Published 24th November, 2021.
 Last updated 24th November, 2021.
 Page 2 of 3.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopad.com>