Побитовые операции и хранение переменных

МИНУТКА ЧЕРНОЙ МАГИИ

```cpp
#include <iostream>
using namespace std;

int main() {
    char a = 0;
    while (true) {
        cout << (int)a++;
        cin.get();
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main() {
    unsigned long int a = 10;
    while (true) {
        cout << (a *= a);
        cin.get();
    }
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int main() {
    unsigned long long int res_short = 1;
    int i_short = 1;
    unsigned long long int res_long = 1;
    unsigned long long int i_long = 1;
    while (true) {
        i_short *= 2;
        i_long *= 2;
        res_short += 16 * i_short;
        res_long += 16 * i_long;
        cout << res_short << "\t\t\t" << res_long;
        cin.get();
    }
    return 0;
}
```

ПРИОРИТЕТ ОПЕРАТОРОВ

Приоритет	Операция	Описание	Ассоциативность
1	::	Область видимости	Слева направо
2	()	Вызов функции	
	[]	Обращение к массиву по индексу	
	.	Выбор элемента по ссылке	
	–>	Выбор элемента по указателю	
3	++a --a	Прединкремент и преддекремент	Справа налево
	+ –	Унарный плюс и минус	
	! ~	Логическое НЕ и побитовое НЕ	
	(type)	Приведение к типу type	
	*	Indirection (разыменование)	
	&	Адрес	
	sizeof	Размер	
	new, new[]	Динамическое выделение памяти	
	delete, delete[]	Динамическое освобождение памяти	
4	.* ->*	Указатель на член	Слева направо
5	* / %	Умножение, деление и остаток	
6	+ –	Сложение и вычитание	
7	<< >>	Побитовый сдвиг влево и вправо	
8	< <=	Операции сравнения < и ≤	
	> >=	Операции сравнения > и ≥	
9	== !=	Операции сравнения = и ≠	
10	&	Побитовое И	
11	^	Побитовый XOR (исключающее ИЛИ)	
12	|	Побитовое ИЛИ (inclusive or)	
13	&&	Логическое И	
14	||	Логическое ИЛИ	
15	?:	Тернарное условие	Справа налево
	=	Прямое присваивание (предоставляемое по умолчанию для C++ классов)	
	+= –=	Присвоение с суммированием и разностью	
	*= /= %=	Присвоение с умножением, делением и остатком от деления	
	<<= >>=	Присвоение с побитовым сдвигом слево и вправо	
	&= ^= |=	Присвоение с побитовыми логическими операциями (И, XOR, ИЛИ)	
16	throw	Операция выброса исключения	
17	a++ a--	Постинкремент и постдекремент	Слева направо

ПРИОРИТЕТ ОПЕРАТОРОВ

Приоритет	Операция	Описание	Ассоциативность
3	++a --a	Прединкремент и преддекремент	Справа налево
	! ~	Логическое НЕ и побитовое НЕ	
5	* / %	Умножение, деление и остаток	Слева направо
6	+ −	Сложение и вычитание	
7	<< >>	Побитовый сдвиг влево и вправо	
8	< <=	Операции сравнения < и ≤	
	> >=	Операции сравнения > и ≥	
9	== !=	Операции сравнения = и ≠	
10	&	Побитовое И	
11	^	Побитовый XOR (исключающее ИЛИ)	
12	|	Побитовое ИЛИ (inclusive or)	
13	&&	Логическое И	
14	||	Логическое ИЛИ	
15	=	Прямое присваивание (предоставляемое по умолчанию для C++ классов)	Справа налево
	+= −=	Присвоение с суммированием и разностью	
	*= /= %=	Присвоение с умножением, делением и остатком от деления	
	<<= >>=	Присвоение с побитовым сдвигом слево и вправо	
	&= ^= |=	Присвоение с побитовыми логическими операциями (И, XOR, ИЛИ)	
17	a++ a--	Постинкремент и постдекремент	Слева направо

ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

- Десятичная

$615_{10} = 6*10^2 + 1*10^1 + 5*10^0$

10^3	10^2	10^1	10^0
0	6	1	5

- Двоичная

$1101_2 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 =$

$8 + 4 + 1 = 13_{10}$

$2^3 = 8$ $2^2 = 4$ $2^1 = 2$ $2^0 = 1$

1	1	0	1

СИСТЕМЫ СЧИСЛЕНИЯ В КОМПЬЮТЕРАХ

DEC	BIN	OCT	HEX	BCD
0	0000	0	0	0000
1	0001	1	1	0001
2	0010	2	2	0010
3	0011	3	3	0011
4	0100	4	4	0100
5	0101	5	5	0101
6	0110	6	6	0110
7	0111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

Двоично-десятичный код
(англ. binary-coded decimal)

36-ричная система счисления (все буквы + цифры),
пример - серийный номер программного обеспечения

https://youtu.be/mBgk8vGL6ic

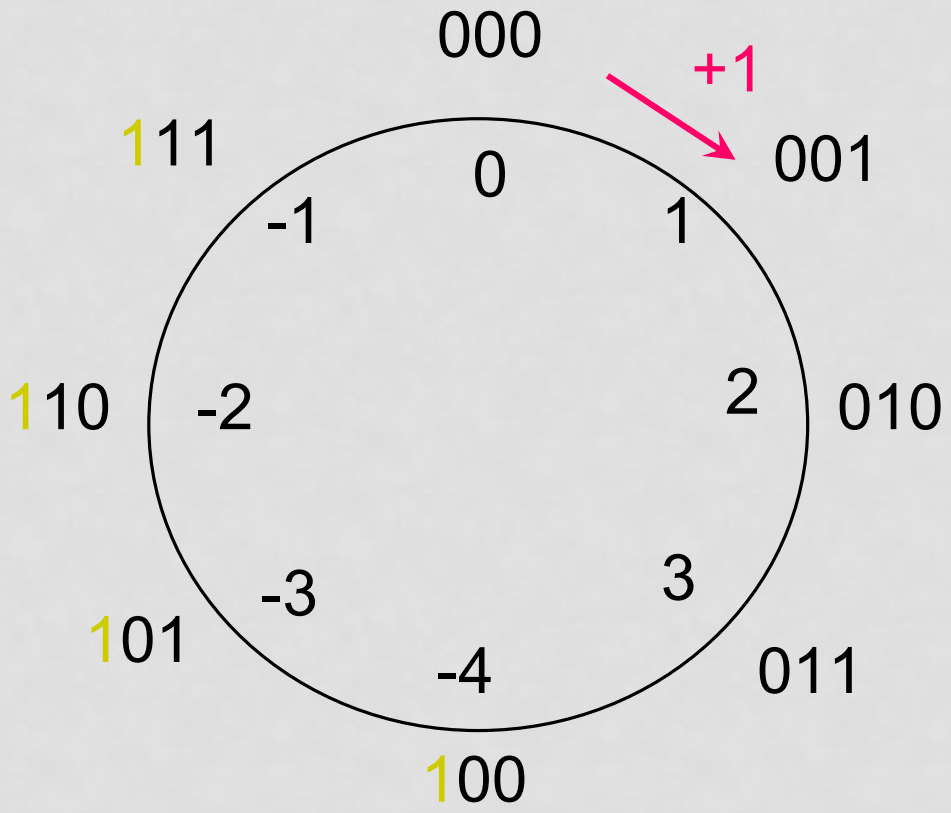
ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

x + (-x) = 0 в побитовом сложении

- Прямой код (знаковый бит)
  3: 0011
  -3: 1011
- Обратный код (~x)
  3: 0011
  -3: 1100
- Дополнительный код: x + (-x) = 0

```
   3:   0011
  -3:   1101   (обратный +1)
  Σ:   10000
```

ДИАПАЗОН ЗНАКОВЫХ ЧИСЕЛ



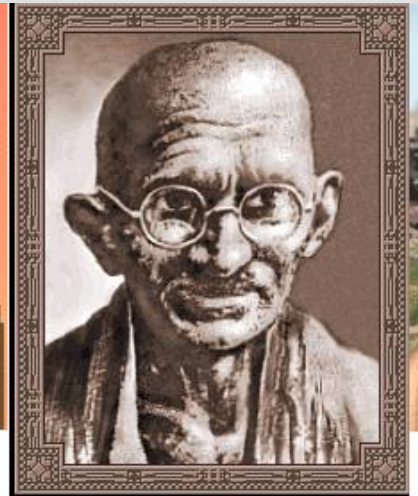
- 3 битные числа
- $2^3 = 8$ чисел
- -4 … 3

ПРЕДСТАВЛЕНИЕ ТИПОВ

- unsigned – прямой код

```
int x = -1;
unsigned u = x; // warning!
cout << u;
```

- signed –
  - прямой код для положительных
  - дополнительный код для отрицательных



ПОБИТОВЫЕ ОПЕРАЦИИ

0000 0101

```
char x = 5;
```

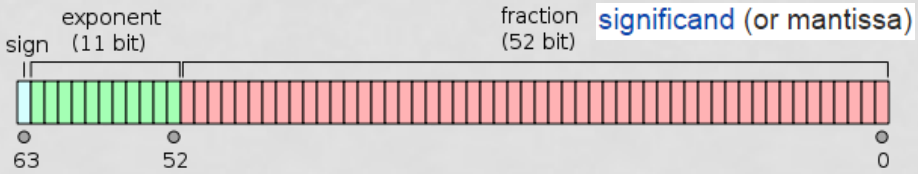
0010 1000

```
char y = x << 3;
```

0010 1101

```
char j = x | y;
```

ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ



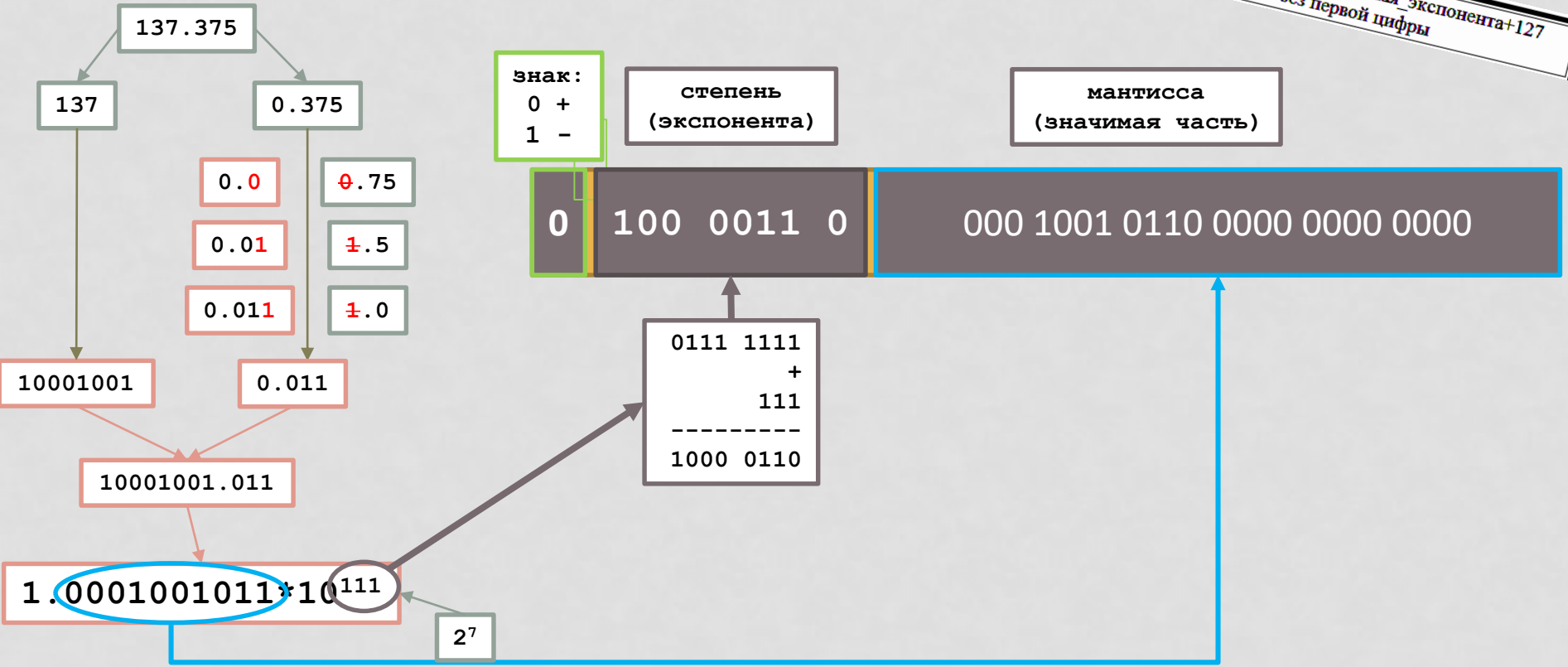
Короткое вещественное	Бит 31 – знак мантиссы. Биты 30-23 – 8-битная_экспонента+127 Биты 22-0 – 23-битная_мантисса без первой цифры
Длинное вещественное	Бит 63 – знак мантиссы. Биты 62-52 – 11-битная экспонента+1024 Биты 51-0 – 52-битная мантисса без первой цифры
Супер вещественное	Бит 79 – знак мантиссы. Биты 78-64 – 15-битная экспонента+16383 Биты 63-0 – 64-битная мантисса с первой цифрой (бит 63 равен 1)

`float` `double` `???`

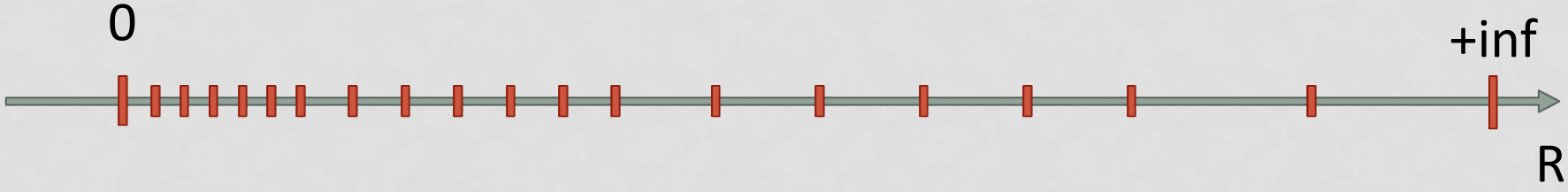
Точность	Одинарная	Двойная	Расширенная
Размер (байты)	4	8	10
Число десятичных знаков	~7.2	~15.9	~19.2
Наименьшее значение (>0), denorm	$1{,}4 \cdot 10^{-45}$	$5{,}0 \cdot 10^{-324}$	$1{,}9 \cdot 10^{-4951}$
Наименьшее значение (>0), normal	$1{,}2 \cdot 10^{-38}$	$2{,}3 \cdot 10^{-308}$	$3{,}4 \cdot 10^{-4932}$
Наибольшее значение	$3{,}4 \times 10^{+38}$	$1{,}7 \times 10^{+308}$	$1{,}1 \times 10^{+4932}$
Поля	S-E-F	S-E-F	S-E-I-F
Размеры полей	1-8-23	1-11-52	1-15-1-63

IEEE 754-2008

Короткое вещественное	Бит 31 – знак мантиссы. Биты 30-23 – 8-битная_экспонента+127 Биты 22-0 – 23-битная_мантисса без первой цифры

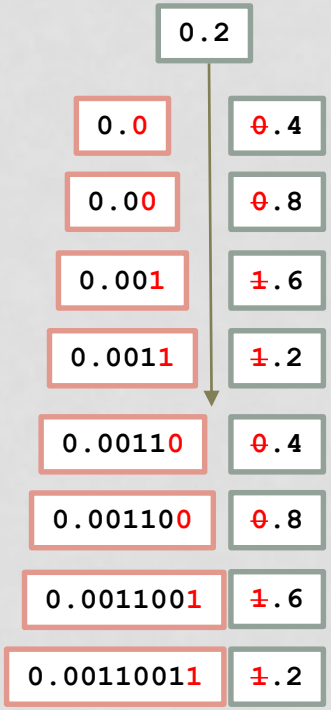


ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ И ДЕЙСТВИТЕЛЬНЫЕ ЧИСЛА



float – 2^{32} разных значения
double – 2^{64} разных значения
[0, 1] – континуум

НЕАССОЦИАТИВНОСТЬ И ОКРУГЛЕНИЕ



- переполнение float и переполнение мантиссы
- 0.2 и другие периодические дроби
- проверка равенства:

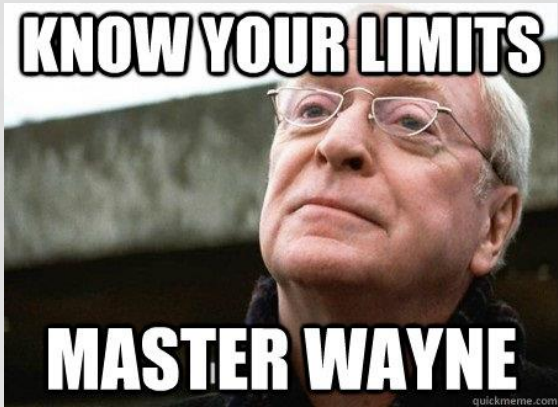
```
float a = 0.2, b = 0.4 * 0.957 / 5 / 1.914 * 5;
if (a == b) …
if (fabs(a - b) < ZERO_TOLERANCE) …
        // константа, предусмотренная вами
        //и заданная через const или #define
```

- a * b * c * d * e / f / g / h / i / j != a / f * b / g * c / h * d / i * e / j
- (a - b)*c != a*c - b*c

name	value	stands for	expresses
FLT_RADIX	2 or greater	RADIX	Base for all floating-point types (float, double and long double).
FLT_MANT_DIG DBL_MANT_DIG LDBL_MANT_DIG		MANTissa DIGits	Precision of *significand*, i.e. the number of digits that conform the *significand*.
FLT_DIG DBL_DIG LDBL_DIG	6 or greater 10 or greater 10 or greater	DIGits	Number of **decimal digits** that can be rounded into a floating-point and back without change in the number of decimal digits.
FLT_MIN_EXP DBL_MIN_EXP LDBL_MIN_EXP		MINimum EXPonent	Minimum negative integer value for the *exponent* that generates a normalized floating-point number.
FLT_MIN_10_EXP DBL_MIN_10_EXP LDBL_MIN_10_EXP	-37 or smaller -37 or smaller -37 or smaller	MINimum base-10 EXPonent	Minimum negative integer value for the *exponent* of a base-10 expression that would generate a normalized floating-point number.
FLT_MAX_EXP DBL_MAX_EXP LDBL_MAX_EXP		MAXimum EXPonent	Maximum integer value for the *exponent* that generates a normalized floating-point number.
FLT_MAX_10_EXP DBL_MAX_10_EXP LDBL_MAX_10_EXP	37 or greater 37 or greater 37 or greater	MAXimum base-10 EXPonent	Maximum integer value for the *exponent* of a base-10 expression that would generate a normalized floating-point number.
FLT_MAX DBL_MAX LDBL_MAX	1E+37 or greater 1E+37 or greater 1E+37 or greater	MAXimum	Maximum finite representable floating-point number.
FLT_EPSILON DBL_EPSILON LDBL_EPSILON	1E-5 or smaller 1E-9 or smaller 1E-9 or smaller	EPSILON	Difference between 1 and the least value greater than 1 that is representable.
FLT_MIN DBL_MIN LDBL_MIN	1E-37 or smaller 1E-37 or smaller 1E-37 or smaller	MINimum	Minimum representable floating-point number.
FLT_ROUNDS		ROUND	Rounding behavior. Possible values: -1 undetermined 0 toward zero 1 to nearest 2 toward positive infinity 3 toward negative infinity Applies to all floating-point types (float, double and long double).
FLT_EVAL_METHOD		EVALuation METHOD	Properties of the evaluation format. Possible values: -1 undetermined 0 evaluate just to the range and precision of the type 1 evaluate float and double as double, and long double as long double. 2 evaluate all as long double Other negative values indicate an implementation-defined behavior. Applies to all floating-point types (float, double and long double).
DECIMAL_DIG		DECIMAL DIGits	Number of decimal digits that can be rounded into a floating-point type and back again to the same decimal digits, without loss in precision.

Пределы

<cfloat> (float.h)



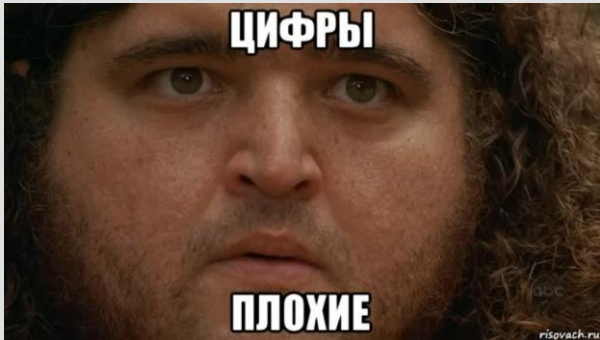
NaN

Not-a-Number

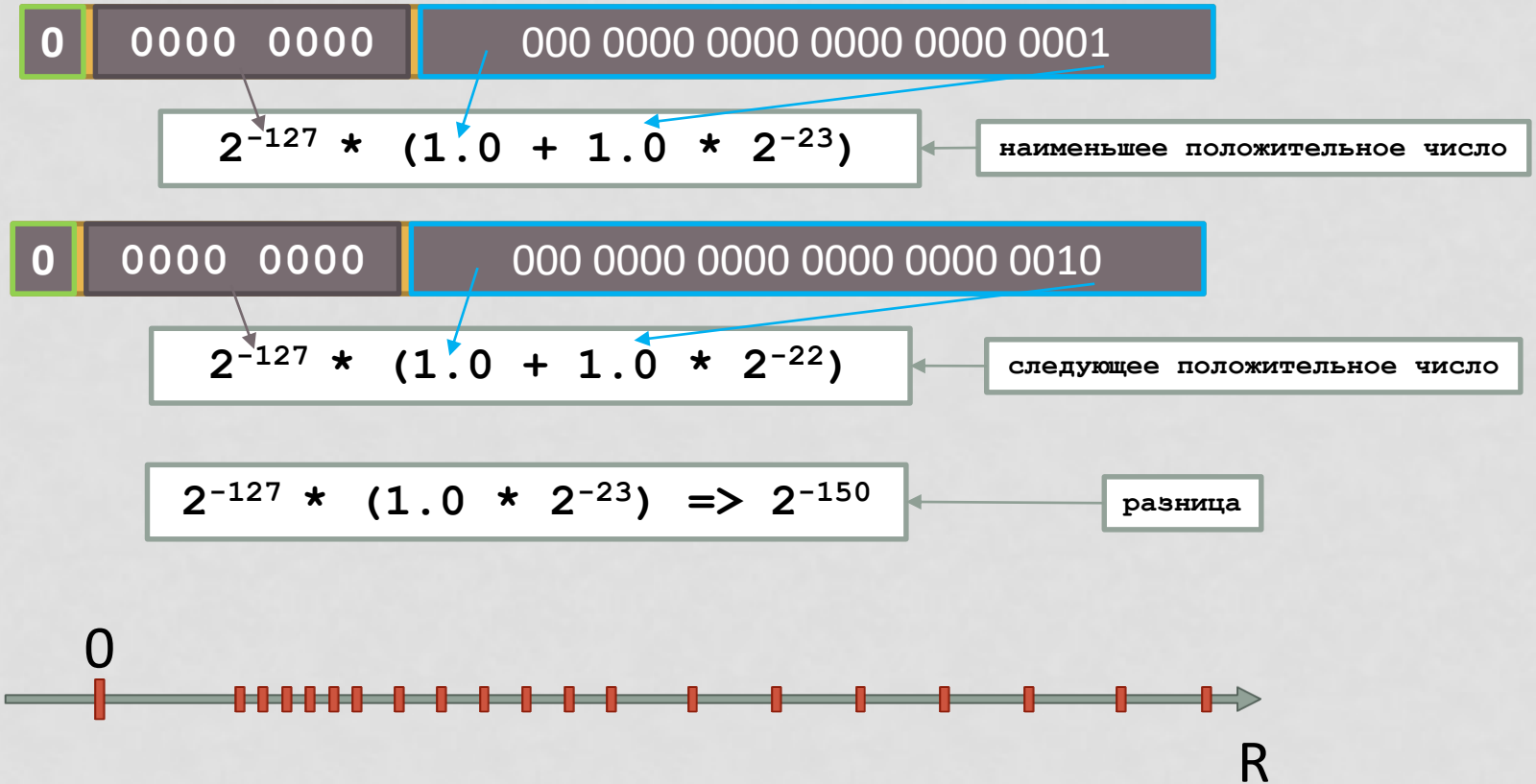
- NaN не равен ни одному другому значению (даже самому себе)
- Любая нетривиальная операция, принимающая NaN как аргумент, всегда возвращает NaN (кроме функции max и min, которые возвращают значение второго аргумента).
- Тривиальные операции, являющиеся тождеством, обрабатываются особо: так, например, 1^{NaN} равно 1.

К операциям, приводящим к появлению NaN в качестве ответа, относятся:

- все математические операции, содержащие NaN в качестве одного из операндов;
- деление ноля на ноль
- деление бесконечности на бесконечность;
- умножение ноля на бесконечность;
- сложение бесконечности с бесконечностью противоположного знака;
- вычисление квадратного корня отрицательного числа;
- логарифмирование отрицательного числа.

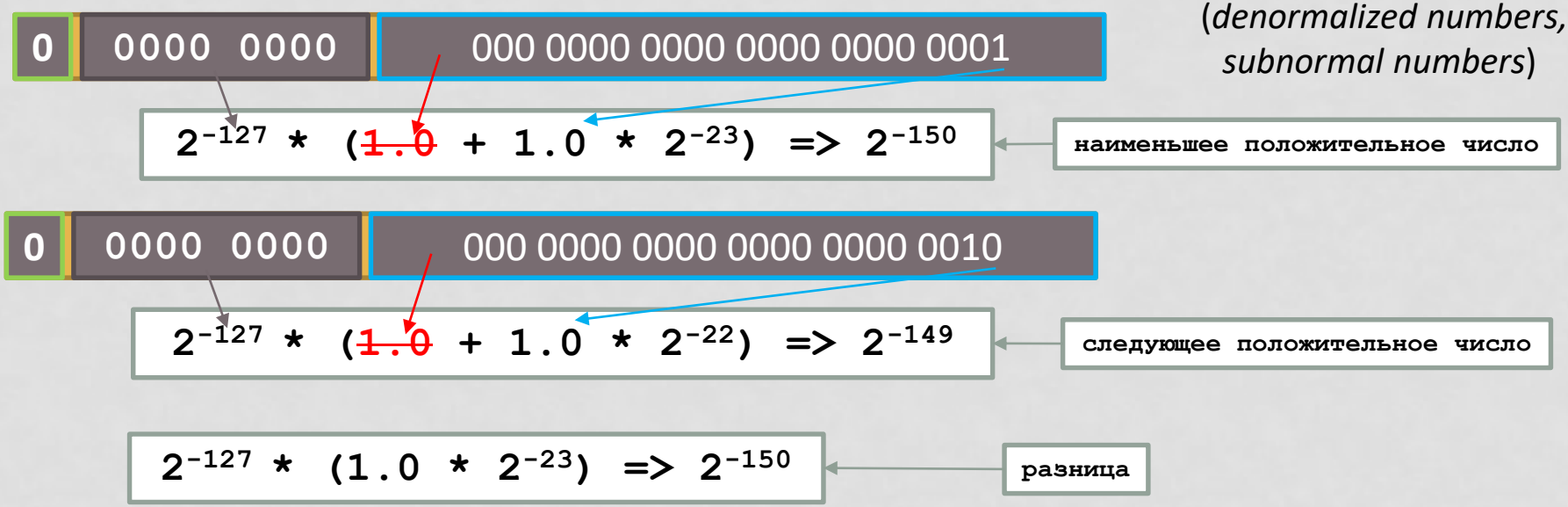


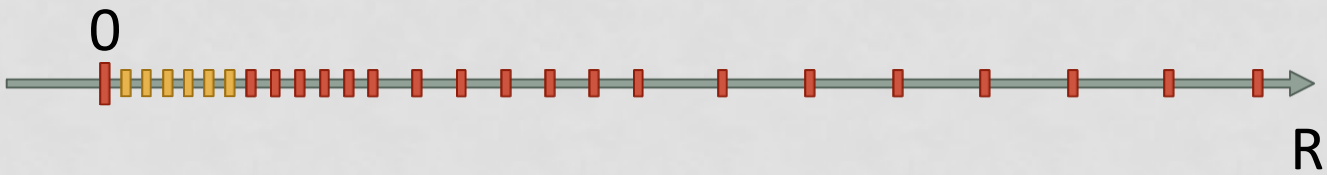
АНТИПЕРЕПОЛНЕНИЕ



ДЕНОРМАЛИЗОВАННЫЕ ЧИСЛА

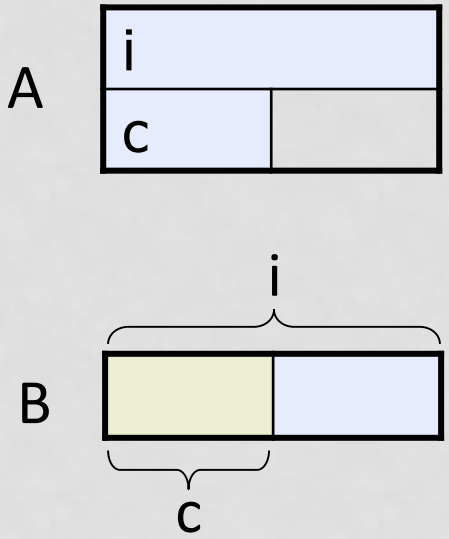
(*denormalized numbers, subnormal numbers*)

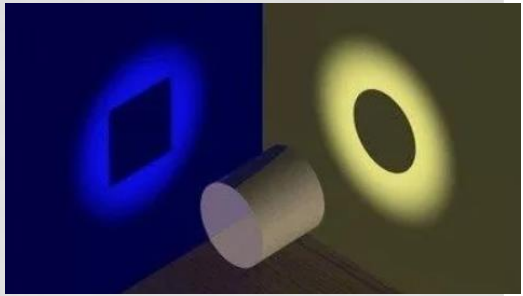


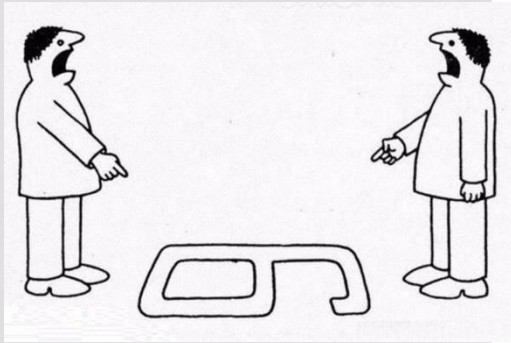


UNION

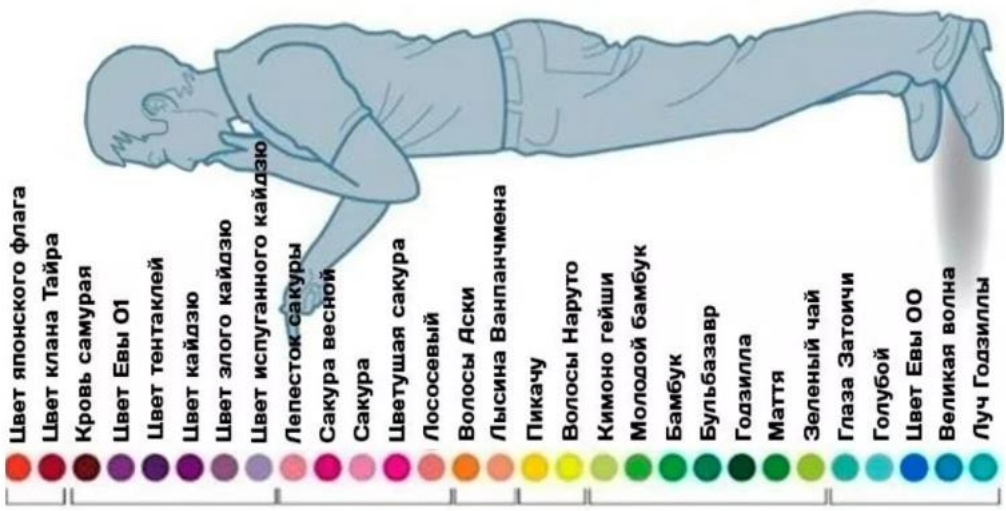
```
struct A {
    int i;
    char c;
};
union B {
    int i;
    char c;
};
```

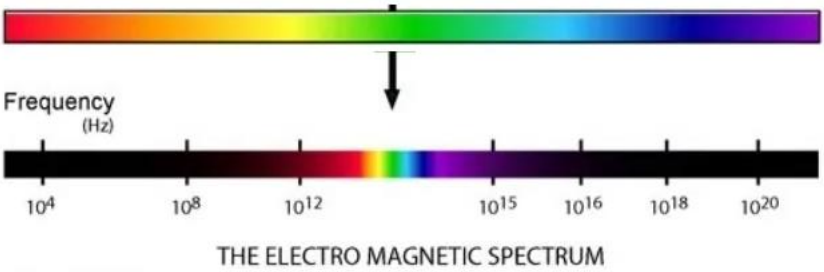






все поля начинаются
с одного места в памяти





THE ELECTRO MAGNETIC SPECTRUM