**Оглавление**

[Базовые понятия 2](#__RefHeading___Toc725_2239961777)

[Введение 2](#__RefHeading___Toc727_2239961777)

[Дополнительно 3](#__RefHeading___Toc768_3742768218)

[Сложение (Краткий алгоритм) 3](#__RefHeading___Toc729_2239961777)

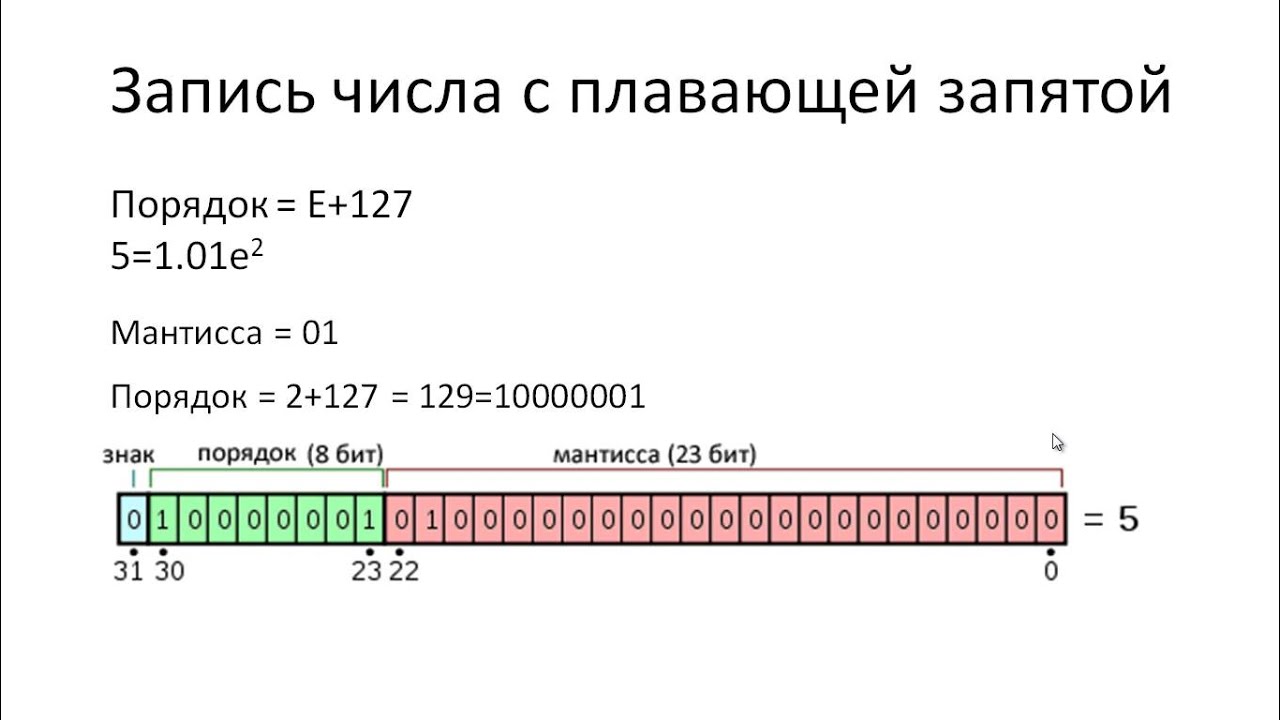
[Пример1: 5](#__RefHeading___Toc731_2239961777)

[Пример2: 5](#__RefHeading___Toc733_2239961777)

Алгоритм сложения float point чисел

# Базовые понятия

**Число с плавающей запятой** — экспоненциальная форма представления вещественных (действительных) чисел, в которой число хранится в виде мантиссы и порядка (показателя степени).



**IEEE 754** — широко используемый стандарт IEEE, описывающий формат представления чисел с плавающей точкой. Используется в программных (компиляторы разных языков программирования) и аппаратных (CPU и FPU) реализациях арифметических действий.

# Введение

Cтандарт **IEEE 754**, описывает как должны быть записаны числа с плавающей запятой и как выполнять арифметические операции над ними. Его можно найти в открытом доступе.

Оригинал документа: [IEEE Std 754™-2008 (Revision of IEEE Std 754-1985), IEEE Standard for Floating-Point Arithmetic (andrewt0301.github.io)](https://andrewt0301.github.io/hse-acos-course/part1ca/07_FP/ieee-754-2008.pdf)

Для удобного представления float point чисел, можно использовать онлайн конвертер: [IEEE-754 Floating Point Converter (h-schmidt.net)](https://www.h-schmidt.net/FloatConverter/IEEE754.html)

# Дополнительно

Есть блок-схема при необходимости смотреть в

(Блок-схема сложение двух Float Point чисел.svg)

Float point числа бывают 2-х видов:

1. Нормализованное   
   (Пример: было 121,4124 стало 1,214124 \* 10^3 или 12,14124 \* 10^2)
2. Денормализованное  
   (Пример: было 121,4124 стало 0,1214124 \* 10^4)

Если на вход или выход поступают Nan или inf, то выводить не значение, а символы Nan или inf

inf получается если exp полностью заполнена, а мантисса нулевая 0/1 11111111 000...00

Nan получается если exp полностью заполнена и есть хоть 1 бит в мантиссе 0/1 11111111 010.

# Сложение (Краткий алгоритм)

1. Подготовка
   1. Разделяем каждое число на экспоненту (exp), мантиссу (mant) и знак каждого числа.
   2. Расширяем обе mant вправо на количество длинны одной mant. Та часть, которую мы расширили это хвост(tail), длина tail не меньше длинны мантиссы.
   3. Добавить два бита скрытой единицы (01) в начало обоих mant
2. Найти разницу exp
3. Анализ разницы exp
   1. Если разница больше длинны mant, то в ответ записывается наибольшее число
   2. Если разница exp !=0
      1. mant с наименьшей exp сдвигается в право на разницу exp
      2. То, что выпадает после сдвига, должно попасть в tail
      3. В переменную otv\_exp записывается значение наибольшей exp
   3. Если разница 0, то переходим в пункт (5)
4. Если знаки чисел совпадают
   1. переходим в пункт (5).

Если знаки не совпадают:

* + 1. Найти наименьшее mant по модулю и инвертируем его вместе с tail
    2. Выполняем пункт (5)
    3. Добавляем +1 к tail
    4. Переходим к пункту (6).

1. Два числа складываются (хвосты тоже) получаем otv\_mant и otv\_tail
2. Нормализация:

Если денормализованное:

* 1. Если лишняя единица слева: сдвиг otv\_mant в право на 1

* 1. +1 к otv\_exp.
  2. Сдвигать otv\_mant в лево на 1 и -1 к otv\_exp повторять до тех пор, пока otv\_exp >0 или otv\_mant не станет нормализованным.

1. Округляем:  
   Условия округления
   * 1. Если последний бит в otv\_mant =1 и первый бит в tail = 1
     2. Если последний бит в otv\_mant =0 и первый бит в tail =1 и есть еще хоть один бит в tail.
2. Убираем скрытый бит.
3. Соединяем знак, otv\_exp и otv\_mant

# Пример1:

Есть 2 числа   
ch1=0b 0 11011101 11101110111010100001111 = 0x 2ef7750f  
ch2=0b 0 11100100 10100011100101110010101 = 0x 2251cb95

Sign1=0 Sign2=0

exp1= 11011101 = 94  
exp2= 11100100 = 101

Mant1= 11101110111010100001111  
Mant2= 10100011100101110010101

Mant1= 01 111 0111 0111 0101 0000 1111 -> мнимая единица  
Mant2= 01 101 0001 1100 1011 1001 0101 -> мнимая единица

exp2-exp1=7-> разница exp

Mant1>>7= 000000011110111011101010 tail1= 00011110000000000000000  
Mant2 = 110100011100101110010101 tail2= 00000000000000000000000

Otv\_exp=exp2

Mant1+Mant2=Otv\_mant= 110100111011101001111111  
Tail1+Tail2= Otv\_tail= 00011110000000000000000

Нормализация -  
Округление -

Otv= 0 11100100 10100111011101001111111 0x7253ba7f

# Пример2:

Ch1= 1 10100101 00110111010101011010111 0xd29baad7

Ch2= 0 10110010 10111111010101011000011 0x595faac3

Sign1=1 Sign2=0

exp1= 10100101 = 38 (165-127=38)  
exp2= 10110010 = 51 (178-127=51)

Mant1= 01 001 1011 1010 1010 1101 0111 -> мнимая 01  
Mant2= 01 101 1111 1010 1010 1100 0011

Exp2-Exp1=13 -> разница exp

Otv\_exp= exp2=51

Mant1>>13= 000000000000010011011101 tail= 01010110101110000000000

Так как знаки разные находим наименьшее по модулю  
наименьшее Ch1. Инвертируем его

~Mant1 = (11)111 1111 1111 1011 0010 0010 ~tail1=10101001010001111111111

Otv\_mant=mant1+mant2

Otv\_mant= 0 1101 1111 1010 0101 1110 0101 otv\_tail= 10101001010001111111111  
Прибавляем +1 к otv\_tail получаем 10101001010010000000000

Округляем так как последний бит Otv\_mant=0, а первый в otv\_tail =1

Otv\_mant+1= 110111111010010111100110 otv\_tail=000000000000000000000000

Убираем мнимую единицу Otv\_mant= 10111111010010111100110

Otv=0 10110010 10111111010010111100110 0x595fa5e6