



# Форматы представления данных в компьютере

---

Для вопросов по курсу:

[natalya.shevskaya@moevm.info](mailto:natalya.shevskaya@moevm.info)

Префикс в теме письма [CS\_23XX]

*Шевская Наталья Владимировна  
СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ*

# Представление целых чисел

---



# Позиционные системы счисления

- Десятичная система счисления

$$56789_{10} = 5 * 10^4 + 6 * 10^3 + 7 * 10^2 + 8 * 10^1 + 9 * 10^0$$

- Двоичная система счисления

$$10011_2 = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

- Восьмеричная система счисления

$$567_8 = 5 * 8^2 + 6 * 8^1 + 7 * 8^0$$

- Шестнадцатеричная система счисления

$$56A8C_{16} = 5 * 16^4 + 6 * 16^3 + A * 16^2 + 8 * 16^1 + C * 16^0$$



# Вспомогательные функции Python

```
>>> b = 127
```

```
>>> b.bit_length()
```

```
....
```

```
>>> bin(127)
```

```
....
```

```
>>> "{:b}".format(127)
```

```
>>> int(0b10010)
```

```
....
```

```
>>> 0b11101000 + 0b10010
```

```
....
```

```
>>> '{:b}'.format(0b11101000 >> 2)
```

```
....
```



# Формат представления чисел на компьютере

Числа конечной точности - числа, представляемые в фиксированном количестве разрядов.

Арифметические операции с числами конечной точности имеют ограничения и могут вызвать **переполнение**.

Переполнение -- ситуация когда результат операции не помещается в наперед заданное количество разрядов.



# Формат представления **беззнаковых** целых чисел

123:

7	6	5	4	3	2	1	0
0	1	1	1	1	0	1	1

Диапазон значений:

$0 \dots 2^n - 1$ ,

где  $n$  - разрядность архитектуры

Для  $n = 8$ :

$0 \dots 2^8 - 1 \Rightarrow 0 \dots 255$

42:

7	6	5	4	3	2	1	0
0	0	1	0	1	0	1	0

128:

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0

255:

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

305:

7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1

**ПЕРЕПОЛНЕНИЕ**



# Формат представления **знаковых** целых чисел

$n = 8$

➤ Прямой код  $(-2^{n-1} + 1 \dots 2^{n-1} - 1)$   $(-2^{8-1} + 1 \dots 2^{8-1} - 1) \Rightarrow (-2^7 + 1 \dots 2^7 - 1) \Rightarrow$   
 $(-128 + 1 \dots 128 - 1) \Rightarrow \text{(-127 ... 127)}$

-123: 

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

➤ Обратный код  $(-2^{n-1} + 1 \dots 2^{n-1} - 1)$   $(-2^{8-1} + 1 \dots 2^{8-1} - 1) \Rightarrow \dots \Rightarrow \text{(-127 ... 127)}$

-123: 

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

➤ Доп. код  $(-2^{n-1} \dots 2^{n-1} - 1)$   $(-2^{8-1} \dots 2^{8-1} - 1) \Rightarrow (-2^7 \dots 2^7 - 1) \Rightarrow \text{(-128 ... 127)}$

-123: 

1	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---



## Дополнительный код -- единственный ноль

В 10-СС	В 2-СС	Инвертируем	Выделяем знаковый бит	Дополняем до двух	В 10-СС и доп. кода
0	000	111	1 11	0 00	0
1	001	110	1 10	1 11	-1
2	010	101	1 01	1 10	-2
3	011	100	1 00	1 01	-3
4	100	011	0 11	1 00	-4
5	101	010	0 10	0 11	3
6	110	001	0 01	0 10	2
7	111	000	0 00	0 01	1





# Примеры диапазонов

Разрядность	Диапазон беззнаковых чисел	Диапазон знаковых чисел (дополнительный код)
8	От 0 до 255	От -128 до 127
32	От 0 до 4 294 967 295	От -2 147 483 648 до 2 147 483 647
64	От 0 до 18 446 744 073 709 551 615	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

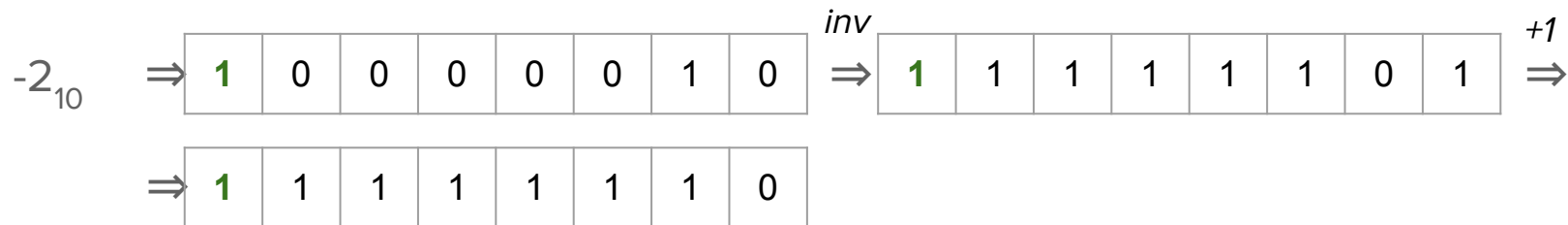
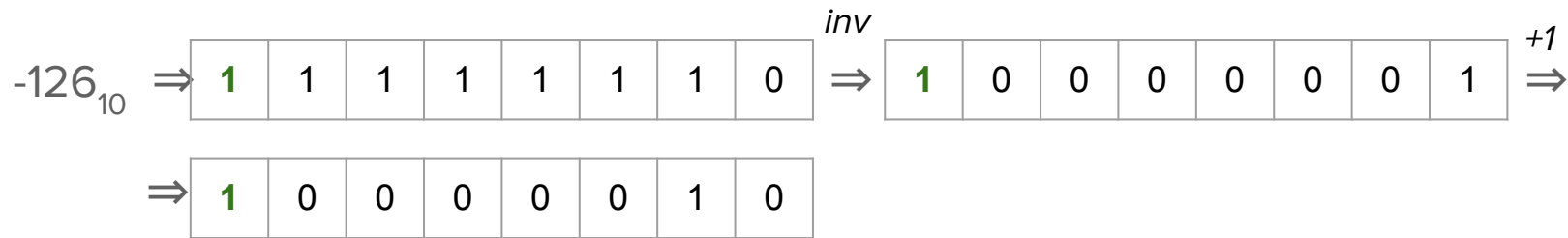


# Прямой, обратный и дополнительный код

Пример для 8 бит	Прямой	Обратный	Дополнительный
123	0 1 1 1 1 0 1 1	0 1 1 1 1 0 1 1	0 1 1 1 1 0 1 1
-123	1 1 1 1 1 0 1 1	1 0 0 0 0 1 0 0	1 0 0 0 0 1 0 1
-42	1 0 1 0 1 0 1 0	1 1 0 1 0 1 0 1	1 1 0 1 0 1 1 0
-127	1 1 1 1 1 1 1 1	1 0 0 0 0 0 0 0	1 0 0 0 0 0 0 1
-128	impossible	impossible	1 0 0 0 0 0 0 0



## Примеры с доп. кодом: $-126 - 2 = -128$





Примеры с доп. кодом:  $-126 - 2 = -128$

$-126_{10}$

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

$-2_{10}$

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

---

$-128_{10}$

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---





Примеры с доп. кодом:  $126 - 2 = 124$

$126_{10}$

0	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

+

$-2_{10}$

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

---

$124_{10}$

0	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---





# Примеры с доп. кодом: $-126 + 2 = -124$

$-126_{10}$

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

$2_{10}$

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

$-124_{10}$

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

из доп. в прямой код:

1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

inv

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

+1

1	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---



чтобы проверить



# Примеры с доп. кодом: $-126 - 2 = -128$

$-126_{10}$

1	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

+

$-2_{10}$

1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---

$-128_{10}$

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

чтобы проверить

из доп. в прямой код:

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

inv

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

+1

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---



# Представление чисел с плавающей точкой

---





# Формат представления **чисел с плавающей точкой**

Стандарт IEEE 754:

- **одинарная точность** (single precision) - 4 байта.  
Пример: float в C  
примерно от  $10^{-38}$  до  $10^{38}$
- **двойная точность** (double precision) - 8 байт.  
Примеры: double в C, float в Python  
примерно от  $10^{-308}$  до  $10^{308}$



# Одинарная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 8 бит - порядок
- 23 бита - дробная значащая часть числа - мантисса
- 127 - смещение

$$111,1101 = 1,111101 * 2^2$$

1,111101 - мантисса, записывается только дробная часть

2 - *истинный* порядок, 129 - *смещенный* порядок

знак	порядок								мантисса									
0	1	0	0	0	0	0	0	1	1	1	1	1	0	1	...	0	0	0



# Одинарная точность

Специальные случаи:

- Если порядок и мантисса равны 0, число равно 0.
- Если порядок равен 255 и мантисса равна 0, число в зависимости от знака  $-\infty$  или  $+\infty$ .
- Если порядок равен 255 и мантисса не равна 0, значение считается недопустимым числом и является NaN (Not a Number).

знак	порядок								мантисса						
0	0	1	0	0	0	0	0	1	1	1	1	...	0	0	0



# Двойная точность

- 1 бит - знак (0 - положительные числа, 1 - отрицательные)
- 11 бит - порядок
- 52 бита - дробная значащая часть числа - мантисса
- 1023 - смещение

$$111,1101 = 1,111101 * 2^2$$

1,111101 - мантисса

2 - *истинный* порядок, 1025 - *смещенный* порядок

знак	порядок							мантисса						
0	0	1	0	...	0	0	1	1	1	1	...	0	0	0



## Пример 446.15625 в одинарной точности

$446,15625_{10} \Rightarrow$  *отдельно обрабатываем целую часть*

$446_{10} \Rightarrow 110111110_2$  *ставим запятую и начинаем думать про хвост*

*не понимаем, какой хвост?*

$\Rightarrow 110111110,?????...$   $\Rightarrow$  *сначала выделим порядок*

*порядок: 8 знаков влево*  $\Rightarrow 1,10111110$  ???????????????? *хвост из 15 знаков*

собираем хвост....



# 0.15625 в бинарном виде

0,15625	* 2	=	0,3125	⇒	1,10111110 0???????????????
0,3125	* 2	=	0,625	⇒	1,10111110 00???????????????
0,625	* 2	=	1,25	-1 ⇒	1,10111110 001???????????????
0,25	* 2	=	0,5	⇒	1,10111110 0010???????????????
0,5	* 2	=	1,0	-1 ⇒	1,10111110 00101???????????????

Где взять все остальные знаки? Забить нулями!

1,10111110 0010100000000000

превратим в знак | порядок | мантисса




# 446.15625 в памяти компьютера

Знак: 0

Порядок: истинный 8 + смещение  $127 = 135_{10} \Rightarrow 1000\ 011_2$

Мантисса: 1,101111100010100000000000

Итого:

0 | 1000 0111 | 101111100010100000000000 

---

Двойная точность:

Знак: 0

Порядок: истинный 8 + смещение  $1023 = 1031_{10} \Rightarrow 100\ 0000\ 011_2$

Мантисса: 1,101111100010100000000000 + 29 нулей



# Собираем число обратно в 10-СС

1 | 1000 0111 | 110 0011 1011 0000 0000 0000

Знак: 1 -- отрицательное

Порядок:  $1000\ 0111_2 \Rightarrow 135_{10}$  (смещенный)  $\Rightarrow 135 - 127 = 8$  (истинный)

Мантисса:

110 0011 1011 0000 0000 0000 *вспоминаем про единицу*

1,110 0011 1011 0000 0000 0000 *двигаем запятую на 8 знаков вправо*

1110 0011 1,011 0000 0000 0000

1 1100 011,011 0000 0000 0000 *извлекаем целую часть и переводим обычным образом*

$1\ 1100\ 011_2 \Rightarrow 455_{10}$

переводим хвост привычным способом:






# Переводим 0,011 в 10-СС

0,011 0000 0000 0000

0	,	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
0	,	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

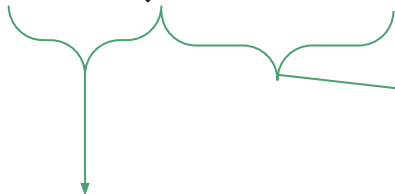
$$= 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 0 \cdot 2^{-5} + \dots + 0 \cdot 2^{-16} = 0 + \frac{1}{4} + \frac{1}{8} + 0 + \dots + 0 = 0,375$$

Соединяем “голову” и хвост, припоминая про отрицательный знак:

- 455,375 



446,15625



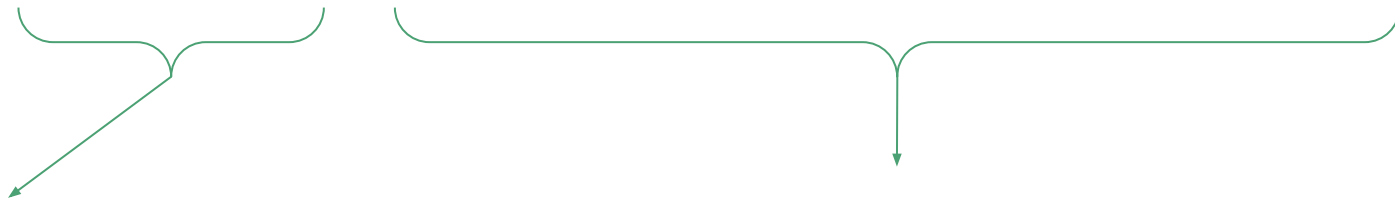
переводим 2-СС делением на 2  
собираем **остатки в обратном порядке**



переводим в 2-СС умножением на 2  
собираем **целые части в прямом порядке**



1 | 1000 0111 | 110 0011 1011 0000 0000 0000



1) вычисляем истинный порядок

2) идем в мантиссу и *выделяем* оттуда целую часть 10-го числа

целая часть

дробная часть

3) переводим двоичное представление целого в 10-СС

4) двоичный “хвост” переводим в 10-СС с отрицательными степенями

5) собираем число из знака, целой и вещественной части



# Чем двойная точность отличается от одинарной?

Знак -- тот же самый (1 бит)

Порядок -- вместо 8 работаем с **11** битами  
(смещение из-за этого равно не 127, а **1023**)

Мантисса -- вместо 23 бита работаем с **52** битами  
(в худшем случае при переводе дробной части выполним на 29 итераций больше)



# Пример с “плохим” хвостом числа

0,8919

Какая целая часть у этого числа?

Какой **порядок** у этого числа?

**Сколько итераций** для вычисления **мантиссы** придется сделать?



0,8919 переводим в 2-СС

$$1) 0,8919 * 2 = 1,7838 \implies 0,1 \quad (-1)$$

$$2) 0,7838 * 2 = 1,5676 \implies 0,11 \quad (-1)$$

$$3) 0,5676 * 2 = 1,1352 \implies 0,111 \quad (-1)$$

$$4) 0,1352 * 2 = 0,2704 \implies 0,1110$$

$$5) 0,2704 * 2 = 0,5408 \implies 0,11100$$

$$6) 0,5408 * 2 = 1,0816 \implies 0,111001 \quad (-1)$$

....



Мантисса и порядок для числа 0,8919

$$0,111001?.. = 1,11001?.. * 2^{(-1)}$$

....

$$0,0816 * 2 = ....$$

---

истинный -1 ==> -1 + 127 = 126 ==> 0111 1110

0 | 0111 1110 | 11001?

сколько итераций нужно сделать еще?

$$23-5 = 18$$

Для вопросов по курсу:  
`natalya.shevskaya@moevm.info`

---

Префикс в теме письма [CS\_23XX]