



# ОСНОВЫ ПРОГРАММИРОВАНИЯ

## Раздел 2. Математические основы программирования

### *Представление чисел в ЭВМ*

Старший преподаватель  
Кафедры ВС, к.т.н.  
Поляков Артем Юрьевич



# Рассматриваемые вопросы

- Представление целых чисел
  - представление целых без знака
  - представление знаковых целых
- Представление вещественных чисел
  - математическая основа представления вещественных чисел
  - современный стандарт представления вещественных чисел



# Представление чисел в ЭВМ

## Особенности представления чисел в ЭВМ:

- используется двоичная система счисления
- ограниченное количество разрядов
- в ЭВМ используется память, элементарная ячейка которой (бит) может иметь только 2 состояния (0 и 1). Поэтому отсутствует возможность хранить знаки: ( "+", "-" и "." ).



# Представление целых чисел без знака

- Пусть дана ячейка памяти, размер которой  $k$  бит.
- Существует  $2^k$  всевозможных битовых наборов длины  $k$ .
- Следовательно, всего возможно представить  $2^k$  различных чисел.
- Существует всего  $(2^k)!$  (количество перестановок из  $2^k$  элементов) способов закодировать беззнаковые числа битовыми наборами.
- Среди всех теоретически возможных способов **наиболее удобно использовать  $k$ -разрядную запись этого числа в двоичной системе счисления.** Это позволяет естественным образом реализовать арифметические операции над числами.



# Представление целых чисел без знака (примеры)

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

$$175_{10} = 1010\ 1111_2$$

$$36_{10} = 10\ 0100_2$$



# Представление знаковых целых

Отрицательные числа могут быть представлены в ЭВМ несколькими способами, выбор одного из которых оказывает влияние на реализацию арифметических операций.

Для представления знаковых целых чисел используются три способа:

- 1) прямой код;
- 2) обратный код;
- 3) дополнительный код.

**Далее будут рассмотрены каждый из способов для десятичной и двоичной систем счисления (СС).**



# Прямой код (десятичная СС, 6 разрядов)

Это наиболее очевидный и близкий к естественному способ записи чисел.

Число формируется из одного знака и фиксированного количества разрядов (цифр), представляющих собой модуль числа. Например:

+123 456

-001 000

-012 123



# Прямой код (двоичная СС, 8 бит)

При формировании прямого кода в двоичной СС старший бит (разряд) используется для хранения знака: 0 ~ "+", 1 ~ "-".

	0	1	2	3	4	5	6	7
<b>-100</b>	0	0	1	0	0	1	1	1
<b>-36</b>	0	0	1	0	0	1	0	1

**знак**

$$\begin{aligned} -100_{10} &= 1110\ 0100_2, & 100_{10} &= 0110\ 0100_2 \\ -36_{10} &= 1010\ 0100_2, & 36_{10} &= 10\ 0100_2 \end{aligned}$$



# Обратный код (десятичная СС, 6 разрядов)

Отрицательные числа также состоят из фиксированного числа разрядов и формируются путем дополнения каждого разряда до 9:

$$- 123\ 456 \leftrightarrow 876\ 543$$

$$1 \leftrightarrow 9 - 1 = 8$$

$$2 \leftrightarrow 9 - 2 = 7$$

$$3 \leftrightarrow 9 - 3 = 6$$

$$4 \leftrightarrow 9 - 4 = 5$$

$$5 \leftrightarrow 9 - 5 = 4$$

$$6 \leftrightarrow 9 - 6 = 3$$

$$- 123\ 456 = \mathbf{999\ 999} - \mathbf{123\ 456} = \mathbf{876\ 543}$$



# Обратный код (двоичная СС, 8 бит)

Обратный код отрицательного числа в двоичной СС формируется путем инвертирования каждого разряда, включая знак, на противоположный.

	0	1	2	3	4	5	6	7
<b>- 100</b>	1	1	0	1	1	0	0	1
<b>- 36</b>	1	1	0	1	1	0	1	1

**знак**

$$100_{10} = 0110\ 0100_2, \quad -100_{10} = 1001\ 1011_2$$
$$36_{10} = 10\ 0100_2, \quad -36_{10} = 1101\ 1011_2$$



# Обратный код (двоичная СС, 8 бит)

Для формирования обратного кода также может использоваться подход, аналогичный приведенному для десятичной СС:

$$- 100_{10} = \mathbf{1111\ 1111}_2 - \mathbf{0110\ 0100}_2 = \mathbf{1001\ 1011}_2$$



# Дополнительный код (десятичная СС, 6 разрядов)

Отрицательные числа в дополнительном коде представляются следующим образом:

$$-x = 0 - x = 1\,000\,000 - x$$

$$-1 = 0 - 1 = 1\,000\,000 - 1 = 999\,999$$

$$-500\,000 = 0 - 500\,000 =$$

$$1\,000\,000 - 500\,000 = 500\,000$$

Диапазон: -500 000 до 499 999

$$(499\,999 - 500\,000)$$



# Связь дополнительного и обратного кода (дес. СС, 6 разрядов)

$$\langle \text{Доп. код} \rangle = \langle \text{Обр. код} \rangle + 1$$

Дополнительный код

$$-x = 0 - x = 1\,000\,000 - x$$

$$\begin{aligned} -1 &= 0 - 1 = 1\,000\,000 - 1 \\ &= 999\,999 \end{aligned}$$

Обратный код

$$-x = 0 - x = 999\,999 - x$$

$$\begin{aligned} -1 &= 0 - 1 = 999\,999 - 1 \\ &= 999\,998 \end{aligned}$$



# Дополнительный код (двоичная СС, 8 бит)

	0	1	2	3	4	5	6	7
<b>- 100</b>	0	0	1	1	1	0	0	1
<b>- 36</b>	0	0	1	1	1	0	1	1

Знак

$$\begin{aligned} \underline{100}_{10} &= 0110\ 0100_2, \quad \underline{36}_{10} = 10\ 0100_2 \\ -100_{10} &= \mathbf{1}\ 0000\ 0000_2 - 0110\ 0100_2 = \\ &10011100_2 = 156_{10} \\ -36_{10} &= \mathbf{1}\ 0000\ 0000_2 - 0010\ 0100_2 = \\ &11011100_2 = 220_{10} \end{aligned}$$



# Сравнение различных представлений знаковых чисел (двоичная СС, 3 бит)

Битовый набор	Без знака	Прямой код	Обратный код	Доп. код
000	0	+0	+0	0
001	1	+1	+1	+1
010	2	+2	+2	+2
011	3	+3	+3	+3
<b>100</b>	<b>4</b>	<b>-0</b>	<b>-3</b>	<b>-4</b>
101	5	-1	-2	-3
110	6	-2	-1	-2
<b>111</b>	<b>7</b>	<b>-3</b>	<b>-0</b>	<b>-1</b>



# Сложение и вычитание целых чисел без знака

Сложение и вычитание  $k$ -разрядных целых чисел без знака происходит по обычным для позиционных систем счисления правилам. Примеры (для  $k = 3$ ):

$$001_2 + 100_2 = 101_2;$$

$$101_2 - 010_2 = 011_2.$$

Ситуации, когда **уменьшаемое меньше вычитаемого** или когда **результат суммы не уместится в  $k$  разрядов**, считаются ошибочными и должны отслеживаться устройством компьютера.



# Обработка переполнения при сложении целых чисел без знака

Если при сложении двух  $k$ -разрядных чисел возникает  $(k+1)$ -й разряд, то он будет отброшен.  
Пример (для  $k = 3$ ):

$$\begin{aligned}5_{10} + 4_{10} &= \\101_2 + 100_2 &= \mathbf{1001}_2 = \\001_2 &= 1_{10}\end{aligned}$$



# Сложение целых знаковых чисел в дополнительном коде

- Сложение  $k$ -разрядных знаковых чисел производится аналогично алгоритму для целых чисел без знака.
- Складываются все разряды, включая знаковый
- Если в результате сложения возникает  $(k+1)$ -й разряд, то он отбрасывается.

Для  $k = 3$ :

$$+3_{10} + (-1_{10}) = 011_2 + 111_2 = 1010_2 \Rightarrow \\ 010_2 = +2_{10}.$$



# Вычитание целых знаковых чисел в дополнительном коде

При вычитании также используется аналогичный алгоритм, однако если уменьшаемое меньше вычитаемого, к двоичному коду уменьшаемого слева приписывается единица (т.е. добавляется  $2^k$ ) и только после этого производится вычитание (такой способ называется вычитание по модулю  $2^k$ ).

Для  $k = 3$ :

$$1_{10} - 3_{10} = 001_2 - 011_2 \Rightarrow$$
$$1001_2 - 011_2 = 110_2 = -2_{10}.$$



# Сложение и вычитание по модулю $2^k$

$$(x + y) \bmod 2^k = \begin{cases} x + y, & \text{если } (x + y) < 2^k \\ x + y - 2^k, & \text{если } (x + y) \geq 2^k \end{cases}$$

$$(x - y) \bmod 2^k = \begin{cases} x - y, & \text{если } x \geq y \\ 2^k + x - y, & \text{если } x < y \end{cases}$$



# Вещественные числа

- В памяти ЭВМ не предусмотрено специальных средств для запоминания десятичной точки.
- Существует два способа представить вещественное число в виде десятичной дроби:
  1. Явно указав позицию запятой внутри числа:  
**23.456, 0.0098;**
  2. Научный, с помощью степени основания 10:  
 **$2.3456 \cdot 10^1, 0.98 \cdot 10^{-2}$**



# Вещественные числа с фиксированной точкой

Данное представление соответствует первому варианту записи вещественных чисел.

Положение точки внутри байта (байтов) задается фиксировано, например: для хранения используется 1 байт, пусть биты с 0 по 4 – целая часть, а биты с 5 по 7 – дробная часть:

0000 0.000

0000 0.001

0000 0.010

.....

1111 1.110

1111 1.111



# Вещественные числа с фиксированной точкой (недостатки)

- 1) Малый диапазон значений.
- 2) Неэффективное использование памяти при работе только с малыми числами или только с большими числами.



# Вещественные числа с плавающей точкой

$p$ -разрядным числом с плавающей точкой по основанию  $b$  с избытком  $q$  называется пара величин  $(e, f)$ , которой соответствует значение:

$$(e, f) = f \cdot b^{(e - q)}$$

$e$  – порядок – беззнаковое целое число, изменяющаяся в определенном промежутке.

$f$  – мантисса – знаковое вещественное число с фиксированной точкой, при этом:

$$|f| < 1,$$

т.е. разделяющая точка находится в крайней слева позиции



# Научная форма записи вещественных чисел

Научная форма записи вещественных чисел позволяет представить одно и то же число множеством различных способов.

Например, постоянная Планка  $h = 6.6261 \cdot 10^{-27}$  может быть записана одним из следующих способов:

- 1)  $66.261 \cdot 10^{-28}$
- 2)  $66261 \cdot 10^{-31}$
- 3)  $0.66261 \cdot 10^{-26}$
- 4)  $0.00066261 \cdot 10^{-23}$
- 5) ...



# Нормализованная форма

Представление информации в ЭВМ требует определенности, поэтому вещественные числа представляются в **нормализованной форме**.

Число с плавающей точкой является **нормализованным**, если:

1) наиболее значимая цифра в представлении  $f$  отлична от нуля:

$$1/b \leq |f| < 1$$

2)  $f = 0$  и  $e$  принимает наименьшее возможное значение

**Например:** 0.615, 0.101, ~~6.15~~, ~~0.01~~



# $p$ -разрядные нормализованные вещественные числа

$$(e, f) = f \cdot b^{e-q}$$

В ЭВМ для хранения вещественных чисел отводится **ограниченное число разрядов**:

$$-b^p \leq f \cdot b^{e-q} \leq b^p$$

**Пример:** рассмотрим следующий пример представления десятичных чисел:  $f$  – 8-разрядов,  $e$  – 2 разряда, избыток  $q = 50$ , основание  $b = 10$ .

**Число Авогадро:**  $N = 6,02214 \cdot 10^{23} = 0.602214 \cdot 10^{24} =$   
 $((24+50), +.60221400) = (74, +.60221400)$

**Постоян. Планка:**  $h = 6.6261 \cdot 10^{-27} = 0.66261 \cdot 10^{-26}$   
=

$((-26+50), +.66261000) = (24, +.66261000)$



# Упаковка вещественных числа с одинарной точностью (тип float)

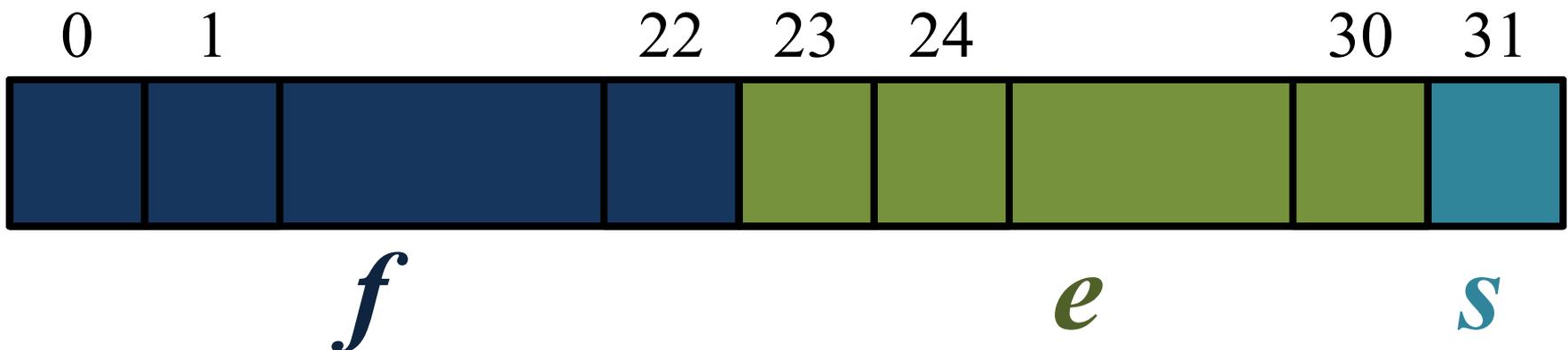
**Размер, выделяемый для хранения:**

4 байта (32 бита)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

$f$ : 23 бит;  $e$ : 8 бит;  $s$ : 1 бит,

$$b = 2, q = 2^{8-1} - 1 = 127$$





# Примеры вещественных чисел с одинарной точностью

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	1				
Двоичное представление	$1 \cdot 2^0$				
Результат упаковки (hex)	3f 80 00 00				
Двоичное представление	<table border="1"><tr><td>0011 1111</td><td>1000 0000</td></tr><tr><td>0000 0000</td><td>0000 0000</td></tr></table>	0011 1111	1000 0000	0000 0000	0000 0000
0011 1111	1000 0000				
0000 0000	0000 0000				
$f$	0				
$e$	$0111\ 1111_2 = 7F_{16} = 127_{10}$				
$s$	0 (~ "+")				

$$(127, 0, 0) = +1.0 \cdot 2^{127-127} = +1.0$$



## Примеры вещественных чисел с одинарной точностью (2)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	-1								
Двоичное представление	$-1 \cdot 2^0$								
Результат упаковки (hex)	bf 80 00 00								
Двоичное представление	<table border="1"><tr><td>1</td><td>011 1111</td><td>1</td><td>000 0000</td></tr><tr><td>0000</td><td>0000</td><td>0000</td><td>0000</td></tr></table>	1	011 1111	1	000 0000	0000	0000	0000	0000
1	011 1111	1	000 0000						
0000	0000	0000	0000						
$f$	0								
$e$	$0111\ 1111_2 = 7F_{16} = 127_{10}$								
$s$	1 (~ "-")								

$$(127, 0, 1) = -1.0 \cdot 2^{127-127} = -1.0$$



## Примеры вещественных чисел с одинарной точностью (3)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	0.5		
Двоичное представление	$1 \cdot 2^{-1} = 0.1_2$		
Результат упаковки (hex)	3f 00 00 00		
Двоичное представление	<table border="1"><tr><td>0011 1111 0000 0000</td></tr><tr><td>0000 0000 0000 0000</td></tr></table>	0011 1111 0000 0000	0000 0000 0000 0000
0011 1111 0000 0000			
0000 0000 0000 0000			
$f$	0		
$e$	$0111 1110_2 = 7E_{16} = 126_{10}$		
$s$	0 (~ "+")		

$$(126, 0, 0) = +1.0 \cdot 2^{126-127} = +0.5$$



# Примеры вещественных чисел с одинарной точностью (4)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	0.0625								
Двоичное представление	$1 \cdot 2^{-4} = 0.0001$								
Результат упаковки (hex)	3d 80 00 00								
Двоичное представление	<table border="1"><tr><td>0</td><td>011 1101</td><td>1</td><td>0000 0000</td></tr><tr><td>0000</td><td>0000</td><td>0000</td><td>0000</td></tr></table>	0	011 1101	1	0000 0000	0000	0000	0000	0000
0	011 1101	1	0000 0000						
0000	0000	0000	0000						
$f$	0								
$e$	$0111\ 1011_2 = 7B_{16} = 123_{10}$								
$s$	0 (~ "+")								

$$(127, 0, 0) = +1.0 \cdot 2^{123-127} = +2^{-4} = 0.0625$$



# Перевод вещественных чисел из десятичной СС в двоичную СС

Пусть дано десятичное вещественное число  $A_{10} = A'_{10} \cdot A''_{10}$

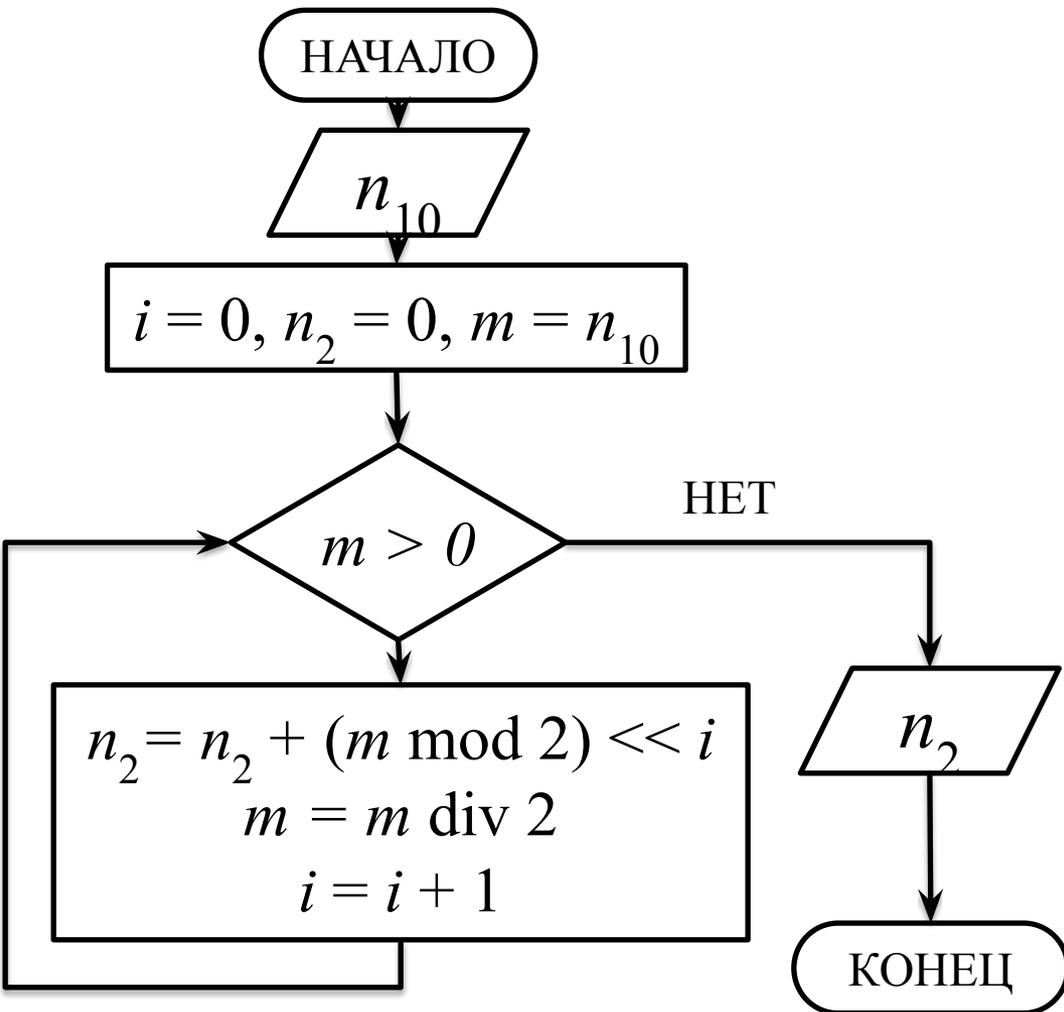
**Необходимо:** найти  $A_2$

**Решение:**

1. Найти  $A'_2$  из  $A'_{10}$ , используя алгоритм  $DB^1$  перевода целых десятичных чисел в двоичные.
2. Найти  $A''_2$  из  $A''_{10}$ , используя алгоритм  $DB^2$  перевода дробных ( $<1$ ) десятичных чисел в двоичные.
3.  $A_2 = A'_2 \cdot A''_2$



# Алгоритм ДВ<sup>1</sup>



ВВОД  $n_{10}$

$i = 0, n_2 = 0, m = n_{10}$

ПОКА  $m > 0$  ДЕЛАТЬ

$n_2 = n_2 + (m \bmod 2) \ll i$

$m = m \operatorname{div} 2$

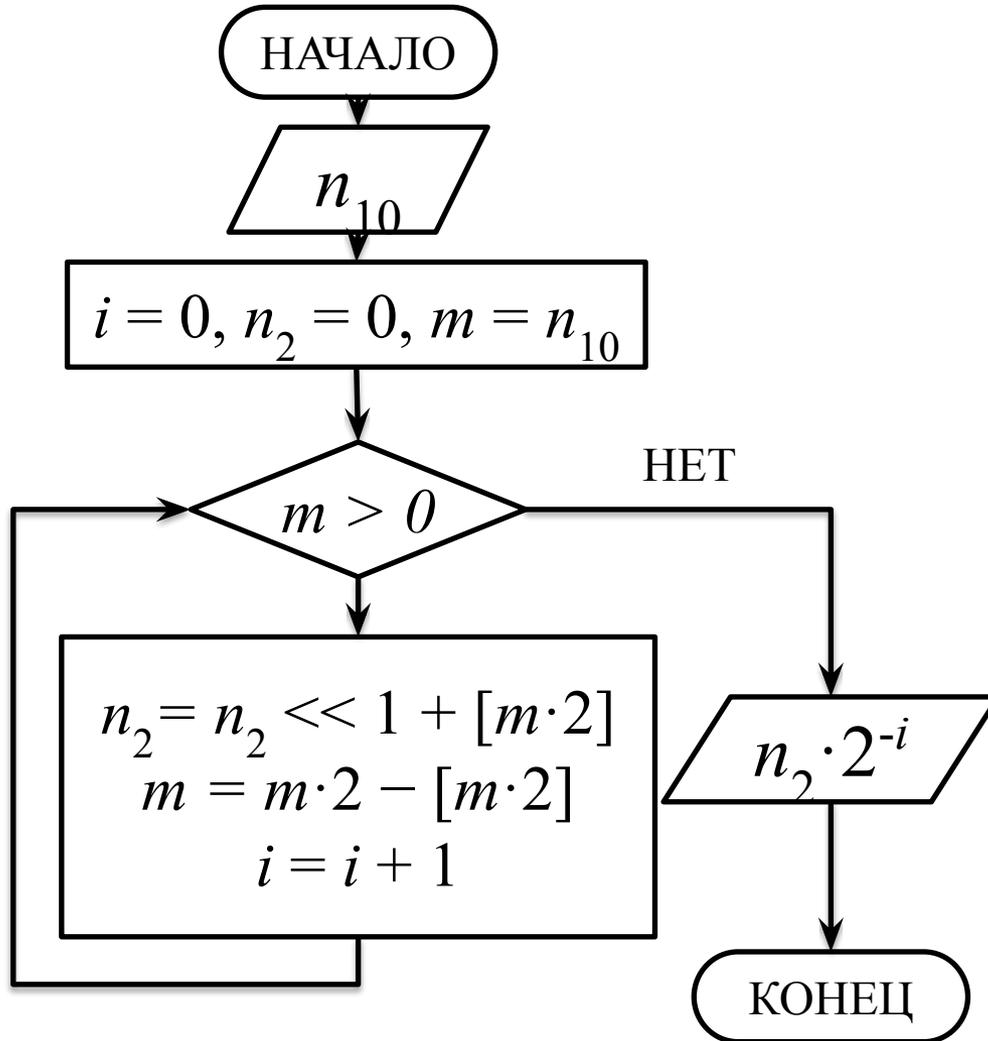
$i = i + 1$

КОНЕЦ ПОКА

ВЫВОД  $n_2$



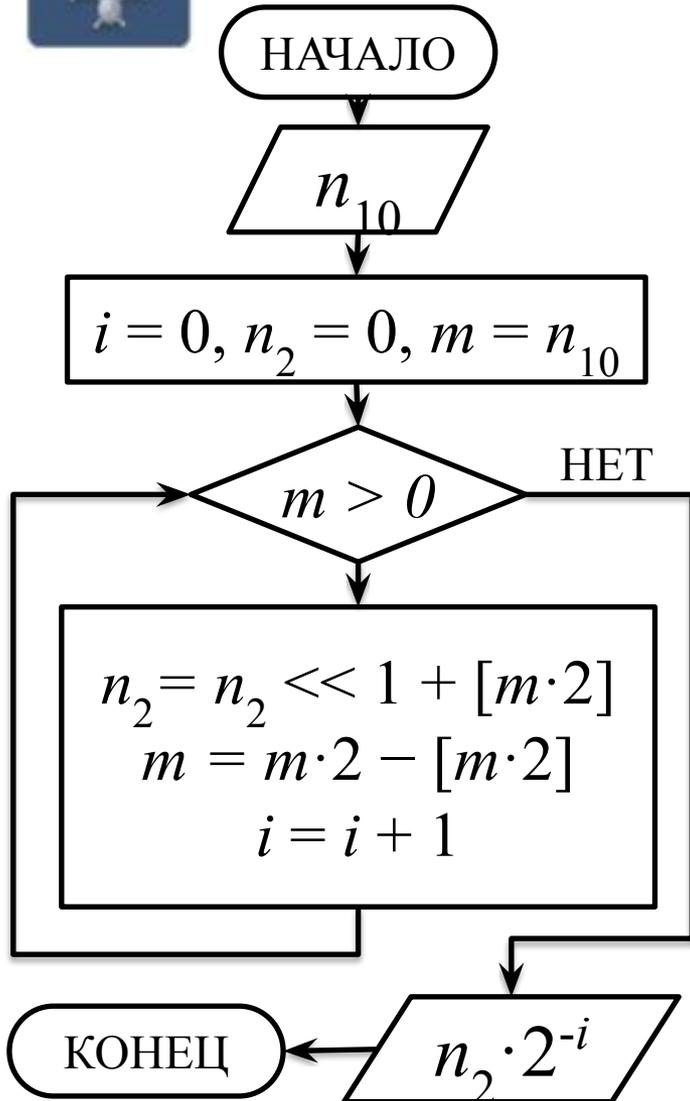
# Алгоритм $DB^2$



ВВОД  $n_{10}$   
 $i = 0, n_2 = 0, m = n_{10}$   
ПОКА  $m > 0$  ДЕЛАТЬ  
 $n_2 = n_2 \ll 1 + [m \cdot 2]$   
 $m = m \cdot 2 - [m \cdot 2]$   
 $i = i + 1$   
КОНЕЦ ПОКА  
ВЫВОД  $n_2 \cdot 2^{-i}$



# Представление числа $0.3_{10}$



1.  $n_2 = 0_2 \ll 1 + [0.3 \cdot 2] = 0_2$   
 $m = 0.6, i = 1$

2.  $n_2 = 0 \ll 1 + [0.6 \cdot 2] = 1_2$   
 $m = 1.2 - 1 = 0.2, i = 2$

3.  $n_2 = 1_2 \ll 1 + [0.2 \cdot 2] = 10_2$   
 $m = 0.4 - 0 = 0.4, i = 3$

4.  $n_2 = 10_2 \ll 1 + [0.4 \cdot 2] = 100_2$   
 $m = 0.8 - 0 = 0.8, i = 4$

5.  $n_2 = 100 \ll 1 + [0.8 \cdot 2] = 1001_2$   
 $m = 1.6 - 1 = \mathbf{0.6}, i = 5$

$$n_2 \cdot 2^{-5} = 0.01001$$

6. ...

$$0.3_{10} = 0.01001 \dots 1001 \ 1001 \ 1001 \ 1001 \ \dots$$



# Примеры вещественных чисел с одинарной точностью (5)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	0.3								
Двоичное представление	0.01001100110011001 ...								
Результат упаковки (hex)	3e 99 99 9a								
Двоичное представление	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0011</td> <td>1110</td> <td>1001</td> <td>1001</td> </tr> <tr> <td>1001</td> <td>1001</td> <td>1001</td> <td>1010</td> </tr> </table>	0011	1110	1001	1001	1001	1001	1001	1010
0011	1110	1001	1001						
1001	1001	1001	1010						
<i>f</i>	1.001 1001 1001 1001 1001 1010								
<i>e</i>	0111 1101 <sub>2</sub> =7D <sub>16</sub> =125 <sub>10</sub>								
<i>s</i>	0 (~ "+")								

$$(127, 0, 0) = +1.0011001... \cdot 2^{125-127} =$$

$$1.0011001... \cdot 2^{-2} = 0.010011001....$$



# Примеры вещественных чисел с одинарной точностью (6)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	5.3				
Двоичное представление	101.0100110011001...				
Результат упаковки (hex)	40 a9 99 9a				
Двоичное представление	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="border: 1px solid green;">0100 0000</td> <td style="border: 1px solid blue;">1010 1001</td> </tr> <tr> <td style="border: 1px solid blue;">1001 1001</td> <td style="border: 1px solid blue;">1001 1010</td> </tr> </table>	0100 0000	1010 1001	1001 1001	1001 1010
0100 0000	1010 1001				
1001 1001	1001 1010				
<i>f</i>	1.010 10011001 1001 1001 1010				
<i>e</i>	0100 0000 <sub>2</sub> =81 <sub>16</sub> =129 <sub>10</sub>				
<i>s</i>	0 (~ "+")				

$$(127, 0, 0) = +1.01010011001... \cdot 2^{129-127} =$$

$$1.01010011001... \cdot 2^2 = 101.010011001....$$



## Анализ внутреннего представления числа $5.3_{10}$ в типе float

$$\begin{aligned} & 1.0101\ 0011\ 0011\ 0011\ 0011\ 010_2 \cdot 2^2 = \\ & = 101.01\ 0011\ 0011\ 0011\ 0011\ \underline{010}_2 = \\ & \quad 5 + 2^{-2} + 2^{-5} + 2^{-6} + 2^{-9} + 2^{-10} + \\ & \quad 2^{-13} + 2^{-14} + 2^{-17} + 2^{-18} + 2^{-20} = \\ & \quad 5 + 0.25 + 0,03125 + 0,015625 + \\ & \quad + 0,001953125 + 0,000976562 + \\ & \quad + 0,00012207 + 0,000061035 + \\ & + 0,000007629 + 0,000003815 + \underline{0,000000954} = \\ & = 5,299999237 + \underline{0,000000954} = 5,300000191 \end{aligned}$$



# Примеры вещественных чисел с одинарной точностью (7)

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	5.1				
Двоичное представление	$1 \cdot 2^{-4} = 101.0\ 0011\ 0011\ 0011\dots$				
Результат упаковки (hex)	40 a3 33 33				
Двоичное представление	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="border: 1px solid green;">0100 0000</td> <td style="border: 1px solid green;">1010 0011</td> </tr> <tr> <td style="border: 1px solid blue;">0011 0011</td> <td style="border: 1px solid blue;">0011 0011</td> </tr> </table>	0100 0000	1010 0011	0011 0011	0011 0011
0100 0000	1010 0011				
0011 0011	0011 0011				
<i>f</i>	<b>1</b> .010 0011 0011 0011 0011 0011				
<i>e</i>	$0100\ 0000_2 = 81_{16} = 129_{10}$				
<i>s</i>	0 (~ "+")				

$$(127, 0, 0) = +1.010\ 0011\ 0011\ \dots \cdot 2^{129-127} =$$

$$1.010\ 0011\ 0011\ \dots \cdot 2^2 = 101.0\ 0011\ 0011\ \dots$$



## Анализ внутреннего представления числа $5.1_{10}$ в типе float

$$\begin{aligned} & 1.010\ 0011\ 0011\ 0011\ 0011\ 0011_2 \cdot 2^2 = \\ & = 101.0\ 0011\ 0011\ 0011\ 0011\ 0011_2 = \\ & \quad 5 + 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \\ & \quad 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} = \\ & \quad 5 + 0,0625 + 0,03125 + \\ & \quad 0,00390625 + 0,001953125 + \\ & \quad + 0,000244141 + 0,00012207 + \\ & \quad + 0,000015259 + 0,000007629 + \\ & \quad 0,000000954 + 0,000000477 = 5,099999905 \end{aligned}$$



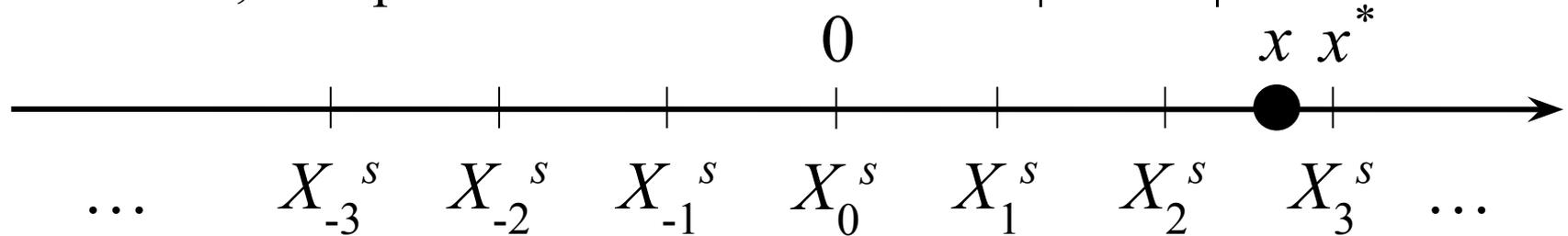
# Ошибки округления

- Технические средства **не позволяют** хранить числа, заданные бесконечными дробями
- Необходима замена любого числа конечной дробью, ограниченной доступным количеством разрядов. Данная операция называется *округлением*.
- Округлением числа  $x = \pm d_n \dots d_{s+1} d_s d_{s-1} d_{s-2}$  до  $s$  разрядов в заданной СС называется его замена числом  $x_s = \pm d_n \dots d_{s+1} d_s$ , в котором разряды  $(s - 1)$ ,  $(s - 2)$ , ... являются нулевыми.
- Разность  $(x - x_s)$  называется *ошибкой округления*.



## Способы округления

1. Отбрасывание разрядов  $(s - 1), (s - 2), \dots$  как это реализовано для целых чисел. **Достоинством** данного подхода является простота реализации в аппаратуре. **Недостатком** служит то, что знак ошибки  $(x^* - x)$  всегда противоположен знаку  $x$ .
2. Округление по правилам, используемым в школе. Вещественные числа  $X_i^s$ , имеющие нулевые младшие разряды  $(s - 1), (s - 2), \dots$  располагаются на числовой оси с шагом  $b^s$ , как показано на рисунке. При этом среди этих чисел есть число  $x^*$ , которое наиболее близко к  $x$  и  $|x^* - x| \leq b^s$



**b = 2, s = -2:** -0.11   -0.10   -0.01   0.00   0.01   0.10   0.11



## Примеры применения алгоритмов округления для числа $5.3_{10}$

Отбрасывание разрядов:

$$\begin{aligned} 5.3 &= 101.01\ 0011\ 0011\ 0011\ 0011\ 0011\ \underline{00110011}_2 = \\ &5,299999237 + 2^{-21} = 5,299999714 \\ &5,299999714 - 5,3 = -2.86 \cdot 10^{-7} \end{aligned}$$

Округление с использованием школьной арифметики

$$\begin{aligned} 5.3 &= 101.01\ 0011\ 0011\ 0011\ 0011\ 0011\ \underline{00110011}_2 = \\ &101.01\ 0011\ 0011\ 0011\ 0011\ 0011\ \underline{001} + 2^{-21} = \\ &101.01\ 0011\ 0011\ 0011\ 0011\ 0011\ \underline{010} = \\ &5,299999237 + 2^{-20} = 5,300000191 \\ &5,300000191 - 5,3 = 1.91 \cdot 10^{-7} \end{aligned}$$



## Примеры применения алгоритмов округления для числа $5.1_{10}$

Отбрасывание разрядов:

$$5.1 = 101.0\ 0011\ 0011\ 0011\ 0011\ 0011\ 1\ \cancel{0011}_2 =$$
$$5,099999905$$

$$5,099999905 - 5,1 = -0.95 \cdot 10^{-7}$$

Округление с использованием школьной арифметики

$$5.1 = 101.0\ 0011\ 0011\ 0011\ 0011\ 0011\ 1\ \cancel{0011}_2 =$$
$$5,099999905$$

$$5,099999905 - 5,1 = -0.95 \cdot 10^{-7}$$



# Машинный ноль

$$(e, f, s) = s1.f \cdot b^{e-q}$$

Десятичное представление	0										
Двоичное представление	0										
Результат упаковки (hex)	00 00 00 00										
Двоичное представление	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: 1px solid blue;">0</td> <td style="border: 1px solid green;">0000</td> <td style="border: 1px solid green;">0000</td> <td style="border: 1px solid blue;">0000</td> <td style="border: 1px solid blue;">0000</td> </tr> <tr> <td style="border: 1px solid blue;">0000</td> </tr> </table>	0	0000	0000	0000	0000	0000	0000	0000	0000	0000
0	0000	0000	0000	0000							
0000	0000	0000	0000	0000							
<i>f</i>	1.000 0000 0000 0000 0000 0000										
<i>e</i>	0000 0000 <sub>2</sub> = 00 <sub>16</sub> = 0 <sub>10</sub>										
<i>s</i>	0 (~ "+")										

$$(0, 0, 0) = +1.000\ 0000\ 0000\ \dots \cdot 2^{0-127} =$$

$$1.000\ 0000\ 0000\ \dots \cdot 2^{-127} = 2^{-127}$$



## Машинный ноль

- Число  $\omega$ , для которого  $f = 0$ , мантисса  $m = 1.0$ , а порядок  $e$  принимает наименьшее значение (для float  $e = 0 \Rightarrow (e - q) = -127$ ) называется машинным нулем.
- Оно совпадает с минимальным положительным числом, которое может быть представлено числом с плавающей точкой при заданных размерностях  $f$  и  $e$  (для float размерность  $f - 23$  бита,  $e - 8$  бит).
- Любое число  $x < \omega$  рассматривается как 0.



## Погрешности измерений

- **Абсолютная погрешность** ( $\Delta x$ ) – разность между приближенным значением некоторой величины и ее точным значением:  $\Delta x = |x_{\text{п}} - x_{\text{т}}|$ , где  $x_{\text{т}}$  – точное значение, а  $x_{\text{п}}$  – приближенное.
- **Относительная погрешность** ( $\delta x$ ) – погрешность измерения, выраженная отношением абсолютной погрешности измерения к действительному или измеренному значению измеряемой величины:

$$\delta x = \Delta x / x_{\text{п}},$$

$$\delta x = \Delta x / x_{\text{т}}.$$

**$\delta x$  является безразмерной величиной**



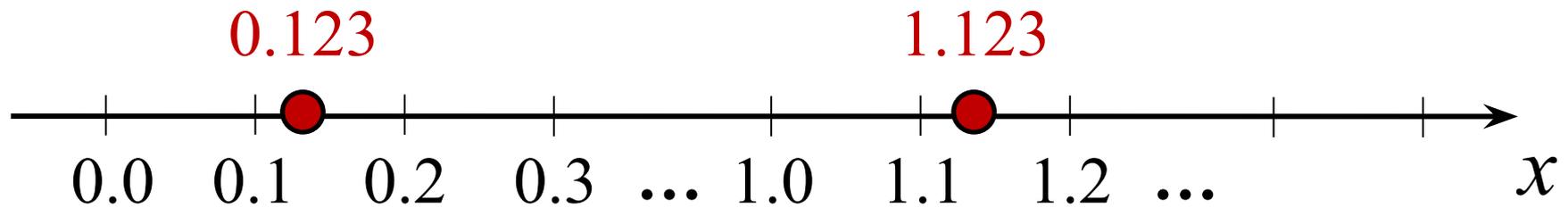
## Погрешность представления вещественных чисел

- В отличие от чисел с фиксированной точкой, сетка чисел, которые способна отобразить арифметика с плавающей точкой, неравномерна: она более густая для чисел с малыми порядками и более редкая — для чисел с большими порядками.
- Для чисел с фиксированной точкой постоянным является *порядок* абсолютной погрешности.
- Для чисел с плавающей точкой постоянным является *порядок* относительной погрешности.



## Погрешности вещественных чисел с фиксированной точкой

Рассмотрим 2-хразрядные числа с фиксированной точкой, один разряд отводится под целую часть, второй – под дробную.



$$\Delta x_1 = |x_{\text{п}} - x_{\text{т}}| = |0.123 - 0.1| = 0.023, \text{ порядок } 10^{-2}.$$

$$\Delta x_2 = |x_{\text{п}} - x_{\text{т}}| = |1.123 - 1.1| = 0.023, \text{ порядок } 10^{-2}.$$

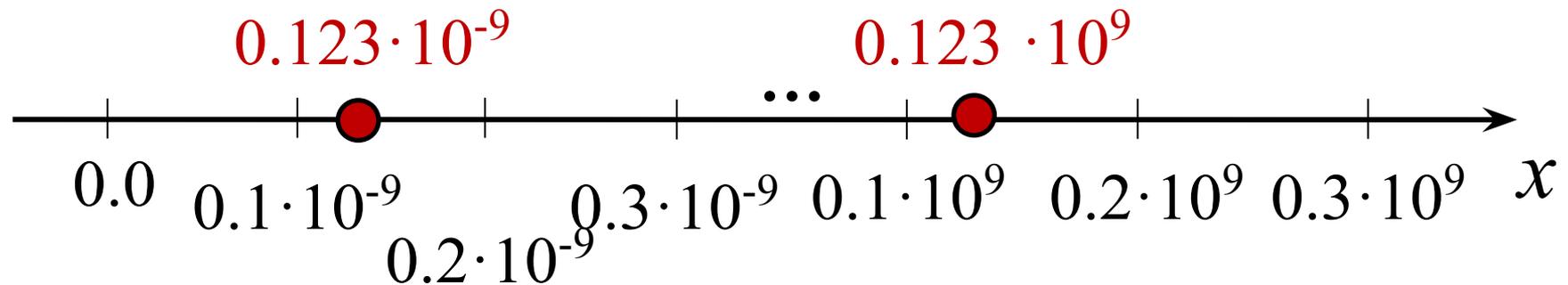
$$\delta x_1 = \Delta x / x_{\text{п}} = 0.023 / 0.123 \approx 0.186, \text{ порядок } 10^{-1}$$

$$\delta x_2 = \Delta x / x_{\text{п}} = 0.023 / 1.123 \approx 0,020, \text{ порядок } 10^{-2}$$



## Погрешности вещественных чисел с плавающей точкой

Рассмотрим 2-хразрядные числа с плавающей точкой, один разряд отводится под мантиссу, второй – под порядок.



$$\Delta x_1 = |x_{\Pi} - x_{\Gamma}| = 0.023 \cdot 10^{-9}, \text{ порядок } 10^{-10}.$$

$$\Delta x_2 = |x_{\Pi} - x_{\Gamma}| = 0.023 \cdot 10^9, \text{ порядок } 10^8.$$

$$\delta x_1 = \Delta x / x_{\Pi} = 0.023 \cdot 10^{-9} / 0.123 \cdot 10^{-9} \approx 0.186, \text{ порядок } 10^{-1}$$

$$\delta x_2 = \Delta x / x_{\Pi} = 0.023 \cdot 10^9 / 0.123 \cdot 10^9 \approx 0.186, \text{ порядок } 10^{-1}$$



## Машинный эпсилон

Машинным эпсилоном называется наименьшее положительное число  $\varepsilon$  такое, что

$$1 \oplus \varepsilon \neq 1,$$

где  $\oplus$  – машинное сложение.

### Пример:

Пусть даны два числа:  $a$  и  $b = a + \gamma$ .

Если  $\gamma < a \cdot \varepsilon$ , то с точки зрения машины  $a = b$ .

$$a < a \cdot (1 \oplus \gamma) < a \cdot (1 \oplus \varepsilon)$$

$$1 \oplus \gamma = 1$$



# Программа поиска машинного эпсилона

```
#include <stdio.h>
int main()
{
    float e,e1;
    int k=0;
    e=1.0;
    do
    {
        e=e/2.0;
        e1=e+1.0;
        k++;
    } while (e1>1.0);
    printf("Число делений на 2: %d\n",k);
    printf("Машинный эпсилон: %e\n",e);
    return 0;
}
```

Результаты работы:

Число делений на 2: 24

Машинный эпсилон: 5.960464e-08



# Алгоритм сложения чисел с плавающей точкой

Задача: найти число  $\omega = u \oplus v$

$a$  :  $(2p + 1)$  разрядный аккумулятор

- 1 ВВОД  $u, v$
- 2 распаковать  $u \rightarrow (e_u, \pm f_u)$  и  $v \rightarrow (e_v, \pm f_v)$
- 3 если  $e_u > e_v$  то  $u \leftrightarrow v$  ( $e_u \leftrightarrow e_v$  и  $f_u \leftrightarrow f_v$ ) конец если
- 4  $e_\omega = e_u$
- 5 если  $(e_u - e_v) \geq p + 2$  то  $f_\omega = f_v$
- 6 иначе
- 7 сдвиг  $f_v \gg (e_u - e_v) \Rightarrow a = f_v / b^{(e_u - e_v)}$
- 8  $a = f_u + a, f_\omega = \text{окр}(a, p)$
- 9 конец если
- 10 нормализация  $(e_\omega, \pm f_\omega)$  и упаковка  $(e_\omega, \pm f_\omega) \rightarrow \omega$



# Алгоритм нормализации чисел с плавающей точкой

почему?

Задача: обеспечить для числа  $\omega = (e_\omega, \pm f_\omega)$ ,  $f_\omega < b$   
выполнение условия:  $1/b \leq |f_\omega| < 1$

- 1 ВВОД  $\omega$
- 2 распаковать  $v \rightarrow (e_v, \pm f_v)$  и  $v \rightarrow (e_v, \pm f_v)$
- 3 если  $e_v > e_v$  то  $v \leftrightarrow v$  ( $e_v \leftrightarrow e_v$  и  $f_v \leftrightarrow f_v$ ) конец если
- 4  $e_\omega = e_v$
- 5 если  $(e_v - e_v) \geq p + 2$  то  $f_\omega = f_v$
- 6 иначе
- 7 сдвиг  $f_v \gg (e_v - e_v) \Rightarrow a = f_v / b^{(e_v - e_v)}$
- 8  $a = f_v + a$ ,  $f_\omega = \text{окр}(a, p)$
- 9 конец если
- 10 нормализация  $(e_\omega, \pm f_\omega)$  и упаковка  $(e_\omega, \pm f_\omega) \rightarrow \omega$



# ИТОГИ

- Рассмотрены внутренние представления целых чисел:
  - целых чисел без знака
  - целых чисел со знаком
    - прямой код, обратный код, дополнительный код;
    - арифметические действия (сложение и вычитание) над знаковыми числами в дополнительном коде.
- Внутреннее представление вещественных чисел:
  - вещественные числа с фиксированной и плавающей запятой;
  - упаковка вещественных чисел;
  - понятие машинного нуля, понятие погрешности вычислений, понятие машинного эпсилона;
  - арифметические действия (сложение и вычитание) над вещественными числами с плавающей точкой.



# ЛИТЕРАТУРА

- 1) Кнут, Д.Э. Искусство программирования. Том 2. Получисленные алгоритмы. – Вильямс, Addison Wesley Longman, 2000. – 500 с. – (Серия: Искусство программирования). – ISBN 0-201-89684-2.
- 2) Воеводин, В.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ, 2006. – 112 с. – ISBN 5-211-05310-9.
- 3) Вылиток, А.А. Представление чисел в ЭВМ [cmcmsu.no-ip.info/download/pc.number.representation.pdf](http://cmcmsu.no-ip.info/download/pc.number.representation.pdf)
- 4) Википедия, стандарт IEEE 754 1985 года [http://en.wikipedia.org/wiki/IEEE\\_754-1985](http://en.wikipedia.org/wiki/IEEE_754-1985).
- 5) Википедия, стандарт IEEE754 2008г. [http://en.wikipedia.org/wiki/IEEE\\_754-2008](http://en.wikipedia.org/wiki/IEEE_754-2008).