Курс «Алгоритмы и алгоритмические языки»

Лекция 8

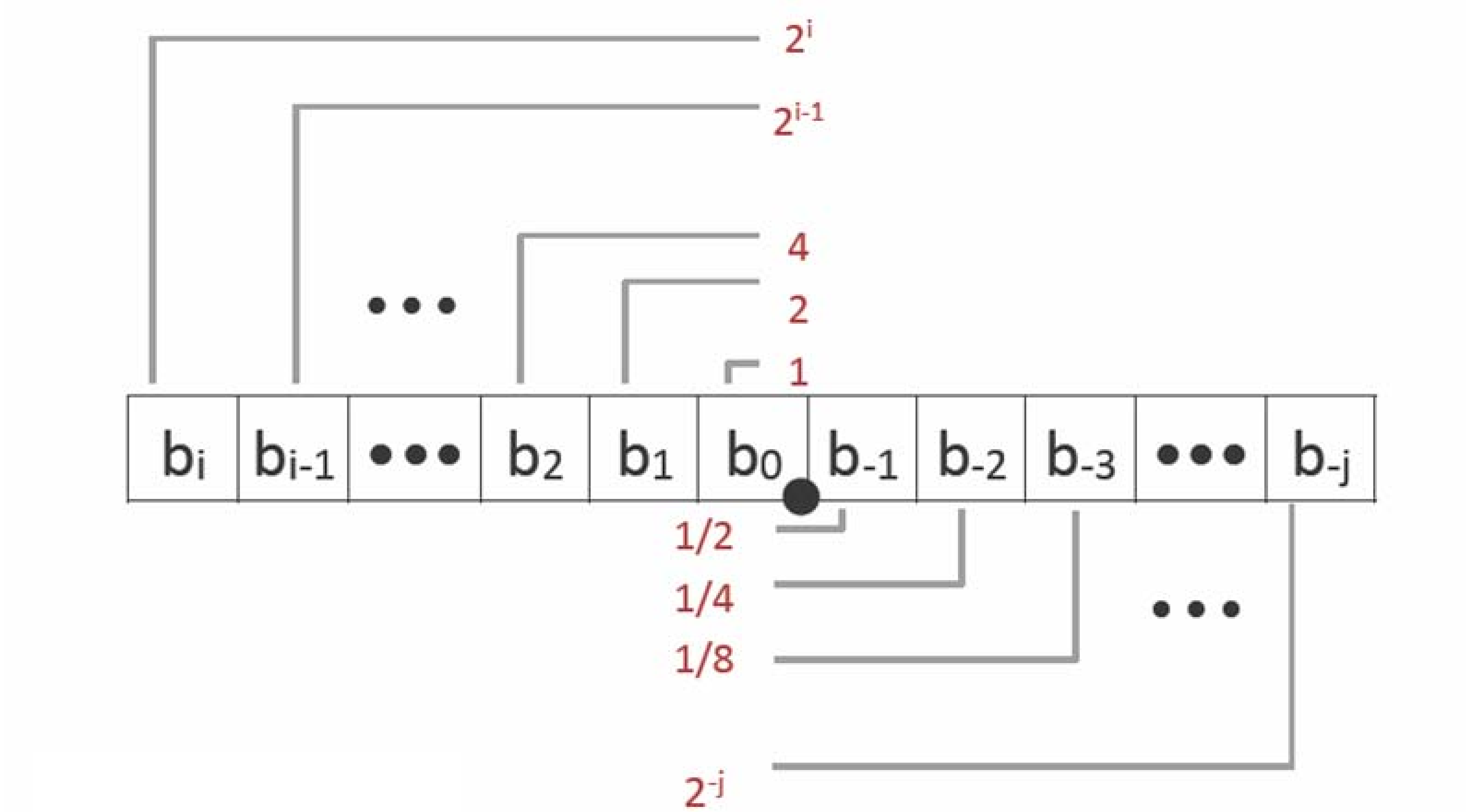
Дробные двоичные числа

- Что такое 1011.101_2 ?

$$1\times 2^{3}+0\times 2^{2}+1\times 2^{1}+1\times 2^{0}+1\times 2^{-1}+0\times 2^{-2}+1\times 2^{-3}=$$

$$=11\frac{5}{8}=11.625$$

Дробные двоичные числа



- Черное пятнышко – двоичная точка
- Биты слева от точки умножаются на положительные степени 2
- Биты справа от точки умножаются на отрицательные степени 2

Дробные двоичные числа

- $0.111111\ldots_2 = 1.0 - \varepsilon \ (\varepsilon \to 0)$, так как

$$\tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + \tfrac{1}{2^n} + \ldots \to 1 \quad \text{при } n \to \infty$$

- Точно можно представить только числа вида $\tfrac{x}{2^k}$

- Остальные рациональные числа представляются периодическими двоичными дробями:

$$\tfrac{1}{5} = 0.(0011)_2$$

- Иррациональные числа представляются апериодическими двоичными дробями и могут быть представлены только приближенно

Представление чисел с плавающей точкой (IEEE 754)

- Числа с плавающей точкой представляются в нормализованной форме: $(-1^{s})\, M\, 2^{e}$
  - s – код знака числа (он же знак мантиссы)
  - M – мантисса ($1 \le M < 2$)
  - e – (двоичный) порядок

- Первая цифра мантиссы в нормализованном представлении всегда 1. В стандарте принято решение не записывать в представление числа эту единицу (тем самым мантисса как бы увеличивается на разряд).

  Экономия связана с тем, что в представление числа записывается не M, а $frac = M - 1$

Представление чисел с плавающей точкой

- Чтобы не записывать отрицательных чисел в поле порядка, вводится ***смещение*** $bias = 2^{k-1} - 1$, где k – количество бит в поле для записи порядка, и вместо порядка e записывается код порядка exp, связанный с e соотношением $e = exp - bias$.

- Нормализованное число (-1^s) $M\ 2^e$ упаковывается в машинное слово (структуру) с полями s, $frac$ и exp

s	exp (код порядка)	$frac$ (код мантиссы)

Ширина поля s всегда равна 1.

Ширина полей exp и $frac$ зависит от точности числа

Представление чисел с плавающей точкой

- Одинарная точность (32 бита):

s	*exp* (код порядка)	*frac* (код мантиссы)

8 бит   23 бита

$bias = 127; \quad -126 \le e \le 127\ ; \quad 1 \le exp \le 254$

- Двойная точность (64 бита):

s	*exp* (код порядка)	*frac* (код мантиссы)

11 бит   52 бита

$bias = 1023; \quad -1022 \le e \le 1023\ ; \quad 1 \le exp \le 2046$

- Повышенная точность (80 бит):

s	*exp* (код порядка)	*frac* (код мантиссы)

15 бит   64 бита

Представление чисел с плавающей точкой

- Пример 1
  - **Значение** `float f = 15213.0`

    $15213_{10} = 11101101101101_2 =$

    $1.1101101101101_2 \times 2^{13}$

  - **Значащая часть**

    $M = 1.\underline{1101101101101}_2$,

    $frac = \underline{1101101101101}0000000000_2$

  - **Порядок**

    $e = 13$

    $bias = 127$

    $exp = 140 = 10001100_2$

  - **Результат**

0	10001100	11011011011010000000000
s	exp	frac

Представление нуля

- Для типа **`float`** код порядка **exp** изменяется от **`00000001`** до **`11111111`** (значению **`00000001`** соответствует порядок **`e = - 126`**, значению **`11111111`** – порядок **`e = 127`**)

- Код **exp = `00000000, frac = 000…0`** представляет нулевое значение; в зависимости от значения знакового разряда **`s`** это либо **`+0`** либо **`-0`**

- А какое значение представляет код **`exp = 00000000, frac`** $\neq$ **`000…0`**?

Большие числа

Пусть **`exp = 111…1`**

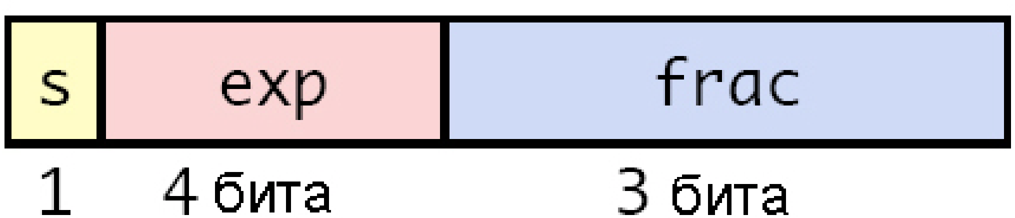
- если при этом **`frac = 000…0`**, то коду будет соответствовать значение ∞ (со знаком **`s`**)
- если же **`frac`** $\neq$ **`000…0`**, то код не будет представлять никакое число («значение», представляемое таким кодом, так и называется: «не число» – **`NaN`** – Not a number)

Денормализованные числа

- Это числа, представляемые кодами
  `exp = 00000000, frac ≠ 000…0`?

- **`exp`** вносит в значение такого числа постоянный вклад 2^{-k-2},
  `frac` меняется от **`000…01`** до **`111…1`** ***и*** рассматривается уже не как мантисса, а как значение, умножаемое на **`exp`**

- Рассмотрим это на модельном примере:

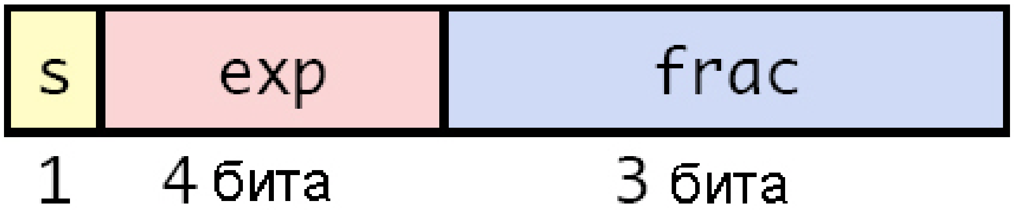


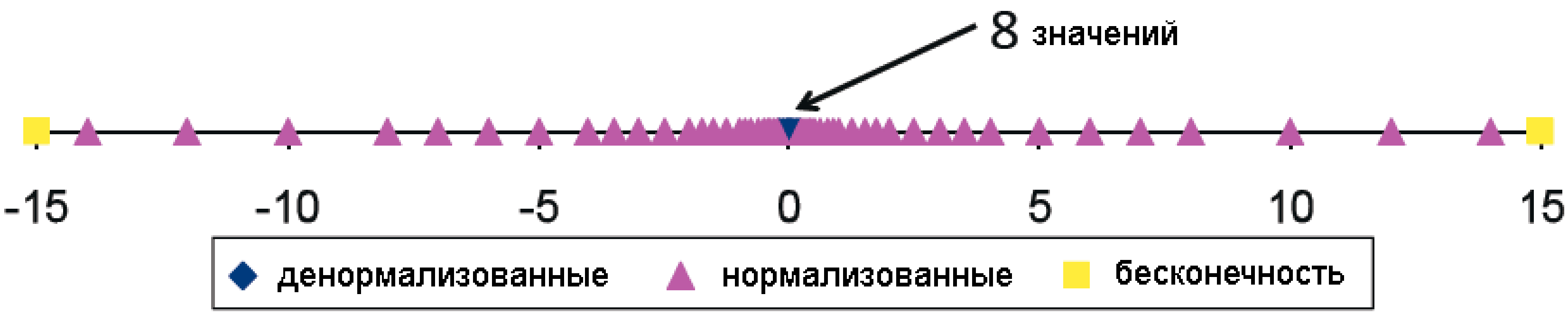
8-разрядные числа с плавающей точкой (положительные)

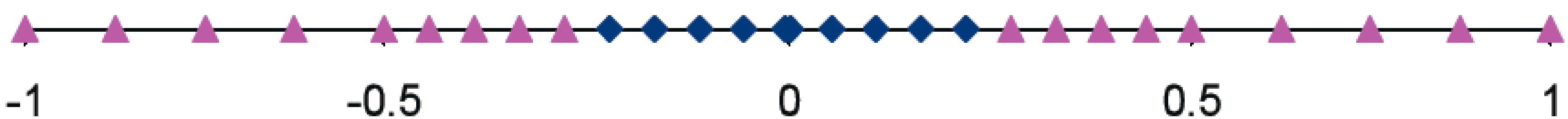
s	exp	frac
1	4 бита	3 бита

	s	exp	frac	E	Value	
Ненормализованные числа	0	0000	000	-6	0	
	0	0000	001	-6	1/8*1/64 = 1/512	Близкие к 0
	0	0000	010	-6	2/8*1/64 = 2/512	
	…					
	0	0000	110	-6	6/8*1/64 = 6/512	
	0	0000	111	-6	7/8*1/64 = 7/512	Наибольшее ненормализованное
Нормализованные числа	0	0001	000	-6	8/8*1/64 = 8/512	Наименьшее нормализованное
	0	0001	001	-6	9/8*1/64 = 9/512	
	…					
	0	0110	110	-1	14/8*1/2 = 14/16	
	0	0110	111	-1	15/8*1/2 = 15/16	Ближайшее к 1 снизу
	0	0111	000	0	8/8*1 = 1	
	0	0111	001	0	9/8*1 = 9/8	Ближайшее к 1 сверху
	0	0111	010	0	10/8*1 = 10/8	
	…					
	0	1110	110	7	14/8*128 = 224	
	0	1110	111	7	15/8*128 = 240	Наибольшее нормализованное
	0	1111	000	n/a	inf	

8-разрядные числа с плавающей точкой



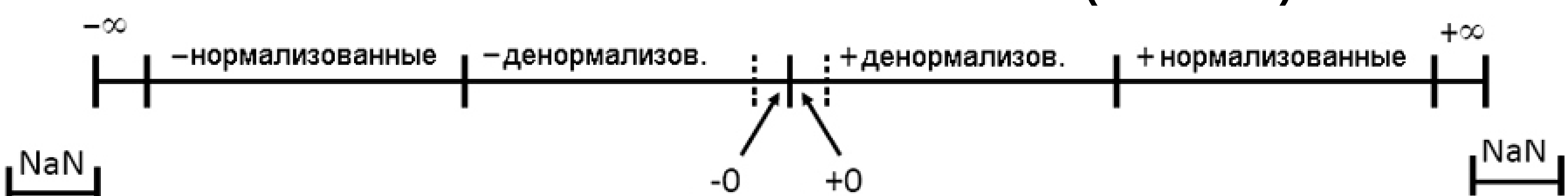




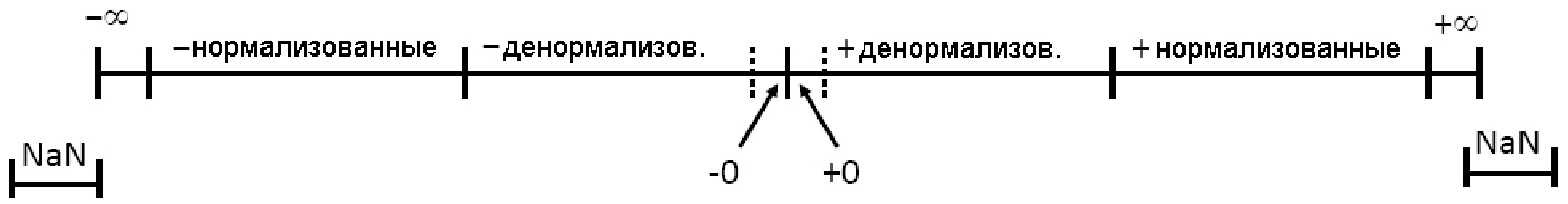
Центральная область более крупно

Важные примеры

	exp	frac	Численное значение
◊ Нуль	00…00	00…00	0.0
◊ Наим. положит. денорм.	00…00	00…01	
♦ **float** ≈ 1.4×10^{-45}			$2^{-23} \times 2^{-126}$
♦ **double** ≈ 4.9×10^{-324}			$2^{-52} \times 2^{-1022}$
◊ Наиб. положит. денорм.	00…00	11…11	
♦ **float** ≈ 1.18×10^{-38}			$(1.0 - \varepsilon) \times 2^{-126}$
♦ **double** ≈ 2.2×10^{-308}			$(1.0 - \varepsilon) \times 2^{-1022}$
◊ Наим. положит. норм.	00…01	00…00	
♦ **float**			1.0×2^{-126}
♦ **double**			1.0×2^{-1022}
◊ Единица	01…11	00…00	1.0
◊ Наиб. положит. норм.			
♦ **float** ≈ 3.4×10^{38}			$(2.0 - \varepsilon) \times 2^{127}$
♦ **double** ≈ 1.8×10^{308}			$(2.0 - \varepsilon) \times 2^{1023}$



Представление чисел с плавающей точкой



Операции над числами с плавающей точкой

- По определению:

$$x +_{FP} y = Round(x + y)$$

$$x \times_{FP} y = Round(x \times y)$$

где $Round()$ означает округление

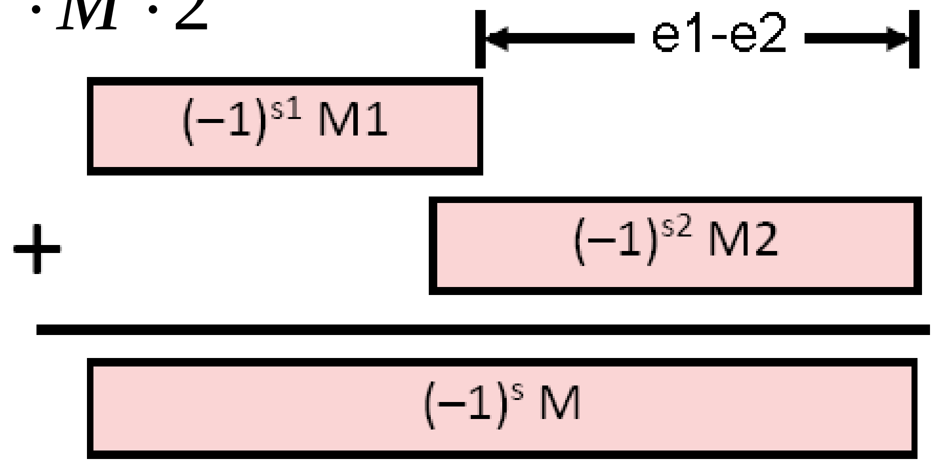
- Выполнение операции
  - Сначала вычисляется точный результат (получается более длинная мантисса, чем запоминаемая, иногда в два раза)
  - Потом фиксируется исключение (например, переполнение)
  - Потом результат округляется, чтобы поместиться в поле $frac$

Умножение чисел с плавающей точкой

- $(-1)^{s1} \cdot M1 \cdot 2^{e1} \times (-1)^{s2} \cdot M2 \cdot 2^{e2}$
- Точный результат $(-1)^{s} \cdot M \cdot 2^{e}$
  - Знак s — $s1 \wedge s2$
  - Значащие цифры M — $M1 \times M2$
  - Порядок e — $e1 + e2$
- Преобразование
  - Если $M \geq 2$, сдвиг M вправо с одновременным увеличением
  - Если e не помещается в поле exp, переполнение
  - Округление M, чтобы оно поместилось в поле $frac$
- Основные затраты на перемножение мантисс.

Сложение чисел с плавающей точкой

- $$(-1)^{s1} \cdot M1 \cdot 2^{e1} + (-1)^{s2} \cdot M2 \cdot 2^{e2}$$
  Пусть $e1 > e2$
- Точный результат $(-1)^{s} \cdot M \cdot 2^{e}$
  - Знак s и значащие цифры M вычисляются как показано на рисунке
  - Порядок суммы – $e1$



- Преобразование
  - Если $M \geq 2$, сдвиг M вправо с одновременным увеличением e
  - Если $M < 1$, сдвиг M влево на k позиций с одновременным вычитанием k из e
  - Если e не помещается в поле exp, переполнение
  - Округление M, чтобы оно поместилось в поле $frac$

Плавающие типы языка Си

`float`, `double`, `long double`

- ***Операции над данными с плавающей точкой*.**
  - *Одноместные*: изменение знака («одноместный минус»: **`-`**), одноместный плюс (**`+`**).
  - *Двухместные*: сложение (**`+`**), вычитание (**`-`**), умножение (**`*`**), деление (**`/`**).
- ***Порядок выполнения арифметических операций в выражениях (приоритет).***
  - самый низкий приоритет у двуместных **`+`** и **`-`**,
  - более высокий приоритет у двуместных **`*`** и **`/`**
  - еще более высокий приоритет у одноместных **`+`** и **`-`**.
  - В выражениях без скобок операции с более высоким приоритетом выполняются раньше.
  - Скобки позволяют изменить порядок выполнения операций.

Пример 1. Вычисление суммы 5 чисел типа **`float`**

(мантисса – 6 десятичных цифр, порядок – 2 десятичных цифры):

$0.231876*10^{02} + 0.645391*10^{-03} + 0.231834*10^{-01} + 0.245383*10^{-02} + 0.945722*10^{-03} =$

a) $\mathbf{0.231876*10^{02} + 0.645391*10^{-03} + 0.231834*10^{-01} + 0.245383*10^{-02} + 0.945722*10^{-03} = 0.232147*10^{02}}$**;**

$23.1876 + 0.000645391 = 23.188245391 = 23.1882 = 0.\ 231882*10^{02}$;

$23.1882 + 0.0231834 = 23.2113834 = 23.2114 = 0.232114*10^{02}$;

$23.2114 + 0.00245383 = 23.21385383 = 23.2138*10^{02}$;

$23.\ 2138 + 0.000945722 = 23.214745722 = 23.2147 = 0.232147*10^{02}$;

b) $\mathbf{0.645391*10^{-03} + 0.9457*10^{-03} + 0.245383*10^{-02} + 0.231834*10^{-01} + 0.231876*10^{02} = 0.232157*10^{02}}$**;**

$0.000645391 + 0.000945722 = 0.001591113 = 0.00159111 = 0.\ 159111*10^{-02}$;

$0.00159111 + 0.00245383 = 0.00494493 = 0.\ 494493*10^{-02}$;

$0.00494493 + 0.0231834 = 0.02812833 = 0.0281283 = 0.\ 281283*10^{-01}$;

$0.0281283 + 23.1876 = 23.2157283 = 23.2157 = 0.232157*10^{02}$;

Пример 2. Вычисление разности плавающих чисел

(мантисса – 6 десятичных цифр, порядок – 2 десятичных цифры):

$\mathbf{0.238617*10^{02} - 0.238616*10^{02} + 0.645391*10^{04} - 0.645392*10^{04} + 0.845791*10^{00} - 0.835790*10^{00} =}$

a) $0.238617*10^{02} - 0.238616*10^{02} + 0.645391*10^{04} - 0.645392*10^{04} + 0.845791*10^{00} - 0.835790*10^{00} = \mathbf{0.100000*10^{-05}}$

$0.238617*10^{02} - 0.238616*10^{02} = 23.8617 - 23.8616 = 0.0001 = \mathbf{0.100000*10^{03}}$

$0.100000*10^{-03} + 0.645391*10^{04} = 0.0001 + 6453.91 = 6453.9101 = \mathbf{0.645391*10^{04}}$

$0.645391*10^{04} - 0.645392*10^{04} = -\,0.000001*10^{04} = \mathbf{-\,0.100000*10^{-01}}$

$-\,0.100000*10^{-01} + 0.845791*10^{00} = -\,0.01 + 0.845791 = 0.835791*10^{00}$

$0.835791*10^{00} - 0.835790*10^{00} = \mathbf{0.000001*10^{00} = 0.100000*10^{-05}}$

b) $0.238617*10^{02} + 0.645391*10^{04} + 0.845791*10^{00} - (0.238616*10^{02} + 0.645392*10^{04} + 0.835790*10^{00}) = \mathbf{0.100000*10^{00}}$

$0.238617*10^{02} + 0.645391*10^{04} = 23.8617 - 6453.91 = 6478.6 = \mathbf{0.647777*10^{04}}$

$0.647777*10^{04} + 0.845791*10^{00} = 6477.77 + 0.845791 = 6478.615791 =$ ***$0.647862*10^{04}$***

$0.238616*10^{02} + 0.645392*10^{04} = 23.8616 + 6453.92 = 6477.7816 = \mathbf{6477.78*10^{04}}$

$6477.78*10^{04} + 0.835790*10^{00} = 6477.78 + 0.835790 = 6478.61579 =$ ***$0.647852*10^{04}$***

$0.647862*10^{04} - 0.647852*10^{04} = \mathbf{0.000010*10^{04} = 0.100000*10^{-00}}$

Выводы.

(1) ***При вычислении суммы чисел с одинаковыми знаками*** необходимо упорядочить слагаемые по возрастанию и складывать, начиная с наименьших слагаемых.

(2) ***При вычислении суммы чисел с разными знаками*** необходимо сначала сложить все положительные числа, потом – все отрицательные числа и в конце выполнить одно вычитание.

(3) ***Вычитание*** (сложение чисел с противоположными знаками) ***часто приводит к потере точности***, которая у чисел с плавающей точкой определяется количеством значащих цифр в мантиссе (при вычитании двух близких чисел мантисса «исчезает», что ведет к резкой потере точности). Итак, ***чем меньше вычитаний, тем точнее результат.***

Замечание

Значащими цифрами числа с плавающей точкой называются все цифры его мантиссы за исключением нулей, стоящих в ее конце. Например, у числа 0.**6700089**0000 * 10^3 все цифры, выделенные жирным шрифтом, значащие. При вычитании двух близких чисел почти все значащие цифры пропадают. Например, 0.67000890 * 10^3 - 0.67000880 * 10^3 = 0.00000010 * 10^3 = 0.10 * 10^{-4}. Таким образом, у результата всего одна значащая цифра, хотя у операндов было по 7 значащих цифр.