Программирование на языке Java

9. Типы с плавающей точкой
10. Методы класса Методы класса Math

Программирование на языке Java

Тема 9. Типы с плавающей точкой

Типы с плавающей точкой – 1

Числа с плавающей точкой, (в математике действительные или вещественные числа), используются при вычислениях, которые требуют получения результата с точностью до определенного десятичного знака.

Пример. Вычисление квадратного корня, трансцендентных функций (**`sin()`**, **`cos()`**, …).

В Java существует два типа с плавающей точкой: **`float`** и **`double`** (числа одинарной и двойной точности).

Типы с плавающей точкой – 2

Стандарт **IEEE754**

Число представлено в виде $\pm m \cdot 2^e$,

где m – мантисса, e – порядок (экспонента)

Тип	Бит	Знак	Мантисса	Порядок	Min	Max
float	32	1	23	8	1.4e-045	3.4e038
double	64	1	52	11	4.9e-324	1.8e308

Типы с плавающей точкой – 3

Тип **`float`** используется, когда требуется дробная часть без **особой точности**, например, для представления денежных сумм в рублях и копейках.

Применение типа **`double`** наиболее рационально, когда требуется сохранение точности множества последовательных вычислений или манипулирование большими числами.

Все трансцендентные математические функции (**`sin()`**, **`cos()`**, **`sqrt()`**,…) возвращают значения типа **`double`**.

Константы с плавающей точкой – 1

Числа с плавающей точкой представляют десятичные значения с дробной частью.

Стандартная форма записи десятичного числа состоит из:

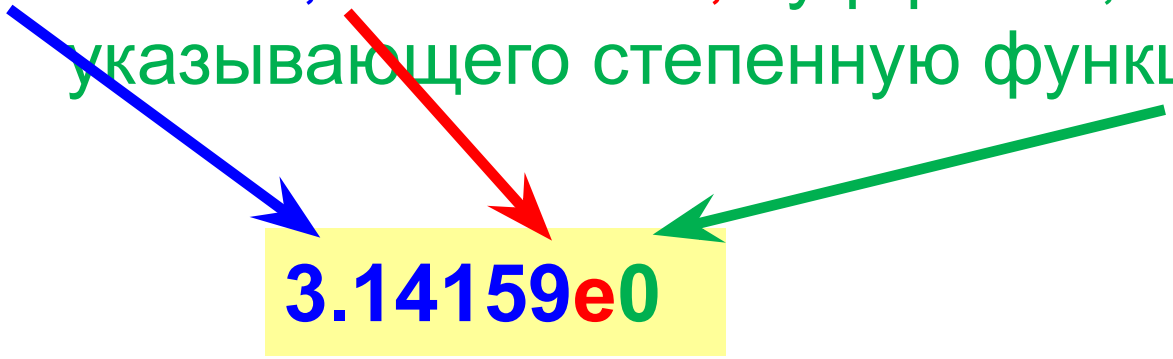
целого числа; десятичной точки; дробной части.

Константы с плавающей точкой – 2

Научная форма записи десятичного числа

состоит из:

мантиссы; символа E, суффикса,
указывающего степенную функцию числа 10



Константы с плавающей точкой – 3

Задача. Записать в стандартной форме

1.44e-6 = 0.00000144

0.832e8= 83200000.0

0.000034e7= 340.0

0.00524e-1= 0.000524

Константы с плавающей точкой – 4

По умолчанию в **Java** константам с плавающей точкой присвоен тип **double**.

Для указания константы типа **float**, к ней нужно дописать символ **F** или **f**.

```
float x;
x = 23.48f;
```

Также существует суффикс **d** или **D**

```
double y = 3D;
```

Особые случаи: бесконечность

- Деление положительного числа на 0.0 дает +∞
- Деление отрицательного числа на 0.0 дает −∞
- Переполнение дает +∞ или −∞, в зависимости от направления

```
double posInfinity =  1.0 / 0.0;
double negInfinity = -1.0 / 0.0;
double x = posInfinity + 1;
```

Какое значение примет x?

Особые случаи: NaN

- Деление 0.0 на 0.0 дает **NaN** (Not a Number – не число)
- Любая арифметическая операция с NaN дает NaN
- NaN != NaN

```
double nan =  0.0 / 0.0;
nan = posInfinity + negInfinity ;
```

Значение NaN

К получению **NaN** приводит:

- все математические операции с NaN;
- деление нуля на ноль;
- деление бесконечности на бесконечность;
- умножение нуля на бесконечность;
- сложение бесконечностей с противоположными знаками;
- вычисление квадратного корня отрицательного числа;
- логарифмирование отрицательного числа.

Точность вычислений – 1

- Для любого a вещественного типа существует ε>0: **a + ε == a**

```
double t = 0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
           0.1 + 0.1 + 0.1 + 0.1 + 0.1;
System.out.println(t);
```

```
0.9999999999999999
```

Почему?

- Из-за погрешности вычислений, поэтому при сравнении вещественных чисел проверяют не равенство (a==b), а близость этих чисел

$$|a - b| < \varepsilon$$

Почему по модулю?

Точность вычислений – 2

- При сравнении вещественных чисел проверяют не равенство (a==b), а близость этих чисел

$$|a - b| < \varepsilon$$

```
double t = 0.1 + 0.1 + 0.1 + 0.1 + 0.1 +
           0.1 + 0.1 + 0.1 + 0.1 + 0.1;
System.out.println(t == 1);
double eps = 1e-10;
System.out.println(Math.abs(t - 1) < eps);
```

```
false
true
```

Модификатор `strictfp`

- Java использует математический сопроцессор (FPU – Floating Point Unit) для вычислений с плавающей точкой;
- Регистры FPU могут быть шире 64 бит
- Результаты вычислений могут отличаться

- Модификатор **`strictfp`** включает режим строгой совместимости, результаты будут идентичны на любом процессоре

```
public strictfp class ClassName {
    ...
}
```

Ввод с клавиатуры

Аналогично вводу целых чисел

```
Scanner in = new Scanner(System.in);
System.out.println("Введите x");
float x = in.nextFloat();
System.out.println("Введите y");
double y = in.nextDouble();
```

```
Введите x
12,567
Введите y
0,00034
```

Внимание! При считывании с клавиатуры используется разделитель запятая

Почему запятая?

Форматный вывод – 1

Спецификаторы формата **%f, %e, %g**.

```
double x = 12345.6789;
System.out.printf ("%f",
x);
```

```
12345,678900
```

минимальное число позиций, **6 цифр** в дробной части

```
System.out.printf ("%e",
x);
```

```
1.234568e+04
```

Научная нотация: $1{,}23456 \cdot 10^4$

Форматный вывод. Указание точности

```
double x = 12345.6789;
System.out.printf ("%10.3f", x);
```

всего 10 позиций, **3 цифры** в дробной части

```
 12345,679
```

```
System.out.printf ("%10.2e",
x);
```

```
  1.23e+04
```

всего 10 позиций, **2 цифры** в дробной части мантиссы

Задача

Задача. Вычислить площадь круга заданного радиуса.

```
double pi, s, r;
r = in.nextDouble();
pi = 3.1415926;
s = pi * r * r;


System.out.printf("Площадь = %f", s);
```

Считываем с клавиатуры

Приблизительное значение π

Вычисление площади круга

Программирование на языке Java

Тема 10. Методы класса Math

Класс Math

Разработчику на **Java** доступно множество готовых (или библиотечных) классов и методов, полезных для использования в собственных программах.

Наличие библиотечных решений позволяет изящно решать множество типовых задач.

Класс **Math** содержит методы, которые используются в геометрии и тригонометрии, а также некоторые методы общего назначения.

Константы класса Math

~3,14

~2,72

2 константы типа **double**:

Math.PI – число π с точностью в 15 десятичных знаков.

Math.E – основание натурального логарифма с точностью в 15 десятичных знаков.

```
System.out.println(Math.PI);
System.out.println(Math.E);
```

```
3.141592653589793
2.718281828459045
```

Прямые трансцендентные методы

Метод	Описание
`double sin(double arg)`	Возвращает синус угла **arg**, переданного в **радианах**
`double cos(double arg)`	Возвращает косинус угла **arg**, переданного в **радианах**
`double tan(double arg)`	Возвращает тангенс угла **arg**, переданного в **радианах**

Прямые трансцендентные методы. Пример

```
System.out.println(Math.sin(Math.PI/2));
System.out.println(Math.cos(Math.PI/2));
System.out.println(Math.tan(Math.PI/4));
```

```
1.0
6.123233995736766E-17
0.9999999999999999
```

Значение близкое к нулю

Значение близкое к единице

Почему не 0 и 1 ?

Обратные трансцендентные методы

Метод	Описание
`double asin(double arg)`	Возвращает угол, синус которого равен **`arg`**.
`double acos(double arg)`	Возвращает угол, косинус которого равен **`arg`**.
`double atan(double arg)`	Возвращает угол, тангенс которого равен **`arg`**.
`double atan2(double x, double y)`	Возвращает угол, тангенс которого равен **`x/y`**.

Обратные трансцендентные методы. Пример

```
System.out.println(Math.asin(1)*2);
System.out.println(Math.acos(1));
System.out.println(Math.atan(0));
System.out.println(Math.atan2(1,1)*4);
```

```
3.141592653589793
0.0
0.0
3.141592653589793
```

Гиперболические методы

Метод	Описание
`double sinh(double arg)`	Возвращает гиперболический синус угла **`arg`**, переданного в **радианах**.
`double cosh(double arg)`	Возвращает гиперболический косинус угла **`arg`**, переданного в **радианах**.
`double tanh(double arg)`	Возвращает гиперболический тангенс угла **`arg`**, переданного в **радианах**.

Экспоненциальные методы

Метод	Описание
`double exp(double arg)`	Возвращает экспоненту **arg**.
`double log(double arg)`	Возвращает натуральный логарифм **arg**.
`double log10(double arg)`	Возвращает логарифм по основанию 10 от **arg**.
`double pow(double y, double x)`	Возвращает **y** в степени **x**
`double sqrt(double arg)`	Возвращает квадратный корень из **arg**.

Экспоненциальные методы. Пример

```
System.out.println(Math.exp(1));
System.out.println(Math.exp(2));
System.out.println(Math.log(1));
System.out.println(Math.log(Math.E));
System.out.println(Math.log10(1000));
System.out.println(Math.pow(2, 3));
System.out.println(Math.sqrt(25));
```

```
2.7182818284590455
7.38905609893065
0.0
1.0
3.0
8.0
5.0
```

Абсолютное значение

Метод	Описание
`int abs(int arg)`	Возвращает абсолютное значение **arg**.
`long abs(long arg)`	Возвращает абсолютное значение **arg**.
`float abs(float arg)`	Возвращает абсолютное значение **arg**.
`double abs(double arg)`	Возвращает абсолютное значение **arg**.

Чем эти методы отличаются?

Абсолютное значение. Пример

```
System.out.println(Math.abs(5));
System.out.println(Math.abs(-5));
System.out.println(Math.abs(10.3));
System.out.println(Math.abs(-10.3));
```

```
5
5
10.3
10.3
```

Методы округления

Метод	Описание
`double ceil(double arg)` потолок	Возвращает наименьшее целое число, которое больше **`arg`**.
`double floor(double arg)` пол	Возвращает наибольшее целое число, которое меньше или равно **`arg`**.
`int round(float arg)`	Возвращает **`arg`**, округленное до ближайшего **`int`**.
`long round(double arg)`	Возвращает **`arg`**, округленное до ближайшего **`long`**.

Методы округления. Пример

потолок

```
System.out.println(Math.ceil(5.4));
System.out.println(Math.floor(5.4));
System.out.println(Math.round(5.4));
System.out.println(Math.round(5.6));
System.out.println(Math.round(5.5));
```

```
6.0
5.0
5
6
6
```

пол

Максимум

Метод	Описание
`int max(int x, int y)`	Возвращает большее из двух чисел **x** и **y**.
`long max(long x, long y)`	Возвращает большее из двух чисел **x** и **y**.
`float max(float x, float y)`	Возвращает большее из двух чисел **x** и **y**.
`double max(double x, double y)`	Возвращает большее из двух чисел **x** и **y**.

Минимум

Метод	Описание
`int min(int x, int y)`	Возвращает меньшее из двух чисел **x** и **y**.
`long min(long x, long y)`	Возвращает меньшее из двух чисел **x** и **y**.
`float min(float x, float y)`	Возвращает меньшее из двух чисел **x** и **y**.
`double min(double x, double y)`	Возвращает меньшее из двух чисел **x** и **y**.

Максимум и минимум. Пример

```
System.out.println(Math.max(2, 4));
System.out.println(Math.min(2, 4));
System.out.println(Math.max(10.3, 4));
System.out.println(Math.min(10.3, 4));
```

```
4
2
10.3
4.0
```

Почему 4.0, а не 4?

Как вычислить максимум из трех чисел?

Вспомогательные функции

Метод	Описание
`double toDegrees(double angle)`	Преобразует радианы в градусы. Переданный в **`angle`** угол должен быть указан в радианах. Возвращается результат в градусах.
`double toRadians(double angle)`	Преобразует градусы в радианы. Переданный в **`angle`** угол должен быть указан в градусах. Возвращается результат в радианах.

Вспомогательные функции. Пример

```
System.out.println(Math.toDegrees(Math.PI));
System.out.println(Math.toDegrees(Math.PI/4))
;
System.out.println(Math.toRadians(180));
System.out.println(Math.toRadians(90));
```

```
180.0
45.0
3.141592653589793
1.5707963267948966
```

Псевдослучайные числа

Метод **`Math.random()`** возвращает псевдослучайное вещественное число из промежутка [0;1).

```
System.out.println(Math.random());
System.out.println(Math.random());
System.out.println(Math.random());
```

```
0.8701659383706429
0.5194884184661862
0.3324845299964946
```

Случайные числа

Целые числа в заданном интервале – 1

Целые числа в интервале [0,n-1]:

```
(int)(Math.random()* n);
```

Примеры:

```
x = (int)(Math.random () * 100); // [0,99]
x = (int)(Math.random () * z);   // [0,z-1]
```

Целые числа в интервале [a,b]:

```
x = (int)(Math.random()*(b - a + 1)) + a;
// [a,b]
```

Целые числа в заданном интервале – 2

Задача. Получить случайное число в интервале от -10 до 10.

```
int x = (int)(Math.random() * 21) - 10;
```

Методы класса Math. Задача – 1

```
System.out.println(Math.abs(-2.33));
System.out.println(Math.round(Math.PI));
System.out.println(Math.round(9.5));
System.out.println(Math.round(9.5-0.001))
;
System.out.println(Math.ceil(9.4));
double c = Math.sqrt(3*3 + 4*4);
System.out.println(c);
double s1 = Math.cos(Math.toRadians(60));
System.out.println(s1);
```

```
2.33
3
10
9
10.0

5.0

0.5
```

Методы класса Math. Задача – 2

Записать в стандартной форме записи числа

-12.3E+2 **= -1230.0**

-0.8E-6 **= -0.0000008**

1E+3 **= 1000.0**

+1E-6 **= 0.000001**

Методы класса Math. Задача – 3

Какие круглые скобки можно убрать, не изменив порядка вычисления выражений

```
(a+b)/c
a+b/c
a/(b*c)
x1/x2*y
Math.sqrt(p)*q/r
a-b-c-d-e
(a-b)-(c-d)-e
```

Методы класса Math. Задача – 4

Записать следующие выражения на Java

x^5

```
Math.pow(x,5)
```

$\cos^8 x^4$

```
Math.pow(Math.cos(Math.pow(x,4)),8
)
```

$\log_{10}(x/5)$

```
Math.log10(x/5
)
```

$|x^{-3}|$

```
Math.abs(Math.pow(x,-3)
)
```

2^{x+1}

```
Math.pow(2,x+1
)
```

$\sin 8°$

```
Math.sin(Math.toRadians(8))
```

Методы класса Math. Задача – 5

Определить типы выражений

```
double x, y, z;
int i, j, k;
x+y*i;
i+j-k;
i/j+x;
i*x+j*y;
```

double

int

double

double

Задача

Задача. Дано целое число **x**. Вывести количество цифр данного числа.

```
int x;
x = in.nextInt();
double count = Math.floor(Math.log10(x)) +
1;
System.out.println(count);
```

Что плохо?

Как обрабатывать отрицательные значения?

Как обрабатывать 0?