

## Требования к проверяемой реализации функции вычисления квадратного корня

### Числа с плавающей точкой

Двоичные числа с плавающей точкой представляются в виде массивов бит длиной  $n$ , которые делятся на три части: один знаковый бит  $S$ , порядок  $E$  из  $k$  бит, мантисса  $M$  из  $(n-k-1)$  бит. Представленное число при этом вычисляется по следующим правилам. Используемое в них число  $B = 2^{k-1}-1$  называется смещением порядка.

- если  $E \neq 0$  и  $E \neq 2^k-1$  (порядок не состоит из одних нулей или одних единиц)  
$$x = (-1)^S \cdot 2^{(E-B)} \cdot (1 + M/2^{n-k-1})$$
  
это нормализованные числа
- если  $E = 0$   
$$x = (-1)^S \cdot 2^{-(B+1)} \cdot (M/2^{n-k-1})$$
  
это денормализованные числа (при ненулевой мантиссе) и  $\pm 0$  (при нулевой мантиссе, считается, что  $+0 = -0$ , однако некоторые операции могут по-разному обрабатывать  $+0$  и  $-0$ )
- если  $E = 2^k-1$   
при  $M = 0$ ,  $x = (-1)^S \cdot \infty$  (используются для представления бесконечных, например,  $+1/+0 = +\infty$  и  $-1/+0 = -\infty$ , или слишком больших по абсолютной величине результатов)  
при  $M \neq 0$ ,  $x = \text{NaN}$  (не-число, используется для представления результатов, которым нельзя согласованно с остальными правилами приписать конечное или бесконечное значение, например, результат деления  $0$  на  $0$ ).

Для типа `double` (числа с плавающей точкой двойной точности) используются параметры  $n = 64$ ,  $k = 11$ .

### Требования к `sqrt`

1. Результат вычисления функции `double sqrt (double x)` в обычной ситуации должен быть точным математическим результатом, корректно округленным к ближайшему представимому в рамках типа `double`. Если есть два числа типа `double`, находящихся на одном расстоянии от точного результата, ближайшим считается то, которое имеет в качестве последнего бита мантиссы  $0$  (округление к ближайшему четному).
2. Результатом вычисления `sqrt` с отрицательным аргументом (конечным или бесконечным, но не  $-0$ ) должно быть `NaN`.
3. Результатом вычисления `sqrt` с аргументом  $-0$  должен быть  $-0$ .
4. Результатом вычисления `sqrt` с аргументом  $+\infty$  должна быть  $+\infty$ .
5. Результатом вычисления `sqrt` с аргументом `NaN` должно быть `NaN`.

### Задание

Разработать набор тестов с использованием библиотеки **TestNG** для реализации функции вычисления квадратного корня функции `double sqrt (double x)` в классе `root.sqrt.AdvSqrt`.

Набор тестов должен покрывать все требования и все классы чисел с плавающей точкой, естественно выделяемые на основе их структуры (нормализованные, денормализованные, нули, бесконечности и `NaN`).

Кроме того, набор тестов должен обеспечивать покрытие всех ветвлений в коде и всех отдельных дизъюнктов в условиях ветвлений.

К разработанным тестам должен прилагать отчет по покрытию (JaCoCo Report – см. пример `examples/coverage`)

Тесты следует присылать в виде проекта с исходным кодом (исправленным если найдены ошибки), тестами и всеми библиотеками.

К архиву с проектом следует приложить список обнаруженных ошибок и исправлений (если были) по следующему формату

1. Код до исправления
2. Данные на которых наблюдается некорректное поведение
3. Полученное значение, ожидаемое значение
4. Код после исправления

## Замечания

Для конвертации числа типа `double` в битовое представление и обратно в Java используются методы `Double.doubleToLongBits(double x)` и `Double.longBitsToDouble(long n)`.

Реализация устроена следующим образом.

Сначала обрабатываются специальные случаи (отрицательные числа, NaN, бесконечности и нули).

Затем, если аргумент денормализованный, он умножается на  $2^{52}$ , чтобы стать нормализованным числом.

После этого порядок аргумента заменяется на 0 или -1.

Квадратный корень из полученного числа в интервале  $[1/2, 2)$  вычисляется с помощью метода Ньютона, уточняющие итерации продолжаются, пока относительная разность между квадратом результата и аргументом не станет не больше, чем величина последнего бита мантииссы ( $2^{-52} \approx 2.25e-16$ ). Благодаря смещению в указанный интервал, количество выполняемых итераций ограничено 5.

В конце выполняется замена порядка полученного результата в соответствии с исходным порядком аргумента.

Результат, получаемый данной реализацией либо является ближайшим к математическому результату числом типа `double`, либо отстоит от такого числа на один шаг (отличается на величину последнего бита мантииссы). Для получения корректно округленного значения можно использовать результаты `java.lang.Math.sqrt()`.