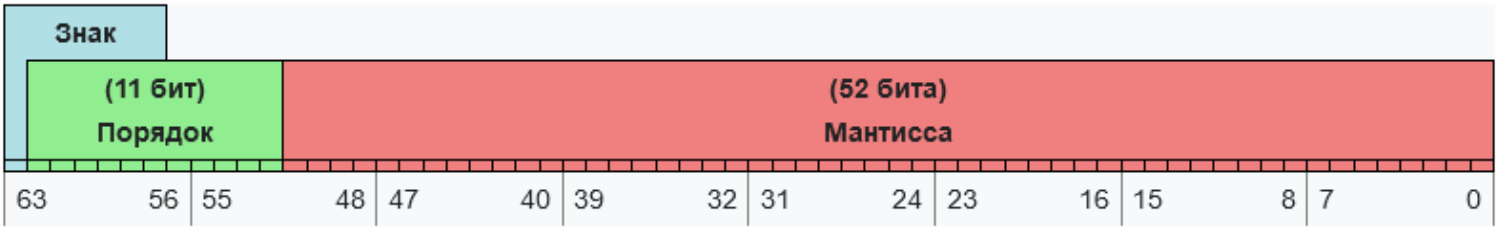
СОДЕРЖАНИЕ

ИНФОРМАТИКА 2

$\boldsymbol{exput(double\&\ x, short\ a)}$: Смена порядка $\boldsymbol{double}$ числа $\boldsymbol{x}$ на $\boldsymbol{a}$ с возвращением разницы конечного и начального порядков

Число двойной точности на битовом уровне представляется следующим образом:



Для порядков больших 00000000000 и меньших 11111111111 в число в десятичной системе счисления равно $2^{\text{порядок}-1023} \cdot (1, \text{мантисса})$. В связи с этим умножение на 2^n для целых n эквивалентно прибавлению n к битам порядка, а деление – вычитанию. Примем, что $a = \text{порядок} - 1023$.

Если был получен $y > 1024$, или $y < -1024$, вызвать исключение $OutOfDomain$.

$\boldsymbol{x2(double\ x, short\ a)}$: Быстрое умножение на $\boldsymbol{2^a}$

1) Для порядков больших 00000000000 и меньших 11111111111 в число в десятичной системе счисления равно $2^{\text{порядок}-1023} \cdot (1, \text{мантисса})$. В связи с этим умножение на $\boldsymbol{2^a}$ для целых $\boldsymbol{a}$ эквивалентно прибавлению $\boldsymbol{a}$ к битам порядка.

2) Если имелся порядок 11111111111, то умножение следует отменить и вернуть имеющееся число, равное +inf, -inf, или NaN. Если порядок 11111111111 получается в результате умножения, то необходимо обнулить биты мантиссы, что будет соответствовать +inf или -inf в зависимости от знака числа.

3) Если имелся или был получен порядок 00000000000, то число приобретает вид $2{-}1023 \cdot (0,$ мантисса$)$. Умножение числа на отрицательную степень двойки будет эквивалентно сдвигу бит мантиссы вправо. Умножение числа на положительную будет эквивалентно сдвигу бит мантиссы влево. Если при этом ведущий бит мантиссы равен 1, то порядок становится равен единице и дальнейшее умножение проводится в соответствии с п. 1.

$\boldsymbol{sgn}(\boldsymbol{double\ x})$: Знак числа

$$sgn(x) = \begin{cases} -1; x < 0 \\ 0; x = 0 \\ 1; x > 1 \end{cases}$$

$\boldsymbol{abs}(\boldsymbol{double\ x})$: Модуль числа

$$abs(x) = \begin{cases} -x; x < 0 \\ x; x \geq 0 \end{cases}$$

$\boldsymbol{floor}(\boldsymbol{double\ x})$: Целая часть $\boldsymbol{double}$ числа, она же округление вниз

Для порядков больших 00000000000 и меньших 11111111111 в число в десятичной системе счисления равно $2^{\text{порядок}-1023} \cdot (1, \text{мантисса})$.

1) Число с порядком больше 1074 полностью целое.

1) Число с порядком $\in [1023, 1074]$ имеет в качестве целого числа следующее число: [знак]. [порядок]. [(порядок $-$ 1023) битов мантиссы, дополненные нулями].

2) Число с порядком меньше 1023 полностью дробное, целая часть равна 0.

$\boldsymbol{mant}(\boldsymbol{double\ x})$: Дробная часть $\boldsymbol{double}$ числа

$$mant(x) = x - floor(x)$$

$\boldsymbol{ceil}(\boldsymbol{double\ x})$: Округление вверх

$$ceil(x) = \begin{cases} x; floor(x) = x \\ floor(x) + 1; floor(x) \neq x \end{cases}$$

$\boldsymbol{round}(\boldsymbol{double\ x})$: Округление к ближайшему целому

$$round(x) = \begin{cases} floor(x); x - floor(x) < 0.5 \\ floor(x) + 1; x - floor(x) \geq 0.5 \end{cases}$$

$\boldsymbol{order}(\boldsymbol{double\ x})$: Порядок

Возвращает истинное значение порядка, записанное в число x. Так, для $x = 4$ будет возвращено 2 ($4 = 2^2$).

$\boldsymbol{ipow(double\ x, long\ long\ a)}$: Быстрое возведение числа в целую степень

Число может быть умножено на себя с получением квадрата. Это может быть повторено несколько раз, например $x^2 = x \cdot x;\ x^4 = x^2 \cdot x^2; x^8 = x^4 \cdot x^4$ и так далее. Так, за n умножений можно получить максимальную степень 2^n, проходясь по степеням двойки.

Исходя из степеней двойки и из двоичной записи этой степени, можно возвести число в любую целую степень n благодаря свойству степеней $a^{b+c} = a^b \cdot a^c$.

Например, $5^5 = 5^{101_2} = 5^{1_2+00_2+100_2} = 5^{100_2} \cdot 5^{00_2} \cdot 5^{1_2} = 5^1 \cdot 5^0 \cdot 5^4 = 3125$.

Для отрицательных степеней нужно в конце выполнить одно деление: $5^{-5} = \frac{1}{5^5}$

$\boldsymbol{bin(double\ x)}$: Перевод $\boldsymbol{double}$ числа в бинарную строку

Наиболее эффективно произвести перевод можно, выполнив $reinterpret\_cast$ адреса числа в ссылку на $unsigned\ long\ long$, затем проверяя крайний левый бит числа, проводя сравнение число $\geq 0x8000000000000000$, а затем сдвигая число влево на 1 бит. Повторять 64 раза.

МАТАН 4

$\boldsymbol{ln(double\ x)}$: Натуральный логарифм $\boldsymbol{x}$

Натуральный логарифм раскладывается в ряд Маклорена:

$$\ln(x) = \ln(1 + x - 1) = [y = x - 1] = \ln(1 + y) = \sum_{n=1}^{\infty} \frac{(-1)^n}{n} y^n$$

Данный ряд сходится при выполнении следующего условия (признак Даламбера):

$$\left| \lim_{n\to\infty} \frac{\frac{(-1)^{n+1}}{n+1} y^{n+1}}{\frac{(-1)^n}{n} y^n} \right| < 1 => |y| < 1$$

Также данный ряд сходится условно при $y = 1$ (признак Лейбница).

Новая задача – уменьшение модуля числа y, а соответственно и x. Вернёмся к исходному выражению. Примем, что $x \geq 2$.

$$\ln(x) = \ln\left(2^{exput(x,-1)} \cdot x\right) = exput(x, -1)\ln(2) + \ln(1 + y)$$

Нам удалось уменьшить аргумент до необходимого и достаточного значения, $y \in (-1; 0)$. Наличие $\ln(2)$ не является проблемой, так как ряд сходится при $x = 2$, а значит вычислить его можно и так. В идеале, повторять, пока $n -$ ный член не обернётся машинным нулём – если такого не случится, провести необходимое на взгляд разработчика количество итераций. Для дальнейшего применения программой без вычисления заново полученный результат перевести в бинарную строку функцией bin, присвоить её $unsigned\ long\ long$, а затем произвести $reinterpret\_cast$ числа в ссылку на $double$.

Заметим также, что логарифм неположительного числа не определён, поэтому, если происходит попытка логарифмирования неположительного числа вызвать исключение $OutOfDomain$.

$\boldsymbol{log(double\ a, double\ x)}$: Логарифм по основанию $\boldsymbol{a}$ от $\boldsymbol{x}$

$$\log_a(x) = \frac{\ln(x)}{\ln(a)}$$

$\boldsymbol{exp(double\ x)}$: Экспонента

Экспонента раскладывается в ряд Маклорена:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Данный ряд сходится для любых x, однако чем больше x, тем больше погрешность. Для получения наиболее точного результата необходимо уменьшить степень. Сделаем это следующим образом:

$$e^x = e^{floor(x)+mant(x)} = e^{floor(x)}e^{mant(x)} = ipow\big(e, floor(x)\big)e^{mant(x)}$$

Нам удалось понизить аргумент так, чтобы по модулю он был меньше единицы. В выражении необходимо непосредственно число Эйлера, однако его можно посчитать напрямую. В идеале, повторять, пока $n-$ный член не обернётся машинным нулём – если такого не случится, провести необходимое на взгляд разработчика количество итераций. Для дальнейшего применения программой без вычисления заново полученный результат перевести в бинарную строку функцией bin, присвоить её $unsigned\ long\ long$, а затем произвести $reinterpret\_cast$ числа в ссылку на $double$.

$\boldsymbol{pow(double\ x, double\ a)}$: Возведение $\boldsymbol{x}$ произвольную степень $\boldsymbol{a}$

Для целых a применить $ipow(x, a)$.

Для нецелых a:

1. $x = 0/1 => x^a = 0/1$

2. $x > 0 => x^a = e^{aln(x)}$

3. Для отрицательных чисел не существует однозначного способа получения корректного результата. Покажем на примере: $(-2)^{\frac{3}{2}} = \left(\sqrt[2]{-2}\right)^3 -$ не определён, однако $(-2)^{\frac{3}{5}} = \left(\sqrt[5]{-2}\right)^3 -$ определённое значение. В десятичной системе счисления вычислимость корня отрицательного числа определяется чётностью корня: нечётный – определено, чётный – не определено.

Имея десятичную дробь вычислимость можно определить следующим образом: пусть имеется степень в виде десятичной непериодической дроби $a \in (0; 1)$. Представим её в виде:

$$a = \frac{x \cdot 2^n}{y \cdot 2^m}$$

где $x, y, n, m -$ целые числа, $n, m -$ максимально возможные неотрицательные целые числа. Если чётность числителя больше или равна чётности знаменателя ($n \geq m$), то степень вычислима, так как знаменатель станет нечётным, иначе не вычислима. Например, для $x^{0,72}$:

$$a = 0.72 = \frac{72}{100} = \frac{9 \cdot 2^3}{25 \cdot 2^2} = \frac{18}{25} => x^{0,72} = \left(\sqrt[25]{x}\right)^{18} => x - \text{любой}$$

А для $x^{0.5}$:

$$a = 0.5 = \frac{5}{10} = \frac{5 \cdot 2^0}{5 \cdot 2^1} = \frac{1}{2} => x^{0,5} = \left(\sqrt[2]{x}\right)^1 => x \geq 0$$

Проблема заключается в том, что если таким образом проверять чётности для чисел $double$, результат получится совершенно бессмысленным, так как ненулевая дробная часть $double$ числа состоит из суммы отрицательных степеней двойки, которые всегда заканчиваются на пять и эта пятёрка никогда не находится в одном и том же месте, например:

$$0{,}875 = 0{,}5 + 0{,}25 + 0{,}125$$

Соответственно числитель такой дроби всегда имеет нулевую чётность, а числитель всегда имеет большую:

$$a = 0.875 = \frac{875}{1000} = \frac{875 \cdot 2^0}{125 \cdot 2^3} => 0 < 3 => x \geq 0$$

Таким образом, дробную степень, которую очевидно можно посчитать из отрицательного числа, например, корень третьей степени, в $double$ представить невозможно.

Поэтому принято решение использовать два режима вычисления степени, которые могут переключаться пользователем и хранятся в булевой переменной $negAllow$.

При $false$ полностью запрещать вычисление нецелой степени отрицательного числа и вызывать исключение $OutOfDomain$.

При $true$ для отрицательных x использовать формулу $x^a = -e^{aln(x)}$.

$\boldsymbol{root(double\ a, double\ x)}$: Корень из $\boldsymbol{x}$ произвольной степени $\boldsymbol{a}$

$\sqrt[a]{x} = x^{\frac{1}{a}}$

$\boldsymbol{sqrt(double\ x)}$: Корень квадратный из $\boldsymbol{x}$

$sqrt(x) = root(2, x)$

$\boldsymbol{arctg(double\ x)}$: Арктангенс

Арктангенс раскладывается в ряд Маклорена:

$$\operatorname{arctg}(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1}$$

Данный ряд сходится при выполнении следующего условия (признак Даламбера):

$$\left| \lim_{n\to\infty} \frac{\frac{(-1)^{n+1}}{2n+3} x^{2n+3}}{\frac{(-1)^n}{2n+1} x^{2n+1}} \right| < 1 \Rightarrow |x| < 1$$

Также данный ряд сходится условно при $x = 1; -1$ (признак Лейбница).

Возникает задача уменьшения модуля аргумента. Начнём со свойства:

$$\operatorname{arctg}(-x) = -\operatorname{arctg}(x)$$

Теперь можно не беспокоясь о знаке понижать модуль числа. Для этого воспользуемся свойством:

$$\operatorname{arctg}(x) = 2\operatorname{arctg}\left(\frac{x}{1+\sqrt{1+x^2}}\right)$$

Повторять, пока аргумент не станет меньше единицы. Однако, несмотря на рекуррентную зависимость, не следует вычислять функцию, применяя рекурсию. Лучше вычислить количество итераций до понижения аргумента, а затем вычислить арктангенс от этого одного аргумента, а затем применить умножить на $2^{\text{количество итераций}}$ с помощью $x2(\text{полученный арктангенс}, \text{количество итераций})$.

Также, с помощью арктангенса следует вычислить π, которое пригодится позднее. Сделать это следует по формуле $\pi = 4\operatorname{arctg}(1)$. В идеале, повторять, пока $n-$ный член не обернётся машинным нулём – если такого не случится, провести необходимое на взгляд разработчика количество итераций. Для дальнейшего применения программой без вычисления заново полученный результат перевести в бинарную строку функцией bin, присвоить её $unsigned\ long\ long$, а затем произвести $reinterpret\_cast$ числа в ссылку на $double$.

$\boldsymbol{arcctg(double\ x)}$: Арккотангенс

$$\operatorname{arcctg}(x) = \frac{\pi}{2} - \operatorname{arctg}(x)$$

$\boldsymbol{arcsin(double\ x)}$: Арксинус

$$\arcsin(x) = 2\operatorname{arctg}\left(\frac{x}{1+\sqrt{1-x^2}}\right)$$

$\boldsymbol{arccos(double\ x)}$: Арксинус

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x)$$

$\boldsymbol{sin(double\ x)}$: Синус

Синус раскладывается в ряд Маклорена:

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Данный ряд сходится для любых x, однако чем больше x, тем больше погрешность. Для получения наиболее точного результата необходимо уменьшить аргумент. Начнём со свойства:

$$\sin(-x) = -\sin(x)$$

Теперь можно не беспокоясь о знаке понижать модуль числа. Примем, что $x \geq 2\pi$ и воспользуемся периодичностью синуса.

$$\sin(x) = \sin\left(x - 2\pi floor\left(\frac{x}{2\pi}\right)\right)$$

Теперь $x \in (0; 2\pi)$, однако его можно уменьшить дальше. Примем, что $x \geq \pi$. Тогда:

$$\sin(x) = -\sin(x - \pi)$$

Теперь $x \in (0; \pi)$, однако его можно уменьшить дальше. Примем, что $x \geq \frac{\pi}{2}$. Тогда:

$$\sin(x) = \cos\left(x - \frac{\pi}{2}\right)$$

Иначе говоря, если получился угол $\left(0; \frac{\pi}{2}\right)$ – вычислять синус через ряд. Иначе вычислять косинус.

$\boldsymbol{cos(double\ x)}$: Косинус

Косинус раскладывается в ряд Маклорена:

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

Данный ряд сходится для любых x, однако чем больше x, тем больше погрешность. Для получения наиболее точного результата необходимо уменьшить аргумент. Начнём со свойства:

$$\cos(-x) = \cos(x)$$

Теперь можно не беспокоясь о знаке понижать модуль числа. Примем, что $x \geq 2\pi$ и воспользуемся периодичностью косинуса.

$$\cos(x) = \cos\left(x - 2\pi floor\left(\frac{x}{2\pi}\right)\right)$$

Теперь $x \in (0; 2\pi)$, однако его можно уменьшить дальше. Примем, что $x \geq \pi$. Тогда:

$$\cos(x) = -\cos(x - \pi)$$

Теперь $x \in (0; \pi)$, однако его можно уменьшить дальше. Примем, что $x \geq \frac{\pi}{2}$. Тогда:

$$\cos(x) = -\sin\left(x - \frac{\pi}{2}\right)$$

Иначе говоря, если получился угол $\left(0; \frac{\pi}{2}\right)$ – вычислять косинус через ряд. Иначе вычислять синус.

$\boldsymbol{tg(double\ x)}$: Тангенс

$$\mathrm{tg}(x) = \frac{\sin(x)}{\cos(x)}$$

$\boldsymbol{ctg(double\ x)}$: Котангенс

$$\mathrm{ctg}(x) = \frac{\cos(x)}{\sin(x)}$$

СИАОД 3

Бинарное дерево выражений, хранящееся в глобальной переменной $\boldsymbol{f}$:

Дерево выражений, строящееся по выражению, заданному в виде строки.

Структура узла дерева: массив[2] указателей на дочерние узлы, значение double. Значение узла определяется по следующим правилам:

1. Оба дочерних узла не nullptr: бинарный оператор. Каждый оператор закодирован значением. 0 – сложение (+), 1 – вычитание (-), 2 – умножение (*), 3 – деление (/), 4 – степень (^) 5 – корень произвольной степени ($root(a,x)$), 6 – логарифм произвольного основания ($log(a,x)$).

2. Только второй узел nullptr: унарный оператор. Каждый оператор закодирован значением. 0 – унарный минус (-), 1 – модуль ($|x|$), 2 – натуральный логарифм ($ln(x)$), 3 – экспонента ($exp(x)$), 4 – корень квадратный ($sqrt(x)$), 5 – синус ($sin(x)$), 6 – косинус ($cos(x)$), 7 – тангенс ($tg(x)$), 8 – котангенс ($ctg(x)$), 9 – арксинус ($arcsin(x)$), 10 – арккосинус ($arccos(x)$), 11 – арктангенс ($arctg(x)$), 12 – арккотангенс ($arcctg(x)$).

3. Оба узла nullptr: число. Если значение равно особому сигнализирующему NaN (0x7ff8000000000000) – переменная x, иначе число, равное указанному значению.

Порядок действий определяется в соответствие с обыкновенными правилами математики.

Значение функции, заданной бинарным деревом выражений, нужно получать при помощи оператора [], условный синтаксис $double\ value = f[1.337]$. При вычислении значения обход дерева в глубину выполнять итерационно, не рекурсивно, для этого реализовать простенький стек.

Смена функции сопровождается удалением текущей без утечек памяти и составлением нового дерева.

$\boldsymbol{verify(string\ func)}$: Верификация выражения

Возвращает $true$ или $false$ в зависимости от корректности выражения. Проверять следующие правила:

1. Баланс круглых и модульных скобок. Каждая неоператорная функция требует аргументы строго в скобках, независимо от сложности аргумента, например $\sin(x\text{^}2)$, $\sin(x)$.

2. Орфография. Каждая функция должна иметь своё верное название, например, $\text{arctg}(x)$ – верно, $\text{arct}(x)$ – неверно, $\text{arctan}(x)$ – неверно. Между всеми операторами, кроме скобок, должен стоять пробел. Переменная x указывается, как “x”.

Пример верного выражения со всеми функциями:

-213 + ((105 – x) * (2 / x) ^ [ПЕРЕНОС ДЛЯ ЧИТАЕМОСТИ, НА ДЕЛЕ СТРОКА ОДНА]
root(2, log(3, |ln(exp(sqrt(sin(cos(tg(ctg(arcsin(arccos(arctg(arcctg(x))))))))))|)))

После верификации выражение можно отправлять в бинарное дерево выражений или уведомлять об ошибке.

$parse(string\ com)$: Парсер команд

Парсит полученную команду и выполняет необходимый для её исполнения код. Если команда подана неверно, вывести сообщение об ошибке. Допустимые команды:

- new – войти в режим задания функции. Выводит на следующую строку ”f(x) = ” и ждёт от пользователя выражение, которое верифицируется, и, если оно безошибочно, строит по нему дерево, иначе уведомляет пользователя об ошибке. Этот режим также открывается сразу же при входе в программу.
- f([ЧИСЛО]) – вычисляет значение функции от указанного числа с помощью оператора []. Если в результате получено исключение $OutOfDomain$, вывести сообщение об ошибке.
- f’([ЧИСЛО]) – вычисляет значение производной от указанного числа с помощью функции $derive(x)$. Если в результате получено исключение $OutOfDomain$, вывести сообщение об ошибке.
- f’([ЦЕЛОЕ ЧИСЛО n], [ЧИСЛО]) – вычисляет значение n-ной производной от указанного числа с помощью функции $derive(n, x)$. Если в результате получено исключение $OutOfDomain$, вывести сообщение об ошибке.
- sf([ЧИСЛО1], [ЧИСЛО2]) – решает уравнение на указанном промежутке с помощью функции $solve(a, b)$. Если в результате получено исключение $OutOfDomain$, вывести сообщение об ошибке. Если в результате получено исключение $SolutionFail$, вывести сообщение о неудавшемся решении.
- if([ЧИСЛО1], [ЧИСЛО2]) – находит интеграл на указанном промежутке с помощью функции $integrate(a, b)$. Если в результате получено исключение $OutOfDomain$, вывести сообщение об ошибке.
- mode – меняет режим дробных степеней $negAllow$ на противоположный. По умолчанию $false$.

ВЫЧМАТ 2

$\boldsymbol{derive(double\ x)}$: Производная в точке $\boldsymbol{x}$

Для вычисления производной сначала попробовать более качественное приближение:

$$f'(x) = \frac{f[x+h] - f[x-h]}{2h}$$

Следует подбирать h так, чтобы это был минимальный h, прибавление и вычитание которого влияло бы на x, и при этом не вызвало погрешностей, связанных с неточностью чисел с плавающей точкой. Оптимальным вариантом будет единица в середине мантиссы того же порядка. Например, пусть $x = 1$, имеет порядок 0. В этом случае $h = 2^{-26}$ $\left(\left\lfloor\frac{\text{порядок}-\text{длина мантиссы}}{2}\right\rfloor = \left\lfloor\frac{\text{порядок}}{2}\right\rfloor - 26\right)$. Если в результате получился $\text{порядок} < -1023$, установить порядок -1023. Отобразим это на формуле:

$$f'(x) = \begin{bmatrix} h = 1 \\ a = \dfrac{order(x)}{2} - 26 \\ a = a < -1023\ ?\ -1023 : a \\ exput(h, a) \end{bmatrix} = 2x((f[x+h] - f[x-h]), -a - 1)$$

Если значение функции найти не удалось из-за исключения $OutOfDomain$, попробовать приближение похуже, но требующее определённости функции на меньшем участке:

$$f'(x) = \frac{f[x+h] - f[x]}{h} = \begin{bmatrix} h = 1 \\ a = \dfrac{order(x)}{2} - 26 \\ a = a < -1023\ ?\ -1023 : a \\ exput(h, a) \end{bmatrix} = 2x((f[x+h] - f[x]), -a)$$

Если и тут было получено то же исключение, ничего не делать, оно будет обработано выше.

$\boldsymbol{derive(double\ x, unsigned\ char\ n)}$: n-ная производная в точке $\boldsymbol{x}$

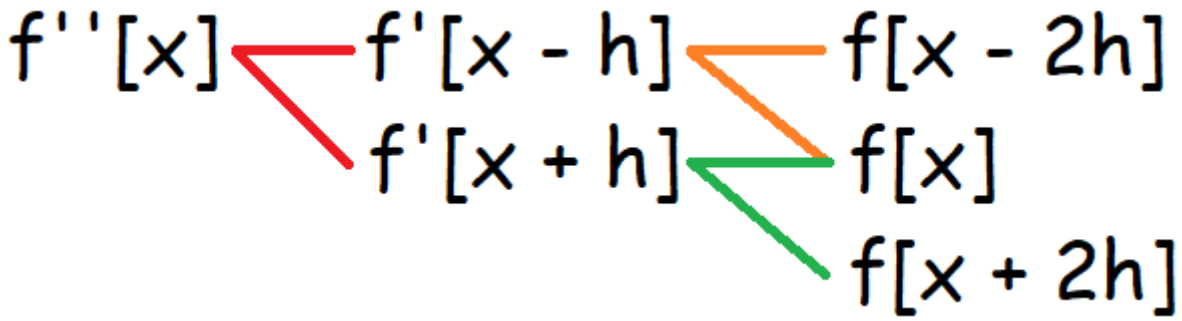
Воспользуемся тем, что n-ная производная является производной (n-1)-ой производной, каждая из которых может быть получена численным методом. Разумеется, $n \in [1; 255]$, если это не так, вызвать исключение $OutOfDomain$.

Рассмотрим на примере второй (чётной) и третьей (нечётной) производных.

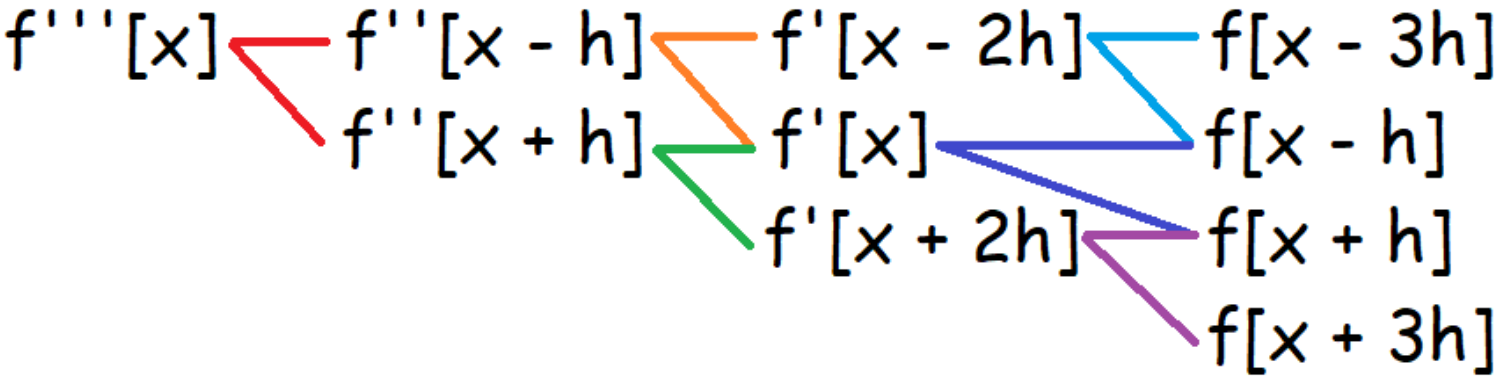
Сначала попытаться использовать более качественное приближение:

$$f'(x) = \frac{f[x+h] - f[x-h]}{2h}$$

Для второй производной последовательность необходимых значений функций и производных выглядит вот так:



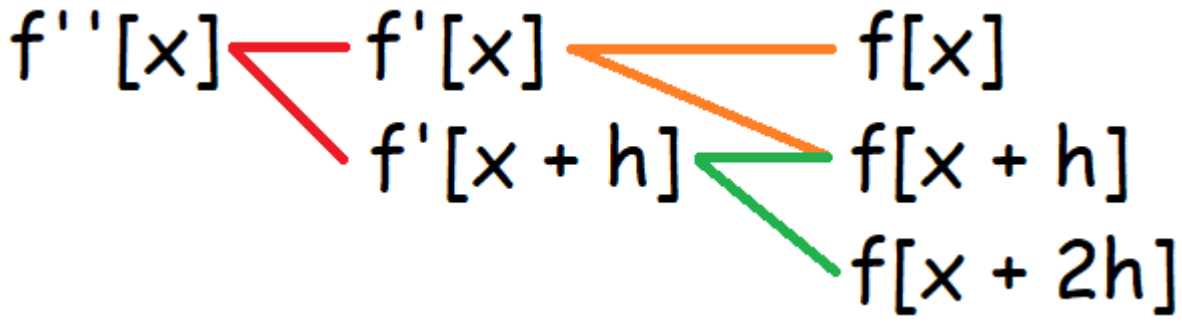
Для третьей:



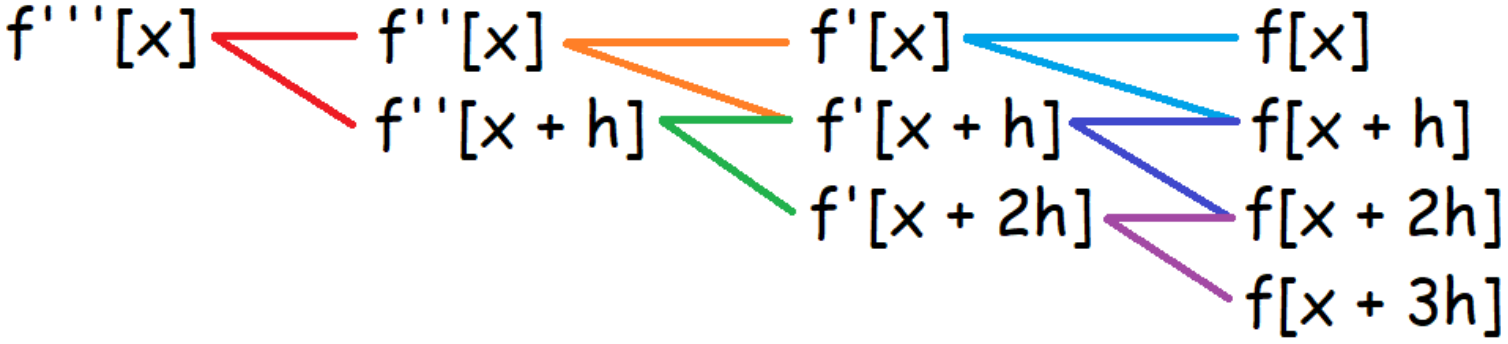
Если значение функции найти не удалось из-за исключения $OutOfDomain$, попробовать приближение похуже, но требующее определённости функции на меньшем участке:

$$f'(x) = \frac{f[x+h] - f[x]}{h}$$

Тогда последовательность для второй производной выглядит так:



Для третьей:



Если и тут было получено то же исключение, ничего не делать, оно будет обработано выше.

$\boldsymbol{solve(double\ a, double\ b)}$: Решение уравнения $\boldsymbol{f(x) = 0}$

Примем интервал $[a, b], b > a$, а также у чисел разные знаки. Если одно из условий не выполняется, вызвать исключение $SolutionFail$. Также следует проверить, являются ли a или b корнями уравнения.

Для решения уравнения $f(x) = 0$ сначала попробовать использовать итерационный метод Ньютона:

$$x_0 = a$$

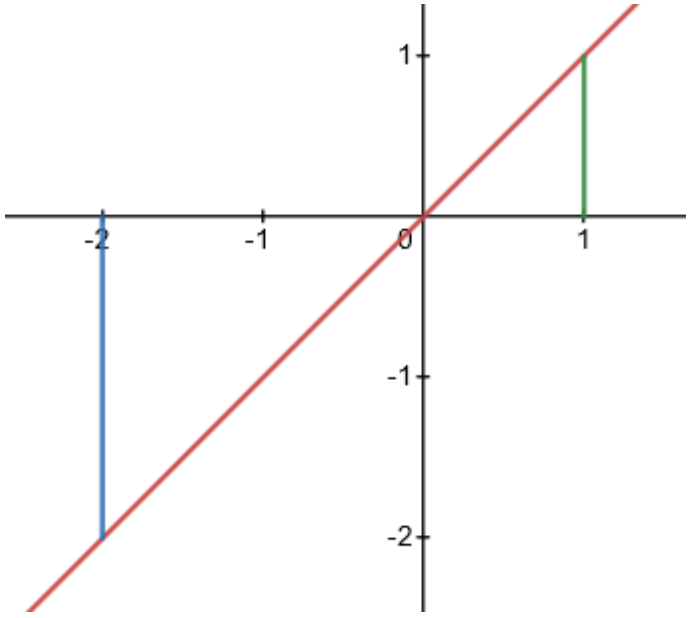
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

В идеале повторять, пока $\frac{f(x_i)}{f'(x_i)}$ не обернётся машинным нулём. Если это не происходит, выбрать нужное количество итераций на усмотрение разработчика. Если было получено исключение $OutOfDomain$, или x_i вышел за пределы интервала, попробовать другое начальное приближение:

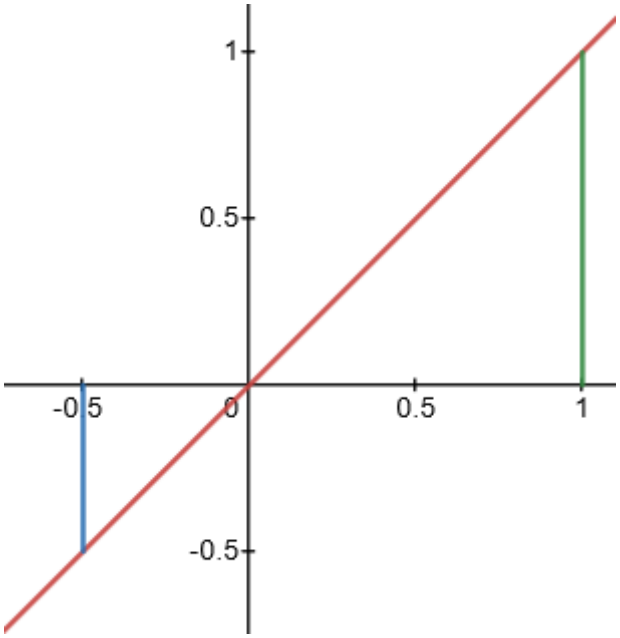
$$x_0 = b$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

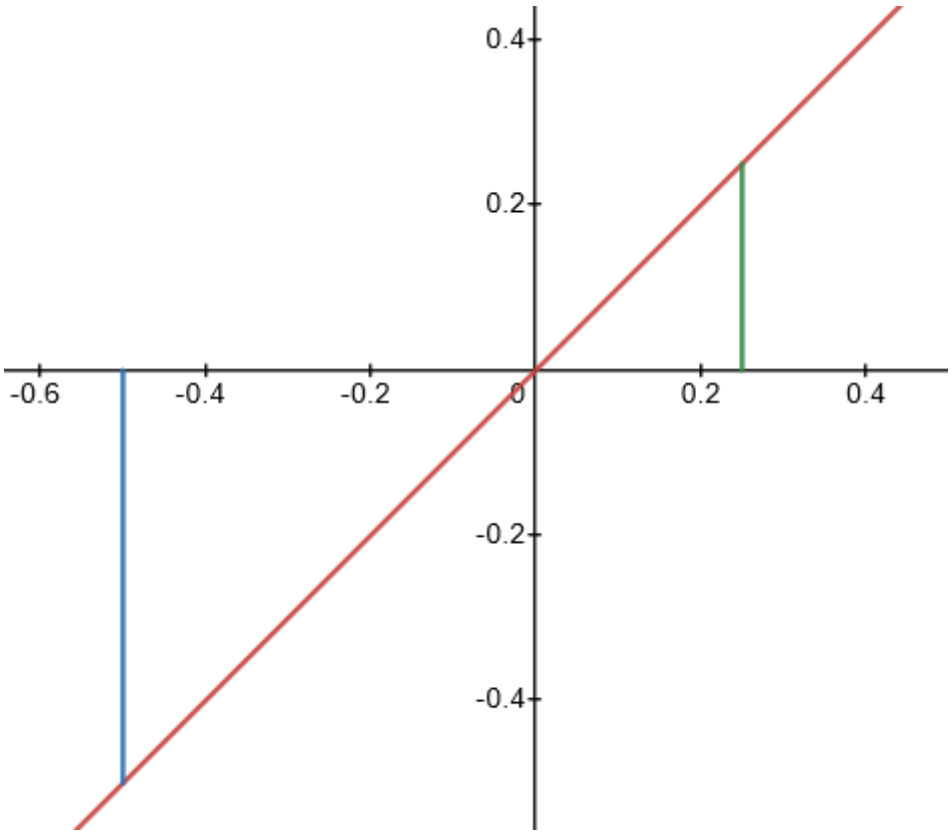
Если опять произошла ошибка, попробовать самый простой метод решения уравнений, не требующий вычисления производных – метод половинного деления. Метод трудно описать формально, поэтому продемонстрируем картинками. Пусть мы пытаемся решить уравнение $x = 0$ на $[-2,1]$. Синяя и зелёная прямые – концы отрезков, красная – $f(x)$.



Примем начальное приближение $x_0 = a = -2$. Метод половинного деления пользуется тем, что корень ”где-то посередине”, находим эту середину $\frac{-2+1}{2} = -\frac{1}{2}$. Вычисляем значение функции и её знак, он в этой точке тот же, что и в a (-), заменяем a на $-\frac{1}{2}$.



Находим новую середину $\frac{-\frac{1}{2}+1}{2} = \frac{1}{4}$. Значение функции в этой точке имеет тот же знак, что и в b (-), заменяем b на $\frac{1}{4}$.



Таким образом мы не самыми быстрыми шагами, но приближаемся к истинному значению корня. Количество итераций на усмотрение разработчика.

Если было получено исключение $OutOfDomain$, попробовать другое начальное приближение: $x_0 = b$. Если опять поднялось то же исключение, вызвать исключение $SolutionFail$.

$\boldsymbol{integrate(double\ a, double\ b)}$: Определённый интеграл

Примем интервал $[a, b], b > a$. Если $a = b$, то вернуть 0. Если $b > a$, после вычисления интеграла поменять знак на противоположный. Для вычисления интеграла использовать метод парабол Симпсона. Количество разбиений отрезка и шаг на усмотрение разработчика.

$$\int_a^b f(x)dx = \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + f(x_n))$$