Алгоритмы.

Арифметические операции.
Реализация в языках программирования.

Арифметические операции

Арифметика - раздел математики, изучающий числа, их отношения и свойства. Предметом арифметики является понятие числа (натуральные, целые, рациональные, вещественные, комплексные числа) и его свойства.

В арифметике определенны следующие вычислительные операции (прямые и обратные им).

Прямая операция	Обратная операция
Сложение	Вычитание
Умножение	Деление
Возведение в степень	Извлечение корня

Натуральные числа

Натуральные числа — числа, возникающие естественным образом при счёте (например, 1, 2, 3, 4, …).

Внимание! Отрицательные и нецелые числа к натуральным не относят.

Место нуля в понятии натурального числа

Существуют два подхода к определению натуральных чисел:

1) числа, возникающие при подсчёте (нумерации) предметов: первый, второй, третий, четвёртый, пятый…;
2) числа, возникающие при обозначении количества предметов: 0 предметов, 1 предмет, 2 предмета, 3 предмета, 4 предмета, 5 предметов…

Обозначение

$\mathbb{N}$ - Натуральные числа включающие ноль

$\mathbb{N}^*$ - Натуральные числа без нуля

Целое число

Целые числа - расширение множества натуральных чисел, получаемое добавлением к нему нуля и отрицательных чисел.

Согласно своему построению, множество целых чисел состоит из трёх частей:

1) Натуральные числа (или, что то же самое, целые положительные).
2) Ноль — число, обозначаемое 0 . Его определяющее свойство: $0 + n = n + 0 = n$ для любого n.
3) Целые отрицательные числа. Отрицательные числа при записи помечаются спереди знаком минус: -1 , -2 , -3 …

Для каждого целого числа a существует и единственно противоположное ему число, обозначаемое $-a$ и обладающее тем свойством, что $a + (-a) = 0$. Если a положительно, то противоположное ему отрицательно, и наоборот. Ноль противоположен самому себе.

Обозначение	
$\mathbb{Z}$	Множество целых чисел

Рациональные числа

Рациональное число - число, которое можно представить обыкновенной дробью.

$\frac{m}{n}$ m — целое, n - натуральное

Обозначение

$\mathbb{Q}$ Множество рациональных чисел

Вещественные числа

Вещественное, или действительное, число — расширение множества рациональных чисел, путем добавления иррациональных чисел.

Иррациональное число — число которое не может быть представлено в виде обычной дроби.

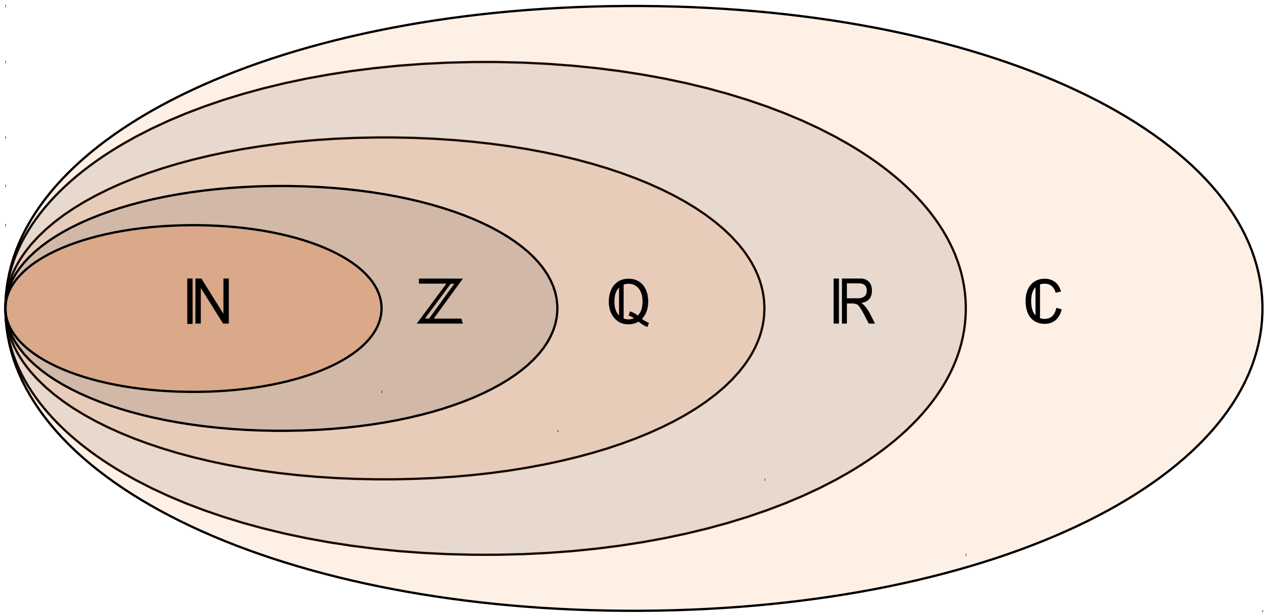
Обозначение	
$\mathbb{R}$	Множество вещественных чисел

Комплексные числа

Комплексные числа - числа вида a + bi, где a, b - вещественные числа, i - мнимая единица, то есть число, для которого выполняется равенство: $i^2 = -1$.

Обозначение	
$\mathbb{C}$	Множество комплексных чисел

Иерархия чисел



Представление чисел в языках Java и Python

Java	Python
Целые числа	**Целые числа**
byte, **short**, **char**, **int**, **long** — примитивные типы	**int** — ссылочный тип (длинная арифметика)
BigInteger — ссылочный тип (длинная арифметика)	
Рациональны числа	**Рациональны числа**
Не реализовано	**Fraction**
Вещественные числа	**Вещественные числа**
float, double — примитивные типы	**float** — ссылочный тип (ограниченная точность)
BigDecimal — ссылочный тип(длинная арифметика)	**Decimal** — ссылочный тип(длинная арифметика)
Комплексные числа	**Комплексные числа**
Не реализовано	**Complex** — ссылочный тип (ограниченная точность)

Приоритет арифметических операций

Порядок выполнения операций указывается скобками. Если скобок нет, то приоритет операций, в порядке убывания, следующий.

1) **Возведение в степень.**
2) **Умножение и деление.**
3) **Сложение и вычитание.**

Если в выражении используются операции с одинаковым приоритетом то вычисления производятся слева направо.

$$a^{b^c} = a^{\left(b^c\right)}$$

$$a/b \cdot c = \frac{a}{b} \cdot c$$

$$a + b \cdot c = a + (b \cdot c)$$

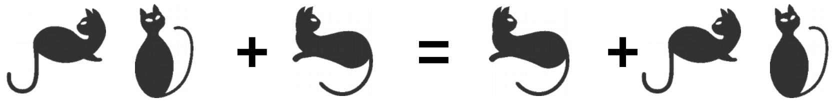
Сложение

Сложение — одна из основных бинарных математических операций (арифметических действий) двух аргументов (слагаемых), результатом которой является новое число (сумма), получаемое увеличением значения первого аргумента на значение второго аргумента.

$a + b = c$

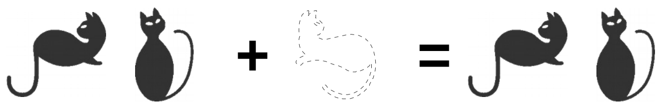
У сложения есть несколько важных свойств

Коммутативность: $a + b = b + a$



Ассоциативность: $(a + b) + c = a + (b + c)$

Наличие нуля: $a + 0 = a$

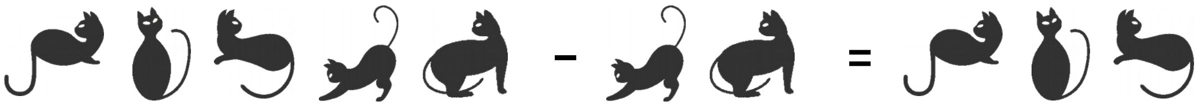


Вычитание

Вычитание - одна из вспомогательных бинарных математических операций двух аргументов (уменьшаемого и вычитаемого), результатом которой является новое число (разность), получаемое уменьшением значения первого аргумента на значение второго аргумента.

Разность — число результат сложения которого с вычитаемым дает уменьшаемое.

$a - b = c$
$c + b = a$



a b c

На письме обычно обозначается с помощью знака «минус»: $a - b = c$.

Вычитание — операция обратная сложению.

У вычитания есть несколько важных свойств

Антикоммутативность: $a - b = -(b - a)$

Неассоциативность: $(a - b) - c \neq a - (b - c)$

Вычитание 0 (нулевого элемента) даёт число равное исходному: $x - 0 = x$

Реализация на Java

Для примитивных типов используются операторы **+** и **-** . Для ссылочных типов используются соответствующие вызовы методов.

```java
package com.gmail.tsa;

public class Main {

	public static void main(String[] args) {

		int a = 2;
		int b = 3;
		int c = a + b;

		double a1 = 2.5;
		double b1 = 3.5;
		double c1 = b1 - a1;

	}

}
```

Примитивные типы

```java
package com.gmail.tsa;

import java.math.BigInteger;

public class Main {

	public static void main(String[] args) {

		BigInteger a = new BigInteger("2000");
		BigInteger b = new BigInteger("20");
		BigInteger c = a.add(b);

	}

}
```

Ссылочные типы

Особенности реализации

Для целочисленных примитивных типов существует переполнение. Так, как каждый примитивный тип способен описать число из ограниченного диапазона, то выход за его пределы приводит к переходу в другой конец диапазона.

```
package com.gmail.tsa;

public class Main {

    public static void main(String[] args) {

        int a = 2_000_000_000;
        int b = 2_000_000_000;
        int c = a + b;
        System.out.println(c);
    }
}
```

```
Вывод на экран: -294967296
```

Особенности реализации

Для вещественных примитивных типов точность представления числа ограниченна (формате IEEE 754 с плавающей точкой). Например для double точность до 15 знаков после запятой. Что приводит к элементам несоответствия операций для вещественных типов в Java и в арифметике.

```
package com.gmail.tsa;

public class Main {

    public static void main(String[] args) {

        double a = 0.3;
        double b = 0.1;
        double c = a - (b + b + b);
        System.out.println(c);
    }
}
```

```
Вывод на экран: -5.551115123125783E-17
```

Реализация на Python

Используются операторы **+** и **-** .

```
import decimal
a = 3
b = 2
c = a + b

a1 = 2.5
b1 = 3.5
c = a1 + b1

a2 = decimal.Decimal("0.3")
b2 = decimal.Decimal("0.1")
c2 = a2 - (b2 + b2 + b2)
```

Особенности реализации

Для float точность представления числа ограниченна (формате IEEE 754 с плавающей точкой). Что приводит к элементам несоответствия операций для вещественных типов в Python и в арифметике.

```
a = 0.3

b = 0.1

c = a - (b + b + b)

print(c)
```

```
Вывод на экран: -5.551115123125783e-17
```

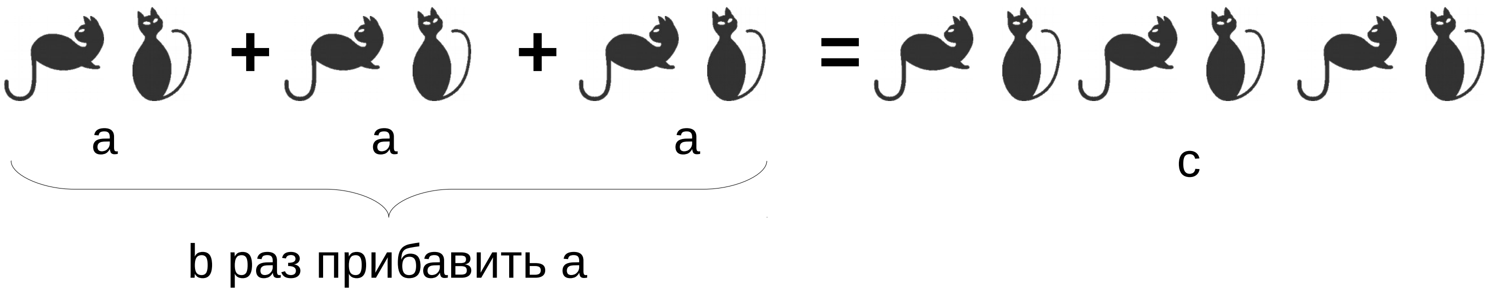
Умножение

Умножение — одна из основных математических операций над двумя аргументами (множителями, сомножителями). Первый аргумент называют множимым, а второй множителем; результат умножения двух аргументов называется их произведением.

Умножение имеет различный конкретный смысл и соответственно различные конкретные определения в зависимости от конкретного вида сомножителей и произведения.

Так, для натуральных чисел умножение определяется как многократное сложение — чтобы умножить число a на число b надо сложить b чисел a:

a * b = c



Умножение свойства

Коммутативность: $a \cdot b = b \cdot a$

Ассоциативность: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

Дистрибутивность: $x \cdot (a + b) = (x \cdot a) + (x \cdot b)$

Относительно умножения существует единственный нейтральный элемент 1. Умножение любого числа на 1(нейтральный элемент) даёт число, равное исходному:

Нейтральный элемент: $x \cdot 1 = 1 \cdot x = x$

Умножение на 1 идемпотентно, то есть повторное применение операции к объекту даёт тот же результат, что и одинарное:

Идемпотентность: $x = x \cdot 1 = (x \cdot 1) \cdot 1 = ((x \cdot 1) \cdot 1) \cdot . . . \cdot 1$

Умножение на 0 даёт 0:

Нулевой элемент: $x \cdot 0 = 0 \cdot x = 0$

Реализация на Java

Для примитивных типов используется оператор *. Для ссылочных типов используются соответствующие вызовы методов.

```java
package com.gmail.tsa;

public class Main {

	public static void main(String[] args) {

		int a = 2;
		int b = 3;
		int c = a * b;

		double a1 = 2.5;
		double b1 = 3.5;
		double c1 = b1 * a1;

	}

}
```

Примитивные типы

```java
package com.gmail.tsa;

import java.math.BigInteger;

public class Main {

	public static void main(String[] args) {

		BigInteger a = new BigInteger("2");
		BigInteger b = new BigInteger("3");
		BigInteger c = a.multiply(b);

	}

}
```

Ссылочные типы

Реализация на Python

Используется оператор * .

```
import decimal
a = 3
b = 2
c = a * b


a1 = 2.5
b1 = 3.5
c = a1 * b1


a2 = decimal.Decimal("0.3")
b2 = decimal.Decimal("0.1")
c2 = a2 * b2
```

Деление

Деление — действие, обратное умножению.

$a/b = c$

a — делимое

b — делитель

c — частное

Нахождение частного сводится к нахождению числа которое при умножении на делитель дает делимое.

Свойства деления

Не коммутативно: $a : b \neq b : a$

Не ассоциативно: $(a : b) : c \neq a : (b : c)$

Дистрибутивность: $(a + b) : x = (a : x) + (b : x)$

Нейтральный элемент справа: $x : 1 = x$

Существует единственный обратный элемент, получаемый делением единицы на число, что даёт число, обратное исходному.
Обратный элемент: $1 : x = x^{-1}$, $x \neq 0$

Нулевой элемент слева: $0 : x = 0$

Деление на ноль 0 (нулевой элемент) не определено. Деление на ноль: $x : 0 = \infty$

Деление на противоположный элемент даёт минус единицу: $x : (-x) = -1$

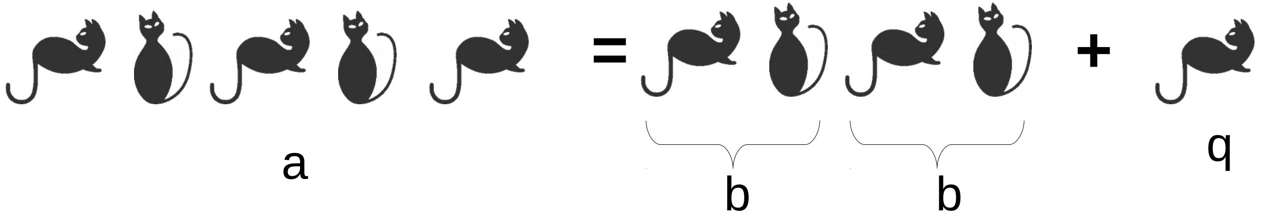
Деление целых чисел. Вычисление остатка.

Для целых чисел деление определяться следующим образом.

a = b * r + q

a — делимое (целое)
b — делитель (целое)
r — неполное частное (целое)
q — остаток от деления. 0 ≤ q < |b| (натуральное).
Важно! Остаток всегда должен быть положительным.

Например рассмотрим деление 5 на 2. запишем :
5 = 2*2 + 1



Примеры деления целых чисел и вычисления остатка

$5/3 \Rightarrow 5=3\cdot 1+2$ r = 1, q = 2

$5/(-3) \Rightarrow 5=(-3)\cdot(-1)+2$ r = -1, q = 2

$-5/3 \Rightarrow -5=3\cdot(-2)+1$ r = -2, q = 1

$-5/-3 \Rightarrow -5=-3\cdot 2+1$ r = 2, q = 1

Реализация на Java

Для примитивных типов используется оператор / (деление) и % (вычисление остатка). Для ссылочных типов используются соответствующие вызовы методов.

```
package com.gmail.tsa;

public class Main {

    public static void main(String[] args) {

        int a = 5;
        int b = 3;

        int r = a / b;
        int q = a % b;

    }

}
```

```
package com.gmail.tsa;

import java.math.BigInteger;

public class Main {

    public static void main(String[] args) {

        BigInteger a = new BigInteger("5");
        BigInteger b = new BigInteger("3");
        BigInteger r = a.divide(b);
        BigInteger q = a.remainder(b);

    }

}
```

Примитивные типы

Ссылочные типы

Особенности реализации

Для примитивных типов и BigInteger деление и вычисление остатка, не всегда согласованно с математическим определением.

Делимое (a)	Делитель (b)	Неполное частное (r)	Остаток от деления (q)	Согласованность с мат. определением
5	3	1	2	Да
5	-3	-1	2	Да
-5	3	-1	-2	**Нет**
-5	-3	1	-2	**Нет**

Реализация дополнительных методов (≥1.8)

Существуют методы для вычисления неполного частного `Math.floorDiv(a, b)` и остатка от деления `Math.floorMod(a, b)`.

Делимое (a)	Делитель (b)	Неполное частное (r)	Остаток от деления (q)	Согласованность с мат. определением
5	3	1	2	Да
5	-3	-2	-1	**Нет**
-5	3	-2	1	Да
-5	-3	1	-2	**Нет**

Реализация на Python

Для целочисленного деления и вычисления остатка используются операторы **//** и **%**

```
a = 5
b = 3
r = a//b
q = a % b
```

Особенности реализации

Деление и вычисление остатка, не всегда согласованно с математическим определением.

Делимое (a)	Делитель (b)	Неполное частное (r)	Остаток от деления (q)	Согласованность с мат. определением
5	3	1	2	Да
5	-3	-2	-1	**Нет**
-5	3	-2	1	Да
-5	-3	1	-2	**Нет**

Деление вещественных чисел. Реализация.

Java

Для примитивных типов используется оператор /

```
double a = 5.2;
double b = 2.1;
double c = a / b;
```

Для ссылочных вызов соответствующего метода

```
BigDecimal a = new BigDecimal("5.2").setScale(25);
BigDecimal b = new BigDecimal("2.1").setScale(25);
BigDecimal c = a.divide(b, RoundingMode.CEILING);
```

Python

Используется оператор /

```
import decimal

a = 5.2
b = 2.1
c = a/b


a1 = decimal.Decimal("5.2")
b1 = decimal.Decimal("2.1")
c1 = a1 / b1
```

Возведение в степень

Возведение в степень — арифметическая операция, первоначально определяемая как результат многократного умножения числа на себя.

Степень с основанием a и натуральным показателем b обозначается как a^b

$$a^b = \underbrace{a \cdot a \cdot a \ldots \cdot a}_{b}$$

Свойства возведения в степень

Не коммутативно $a^b \neq b^a$

Не ассоциативно $(a^b)^c \neq a^{(b^c)}$

Существование единицы $(a)^1 = a$

Возведение в степень 0 $(a)^0 = 1$

Дистрибутивность относительно умножения $(a \cdot b)^c = a^c \cdot b^c$

$$a^b \cdot a^c = a^{b+c}$$

$$(a^b)^c = a^{b \cdot c}$$

Возведение в степень. Реализация.

Java

Для примитивных типов используется метод Math.pow

```
int a = 3;
int b = 2;
double c = Math.pow(a, b);
```

Для ссылочных вызов соответствующего метода

```
BigInteger a = new BigInteger("3");
BigInteger b = new BigInteger("2");

BigInteger c = a.pow(b.intValue());
```

Python

Используется оператор **

```
a = 3
b = 2
c = a ** b
```

Извлечение корня

Корень n-й степени из числа a определяется как такое число b, что $b^n = a$. Здесь n — натуральное число, называемое показателем корня (или степенью корня).

$$b=\sqrt[n]{a}$$

Свойства корня:

$$\sqrt[n]{a^n}=\begin{cases} n-\text{четное}, a \\ n-\text{нечетное}, |a| \end{cases}$$

$$\sqrt[n]{a\cdot b}=\sqrt[n]{a}\cdot\sqrt[n]{b}$$

$$\sqrt[n]{a^m}=a^{\frac{m}{n}}$$

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

Извлечение корня. Реализация.

Java

Частный случай

$\sqrt{x}$ `Math.sqrt(x)`

$\sqrt[3]{x}$ `Math.cbrt(x)`

В остальных случая через возведение в степень

Python

Частный случай

$\sqrt{x}$ `math.sqrt(x)`

В остальных случая через возведение в степень

Список литературы

1) Р. Курант, Г. Роббинс: Что такое математика? — 3-е изд., испр. и доп. — М.: ЦНМО, 2001. — 568 с.
2) James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley: The Java® Language Specification Java SE 8 Edition , 2015-02-13 - https://docs.oracle.com/javase/specs/jls/se8/html/index.html
3) The Python Standard Library documentation - https://docs.python.org/3/library/index.html