Лабораторна робота №1

Структура програми Java. Класи, змінні, методи, пакети

Короткі теоретичні відомості

Огляд структури Java-програми

Усі Java-програми містять у собі 4 основні різновиди будівельних блоків: типи (classes/interfaces/enums/records), методи (methods), змінні (variables) і пакети (packages). Якою б мовою Ви не програмували раніше, Ви скоріш за все вже добре знайомі з методами, які є не що інше, ніж функції чи підпрограми, та зі змінними, в яких зберігаються дані. З іншого боку, класи, інтерфейси, енуми та рекорди являються фундаментом об'єктно-орієнтованих властивостей мови. Поки що, для простоти, можна вважати клас деяким цілим, що містить у собі змінні та методи. Нарешті, пакети містять у собі класи й допомагають компілятору знайти ті класи, що потрібні йому для компіляції прикладної програми. Java-програма може містити в собі будь-яку кількість класів, але один з них завжди має особливий статус, і безпосередньо взаємодіє з віртуальною машиною Java (JVM). Цей клас називають первинним класом (primary class). Коли програма запускається з командного рядка, системі потрібен тільки один спеціальний метод, що повинен бути присутнім у первинному класі, — метод main. Розглянемо приклад програми мовою Java:

```
// імпортування класу LocalDate зі стандартного пакета java.time
import java.time.LocalDate;

public class Main {
    public static void main(String[] S) {
        System.out.println("Hello, Java!");
        LocalDate d = LocalDate.now();
        System.out.println("Date: " + d);
    }
}
```

Наведена програма виводить на екран повідомлення "Hello, Java!" та поточну системну дату.

Нововведення Java 25 для початківців

Компактні файли Java 25 — це новий формат вихідних файлів у JDK 25, який дозволяє писати Java-код без оголошення класу, без `public`, без `static`, з автоматичними імпортами та мінімумом «церемоній». Це частина JEP 512, спрямована на спрощення навчання та написання малих програм.

Компактний файл — це `Java`-файл, у якому методи та поля можна писати на верхньому рівні, без оголошення класу. Компілятор автоматично створює прихований фінальний клас,

який містить ці методи та поля.

```
// Hello.java
void main() {
    IO.println("Hello, World!");
}
```

Жодних класів, жодних static, жодних імпортів.

Основні можливості компактних файлів

1. Не потрібен клас

Файл може містити методи та поля на верхньому рівні. Java сама створює прихований клас.

2. Автоматичний імпорт

Усі публічні типи з модуля java.base доступні без import (наприклад, List, Map, BigInteger).

3. Instance main

Метод може бути:

- нестатичним,
- непублічним,
- без параметрів або з параметрами.

При запуску JVM сама створить екземпляр прихованого класу та викличе main().

4. Новий клас java.lang.IO

Спрощений ввід/вивід:

```
IO.println("Hello");
var s = IO.readln();
```

Це альтернатива `System.out.println` для початківців.

Обмеження компактних файлів

- Файл повинен містити main, що використовується для запуску (інакше це не компактний файл).
- Компактні файли завжди в unnamed package/module.
- Не підходять для великих проєктів — це інструмент для «programming-in-the-small».

Стандартні типи даних Java

Усі змінні та вирази в мові програмування Java можуть бути віднесені до однієї з двох

великих груп типів: примітивних типів (primitive types) або посилальних типів (reference types), що містять у собі типи, які визначені користувачем, і масиви. До примітивних типів відносяться стандартні, вбудовані в мову типи для представлення чисельних значень, поодиноких символів та логічних значень. Навпаки, усі посилальні типи є динамічними типами. Головні розбіжності між двома згаданими групами типів перелічені в наступній таблиці:

Table 1. Примітивні та посилальні типи

Характеристика	**Примітивні типи**	**Посилальні типи**
Чи визначені в самій мові Java?	Так	Ні
Чи мають визначений розмір?	Так	Ні
Чи повинна для змінних цих типів виділятися пам’ять під час роботи програми?	Ні	Так

На практиці найважливішим розходженням між примітивними та посилальними типами є те, про що свідчить останній рядок цієї таблиці, а саме — що пам’ять для змінних посилального типу повинна виділятися під час виконання програми. Використовуючи змінні посилальних типів, ми повинні явно вимагати необхідну кількість пам’яті для кожної змінної перш ніж ми зможемо зберегти в цієї змінний деяке значення. Причина цього проста: JVM сама по собі не знає, яка кількість пам’яті потрібна для того чи іншого посилального типу. Усього в мові Java визначено вісім примітивних типів, що перелічені в таблиці

Table 2. Примітивні типи мови Java

Тип	**Розмір**	**Діапазон**	**Приклад**
byte	1 байт	від -128 до 127	125
short	2 байти	від -32768 до 32767	-42
int	4 байти	від -2^{31} до $2^{31}-1$	198
long	8 байтів	від -2^{63} до $2^{63}-1$	12345678991
float	4 байти	Залежить від розрядності числа	1.2f
double	8 байтів	Залежить від розрядності числа	123.4
boolean		false, true	true
char	2 байти	Усі символи стандарту Unicode	'y'

Стандартні математичні функції

Оскільки мова Java є об’єктно-орієнтованою, то математичні функції повинні належати до

деякого класу. Фактично існують два класи, що визначають математичні операції: Math та StrictMath. Останній клас, призначений для виконання обчислень із ”підвищеною точністю”, але через поширення вбудованих у процесори математичних модулів, ”звичайна” і ”підвищена” точність у сучасній Java не розрізняються. Тому найчастіше використовується саме клас Math.

Усі стандартні математичні функції в мові Java є статичними методами класу Math, який визначений з модифікатором `final`, тобто не припускає спадкування. Крім того, клас Math має декілька визначених констант, наведемо дві з них:

Константа	**Значення**
Math.PI	3.1415926…
Math.E	2.7182818…

Деякі статичні методи класу Math наведені в наступній таблиці:

Функція – метод	**Пояснення**
Math.abs(x)	Модуль числа x
Math.acos(x)	Арккосинус x
Math.asin(x)	Арксинус x
Math.atan(x)	Арктангенс x
Math.cbrt(x)	Кубічний корінь з x
Math.ceil(x)	Найближче число до x, що не містить дробової частини та більше за x
Math.cos(x)	Косинус x
Math.exp(x)	Експонента від x - ейлерове число e піднесене до степеня x
Math.floor(x)	Найближче число до x, що не містить дробової частини та менше за x
Math.hypot(x,y)	Гіпотенуза прямокутного трикутника зі сторонами x, y
Math.log(x)	Натуральний логарифм x
Math.max(x,y)	Більше з двох чисел
Math.min(x,y)	Менше з двох чисел
Math.pow(x,y)	x в степені y
Math.random()	Випадкове число з проміжку [0;1)
Math.rint(x)	Найближче число до x, що не містить дробової частини
Math.round(x)	Найближче до x ціле число
Math.sin(x)	Синус x

Функція – метод	Пояснення
Math.sqrt(x)	Квадратний корінь з x
Math.tan(x)	Тангенс x
Math.toDegrees(x)	Перетворення кута з радіанів у градуси
Math.toRadians(x)	Перетворення кута з градусів у радіани

Примітка. У мові Java є можливість імпорту статичних змінних та методів класу за допомогою директиви `import static` на початку програми. Наприклад:

```
import static java.lang.Math.*;
// імпортування статичних змінних і методів класу Math

public class OurPrimaryClass {
    public static void main(String[] S) {
        double x;
        x = sin(PI/6);
       // без статичного імпорту треба писати x=Math.sin(Math.PI/6);
        System.out.println(x);
    }
}
```

Виведення даних у консолі Java-програм

Для виведення інформації на консоль використовуються методи стандартного класу PrintStream:

- print
- println
- printf
- format (точна копія printf)

Кожна програма мовою Java містить стандартний об'єкт типу `PrintStream` – `System.out`. Таким чином, виведення інформації на екран буде записуватися як `System.out.print(⋯)`, `System.out.println(⋯)`, або `System.out.printf(⋯)`. Методи `print` та `println` повинні завжди мати один параметр – вираз будь-якого типу, що може бути автоматично приведений до рядкового типу.

Наприклад,

```
System.out.println("2+2="+(2+2)); // буде виведено 2+2=4
System.out.println("Значення суми="+s);
// буде виведено Значення суми=xxx , де xxx  значення змінної S
```

Методи `printf` та `format` можуть мати список параметрів, що розділяються комами. Перший параметр – рядок, що містить текст для виведення і форматні шаблони для виведення

значень інших параметрів. Наприклад, якщо a=2, b=3

```
System.out.printf("Значення %d + %d = %d", a, b, a+b);
// буде виведено Значення 2 + 3 = 5
```

Форматні шаблони для виведення звичайних, символьних та числових типів мають наступний синтаксис:

`%[індекс_аргумента$][опції][ширина][.точність]перетворення`

Необов'язковий параметр `індекс_аргумента` є цілим числом, що вказує позицію в списку аргументів. Посилання на перший аргумент буде записане як "1$", на другий – "2$", і т.д.

Необов'язковий параметр опції – це набір символів, що змінюють формат виведення. Набір припустимих опцій залежить від типу перетворення.

Необов'язковий параметр ширина – це невід'ємне ціле число, що показує мінімальну кількість символів, що їх треба вивести.

Необов'язковий параметр точність – це невід'ємне ціле число, що зазвичай використовується для обмеження кількості символів, що будуть виведені. Його дія залежить від параметра перетворення.

Обов'язковий параметр перетворення – це один символ, що вказує як аргумент буде відформатований. Набір припустимих перетворень для вказаного аргументу залежить від типу даних аргументу.

Код

```
System.out.printf("Hello, World!");
```

виведе

```
Hello, World!
```

Для того, щоб після виведення, перейти на новий рядок треба до виведення додати %n або \n

```
System.out.printf("Hello, World!%n");
або
System.out.printf("Hello, World!\n");
```

Для форматування цілих чисел можна використати шаблон %d

```
System.out.printf("Sum %d + %d = %d", a, b, a+b);
```

буде виведено

```
Sum 15 + 2 = 17
```

Для форматування дробових (дійсних) чисел можна використати шаблон %f

```
System.out.printf("Const of Pi =%5.2f", Math.PI);
```

буде виведено

```
Const of Pi = 3,14
```

Основні типи - символи перетворень

Перетворення	**Категорія**	**Опис**
'b', 'B'	boolean	Якщо аргумент arg є null, тоді результатом буде "false". Якщо arg належить до типу boolean або Boolean, то результатом буде рядок – ”true” або ”false” залежно від значення arg. У всіх інших випадках результатом буде "true".
's', 'S'	general	Якщо аргумент arg є null, тоді результатом буде "null". Якщо arg має метод formatTo, то він буде викликаний. Інакше, результат буде отриманий через виклик arg.toString().
'c', 'C'	character	Результатом буде символ Unicode
'd'	integral	Результат буде відформатований, як ціле десяткове число
'e', 'E'	floating point	Результат буде відформатований, як число з плаваючою точкою у ”науковому” форматі
'f'	floating point	Результат буде відформатований, як десяткове число

Перетворення	Категорія	Опис
'g', 'G'	floating point	Результат буде відформатований, як число з плаваючою точкою у ”науковому” форматі
'%'	percent	Результатом буде символ '%' ('\u0025')
'n'	line separator	Результатом буде символ, що відокремлює рядки залежно від платформи.

Введення даних з консолі

Для введення даних у мові програмування Java можна скористатися різними засобами. Один з них використовує спеціальний об’єкт, що належить до класу Scanner. Цей клас містить методи для введення найрізноманітніших типів даних. Приклад його використання наведений нижче:

```
import java.io.*;
import java.util.*;

public class InOutExample {
    public static void main(String[] s) {
        Scanner s = new Scanner(System.in);
        // Читання цілого числа з рядка
        int i = s.nextInt();
        // Читання дійсного числа з рядка
        double x = s.nextDouble();
      //.................................
    }
}
```

Операції з датами / часом. Форматування

Клас *Clock*

Clock забезпечує доступ до поточної дати та часу. Об’єкт класу Clock знає про часову зону і може бути використаний замість `System.currentTimeMillis()`, щоб отримати поточний час у мілісекундах. Така миттєва точка на часовій лінії також представлена класом Instant. Об’єкт класу Instant може бути використаний для створення традиційних об’єктів класу java.util.Date.

```
Clock clock = Clock.systemDefaultZone();
long millis = clock.millis();
Instant instant = clock.instant();
```

```
Date legacyDate = Date.from(instant); // legacy java.util.Date
```

Часові пояси (*Timezones*)

Часові пояси представлені в ZoneId. Вони можуть бути легко отримані за допомогою статичних фабричних методів. Часові пояси визначають зміщення між моментами, які є важливими для перетворень між миттєвостями (Instant) і локальними значеннями дати та часу.

```
System.out.println(ZoneId.getAvailableZoneIds());
// prints all available timezone ids
ZoneId zone1 = ZoneId.of("Europe/Berlin");
ZoneId zone2 = ZoneId.of("Brazil/East");
System.out.println(zone1.getRules());
System.out.println(zone2.getRules());
// ZoneRules[currentStandardOffset=+01:00]
// ZoneRules[currentStandardOffset=-03:00]
```

LocalTime

LocalTime представляє час без урахування часового поясу, наприклад, 10pm або 17:30:15. У наступному прикладі створюються два об'єкти LocalTime для часових поясів, зазначених вище. Потім ми порівнюємо ці об'єкти та обчислюємо різницю в годинах і хвилинах між цими моментами часу.

```
LocalTime now1 = LocalTime.now(zone1);
LocalTime now2 = LocalTime.now(zone2);
System.out.println(now1.isBefore(now2)); // false
long hoursBetween = ChronoUnit.HOURS.between(now1, now2);
long minutesBetween = ChronoUnit.MINUTES.between(now1, now2);
System.out.println(hoursBetween); // -3
System.out.println(minutesBetween); // -239
```

LocalTime містить різноманітні фабричні методи для спрощення створення нових екземплярів, включно з конвертацією з рядка.

```
LocalTime late = LocalTime.of(23, 59, 59);
System.out.println(late); // 23:59:59
DateTimeFormatter germanFormatter =
DateTimeFormatter.ofLocalizedTime(FormatStyle.SHORT).withLocale(Locale.GERMAN);
LocalTime leetTime = LocalTime.parse("13:37", germanFormatter);
System.out.println(leetTime); // 13:37
```

LocalDate

Об'єкт LocalDate представляє конкретну дату, наприклад, 2024-02-06. Він незмінний і працює аналогічно LocalTime. Приклад демонструє обчислення нових дат за допомогою додавання

або віднімання днів, місяців або років. Необхідно мати на увазі, що кожна така маніпуляція повертає новий об'єкт.

```
import java.time.Month;

LocalDate today = LocalDate.now();
LocalDate tomorrow = today.plus(1, ChronoUnit.DAYS);
LocalDate yesterday = tomorrow.minusDays(2);
LocalDate independenceDay = LocalDate.of(2023, Month.AUGUST, 24);
DayOfWeek dayOfWeek = independenceDay.getDayOfWeek();
System.out.println(dayOfWeek); // THURSDAY
```

Отримати об'єкт LocalDate із рядка так само просто, як і LocalTime

```
DateTimeFormatter myFormatter = DateTimeFormatter
        .ofLocalizedDate(FormatStyle.LONG)
        .withLocale(Locale.getDefault());

LocalDate programmersDay = LocalDate
        .parse("September 13, 2026", myFormatter);
System.out.println(programmersDay);
```

Використання форматування дати-часу: Клас DateTimeFormatter використовується для виведення, розбору об'єктів дати і часу. Отримати DateTimeFormatter можна трьома способами:

- З використанням зумовлених констант, типу ISO_LOCAL_DATE
- З використанням шаблонних букв типу uuuu-MMM-dd
- З використанням локалізованих стилів, таких, як long або medium

Основні класи дати-часу надають два методи - один для форматування, тобто перетворення дати-часу в рядок:

```
format(DateTimeFormatter formatter)
```

і ще один - для розбору, тобто отримання дати-часу з рядка,

```
parse(CharSequence text, DateTimeFormatter formatter);
```

```
LocalDate date = LocalDate.now();
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy MMMM dd");
String text = date.format(formatter);
LocalDate parsedDate = LocalDate.parse(text, formatter);
```

Шаблони для форматування та розбору

Шаблони засновані на простих послідовностях букв і символів. Шаблон використовується для створення об’єкта форматування з використанням методів ofPattern(String) і ofPattern(String, Locale). Наприклад, "d MMM uuuu" форматуватиме 2026-09-13 як '13 сен 2026'. Об’єкт форматування, створений із шаблону, може бути використаний стільки разів, скільки потрібно, оскільки він незмінний та потокобезпечний.

Символи форматування

Symbol	Meaning	Presentation	Examples
G	era	text	AD; Anno Domini; A
u	year	year	2004; 04
y	year-of-era	year	2004; 04
D	day-of-year	number	189
M/L	month-of-year	number/text	7; 07; Jul; July; J
d	day-of-month	number	10
Q/q	quarter-of-year	number/text	3; 03; Q3; 3rd quarter
Y	week-based-year	year	1996; 96
w	week-of-week-based-year	number	27
W	week-of-month	number	4
E	day-of-week	text	Tue; Tuesday; T
e/c	localized day-of-week	number/text	2; 02; Tue; Tuesday; T
F	week-of-month	number	3
a	am-pm-of-day	text	PM
h	clock-hour-of-am-pm (1-12)	number	12
K	hour-of-am-pm (0-11)	number	0
k	clock-hour-of-am-pm (1-24)	number	0
H	hour-of-day (0-23)	number	0
m	minute-of-hour	number	30
s	second-of-minute	number	55
S	fraction-of-second	fraction	978
A	milli-of-day	number	1234
n	nano-of-second	number	987654321
N	nano-of-day	number	1234000000

Symbol	Meaning	Presentation	Examples
V	time-zone ID	zone-id	America/Los_Angeles; Z; -08:30
z	time-zone name	zone-name	Pacific Standard Time; PST
O	localized zone-offset	offset-O	GMT+8; GMT+08:00; UTC-08:00;
X	zone-offset 'Z' for zero	offset-X	Z; -08; -0830; -08:30; -083015; -08:30:15;
x	zone-offset	offset-x	+0000; -08; -0830; -08:30; -083015; -08:30:15;
Z	zone-offset	offset-Z	+0000; -0800; -08:00;

Завдання

Створити порожній Java проєкт, або завантажити його з https://github.com/kafedra-iust/lab1oop.git

Завдання 1.1

A. Створити у цьому проєкті компактний файл (Java Compact File)

B. Додати у нього метод main та методи для обчислення за формулами, вказаними в умові завдання відповідно до варіанту

C. В методі main реалізувати введення початкових даних, виклик функцій, створених за пунктом B та вивести результати

D. Доповнити метод main виведенням поточних дати/часу згідно з варіантом

E. Виконати програму та зафіксувати результати

Задання 1.2

A. Створити у проєкті один клас Main, що розташований у пакеті main.

B. У цьому класі визначити метод main (з відповідними модифікаторами), що необхідний для його запуску, як автономної програми.

C. Доповнити клас методом run(), що викликається з методу main та виконує введення вхідних даних.

D. Створити клас Calculator, що має методи для обчислень за формулами, вказаними в умові завдання.

E. У методі run() класу Main, після введення вхідних даних, створити об’єкт класу Calculator, та викликати його методи для обчислень за формулами.

F. Створити клас Printer, який має метод виведення на екран значення, супроводжуючи

його відповідним описом.

G. У методі run() класу Main, після обчислень за формулами, додайте створення об’єкта класу Printer.

H. Використовуючи створений об’єкт, виведіть значення вхідних даних та результати обчислень.

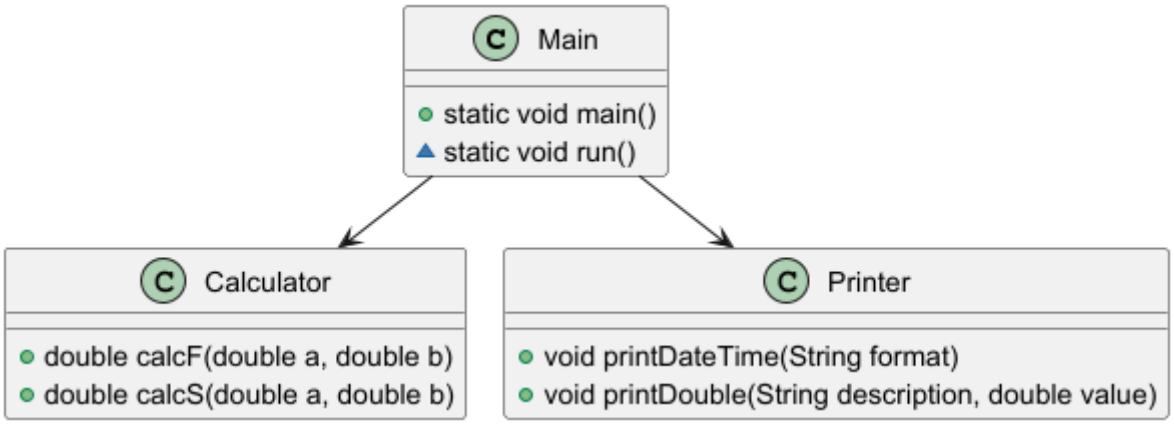
I. Додайте в клас Printer метод, що друкує поточні час і/або дату у вказаному форматі (формат передавати, як параметр).

J. У методі run() класу Main додайте виклик метода виведення дати/часу.

K. Виконайте створену програму і збережіть її результати.

L. Порівняйте отримані результати з результатами **Завдання 1.1**

Структура класів програми, яка рекомендується, наведена на рисунку



Варіанти

Варіант	Розрахункові формули	Значення вхідних даних	Формат дати і часу
1	$R = x^2(x+1)/b - sin^2(x+a); s = \sqrt{\frac{x \cdot b}{a}} + cos^2(x+b)^3$	$a = 0.7$ $b = 0.05$ $c = 0.5$	Дата у форматі рр-мм-дд
2	$f = \sqrt[3]{m \cdot tgt + \lvert c \cdot sint \rvert}; z = m \cdot cos(b \cdot t \cdot sint) + c$	$m = 2$ $c = -1$ $t = 1.2$ $b = 0.7$	Дата та час з точністю до мілісекунд
3	$y = b \cdot tg^2 x - \frac{a}{sin^2(x/a)}; d = a \cdot e^{-\sqrt{a}} cos(b \cdot x/a)$	$a = 3.2$ $b = 17.5$ $x = -4.8$	Місяць, день, рік, та день тижня
4	$s = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}; f = x(sinx^3 + cos^3 y)$	$x = 0.335$ $y = 0.025$	Час у форматі гг:хх:сс
5	$s = x^3 tg^2(x+b)^2 + \frac{a}{\sqrt{x+b}}; Q = \frac{b \cdot x^2 - a}{e^{a \cdot x} - 1}$	$a = 16.5$ $b = 3.4$ $x = 0.61$	Дата у форматі дд міс рррр
6	$y = e^{-bt} sin(a \cdot t + b) - \sqrt{\lvert b \cdot t + a \rvert}; s = b \cdot sin(a \cdot t^2 \cdot cos(2t)) - 1$	$a = -0.5$ $b = 1.7$ $t = 0.44$	Дата у форматі дд місяць рррр
7	$y = sin^3(x^2 + a)^2 - \sqrt{\frac{x}{b}}; z = \frac{x^2}{a} + cos(x+b)^3$	$a = 1.1$ $b = 0.004$ $x = 0.2$	День тижня, число і місяць
8	$a = \frac{2 \cdot cos(x - \pi/6)}{1/2 + sin^2 y}; b = 1 + \frac{z^2}{3 + z^2/5}$	$x = 1.426$ $y = -1.220$ $z = 3.5$	Дата у форматі дд-мм-рр
9	$w = \sqrt{x^2 + b} - b^2 sin^3(x+a)/x; y = cos^2 x^3 - \frac{x}{\sqrt{a^2 + b^2}}$		День тижня та час

		$a = 1.5$ $b = -15.5$ $x = -2.8$	
10	$c = \lvert x^{y/x} - \sqrt[3]{y/x}\rvert; f = (y-x)\dfrac{y - z/(y-z)}{1+(y-x)^2}$	$x = 1.825$ $y = 18.225$ $z = -3.298$	Час у форматі гг:хх та дата дд-мм-рр
11	$a = \dfrac{2cos(x+\pi/6)}{3/2 - sin^2 y};\ b = 1/e^z + \dfrac{z^3}{2+z^2/7}$	$x = 1.246$ $y = -1.23$ $z = -3.5$	Дата у форматі дд-мм-рр
12	$R = x^3(x+1)/b - sin^2(x+a);\ s = \sqrt{\dfrac{\lvert x \cdot b\rvert}{a^{2/5}}} + cos(x+b)^2$	$a = 0.7$ $b = 0.05$ $c = 0.5$	Дата у форматі рр-мм-дд
13	$c = \lvert e^{y/x} - \sqrt[5]{y/x}\rvert; f = (x-y)\dfrac{x - z/(y-x)}{1+(y-x)^3}$	$x = 1.85$ $y = 18.25$ $z = -3.28$	Місяць, день, рік та день тижня
14	$f = \sqrt[3]{m \cdot tgt + \lvert c \cdot sint\rvert};\ z = m \cdot cos(\dfrac{b}{t} sint) - c$	$m = 2$ $c = -1$ $t = 1.2$ $b = 0.7$	Дата і час з точністю до мілісекунд
15	$y = b \cdot tg^2 x + a \cdot e^{-\sqrt{a}};\ d = cos(b \cdot x/a) - \dfrac{a}{sin^2(x/a)}$	$a = 3.2$ $b = 17.5$ $x = -4.8$	Час у форматі гг:хх:сс