

# Отчет по лабораторной работе №4 предмета ООП

Выполнил Брусенцев Степан Эдуардович  
Студент группы 6204-010302D

## Задание на лабораторную работу:

Расширить возможности пакета для работы с функциями одной переменной добавив интерфейсы и классы для аналитически заданных функций, а также методы ввода и вывода табулированных функций.

### Задание 1

В классах `ArrayTabulatedFunction` и `LinkedListTabulatedFunction` добавил конструкторы, получающие сразу все точки функции в виде массива объектов типа `FunctionPoint`.

```
public ArrayTabulatedFunction(FunctionPoint[] points) 2 usages
{
    if (points == null) {
        throw new IllegalArgumentException("Points array must not be null");
    }
    if (points.length < 2) {
        throw new IllegalArgumentException("The number of points must be at least 2");
    }

    // Проверка упорядоченности по абсциссе
    for (int i = 0; i < points.length - 1; i++) {
        if (points[i] == null) {
            throw new IllegalArgumentException("Points array must not contain null elements");
        }
        if (points[i].getX() >= points[i + 1].getX() - EPSILON) {
            throw new IllegalArgumentException("Points must be ordered by x-coordinate in ascending order");
        }
    }
    if (points[points.length - 1] == null) {
        throw new IllegalArgumentException("Points array must not contain null elements");
    }

    // Создание копий для обеспечения инкапсуляции
    this.funct = new FunctionPoint[points.length];
    this.len = points.length;
    for (int i = 0; i < len; i++) {
        funct[i] = new FunctionPoint(points[i]);
    }
}
```

```

public LinkedListTabulatedFunction(FunctionPoint[] points) { no usages
    // Инициализация полей (то, что делает дефолтный конструктор)
    head = new FunctionNode( value: null);
    head.next = head;
    head.prev = head;
    size = 0;
    cachedNode = null;
    cachedIndex = -1;

    if (points == null) {
        throw new IllegalArgumentException("Points array must not be null");
    }
    if (points.length < 2) {
        throw new IllegalArgumentException("The number of points must be at least 2");
    }

    // Проверка упорядоченности по абсциссе
    for (int i = 0; i < points.length - 1; i++) {
        if (points[i] == null) {
            throw new IllegalArgumentException("Points array must not contain null elements");
        }
        if (points[i].getX() >= points[i + 1].getX() - EPSILON) {
            throw new IllegalArgumentException("Points must be ordered by x-coordinate in ascending order");
        }
    }
    if (points[points.length - 1] == null) {
        throw new IllegalArgumentException("Points array must not contain null elements");
    }

    // Добавление точек с созданием копий для обеспечения инкапсуляции
    for (int i = 0; i < points.length; i++) {
        addNodeToTailWithValue(points[i]);
    }
}

```

Рис. 1,2 Конструкторы

## Задание 2

В пакете functions создал интерфейс Function, описывающий функции одной переменной и содержащий следующие методы:

- `public double getLeftDomainBorder()` – возвращает значение левой границы области определения функции;
- `public double getRightDomainBorder()` – возвращает значение правой границы области определения функции;
- `public double getFunctionValue(double x)` – возвращает значение функции в заданной точке.

```

package functions;

public interface Function {
    /**
     * Возвращает значение левой границы области определения функции.
     * @return левая граница области определения
     */
    double getLeftDomainBorder(); 11 implementations

    /**
     * Возвращает значение правой границы области определения функции.
     * @return правая граница области определения
     */
    double getRightDomainBorder(); 11 implementations

    /**
     * Возвращает значение функции в заданной точке.
     * @param x точка, в которой вычисляется значение функции
     * @return значение функции в точке x
     */
    double getFunctionValue(double x); 13 implementations
}

```

Рис. 3 Интерфейс

### Задание 3

Создал пакет `functions.basic`, в нём описаны классы ряда функций, заданных аналитически.

Создал класс `TrigonometricFunction`, реализующий интерфейс `Function` и описывающий методы получения границ области определения для работы с тригонометрическими функциями.

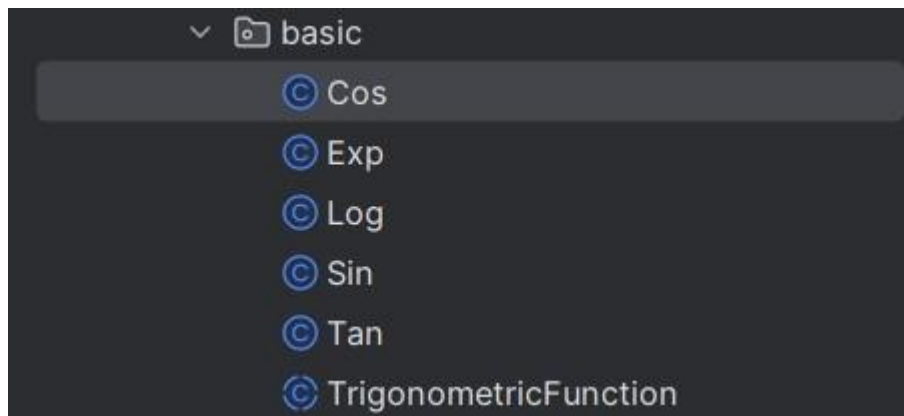


Рис. 4 папка с классами

```
/**
 * Класс для вычисления логарифма по заданному основанию.
 * Реализует интерфейс Function.
 */
public class Log implements Function { 4 usages
    private static final double EPSILON = 1e-10; 1 usage
    private final double base; 2 usages

    /**
     * Конструктор класса Log.
     * @param base основание логарифма (должно быть положительным и не равным 1)
     */
    public Log(double base) { 2 usages
        if (base <= 0 || Math.abs(base - 1.0) < EPSILON) {
            throw new IllegalArgumentException("Base must be positive and not equal to 1");
        }
        this.base = base;
    }

    @Override
    public double getLeftDomainBorder() { return 0; }

    @Override
    public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }

    @Override
    public double getFunctionValue(double x) {
        if (x <= 0) {
            return Double.NaN;
        }
        // log_base(x) = ln(x) / ln(base)
        return Math.Log(x) / Math.log(base);
    }
}
```

```

package functions.basic;

import functions.Function;

/**
 * Класс для вычисления экспоненты  $e^x$ .
 * Реализует интерфейс Function.
 */
public class Exp implements Function { 4 usages
    @Override
    public double getLeftDomainBorder() { return Double.NEGATIVE_INFINITY; }

    @Override
    public double getRightDomainBorder() { return Double.POSITIVE_INFINITY; }

    @Override
    public double getFunctionValue(double x) { return Math.exp(x); }
}

```

```

package functions.basic;

/**
 * Класс для вычисления косинуса.
 * Наследуется от TrigonometricFunction.
 */
public class Cos extends TrigonometricFunction { 4 usages
    @Override
    public double getFunctionValue(double x) { return Math.cos(x); }
}

```

Рис. 5,6,7 классы

## Задание 4

Создал пакет functions.meta, в нём будут описаны классы функций, позволяющие комбинировать функции.



## Задание 5

В пакете `functions` создал класс `Functions`, содержащий вспомогательные статические методы для работы с функциями.

`public static Function shift(Function f, double shiftX, double shiftY)` – возвращает объект функции, полученной из исходной сдвигом вдоль осей;

`public static Function scale(Function f, double scaleX, double scaleY)` – возвращает объект функции, полученной из исходной масштабированием вдоль осей;

`public static Function power(Function f, double power)` – возвращает объект функции, являющейся заданной степенью исходной;

`public static Function sum(Function f1, Function f2)` – возвращает объект функции, являющейся суммой двух исходных;

`public static Function mult(Function f1, Function f2)` – возвращает объект функции, являющейся произведением двух исходных;

`public static Function composition(Function f1, Function f2)` – возвращает объект функции, являющейся композицией двух исходных.

## Задание 6

В пакете `functions` создал класс `TabulatedFunctions`, содержащий вспомогательные статические методы для работы с табулированными функциями.

Описал в классе метод `public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount)`, получающий функцию и возвращающий её табулированный аналог на заданном отрезке с заданным количеством точек и добавил исключение.

## Задание 7

В класс `TabulatedFunctions` добавил следующие методы.

Метод вывода табулированной функции в байтовый поток `public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out)`

Метод ввода табулированной функции из байтового потока `public static TabulatedFunction inputTabulatedFunction(InputStream in)`

Метод записи табулированной функции в символьный поток `public static void writeTabulatedFunction(TabulatedFunction function, Writer out)`

Метод чтения табулированной функции из символьного потока `public static TabulatedFunction readTabulatedFunction(Reader in)`

## Задание 8

Проверил работу написанных классов.

Создайте по одному объекту классов `Sin` и `Cos`, выведите в консоль значения этих функций на отрезке от 0 до  $\pi$  с шагом 0,1.

## Задание 9

Сделал так, чтобы объекты всех классов, реализующих интерфейс `TabulatedFunction`, были сериализуемыми.

Для этого рассмотрел два случая:

с использованием интерфейса `java.io.Serializable`

с использованием интерфейса `java.io.Externalizable`

Проверил работу написанных классов. С помощью метода `TabulatedFunctions.tabulate()` и метода класса `Functions` создал

табулированный аналог логарифма по натуральному основанию, взятого от экспоненты на отрезке от 0 до 10 с 11 точками. Сериализовал полученный объект в файл (имя файла должно отличаться от предыдущих случаев).

Далее десериализовал табулированную функцию из этого файла. Вывел значения исходной и считанной функции на отрезке от 0 до 10 с шагом 1.

Изучил содержимое файлов, получаемых при реализации механизма сериализации с использованием интерфейса `java.io.Serializable` и при реализации механизма сериализации с использованием интерфейса `java.io.Externalizable`. Сделал выводы о преимуществах и недостатках каждого из способов.

Все задания лабораторной №4 выполнены.



# ТЕСТЫ

## === ТЕСТ СЕРИАЛИЗАЦИИ ===

Создана табулированная функция  $\ln(\exp(x))$  на отрезке  $[0, 10]$  с 11 точками

## === ТЕСТ 1: SERIALIZABLE ===

Сравнение исходной и десериализованной функции на отрезке  $[0, 10]$  с шагом 1:

x	Исходная	Десериализованная	Разница
0,0	0,0000000000	0,0000000000	0,0000000000
1,0	1,0000000000	1,0000000000	0,0000000000
2,0	2,0000000000	2,0000000000	0,0000000000
3,0	3,0000000000	3,0000000000	0,0000000000
4,0	4,0000000000	4,0000000000	0,0000000000
5,0	5,0000000000	5,0000000000	0,0000000000
6,0	6,0000000000	6,0000000000	0,0000000000
7,0	7,0000000000	7,0000000000	0,0000000000
8,0	8,0000000000	8,0000000000	0,0000000000
9,0	9,0000000000	9,0000000000	0,0000000000
10,0	10,0000000000	10,0000000000	0,0000000000

Размер файла: 431 байт

## === ТЕСТ 2: EXTERNALIZABLE ===

Сравнение исходной и десериализованной функции на отрезке  $[0, 10]$  с шагом 1:

x	Исходная	Десериализованная	Разница
0,0	0,0000000000	0,0000000000	0,0000000000
1,0	1,0000000000	1,0000000000	0,0000000000
2,0	2,0000000000	2,0000000000	0,0000000000
3,0	3,0000000000	3,0000000000	0,0000000000
4,0	4,0000000000	4,0000000000	0,0000000000
5,0	5,0000000000	5,0000000000	0,0000000000
6,0	6,0000000000	6,0000000000	0,0000000000
7,0	7,0000000000	7,0000000000	0,0000000000
8,0	8,0000000000	8,0000000000	0,0000000000
9,0	9,0000000000	9,0000000000	0,0000000000
10,0	10,0000000000	10,0000000000	0,0000000000

Размер файла: 241 байт

## === ТЕСТ 3: АНАЛИЗ ФАЙЛОВ ===

### 1. РАЗМЕРЫ ФАЙЛОВ:

Serializable: 431 байт

Externalizable: 241 байт

Разница: 190 байт

## 2. СОДЕРЖИМОЕ ФАЙЛОВ (первые 100 байт в hex):

Serializable:

```
AC ED 00 05 73 72 00 20 66 75 6E 63 74 69 6F 6E
73 2E 41 72 72 61 79 54 61 62 75 6C 61 74 65 64
46 75 6E 63 74 69 6F 6E 00 00 00 00 00 00 00 01
02 00 02 49 00 03 6C 65 6E 5B 00 05 66 75 6E 63
74 74 00 1A 5B 4C 66 75 6E 63 74 69 6F 6E 73 2F
46 75 6E 63 74 69 6F 6E 50 6F 69 6E 74 3B 78 70
00 00 00 0B
```

Externalizable:

```
AC ED 00 05 73 72 00 25 66 75 6E 63 74 69 6F 6E
73 2E 4C 69 6E 6B 65 64 4C 69 73 74 54 61 62 75
6C 61 74 65 64 46 75 6E 63 74 69 6F 6E 1B E4 70
C3 33 63 FD 50 0C 00 00 78 70 77 B4 00 00 00 0B
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3F F0 00 00 00 00 00 00 3F F0 00 00 00 00 00 00
40 00 00 00
```

## 3. ВЫВОДЫ О ПРЕИМУЩЕСТВАХ И НЕДОСТАТКАХ:

SERIALIZABLE:

Преимущества:

- Простая реализация (не требует дополнительных методов)
- Автоматическая сериализация всех полей
- Поддержка версионирования через serialVersionUID
- Обратная совместимость при добавлении полей

Недостатки:

- Большой размер файла (метаданные, ссылки)
- Медленнее (рефлексия, обработка графов объектов)
- Сериализует все поля, даже временные/кэшированные
- Может сериализовать лишние данные (например, кэш)

EXTERNALIZABLE:

Преимущества:

- Полный контроль над процессом сериализации
- Меньший размер файла (только необходимые данные)
- Быстрее (нет рефлексии, прямой ввод/вывод)
- Можно исключить временные поля (кэш, служебные данные)
- Оптимизация формата хранения

Недостатки:

- Требуется ручная реализация методов readExternal/writeExternal
- Больше кода для поддержки
- Нужно обновлять методы при изменении структуры класса
- Нет автоматической поддержки версионирования
- Легче допустить ошибку при реализации

Значения  $\sin(x)$  и  $\cos(x)$  на отрезке  $[0, \pi]$  с шагом 0.1:

x	$\sin(x)$	$\cos(x)$
-----		
0,00	0,000000	1,000000
0,10	0,099833	0,995004
0,20	0,198669	0,980067
0,30	0,295520	0,955336
0,40	0,389418	0,921061
0,50	0,479426	0,877583
0,60	0,564642	0,825336
0,70	0,644218	0,764842
0,80	0,717356	0,696707
0,90	0,783327	0,621610
1,00	0,841471	0,540302
1,10	0,891207	0,453596
1,20	0,932039	0,362358
1,30	0,963558	0,267499
1,40	0,985450	0,169967
1,50	0,997495	0,070737
1,60	0,999574	-0,029200
1,70	0,991665	-0,128844
1,80	0,973848	-0,227202
1,90	0,946300	-0,323290
2,00	0,909297	-0,416147
2,10	0,863209	-0,504846
2,20	0,808496	-0,588501
2,30	0,745705	-0,666276
2,40	0,675463	-0,737394
2,50	0,598472	-0,801144
2,60	0,515501	-0,856889
2,70	0,427380	-0,904072
2,80	0,334988	-0,942222

2,90	0,239249	-0,970958
3,00	0,141120	-0,989992
3,10	0,041581	-0,999135

=== ТЕСТ 2: ТАБУЛИРОВАННЫЕ АНАЛОГИ SIN И COS ===

Сравнение исходных и табулированных функций на отрезке  $[0, \pi]$  с шагом 0.1:

x	sin(x)	tab_sin(x)	cos(x)	tab_cos(x)
-----				
0,00	0,000000	0,000000	1,000000	1,000000
0,10	0,099833	0,097982	0,995004	0,982723
0,20	0,198669	0,195963	0,980067	0,965446
0,30	0,295520	0,293945	0,955336	0,948170
0,40	0,389418	0,385907	0,921061	0,914355
0,50	0,479426	0,472070	0,877583	0,864608
0,60	0,564642	0,558234	0,825336	0,814862
0,70	0,644218	0,643982	0,764842	0,764620
0,80	0,717356	0,707935	0,696707	0,688404
0,90	0,783327	0,771888	0,621610	0,612188
1,00	0,841471	0,835841	0,540302	0,535972
1,10	0,891207	0,883993	0,453596	0,450633
1,20	0,932039	0,918022	0,362358	0,357141
1,30	0,963558	0,952051	0,267499	0,263648
1,40	0,985450	0,984808	0,169967	0,169931
1,50	0,997495	0,984808	0,070737	0,070437
1,60	0,999574	0,984808	-0,029200	-0,029056
1,70	0,991665	0,984808	-0,128844	-0,128549
1,80	0,973848	0,966204	-0,227202	-0,224761
1,90	0,946300	0,932175	-0,323290	-0,318254
2,00	0,909297	0,898147	-0,416147	-0,411747
2,10	0,863209	0,862441	-0,504846	-0,504272
2,20	0,808496	0,798488	-0,588501	-0,580488
2,30	0,745705	0,734535	-0,666276	-0,656704
2,40	0,675463	0,670582	-0,737394	-0,732920
2,50	0,598472	0,594072	-0,801144	-0,794171
2,60	0,515501	0,507908	-0,856889	-0,843917
2,70	0,427380	0,421745	-0,904072	-0,893664
2,80	0,334988	0,334698	-0,942222	-0,940984
2,90	0,239249	0,236716	-0,970958	-0,958261
3,00	0,141120	0,138735	-0,989992	-0,975537
3,10	0,041581	0,040753	-0,999135	-0,992814

Максимальная погрешность для sin: 0,014766

Максимальная погрешность для cos: 0,014620

=== ТЕСТ 3: СУММА КВАДРАТОВ ТАБУЛИРОВАННЫХ SIN И COS ===

Количество точек табуляции: 10

x       $\sin^2(x) + \cos^2(x)$       Ожидаемое ( $\approx 1.0$ )

-----

0,00	1,0000000000	1,0000000000
0,10	0,9753452893	1,0000000000
0,20	0,9704883234	1,0000000000
0,30	0,9854291023	1,0000000000
0,40	0,9849684785	1,0000000000
0,50	0,9703975808	1,0000000000
0,60	0,9756244278	1,0000000000
0,70	0,9993578909	1,0000000000
0,80	0,9750730614	1,0000000000
0,90	0,9705859766	1,0000000000
1,00	0,9858966365	1,0000000000
1,10	0,9845147652	1,0000000000
1,20	0,9703137486	1,0000000000
1,30	0,9759104767	1,0000000000
1,40	0,9987226923	1,0000000000
1,50	0,9748077439	1,0000000000
1,60	0,9706905402	1,0000000000
1,70	0,9863710813	1,0000000000
1,80	0,9840679624	1,0000000000
1,90	0,9702368269	1,0000000000
2,00	0,9762034362	1,0000000000
2,10	0,9980944042	1,0000000000
2,20	0,9745493369	1,0000000000
2,30	0,9708020143	1,0000000000
2,40	0,9868524365	1,0000000000
2,50	0,9836280701	1,0000000000
2,60	0,9701668157	1,0000000000
2,70	0,9765033061	1,0000000000
2,80	0,9974730266	1,0000000000
2,90	0,9742978404	1,0000000000
3,00	0,9709203989	1,0000000000
3,10	0,9873407022	1,0000000000

Максимальное отклонение от 1.0: 0,0298331843

Количество точек табуляции: 20

x	$\sin^2(x)+\cos^2(x)$	Ожидаемое ( $\approx 1.0$ )
---	-----------------------	-----------------------------

0,00	1,0000000000	1,0000000000
0,10	0,9934801763	1,0000000000
0,20	0,9954813686	1,0000000000
0,30	0,9958763729	1,0000000000
0,40	0,9933589338	1,0000000000
0,50	0,9993625107	1,0000000000
0,60	0,9936326956	1,0000000000
0,70	0,9951176412	1,0000000000
0,80	0,9963026541	1,0000000000
0,90	0,9932689682	1,0000000000
1,00	0,9987562984	1,0000000000
1,10	0,9938164918	1,0000000000
1,20	0,9947851906	1,0000000000
1,30	0,9967602121	1,0000000000
1,40	0,9932102795	1,0000000000
1,50	0,9981813629	1,0000000000
1,60	0,9940315649	1,0000000000
1,70	0,9944840169	1,0000000000
1,80	0,9972490470	1,0000000000
1,90	0,9931828676	1,0000000000
2,00	0,9976377042	1,0000000000
2,10	0,9942779149	1,0000000000
2,20	0,9942141201	1,0000000000
2,30	0,9977691588	1,0000000000
2,40	0,9931867326	1,0000000000
2,50	0,9971253225	1,0000000000
2,60	0,9945555417	1,0000000000
2,70	0,9939755002	1,0000000000
2,80	0,9983205475	1,0000000000
2,90	0,9932218745	1,0000000000
3,00	0,9966442176	1,0000000000
3,10	0,9948644455	1,0000000000

Максимальное отклонение от 1.0: 0,0068171324

Количество точек табуляции: 50

x	$\sin^2(x)+\cos^2(x)$	Ожидаемое ( $\approx 1.0$ )
---	-----------------------	-----------------------------

0,00	1,0000000000	1,0000000000
0,10	0,9989873510	1,0000000000

0,20	0,9995678269	1,0000000000
0,30	0,9991045883	1,0000000000
0,40	0,9992528910	1,0000000000
0,50	0,9993390629	1,0000000000
0,60	0,9990551924	1,0000000000
0,70	0,9996907747	1,0000000000
0,80	0,9989747311	1,0000000000
0,90	0,9998518124	1,0000000000
1,00	0,9990115071	1,0000000000
1,10	0,9994564152	1,0000000000
1,20	0,9991655204	1,0000000000
1,30	0,9991782553	1,0000000000
1,40	0,9994367710	1,0000000000
1,50	0,9990173328	1,0000000000
1,60	0,9998252588	1,0000000000
1,70	0,9989736475	1,0000000000
1,80	0,9997151609	1,0000000000
1,90	0,9990471994	1,0000000000
2,00	0,9993565398	1,0000000000
2,10	0,9992379887	1,0000000000
2,20	0,9991151559	1,0000000000
2,30	0,9995460152	1,0000000000
2,40	0,9989910093	1,0000000000
2,50	0,9999712791	1,0000000000
2,60	0,9989841000	1,0000000000
2,70	0,9995900457	1,0000000000
2,80	0,9990944280	1,0000000000
2,90	0,9992682005	1,0000000000
3,00	0,9993219932	1,0000000000
3,10	0,9990635926	1,0000000000

Максимальное отклонение от 1.0: 0,0010263525

#### === ТЕСТ 4: ЭКСПОНЕНТА С СИМВОЛЬНЫМ ПОТОКОМ ===

Сравнение исходной и считанной функции на отрезке  $[0, 10]$  с шагом

1: x	Исходная $\exp(x)$	Считанная $\exp(x)$	Разница
-----			
0,0	1,0000000000	1,0000000000	0,0000000000
1,0	2,7182818285	2,7182818285	0,0000000000
2,0	7,3890560989	7,3890560989	0,0000000000
3,0	20,0855369232	20,0855369232	0,0000000000
4,0	54,5981500331	54,5981500331	0,0000000000

5,0	148,4131591026	148,4131591026	0,0000000000
6,0	403,4287934927	403,4287934927	0,0000000000
7,0	1096,6331584285	1096,6331584285	0,0000000000
8,0	2980,9579870417	2980,9579870417	0,0000000000
9,0	8103,0839275754	8103,0839275754	0,0000000000
10,0	22026,4657948067	22026,4657948067	0,0000000000

### === ТЕСТ 5: ЛОГАРИФМ С БАЙТОВЫМ ПОТОКОМ ===

Сравнение исходной и считанной функции на отрезке [0.1, 10] с шагом 1:

x	Исходная ln(x)	Считанная ln(x)	Разница
-----			
0,1	-2,3025850930	-2,3025850930	0,0000000000
1,1	0,0927048700	0,0927048700	0,0000000000
2,1	0,7402327355	0,7402327355	0,0000000000
3,1	1,1301474226	1,1301474226	0,0000000000
4,1	1,4099993481	1,4099993481	0,0000000000
5,1	1,6284294437	1,6284294437	0,0000000000
6,1	1,8076029631	1,8076029631	0,0000000000
7,1	1,9595024837	1,9595024837	0,0000000000
8,1	2,0913442118	2,0913442118	0,0000000000
9,1	2,2078123463	2,2078123463	0,0000000000

### === ТЕСТ 6: АНАЛИЗ ФАЙЛОВ ===

#### 1. СОДЕРЖИМОЕ ТЕКСТОВОГО ФАЙЛА (exp\_text.txt):

Содержимое: 11 0.0 1.0 1.0 2.718281828459045 2.0 7.38905609893065 3.0  
20.085536923187668 4.0 54.598150033144236 5.0 148.4131591025766 6.0  
403.4287934927351 7.0 1096.6331584284585 8.0 2980.9579870417283 9.0  
8103.083927575384 10.0 22026.465794806718

Размер файла: 235 байт

#### 2. СОДЕРЖИМОЕ БАЙТОВОГО ФАЙЛА (ln\_binary.bin):

Размер файла: 180 байт

Первые 50 байт (в шестнадцатеричном виде): 00 00 00 0B 3F B9 99 99 99 99 99 9A C0  
02 6B B1 BB B5 55 15 3F F1 70 A3 D7 0A 3D 71 3F B6 0F BD D2 FF FC 37 40 00 A3 D7  
0A 3D 70 A4 3F E7 6F 8E CB 04

#### 3. ВЫВОДЫ О ПРЕИМУЩЕСТВАХ И НЕДОСТАТКАХ: ТЕКСТОВЫЙ ФОРМАТ:

Преимущества:

- Человекочитаемый формат



- Легко отлаживать и проверять
- Можно редактировать вручную
- Кроссплатформенн

ый Недостатки:

- Большой размер файла
- Медленнее парсинг
- Зависимость от локали (разделитель десятичных чисел)

## БАЙТОВЫЙ ФОРМАТ:

Преимущества:

- Компактный размер файла
- Быстрый ввод/вывод
- Точное представление чисел (без потери точности при парсинге)
- Независимость от

локали Недостатки:

- Нечитаемый формат
- Сложнее отлаживать
- Невозможно редактировать вручную
- Зависимость от порядка байт (endianness)