ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА НОМЕР 4

Студент Горейнов Д.В.

Группа 6301-030301D

Проверил Борисов.Д.С

Оценка _____

САМАРА 2025

Задание номер 1

В классах ArrayTabulatedFunction и LinkedListTabulatedFunction добавил конструкторы, получающие сразу все точки функции в виде массива объектов типа FunctionPoint. Если точек задано меньше двух, или если точки в массиве не упорядочены по значению абсциссы, конструкторы должны выбрасывать исключение IllegalArgumentException. При написании конструкторов обеспечил корректную инкапсуляцию.(фото 1-2)

```
public LinkedListTabulatedFunction(FunctionPoint[] points) throws IllegalArgumentException {
if (points.length < 2) {
    throw new IllegalArgumentException("Number of points is less than 2");
}

// Проверка упорядоченности точек
for (int i = 1; i < points.length; i++) {
    if (points[i].getX() <= points[i - 1].getX()) {
        throw new IllegalArgumentException("Points are not ordered by X coordinate");
    }
}

head = new FunctionNode();
head.setNext(head);
head.setPrev(head);
pointsCount = 0;

for (int i = 0; i < points.length; i++) {
    FunctionPoint point = new FunctionPoint(points[i].getX(), points[i].getY());
    addNodeToTail().setPoint(point);
}
}
```

```
public ArrayTabulatedFunction(FunctionPoint[] points) {

if (points.length < 2) {
    throw new IllegalArgumentException("Number of points is less than 2");
}


for (int i = 1; i < points.length; i++) {
    if (points[i].getX() <= points[i - 1].getX()) {
        throw new IllegalArgumentException("Points are not ordered by X coordinate");
    }
}


this.pointsCount = points.length;
this.points = new FunctionPoint[pointsCount];
for (int i = 0; i < pointsCount; i++) {
    this.points[i] = new FunctionPoint(points[i].getX(), points[i].getY());
}
}
```

Фото 1-2

Здание номер 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.

Исключил соответствующие методы из интерфейса TabulatedFunction и сделал так, чтобы он расширял интерфейс Function.

```java
package functions;

public interface Function {
    double getLeftDomainBorder();
    double getRightDomainBorder();
    double getFunctionValue(double x);

}

```

Фото 3

```java
package functions;

public interface TabulatedFunction extends Function{

    void deletePoint(int index) throws FunctionPointIndexOutOfBoundsException;
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
    int getPointsCount();
    FunctionPoint getPoint(int index) throws FunctionPointIndexOutOfBoundsException;
    void setPoint(int index, FunctionPoint point) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;
    double getPointX(int index) throws FunctionPointIndexOutOfBoundsException;
    void setPointX(int index, double x) throws FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException;
    double getPointY(int index) throws FunctionPointIndexOutOfBoundsException;
    void setPointY(int index, double y) throws FunctionPointIndexOutOfBoundsException;

}

```

Фото 4

Задание 3

Создал пакет functions.basic, в нём описаны классы ряда функций, заданных аналитически:cos,log,sin,tan,exp(фото 5)

Cos.java	19.11.2017 20:24	Исходный файл J...	1 КБ
Exp.java	10.11.2025 16:20	Исходный файл J...	1 КБ
Log.java	22.11.2017 22:38	Исходный файл J...	1 КБ
Sin.java	22.11.2017 22:35	Исходный файл J...	1 КБ
Tan.java	15.11.2017 23:00	Исходный файл J...	1 КБ

Фото 5

Создал класс TrigonometricFunction, реализующий интерфейс Function и описывающий методы получения границ области определения.(фото 6).

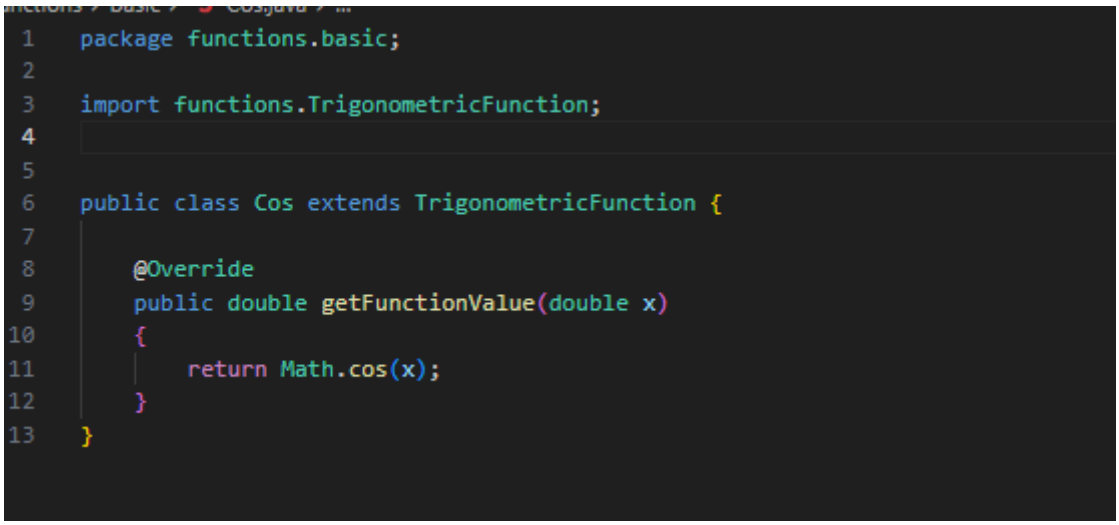
```java
package functions;

public class TrigonometricFunction implements Function{
    public double getLeftDomainBorder()
    {
        return Double.NEGATIVE_INFINITY;
    }
    public double getRightDomainBorder()
    {
        return Double.POSITIVE_INFINITY;
    }

    public double getFunctionValue(double x)
    {
        return 0;
    }
}
```

Фото 6

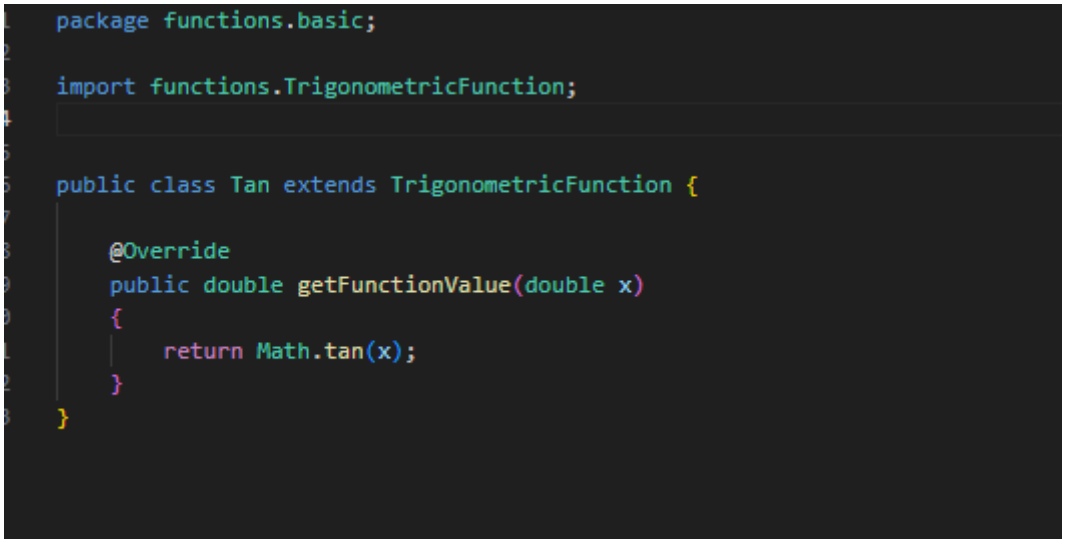
Создал наследующие от него публичные классы Sin, Cos и Tan, объекты которых вычисляют, соответственно, значения синуса, косинуса и тангенса. Для получения значений следует воспользоваться методами Math.sin(), Math.cos() и Math.tan()(фото 7-9)

@Override— встроенная аннотация в языке Java, которая указывает компилятору, что текущий метод переопределяет метод из суперкласса(родительский класс)

```java
package functions.basic;

import functions.TrigonometricFunction;


public class Sin extends TrigonometricFunction{

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



```java
package functions.basic;

import functions.TrigonometricFunction;


public class Cos extends TrigonometricFunction {

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



```java
package functions.basic;

import functions.TrigonometricFunction;


public class Tan extends TrigonometricFunction {

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

Фото 7-9

Задание 4

Создал пакет functions.meta, в нём описаны классы функций, позволяющие комбинировать функции:sum,mult,power,scale,shift,composition.

Sum, объекты которого представляют собой функции, являющиеся суммой двух других функций.

Mult, объекты которого представляют собой функции, являющиеся произведением двух других функций.

Power, объекты которого представляют собой функции, являющиеся степенью другой функции

Scale, объекты которого описывают функции, полученные из исходных функций путём масштабирования вдоль осей координат.

Shift, объекты которого описывают функции, полученные из исходных функций путём сдвига вдоль осей координат.

Composition, объекты которого описывают композицию двух исходных функций (Фото 10-15)

```java
package functions.meta;

import functions.Function;

public class Sum implements Function
{
    Function a,b;
    public Sum(Function a, Function b)
    {
        this.a=a;
        this.b=b;
    }

    @Override
    public double getLeftDomainBorder()
    {
        return Math.max(a.getLeftDomainBorder(), b.getLeftDomainBorder());

    }

    @Override
    public double getRightDomainBorder()
    {
        return Math.min(a.getRightDomainBorder(), b.getRightDomainBorder());
    }

    @Override
    public double getFunctionValue(double x)
    {
        return a.getFunctionValue(x)+b.getFunctionValue(x);
    }
}
```

```java
package functions.meta;

import functions.Function;

public class Mult implements Function
{
    Function a,b;

    public Mult(Function a,Function b)

    {
        this.a=a;
        this.b=b;
    }


    @Override
    public double getLeftDomainBorder()
    {
        return Math.max(a.getLeftDomainBorder(),b.getLeftDomainBorder());
    }

    @Override
    public double getRightDomainBorder()
    {
        return Math.min(a.getLeftDomainBorder(),b.getLeftDomainBorder());
    }

    @Override
    public double getFunctionValue(double x)
    {
        return a.getFunctionValue(x)*b.getFunctionValue(x);
    }



}
```

```java
package functions.meta;
import functions.Function;

public class Power implements Function {

    Function a;
    double b;

    public Power(Function a, double b)
    {
        this.a=a;
        this.b=b;
    }

    @Override
    public double getLeftDomainBorder()
    {
        return a.getLeftDomainBorder();
    }

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

    @Override
    public double getFunctionValue(double x)
    {
        return Math.pow(a.getFunctionValue(x),b);
    }
}
```

```java
package functions.meta;
import functions.Function;

public class Scale implements Function {
    Function a;
    double kx,ky;

    public Scale(Function a, double kx,double ky)
    {
        this.a=a;
        this.kx=kx;
        this.ky=ky;
    }

    @Override
    public double getLeftDomainBorder()
    {
        return a.getLeftDomainBorder();
    }

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

    @Override

    public double getFunctionValue(double x)
    {
        return a.getFunctionValue(x*kx)*ky;
    }
}


```

```java
package functions.meta;

import functions.Function;

public class Shift implements Function
{
Function a;
double plus_x, plus_y;

    public Shift(Function a,double plus_x,double plus_y)
    {
        this.a=a;
        this.plus_x=plus_x;
        this.plus_y=plus_y;
    }


    @Override
    public double getLeftDomainBorder()
    {
        return a.getLeftDomainBorder();
    }

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

    @Override
    public double getFunctionValue(double x)
    {
        return a.getFunctionValue(x+plus_x)+plus_y;
    }

}
```

```java
package functions.meta;

import functions.Function;

public class Composition implements Function
{
    Function a,b;

    public Composition(Function a, Function b)
    {
        this.a=a;
        this.b=b;
    }

    @Override
    public double getLeftDomainBorder()
    {
        return a.getLeftDomainBorder();
    }

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

    @Override
    public double getFunctionValue(double x)
    {
        return a.getFunctionValue(b.getFunctionValue(x));

    }

}
```

Фото 10-15

Задание 5

В пакете functions создал класс Functions, содержащий вспомогательные статические методы для работы с функциями. Сделал так, чтобы в программе вне этого класса нельзя было создать его объект. Класс содержит следующие методы:

- 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.

```
package functions;

import functions.meta.*;

public final class Functions {
public static Function shift(Function f, double shiftX, double shiftY)
{
    return new Shift(f,shiftX,shiftY);

}

public static Function scale(Function f, double scaleX, double scaleY)
{
    return new Scale(f,scaleX,scaleY);

}

public static Function power(Function f, double power)
{
    return new Power(f, power);
}

public static Function sum(Function f1, Function f2)
{
    return new Sum(f1,f2);
}

public static Function mult(Function f1, Function f2)
{
    return new Mult(f1,f2);
}

public static Function composition(Function f1, Function f2)
{
    return new Composition(f1,f2);
}

}
```

Фото 16

Задание 6

В пакете functions создал класс TabulatedFunctions, содержащий вспомогательные статические методы для работы с табулированными функциями. Сделал так, чтобы в программе вне этого класса нельзя было создать его объект.

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.

Если указанные границы для табулирования выходят за область определения функции, метод выбрасывает исключение IllegalArgumentException.

```
package functions;

import java.io.*;

public final class TabulatedFunctions {

    public static TabulatedFunction tabulate(Function function, double leftX, double rightX, int pointsCount) {
        // Проверяем, что интервал внутри области определения
        if (leftX < function.getLeftDomainBorder() || rightX > function.getRightDomainBorder()) {
            throw new IllegalArgumentException("Заданный интервал выходит за границы области определения функции");
        }

        if (pointsCount < 2) {
            throw new IllegalArgumentException("Количество точек должно быть не менее 2");
        }

        FunctionPoint[] points = new FunctionPoint[pointsCount];
        double step = (rightX - leftX) / (pointsCount - 1);

        for (int i = 0; i < pointsCount; i++) {
            double x = leftX + i * step;
            double y = function.getFunctionValue(x);
            points[i] = new FunctionPoint(x, y);
        }

        return new ArrayTabulatedFunction(points);
    }
```

Фото 17

Задание 7

В класс TabulatedFunctions добавьте следующие методы.

Метод вывода табулированной функции в байтовый поток public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out в указанный поток выводит значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек.

Метод ввода табулированной функции из байтового потока public static TabulatedFunction inputTabulatedFunction(InputStream in) считывает из указанного потока данные о табулированной функции, создаёт и настраивает её объект и возвращает его из метода.

Метод записи табулированной функции в символьный поток public static void writeTabulatedFunction(TabulatedFunction function, Writer out) в указанный поток выводит значения, по которым потом можно будет восстановить табулированную функцию, а именно количество точек в ней и значения координат точек.

Метод чтения табулированной функции из символьного потока public static TabulatedFunction readTabulatedFunction(Reader in) считывает из указанного потока данные о табулированной функции, создает и настраивает её объект и возвращает его из метода.

Подумайте и обоснуйте, как следует в этих методах поступить с возникающим исключением IOException:

Объявляется throws IOException

Пробрасываются исключение вызывающему коду

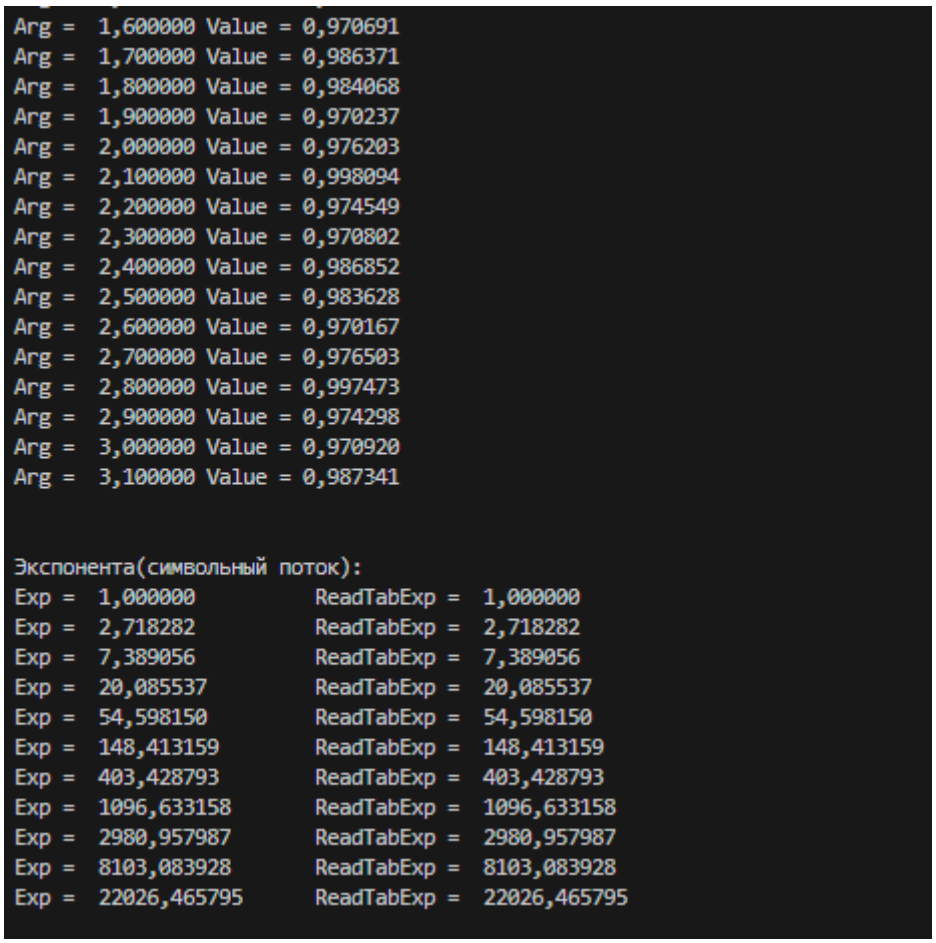
Поток не следует закрывать, так как если бы метод закрыл поток, клиент получил бы исключение при попытке дописать данные.

```java
public static void outputTabulatedFunction(TabulatedFunction function, OutputStream out) throws IOException {
    DataOutputStream dataOut = new DataOutputStream(out);
    int pointCount = function.getPointsCount();
    dataOut.writeInt(pointCount);

    for (int i = 0; i < pointCount; i++) {
        dataOut.writeDouble(function.getPointX(i));
        dataOut.writeDouble(function.getPointY(i));
    }
    dataOut.flush();
}

public static TabulatedFunction inputTabulatedFunction(InputStream in) throws IOException {
    DataInputStream dataIn = new DataInputStream(in);
    int pointCount = dataIn.readInt();
    FunctionPoint[] data = new FunctionPoint[pointCount];

    for (int i = 0; i < pointCount; i++) {
        double x = dataIn.readDouble();
        double y = dataIn.readDouble();
        data[i] = new FunctionPoint(x, y);
    }

    return new ArrayTabulatedFunction(data);
}

public static void writeTabulatedFunction(TabulatedFunction function, Writer out) throws IOException {
    PrintWriter writer = new PrintWriter(out);
    int pointCount = function.getPointsCount();
    writer.println(pointCount);

    for (int i = 0; i < pointCount; i++) {
        writer.println(function.getPointX(i));
        writer.println(function.getPointY(i));
    }
    writer.flush();
}

public static TabulatedFunction readTabulatedFunction(Reader in) throws IOException {
    StreamTokenizer st = new StreamTokenizer(in);
    st.nextToken();
    int itemsCount = (int) st.nval;
    FunctionPoint[] data = new FunctionPoint[itemsCount];

    for (int i = 0; i < itemsCount; i++) {
        st.nextToken();
        double x = st.nval;
        st.nextToken();
        double y = st.nval;
        data[i] = new FunctionPoint(x, y);
    }

    return new ArrayTabulatedFunction(data);
```

Фото 18

Задание 8

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


Сумма квадратов синуса и косинуса:
Arg =  0,000000 Value = 1,000000
Arg =  0,100000 Value = 0,975345
Arg =  0,200000 Value = 0,970488
Arg =  0,300000 Value = 0,985429
Arg =  0,400000 Value = 0,984968
Arg =  0,500000 Value = 0,970398
Arg =  0,600000 Value = 0,975624
Arg =  0,700000 Value = 0,999358
Arg =  0,800000 Value = 0,975073
Arg =  0,900000 Value = 0,970586
Arg =  1,000000 Value = 0,985897
Arg =  1,100000 Value = 0,984515
Arg =  1,200000 Value = 0,970314
Arg =  1,300000 Value = 0,975910
Arg =  1,400000 Value = 0,998723
```



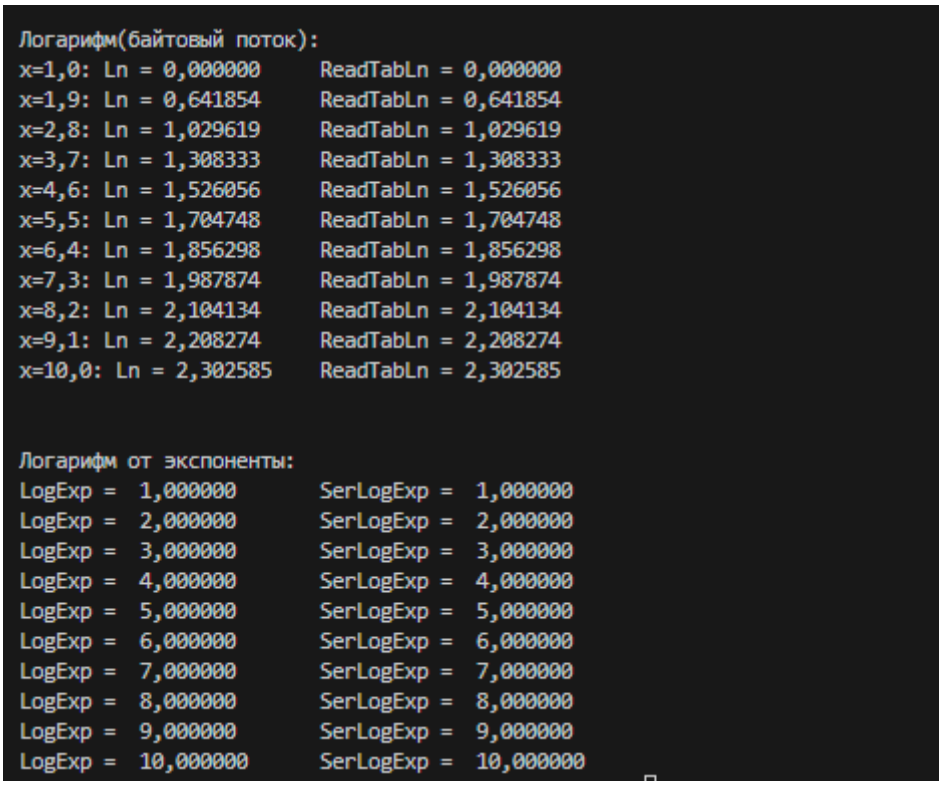


Фото 19-21

Вывод:

Плюсы:

Байтовый поток точно сохраняет значения функции и занимает меньше памяти.

Минусы:

Трудно читать пользователю

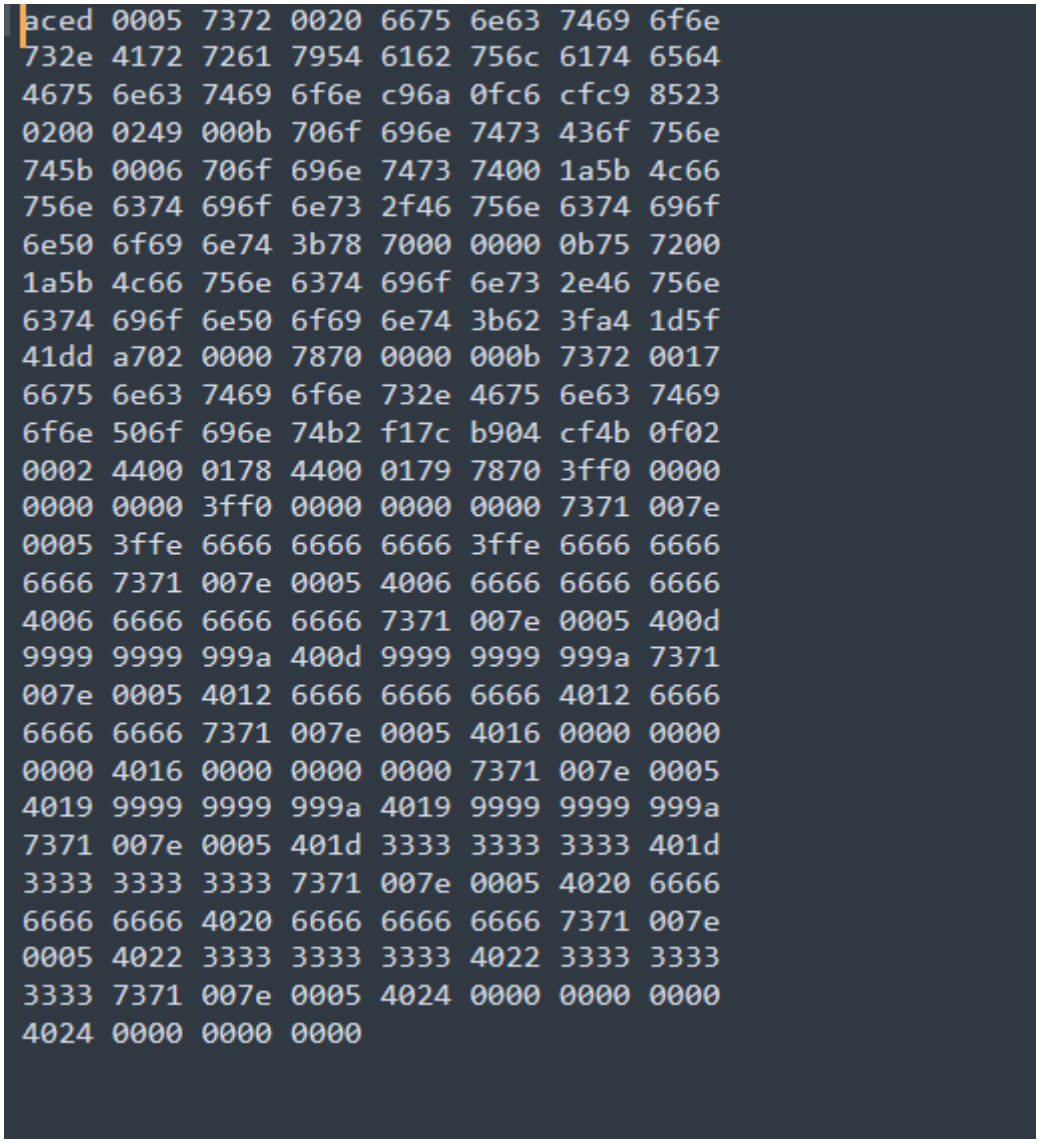


Фото 22

Задание 9

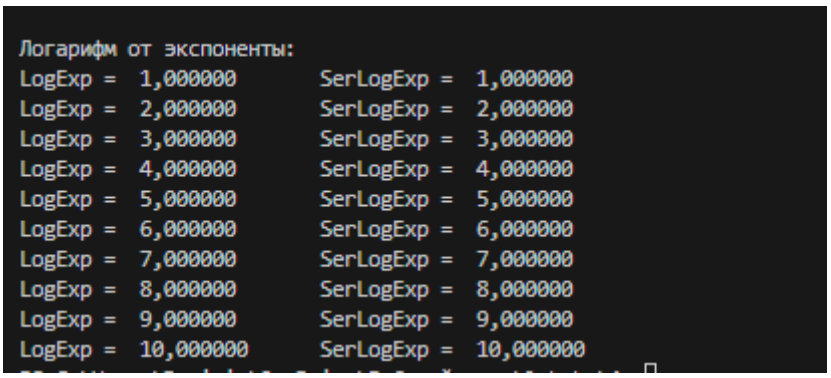


Фото 23

Плюсы Serializable:

Простота использования
Автоматическое управление

Минусы: Serializable

Медленнее

Низкая безопасность

Плюсы Externalizable:

Полный контроль
Быстрый
Высокая защита

Минусы Externalizable:

Сложнее реализовать
Требует ручного управления в методах