Отчет по лабораторной работе №4

«Аналитические функции и методы вывода табулированных функций»

Выполнила: Качкуркина Арина Валерьевна

Группа: 6204-010302D

Год:2025

Задание 1

Я добавила в классы ArrayTabulatedFunction и LinkedListTabulatedFunction конструкторы, которые принимают готовый массив точек. Эти конструкторы проверяют:

- что точек не меньше двух (иначе функция не может быть определена);

- что точки упорядочены по возрастанию координаты X.

При нарушении любого из условий выбрасывается понятное исключение IllegalArgumentException с описанием проблемы.

Задание 2

Создала интерфейс Function, который стал базовым для всех функций в программе. Он содержит три основных метода:

- получение левой и правой границ области определения

- вычисление значения функции в точке.

Интерфейс TabulatedFunction теперь наследуется от Function - табулированные функции являются частным случаем функций одной переменной.

Задание 3

Реализовала пакет functions.basic с аналитическими функциями:

```java
package functions.basic;

import functions.Function;

//экспонента: f(x) = e^x
public class Exp implements Function {  2 usages

    //область определения: все действительные числа(-oo,+oo)
    public double getLeftDomainBorder() { return Double.NEGATIVE_INFINITY;//-oo }

    public double getRightDomainBorder() { return Double.POSITIVE_INFINITY;//+oo }

    //вычисление значения ф-ии в точке x
    public double getFunctionValue(double x) { return Math.exp(x);  //e^x }
}
```

- Exp - экспонента, использует стандартную библиотеку Java

```java
package functions.basic;                                                    1  3

import functions.Function;
//логарифм: f(x) = log a (x)
public class Log implements Function {  2 usages
    private double base;  //основание лог.  2 usages

    //Конструктор получает основание лог.
    public Log(double base) {  2 usages
        if (base <= 0 || Math.abs(base - 1.0) < 1e-10) {
            throw new IllegalArgumentException("Основание логарифма должно быть > 0 и =! 1");
        }
        this.base = base;
    }

    //область определения: x > 0
    public double getLeftDomainBorder() { return 0.0;  //лог. определен только для x > 0 }

    public double getRightDomainBorder() { return Double.POSITIVE_INFINITY;//+oo }

    //вычисление логарифма: log a(x) = ln(x) / ln(a)
    public double getFunctionValue(double x) {
        if (x <= 0) {
            return Double.NaN;  //для x ≤ 0
        }
    //формула: log a (x) = ln(x) / ln(a)
        return Math.log(x) / Math.log(base);
    }
}
```

- Log - логарифм с любым основанием

```java
package functions.basic;

//f(x) = sin(x)
public class Sin extends TrigonometricFunction {  1 usage

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

```java
package functions.basic;

//f(x) = tan(x)
public class Tan extends TrigonometricFunction {  no usages

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

```java
package functions.basic;

//f(x) = cos(x)
public class Cos extends TrigonometricFunction {  1 usage

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

- Тригонометрические функции (Sin, Cos, Tan) - наследуются от общего базового класса TrigonometricFunction.

Задание 4

Создала пакет functions.meta для комбинирования функций. Реализовала:

```java
package functions.meta;                                                    ⚠2 ✓4

import functions.Function;
//сумма ф-ий: f(x) = f1(x) + f2(x)
public class Sum implements Function {  1 usage
    private Function f1;  //первая функция  4 usages
    private Function f2;  //вторая функция  4 usages

    //Конструктор получает две функции
    public Sum(Function f1, Function f2) {  1 usage
        this.f1 = f1;
        this.f2 = f2;
    }

    //область определения - пересечение о.о. f1 и f2
    public double getLeftDomainBorder() { return Math.max(f1.getLeftDomainBorder(), f2.getLeftDoma

    public double getRightDomainBorder() { return Math.min(f1.getRightDomainBorder(), f2.getRightD

    //знач. суммы ф-ий в точке x
    public double getFunctionValue(double x) {
        //проверка, что x в о.о.
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return f1.getFunctionValue(x) + f2.getFunctionValue(x);
    }
}
```

```java
package functions.meta;

import functions.Function;

//f(x) = f1(x) * f2(x)
public class Mult implements Function {  1 usage
    private Function f1;  4 usages
    private Function f2;  4 usages
    //Конструктор
    public Mult(Function f1, Function f2) {  1 usage
    //сохраняем переданные ф-ии в поля класса
        this.f1 = f1;
        this.f2 = f2;
    }

    public double getLeftDomainBorder() { return Math.max(f1.getLeftDomainBorder(), f2.getLeftDomainBorder()); }

    public double getRightDomainBorder() { return Math.min(f1.getRightDomainBorder(), f2.getRightDomainBorder()); }
    //значение произведения функций в точке x
    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;//если x вне области определения
        }
        return f1.getFunctionValue(x) * f2.getFunctionValue(x);
    }
}
```

- Sum и Mult - арифметические операции

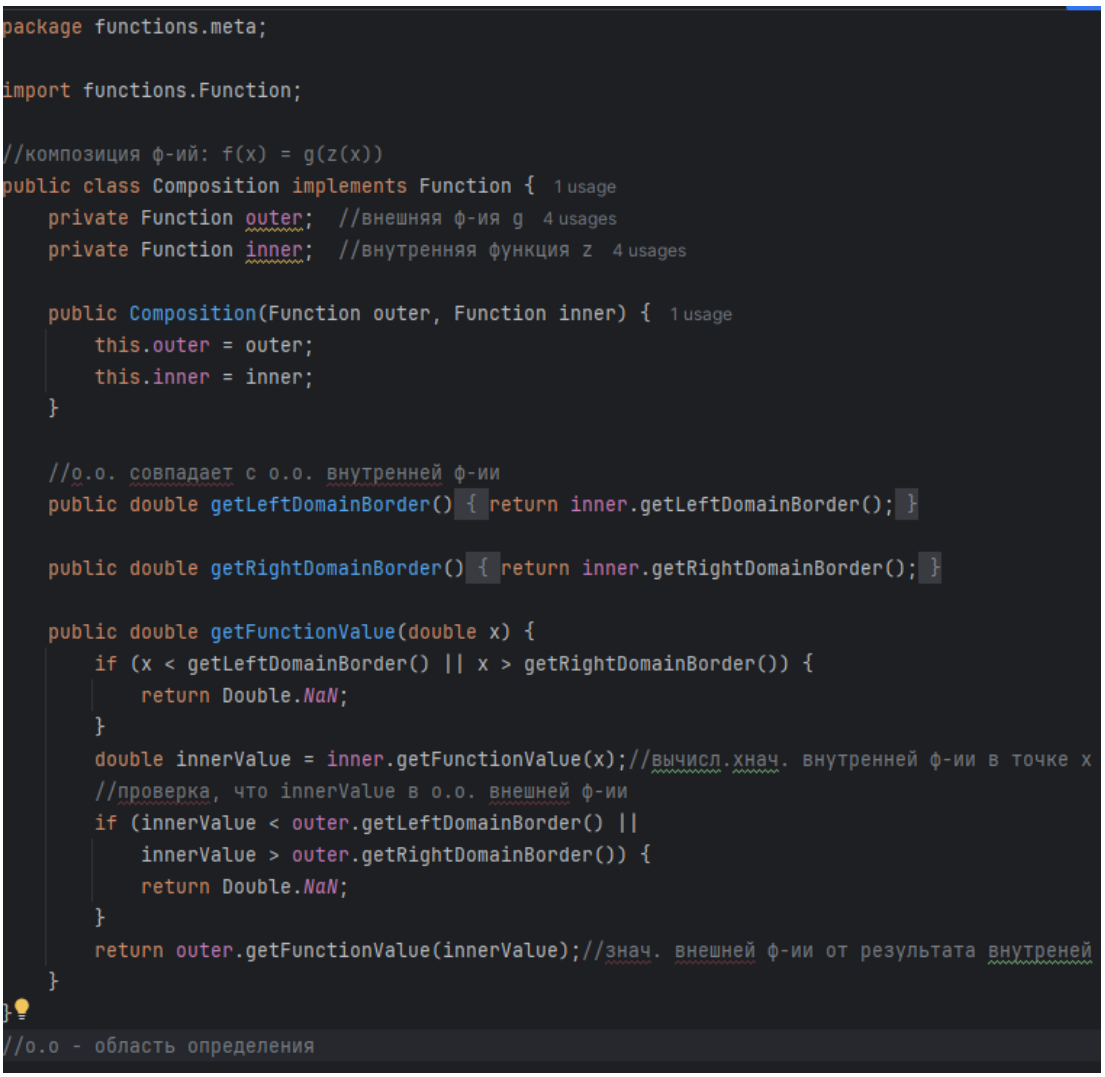
```java
package functions.meta;

import functions.Function;
//f(x) = [g(x)]^power
public class Power implements Function {  1 usage
    private Function baseFunction;  //базовая ф-ия  4 usages
    private double power;           //степень  2 usages

    public Power(Function baseFunction, double power) {  1 usage
        this.baseFunction = baseFunction;
        this.power = power;
    }

    //о.о. совпадает с о.о. базовой ф-ии
    public double getLeftDomainBorder() { return baseFunction.getLeftDomainBorder(); }

    public double getRightDomainBorder() { return baseFunction.getRightDomainBorder(); }

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        double baseValue = baseFunction.getFunctionValue(x);//вычисляем значение базовой ф-ии в точке x
        return Math.pow(baseValue, power);//возводим полученное знач. в заданную степень
    }
}
//о.о.- область определения
```

- Power - возведение в степень

```java
import functions.Function;
//масштабирование функции: f(x) = scaleY * g(scaleX * x)
public class Scale implements Function {  1 usage
    private Function function;  //исходная функция  6 usages
    private double scaleX;      //коэфф. масштабирования по x  10 usages
    private double scaleY;      //коэфф. масштабирования по y  2 usages

    public Scale(Function function, double scaleX, double scaleY) {  1 usage
        this.function = function;
        this.scaleX = scaleX;
        this.scaleY = scaleY;
    }

    //о.о. масштабируется по x
    public double getLeftDomainBorder() {
        if (scaleX > 0) {
            return function.getLeftDomainBorder() / scaleX;
        } else if (scaleX < 0) {
            return function.getRightDomainBorder() / scaleX;
        } else {
            return Double.NaN; //scaleX = 0 - ф-ия не определена
        }
    }

    public double getRightDomainBorder() {
        if (scaleX > 0) {
            return function.getRightDomainBorder() / scaleX;
        } else if (scaleX < 0) {
            return function.getLeftDomainBorder() / scaleX;
        } else {
            return Double.NaN;
        }
    }

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return scaleY * function.getFunctionValue( x: scaleX * x);
```

```java
package functions.meta;

import functions.Function;
//сдвиг ф-ии (+)
public class Shift implements Function {  1 usage
    private Function function;//исходная ф-ия  4 usages
    private double shiftX;//сдвиг по x  4 usages
    private double shiftY;//сдвиг по y  2 usages

    public Shift(Function function, double shiftX, double shiftY) {  1 usage
        this.function = function;
        this.shiftX = shiftX;
        this.shiftY = shiftY;
    }

    //о.о. сдвигается по x
    public double getLeftDomainBorder() { return function.getLeftDomainBorder() + shiftX; }

    public double getRightDomainBorder() { return function.getRightDomainBorder() + shiftX; }

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        return shiftY + function.getFunctionValue( x: x - shiftX);
    }
}
```

- Scale и Shift - геометрические преобразования



```java
package functions.meta;

import functions.Function;

//композиция ф-ий: f(x) = g(z(x))
public class Composition implements Function {  1 usage
    private Function outer;  //внешняя ф-ия g  4 usages
    private Function inner;  //внутренняя функция z  4 usages

    public Composition(Function outer, Function inner) {  1 usage
        this.outer = outer;
        this.inner = inner;
    }

    //о.о. совпадает с о.о. внутренней ф-ии
    public double getLeftDomainBorder() { return inner.getLeftDomainBorder(); }

    public double getRightDomainBorder() { return inner.getRightDomainBorder(); }

    public double getFunctionValue(double x) {
        if (x < getLeftDomainBorder() || x > getRightDomainBorder()) {
            return Double.NaN;
        }
        double innerValue = inner.getFunctionValue(x);//вычисл.знач. внутренней ф-ии в точке x
        //проверка, что innerValue в о.о. внешней ф-ии
        if (innerValue < outer.getLeftDomainBorder() ||
            innerValue > outer.getRightDomainBorder()) {
            return Double.NaN;
        }
        return outer.getFunctionValue(innerValue);//знач. внешней ф-ии от результата внутреней
    }
}
//о.о - область определения
```

- Composition - композиция функций

В Классе Scale нужно было корректно обработать отрицательные коэффициенты масштабирования.

Задание 5

Класс Functions содержит статические методы для удобного создания комбинированных функций. Например, вместо прямого создания объекта Sum можно использовать Functions.sum(f1, f2)

Сделала конструктор приватным, чтобы предотвратить создание экземпляров этого служебного класса.

Задание 6

Метод TabulatedFunctions.tabulate() позволяет преобразовать любую аналитическую функцию в табулированную. Он проверяет, что запрошенный интервал попадает в область определения исходной функции.

Задание 7

Реализовала четыре метода для работы с потоками:

- Байтовые потоки - для эффективного хранения

- Символьные потоки - для читаемого формата

При работе с символьными потоками использовала StreamTokenizer для удобного разбора чисел. Исключения IOException преобразую в RuntimeException для упрощения кода использования.

Задание 8

Протестировала все возможности системы:

1. Базовые функции - убедилась, что Sin и Cos работают корректно

2. Математические тождества - проверила, что sin^2(x) + cos^2(x) =1

3. Ввод-вывод - сравнила разные форматы хранения

Выводы по форматам:

- Текстовый формат удобен для отладки, но занимает больше места

- Бинарный формат компактен и быстр, но не читаем человеком

Задание 9

Реализовала сериализацию через Serializable. Протестировала на композиции ln(exp(x)) - после сериализации и десериализации функция полностью сохраняет свои свойства.

В ходе работы успешно расширена функциональность пакета для работы с функциями. Верно работают сериализации, все значения сохраняются без потерь.

ИТОГ РАБОТЫ ПРОГРАММЫ:

1. Sin и Cos от 0 до π с шагом 0.1:

Sin:

x=0,0: 0,0000

x=0,1: 0,0998

x=0,2: 0,1987

x=0,3: 0,2955

x=0,4: 0,3894

x=0,5: 0,4794

x=0,6: 0,5646

x=0,7: 0,6442

x=0,8: 0,7174

x=0,9: 0,7833

x=1,0: 0,8415

x=1,1: 0,8912

x=1,2: 0,9320

x=1,3: 0,9636

x=1,4: 0,9854

x=1,5: 0,9975

x=1,6: 0,9996

x=1,7: 0,9917

x=1,8: 0,9738

x=1,9: 0,9463

x=2,0: 0,9093

x=2,1: 0,8632

x=2,2: 0,8085

x=2,3: 0,7457

x=2,4: 0,6755

x=2,5: 0,5985

x=2,6: 0,5155

x=2,7: 0,4274

x=2,8: 0,3350

x=2,9: 0,2392

x=3,0: 0,1411

x=3,1: 0,0416

Cos:

x=0,0: 1,0000

x=0,1: 0,9950

x=0,2: 0,9801

x=0,3: 0,9553

x=0,4: 0,9211

x=0,5: 0,8776

x=0,6: 0,8253

x=0,7: 0,7648

x=0,8: 0,6967

x=0,9: 0,6216

x=1,0: 0,5403

x=1,1: 0,4536

x=1,2: 0,3624

x=1,3: 0,2675

x=1,4: 0,1700

x=1,5: 0,0707

x=1,6: -0,0292

x=1,7: -0,1288

x=1,8: -0,2272

x=1,9: -0,3233

x=2,0: -0,4161

x=2,1: -0,5048

x=2,2: -0,5885

x=2,3: -0,6663

x=2,4: -0,7374

x=2,5: -0,8011

x=2,6: -0,8569

x=2,7: -0,9041

x=2,8: -0,9422

x=2,9: -0,9710

x=3,0: -0,9900

x=3,1: -0,9991

2. Табулированные аналоги :

Сравнение Sin:

x=0,0: аналитич. = 0,0000; табул. = 0,0000

x=0,1: аналитич. = 0,0998; табул. = 0,0980

x=0,2: аналитич. = 0,1987; табул. = 0,1960

x=0,3: аналитич. = 0,2955; табул. = 0,2939

x=0,4: аналитич. = 0,3894; табул. = 0,3859

x=0,5: аналитич. = 0,4794; табул. = 0,4721

x=0,6: аналитич. = 0,5646; табул. = 0,5582

x=0,7: аналитич. = 0,6442; табул. = 0,6440

x=0,8: аналитич. = 0,7174; табул. = 0,7079

x=0,9: аналитич. = 0,7833; табул. = 0,7719

x=1,0: аналитич. = 0,8415; табул. = 0,8358

x=1,1: аналитич. = 0,8912; табул. = 0,8840

x=1,2: аналитич. = 0,9320; табул. = 0,9180

x=1,3: аналитич. = 0,9636; табул. = 0,9521

x=1,4: аналитич. = 0,9854; табул. = 0,9848

x=1,5: аналитич. = 0,9975; табул. = 0,9848

x=1,6: аналитич. = 0,9996; табул. = 0,9848

x=1,7: аналитич. = 0,9917; табул. = 0,9848

x=1,8: аналитич. = 0,9738; табул. = 0,9662

x=1,9: аналитич. = 0,9463; табул. = 0,9322

x=2,0: аналитич. = 0,9093; табул. = 0,8981

x=2,1: аналитич. = 0,8632; табул. = 0,8624

x=2,2: аналитич. = 0,8085; табул. = 0,7985

x=2,3: аналитич. = 0,7457; табул. = 0,7345

x=2,4: аналитич. = 0,6755; табул. = 0,6706

x=2,5: аналитич. = 0,5985; табул. = 0,5941

x=2,6: аналитич. = 0,5155; табул. = 0,5079

x=2,7: аналитич. = 0,4274; табул. = 0,4217

x=2,8: аналитич. = 0,3350; табул. = 0,3347

x=2,9: аналитич. = 0,2392; табул. = 0,2367

x=3,0: аналитич. = 0,1411; табул. = 0,1387

x=3,1: аналитич. = 0,0416; табул. = 0,0408

3. ---ТЕСТИРОВАНИЕ КОМБИНАЦИЙ ФУНКЦИЙ---

Проверка тождества sin^2 (x) + cos^2 (x) = 1:

sin^2 (0,0) + cos^2 (0,0) = 1,0000

sin^2 (0,1) + cos^2 (0,1) = 0,9753

sin^2 (0,2) + cos^2 (0,2) = 0,9705

sin^2 (0,3) + cos^2 (0,3) = 0,9854

sin^2 (0,4) + cos^2 (0,4) = 0,9850

sin^2 (0,5) + cos^2 (0,5) = 0,9704

sin^2 (0,6) + cos^2 (0,6) = 0,9756

sin^2 (0,7) + cos^2 (0,7) = 0,9994

sin^2 (0,8) + cos^2 (0,8) = 0,9751

sin^2 (0,9) + cos^2 (0,9) = 0,9706

sin^2 (1,0) + cos^2 (1,0) = 0,9859

sin^2 (1,1) + cos^2 (1,1) = 0,9845

sin^2 (1,2) + cos^2 (1,2) = 0,9703

sin^2 (1,3) + cos^2 (1,3) = 0,9759

sin^2 (1,4) + cos^2 (1,4) = 0,9987

sin^2 (1,5) + cos^2 (1,5) = 0,9748

sin^2 (1,6) + cos^2 (1,6) = 0,9707

sin^2 (1,7) + cos^2 (1,7) = 0,9864

sin^2 (1,8) + cos^2 (1,8) = 0,9841

sin^2 (1,9) + cos^2 (1,9) = 0,9702

sin^2 (2,0) + cos^2 (2,0) = 0,9762

sin^2 (2,1) + cos^2 (2,1) = 0,9981

sin^2 (2,2) + cos^2 (2,2) = 0,9745

sin^2 (2,3) + cos^2 (2,3) = 0,9708

sin^2 (2,4) + cos^2 (2,4) = 0,9869

sin^2 (2,5) + cos^2 (2,5) = 0,9836

sin^2 (2,6) + cos^2 (2,6) = 0,9702

sin^2 (2,7) + cos^2 (2,7) = 0,9765

sin^2 (2,8) + cos^2 (2,8) = 0,9975

sin^2 (2,9) + cos^2 (2,9) = 0,9743

sin^2 (3,0) + cos^2 (3,0) = 0,9709

sin^2 (3,1) + cos^2 (3,1) = 0,9873

4. Экспонента - текстовый файл:

Сравнение экспоненты:

x=0: исходная = 1,0000; из файла = 1,0000

x=1: исходная = 2,7183; из файла = 2,7183

x=2: исходная = 7,3891; из файла = 7,3891

x=3: исходная = 20,0855; из файла = 20,0855

x=4: исходная = 54,5982; из файла = 54,5982

x=5: исходная = 148,4132; из файла = 148,4132

x=6: исходная = 403,4288; из файла = 403,4288

x=7: исходная = 1096,6332; из файла = 1096,6332

x=8: исходная = 2980,9580; из файла = 2980,9580

x=9: исходная = 8103,0839; из файла = 8103,0839

x=10: исходная = 22026,4658; из файла = 22026,4658

5. Логарифм - бинарный файл:

Сравнение логарифма:

x=1: исходная = -0,1310; из файла = -0,1310

x=2: исходная = 0,6802; из файла = 0,6802

x=3: исходная = 1,0942; из файла = 1,0942

x=4: исходная = 1,3842; из файла = 1,3842

x=5: исходная = 1,6084; из файла = 1,6084

x=6: исходная = 1,7912; из файла = 1,7912

x=7: исходная = 1,9456; из файла = 1,9456

x=8: исходная = 2,0793; из файла = 2,0793

x=9: исходная = 2,1972; из файла = 2,1972

x=10: исходная = 2,3026; из файла = 2,3026

--- ТЕСТИРОВАНИЕ СЕРИАЛИЗАЦИИ ---

исходные функции (log(exp(x))):

Исходные функции:

x=0,0: array=0,0000, list=0,0000

x=1,0: array=1,0000, list=1,0000

x=2,0: array=2,0000, list=2,0000

x=3,0: array=3,0000, list=3,0000

x=4,0: array=4,0000, list=4,0000

x=5,0: array=5,0000, list=5,0000

x=6,0: array=6,0000, list=6,0000

x=7,0: array=7,0000, list=7,0000

x=8,0: array=8,0000, list=8,0000

x=9,0: array=9,0000, list=9,0000

x=10,0: array=10,0000, list=10,0000

--- ArrayTabulatedFunction (Serializable) ---

сравнение Array после Serializable:

сравнение исходной и восстановленной функции:

x= 0,0: исходная = 0,0000, полученная = 0,0000

x= 1,0: исходная = 1,0000, полученная = 1,0000

x= 2,0: исходная = 2,0000, полученная = 2,0000

x= 3,0: исходная = 3,0000, полученная = 3,0000

x= 4,0: исходная = 4,0000, полученная = 4,0000

x= 5,0: исходная = 5,0000, полученная = 5,0000

x= 6,0: исходная = 6,0000, полученная = 6,0000

x= 7,0: исходная = 7,0000, полученная = 7,0000

x= 8,0: исходная = 8,0000, полученная = 8,0000

x= 9,0: исходная = 9,0000, полученная = 9,0000

x=10,0: исходная = 10,0000, полученная = 10,0000

--- LinkedListTabulatedFunction (Externalizable) ---

сравнение List после Externalizable:

сравнение исходной и восстановленной функции:

x= 0,0: исходная = 0,0000, полученная = 0,0000

x= 1,0: исходная = 1,0000, полученная = 1,0000

x= 2,0: исходная = 2,0000, полученная = 2,0000

x= 3,0: исходная = 3,0000, полученная = 3,0000

x= 4,0: исходная = 4,0000, полученная = 4,0000

x= 5,0: исходная = 5,0000, полученная = 5,0000

x= 6,0: исходная = 6,0000, полученная = 6,0000

x= 7,0: исходная = 7,0000, полученная = 7,0000

x= 8,0: исходная = 8,0000, полученная = 8,0000

x= 9,0: исходная = 9,0000, полученная = 9,0000

x=10,0: исходная = 10,0000, полученная = 10,0000