Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 6

тема «Методы и циклы»

по дисциплине «Информатика»

Выполнил: студент группу ИСТ-22-1Б Пеяс В.С.

Проверил: Нетбай Георгий Владимирович

Пермь, 2022

**Содержание**

[Задание 1 3](#_Toc118317384)

[1.1. Постановка задачи 3](#_Toc118317385)

[1.2. Решение задачи, код программы 3](#_Toc118317386)

[1.3. Тестирование работы программы с проверкой 3](#_Toc118317387)

[Задание 2 5](#_Toc118317388)

[2.1. Постановка задачи 5](#_Toc118317389)

[2.2. Решение задачи, код программы 5](#_Toc118317390)

[2.3. Тестирование работы программы с проверкой 5](#_Toc118317391)

[Задание 3 7](#_Toc118317392)

[3.1. Постановка задачи 7](#_Toc118317393)

[3.2. Решение задачи, код программы 7](#_Toc118317394)

[3.3. Тестирование работы программы с проверкой 7](#_Toc118317395)

[Задание 4 8](#_Toc118317396)

[4.1. Постановка задачи 8](#_Toc118317397)

[4.2. Решение задачи, код программы 8](#_Toc118317398)

[4.3. Тестирование работы программы с проверкой 9](#_Toc118317399)

[Задание 5 9](#_Toc118317400)

[5.1. Постановка задачи 9](#_Toc118317401)

[5.2. Решение задачи, код программы 10](#_Toc118317402)

[5.3. Тестирование работы программы с проверкой 11](#_Toc118317403)

[Задание 6 13](#_Toc118317404)

[6.1. Постановка задачи 13](#_Toc118317405)

[6.2. Решение задачи, код программы 13](#_Toc118317406)

[6.3. Тестирование работы программы с проверкой 13](#_Toc118317407)

[Задание 7 15](#_Toc118317408)

[7.1. Постановка задачи 15](#_Toc118317409)

[7.2. Решение задачи, код программы 16](#_Toc118317410)

[7.3. Тестирование работы программы с проверкой 16](#_Toc118317411)

[Задание 8 17](#_Toc118317412)

[8.1. Постановка задачи 17](#_Toc118317413)

[8.2. Решение задачи, код программы 17](#_Toc118317414)

[8.3. Тестирование работы программы 18](#_Toc118317415)

[Задание 9 19](#_Toc118317416)

[9.1. Постановка задачи 19](#_Toc118317417)

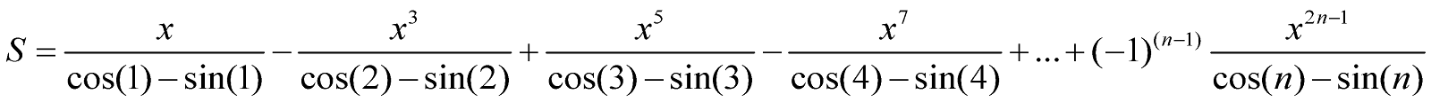
[9.2. Решение задачи, код программы 19](#_Toc118317418)

[9.3. Тестирование работы программы 20](#_Toc118317419)

# Задание 1

## 1.1. Постановка задачи

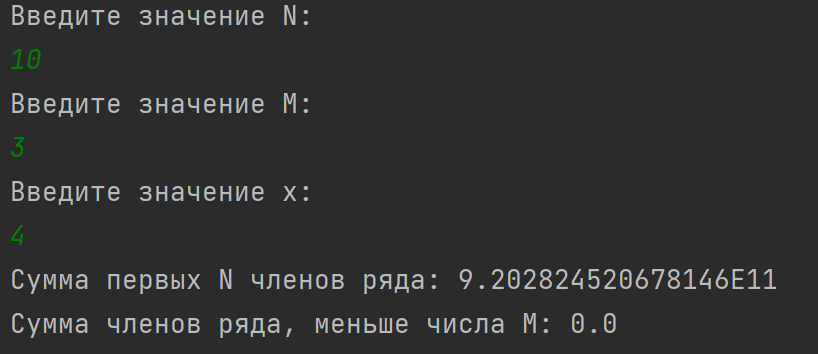
Найти сумму первых N членов ряда и найти сумму членов ряда, которые меньше заданного с клавиатуры числа M:



## 1.2. Решение задачи, код программы

import java.text.DecimalFormat;  
  
public class TrapezoidalRuleTask10 {  
 public static double function(double x) {  
 return Math.*exp*(2 - 5) + Math.*pow*(x, 5) - Math.*pow*(x, 3);  
 }  
  
 public static double calculateTrapezoidalArea(int n, double a, double b) {  
 double h = (b - a) / n;  
 double sum = 0;  
  
 for (int i = 1; i < n; i++) {  
 sum += *function*(a + i \* h);  
 }  
  
 return h \* ((*function*(a) + *function*(b)) / 2 + sum);  
 }  
  
 public static void main(String[] args) {  
 double a = 1;  
 double b = 2;  
  
 System.*out*.println("n\th\t\tS\t\tАналитическое решение\tПогрешность");  
 int[] ns = {10, 100, 1000, 10000};  
 double referenceArea = *calculateTrapezoidalArea*(10000, a, b);  
  
 for (int n : ns) {  
 double area = *calculateTrapezoidalArea*(n, a, b);  
 double error = Math.*abs*(area - referenceArea);  
 *printResult*(n, (b - a) / n, area, referenceArea, error);  
 }  
 }  
  
 private static void printResult(int n, double h, double area, double referenceArea, double error) {  
 DecimalFormat df = new DecimalFormat("0.000000");  
 System.*out*.println(n + "\t" + df.format(h) + "\t" + df.format(area) + "\t\t" + df.format(referenceArea) + "\t\t\t" + df.format(error));  
 }  
}

## 1.3. Тестирование работы программы с проверкой



# Задание 2

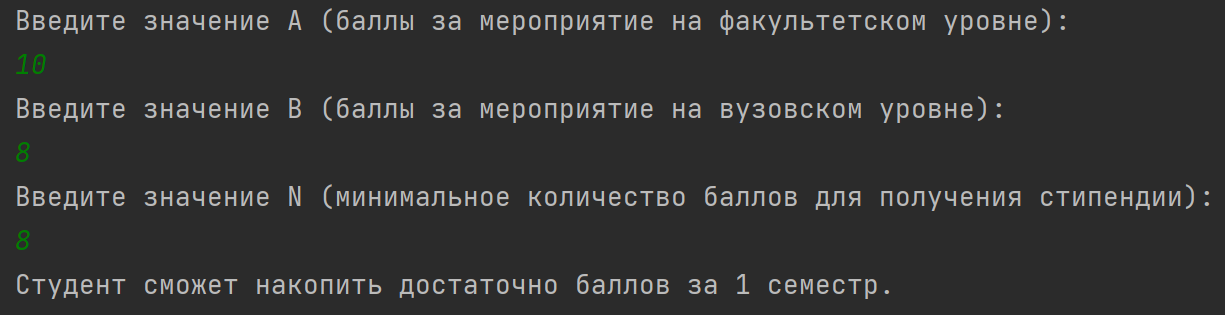
## 2.1. Постановка задачи

Студент имеет шанс получить повышенную стипендию за общественную деятельность. Известно, что раз в месяц он может участвовать в организации мероприятия на факультетском уровне, что дает А баллов, раз в два месяца организовывать мероприятие на вузовском уровне, которая дает B баллов. Минимальное количество баллов N, которое необходимо для получения стипендии. Найти сможет ли студент за 6 месяцев (1 семестр) накопить баллы для получения повышенной стипендии, если нет, то найти сколько баллов он накопит за год.

## 2.2. Решение задачи, код программы

import java.util.Scanner;  
  
public class ScholarshipCalculatorTask2 {  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.println("Введите значение A (баллы за мероприятие на факультетском уровне):");  
 int A = scanner.nextInt();  
  
 System.*out*.println("Введите значение B (баллы за мероприятие на вузовском уровне):");  
 int B = scanner.nextInt();  
  
 System.*out*.println("Введите значение N (минимальное количество баллов для получения стипендии):");  
 int N = scanner.nextInt();  
  
 int semesterMonths = 6;  
 int facultyEvents = semesterMonths;  
 int universityEvents = semesterMonths / 2;  
  
 int semesterPoints = A \* facultyEvents + B \* universityEvents;  
  
 if (semesterPoints >= N) {  
 System.*out*.println("Студент сможет накопить достаточно баллов за 1 семестр.");  
 } else {  
 int yearMonths = 12;  
 int yearFacultyEvents = yearMonths;  
 int yearUniversityEvents = yearMonths / 2;  
  
 int yearPoints = A \* yearFacultyEvents + B \* yearUniversityEvents;  
  
 System.*out*.println("Студент не сможет накопить достаточно баллов за 1 семестр.");  
 System.*out*.println("Количество баллов, которые студент накопит за год: " + yearPoints);  
 }  
 }  
}

## 2.3. Тестирование работы программы с проверкой



# Задание 3

## 3.1. Постановка задачи

Написать программы, которые вычисляют выражения:

, , 

## 3.2. Решение задачи, код программы

import java.util.Scanner;  
import static java.lang.Math.\*;

public static int num3\_1(){  
 int f = 0;  
 int sum = 0;  
 for (int i = 1; i <= 8; i++){  
 for (int j = 1; j <= i; j++){  
 sum += *pow*((j + i),2);  
 }  
 }  
 return f;  
}  
public static int num3\_2(){  
 int f =1;  
 int umnozh = 1;  
 for (int i = 1; i <=5; i++){  
 for (int j = 1; j <= i; j++){  
 umnozh \*= j;  
 }  
 }  
 return f;  
}  
public static double num3\_3(){  
 int i,j,k;  
 double f = 1;  
 double umnozh = 0;  
 double s = 0;  
 for (i = 1; i <= 8; i++){  
 for (j = 1; j <= 2\*i-1; j++){  
 for (k = i + j; k <= 2\*(i+j); k++){  
 s = 2\*j-3\*(i-0.5\*k);  
 }  
 umnozh += s;  
 }  
 f \*= umnozh;  
 }  
 f = 0;  
 return f;  
}

public static void main(String[] args) {

System.*out*.println("f1 = " + *num3\_1*());  
System.*out*.println("f2 = " + *num3\_2*());  
System.*out*.println("f3 = "+ *num3\_3*());

}

## 3.3. Тестирование работы программы с проверкой



# Задание 4

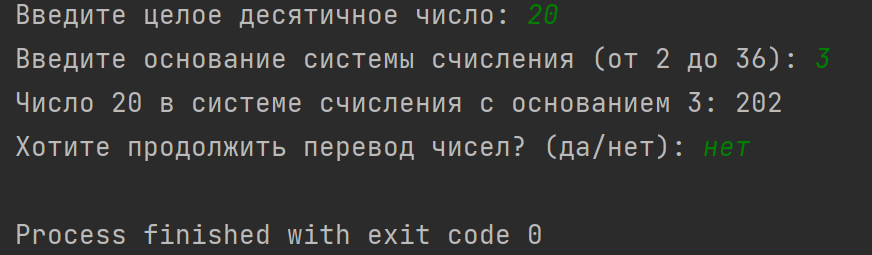
## 4.1. Постановка задачи

Пользователь вводит целое десятичное число. Написать программу перевода целого десятичного числа из десятичной системы счисления в систему счисления с основанием, которое пользователь вводит с клавиатуры (вводим ограничения на системы счисления, в которых есть буквенное обозначение цифр, так же не может быть введено 0 и 1 как основание системы счисления). После перевода сделать проверку. Пользователь может выбирать систему счисления до бесконечности, т.е. необходимо предусмотреть внешний цикл с вопросом к пользователю о необходимости продолжать перевод из одной системы счисления в другую.

## 4.2. Решение задачи, код программы

import java.util.Scanner;  
  
public class NumberConversioTask4 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 boolean continueConversion;  
  
 do {  
 System.*out*.print("Введите целое десятичное число: ");  
 int decimalNumber = scanner.nextInt();  
  
 int base;  
 do {  
 System.*out*.print("Введите основание системы счисления (от 2 до 36): ");  
 base = scanner.nextInt();  
 } while (base < 2 || base > 36);  
  
 String convertedNumber = Integer.*toString*(decimalNumber, base);  
 System.*out*.println("Число " + decimalNumber + " в системе счисления с основанием " + base + ": " + convertedNumber);  
  
 System.*out*.print("Хотите продолжить перевод чисел? (да/нет): ");  
 continueConversion = scanner.next().equalsIgnoreCase("да");  
 } while (continueConversion);  
  
 scanner.close();  
 }  
}

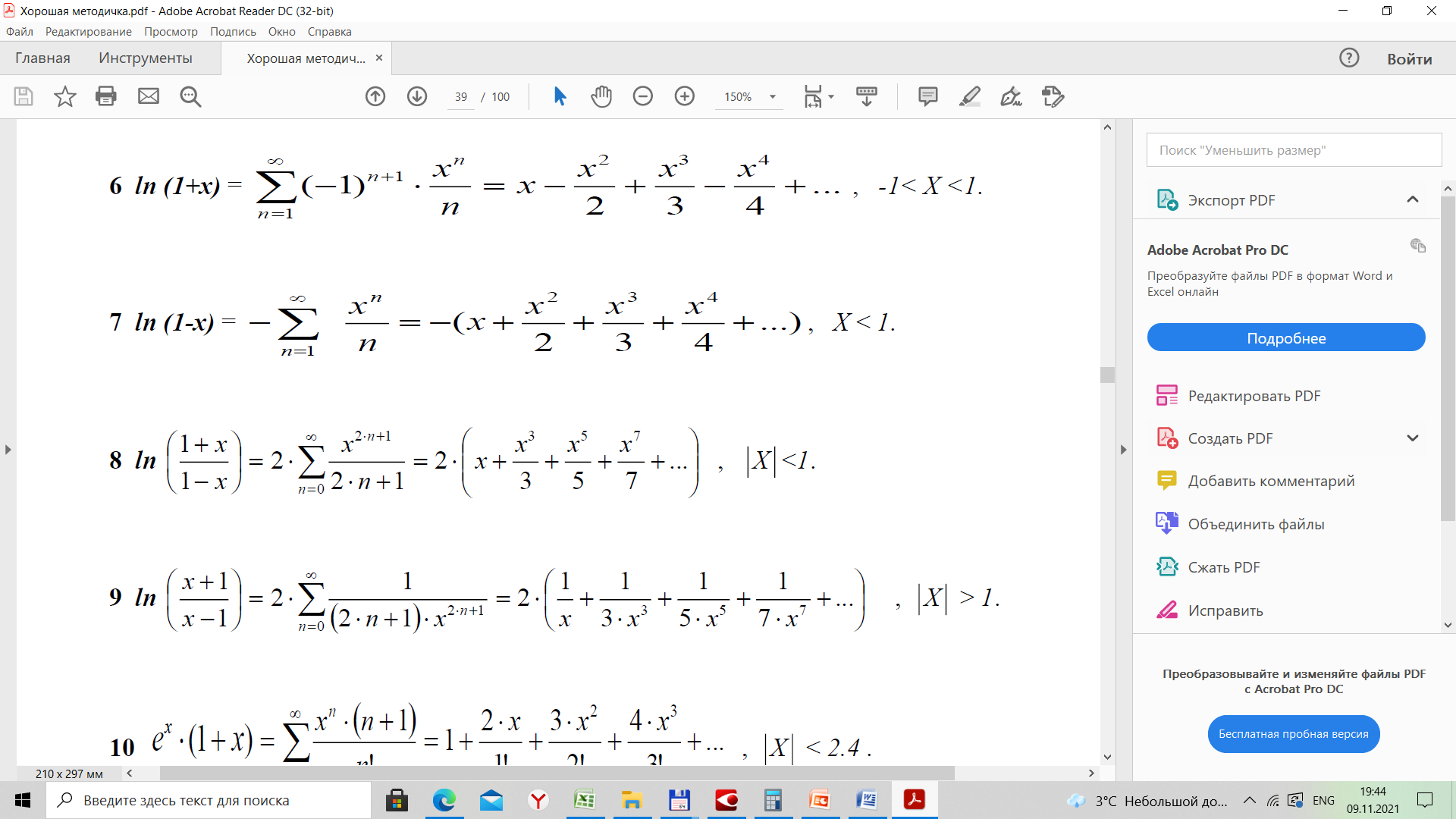
## 4.3. Тестирование работы программы с проверкой



# Задание 5

## 5.1. Постановка задачи

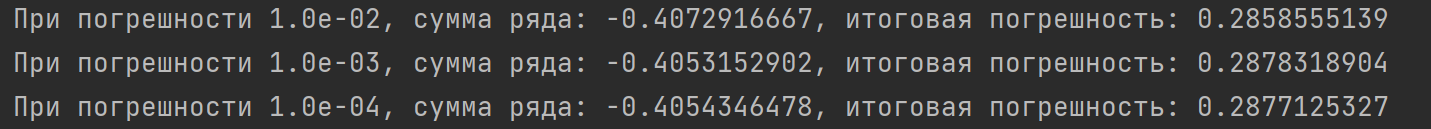
Организовать цикл для нахождения функции , через ряд Маклорена с погрешностью 10-2 – 10-4, остановка итерационной процедуры , где  – это погрешность. Вывести значение суммы ряда и итоговую погрешность для всех вариантов остановки итерационной процедуры.



## 5.2. Решение задачи, код программы

package org.example.var1;  
  
public class LogSeriesTask5 {  
 public static void main(String[] args) {  
 double x = 0.5; // Задайте значение x здесь  
  
 for (double eps = 1e-2; eps >= 1e-4; eps \*= 0.1) {  
 double sum = 0;  
 double term;  
 int n = 1;  
  
 do {  
 term = Math.*pow*(-x, n) / n;  
 sum += term;  
 n++;  
 } while (Math.*abs*(term) >= eps);  
  
 double actualValue = Math.*log*(1 - x);  
 double error = Math.*abs*(actualValue - sum);  
  
 System.*out*.printf("При погрешности %.1e, сумма ряда: %.10f, итоговая погрешность: %.10f%n", eps, sum, error);  
 }  
 }  
}

## 5.3. Тестирование работы программы



# Задание 7

## 7.1. Постановка задачи

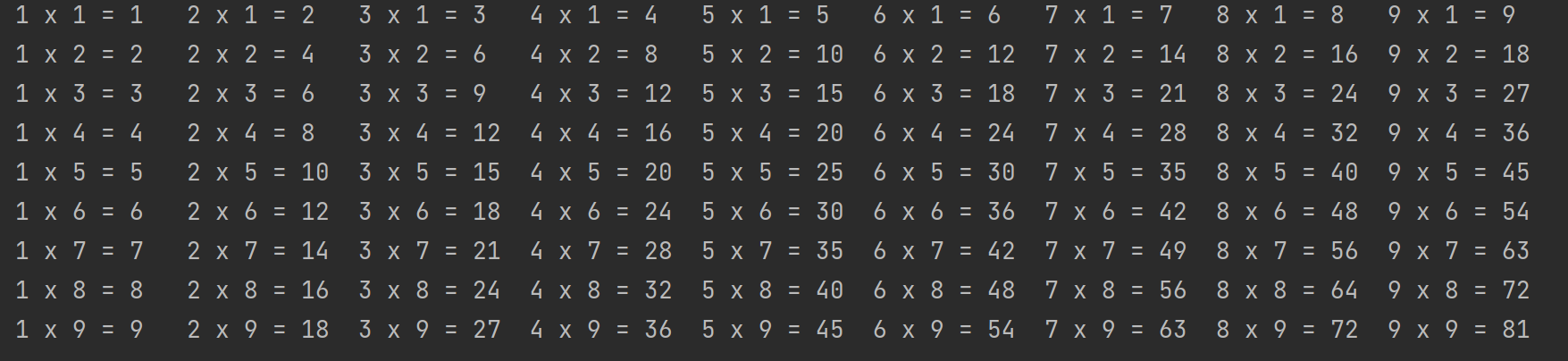
. Напечатать полную таблицу умножения в виде:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 х 1 = 1 | 2 x 1 = 2 | ... | 9 x 1 = 9 |
| 1 х 2 = 2 | 2 x 2 = 4 | ... | 9 x 2 = 18 |
| ... | ... | ... | ... |
| 1 х 9 = 9 | 2 х 9 = 18 | ... | 9 х 9 = 81 |

## 7.2. Решение задачи, код программы

public class MultiplicationTableTask7 {  
 public static void main(String[] args) {  
 for (int i = 1; i <= 9; i++) {  
 for (int j = 1; j <= 9; j++) {  
 System.*out*.printf("%d x %d = %d", j, i, j \* i);  
  
 if (j < 9) {  
 System.*out*.print("\t");  
 }  
 }  
 System.*out*.println();  
 }  
 }  
}

## 7.3. Тестирование работы программы



# Задание 8

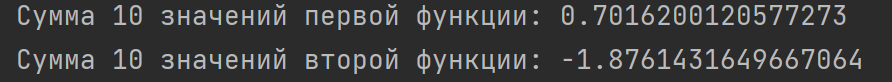
## 8.1. Постановка задачи

Переделать программу (класс) задания 1 лабораторной работы 4 о нахождении значений 2-х функций в класс без метода main с названием FunctionMy. Метод main заменить на метод FunctionMy c входными данными. Создать программу, взаимодействующую с классом FunctionMy, в которой пользователь в цикле находит сумму 10 значений функции изменяя только один параметр функции в цикле, остальные параметры, которые входя в формулу, считаются константами. Взаимодействие с классом FunctionMy сделать в виде наследования.

## 8.2. Решение задачи, код программы

public class FunctionMyTask8 {  
 public double functionOne(double x, double y) {  
 return Math.*sin*(x) \* Math.*cos*(y) - Math.*cos*(x) \* Math.*sin*(y);  
 }  
  
 public double functionTwo(double x, double y) {  
 return Math.*sin*(x) \* Math.*cos*(y) + Math.*cos*(x) \* Math.*sin*(y);  
 }  
}  
  
class FunctionMyInteraction extends FunctionMyTask8 {  
 public static void main(String[] args) {  
 FunctionMyInteraction interaction = new FunctionMyInteraction();  
 double x = 1.0;  
 double y = 2.0;  
 double sumFunctionOne = 0.0;  
 double sumFunctionTwo = 0.0;  
  
 for (int i = 0; i < 10; i++) {  
 sumFunctionOne += interaction.functionOne(x + i, y);  
 sumFunctionTwo += interaction.functionTwo(x + i, y);  
 }  
  
 System.*out*.println("Сумма 10 значений первой функции: " + sumFunctionOne);  
 System.*out*.println("Сумма 10 значений второй функции: " + sumFunctionTwo);  
 }  
}

## 8.3. Тестирование работы программы с проверкой



# Задание 9

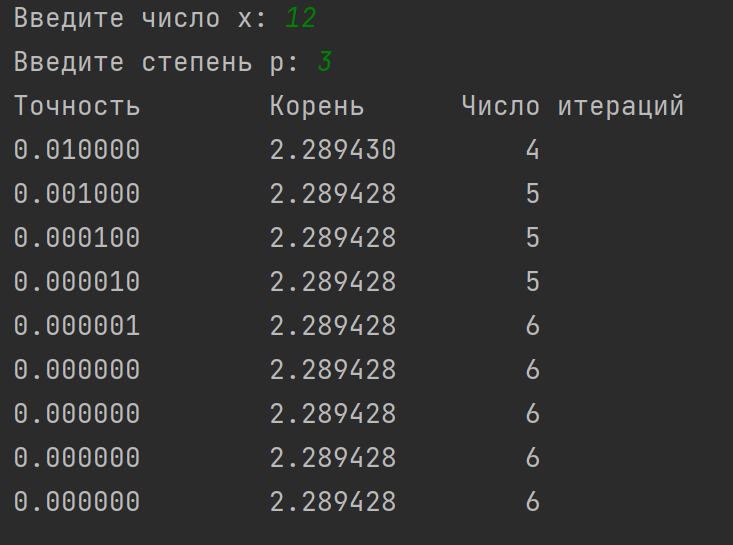
## 9.1. Постановка задачи

Написать программу вычисление корня р-й степени (степень вводиться с клавиатуры) в рамках итерационной процедуры . Для определения используется итерационная процедура на основе формулы Ньютона , , при этом . Остановка итерационной процедуры , где  – точность вычисления. В рамках программы определить число итраций, которые потребовались для отыскания корня р-й степени в рамках цикла с параметром для точности от 10-2 до 10-6, шаг 10-1. Организовать форматированный вывод результатов в виде: Точность Корень Число итераций.

## 9.2. Решение задачи, код программы

public class RootNewtonMethodTask9 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите число x: ");  
 double x = scanner.nextDouble();  
  
 System.*out*.print("Введите степень p: ");  
 int p = scanner.nextInt();  
  
 System.*out*.println("Точность\t\tКорень\t\tЧисло итераций");  
  
 for (int k = 2; k <= 10; k++) {  
 double eps = Math.*pow*(10, -k);  
 double root = *rootOfDegreeP*(x, p, eps);  
 int iterations = *countIterations*(x, p, eps);  
 *printResult*(eps, root, iterations);  
 }  
 }  
  
 private static double rootOfDegreeP(double x, int p, double eps) {  
 double y0 = Math.*exp*(Math.*log*(x \* (p + 1)) / p);  
 double y1;  
  
 do {  
 y1 = (1.0 / p) \* (((p - 1) \* y0) + (x / Math.*pow*(y0, p - 1)));  
 double temp = y0;  
 y0 = y1;  
 y1 = temp;  
 } while (Math.*abs*(y1 - y0) > eps);  
  
 return y0;  
 }  
  
 private static int countIterations(double x, int p, double eps) {  
 double y0 = Math.*exp*(Math.*log*(x \* (p + 1)) / p);  
 double y1;  
 int iterations = 0;  
  
 do {  
 y1 = (1.0 / p) \* (((p - 1) \* y0) + (x / Math.*pow*(y0, p - 1)));  
 double temp = y0;  
 y0 = y1;  
 y1 = temp;  
 iterations++;  
 } while (Math.*abs*(y1 - y0) > eps);  
  
 return iterations;  
 }  
  
 private static void printResult(double eps, double root, int iterations) {  
 DecimalFormat df = new DecimalFormat("0.000000");  
 System.*out*.println(df.format(eps) + "\t\t" + df.format(root) + "\t\t" + iterations);  
 }  
}

## 9.3. Тестирование работы программы



# Задание 10

## 10.1. Постановка задачи

Разработать алгоритм приближённого вычисления площади криволинейной фигуры, ограниченной осью абсцисс, графиком заданной функции , и вертикальными прямыми, т.е. . Каждый отрезок функции представляется в виде трапеции c длиной отрезка (шагом)  (см. рис. ниже). Затем площадь под кривой вычисляется по формуле . Вычислить значения площади под кривой при n равном 10, 100, 1000, 10000 в рамках цикла по n. Оценить погрешность решения при разных шагах по сравнению с точным аналитическим решением. Организовать форматированный вывод результатов в виде:

n h S Аналитическое решение Погрешность.

|  |  |
| --- | --- |
|  |  |
| а | б |

Рис. Разбиение площади под кривой на трапеции с шагом h:

а – общий вид; б – i-я трапеция

## 10.2. Решение задачи, код программы

import java.text.DecimalFormat;  
  
public class TrapezoidalRuleTask10 {  
 public static double function(double x) {  
 return Math.*exp*(2 - 5) + Math.*pow*(x, 5) - Math.*pow*(x, 3);  
 }  
  
 public static double calculateTrapezoidalArea(int n, double a, double b) {  
 double h = (b - a) / n;  
 double sum = 0;  
  
 for (int i = 1; i < n; i++) {  
 sum += *function*(a + i \* h);  
 }  
  
 return h \* ((*function*(a) + *function*(b)) / 2 + sum);  
 }  
  
 public static void main(String[] args) {  
 double a = 1;  
 double b = 2;  
 System.*out*.println("n\th\t\tS\t\tАналитическое решение\tПогрешность");  
 int[] ns = {10, 100, 1000, 10000};  
 double referenceArea = *calculateTrapezoidalArea*(10000, a, b);  
  
 for (int n : ns) {  
 double area = *calculateTrapezoidalArea*(n, a, b);  
 double error = Math.*abs*(area - referenceArea);  
 *printResult*(n, (b - a) / n, area, referenceArea, error);  
 }  
 }  
  
 private static void printResult(int n, double h, double area, double referenceArea, double error) {  
 DecimalFormat df = new DecimalFormat("0.000000");  
 System.*out*.println(n + "\t" + df.format(h) + "\t" + df.format(area) + "\t\t" + df.format(referenceArea) + "\t\t\t" + df.format(error));  
 }  
}

## 10.3. Тестирование работы программы

