Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчёт по лабораторной работе № 5

тема «Циклы в Java»

по дисциплине «Информатика»

Выполнил: студент группу ИСТ-22-1б Радостев К.П.

Проверил: Нетбай Георгий Владимирович

Пермь, 2022

**Содержание**

[Задание 1 3](#_Toc123318235)

[1.1. Постановка задачи 3](#_Toc123318236)

[1.2. Решение задачи, код программы 3](#_Toc123318237)

[1.3. Тестирование кода программы 3](#_Toc123318238)

[Задание 2 4](#_Toc123318239)

[2.1. Постановка задачи 4](#_Toc123318240)

[2.2. Решение задачи, код программы 4](#_Toc123318241)

[2.3. Тестирование кода программы 4](#_Toc123318242)

[Задание 3 5](#_Toc123318243)

[3.1. Постановка задачи 5](#_Toc123318244)

[3.2. Решение задачи, код программы 5](#_Toc123318245)

[3.3. Тестирование кода программы 5](#_Toc123318246)

[Задание 4 6](#_Toc123318247)

[4.1. Постановка задачи 6](#_Toc123318248)

[4.2. Решение задачи, код программы 6](#_Toc123318249)

[4.3. Тестирование кода программы 6](#_Toc123318250)

[Задание 6 7](#_Toc123318251)

[6.1. Постановка задачи 7](#_Toc123318252)

[6.2. Решение задачи, код программы 7](#_Toc123318253)

[6.3. Тестирование кода программы 8](#_Toc123318254)

[Задание 7 9](#_Toc123318255)

[7.1. Постановка задачи 9](#_Toc123318256)

[7.2. Решение задачи, код программы 9](#_Toc123318257)

[7.3. Тестирование кода программы 9](#_Toc123318258)

[Задание 8 10](#_Toc123318259)

[8.1. Постановка задачи 10](#_Toc123318260)

[8.2. Решение задачи, код программы 10](#_Toc123318261)

[8.3. Тестирование кода программы 10](#_Toc123318262)

[Задание 9 11](#_Toc123318263)

[9.1. Постановка задачи 11](#_Toc123318264)

[9.2. Решение задачи, код программы 11](#_Toc123318265)

[9.3. Тестирование кода программы 12](#_Toc123318266)

[Задание 10 13](#_Toc123318267)

[10.1. Постановка задачи 13](#_Toc123318268)

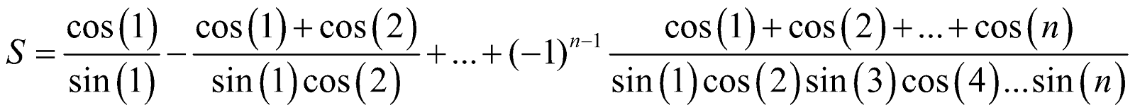
[10.2. Решение задачи, код программы 13](#_Toc123318269)

[10.3. Тестирование кода программы 14](#_Toc123318270)

# Задание 1

## 1.1. Постановка задачи

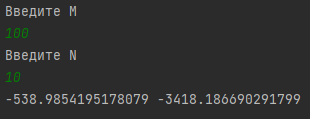
1. Найти сумму первых N членов ряда и найти сумму членов ряда, которые меньше заданного с клавиатуры числа M:

**

## 1.2. Решение задачи, код программы

import java.util.\*;  
import static java.lang.Math.\*;  
public class Zadanie\_1 {  
 public static double chisl(int n) {  
 double r=0;  
 for (int i=1; i<=n; i++) {  
 r+=*cos*(i);  
 }  
 return r;  
 }  
 public static double znam(int n) {  
 double r=1;  
 for (int i=1; i<=n; i++) {  
 if (i%2==0) {  
 r\*=*cos*(i);  
 }  
 else {  
 r\*=*sin*(i);  
 }  
 }  
 return r;  
 }  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 double s = 0, s2=0;  
 int n=1;  
 System.*out*.println("Введите M");  
 double M = in.nextDouble();  
 System.*out*.println("Введите N");  
 double N = in.nextDouble();  
 while (*pow*(-1,n-1)\*(*chisl*(n)/*znam*(n))<M) {  
 s+=*pow*(-1,n-1)\*(*chisl*(n)/*znam*(n));  
 n++;  
 }  
 for (n=1; n<N; n++) {  
 s2+=*pow*(-1,n-1)\*(*chisl*(n)/*znam*(n));  
 }  
 System.*out*.println(s2+" "+s);  
 }  
}

## 1.3. Тестирование кода программы



# Задание 2

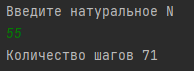
## 2.1. Постановка задачи

2. Возьмем любое натуральное число N. Если оно четное - разделим его пополам, если нечетное - умножим на 3, прибавим 1 и разделим пополам. Повторим эти действия с вновь полученным числом. Гипотеза гласит, что независимо от выбора первого числа рано или поздно мы получим 1. Вывести количество шагов для выполнения данной операции.

## 2.2. Решение задачи, код программы

import java.util.\*;  
  
public class Zadanie\_2 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 int s = 0;  
 System.*out*.println("Введите натуральное N");  
 int n = in.nextInt();  
 while (n!=1) {  
 if (n%2==0) {  
 n/=2;  
 } else {  
 n=(n\*3+1)/2;  
 }  
 s+=1;  
 }  
 System.*out*.println("Количество шагов "+s);  
 }  
}

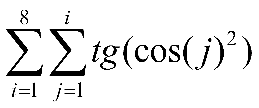
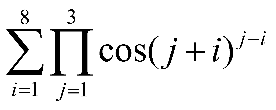
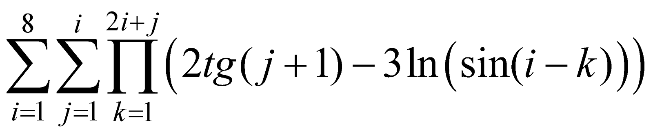
## 2.3. Тестирование кода программы



# Задание 3

## 3.1. Постановка задачи

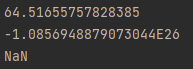
3. Написать  программы, которые вычисляют выражения:

, , 

## 3.2. Решение задачи, код программы

import static java.lang.Math.\*;  
public class Zadanie\_3 {  
 public static void main(String[] args) {  
 double a=0,b=0,c=0;  
 for (int i=1; i<=8; i++) {  
 for (int j=1; j<=i; j++) {  
 a+=*tan*(*pow*(*cos*(j),2));  
 }  
 b+=a;  
 }  
 System.*out*.println(b);  
 a=1;  
 b=0;  
 for (int i=1; i<=8; i++) {  
 for (int j=1; j<=3; j++) {  
 a\*=*pow*(*cos*(j+i),j-i);  
 }  
 b+=a;  
 }  
 System.*out*.println(b);  
 a=1;  
 b=0;  
 for (int i=1; i<=8; i++) {  
 for (int j=i; j<=i; j++) {  
 for (int k=1; k<=2\*i+j; k++) {  
 a\*=(2\**tan*(j+1)-3\**log*(*sin*(i-k)));  
 }  
 b+=a;  
 }  
 c+=b;  
 }  
 System.*out*.println(c);  
 }  
}

## 3.3. Тестирование кода программы

****

# Задание 4

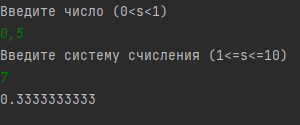
## 4.1. Постановка задачи

4. Пользователь вводит десятичную дробь (например, 0,2345 – у дроби нет целой части, если пользователь введет число, где есть целая часть, то должно выплыть сообщение об ошибке). Написать программу перевода десятичной дроби 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, определить погрешность, если она есть. 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.\*;  
  
public class Zadanie\_4 {  
 public static double perevod(double x,int s) {  
 String x2 = "0.";  
 for (int a=0; a<10; a++) {  
 x=x\*s-(int)x\*s;  
 x2+=Integer.*toString*((int)x);  
 }  
 return Double.*parseDouble*(new StringBuilder(x2).toString());  
 };  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 double a=0;  
 while (a>=1 | a<=0) {  
 System.*out*.println("Введите число (0<s<1)");  
 a=in.nextDouble();  
 }  
 int s=0;  
 while (s<1 | s>10) {  
 System.*out*.println("Введите систему счисления (1<=s<=10)");  
 s=in.nextInt();  
 }  
 System.*out*.println(*perevod*(a,s));  
 }  
}

## 4.3. Тестирование кода программы



# Задание 6

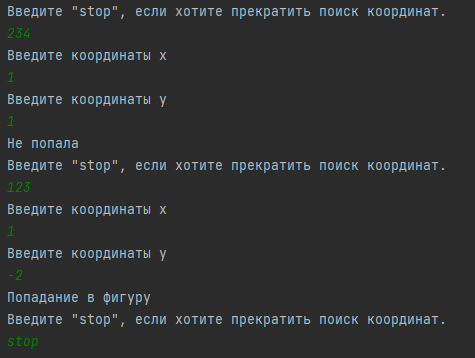
## 6.1. Постановка задачи

6. Переделать программу (класс) задания 6 лабораторной работы 5 о попадании точки в область в класс без метода main с названием Oblast. Метод main заменить на метод Oblast c входными данными в виде координат произвольной точки пространства и выходными данными типа boolean (true – если точка попала в область, false – если точка не попала в область). Создать программу, взаимодействующую с классом Oblast (без использования наследования), в которой пользователь в цикле проверяет попадание точек в область до бесконечности, т.е. необходимо предусмотреть цикл с вопросом к пользователю о необходимости проверки точки.

## 6.2. Решение задачи, код программы

import java.util.Scanner;  
import static java.lang.Math.\*;  
  
public class Zadanie\_6 {  
 public static boolean okr(double x, double y) {  
 if (36 >= x\*x + y\*y & y <= 0) {  
 return true;  
 }  
 return false;  
 }  
 public static boolean notDirHit(double x, double y) {  
 if (4 > *pow*(x-3,2) + *pow*(y+2,2) || (y > 5\*x - 5  
 & y > -5/2\*(x+5) & y > -5 & y < 0)) {  
 return true;  
 }  
 return false;  
 }  
 public static boolean DirHit(double x, double y) {  
 if (1 >= *pow*(x-3,2) + *pow*(y+2,2) ||  
 1 >= *pow*(x+1,2) + *pow*(y+3,2) ||  
 1 >= *pow*(x+3,2) + *pow*(y+1,2)) {  
 return true;  
 }  
 return false;  
 }  
 public static void oblast(double x, double y) {  
 if (*okr*(x,y)) {  
 if (*notDirHit*(x,y)) {  
 if (*DirHit*(x,y)) {  
 System.*out*.println("Попадание в фигуру");  
 } else {  
 System.*out*.println("Не попала");  
 }  
 } else {  
 System.*out*.println("Попадание в фигуру");  
 }  
 } else {  
 System.*out*.println("Не попала");  
 }  
 }  
 public static void main(String[] args) {  
 double x=0, y=0;  
 Scanner in = new Scanner(System.*in*);  
 boolean stop=false;  
 while (!stop) {  
 System.*out*.println("Введите \"stop\", если хотите прекратить поиск координат.");  
 String vvod = in.nextLine();  
 if (vvod.equals("stop")) {  
 stop = true;  
 } else {  
 System.*out*.println("Введите координаты х");  
 x = Double.*parseDouble*(in.nextLine());  
 System.*out*.println("Введите координаты y");  
 y = Double.*parseDouble*(in.nextLine());  
 *oblast*(x,y);  
 }  
 }  
 }  
}

## 6.3. Тестирование кода программы



# Задание 7

## 7.1. Постановка задачи

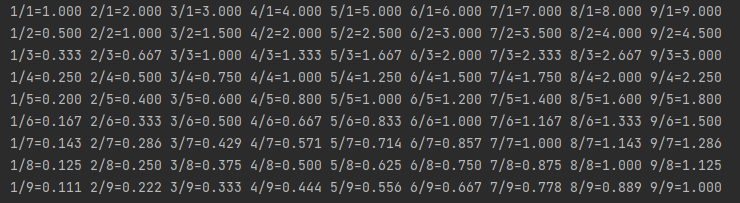
7. Напечатать полную таблицу деления (не целочисленного) в виде:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 / 1 = 1 | 2 / 1 = 2 | ... | 9 / 1 = 9 |
| 1 / 2 = 0,5 | 2 / 2 = 1 | ... | 9 / 2 = 4,5 |
| ... | ... | ... | ... |
| 1 / 9 = 0,1111 | 2 / 9 = 0,2222 | ... | 9 / 9 = 1 |

## 7.2. Решение задачи, код программы

public class Zadanie\_7 {  
 public static void main(String[] args) {  
 for (int i = 1; i < 10; i++) {  
 for (int j = 1; j < 10; j++) {  
 System.*out*.print(j+"/"+i+"="+String.*format*("%.3f",((double)j/i)).replace(',', '.')+" ");  
 }  
 System.*out*.println("");  
 }  
 }  
}

## 7.3. Тестирование кода программы



# Задание 8

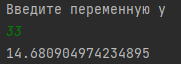
## 8.1. Постановка задачи

8. 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,

## 8.2. Решение задачи, код программы

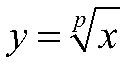
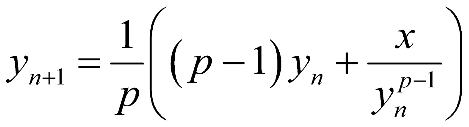
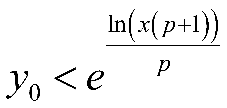
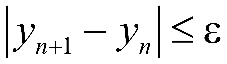
import java.util.Scanner;  
  
public class Zadanie\_8 {  
 public static void main(String[] var0) {  
 Scanner var1 = new Scanner(System.*in*);  
 System.*out*.println("Введите переменную y");  
 double y = var1.nextDouble();  
 System.*out*.print(*fm1*(y)+*fm2*(y));  
 }  
 public static double fm1(double var2) {  
 double var4 = 8;  
 double var6 = (Math.*pow*(Math.*sin*(2.0 \* var2 + 1.0), 2.0) + 0.3 \* var2) / (Math.*log10*(var2 \* var4 + 0.8 \* var4) - Math.*pow*(Math.*E*, 2.0));  
 return var6;  
 }  
 public static double fm2(double var2) {  
 double var1 = 3;  
 double var5 = 5;  
 double var3 = 7;  
 double var7 = (var1 \* var5 + var3 \* var5 / (Math.*pow*(var1, 2.0) + 4.0 \* var1 + Math.*E*)) \* ((2.0 \* var1 + var3) / Math.*pow*(Math.*E*, 2.0 \* Math.*pow*(var5, 2.0)) + Math.*tan*(Math.*cos*(Math.*sin*(var1 + var5 + var3))));  
 return var7;  
 }  
}

## 8.3. Тестирование кода программы



# Задание 9

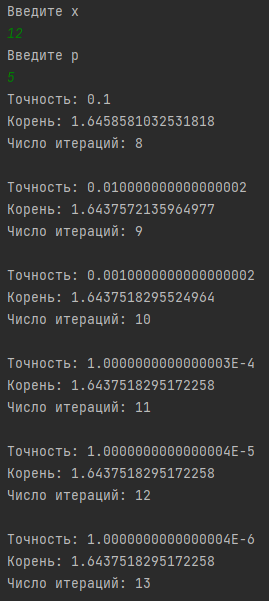
## 9.1. Постановка задачи

9. 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

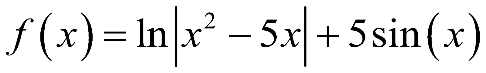
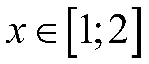
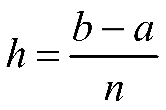
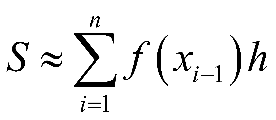
import java.util.\*;  
import static java.lang.Math.\*;  
  
public class Zadanie\_9 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите x");  
 double x = (in.nextDouble());  
 System.*out*.println("Введите p");  
 double p = (in.nextDouble());  
 double y = *log*(x\*(p+1))/p;  
 int c=0;  
 for (double e=*pow*(10,-1); e>*pow*(10,-6); e\*=0.1) {  
 while (true) {  
 double ly=y;  
 y=1/p\*((p-1)\*y+x/*pow*(y,(p-1)));  
 c++;  
 if (*abs*(y-ly)<e) {  
 System.*out*.println("Точность: "+e+"\nКорень: "+y  
 +"\nЧисло итераций: "+c+"\n");  
 break;  
 }  
 }  
 }  
 }  
}

## 9.3. Тестирование кода программы



# Задание 10

## 10.1. Постановка задачи

10. Разработать алгоритм приближённого вычисления площади криволинейной фигуры, ограниченной осью абсцисс, графиком заданной функции , и вертикальными прямыми, т.е. . Каждый отрезок функции представляется в виде прямоугольника c длиной отрезка (шагом)  (см. рис. ниже). Затем площадь под кривой вычисляется по формуле  – левый прямоугольник, т.к. высота прямоугольника берется как значение функции в крайней левой координате каждого прямоугольника. 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 прямоугольники с шагом h:

а – общий вид; б – i-й левый прямоугольник

## 10.2. Решение задачи, код программы

import static java.lang.Math.\*;  
  
public class Zadanie\_10 {  
 public static void main(String[] args) {  
 double sum;  
 double h;  
 for(double n = 10;n <= 10000; n \*= 10){  
 h = 1/n;  
 sum = 0;  
 for (int i = 0; i < n; i++){  
 sum += *log*(*abs*(*pow*(n-h\*i,2)-5\*(n-h\*i))+5\**sin*(n-h\*i)) \* h;  
 }  
 System.*out*.println(n + "\n" + h + "\n" + sum + "\n5.18\n" + *abs*(sum - 5.18)+ "\n");  
 }  
 }  
}

## 10.3. Тестирование кода программы

