 Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра автоматики и процессов управления

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «СМиСПИС»

Вариант №2

|  |  |  |
| --- | --- | --- |
| Студент гр. 5371 |  | Нургазы Б. |
| Студентка гр. 5371 |  | Есенбаев Ч. |
| Студент гр. 5371 |  | Ильясов Е. |
| Преподаватель |  | Кораблев Ю.А. |

Санкт-Петербург

2020

1. **Цель работы**

Изучить основы разработки *Java-*программ. Изучить основные типы *Java,* создание и работу с массивами.

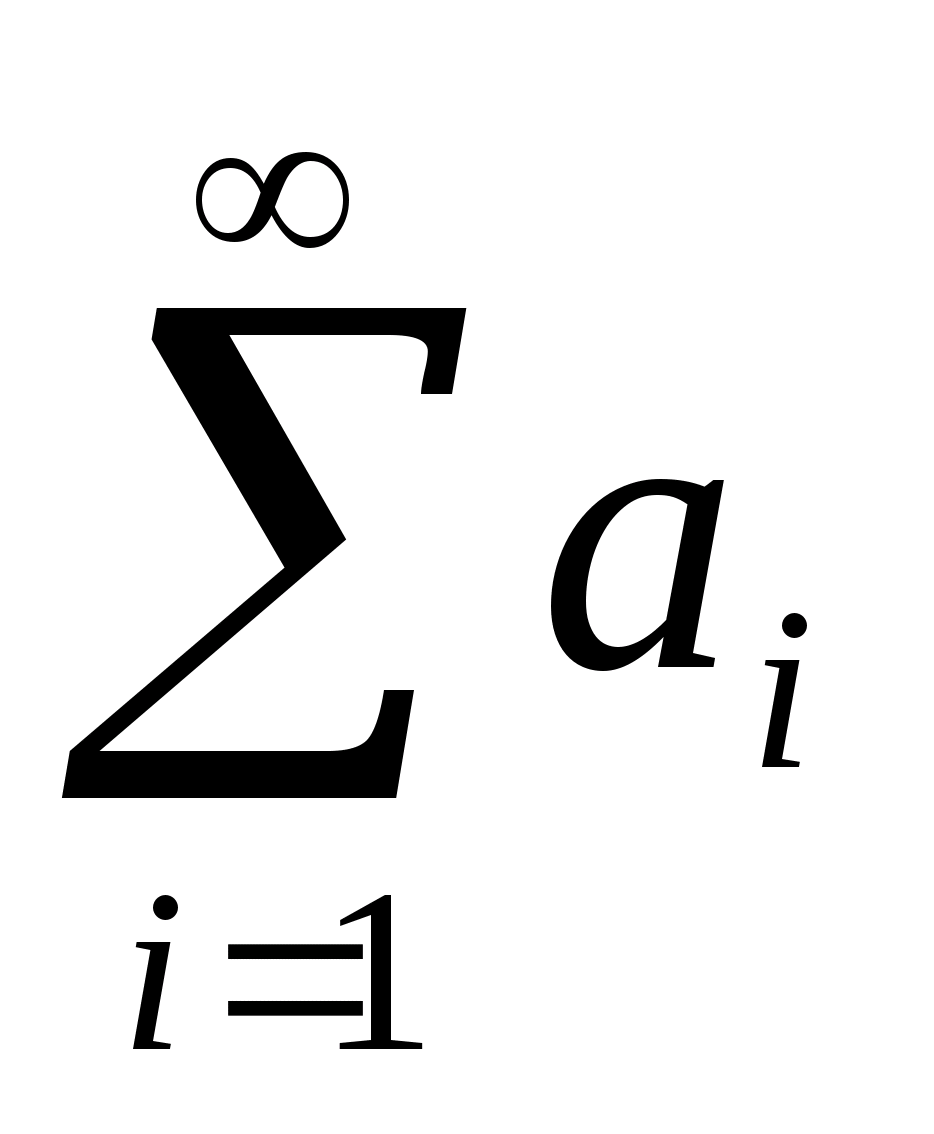
1. **Задание на лабораторную работу №1. Вариант №2.**

Написать программу, которая осуществляет расчет значения функции, используя рекуррентную формулу. Для расчета значений ряда предусмотреть возможность ввода значения x и точности вычисления.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Функция | Ряд | Реккурентная формула |
| 3 | sin(x) |  |  |

1. **Разработка математического и алгоритмического обеспечения**

***Числовым рядом*** называется бесконечная сумма *S* некоторой последовательности.

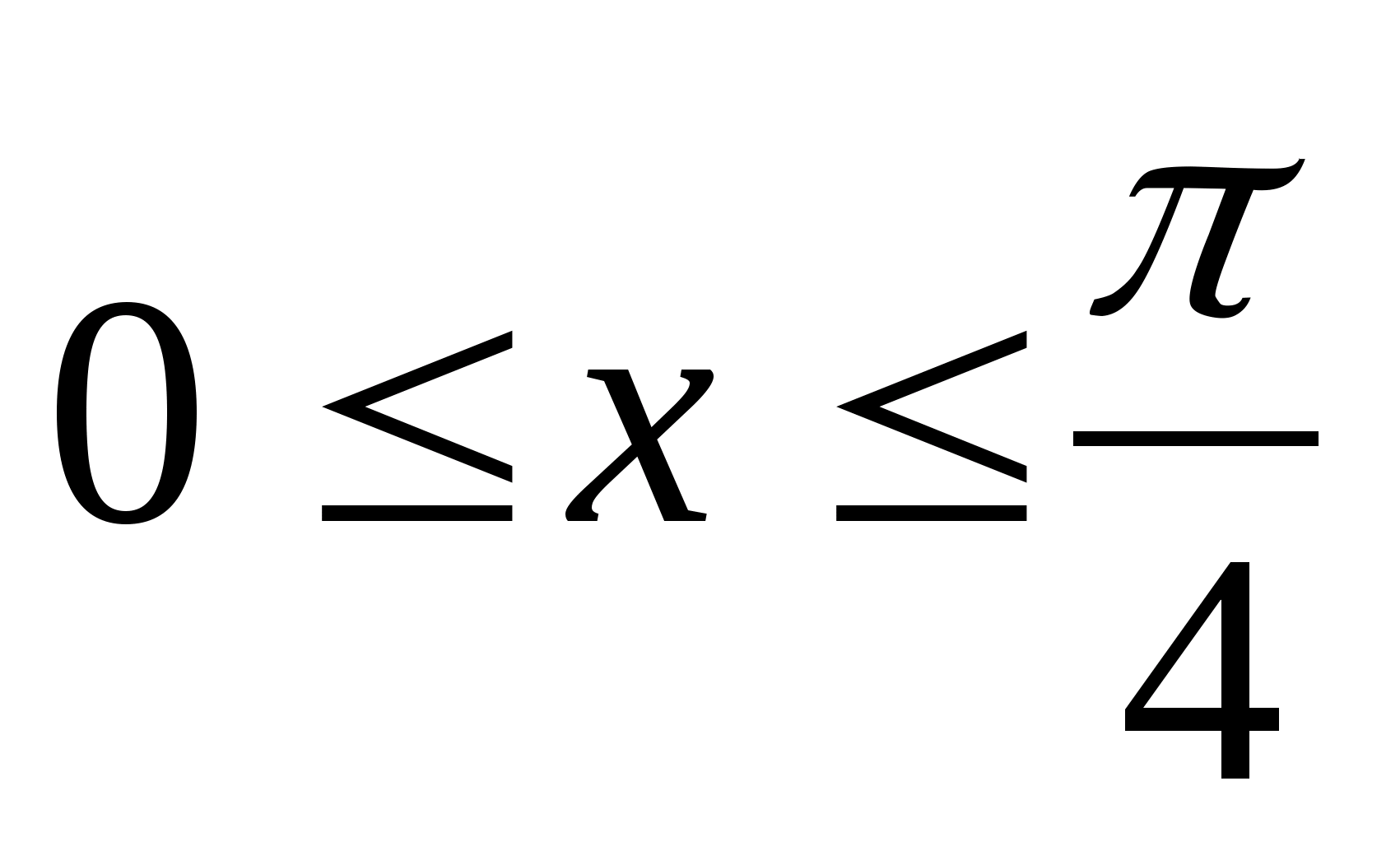
*S* = *a*1+*a*2+*a*3+…+*ai*+… = 

С помощью разложения в числовой ряд можно вычислить многие из элементарных функций, например

A picture containing object, clock

Description automatically generated,

A picture containing object, clock

Description automatically generated при ;

A picture containing object, clock

Description automatically generated при A close up of a sign

Description automatically generated.

Поскольку ряд имеет бесконечное число членов, вычисления производят с определенной точностью, т.е. суммирование прекращают, когда очередной член ряда оказывается меньше *допустимой погрешности вычислений*. Допустимую погрешность вычислений называют иначе *точностью вычислений*. Она задается малым числом ε, где 0<ε<1, (ε=10-2, 103,…), чем меньше ε, тем точнее решение. При ε=10i решение будет точным до *i*-го знака после запятой.

Основная проблема при вычислении суммы числового ряда состоит в вычислении очередного члена последовательности. Можно выделить три подхода к решению этой проблемы:

1) использование формулы общего члена ряда;

2) использование рекуррентного соотношения;

3) смешанный подход, основанный на двух предыдущих.

### Использование рекуррентного соотношения

Понятие ***рекуррентной последовательности*** в курсе математики вводится так: пусть известно *k* чисел: *а1, …, ak*. Эти числа являются началом числовой последовательности. Следующие элементы этой последовательности вычисляются так:

*ak+1 =  F(a1, …, ak); ak+2 = F(a2, …, ak+1); ak+3 = F(a3, …, ak+2) …*

Здесь *F(…)* – функция от *k* аргументов. Формула вида

*ai =  F(ai-k, …, ai-1)*

называется ***рекуррентной формулой***. Другими словами, *рекуррентная последовательность* – это бесконечный ряд чисел, каждое из которых, за исключением *k* начальных, вычисляется через предыдущие члены ряда.

Например, арифметическая и геометрическая прогрессии:

*a*1=1, *a*2=3, *a*3=5,… Рекуррентная формула: A picture containing object

Description automatically generated.

*a*1=1, *a*2=2, *a*3=4, … Рекуррентная формула: A picture containing object

Description automatically generated.

Глубина рекурсии в обоих случаях равна 1 (такую зависимость называют одношаговой рекурсией).

Числа Фибоначчи*:* 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, … Начиная с третьего элемента, каждое число равно сумме двух предыдущих, т.е. это рекуррентная последовательность с глубиной, равной 2. Рекуррентная формула для чисел последовательности Фибоначчи:

A picture containing object

Description automatically generated.

Для вывода рекуррентной формулы часто можно воспользоваться соотношением вида:

A picture containing object

Description automatically generated.

Получив *f(i)*, мы по сути найдем основную часть рекуррентного соотношения, так как

A picture containing object

Description automatically generated.

Правда, при этом нельзя забывать о начальном условии, без которого рекуррентная формула не будет полной.

Определим рекуррентное соотношение для ряда: A picture containing object, clock

Description automatically generated.

ПоложимA picture containing object, clock

Description automatically generated. Тогда

A picture containing object, clock

Description automatically generated.

Получаем рекуррентное соотношение:*A picture containing object

Description automatically generated*.

Рекуррентное соотношение используется в тех случаях, когда следующий член ряда можно выразить через предыдущий (или предыдущие), а использование формулы общего члена приведет к появлению вложенных циклов, что сделает программу неэффективной.

1. **Разработка программного обеспечения**

Для разработки программы для выполнения данной лабораторной работы была разработана программа на языке Java под JVM.

Kotlin (Ко́тлин) — статически типизированный язык программирования, разрабатываемый компанией JetBrains.

**Описание функций программы**

Разработанная программа состоит из 7 функций.

|  |  |
| --- | --- |
| **Название функции** | **Описание функции** |
| main(@NotNull String[] args) | Функция с которой начинается выполнение программы, в которой содержится логика верхнего уровня по управлению программой. |
| getPrecisionValue():int | Считывает с командной строки точность вычисления, которую ввел пользователь и проверяет введенное значение на валидность. |
| getAngleValue()double | Считывает с командной строки угол в радианах для вычисления cos(x), которую ввел пользователь и проверяет введенное значение на валидность. |
| sin(double x, int precision): double | Вычисляет значение функции sin(x) с заданной точностью precision. |
| nextSequenceElement(double i, double x, double prevElement):double | Возвращает i-ый рекуррентный элемент последовательности при входном i-1-ом элементе. |
| private static boolean precisionIsChanging(double sum, int precision, double nextElement):boolean | Возвращает значение типа Boolean, сообщающее о том, достигнута ли заданная точность вычисления функции cos(x) |
| round(double $this$round, int decimals):double | Extension функция для типа Double, округляющее значение типа Double до decimals знаков после запятой. |

**Описание работы программы**

Написанная программа исполняется на JVM, поэтому для запуска необходимо в командной строке ввести команду:

java -jar lab1-1.0-jar-with-dependencies.jar

После чего в командной строке отобразится сообщение о том, что необходимо ввести угол в радианах:

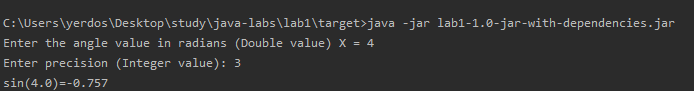
A close up of a sign

Description automatically generated

После ввода угла необходимо ввести точность вычисления:



После ввода точности программа отобразит результат c заданной точностью:



При вводе некорректных данных программа напечатает в консоль сообщение об ошибке и предложит ввести данные заново.

Чтобы выйти из программы необходимо ввести стандартное сочетание клавиш «Control + C».

**Исходный код программы**

package com;  
  
import kotlin.Metadata;  
import kotlin.io.ConsoleKt;  
import kotlin.jvm.internal.Intrinsics;  
import kotlin.math.MathKt;  
import org.jetbrains.annotations.NotNull;  
  
@Metadata(  
 mv = {1, 1, 16},  
 bv = {1, 0, 3},  
 k = 2,  
 d1 = {"\u0000,\n\u0000\n\u0002\u0010\u0006\n\u0002\b\u0002\n\u0002\u0010\b\n\u0002\b\u0003\n\u0002\u0010\u0002\n\u0000\n\u0002\u0010\u0011\n\u0002\u0010\u000e\n\u0002\b\u0005\n\u0002\u0010\u000b\n\u0002\b\u0005\u001a\u0018\u0010\u0000\u001a\u00020\u00012\u0006\u0010\u0002\u001a\u00020\u00012\u0006\u0010\u0003\u001a\u00020\u0004H\u0000\u001a\b\u0010\u0005\u001a\u00020\u0001H\u0000\u001a\b\u0010\u0006\u001a\u00020\u0004H\u0000\u001a\u0019\u0010\u0007\u001a\u00020\b2\f\u0010\t\u001a\b\u0012\u0004\u0012\u00020\u000b0\n¢\u0006\u0002\u0010\f\u001a \u0010\r\u001a\u00020\u00012\u0006\u0010\u000e\u001a\u00020\u00042\u0006\u0010\u0002\u001a\u00020\u00012\u0006\u0010\u000f\u001a\u00020\u0001H\u0002\u001a \u0010\u0010\u001a\u00020\u00112\u0006\u0010\u0012\u001a\u00020\u00012\u0006\u0010\u0003\u001a\u00020\u00042\u0006\u0010\u0013\u001a\u00020\u0001H\u0002\u001a\u0012\u0010\u0014\u001a\u00020\u0001\*\u00020\u00012\u0006\u0010\u0015\u001a\u00020\u0004¨\u0006\u0016"},  
 d2 = {"cos", "", "x", "precision", "", "getAngleValue", "getPrecisionValue", "main", "", "args", "", "", "([Ljava/lang/String;)V", "nextSequenceElement", "i", "prevElement", "precisionIsChanging", "", "sum", "nextElement", "round", "decimals", "lab1"}  
)  
public final class Main {  
 public static void main(@NotNull String[] args) throws Throwable {  
 Intrinsics.*checkParameterIsNotNull*(args, "args");  
  
 while (true) {  
 try {  
 double x = *getAngleValue*();  
 int precision = *getPrecisionValue*();  
 String var4 = "sin(" + x + ")=" + *sin*(x, precision);  
 System.*out*.println(var4);  
 } catch (Exception var6) {  
 String var2 = var6.getMessage();  
 System.*out*.println(var2);  
 }  
 }  
 }  
  
 public static int getPrecisionValue() throws Throwable {  
 String var0 = "Enter precision (Integer value): ";  
 boolean var1 = false;  
 System.*out*.print(var0);  
  
 try {  
 String var10000 = ConsoleKt.*readLine*();  
 if (var10000 == null) {  
 Intrinsics.*throwNpe*();  
 }  
  
 var0 = var10000;  
 int var6 = Integer.*parseInt*(var0);  
 if (var6 > 308) {  
 throw new IllegalArgumentException("Too large value for precision. Max precision 308 decimal places");  
 } else if (var6 < 0) {  
 throw new IllegalArgumentException("Precision must be greater zero.");  
 } else {  
 return var6;  
 }  
 } catch (NumberFormatException var5) {  
 throw new IllegalArgumentException("Incorrect input. Please, enter Integer value.");  
 }  
 }  
  
 public static double getAngleValue() throws Throwable {  
 String var0 = "Enter the angle value in radians (Double value) X = ";  
 System.*out*.print(var0);  
  
 try {  
 String var10000 = ConsoleKt.*readLine*();  
 if (var10000 == null) {  
 Intrinsics.*throwNpe*();  
 }  
  
 var0 = var10000;  
 return Double.*parseDouble*(var0);  
 } catch (NumberFormatException var2) {  
 throw new IllegalArgumentException("Incorrect input. Please, enter Double value.");  
 }  
 }  
  
 public static double sin(double x, int precision) {  
 double angleInRadians = x % 6.283185307179586D;  
 double currentElement = x;  
 double sum = 0.0D;  
  
 for (int var9 = 1; *precisionIsChanging*(sum, precision, currentElement); currentElement = *nextSequenceElement*(var9++, angleInRadians, currentElement)) {  
 sum += currentElement;  
 }  
  
 return *round*(sum, precision);  
 }  
  
 private static double nextSequenceElement(double i, double x, double prevElement) {  
 double var10000 = (double) -1 \* prevElement;  
 double var7 = 2.0D;  
 double var12 = Math.*pow*(x, var7);  
 return var10000 \* (var12 / ((2 \* i + 1) \* 2 \* i));  
 }  
  
 private static boolean precisionIsChanging(double sum, int precision, double nextElement) {  
 return Double.*compare*(*round*(sum, precision), *round*(sum + nextElement, precision)) != 0;  
 }  
  
 public static double round(double $this$round, int decimals) {  
 double multiplier = 1.0D;  
 int var10 = 0;  
  
 for (; var10 < decimals; ++var10) {  
 multiplier \*= 10;  
 }  
  
 return (double) MathKt.*roundToLong*($this$round \* multiplier) / multiplier;  
 }  
}

Все данные по выполненной работе можно найти в [репозитории на Github](https://github.com/Sindicat/krlabs/tree/master/lab1/) (<https://github.com/Hitsugaya7/java-labs/tree/master/lab1>): исходный код в директории src, скомпилированный .jar файл в директории target.

**Выводы**

В лабораторной работе была разработана программа для вычисления функции sin(x) через рекуррентную формулу на JVM языке Java.