 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 №3

|  |  |  |
| --- | --- | --- |
| Студент гр. 5371 |  | Мартынов М. |
| Студентка гр. 5371 |  | Козлова С. |
| Студент гр. 5371 |  | Аверкиев В. |
| Преподаватель |  | Кораблев Ю.А. |

Санкт-Петербург

2019

1. **Цель работы**

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

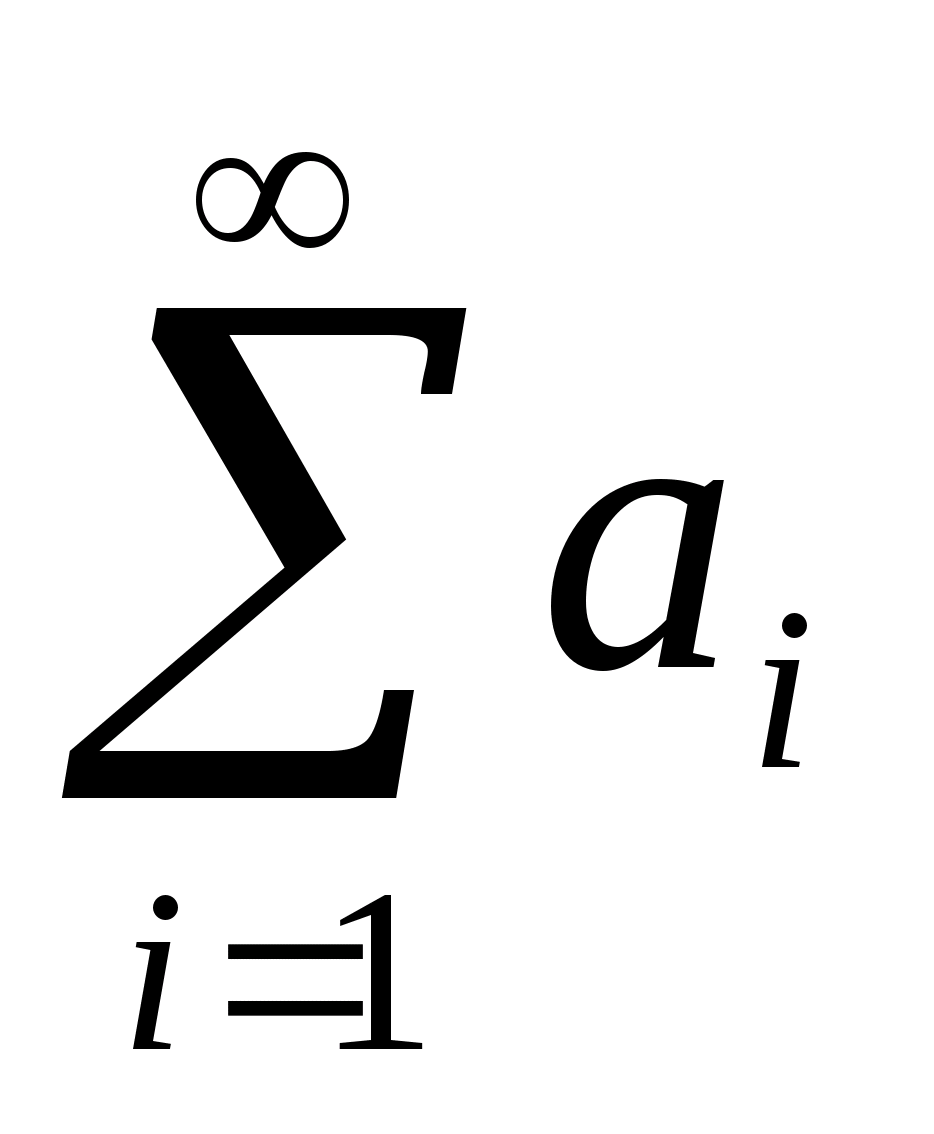
1. **Задание на лабораторную работу №1. Вариант №3.**

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 | cos(x) |  |  |

1. **Разработка математического и алгоритмического обеспечения**

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

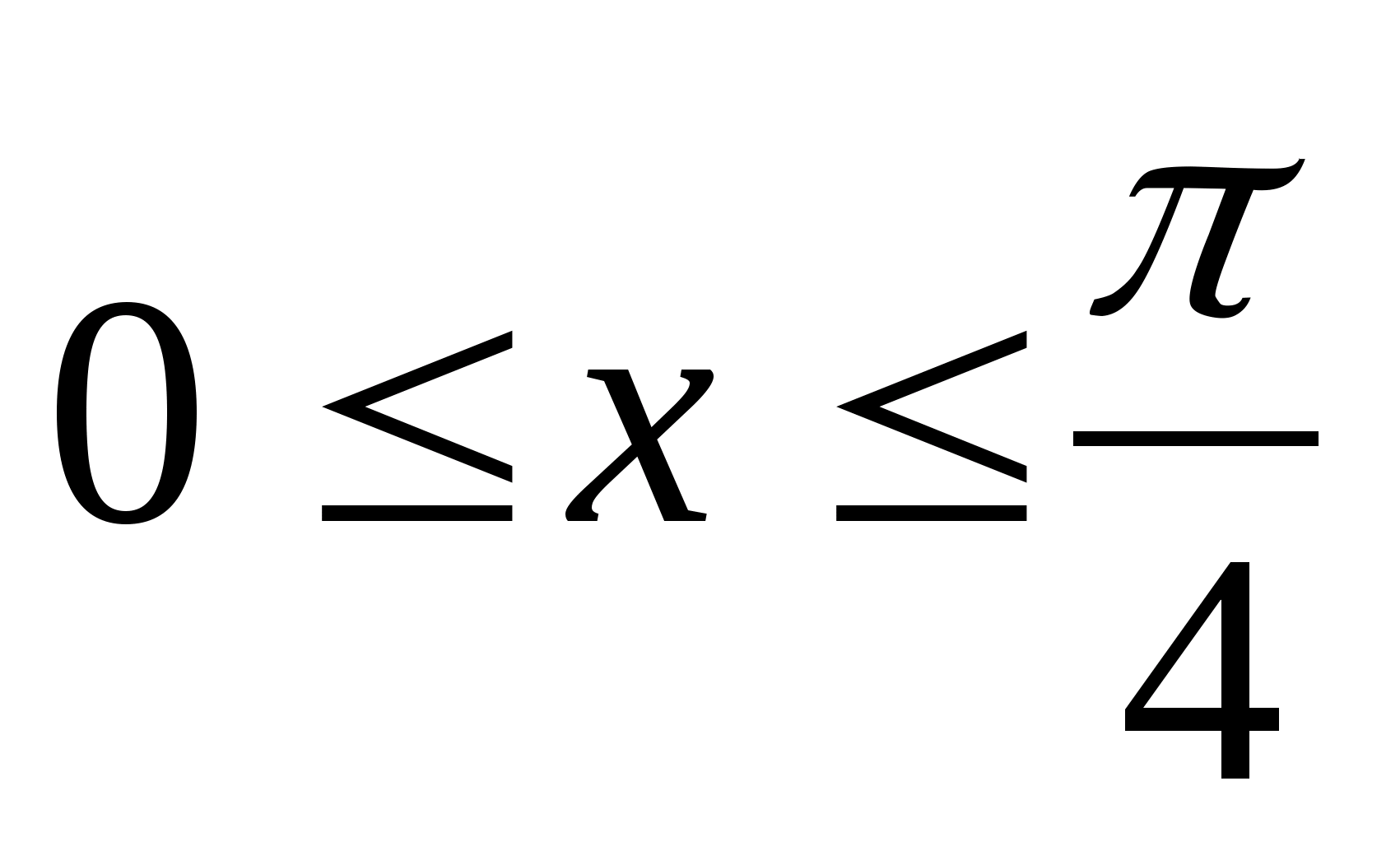
*S* = *a*1+*a*2+*a*3+…+*ai*+… = 

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.

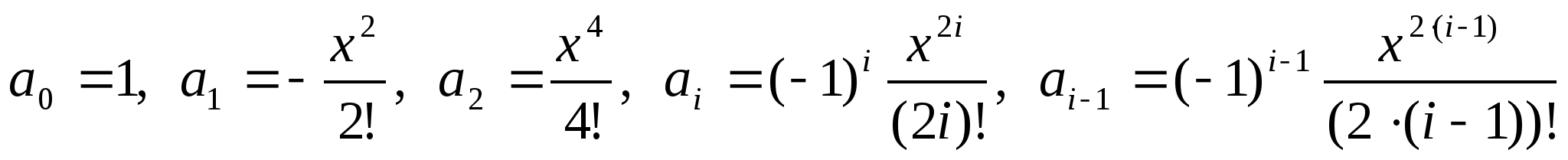
Получаем рекуррентное соотношение:*A picture containing object

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.

**Рекуррентная формула для вычисления функции cos(x)**

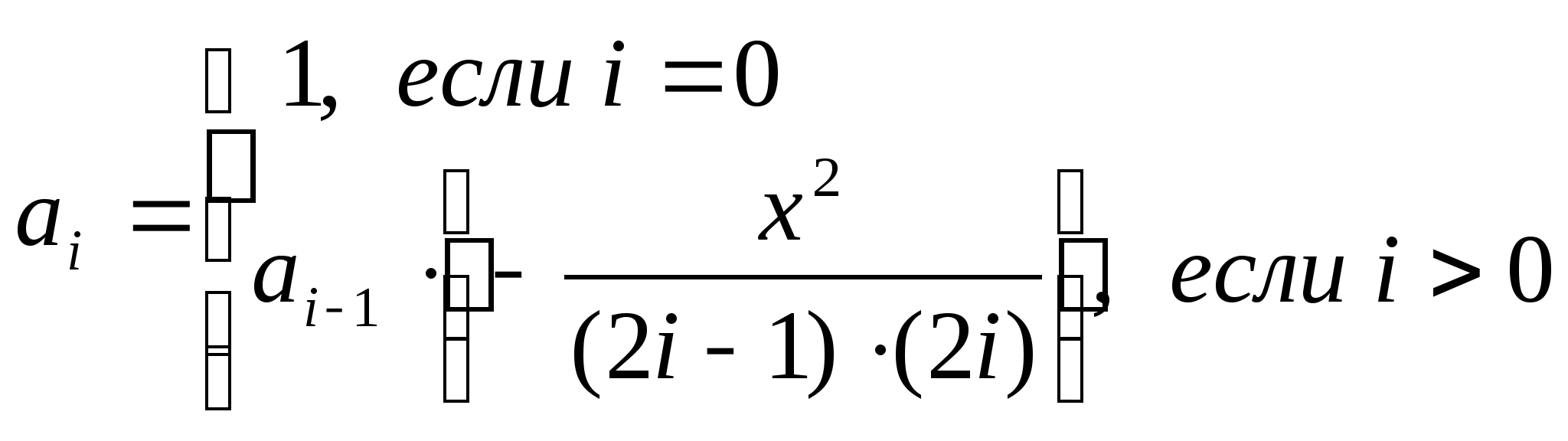
*Выпишем члены ряда*

.

Поскольку каждый член ряда может быть получен из предыдущего домножением на определенный множитель, а вычисление «в лоб» каждого члена ряда приведет к появлению вложенных циклов, эффективнее будет использовать рекуррентное соотношение. Выведем его

A picture containing object

Description automatically generated

Полученное рекуррентное cоотношение: **

1. **Разработка программного обеспечения**

Для разработки программы для выполнения данной лабораторной работы была разработана программа на языке Kotlin 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(args: Array<String>) | 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. |
| cos(x: Double, precision: Int): Double | Вычисляет значение функции cos(x) с заданной точностью precision. |
| nextSequenceElement(i: Int, x: Double, prevElement: Double): Double | Возвращает i-ый рекуррентный элемент последовательности при входном i-1-ом элементе. |
| precisionIsChanging(sum: Double, precision: Int, nextElement: Double): Boolean | Возвращает значение типа Boolean, сообщающее о том, достигнута ли заданная точность вычисления функции cos(x) |
| Double.round(decimals: Int): 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.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:

A black sign with white text

Description automatically generated

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.github.sindicat

import java.lang.Exception

import kotlin.math.pow

import kotlin.math.roundToLong

fun main(args: Array<String>) {

while (true) {

try {

val x: Double = getAngleValue()

val precision: Int = getPrecisionValue()

println("cos($x)=${cos(x, precision)}")

} catch (e: Exception) {

println(e.message)

}

}

}

internal fun getPrecisionValue(): Int {

print("Enter precision (Integer value): ")

try {

return readLine()!!.toInt().also {

if (it > 308) throw IllegalArgumentException("Too large value for precision. Max precision 308 decimal places")

if (it < 0) throw IllegalArgumentException("Precision must be greater zero.")

}

} catch (e: NumberFormatException) {

throw IllegalArgumentException("Incorrect input. Please, enter Integer value.")

}

}

internal fun getAngleValue(): Double {

print("Enter the angle value in radians (Double value) X = ")

try {

return readLine()!!.toDouble()

} catch (e: NumberFormatException) {

throw IllegalArgumentException("Incorrect input. Please, enter Double value.")

}

}

internal fun cos(x: Double, precision: Int): Double {

var angleInRadians = x % (2 \* Math.PI)

var currentElement = 1.0

var sum = 0.0

var index = 1

while (precisionIsChanging(sum, precision, currentElement)) {

sum += currentElement

currentElement = nextSequenceElement(index++, angleInRadians, currentElement)

}

return sum.round(precision)

}

private fun nextSequenceElement(i: Int, x: Double, prevElement: Double): Double =

-1 \* prevElement \* ((x.pow(2.0)) / ((2 \* i - 1) \* 2 \* i))

private fun precisionIsChanging(sum: Double, precision: Int, nextElement: Double): Boolean =

sum.round(precision).compareTo((sum + nextElement).round(precision)) != 0

fun Double.round(decimals: Int): Double {

var multiplier = 1.0

repeat(decimals) { multiplier \*= 10 }

return (this \* multiplier).roundToLong() / multiplier

}

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

**Выводы**

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