Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Управляющие конструкции: функции, паттерн-матчинг, исключения
Тема

Преподаватель _____ А. А. Чикизов
подпись, дата — инициалы, фамилия

Студент КИ19-16/1б 031939175 _____ А. Д. Непомнящий
номер группы, зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2022

1 Задание

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $X_{нач}$ до $X_{кон}$ с шагом dx с точностью e. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

Вариант 11.

$$11.\quad \operatorname{arth} x = \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = \frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \ldots,\ |x| < 1.$$

2 Исходный код основного алгоритма

Исходный кот программы приведен на листинге 1.

Листинг 1 — исходный код основного алгоритма

```
package com.github.durakin
package lab1

import scala.annotation.tailrec
import scala.io.StdIn.readLine
import scala.sys.exit


@main
def Main(): Unit =
  val x0 = scanDouble("Enter X0")
  val x1 = scanDouble("Enter Xn")
  val dx = math.abs(scanDouble("Enter dX (sign will be omitted)"))
    val e = math.abs(scanDouble("Enter required precision (sign will be
omitted)"))
  val result = taylorArtanhInterval(x0, x1, dx, e)
  println("Every X with absolute value not above 1 will be skipped")
  val s = "%-8s | %-12s | %-12s | %-18s | %-8s | Iterations"
  println(s.format("x", "~f(x)", "f(x)", "delta", "delta/e"))
    result.foreach(x => println(f"${x._1}%08.4f | ${x._2._1}%012.9f | $
{artanh(x._1)}%012.9f | ${math.abs(x._2._1 - artanh(x._1))}%1.16f | $
{math.abs(x._2._1 - artanh(x._1)) / e}%08.6f | ${x._2._2}%6d"))
  exit(0)

def artanh(x: Double): Double =
  0.5 * math.log((1.0 + x) / (x - 1.0))

def taylorArtanhMono(x: Double, n: Int): Double =
  1.0 / ((2 * n + 1) * math.pow(x, 2 * n + 1))

@tailrec
def taylorArtanhRow(x: Double, e: Double, n: Int = 0, res: Double = 0): (Double,
Int) =
```

Окончание листинга 1

```
  if math.abs(taylorArtanhMono(x, n)) < e then (res, n)
  else
    taylorArtanhRow(x, e, n + 1, res + taylorArtanhMono(x, n))


def taylorArtanhInterval(x0: Double, x1: Double, dx: Double, e: Double):
List[(Double, (Double, Int))] =
  if x0 >= x1 then Nil
    else if math.abs(x0) > 1.1 then List((x0, taylorArtanhRow(x0, e))) :::
taylorArtanhInterval(x0 + dx, x1, dx, e)
  else taylorArtanhInterval(x0 + dx, x1, dx, e)


def scanDouble(msg: String = ""): Double =
  println(msg)
  try
    readLine().toDouble
  catch
    case nfe: NumberFormatException =>
      println("Got a NumberFormatException.")
      exit(-1)
```

3 Результаты

x	~f(x)	f(x)	delta	delta/e	Iterat
-10.0000	-0.100333333	-0.100335348	0.0000020143977422	0.201440	2
-09.7500	-0.102923740	-0.102926027	0.0000022870974665	0.228710	2
-09.5000	-0.105651941	-0.105654547	0.0000026053455343	0.260535	2
-09.2500	-0.108529274	-0.108532253	0.0000029782751763	0.297828	2
-09.0000	-0.111568358	-0.111571776	0.0000034171751661	0.341718	2
-08.7500	-0.114783285	-0.114787221	0.0000039360797817	0.393608	2
-08.5000	-0.118189836	-0.118194389	0.0000045525438867	0.455254	2
-08.2500	-0.121805753	-0.121811041	0.0000052886691281	0.528867	2
-08.0000	-0.125651042	-0.125657214	0.0000061724737864	0.617247	2
-07.7500	-0.129748358	-0.129755598	0.0000072397373282	0.723974	2
-07.5000	-0.134123457	-0.134131993	0.0000085365072162	0.853651	2
-07.2500	-0.138805746	-0.138815868	0.0000101225394393	1.012254	2
-07.0000	-0.143840860	-0.143841036	0.0000001762667464	0.017627	3
-06.7500	-0.149246267	-0.149246494	0.0000002276459823	0.022765	3
-06.5000	-0.155077167	-0.155077464	0.0000002968833026	0.029688	3
-06.2500	-0.161386305	-0.161386696	0.0000003912781922	0.039128	3
-06.0000	-0.168235597	-0.168236118	0.0000005216027875	0.052160	3
-05.7500	-0.175698239	-0.175698943	0.0000007040098057	0.070401	3
-05.5000	-0.183861427	-0.183862390	0.0000009631440635	0.096314	3
-05.2500	-0.192829903	-0.192831240	0.0000013373257306	0.133733	3
-05.0000	-0.202730667	-0.202732554	0.0000018873874155	0.188739	3
-04.7500	-0.213719295	-0.213722007	0.0000027121295036	0.271213	3
-04.5000	-0.225988586	-0.225992562	0.0000039761207115	0.397612	3
-04.2500	-0.239780579	-0.239786540	0.0000059612886137	0.596129	3
-04.0000	-0.255403646	-0.255412812	0.0000091660496621	0.916605	3
-03.7500	-0.273271049	-0.273271853	0.0000008045721074	0.080457	4
-03.5000	-0.293891822	-0.293893332	0.0000015109178155	0.151092	4
-03.2500	-0.317991405	-0.317994383	0.0000029780051610	0.297801	4
-03.0000	-0.346567379	-0.346573590	0.0000062116138286	0.621161	4
-02.7500	-0.381068521	-0.381070026	0.0000015053701199	0.150537	5
-02.5000	-0.423644518	-0.423648930	0.0000044126063002	0.441261	5
-02.2500	-0.477753270	-0.477755723	0.0000024521937219	0.245219	6
-02.0000	-0.549294141	-0.549306144	0.0000120031453826	1.200315	6
-01.7500	-0.649635344	-0.649641492	0.0000061482540488	0.614825	8
-01.5000	-0.804702837	-0.804718956	0.0000161196626847	1.611966	10

-01.2500	-1.098594326	-1.098612289	0.0000179631494921	1.796315	18
001.2500	01.098594326	01.098612289	0.0000179631494921	1.796315	18
001.5000	00.804702837	00.804718956	0.0000161196626847	1.611966	10
001.7500	00.649635344	00.649641492	0.0000061482540488	0.614825	8
002.0000	00.549294141	00.549306144	0.0000120031453826	1.200315	6
002.2500	00.477753270	00.477755723	0.0000024521937219	0.245219	6
002.5000	00.423644518	00.423648930	0.0000044126063002	0.441261	5
002.7500	00.381068521	00.381070026	0.0000015053701199	0.150537	5
003.0000	00.346567379	00.346573590	0.0000062116138286	0.621161	4
003.2500	00.317991405	00.317994383	0.0000029780051610	0.297801	4
003.5000	00.293891822	00.293893332	0.0000015109178155	0.151092	4
003.7500	00.273271049	00.273271853	0.0000008045721074	0.080457	4
004.0000	00.255403646	00.255412812	0.0000091660496621	0.916605	3
004.2500	00.239780579	00.239786540	0.0000059612886137	0.596129	3
004.5000	00.225988586	00.225992562	0.0000039761207115	0.397612	3
004.7500	00.213719295	00.213722007	0.0000027121295037	0.271213	3
005.0000	00.202730667	00.202732554	0.0000018873874155	0.188739	3
005.2500	00.192829903	00.192831240	0.0000013373257307	0.133733	3
005.5000	00.183861427	00.183862390	0.0000009631440635	0.096314	3
005.7500	00.175698239	00.175698943	0.0000007040098057	0.070401	3
006.0000	00.168235597	00.168236118	0.0000005216027875	0.052160	3
006.2500	00.161386305	00.161386696	0.0000003912781922	0.039128	3
006.5000	00.155077167	00.155077464	0.0000002968833026	0.029688	3
006.7500	00.149246267	00.149246494	0.0000002276459823	0.022765	3
007.0000	00.143840860	00.143841036	0.0000001762667464	0.017627	3
007.2500	00.138805746	00.138815868	0.0000101225394394	1.012254	2
007.5000	00.134123457	00.134131993	0.0000085365072162	0.853651	2
007.7500	00.129748358	00.129755598	0.0000072397373282	0.723974	2
008.0000	00.125651042	00.125657214	0.0000061724737864	0.617247	2
008.2500	00.121805753	00.121811041	0.0000052886691281	0.528867	2
008.5000	00.118189836	00.118194389	0.0000045525438867	0.455254	2
008.7500	00.114783285	00.114787221	0.0000039360797817	0.393608	2
009.0000	00.111568358	00.111571776	0.0000034171751662	0.341718	2
009.2500	00.108529274	00.108532253	0.0000029782751763	0.297828	2
009.5000	00.105651941	00.105654547	0.0000026053455343	0.260535	2
009.7500	00.102923740	00.102926027	0.0000022870974666	0.228710	2