ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | И.Д. Свеженин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| Функциональное программирование |
| по курсу: Кроссплатформенное программирование |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

Содержание

[1. Цель работы: 3](#_Toc209582462)

[2. Задание: 3](#_Toc209582463)

[3. Скриншоты, иллюстрирующие результаты работы программы: 4](#_Toc209582464)

[4. Вывод: 7](#_Toc209582465)

# **1. Цель работы:**

Изучение и практическое применение синтаксиса и возможностей языка высокого уровня Kotlin с использованием парадигмы функционального программирования.

# **2. Задание:**

Программа должна вычислять значение заданной функции путем разложения в ряд Маклорена с заданной точностью и с использованием стандартной функции класса Math. Аргумент функции и точность должны задаваться пользователем. Ввод и вывод информации можно осуществлять через командную строку. При разработке следует придерживаться принципов функционального программирования. Выбранный вариант – 7.

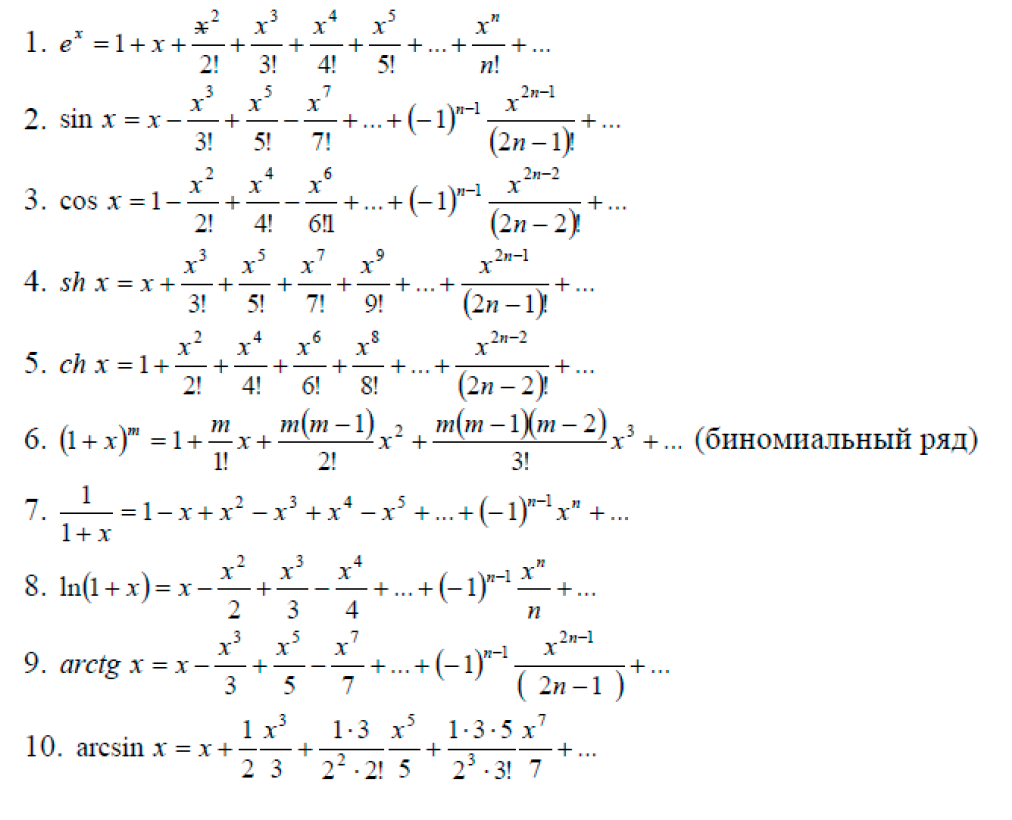


Рисунок 1.1 – Таблица с вариантами функций.

**3. Краткое описание хода разработки, алгоритма работы программы и назначение используемых технологий**

Ход разработки:

1. Подключение функции класса Math.
2. Разработка главной функции main:
   1. Главный цикл вычислений n-ного члена ряда.
   2. Вычисление значения суммы членов ряда по упрощённой формуле.
3. Добавление вывода результата вычислений.
4. Добавление ввода значения х и точности.
5. Исправление ошибок в вычислении суммы членов ряда.

**Используемые технологии:**

* Kotlin – используется для работы написанного кода
* Math – используется для функции pow – возведения в степень (в других вариантах для вычисления стандартной функции)

**Описание алгоритма работы программы:**

*Вывод информации пользователю*

На экран выводится название задачи, формула функции

и область сходимости ряда ().

*Ввод данных*

* Пользователь вводит число x.
* Пользователь задаёт требуемую точность ε (маленькое положительное число, например 0.001).

*Проверка условия сходимости*

Если введённый x не удовлетворяет условию , программа сообщает об ошибке и завершается.

*Инициализация переменных для расчёта ряда*

* sum = 0.0 — накопленная сумма ряда.
* n = 0 — номер текущего члена ряда.
* term — значение текущего члена ряда.
* lastTerm — переменная для контроля величины последнего члена.

*Циклическое вычисление ряда Маклорена*

В цикле do ... while вычисляется очередной член ряда по формуле:

Этот член прибавляется к сумме: sum += term.

Значение и номер члена выводятся в виде таблицы: номер, член ряда, промежуточная сумма.

Переменная n увеличивается на 1.

Цикл продолжается, пока модуль последнего члена ряда больше заданной точности ε.

*Вычисление точного значения функции*

После выхода из цикла программа считает «стандартное» значение функции напрямую по формуле: f​

*Сравнение результатов*

* Выводится итоговая сумма ряда.
* Выводится точное значение функции.
* Считается разница между ними:

*Вывод итогов*

На экран печатаются:

* введённые данные (x, ε),
* количество вычисленных членов ряда,
* итоговая сумма ряда,
* точное значение функции,
* разница между ними.

*Смысл алгоритма*

Программа **приближённо вычисляет функцию** f​ с помощью ряда Маклорена, **сравнивает результат с точным значением** и показывает, насколько быстро и точно ряд сходится при заданном значении x.

# **4. Скриншоты, иллюстрирующие результаты работы программы:**

В качестве демонстрации работы программы приведём несколько скриншотов с разными значениями х и точностью.

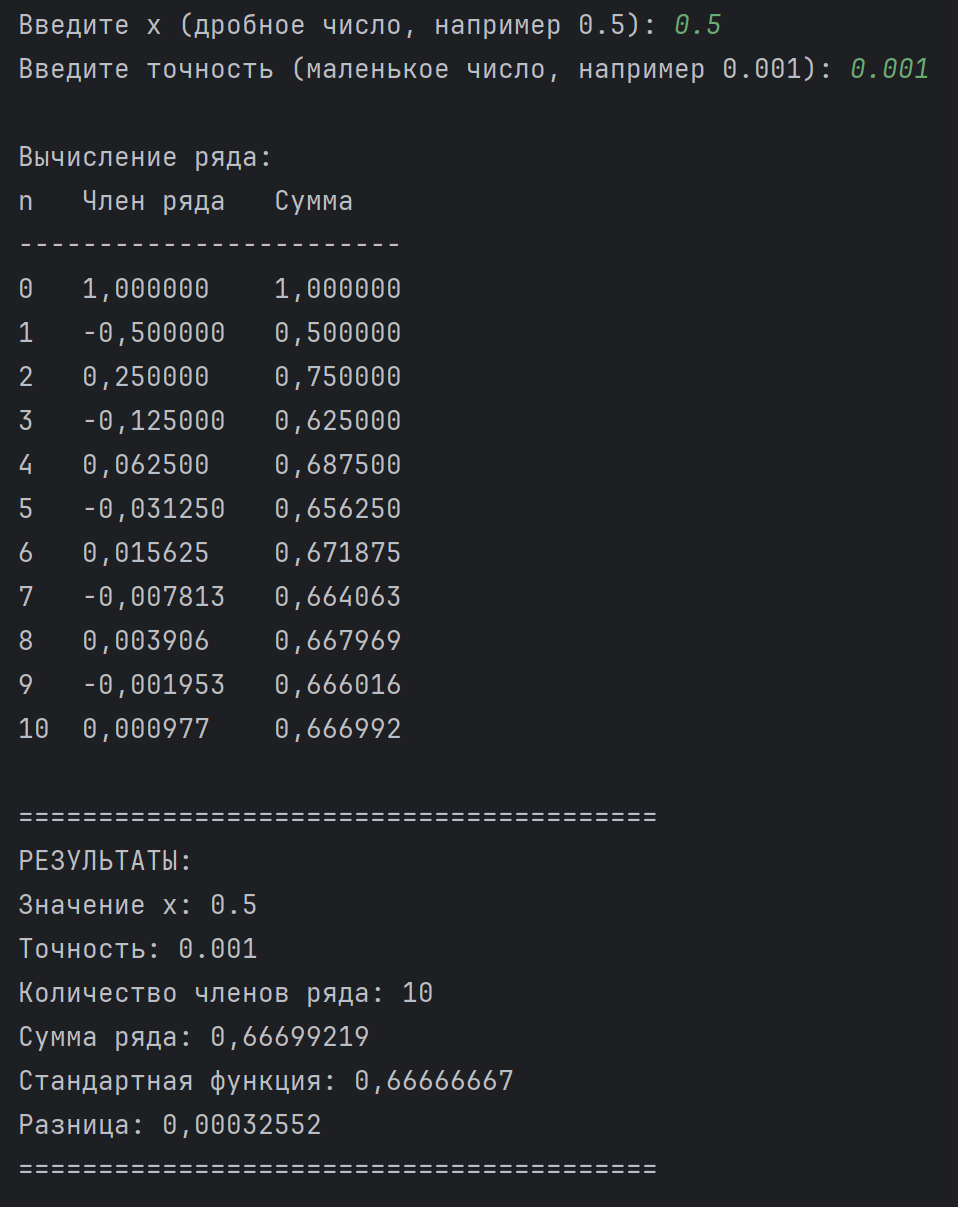


Рисунок 2.1 – Результат работы программы, пример 1

Как мы видим из значения разницы в вычислении значения стандартной функции и суммы членов разложенного ряда, результаты почти сходятся, что говорит о правильности работы программы.

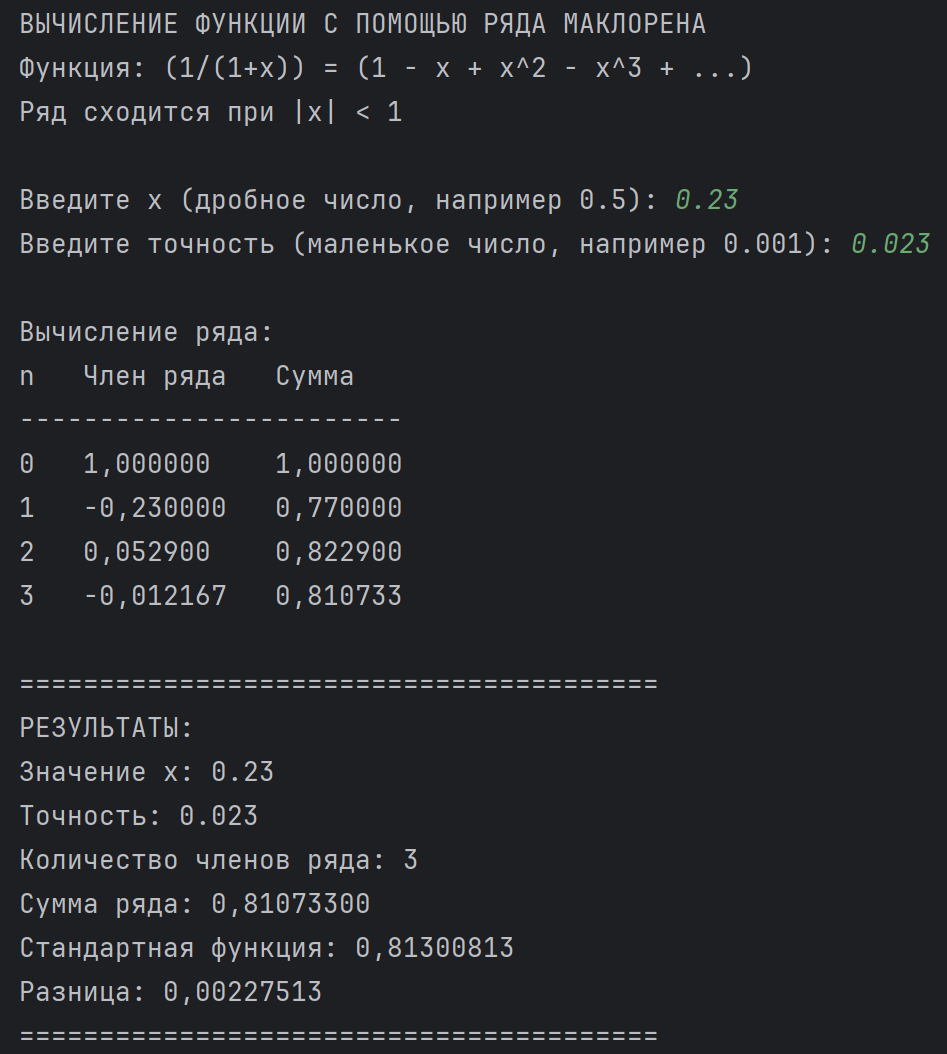


Рисунок 2.2 – Результат работы программы, пример 2

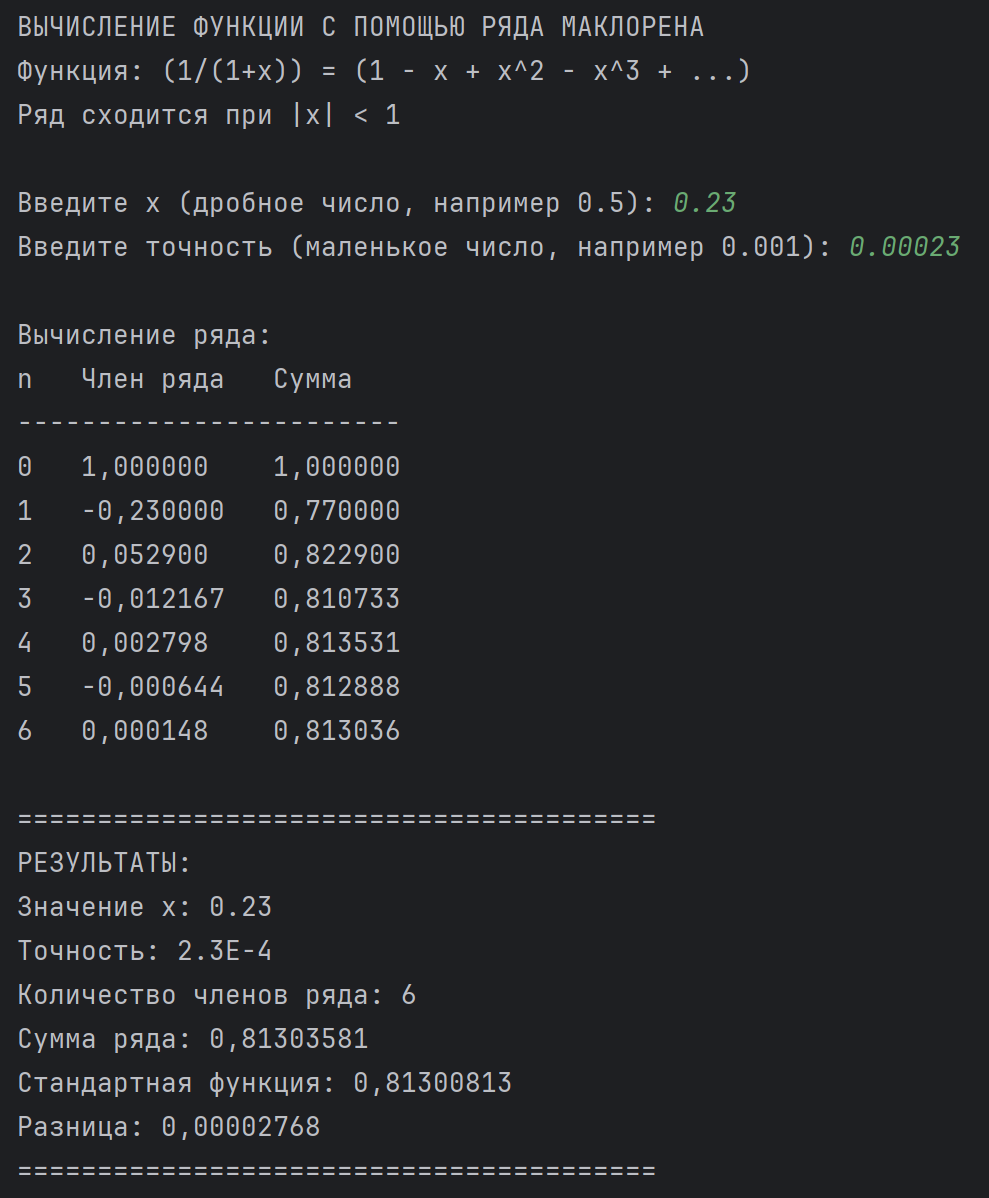


Рисунок 2.3 – Результат работы программы, пример 3

Сравнив результаты работы с рисунков 2.2 и 2.3 можем увидеть, как влияет значение точности на вычисления – увеличивается количество членов, результат через ряд становится точнее.

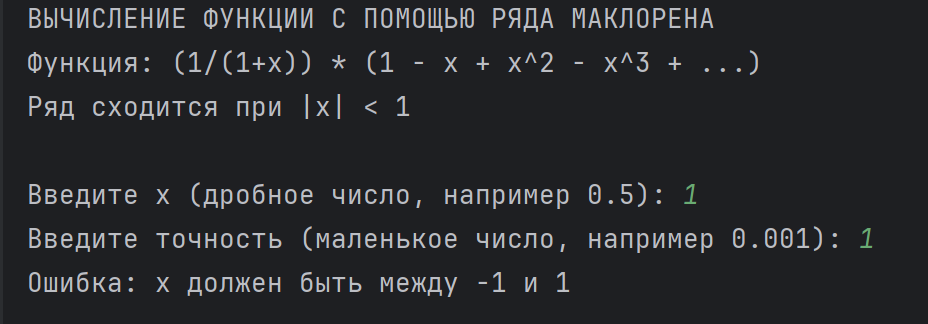


Рисунок 3.1 – Попытка ввести плохие значения

Если попытаться ввести х целым числом, а не дробным, как требуется для схождения ряда, то программа предупредит пользователя о неправильном вводе.

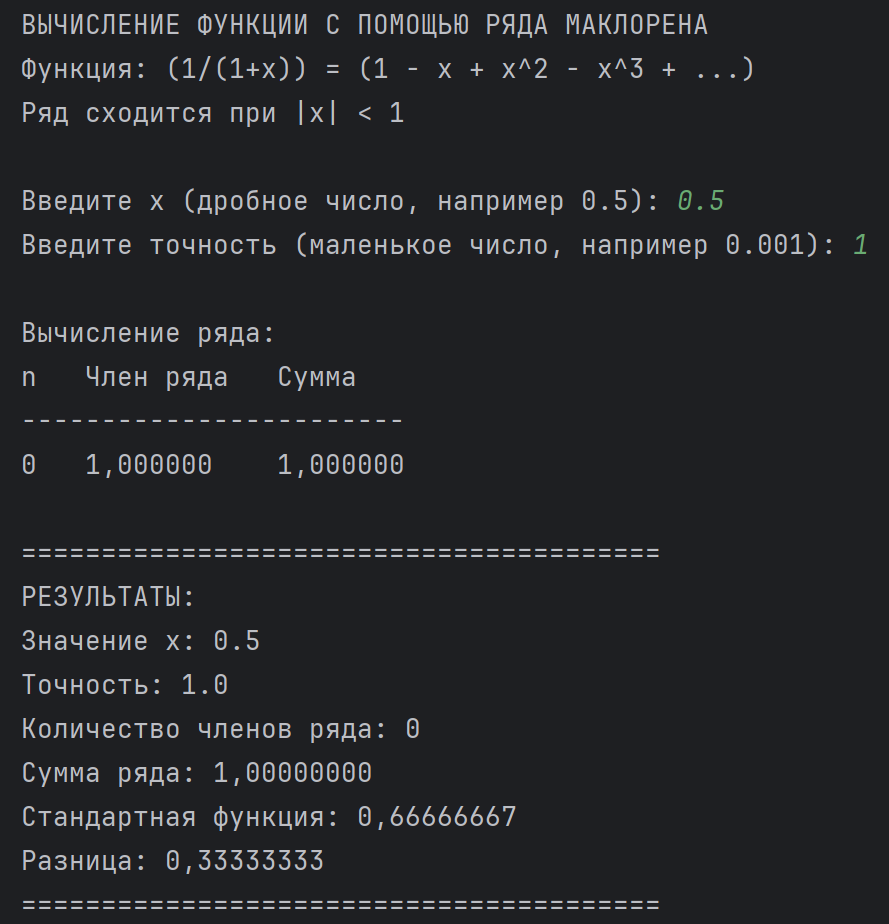


Рисунок 3.2 – Попытка ввести плохую точность

Если же ввести не дробную точность, а целую, то результат вычислений будет просто не точным настолько, насколько большое число ввели. Чем число меньше – тем точнее вычисления.

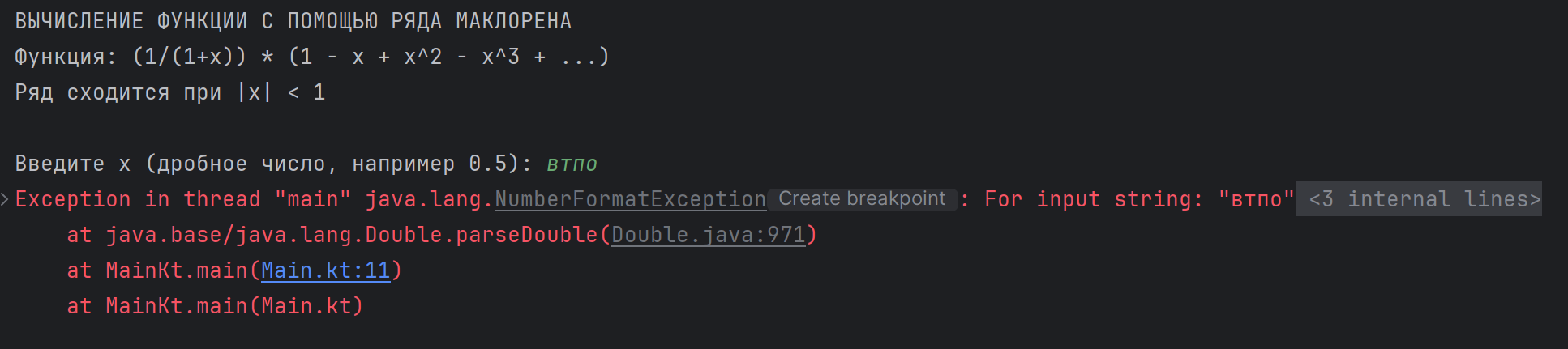


Рисунок 3.3 – Попытка ввести не число

В случае, если пользователь ввёл вместо числа что-угодно другое, программа прекратит свою работу, завершившись с ошибкой.

# **5. Вывод:**

В данной работе мы изучили синтаксис языка высокого уровня – Kotlin, написав программу, вычисляющую значение заданной функции путем разложения в ряд Маклорена с заданной точностью и с использованием стандартной функции класса Math. Аргумент функции х и точность вычислений вводится пользователем в консоль, вывод выполняется также в консоль.

ПРИЛОЖЕНИЕ А

Листинг программы:

import kotlin.math.pow  
  
fun main() {  
 *println*("ВЫЧИСЛЕНИЕ ФУНКЦИИ С ПОМОЩЬЮ РЯДА МАКЛОРЕНА")  
 *println*("Функция: (1/(1+x)) = (1 - x + x^2 - x^3 + ...)")  
 *println*("Ряд сходится при |x| < 1")  
 *println*()  
  
 // Ввод данных  
 *print*("Введите x (дробное число, например 0.5): ")  
 val x = *readLine*()!!.*toDouble*()  
  
 *print*("Введите точность (маленькое число, например 0.001): ")  
 val epsilon = *readLine*()!!.*toDouble*()  
  
 // Проверка условия сходимости  
 if (x >= 1 || x <= -1) {  
 *println*("Ошибка: x должен быть между -1 и 1")  
 return  
 }  
  
 // Вычисление ряда Маклорена  
 var sum = 0.0  
 var n = 0  
 var term: Double  
 var lastTerm: Double  
  
 *println*("\nВычисление ряда:")  
 *println*("n\tЧлен ряда\tСумма")  
 *println*("------------------------")  
  
 do {  
 // Вычисляем n-й член ряда: (-1)^(n-1) \* x^n  
 term = (-1.0).*pow*(n) \* x.*pow*(n)  
 sum += term  
 lastTerm = term  
  
 *println*("$n\t${"%.6f".*format*(term)}\t${"%.6f".*format*(sum)}")  
 n++  
  
 } while (Math.abs(lastTerm) > epsilon) // Продолжаем пока член ряда больше точности  
  
 // Вычисление стандартной функции для сравнения  
 val standard = 1.0 / (1.0 + x) // Упрощенная функция для сравнения  
  
 // Вывод результатов  
 *println*("\n" + "=".*repeat*(40))  
 *println*("РЕЗУЛЬТАТЫ:")  
 *println*("Значение x: $x")  
 *println*("Точность: $epsilon")  
 *println*("Количество членов ряда: ${n-1}")  
 *println*("Сумма ряда: ${"%.8f".*format*(sum)}")  
 *println*("Стандартная функция: ${"%.8f".*format*(standard)}")  
  
 val difference = Math.abs(sum - standard)  
 *println*("Разница: ${"%.8f".*format*(difference)}")  
 *println*("=".*repeat*(40))  
}