Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет

ИТМО»

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

по “Тестирование программного обеспечения”

Вариант №68733

Выполнил:

Студент группы P3318

Горло Евгений Николаевич

Преподаватель:

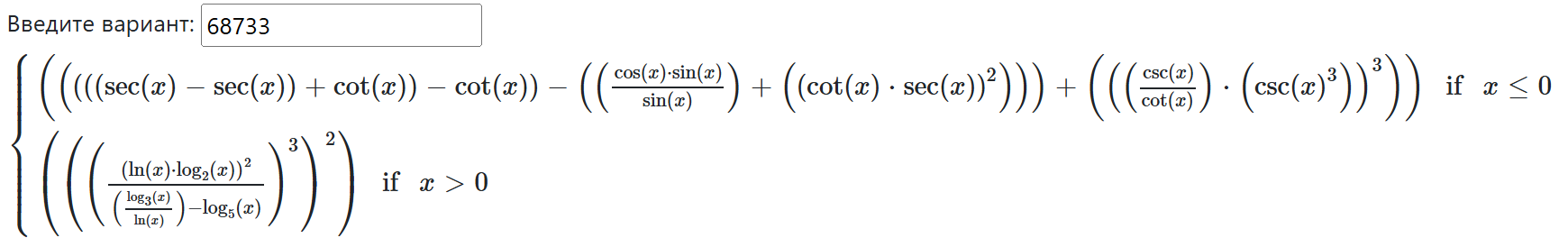
Кулинич Ярослав Вадимович

г. Санкт-Петербург

2025 год

# Задание

Провести интеграционное тестирование программы, осуществляющей вычисление системы функций (в соответствии с вариантом).



**x <= 0:** (((((sec(x) - sec(x)) + cot(x)) - cot(x)) - (((cos(x) \* sin(x)) / sin(x)) + ((cot(x) \* sec(x)) ^ 2))) + (((csc(x) / cot(x)) \* (csc(x) ^ 3)) ^ 3))

**x > 0:** (((((ln(x) \* log\_2(x)) ^ 2) / ((log\_3(x) / ln(x)) - log\_5(x))) ^ 3) ^ 2)

**Правила выполнения работы:**

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):
3. Обе "базовые" функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.
5. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в сsv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

**Порядок выполнения работы:**

1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт <https://www.wolframalpha.com/>.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

# Выполнение

UML диаграмма классов приложения

Изображение выглядит как диаграмма, текст, линия, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Для тестирования выбраны следующие значения:

Изображение выглядит как текст, снимок экрана, Параллельный, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

График при x > 0:

Изображение выглядит как текст, линия, График, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

Графики при x <= 0:

Изображение выглядит как текст, линия, Параллельный, График

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, Параллельный, линия, График

Содержимое, созданное искусственным интеллектом, может быть неверным.

Интеграционное тестирование:

Изображение выглядит как текст, снимок экрана, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Графики, полученные по csv выгрузкам:

От -5.5 до 3 с шагом 0.1

Изображение выглядит как линия, текст, Параллельный, Прямоугольник

Содержимое, созданное искусственным интеллектом, может быть неверным.

Репозиторий с кодом: https://github.com/Djerden/TPO/tree/main/Lab2

# Вывод

В рамках лабораторной работы я познакомился с понятием интеграционного тестирования программы. Реализовал я его согласно подходу «сверху вниз», заменяя все зависимые модули заглушками.