

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования «Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа №2
по дисциплине «Тестирование программного обеспечения»
Вариант №999

Группа: Р3312

Выполнил: Балин А. А., Кобелев Р. П.

Проверил: Кривоносов Е. Д.

Оглавление

Цель работы	3
Порядок выполнения работы	4
Выполнение	5
UML-диаграмма классов разработанного приложения	5
Описание тестового покрытия	5
CSV-выгрузки после тестирования	6
Вывод	8

Цель работы

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

$$\begin{cases} \left(\left(\left(\frac{(\cos(x) + \sin(x)) \cdot \cot(x)}{\tan(x)} \right)^2 \right) + \left(\tan(x) \cdot (\cos(x) - \csc(x)) \right) - \left(\left(\frac{\cot(x) + \csc(x)}{\csc(x)} \right) + \left(\frac{\frac{\cos(x)}{\cot(x)}}{\sec(x)} \right) \right) \right) & \text{if } x \leq 0 \\ \left(\left(\left(\left(\frac{\log_{10}(x)}{\log_3(x)} \right) - \log_5(x) \right)^3 \right) \cdot (\log_2(x) - \log_2(x)) \right) \cdot \ln(x) & \text{if } x > 0 \end{cases}$$

$x \leq 0 : (((((\cos(x) + \sin(x)) * \cot(x)) / \tan(x)) ^ 2) + ((\tan(x) * (\cos(x) - \csc(x))) - (((\cot(x) + \csc(x)) / \csc(x)) + ((\cos(x) / \cot(x)) / \sec(x)))))$
 $x > 0 : (((((\log_{10}(x) / \log_3(x)) - \log_5(x)) ^ 3) * (\log_2(x) - \log_2(x))) * \ln(x))$

Рис. 1. Система функций по варианту.

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 $\sin(x)$):

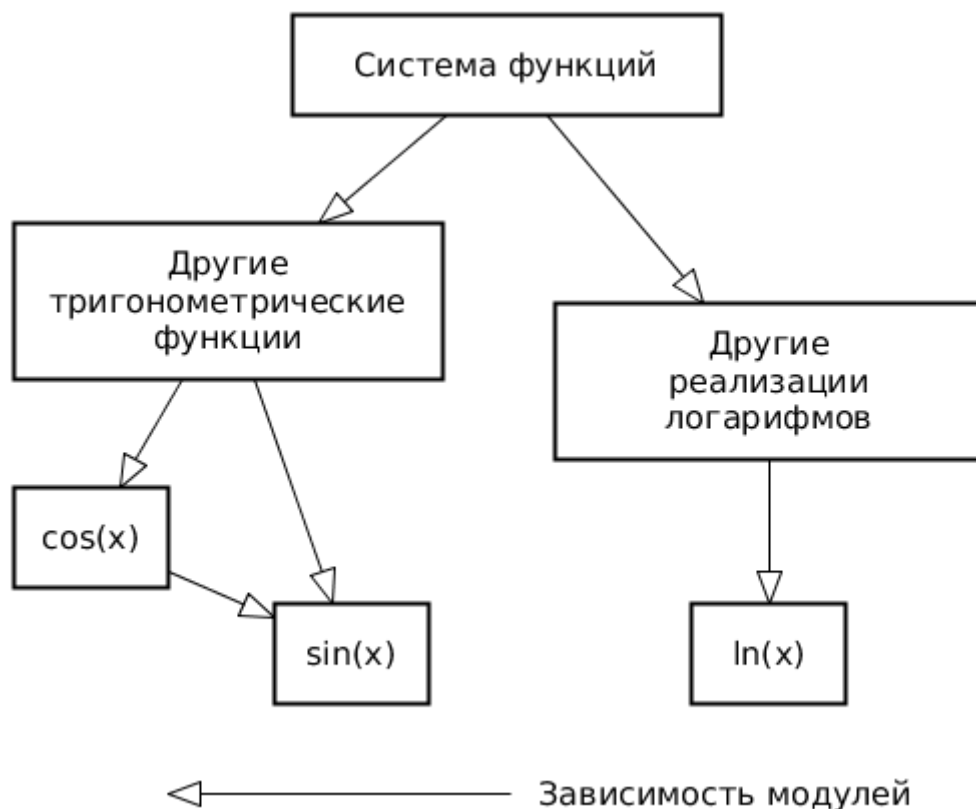


Рис. 2. Пример структуры приложения.

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 csv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания X. 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

Исходный код: <https://github.com/Romariok/Software-Testing/tree/main/lab2>

UML-диаграмма классов разработанного приложения

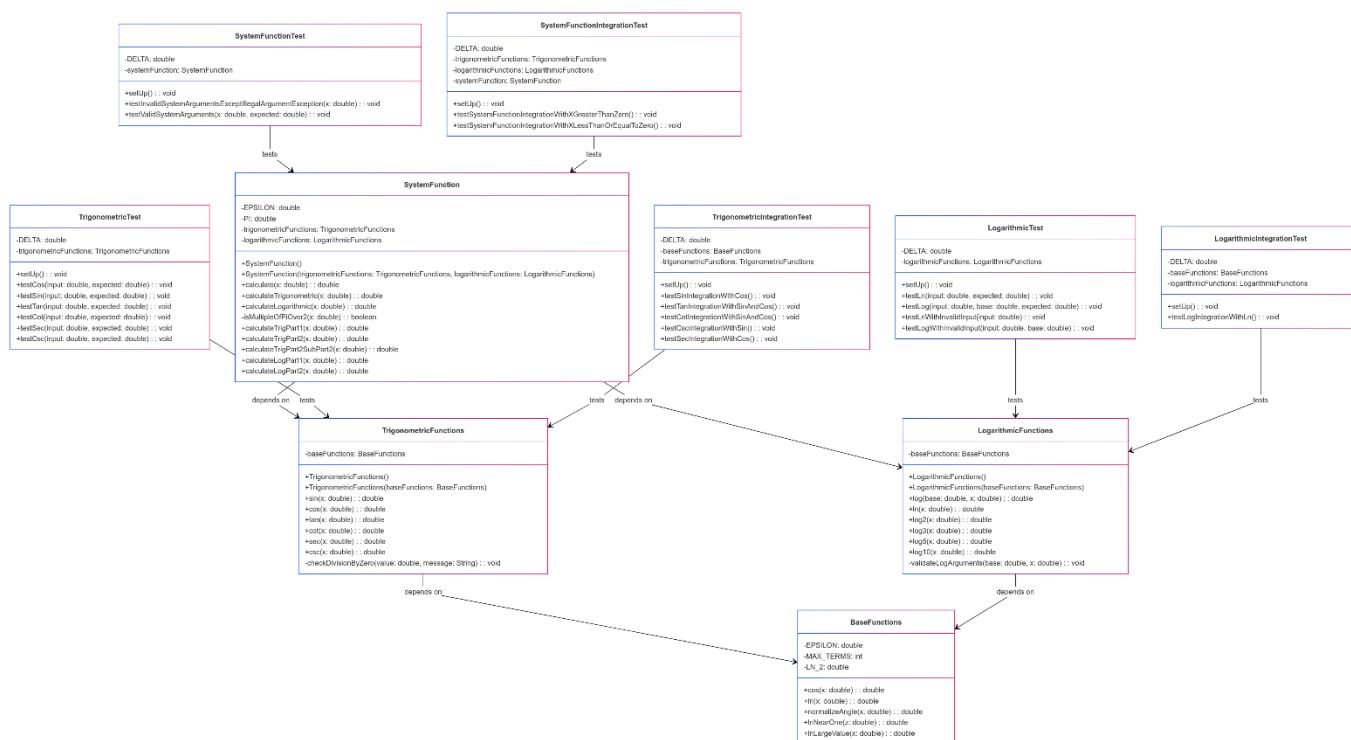


Рис. 3. UML-диаграмма классов.

Описание тестового покрытия

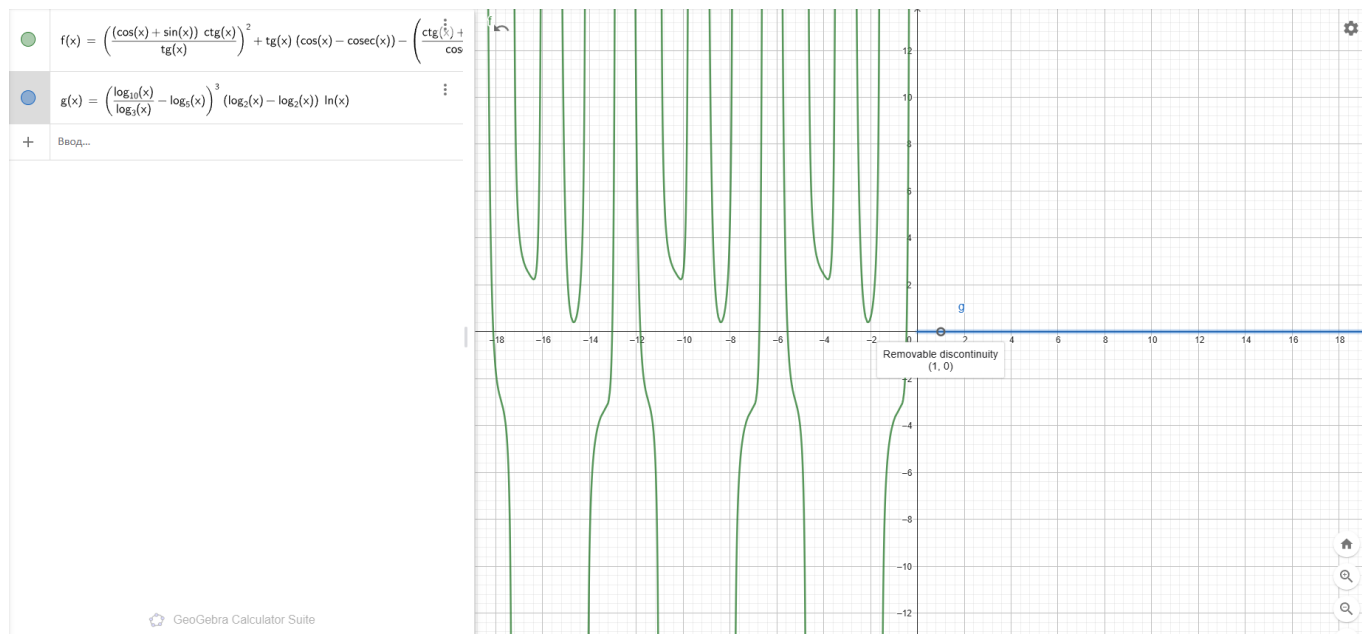


Рис. 4. График системы функций.


Анализ системы для $x \leq 0$: В знаменателе у нас $\tan(x)$, $\csc(x)$ и $\sec(x)$. Они равны 0 при $x = \frac{\pi n}{2}, n \in \mathbb{Z}$. Эти числа являются недействительными для данной функции.

Анализ системы для $x > 0$: В знаменателе у нас только $\log_3(x)$. Он равен 0 при $x = 1$. Это число является недействительным для данной функции.

Тестовое покрытие выбиралось следующим образом:

- тригонометрическая часть функции ведёт себя периодически, поэтому можно проверять её, рассмотрев лишь один период
- Взяты точки, в которых функция не существует
- Оставшаяся часть разбита на интервалы; для каждого интервала были взяты точки, близкие к точкам в пункте выше, а также точка между ними

CSV-выгрузки после тестирования



```
src > test > resources > dataset > cosTest.csv
1  0;1
2  1.5707963267948966;0
3  3.141592653589793;-1
4  4.71238898038469;0
5  6.283185307179586;1
6  -1.5707963267948966;0
7  -3.141592653589793;-1
8  -4.71238898038469;0
9  -6.283185307179586;1
10 0.5235987755982989;0.8660254037844386
11 1;0.5403023058681398
```

Рис. 5. Пример файла `target/test/dataset/cscTest.csv`.

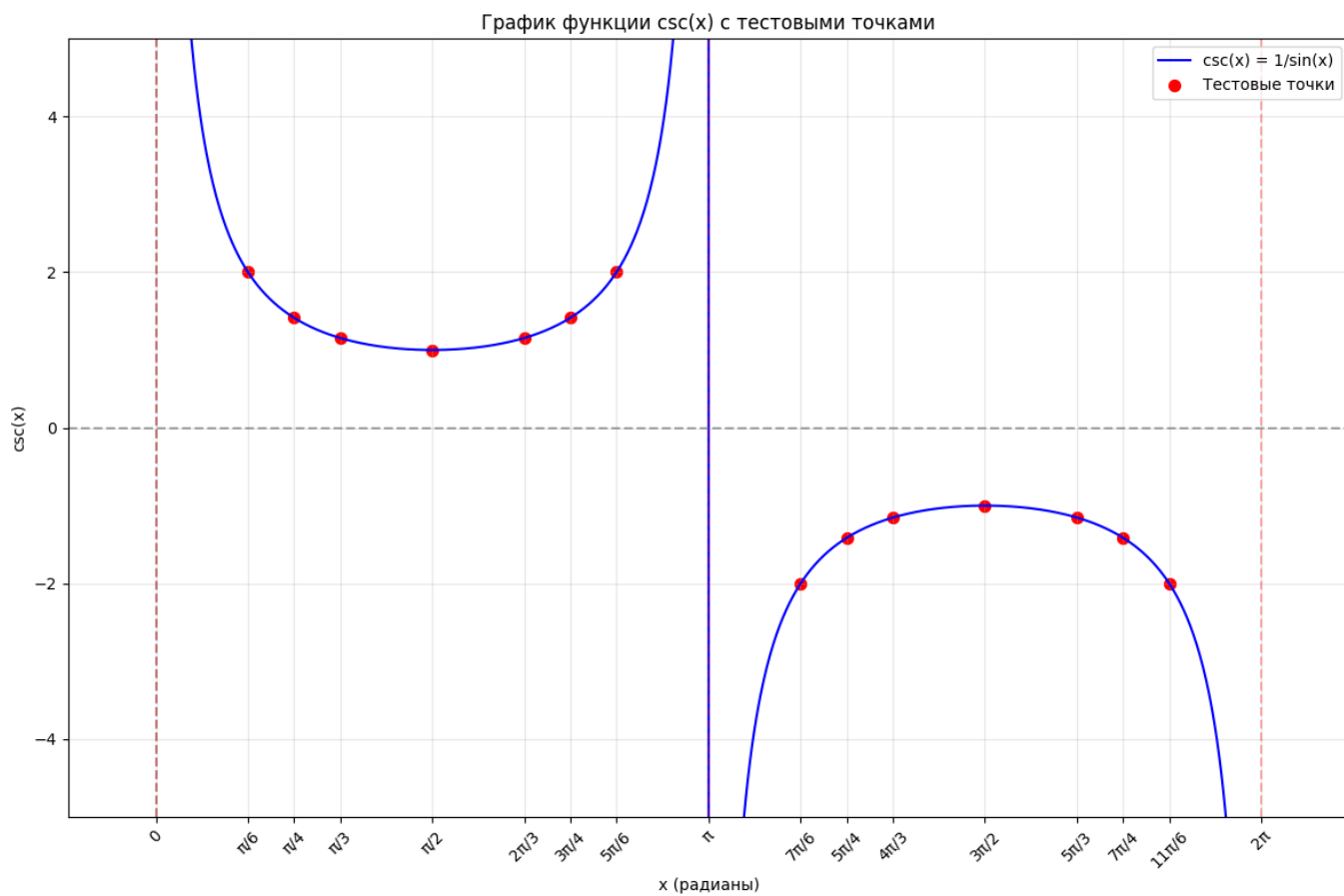


Рис. 6. Построенные на плоскости точки и график функции $\csc(x)$.

Вывод

В рамках данной работы была реализована система математических функций, состоящая из тригонометрических и логарифмических компонентов. Основные функции (косинус и натуральный логарифм) были реализованы через разложение в ряд Тейлора с заданной точностью.

Архитектура проекта построена на принципах модульности, где каждый модуль отвечает за определенную группу функций:

- *BaseFunctions* - базовые функции (\cos , \ln)
- *TrigonometricFunctions* - тригонометрические функции (\sin , \tan , \cot , \sec , \csc)
- *LogarithmicFunctions* - логарифмические функции (\log с различными основаниями)
- *SystemFunction* - интеграция всех функций в единую систему

Для каждого модуля были разработаны модульные и интеграционные тесты с использованием табличных данных. Тесты проверяют корректность работы функций на различных входных данных, включая граничные случаи и особые точки.

Система корректно обрабатывает области определения функций и выбрасывает исключения при недопустимых входных данных. Результаты вычислений сохраняются в CSV-файлы, что позволяет анализировать поведение функций и строить графики.

Для выполнения были использованы Junit и Mockito.