



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
“Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ
И КОМПЬЮТЕРНОЙ ТЕХНИКИ



ТПО.

0000-0?

000 046 000
002 051 012
010 000 020

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

“Тестирование программного обеспечения”

Вариант: 881.

ИАН ГАГАУЗ

ИАН МОНГОЛ

ПИИКТ
—
Итог семестра

Работу выполнил:
Студент группы Р3311
Болорболд Аригуун

Лектор:

Клименков Сергей Викторович

Практик:

Птицын Максим Евгеньевич



г. Санкт-Петербург
2025 г.

Содержимое

1 Текст задания

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

Функция:

$$f(x) = \begin{cases} (((((\tan(x) \cdot \cos(x)) + (\cot(x) + \sec(x))) + \csc(x)) \cdot \csc(x)) + \sec(x)) & \text{if } x \leq 0 \\ (((((\log_{10}(x) - \log_{10}(x)) \cdot (\log_2(x) - \log_3(x))) \cdot (\frac{\log_3(x) - \log_5(x)}{\log_{10}(x)})^3) + (\frac{\ln(x) \cdot \log_{10}(x)}{\log_2(x)})) & \text{if } x > 0 \end{cases} \quad (1)$$

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 csv файл вида «Х, Результаты модуля (Х)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

Порядок выполнения работы:

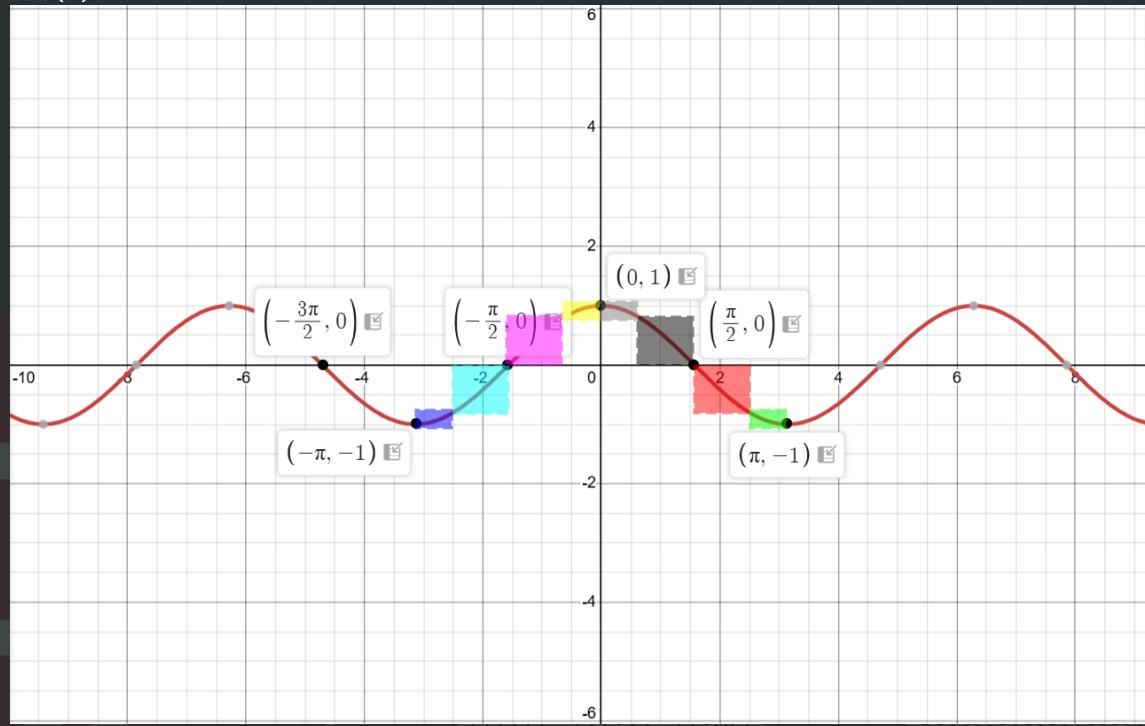
1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт <https://www.wolframalpha.com/>.
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.

2 Выполнение

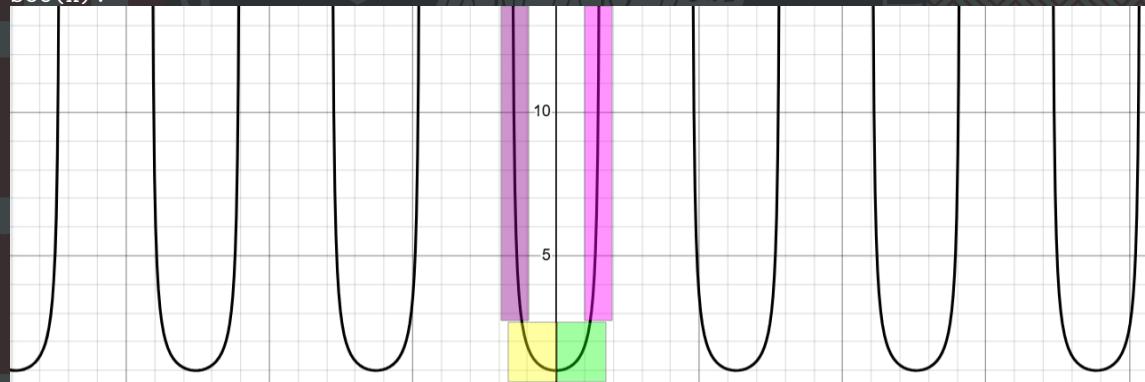
Ссылка на репозиторию: [GitHub](#)

2.1 Графики тригонометрических функций (с анализом эквивалентности)

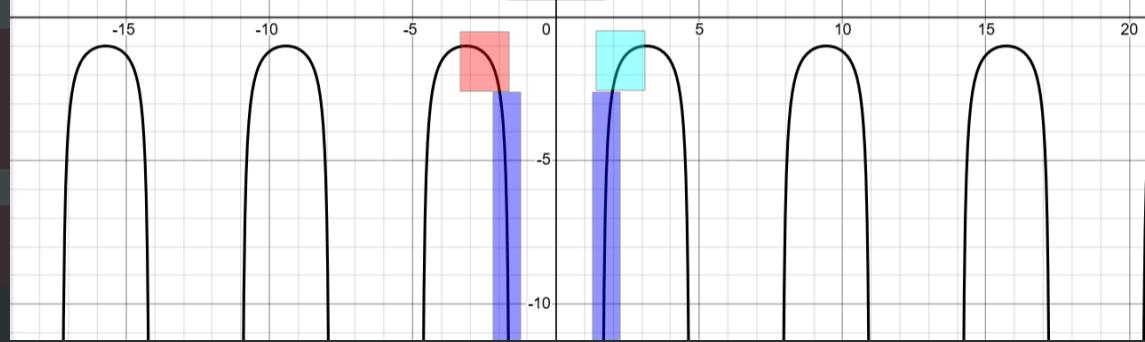
1. $\cos(x)$:

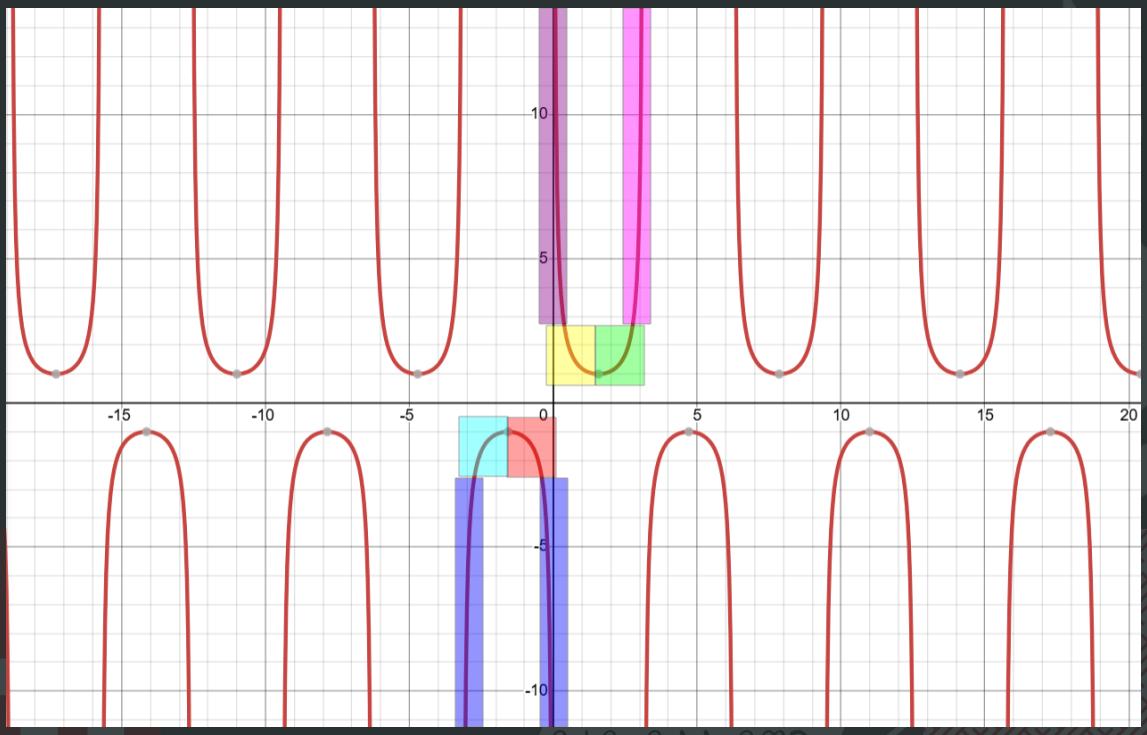


2. $\sec(x)$:

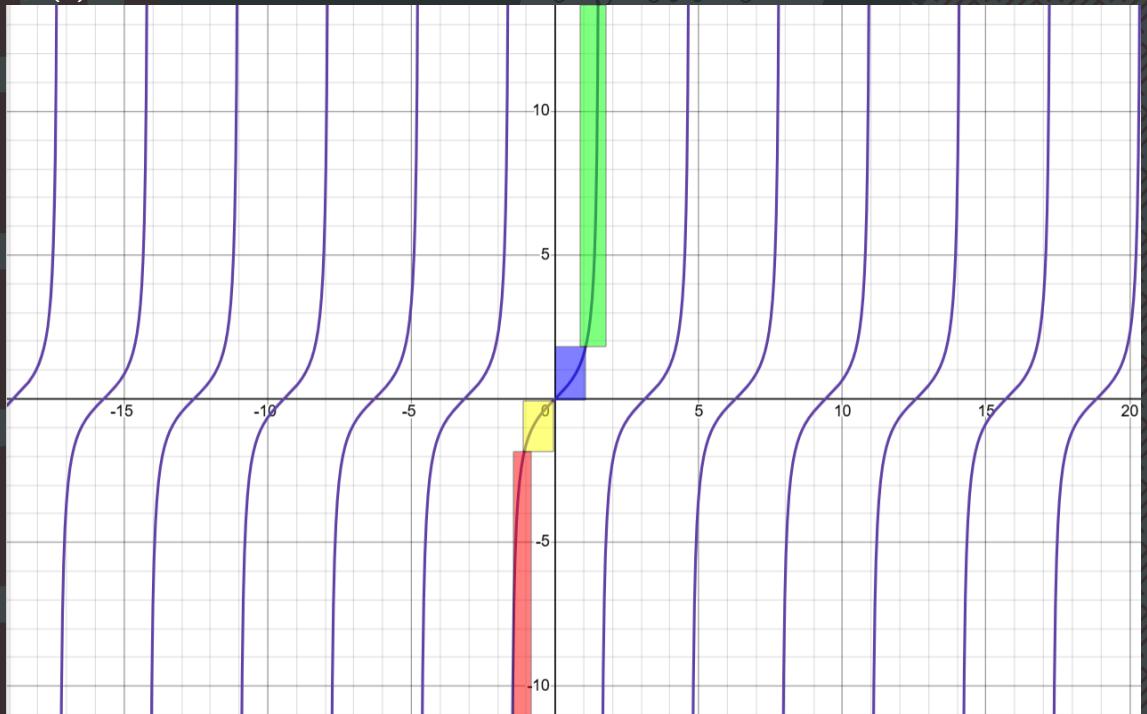


3. $\csc(x)$:

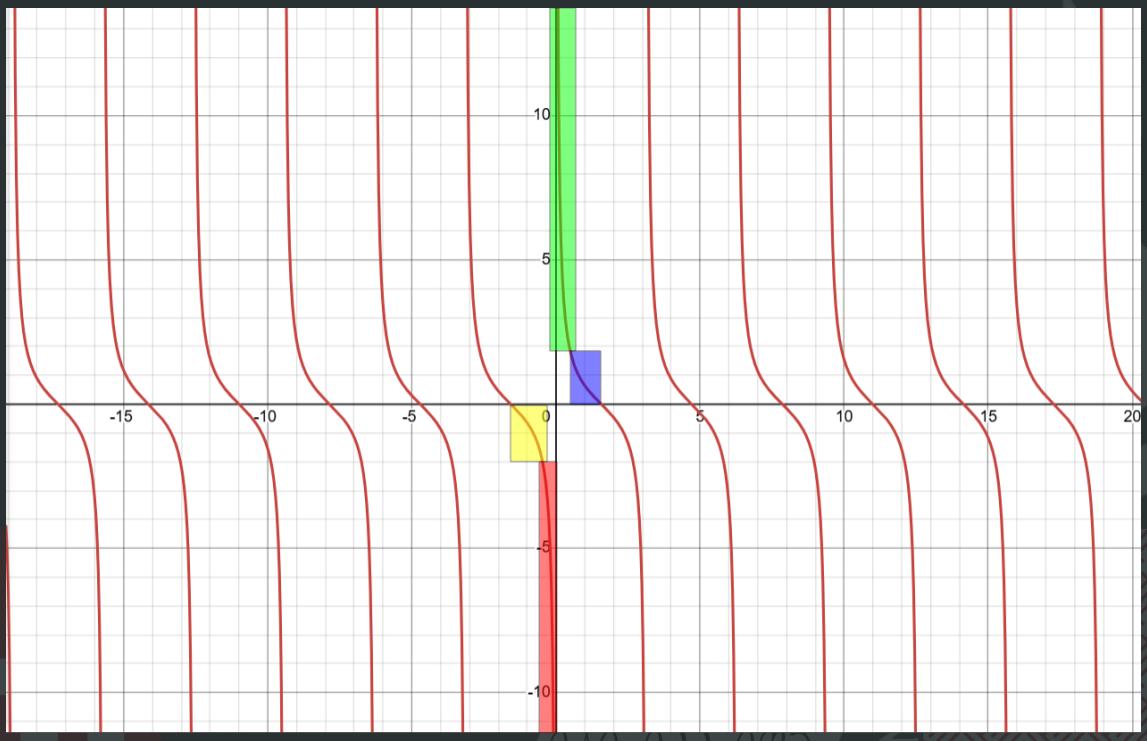




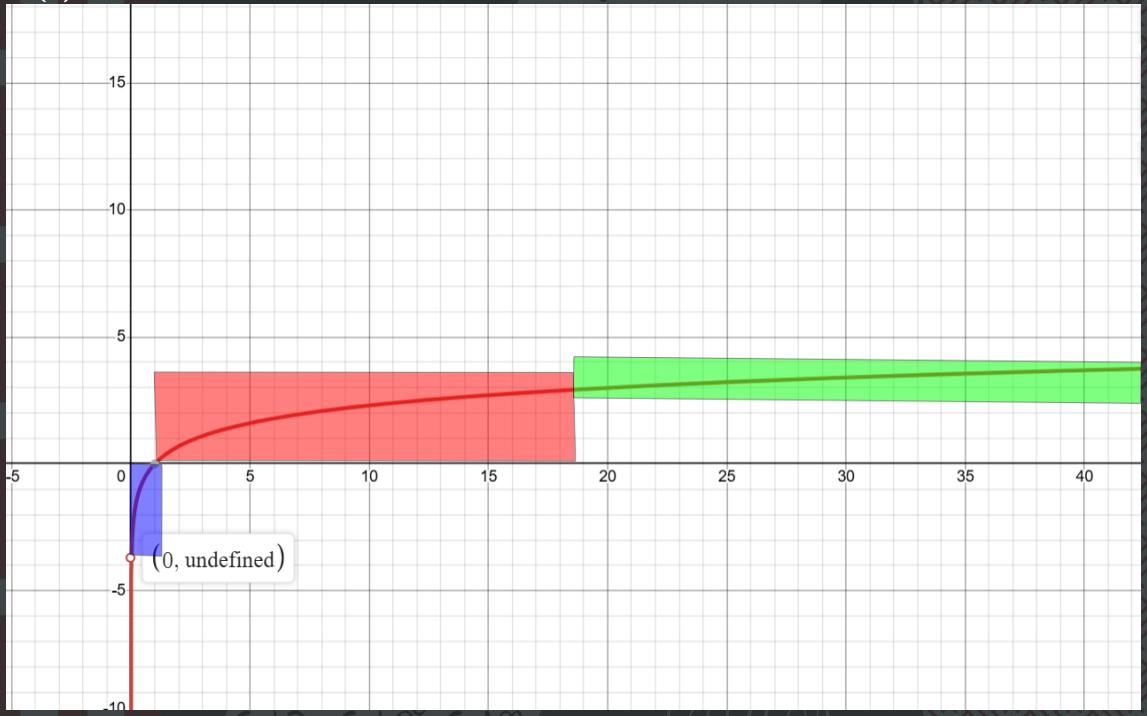
4. $\tan(x) :$



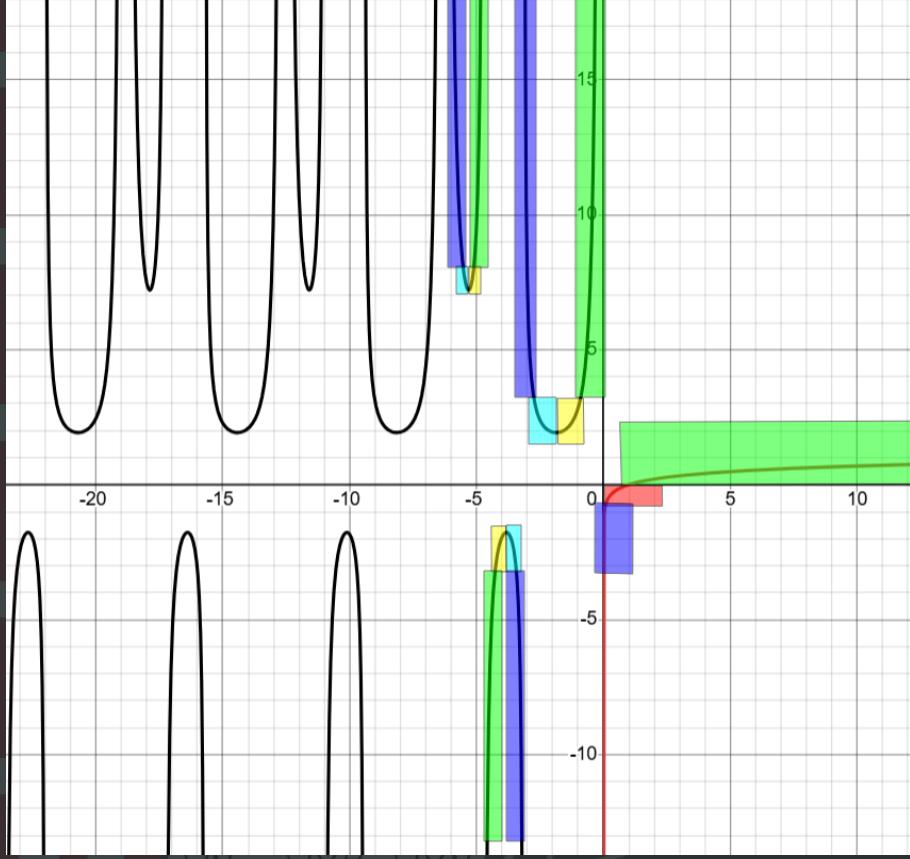
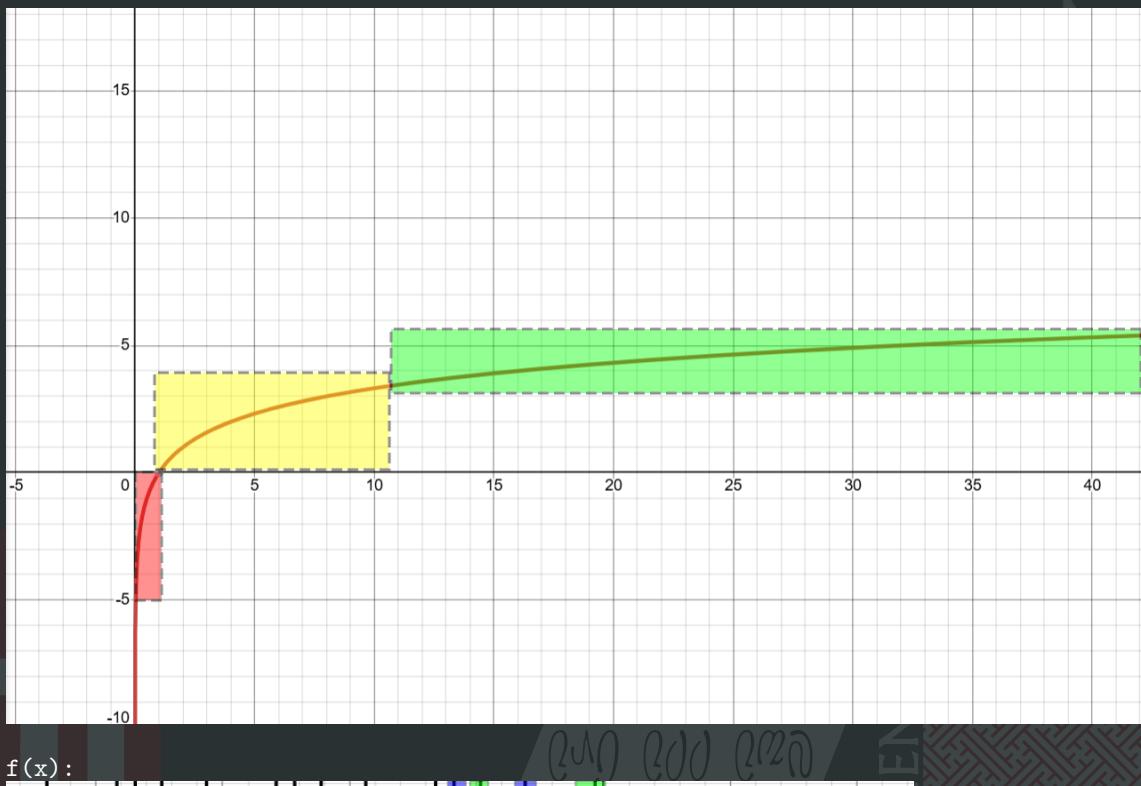
5. $\cot(x) :$



6. $\ln(x)$:



7. $\log(x)$:



Один из методов теста:

lab2/LogarithmIntegrationTest.java

```
@Test  
void shouldCallLn() {
```

```

        final BaseNLogarithm logarithm = new BaseNLogarithm(5, spyLn);
        logarithm.calculate(new BigDecimal(993), new BigDecimal("0.001"));
        verify(spyLn, atLeastOnce()).calculate(any(BigDecimal.class), any(BigDecimal.class));
    }

    @Test
    void shouldCalculateWithMockLn() {
        final BigDecimal arg = new BigDecimal(1488);
        when(mockLn.calculate(eq(new BigDecimal(1488)), any(BigDecimal.class))).thenReturn(new BigDecimal("4.5389687"));
        when(mockLn.calculate(eq(new BigDecimal(5)), any(BigDecimal.class))).thenReturn(new BigDecimal("1.648721270700127"));

        final BaseNLogarithm log5 = new BaseNLogarithm(5, mockLn);
        final BigDecimal expected = new BigDecimal("4.5389687");
        assertEquals(expected, log5.calculate(arg, PRECISION));
    }
}

```

3 Дополнительные задания.

- Использовать шпионы для создания частичной заглушки.
- Реализовать проверку тестового покрытия.

3.1 Реализация

`LogarithmIntegrationTest.java`

```

@Mock
private NaturalLogarithm mockLn;
@Spy
private NaturalLogarithm spyLn;

@Test
void shouldCallLn() {
    final BaseNLogarithm logarithm = new BaseNLogarithm(5, spyLn);
    logarithm.calculate(new BigDecimal(993), new BigDecimal("0.001"));
    verify(spyLn, atLeastOnce()).calculate(any(BigDecimal.class), any(BigDecimal.class));
}

```

Шпион зовет сам метод, в отличие от полной заглушки.

`build.gradle.kts`

```

jacoco {
    toolVersion = "0.8.12"
    reportsDirectory = layout.buildDirectory.dir("customJacocoReportDir")
}

tasks.test {
    extensions.configure(JacocoTaskExtension::class) {
        setDestinationFile(file(layout.buildDirectory.file("jacoco/jacoco.exec")))
    }
    finalizedBy(tasks.jacocoTestReport)
}

tasks.jacocoTestReport {
    dependsOn(tasks.test)
}

```

```

reports {
    xml.required = false
    csv.required = false
    html.required = true
    html.outputLocation = file(layout.buildDirectory.dir("reports/coverage"))
}
finalizedBy(tasks.jacocoTestCoverageVerification)
}

tasks.jacocoTestCoverageVerification {
    violationRules {
        rule {
            limit {
                minimum = "0.85".toBigDecimal()
            }
        }
    }
}

```

3.2 Результат

```

Run Lab2 [test] ×
Test Results 6 sec 995 ms Tests passed: 268 of 268 tests - 6 sec 995 ms
FunctionRuleTest 615 ms
shouldNotAcceptOut 210 ms
shouldNotAcceptNull 82 ms
shouldAcceptNull 84 ms
shouldAcceptArgAndl 65 ms
shouldCallTri 2 sec 163 ms
shouldCallLogFunc 53 ms
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
Starting Gradle Daemon...
Gradle Daemon started in 1 s 523 ms
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources UP-TO-DATE
> Task :testClasses UP-TO-DATE
> Task :test
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
Tests passed: 268 of 268 tests - 6 sec 995 ms

Тесты в "Lab2.test" ×
Test Results 1 sec 880 ms Tests passed: 4 of 4 tests - 1sec 880 ms
CSVGraphWriterTest 1 sec 880 ms
shouldCallFlush 1 sec 805 ms
shouldCreateFile 15 ms
shouldHandleArithmetc 24 ms
shouldWriteToFile 36 ms
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
> Task :test
> Task :jacocoTestReport
> Task :jacocoTestCoverageVerification FAILED
[ant:jacocoReport] Rule violated for bundle Lab2: instructions covered ratio is 0.11, but expected minimum is 0.85
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':jacocoTestCoverageVerification'.
* Rule violated for bundle Lab2: instructions covered ratio is 0.11, but expected minimum is 0.85
* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
> Get more help at https://help.gradle.org.
BUILD FAILED in 4s
6 actionable tasks: 4 executed, 2 up-to-date

Run Tests in "Lab2.test" ×
Test Results 0 sec 456 ms Tests passed: 268 of 268 tests - 0 sec 456 ms
The client will now receive all logging from the daemon (pid: 8756). The daemon log file: C:\Users\Admin\.gradle\daemon\8.8
Starting 61st build in daemon [uptime: 5 hrs 24 mins 6.255 secs, performance: 100%, GC rate: 0.08/s, heap usage: 0% of 512
Using 8 worker leases.
Now considering [D:\ITMO\Software Engineering\Software Testing\Lab2] as hierarchies to watch
Watching the file system is configured to be enabled if available
File system watching is active
Starting Build
Settings evaluated using settings file 'D:\ITMO\Software Engineering\Software Testing\Lab2\settings.gradle.kts'.
Projects loaded. Root project using build file 'D:\ITMO\Software Engineering\Software Testing\Lab2\build.gradle.kts'.
Included projects: [root project 'Lab2']
> Configure project :
Evaluating root project 'Lab2' using build file 'D:\ITMO\Software Engineering\Software Testing\Lab2\build.gradle.kts'.
All projects evaluated.
Task path ':test' matched project '':
Task name matched 'test'
Task path ':test' matched project '':
Task name matched 'test'
Selected primary task 'test' from project :

```

4 ВЫВОД

В рамке этой лабораторной работы я написал модульные тесты по нескольким доменам. Также я упражнял написание .yml файлов для настройки GitHub CI.