

Факультет программной инженерии и компьютерной техники

Тестирование программного обеспечения

Лабораторная работа №2

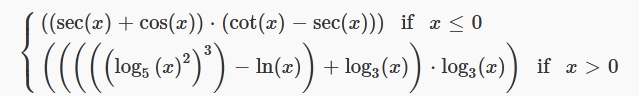
Вариант №33162

Преподаватель: Харитонова Анастасия Евгеньевна

Выполнил: Кульбако Артемий Юрьевич Р33112

# Задание

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



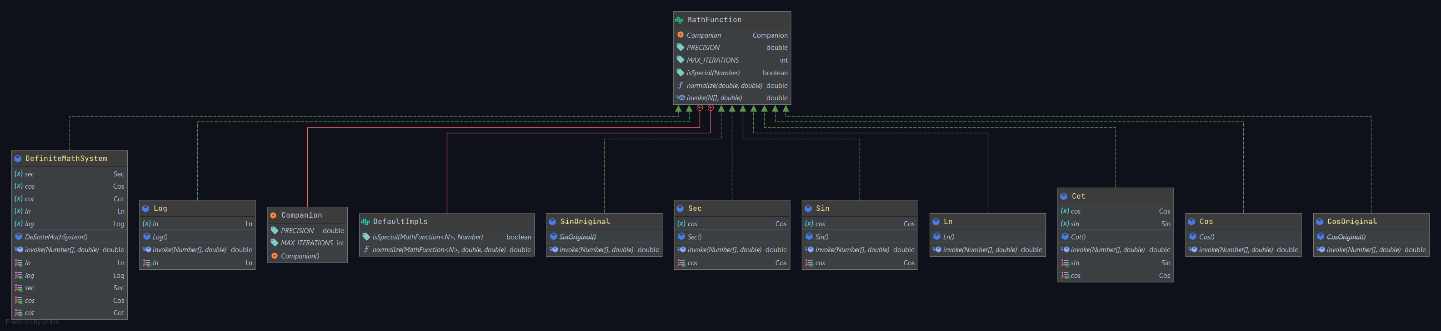
**Правила выполнения работы:**

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.

# Код

Код: <https://github.com/testpassword/Software-testing/tree/master/lab2-07.03.21/src>

# UML



# Модульное тестирование

Для модульный тестов были выбраны крайние точки периодичности функций, точки, где функция меняет знак на противоположный, а по точке с каждой стороны внутри периода функций, а также, если есть места, где она неопределённа, взято случайное значение из этого промежутка (функция должна вернуть NaN).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  |  |  |  | |
|  | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |
|  | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  |  |  |  |  |   Здесь ввиду того, что искомая функция из системы выбирается по условию , необходимо проверить, правильность выбора функции – взять , а на остальных промежутках можно функция будет вести себя одинаково. |

# Интеграционное тестирование

Для интеграционных тестов будем использовать те значения у вызывающих функций, которые у зависимых модулей будут также вызывать ожидаемые значения ( - приводит к вызову; – система функций). Таким образом, в выражении assert вызывающей функции мы будем знать, что true получено благодаря правильно отданному значению зависимой функции.

# Графики (построены по csv выгрузкам)

|  |  |
| --- | --- |
|  |  |
| S |  |
|  |  |

# Вывод

Главная сложность при проведении интеграционного тестирования – подбор возвращаемых значений у вызываемых модулей, ведь только тогда зависимые модули смогут сгенерировать правильное значение. Это требует глубокого понимания кода, с которым мы взаимодействуем, даже несмотря на то, что по факту сами пишем зависимые модули в виде заглушек. Это также порождает вторую проблему – написание большого количество кода для тестов (помочь с этим могут специальные фреймворки, в частности Mockito, если мы говорим про jvm).

.