НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Тестирование программного обеспечения

Лабораторная работа № 2

Вариант 55533

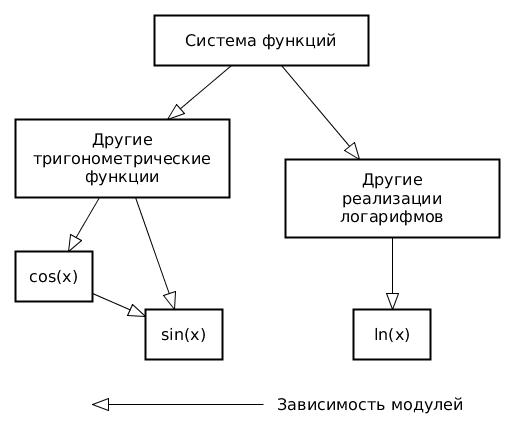
Выполнил студент:

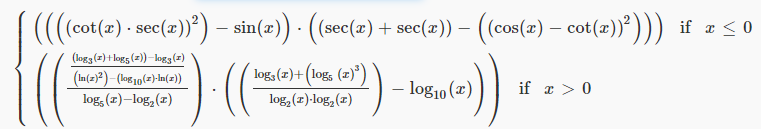
Щелыкалов Виктор

Группа № P33122

г. Санкт-Петербург, 2021

**Задание:**

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.



**Выполнение:**

Реализуем заглушки в виде модулей, возвращающих заведомо истинное значение.

Далее будем опираться на данное значение при сравнении. Теперь, определимся с тестовым покрытием. Для этого определим ОДЗ, выберем точки внутри ОДЗ для x>0 и x<=0. Добавим точки вне ОДЗ.

Тестовое покрытие:

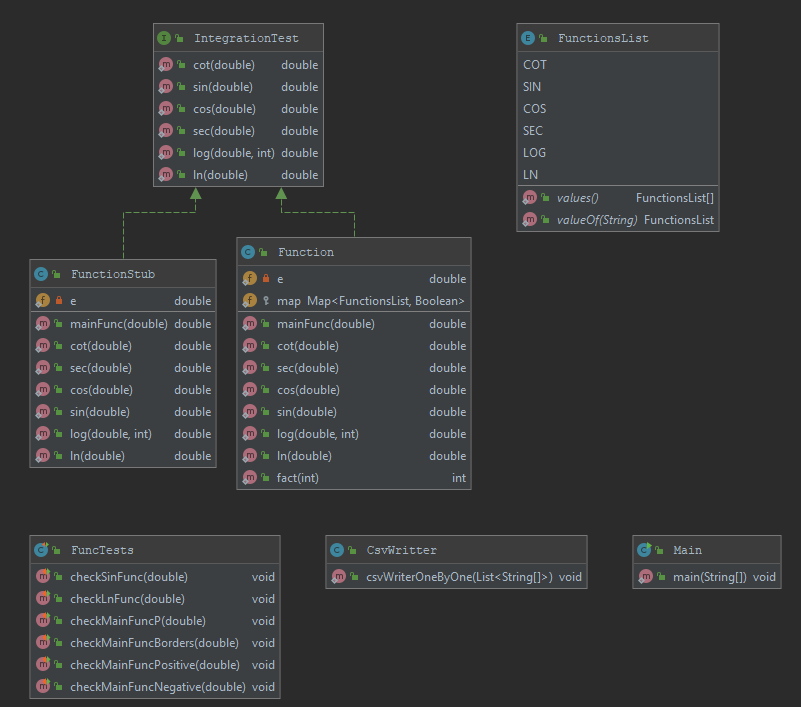
1. Внутри ОДЗ

* X = -1.21, -0.11, -0.45, -0.3111 – в части x <= 0
* X = 0.1, 0.331, 0.09, 0.8 – в части x > 0

1. Вне ОДЗ

* X = 1, 1.00001, 0.999999, PI, 3/2 \* PI.

Данного тестового покрытия достаточного, поскольку мы анализ эквивалентности показывает, что на функция ведет себя корректно на x<=0 и x >0, исключая точки вне ОДЗ. Отдельно проверим точки вне ОДЗ, убедимся в корректности возвращаемого значения.



Пример csv файла:

"-1.21","10.664251574133944"  
"-1.21","10.664251574133937"  
"-1.21","10.66425157413394"  
"-1.21","10.662757222404963"  
"-1.21","10.662757222404963"  
"-1.21","10.662757222404963"  
"-0.11","-8221.905716004045"  
"-0.11","-8221.905716004045"  
"-0.11","-8221.905716004045"  
"-0.11","-8221.944504230865"  
"-0.11","-8221.944504230865"  
"-0.11","-8221.944504230865"  
"-0.45","-37.774702387084744"  
"-0.45","-37.774702387084744"  
"-0.45","-37.774702387084744"  
"-0.45","-37.77442394768932"  
"-0.45","-37.77442394768932"  
"-0.45","-37.77442394768932"  
"-0.3111","-158.07944022880437"  
"-0.3111","-158.07944022880437"  
"-0.3111","-158.07944022880437"  
"-0.3111","-158.12853437824396"  
"-0.3111","-158.12853437824396"  
"-0.3111","-158.12853437824396"  
"-0.09","-17819.838488135865"  
"-0.09","-17819.838488135865"  
"-0.09","-17819.838488135865"  
"-0.09","-17819.8762873618"  
"-0.09","-17819.8762873618"  
"-0.09","-17819.8762873618"  
"1.0","NaN"  
"1.0","NaN"  
"1.0","NaN"  
"1.0","NaN"  
"1.0","NaN"  
"1.0","NaN"

Вывод: хорошая лабораторная работа, благодаря которой я изучил интеграционное тестирование. Долго выбирал формат заглушек для наших модулей. Отказался от библиотеки Mock и реализовал свой вариант. Больше всего разочаровало разложение функций в ряд, поскольку они лишили меня возможности провести полноценное тестирование из-за расходимости рядов. В остальном проблем не возникло.