

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

**Отчет по практической работе №2**

**по дисциплине «Тестирование и верификация программного обеспечения»**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент группы ИКБО-25-20 | |  | Хачатрян Э.К. |
| Принял преподаватель |  |  | Миронов А.И. |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2021 г. |  |  |

Зачтено «\_\_»\_\_\_\_\_\_\_2021 г.

Москва 2021

**Содержание**

[**Документация по проекту «Калькулятор площадей и периметров» от Хачатрян Э.К.:** 3](#_Toc114844336)

[**Документация по проекту «Тригонометрический калькулятор» от Карапетян А.А.** 5](#_Toc114844337)

[**Документация по проекту «Калькулятор» от Меличенков Е.А.** 7](#_Toc114844338)

[**Модульное тестирование** 10](#_Toc114844339)

[**Выводы** 15](#_Toc114844340)

[**Список использованных источников** 16](#_Toc114844341)

# **Документация по проекту «Калькулятор площадей и периметров» от Хачатрян Э.К.:**

Калькулятор Площадей и Периметров

* **Метод** \_\_init\_\_(self): - Конструктор класса
* **Метод** f\_s\_triangle(self, a, h): – считает площадь треугольника
* **Входные параметры :**

2 аргумента типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

* **Метод** f\_s\_par(self, a, h): – считает площадь параллелограмма.
* **Входные параметры :**

2 аргумента типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

* **Метод** f\_s\_trap(self, a, b, h): – считает площадь трапеции.
* **Входные параметры :**

2 аргумента типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

* **Метод** f\_s\_circle(self, r): – считает площадь круга.
* **Входные параметры :**

1 аргумент типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

* **Метод** f\_p\_circle(self, r): – считает периметр круга.
* **Входные параметры :**

1 аргумент типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

* **Метод** f\_p\_par(self, a, b): – считает периметр параллелограмма.
* **Входные параметры :**

2 аргумента типа float

* **Возвращаемые данные :**

Метод возвращает округлённую до 4 знаков после запятой переменную типа float

**Задание:**

Протестировать методы:

f\_s\_triangle(self, a, h):

f\_s\_par(self, a, h):

f\_s\_trap(self, a, b, h):

f\_s\_circle(self, r):

f\_p\_circle(self, r):

f\_p\_par(self, a, b):

# **Документация по проекту «Тригонометрический калькулятор» от Карапетян А.А.**

1. Документация по проекту Карапетян А.А.

Реализован проект «Тригонометрический калькулятор». Калькулятор посчитывает значение синуса, косинуса, тангенса, котангенса, арксинуса и арккосинуса для введенного значения, а также переводит значение угла из градусной меры в радианную меру.

1. Метод n\_rad(self, x\_rad) – переводит значение угла из градусной меры в радианную меру.

Входные данные: 1 аргумент типа float\int – значение угла в градусах.

Выходные данные: Метод возвращает переменную типа float – значение угла в радианах при условии, что тип входных данных соответствует float\int, иначе выводит сообщение "Ошибка, неверный тип данных".

2. Метод f\_sin(self, x) – метод нахождения синуса заданного угла.

Входные данные: 1 аргумент типа float\int – значение угла в радианах.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение синуса введенного угла при условии, что тип входных данных соответствует float\int, иначе выводит сообщение "Ошибка, неверный тип данных".

3. Метод f\_cos(self, x) – метод нахождения косинуса заданного угла.

Входные данные: 1 аргумент типа float\int – значение угла в радианах.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение косинуса введенного угла при условии, что тип входных данных соответствует float\int, иначе выводит сообщение "Ошибка, неверный тип данных".

4. Метод f\_tan(self, x) – метод нахождения тангенса заданного угла.

Входные данные: 1 аргумент типа float\int – значение угла в радианах.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение тангенса введенного угла при условии, что тип входных данных соответствует float\int и значение входных данных соответствует условию x = π/2 + π \* n, n ∈ Z, иначе выводит сообщение "Ошибка, неверный тип данных"/ "Ошибка, тангенса данного значения не существует" соответственно.

5. Метод f\_cot(self, x) – метод нахождения котангенса заданного угла.

Входные данные: 1 аргумент типа float\int – значение угла в радианах.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение котангенса введенного угла при условии, что тип входных данных соответствует float\int и значение входных данных соответствует условию x = π + π \* n, n ∈ Z, иначе выводит сообщение "Ошибка, неверный тип данных"/ "Ошибка, котангенса данного значения не существует" соответственно.

6. Метод f\_arcsin(self, x) – метод нахождения котангенса заданного угла.

Входные данные: 1 аргумент типа float\int – значение от -1 до 1.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение арксинуса введенного значения при условии, что тип входных данных соответствует float\int и значение входных данных соответствует условию x ∈ [-1; 1], иначе выводит сообщение "Ошибка, неверный тип данных"/ "Ошибка, арксинуса данного значения не существует" соответственно.

7. Метод f\_arccos(self, x) – метод нахождения котангенса заданного угла.

Входные данные: 1 аргумент типа float\int – значение от –1 до 1.

Выходные данные: Метод возвращает округленную до 4 знаков после запятой переменную типа float – значение арккосинуса введенного значения при условии, что тип входных данных соответствует float\int и значение входных данных соответствует условию x ∈ [-1; 1], иначе выводит сообщение "Ошибка, неверный тип данных"/ "Ошибка, арккосинуса данного значения не существует" соответственно.

# **Документация по проекту «Калькулятор» от Меличенкова Е.А.**

Документация

Калькулятор

**Задание:** Протестировать все классы и методы конвертера валют на Python

* **Метод** poww(self,x,y) – Метод возводит x в степень y
* **Входные параметры :** 
  + X: число возведенное в степень
  + Y: степень числа
* **Возвращаемые данные :**
  + Метод возвращает полученное число после возведения
* **Метод** logarithm(self, N, x) – Метод считает логарифм
* **Входные параметры :**
* N: число логарифма
* x: основание логарифма
* **Возвращаемые данные :**
  + Метод возвращает полученное число после
* **Метод** sqrtt(x) – Метод извлекает корень
* **Входные параметры :**
  + x: число, извлекает в корень
* **Возвращаемые данные :**
  + Метод возвращает полученное число после извлечения корня
* **Метод** factorial(x) **–** Метод считает факториал
* **Входные параметры :**
  + x: число, которое мы будем считать
* **Возвращаемые данные :**
  + - Метод возвращает переменную double
* **Метод** sum(x,y) **–** Метод считает сумму 2-х чисел
  + **Входные параметры :**
    - x: первое слагаемое
    - y: второе слагаемое
* **Возвращаемые данные :**
  + - Метод возвращает переменную double
* **Метод** minus(x,y) **–** Метод считает вычитание 2-х чисел
  + **Входные параметры :**
    - x: первое слагаемое
    - y: второе слагаемое
* **Возвращаемые данные :**
  + - Метод возвращает переменную double
* **Метод** multy(x,y) **–** Метод считает умножение 2-х чисел
  + **Входные параметры :**
    - x: первое слагаемое
    - y: второе слагаемое
* **Возвращаемые данные :**
  + - Метод возвращает переменную double
* **Метод** division(x,y) **–** Метод считает деление 2-х чисел
  + **Входные параметры :**
    - x: первое слагаемое
    - y: второе слагаемое
* **Возвращаемые данные :**
  + - Метод возвращает переменную double

# **Модульное тестирование**

Модульные тесты приложения Карапетяна А.А. были произведены Хачатряном Э.К.:

*Листинг 1 – модульное тестирование кода Карапетян А.А.*

*import math  
import unittest  
from calculatorr import Calculator  
  
  
class TestCalculator(unittest.TestCase):  
 def setUp(self):  
 self.calculator = Calculator()  
  
 def test\_n\_rad(self):  
 self.assertEqual(self.calculator.n\_rad('string'), "Ошибка, неверный тип данных")  
  
 def test\_sin\_1(self):  
 self.assertEqual(self.calculator.f\_sin(30), 0.5000)  
  
 def test\_sin\_2(self):  
 self.assertEqual(self.calculator.f\_sin('string'), "Ошибка, неверный тип данных")  
  
 def test\_cos(self):  
 self.assertEqual(self.calculator.f\_cos('string'), "Ошибка, неверный тип данных")  
  
 def test\_tan(self):  
 self.assertEqual(self.calculator.f\_tan(90), "Ошибка, тангенса данного значения не существует")  
  
 def test\_cot(self):  
 self.assertEqual(self.calculator.f\_cot(180), "Ошибка, котангенса данного значения не существует")  
  
 def test\_arcsin(self):  
 self.assertEqual(self.calculator.f\_arcsin(2), "Ошибка, арксинуса данного значения не существует")  
  
 def test\_arccos(self):  
 self.assertEqual(self.calculator.f\_arccos(-2), "Ошибка, арккосинуса данного значения не существует")  
  
if \_\_name\_\_=="\_\_main\_\_":*

*Таблица 1 – результаты модульного тестирования кода Карапетян А.А.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название метода** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** | **Соответствие результатов** |
| n\_rad | ‘string | "Ошибка, неверный тип данных" | "Ошибка, неверный тип данных" | Good |
| f\_sin\_1 | 30 | 0.5000 | 0.5 | Good |
| f\_sin\_2 | ‘string’ | "Ошибка, неверный тип данных" | "Ошибка, неверный тип данных" | Good |
| f\_cos | ‘string’ | "Ошибка, неверный тип данных" | "Ошибка, неверный тип данных" | Good |
| f\_tan | 90 | "Ошибка, тангенса данного значения не существует" | "Ошибка, тангенса данного значения не существует" | Good |
| f\_cot | 180 | "Ошибка, котангенса данного значения не существует" | "Ошибка, котангенса данного значения не существует" | Good |
| f\_arcsin | 2 | "Ошибка, арксинуса данного значения не существует" | "Ошибка, арксинуса данного значения не существует" | Good |
| f\_arccos | -2 | "Ошибка, арккосинуса данного значения не существует" | "Ошибка, арккосинуса данного значения не существует" | Good |

Модульные тесты приложения Меличенкова Е.А. были произведены Карапетяном А.А.:

*Листинг 2 – модульное тестирование кода Меличенков Е.А.*

*def* test\_poww(*self*):  
 *self*.assertEqual(*self*.calculator.poww(1, 1), 2)  
  
*def* test\_logarithm(*self*):  
 *self*.assertEqual(*self*.calculator.logarithm(0, 2), 1)  
  
*def* test\_sqrtt(*self*):  
 *self*.assertEqual(*self*.calculator.sqrtt(-2), "Корень из отрицательного числа нельзя брать")  
  
*def* test\_factorial(*self*):  
 *self*.assertEqual(*self*.calculator.factorial(-2), "Факториал нельзя брать от отрицательного числа” )

def test\_sum(self):

self.assertEqual(self.calculator.sum(1, 1), 2)

def test\_minus(self):

self.assertEqual(self.calculator.minus(1, 1), 2)  
  
*def* test\_multy(*self*):  
 *self*.assertEqual(*self*.calculator.multy(2, ‘2’), " Ошибка введенных данных ")  
  
  
*def* test\_division(*self*):  
 *self*.assertEqual(*self*.calculator.division(1, 0), "Делить на ноль нельзя")

*Таблица 2 – Результаты модульного тестирования кода Меличенков Е.А.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название метода** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** | **Соответствие результатов** |
| poww | 1, 1 | 2 | 2 | ok |
| logarithm | 0, 2 | 1 | 1 | ok |
| sqrtt | -2 | "Корень из отрицательного числа нельзя брать" | "Корень из отрицательного числа нельзя брать" | ok |
| Factorial | -2 | "Факториал нельзя брать от отрицательного числа” | "Факториал нельзя брать от отрицательного числа” | ok |
| Sum | 1, 1 | 2 | 2 | ok |
| Minus | 1, -1 | 2 | 2 | ok |
| Multy | 2, ‘2’ | “Ошибка введенных данных” | error | error |
| Division | 1, 0 | "Делить на ноль нельзя" | "Делить на ноль нельзя" | ok |

Модульные тесты приложения Хачатряна Э.К. были произведены Меличенковым Е.А.:

От Карапетян А.А.:

*Листинг 3 – модульные тесты от Карапетян А.А.*

import unittest  
  
  
from main import Calculator  
#Test cases to test Calulator methods  
#You always create a child class derived from unittest.TestCase  
class TestCalculator(unittest.TestCase):  
 #setUp method is overridden from the parent class TestCase  
 def setUp(self):  
 self.calculator = Calculator()  
 #Each test method starts with the keyword test\_  
 def test\_f\_s\_triangle(self):  
 self.assertEqual(self.calculator.f\_s\_triangle(1,7), 3.5)  
 def test\_f\_s\_par(self):  
 self.assertEqual(self.calculator.f\_s\_par(10,-5), "Параметры не могут быть меньше или равны нулю")  
 def test\_f\_s\_trap(self):  
 self.assertEqual(self.calculator.f\_s\_trap(3,'1',7), "Некорректный ввод данных")  
 def test\_f\_s\_circle(self):  
 self.assertEqual(self.calculator.f\_s\_circle(0), "Параметры не могут быть меньше или равны нулю")  
 def test\_f\_p\_circle(self):  
 self.assertEqual(self.calculator.f\_p\_circle('b'), "Некорректный ввод данных")  
 def test\_f\_p\_par(self):  
 self.assertEqual(self.calculator.f\_p\_par('125',5), "Некорректный ввод данных")  
  
# Executing the tests in the above test case class  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

*Таблица 3 – результаты модульного тестирования кода Хачатряна Э.К.. от Карапетян А.А.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название метода** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** | **Соответствие результатов** |
| f\_s\_triangle | 1, 7 | 3.5 | 3.5 | Ok |
| f\_s\_par | 10,-5 | "Параметры не могут быть меньше или равны нулю" | "Параметры не могут быть меньше или равны нулю" | Ok |
| f\_s\_trap | 3,'1',7 | "Некорректный ввод данных" | "Некорректный ввод данных" | Ok |
| f\_s\_circle | 0 | Параметры не могут быть меньше или равны нулю" | Параметры не могут быть меньше или равны нулю" | Ok |
| f\_p\_circle | 'b' | "Некорректный ввод данных" | "Некорректный ввод данных" | Ok |
| f\_p\_par | '125',5 | "Некорректный ввод данных" | "Некорректный ввод данных" | Ok |

От Меличенков Е.А.:

*Листинг 4 – модульные тесты от Меличенков Е.А.*

import unittest  
  
  
from main import Calculator  
#Test cases to test Calulator methods  
#You always create a child class derived from unittest.TestCase  
class TestCalculator(unittest.TestCase):  
 #setUp method is overridden from the parent class TestCase  
 def setUp(self):  
 self.calculator = Calculator()  
 #Each test method starts with the keyword test\_  
 def test\_f\_s\_triangle(self):  
 self.assertEqual(self.calculator.f\_s\_triangle(2,7), 7)  
 def test\_f\_s\_par(self):  
 self.assertEqual(self.calculator.f\_s\_par(10,'10'), "Некорректный ввод данных")  
 def test\_f\_s\_trap(self):  
 self.assertEqual(self.calculator.f\_s\_trap(3, -1, 7), "Параметры не могут быть меньше или равны нулю")  
 def test\_f\_s\_circle(self):  
 self.assertEqual(self.calculator.f\_s\_circle(0), "Параметры не могут быть меньше или равны нулю")  
 def test\_f\_p\_circle(self):  
 self.assertEqual(self.calculator.f\_p\_circle('z'), "Некорректный ввод данных")  
 def test\_f\_p\_par(self):  
 self.assertEqual(self.calculator.f\_p\_par(-7,5), "Параметры не могут быть меньше или равны нулю")  
  
# Executing the tests in the above test case class  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

*Таблица 3 – результаты модульного тестирования кода Хачатряна Э.К.. от Меличенков Е.А.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название метода** | **Входные данные** | **Ожидаемый результат** | **Реальный результат** | **Соответствие результатов** |
| f\_s\_triangle | 2,7 | 7 | 7 | Ok |
| f\_s\_par | 10,'10' | "Некорректный ввод данных" | "Некорректный ввод данных" | Ok |
| f\_s\_trap | 3, -1, 7 | "Параметры не могут быть меньше или равны нулю" | "Параметры не могут быть меньше или равны нулю" | Ok |
| f\_s\_circle | 0 | "Параметры не могут быть меньше или равны нулю" | "Параметры не могут быть меньше или равны нулю" | Ok |
| f\_p\_circle | 'z' | "Некорректный ввод данных" | "Некорректный ввод данных" | Ok |
| f\_p\_par | -7,5 | "Параметры не могут быть меньше или равны нулю" | "Параметры не могут быть меньше или равны нулю" | Ok |

# **Выводы**

Оценка качества чужого кода: Карапетян А.А. (5), Меличенков Е.А. (4)

Соответствие документации: Карапетян А.А. (5), Меличенков Е.А. (5)

Надежность кода: Карапетян А.А. (5), Меличенков Е.А. (4)

# **Список использованных источников**

1. Рекс Блек. Ключевые процессы тестирования - М.: Издательство Лори, 2014. - 544 с.
2. Тестирование в Python[unittest] URL: https://devpractice.ru/unit-testing-in-python-part-1/ (Дата обращения 05.10.21)
3. Виды Тестирования [Электронный ресурс]/ Про Тестинг - Тестирование Программного Обеспечения. URL: http://www.protesting.ru/testing/types/sanity.html (Дата обращения 05.10.21)
4. Винниченко И.В. Автоматизация процессов тестирования. Издательство Питер, 2005.
5. Гребенюк В. М. Oценка целесообразности внедрения автоматизированного тестирования. Институт Государственного управления, права и инновационных технологий (ИГУПИТ). Интернет-журнал «НАУКОВЕДЕНИЕ» №1 2013.