Лабораторная работа №6:

Код программы 1:

1.py > ...

```python
import pytest
from calc import calculate, convert_precision

def test_convert_precision():
    assert convert_precision(0) == 0
    assert convert_precision(1) == 0
    assert convert_precision(0.1) == 1
    assert convert_precision(0.01) == 2
    assert convert_precision(0.000001) == 6

def test_addition():
    assert calculate('+', 1, 2) == 3
    assert calculate('+', 10, 20) == 30
    assert calculate('+', -1, 1) == 0

def test_subtraction():
    assert calculate('-', 5, 3) == 2
    assert calculate('-', 10, 20) == -10
    assert calculate('-', -1, 1) == -2

def test_multiplication():
    assert calculate('*', 2, 3) == 6
    assert calculate('*', 0, 5) == 0
    assert calculate('*', -2, 3) == -6

def test_division():
    assert calculate('/', 15, 3) == 5
    assert calculate('/', 6, 3) == 2
    assert calculate('/', 10, 2) == 5

@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 3),
    ([-1, 0, 1], 0),
])
def test_medium(values, expected):
    assert calculate('medium', *values) == pytest.approx(expected)

@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 2.0),
    ([10, 11, 3, 4, 5], 10.64),
])
```

```python
def test_variance(values, expected):
    assert calculate('variance', *values) == pytest.approx(expected)

@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 1.414214),
    ([-1, 0, 1], 0.816497),
])
def test_std_deviation(values, expected):
    assert calculate('std_deviation', *values) == pytest.approx(expected)

def test_median():
    assert calculate('median', [1, 3, 5]) == 3
    assert calculate('median', [-1, 0, 1]) == 0

def test_q2():
    assert calculate('q2', [1, 3, 5]) == 3
    assert calculate('q2', [-1, 0, 1]) == 0

def test_q3():
    assert calculate('q3', [1, 3, 5]) == pytest.approx(4.0)
    assert calculate('q3', [-1, 0, 1]) == pytest.approx(0.5)

# Запуск тестов
if __name__ == '__main__':
    pytest.main([__file__, "-v"])
```

Результат:

```
23.0
5.0
27.0
5.0
============================= test session starts ==============================
platform win32 -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- C:\Users\sambu\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена Алдар\II курс\Программирование Python, 2 курс\ЛР-6
collected 14 items

1.py::test_convert_precision PASSED                                      [  7%]
1.py::test_addition PASSED                                               [ 14%]
1.py::test_subtraction PASSED                                            [ 21%]
1.py::test_multiplication PASSED                                         [ 28%]
1.py::test_division PASSED                                               [ 35%]
1.py::test_medium[values0-3] PASSED                                      [ 42%]
1.py::test_medium[values1-0] PASSED                                      [ 50%]
1.py::test_variance[values0-2.0] PASSED                                  [ 57%]
1.py::test_variance[values1-10.64] PASSED                                [ 64%]
1.py::test_std_deviation[values0-1.414214] PASSED                        [ 71%]
1.py::test_std_deviation[values1-0.816497] PASSED                        [ 78%]
1.py::test_median PASSED                                                 [ 85%]
1.py::test_q2 PASSED                                                     [ 92%]
1.py::test_q3 PASSED                                                     [100%]

============================== 14 passed in 0.03s ==============================
```

Код программы 2:

2.py > test_q3

```python
import pytest
from calc import calculate, convert_precision

@pytest.mark.parametrize("input_value, expected_output", [
    (0, 0),
    (1, 0),
    (0.1, 1),
    (0.01, 2),
    (0.000001, 6),
])
def test_convert_precision(input_value, expected_output):
    assert convert_precision(input_value) == expected_output

def test_addition():
    assert calculate('+', 1, 2) == 3
    assert calculate('+', 10, 20) == 30
    assert calculate('+', -1, 1) == 0

def test_subtraction():
    assert calculate('-', 5, 3) == 2
    assert calculate('-', 10, 20) == -10
    assert calculate('-', -1, 1) == -2

def test_multiplication():
    assert calculate('*', 2, 3) == 6
    assert calculate('*', 0, 5) == 0
    assert calculate('*', -2, 3) == -6

def test_division():
    assert calculate('/', 6, 3) == 2
    assert calculate('/', 10, 2) == 5
    assert calculate('/', 0, 5) == 0

@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 3),
    ([-1, 0, 1], 0),
])
def test_medium(values, expected):
    assert calculate('medium', *values) == expected
```

```python
@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 2),
    ([-1, 0, 1], 0.6666666666666666),
])
def test_variance(values, expected):
    assert calculate('variance', *values) == pytest.approx(expected)

@pytest.mark.parametrize("values, expected", [
    ([1, 2, 3, 4, 5], 1.414214),
    ([-1, 0, 1], 0.816497),
])
def test_std_deviation(values, expected):
    assert calculate('std_deviation', *values) == pytest.approx(expected)

@pytest.mark.parametrize("values, expected", [
    ([1, 3, 5], 3),
    ([-1, 0, 1], 0),
])
def test_median(values, expected):
    assert calculate('median', *values) == expected

@pytest.mark.parametrize("values, expected", [
    ([1, 3, 5], 3),
    ([-1, 0, 1], 0),
])
def test_q2(values, expected):
    assert calculate('q2', *values) == expected

@pytest.mark.parametrize("values, expected", [
    ([1, 3, 5], 4),
    ([-1, 0, 1], 0.5),
])
def test_q3(values, expected):
    assert calculate('q3', *values) == expected
```

Результат:

```
23.0
5.0
27.0
5.0
```

Код программы calc:

calc.py > ...

```python
import math
import unittest
import numpy as np

def convert_precision(tolerance):
    if tolerance == 0:
        return 0
    try:
        return abs(math.floor(math.log10(abs(tolerance))))
    except Exception as e:
        print(f"Ошибка при конверсии точности: {e}")
        return 0

def calculate(action, *args, tolerance=1e-6):
    if action not in ['+', '-', '*', '/', 'medium', \
            'variance', 'std_deviation', 'median', 'q2', 'q3']:
        raise ValueError("Неверная операция")

    try:
        if (action == '+'):
            result = sum(args)
        elif (action == '-'):
            result = args[0]
            for i in range(1, len(args)):
                result -= args[i]
        elif (action == '*'):
            result = 1
            for i in range(len(args)):
                result *= args[i]
        elif (action == '/'):
            result = args[0]
            for i in range(1, len(args)):
                result /= args[i]

        elif (action == 'medium'):
            result = np.mean(args)
        elif (action == 'variance'):
            result = np.var(args)
        elif (action == 'std_deviation'):
            result = np.std(args)
        elif (action == 'median'):
            result = np.median(args)
```

```
        elif (action == 'q2'):
            result = np.median(args)
        elif (action == 'q3'):
            result = np.percentile(args, 75)
        else:
            raise ValueError("Неверная операция")

        precision = convert_precision(tolerance)
        rounded_result = round(result, precision)

        return rounded_result
    except Exception as e:
        print(f"Ошибка при вычислении: {e}")
        raise
```

(class) BatchCalculatorContextManager

```
class BatchCalculatorContextManager:
    def __init__(self, file):
        self.filename = file
        self.file = None
        self.lines = None
        self.line_count = None

    def __enter__(self):
        try:
            self.file = open(self.filename, "r")
            if not self.file:
                raise IOError("Не удалось открыть файл")
            self.lines = self.file.readlines()
            self.line_count = len(self.lines)
            return self
        except Exception as e:
            print(f"Ошибка при открытии файла или чтении строк: {e}")
            raise

    def __exit__(self, exc_type, exc_val, exc_tb):
        try:
            if self.file:
                self.file.close()
        except Exception as e:
            print(f"Ошибка при закрытии файла: {e}")

    def perform_calculation(self):
        for line in self.lines:
            try:
                a, op, b = line.split()
                yield calculate(op, float(a), float(b))
            except ValueError as ve:
                print(f"Неверное выражение в строке: {ve}")
            except Exception as e:
                print(f"Ошибка при обработке строки: {e}")

try:
    with BatchCalculatorContextManager('file.txt') as calc:
        for i in calc.perform_calculation():
            print(i)
except IOError as io_error:
    print(f"Ошибка при работе с файлом: {io_error}")
except Exception as general_exception:
    print(f"Неожиданная ошибка: {general_exception}")
```