

Лабораторная работа №6:

Код программы 1:

```
1.py > ...
1  import pytest
2  from calc import calculate, convert_precision
3
4  def test_convert_precision():
5      assert convert_precision(0) == 0
6      assert convert_precision(1) == 0
7      assert convert_precision(0.1) == 1
8      assert convert_precision(0.01) == 2
9      assert convert_precision(0.000001) == 6
10
11 def test_addition():
12     assert calculate('+', 1, 2) == 3
13     assert calculate('+', 10, 20) == 30
14     assert calculate('+', -1, 1) == 0
15
16 def test_subtraction():
17     assert calculate('-', 5, 3) == 2
18     assert calculate('-', 10, 20) == -10
19     assert calculate('-', -1, 1) == -2
20
21 def test_multiplication():
22     assert calculate('*', 2, 3) == 6
23     assert calculate('*', 0, 5) == 0
24     assert calculate('*', -2, 3) == -6
25
26 def test_division():
27     assert calculate('/', 15, 3) == 5
28     assert calculate('/', 6, 3) == 2
29     assert calculate('/', 10, 2) == 5
30
31 @pytest.mark.parametrize("values, expected", [
32     ([1, 2, 3, 4, 5], 3),
33     ([-1, 0, 1], 0),
34 ])
35 def test_medium(values, expected):
36     assert calculate('medium', *values) == pytest.approx(expected)
37
38 @pytest.mark.parametrize("values, expected", [
39     ([1, 2, 3, 4, 5], 2.0),
40     ([10, 11, 3, 4, 5], 10.64),
41 ])
```

```

42 def test_variance(values, expected):
43     assert calculate('variance', *values) == pytest.approx(expected)
44
45 @pytest.mark.parametrize("values, expected", [
46     ([1, 2, 3, 4, 5], 1.414214),
47     ([-1, 0, 1], 0.816497),
48 ])
49 def test_std_deviation(values, expected):
50     assert calculate('std_deviation', *values) == pytest.approx(expected)
51
52 def test_median():
53     assert calculate('median', [1, 3, 5]) == 3
54     assert calculate('median', [-1, 0, 1]) == 0
55
56 def test_q2():
57     assert calculate('q2', [1, 3, 5]) == 3
58     assert calculate('q2', [-1, 0, 1]) == 0
59
60 def test_q3():
61     assert calculate('q3', [1, 3, 5]) == pytest.approx(4.0)
62     assert calculate('q3', [-1, 0, 1]) == pytest.approx(0.5)
63
64 # Запуск тестов
65 if __name__ == '__main__':
66     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\IP-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
1  import pytest
2  from calc import calculate, convert_precision
3
4  @pytest.mark.parametrize("input_value, expected_output", [
5      (0, 0),
6      (1, 0),
7      (0.1, 1),
8      (0.01, 2),
9      (0.000001, 6),
10 ])
11 def test_convert_precision(input_value, expected_output):
12     assert convert_precision(input_value) == expected_output
13
14 def test_addition():
15     assert calculate('+', 1, 2) == 3
16     assert calculate('+', 10, 20) == 30
17     assert calculate('+', -1, 1) == 0
18
19 def test_subtraction():
20     assert calculate('-', 5, 3) == 2
21     assert calculate('-', 10, 20) == -10
22     assert calculate('-', -1, 1) == -2
23
24 def test_multiplication():
25     assert calculate('*', 2, 3) == 6
26     assert calculate('*', 0, 5) == 0
27     assert calculate('*', -2, 3) == -6
28
29 def test_division():
30     assert calculate('/', 6, 3) == 2
31     assert calculate('/', 10, 2) == 5
32     assert calculate('/', 0, 5) == 0
33
34 @pytest.mark.parametrize("values, expected", [
35     ([1, 2, 3, 4, 5], 3),
36     ([-1, 0, 1], 0),
37 ])
38 def test_medium(values, expected):
39     assert calculate('medium', *values) == expected
```

```

41 @pytest.mark.parametrize("values, expected", [
42     ([1, 2, 3, 4, 5], 2),
43     ([-1, 0, 1], 0.6666666666666666),
44 ])
45 def test_variance(values, expected):
46     assert calculate('variance', *values) == pytest.approx(expected)
47
48 @pytest.mark.parametrize("values, expected", [
49     ([1, 2, 3, 4, 5], 1.414214),
50     ([-1, 0, 1], 0.816497),
51 ])
52 def test_std_deviation(values, expected):
53     assert calculate('std_deviation', *values) == pytest.approx(expected)
54
55 @pytest.mark.parametrize("values, expected", [
56     ([1, 3, 5], 3),
57     ([-1, 0, 1], 0),
58 ])
59 def test_median(values, expected):
60     assert calculate('median', *values) == expected
61
62 @pytest.mark.parametrize("values, expected", [
63     ([1, 3, 5], 3),
64     ([-1, 0, 1], 0),
65 ])
66 def test_q2(values, expected):
67     assert calculate('q2', *values) == expected
68
69 @pytest.mark.parametrize("values, expected", [
70     ([1, 3, 5], 4),
71     ([-1, 0, 1], 0.5),
72 ])
73 def test_q3(values, expected):
74     assert calculate('q3', *values) == expected

```

Результат:

```

23.0
5.0
27.0
5.0

```

Код программы calc:

```
calc.py > ...
1  import math
2  import unittest
3  import numpy as np
4
5  def convert_precision(tolerance):
6      if tolerance == 0:
7          return 0
8      try:
9          return abs(math.floor(math.log10(abs(tolerance))))
10     except Exception as e:
11         print(f"Ошибка при конверсии точности: {e}")
12         return 0
13
14     def calculate(action, *args, tolerance=1e-6):
15         if action not in ['+', '-', '*', '/', 'medium', \
16             'variance', 'std_deviation', 'median', 'q2', 'q3']:
17             raise ValueError("Неверная операция")
18
19         try:
20             if (action == '+'):
21                 result = sum(args)
22             elif (action == '-'):
23                 result = args[0]
24                 for i in range(1, len(args)):
25                     result -= args[i]
26             elif (action == '*'):
27                 result = 1
28                 for i in range(len(args)):
29                     result *= args[i]
30             elif (action == '/'):
31                 result = args[0]
32                 for i in range(1, len(args)):
33                     result /= args[i]
34
35             elif (action == 'medium'):
36                 result = np.mean(args)
37             elif (action == 'variance'):
38                 result = np.var(args)
39             elif (action == 'std_deviation'):
40                 result = np.std(args)
41             elif (action == 'median'):
42                 result = np.median(args)
```

```

43         elif (action == 'q2'):
44             result = np.median(args)
45         elif (action == 'q3'):
46             result = np.percentile(args, 75)
47         else:
48             raise ValueError("Неверная операция")
49
50         precision = convert_precision(tolerance)
51         rounded_result = round(result, precision)
52
53         return rounded_result
54     except Exception as e:
55         print(f"Ошибка при вычислении: {e}")
56     """
57     (class) BatchCalculatorContextManager
58 class BatchCalculatorContextManager:
59     def __init__(self, file):
60         self.filename = file
61         self.file = None
62         self.lines = None
63         self.line_count = None
64
65     def __enter__(self):
66         try:
67             self.file = open(self.filename, "r")
68             if not self.file:
69                 raise IOError("Не удалось открыть файл")
70             self.lines = self.file.readlines()
71             self.line_count = len(self.lines)
72             return self
73         except Exception as e:
74             print(f"Ошибка при открытии файла или чтении строк: {e}")
75             raise
76

```

```

77     def __exit__(self, exc_type, exc_val, exc_tb):
78         try:
79             if self.file:
80                 self.file.close()
81         except Exception as e:
82             print(f"Ошибка при закрытии файла: {e}")
83
84     def perform_calculation(self):
85         for line in self.lines:
86             try:
87                 a, op, b = line.split()
88                 yield calculate(op, float(a), float(b))
89             except ValueError as ve:
90                 print(f"Неверное выражение в строке: {ve}")
91             except Exception as e:
92                 print(f"Ошибка при обработке строки: {e}")
93
94     try:
95         with BatchCalculatorContextManager('file.txt') as calc:
96             for i in calc.perform_calculation():
97                 print(i)
98     except IOError as io_error:
99         print(f"Ошибка при работе с файлом: {io_error}")
100 except Exception as general_exception:
101     print(f"Неожиданная ошибка: {general_exception}")

```