

**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет Радиотехнический  
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

Выполнил:  
студент группы РТ5-31Б:  
Длютров Тимофей  
Олегович  
Подпись и дата:

Проверил:  
преподаватель каф. ИУ5  
Гапанюк Юрий  
Евгеньевич  
Подпись и дата:

Москва, 2025

## Текст программы

```
import unittest

class House:
    def __init__(self, ID, street_ID, size, is_being_lived, gaz_connected, water_connected, electricity_connected):
        self.ID = ID
        self.street_ID = street_ID
        self.size = size
        self.is_being_lived = is_being_lived
        self.gaz_connected = gaz_connected
        self.water_connected = water_connected
        self.electricity_connected = electricity_connected

class Street:
    def __init__(self, ID, name, length):
        self.ID = ID
        self.name = name
        self.length = length

class SHConnection:
    def __init__(self, house_ID, street_ID):
        self.house_ID = house_ID
        self.street_ID = street_ID

Streets = [Street(1, "Ленина", 2),
            Street(2, "Сталина", 32),
            Street(3, "Оленя", 22),
            Street(4, "Кеннеди", 17),
            Street(5, "Бобра", 9)]

def FindStreetByID(ID):
    global Streets
    for i in Streets:
        if i.ID == ID:
            return i

Houses = [House(1, 1, 45, True, True, True, True),
          House(2, 1, 25, True, False, True, True),
          House(3, 2, 17, True, True, True, True),
          House(4, 2, 128, True, True, False, True),
          House(5, 3, 280, True, False, True, True),
          House(6, 4, 67, True, True, True, False),
          House(7, 5, 324, True, True, True, False)]

def FindHouseByID(ID):
    global Houses
    for i in Houses:
        if i.ID == ID:
            return i

SHConnections = [SHConnection(1, 1),
                 SHConnection(13 % 5, 52 % 6),
                 SHConnection(3, 1),
                 SHConnection(19 % 5, 73 % 6),
                 SHConnection(4, 3),
                 SHConnection(3, 3),
                 SHConnection(5, 2)]
```

```

def NamedByNameStreets():
    res = []
    for i in Streets:
        if i.name[0].isupper():
            res.append((i.ID, i.name))
    return res

def AverageHousesArea():
    AreaSum = {}
    AreaQuantity = {}
    for i in Houses:
        AreaSum[i.street_ID] = AreaSum.get(i.street_ID, 0) + i.size
        AreaQuantity[i.street_ID] = AreaQuantity.get(i.street_ID, 0) + 1
    Result = {i: AreaSum[i] / AreaQuantity[i] for i in AreaSum.keys()}
    res = []
    for i in Result.keys():
        res.append((FindStreetByID(i).name, Result[i]))
    return res

def HousesWithGaz():
    res = []
    for i in SHConnections:
        if (FindHouseByID(i.house_ID).gaz_connected):
            res.append((FindHouseByID(i.house_ID).ID, FindStreetByID(i.street_ID).name))
    return res

tests_counter = 1

class TestStringMethods(unittest.TestCase):
    def setUp(self):
        pass

    def tearDown(self):
        pass

    #Проверка корректности связей (для любых данных)
    def test_connections(self):
        global Houses, SHConnections, Streets
        streets_ids = [i.ID for i in Streets]
        houses_ids = [i.ID for i in Houses]
        for i in Houses:
            self.assertTrue(i.street_ID in streets_ids)
        for i in SHConnections:
            self.assertTrue(i.street_ID in streets_ids)
            self.assertTrue(i.house_ID in houses_ids)

    #Проверка ответа первого запроса (на заготовленных данных, пропускается при произвольных данных)

```

```

def test_req1(self):
    global NamedByStreets
    needed_result = [
        (1, "Ленина"),
        (2, "Сталина"),
        (4, "Кеннеди")
    ]
    needed_result.sort() #Сортировка, чтобы проверка не зависела от порядка
    res = NamedByStreets()
    res.sort()
    self.assertEqual(len(needed_result), len(res))
    for i in range(len(needed_result)):
        self.assertEqual(res[i], needed_result[i])

#Проверка ответа второго запроса (на заготовленных данных, пропускается при произвольных данных)
def test_req2(self):
    global AverageHousesArea
    needed_result = [
        ("Ленина", 35.0),
        ("Сталина", 72.5),
        ("оленья", 280.0),
        ("Кеннеди", 67.0),
        ("бобра", 324.0)
    ]
    needed_result.sort()
    res = AverageHousesArea()
    res.sort()
    self.assertEqual(len(needed_result), len(res))
    for i in range(len(needed_result)):
        self.assertEqual(res[i], needed_result[i])

#Проверка ответа третьего запроса (на заготовленных данных, пропускается при произвольных данных)
def test_req3(self):
    global HousesWithGaz
    needed_result = [
        (1, "Ленина"),
        (3, "Кеннеди"),
        (3, "Ленина"),
        (4, "Ленина"),
        (4, "оленья"),
        (3, "оленья")
    ]
    needed_result.sort()
    res = HousesWithGaz()
    res.sort()
    self.assertEqual(len(needed_result), len(res))
    for i in range(len(needed_result)):
        self.assertEqual(res[i], needed_result[i])

unittest.main()

```

## Вывод программы

Сначала сделаем обычный вывод

```
.....
```

```
Ran 4 tests in 0.047s
```

```
ОК
```

Все тесты успешно пройдены

Добавим некорректную связь (привяжем один из домов к несуществующей улице):

```
F.E.
=====
ERROR: test_req2 (__main__.TestStringMethods.test_req2)
-----
Traceback (most recent call last):
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 128, in test_req2
    res = AverageHousesArea()
          ^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 74, in AverageHousesArea
    res.append((FindStreetByID(i).name, Result[i]))
               ^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'name'

=====
FAIL: test_connections (__main__.TestStringMethods.test_connections)
-----
Traceback (most recent call last):
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 97, in test_connections
    self.assertTrue(i.street_ID in streets_ids)
AssertionError: False is not true

-----
Ran 4 tests in 0.005s

FAILED (failures=1, errors=1)
```

Не отработает проверка связей и второй запрос (проверку запросов можно пропустить, так как данные нестандартные).

Вернём исходные данные и испортим код первого запроса, чтобы он работал неверно:

```
.F..
=====
FAIL: test_req1 (__main__.TestStringMethods.test_req1)
-----
Traceback (most recent call last):
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 113, in test_req1
    self.assertEqual(len(needed_result), len(res))
AssertionError: 3 != 2

-----
Ran 4 tests in 0.007s

FAILED (failures=1)
```

В этот раз проверка данных прошла успешно, но второй тест выдал ошибку, так как вывод запроса не совпал с требуемым.

Вернём первый запрос к исходному виду и испортим второй:

```
..F.
=====
FAIL: test_req2 (__main__.TestStringMethods.test_req2)
-----
Traceback (most recent call last):
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 132, in test_req2
    self.assertEqual(res[i], needed_result[i])
AssertionError: Tuples differ: ('Кеннеди', 63.0) != ('Кеннеди', 67.0)

First differing element 1:
63.0
67.0

- ('Кеннеди', 63.0)
?               ^

+ ('Кеннеди', 67.0)
?               ^

-----
Ran 4 tests in 0.007s

FAILED (failures=1)
```

В этот раз проверка данных и первого запроса прошла успешно, но третий тест выдал ошибку, так как вывод запроса не совпал с требуемым.

Повторим для третьего запроса:

```
...F
=====
FAIL: test_req3 (__main__.TestStringMethods.test_req3)
-----
Traceback (most recent call last):
  File "C:\Users\dlyut\OneDrive\Desktop\Учёба\ОП\Semester III\RK2\main.py", line 148, in test_req3
    self.assertEqual(len(needed_result), len(res))
AssertionError: 6 != 1

-----
Ran 4 tests in 0.003s

FAILED (failures=1)
```

В этот раз проверка данных, первого и второго запросов прошла успешно, но четвёртый тест выдал ошибку, так как вывод запроса не совпал с требуемым.