МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И. С. ТУРГЕНЕВА»

                                                                                                                                                                   Кафедра информационных систем и цифровых технологий

**ОТЧЁТ**

по лабораторной работе № 6

на тему: «Разработка собственного тестового драйвера»

по дисциплине «Качество и тестирование программного обеспечения»

Выполнили: Банных М.А., Мельников А.Е.

Институт приборостроения, автоматизации и информационных

технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 21ПГ

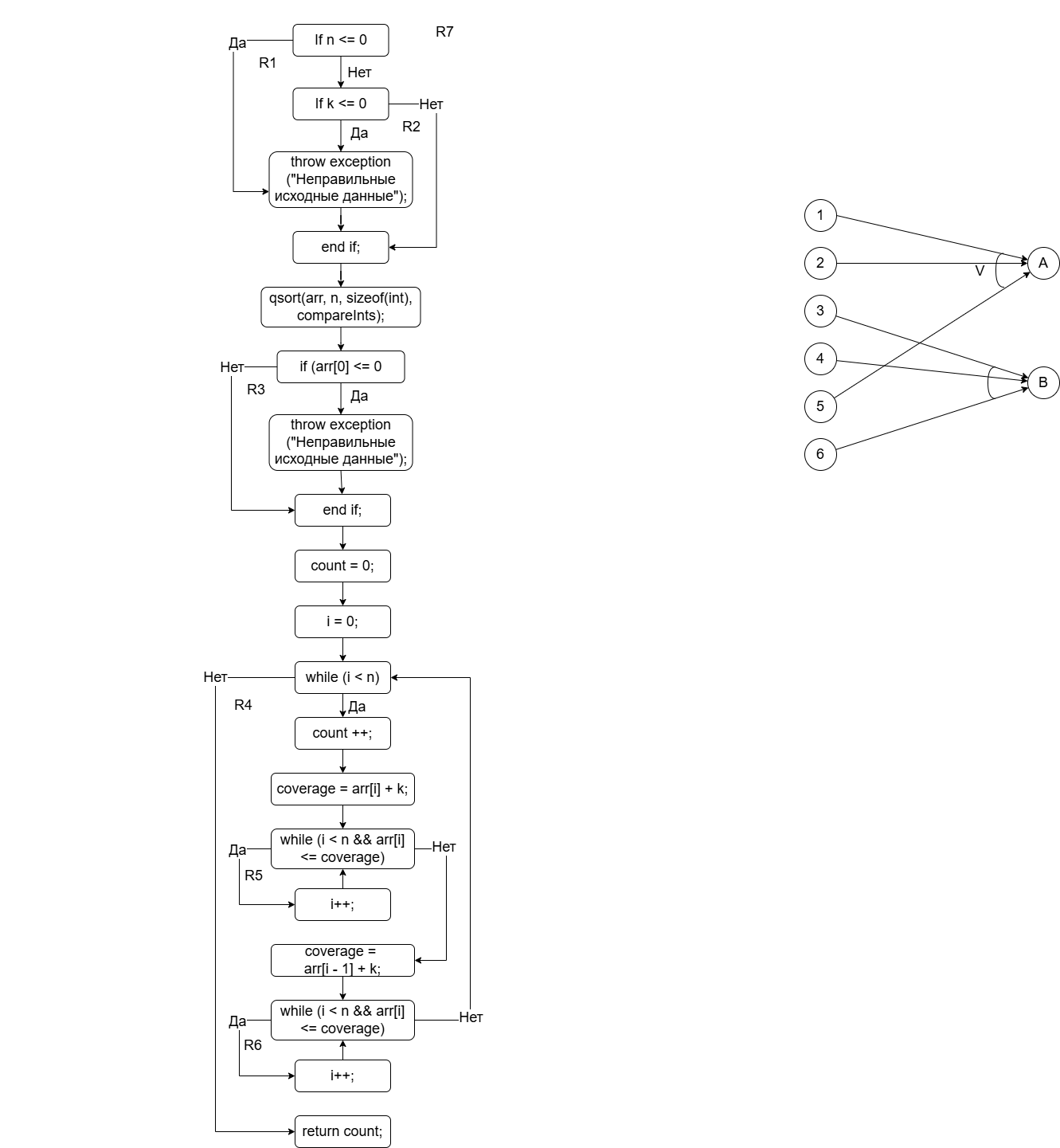
Проверил: Олькина Е.В.

Отметка о зачёте:

Дата: «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Орёл, 2025

**Потоковый граф**

****

Вычисление цикломатической сложности:

1. 24 дуг – 19 узлов + 2 = 7

2. 6 предикатных узлов + 1 = 7

3. 7 регионов

Пути:

1-3

1-2-3

1-2-4-5-6-7

1-2-4-5-6-8-9-10-11-12-13-14-15-16-17-18-19

1-2-4-5-6-8-9-10-11-12-13-14-15-16-17-19

1-2-4-5-6-8-9-10-11-12-13-14-16-17-19 (невозможен по выражению 10 и условию 2)

1-2-4-5-6-8-9-10-11-19 (невозможен по условию 1)

**Тестируемый модуль**

def countTransmitters(arr, n, k):

if n <= 0 or k <= 0:

raise Exception ("Incorrect source data")

arr.sort()

if arr[0] <= 0:

raise Exception ("Incorrect source data")

count = 0

i = 0

while(i < n):

count += 1

coverage = arr[i] + k

while (i < n and arr[i] <= coverage):

i += 1

coverage = arr[i-1] + k

while (i < n and arr[i] <= coverage):

i += 1

return count

**Тесты**

[

{

"name": "countTransmitters",

"arr": [ [] ],

"k": [ 6 ],

"n": [ 0 ],

"expected\_result": "Incorrect source data",

"module": "countingTransmitters"

},

{

"name": "countTransmitters",

"arr": [ [ 3, 1, 4, 6, 2 ] ],

"k": [ 0 ],

"n": [ 5 ],

"expected\_result": "Incorrect source data",

"module": "countingTransmitters"

},

{

"name": "countTransmitters",

"arr": [ [ 3, -1, 4, 0, 2 ] ],

"k": [ 6 ],

"n": [ 5 ],

"expected\_result": "Incorrect source data",

"module": "countingTransmitters"

},

{

"name": "countTransmitters",

"arr": [ [ 19, 23, 24 ] ],

"k": [ 4 ],

"n": [ 3 ],

"expected\_result": 1,

"module": "countingTransmitters"

},

{

"name": "countTransmitters",

"arr": [ [ 1, 9 ] ],

"k": [ 7 ],

"n": [ 2 ],

"expected\_result": 2,

"module": "countingTransmitters"

}

]

**Тестовый драйвер**

import importlib.util

import json

from pathlib import Path

class TestDriver:

def \_\_init\_\_(self, module\_path, test\_file):

self.module\_path = module\_path

self.test\_file = test\_file

def load\_module(self, module\_name):

spec = importlib.util.spec\_from\_file\_location(module\_name, self.module\_path / f"{module\_name}.py")

module = importlib.util.module\_from\_spec(spec)

spec.loader.exec\_module(module)

return module

def run\_tests(self):

with open(self.test\_file, 'r') as file:

tests = json.load(file)

results = []

for i, test in enumerate(tests, start=1):

try:

module = self.load\_module(test['module'])

func = getattr(module, test['name'], None)

arr = test['arr'][0]

k = test['k'][0]

n = test['n'][0]

result = func(arr, n, k)

expected\_result = test['expected\_result']

status = 'success' if result == expected\_result else 'failed'

message = '' if status == 'success' else f"Expected {expected\_result}, got {result}"

results.append({

'test\_number': i,

'received\_result': result,

'expected\_result': expected\_result,

'status': status,

'message': message

})

except Exception as e:

received\_message = str(e)

expected\_message = test.get('expected\_result', '')

status = 'success' if received\_message == expected\_message else 'failed'

results.append({

'test\_number': i,

'received\_result': received\_message,

'expected\_result': expected\_message,

'status': status,

'message': '' if status == 'success' else f"Expected {expected\_message}, got {received\_message}"

})

return results

def generate\_report(self, results):

for result in results:

report = "\n"

report += f"\nTest #{result['test\_number']}: {'Success' if result['status'] == 'success' else 'Failed'}"

report += f"\n\tReceived Result: {result['received\_result']}"

report += f"\n\tExpected Result: {result['expected\_result']}"

report += f"\n\tMessage: {result['message']}" if result['message'] else ""

print(report)

if \_\_name\_\_ == "\_\_main\_\_":

module\_path = Path("test\_modules")

test\_file = "tests.json"

driver = TestDriver(module\_path, test\_file)

results = driver.run\_tests()

driver.generate\_report(results)

