МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Вычислительные системы и технологии»

ОТЧЁТ

по лабораторной работе №1

«Unit - тестирование»

по дисциплине «Тестирование программного обеспечения»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Скорынин С.С.

СТУДЕНТ:

Китов А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

21-ПО

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2025

**Цель работы:** изучить основы разработки модульных тестов, получить навыки работы со средствами тестирования.

**Задание:**

* Изучить средства тестирования, доступные в Visual Studio
* Разработать набор unit-тестов
* Реализовать алгоритм или задач на языке С#, поддерживаемом платформой .NET Использование других языков необходимо заранее согласовать с преподавателем.
* Обеспечить максимально возможное покрытие кода тестами. При написании задачи писать слабосвязанный код. Использовать mock- объекты. Использовать утилиты покрытия кода.
* Продемонстрировать: работающую функциональность в соответствии с заданием. Показать с помощью утилиты процент покрытия кода unit-тестами.
* Сделать отчет по лабораторной работе.

**Ход работы:**

Создан файл bubblesort.py, в котором реализованы:

Функция generate\_random\_list(length) для генерации случайного списка указанной длины.

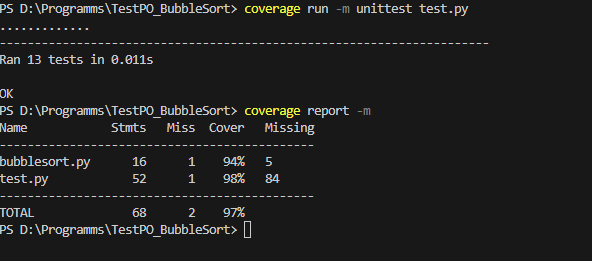
Функция bubble\_sort(arr) для сортировки массива методом пузырька.

Создан файл test\_bubblesort.py, в котором написаны тесты для проверки корректности работы алгоритма.

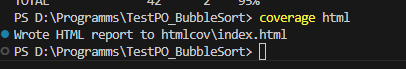
Реализованы следующие тесты:

* Тест на пустой список.
* Тест на отсортированный массив.
* Тест на обратно отсортированный массив.
* Тест на частично упорядоченный массив.
* Тест на массив с повторяющимися элементами.
* Тест на массив, все элементы которого одинаковы.
* Тест на массив с отрицательными числами.
* Тест на массив большого размера.
* Тест на обработку случайных значений с использованием unittest.mock.
* Тест на то, что функция generate\_random\_list вызывается один раз.

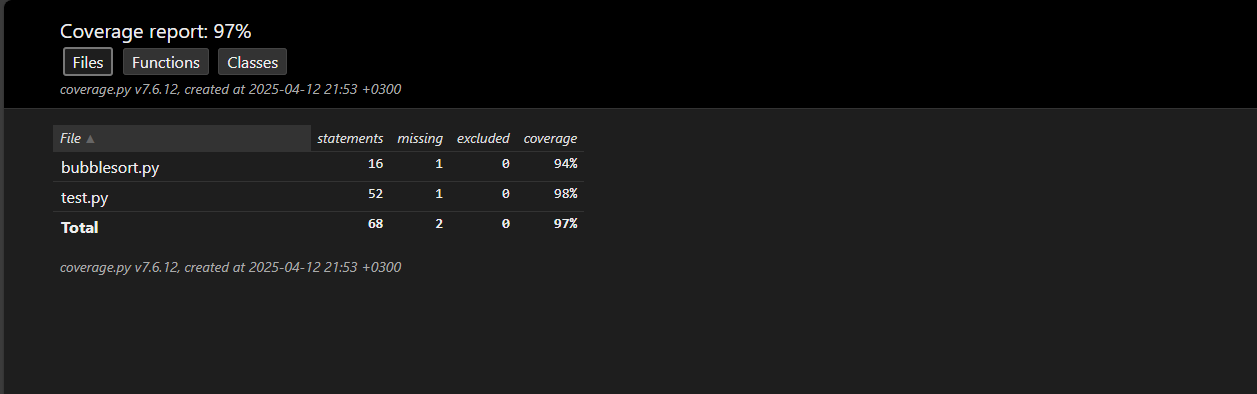
**Результат:**

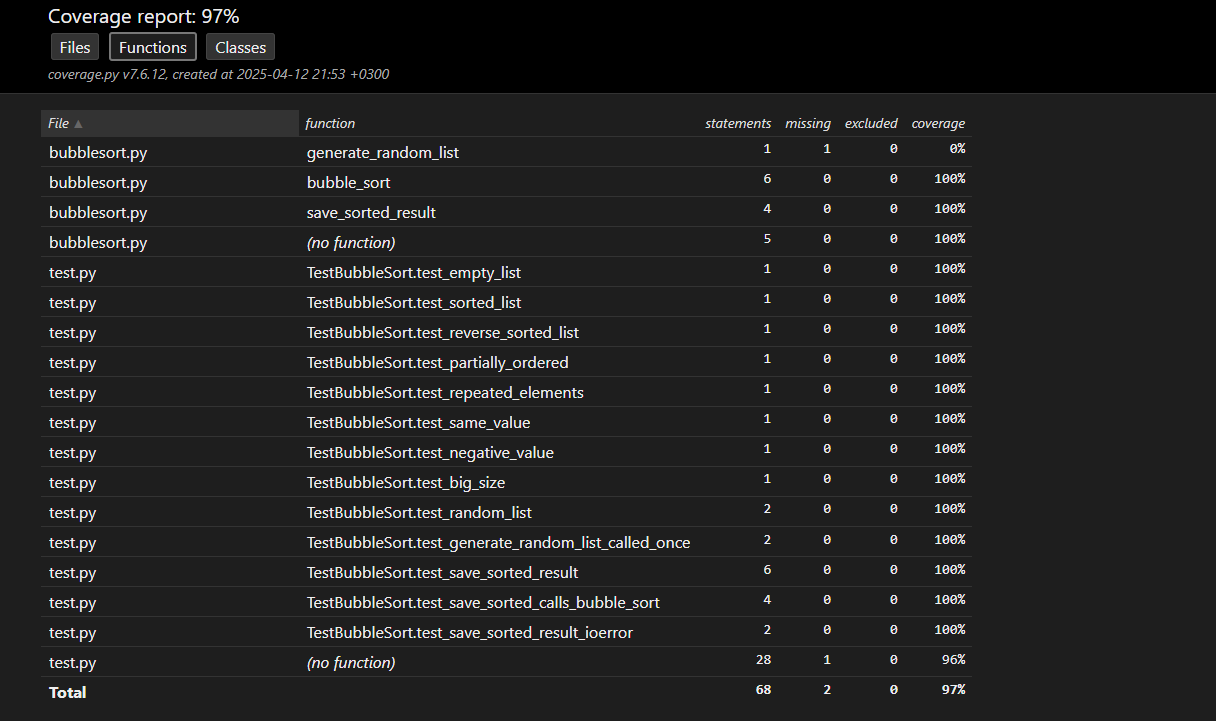


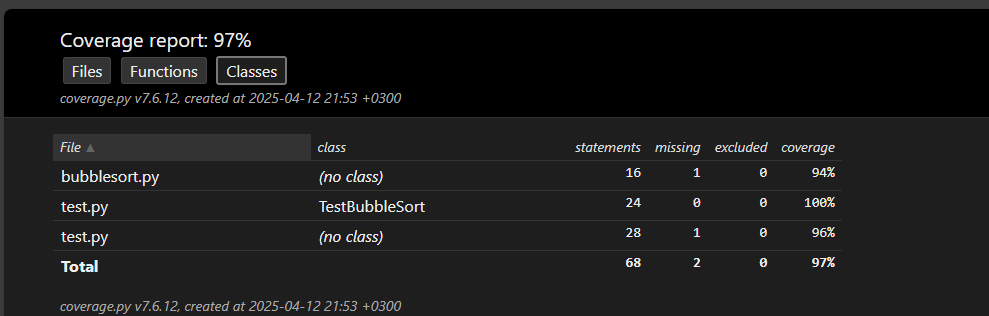
Так же можно сгенерировать отчет в формате html.



**Где более красиво всё показано:**

****

****

****

**Вывод:** изучили основы разработки модульных тестов на языке Python, получили навыки работы со средствами тестирования. Реализовали алгоритм сортировки пузырьком и написали Unit – тесты для проверки корректной работы алгоритма. При написании тестов использовали mock – объекты, а также использовали утилиту покрытия кода coverage.

**Код программы:**

import unittest

import bubblesort

from unittest.mock import patch, mock\_open, MagicMock

import random

import json

class TestBubbleSort(unittest.TestCase):

    # тест на пустой список

    def test\_empty\_list(self):

        self.assertEqual(bubblesort.bubble\_sort([]), [])

    # тест на отсортированный массив

    def test\_sorted\_list(self):

        self.assertEqual(bubblesort.bubble\_sort([1, 2, 3, 4, 5]), [1, 2, 3, 4, 5])

    # тест на обратноотсортированный массив

    def test\_reverse\_sorted\_list(self):

        self.assertEqual(bubblesort.bubble\_sort([5, 4, 3, 2, 1]), [1, 2, 3, 4, 5])

    # тест на частично упорядоченный массив

    def test\_partially\_ordered(self):

        self.assertEqual(bubblesort.bubble\_sort([6, 8, 1, 2, 9, 10, 11, 12]), [1, 2, 6, 8, 9, 10, 11, 12])

    # тест на массив с повторяющимися элементами

    def test\_repeated\_elements(self):

        self.assertEqual(bubblesort.bubble\_sort([3, 3, 4, 2, 5, 5, 2]), [2, 2, 3, 3, 4, 5, 5])

    # тест на массив, все значения которого одинаковы

    def test\_same\_value(self):

        self.assertEqual(bubblesort.bubble\_sort([5, 5, 5, 5, 5, 5, 5, 5]), [5, 5, 5, 5, 5, 5, 5, 5])

    # тест на массив с отрицательными числами

    def test\_negative\_value(self):

        self.assertEqual(bubblesort.bubble\_sort([5, -1, 9, 0, -6]), [-6, -1, 0, 5, 9])

    # тест на массив большого размера

    def test\_big\_size(self):

        self.assertEqual(bubblesort.bubble\_sort([5, 1, 7, 98, 65, 75, 21, 97, 34, 96, 82, 74, 25, 16, 99, 12, 0, 3]), [0, 1, 3, 5, 7, 12, 16, 21, 25, 34, 65, 74, 75, 82, 96, 97, 98, 99])

    # тест на обработку случайных значений

    @patch('random.randint')

    @patch('bubblesort.generate\_random\_list')

    def test\_random\_list(self, mock\_generate\_random\_list, mock\_random):

        mock\_generate\_random\_list.return\_value = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

        self.assertEqual(bubblesort.bubble\_sort(bubblesort.generate\_random\_list(11)), [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9])

    # тест на то, что функция генерирует один список

    @patch('bubblesort.generate\_random\_list')

    def test\_generate\_random\_list\_called\_once(self, mock\_generate\_random\_list):

        bubblesort.bubble\_sort(bubblesort.generate\_random\_list(5))

        mock\_generate\_random\_list.assert\_called\_once()

    # Тест сохранения в файл с моком файловой системы

    @patch('builtins.open', new\_callable=mock\_open)

    @patch('json.dump')

    def test\_save\_sorted\_result(self, mock\_json\_dump, mock\_file):

        test\_data = [3, 1, 4, 2]

        expected\_sorted = [1, 2, 3, 4]

        result = bubblesort.save\_sorted\_result(test\_data, 'dummy.json')

        self.assertTrue(result)

        mock\_file.assert\_called\_once\_with('dummy.json', 'w')

        mock\_json\_dump.assert\_called\_once\_with(expected\_sorted, mock\_file())

    # Тест, что функция save\_sorted\_result вызывает bubble\_sort

    @patch('bubblesort.bubble\_sort')

    def test\_save\_sorted\_calls\_bubble\_sort(self, mock\_bubble\_sort):

        mock\_bubble\_sort.return\_value = [1, 2, 3]

        test\_data = [3, 2, 1]

        bubblesort.save\_sorted\_result(test\_data, 'dummy.json')

        mock\_bubble\_sort.assert\_called\_once\_with(test\_data.copy())

    # Тест обработки ошибки при записи файла

    @patch('builtins.open', side\_effect=IOError('Disk error'))

    def test\_save\_sorted\_result\_ioerror(self, mock\_file):

        with self.assertRaises(IOError):

            bubblesort.save\_sorted\_result([1, 2, 3], 'dummy.json')

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

Код файла с сортировкой:

import random

import json

def generate\_random\_list(length):

    return [random.randint(10, 100) for \_ in range(length)]

def bubble\_sort(arr):

    n = len(arr)

    for i in range(n):

        for j in range(0, n-i-1):

            if arr[j] > arr[j+1]:

                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr

def save\_sorted\_result(arr, filename):

    sorted\_arr = bubble\_sort(arr.copy())

    with open(filename, 'w') as f:

        json.dump(sorted\_arr, f)

    return True