МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №2

по дисциплине «Тестирование и верификация ПО»

Выполнили:
Студенты группы ИКБО-13-22 Горошников К.С, Четин Г.М, Пашкин В.П, Измайлов А.Р.

Проверил:
Преподаватель Запорожских А.И.

2024 г.

ПРАКТИЧЕСКАЯ РАБОТА №2
МОДУЛЬНОЕ ТЕСТИРОВАНИЕ

Цель работы: познакомиться с концепцией модульного тестирования, научиться проектировать и реализовывать модульные тесты для отдельных компонентов программного обеспечения.

1. Горошников К.С. – модуль программы «Работа с хеш-таблицой»

Участник команды написал модуль программы «Работа с хеш-таблицой».

Была написана документация к модулю.

Описание модуля: Программа по работе с базой данных студентов использует хеш-таблицу для хранения и управления информацией о студентах (рис. 1). Каждый студент представлен объектом класса Student. Данный модуль был передан Пашкину В.П для дальнейшего тестирования.

Были реализованы следующие функции:

1) **Добавление студента:** добавляет нового студента в базу данных по его уникальному идентификатору.

2) **Удаление студента:** удаляет студента по его идентификатору. Возвращает true или false как результат удаления.

3) **Поиск студента:** находит студента по его уникальному идентификатору.

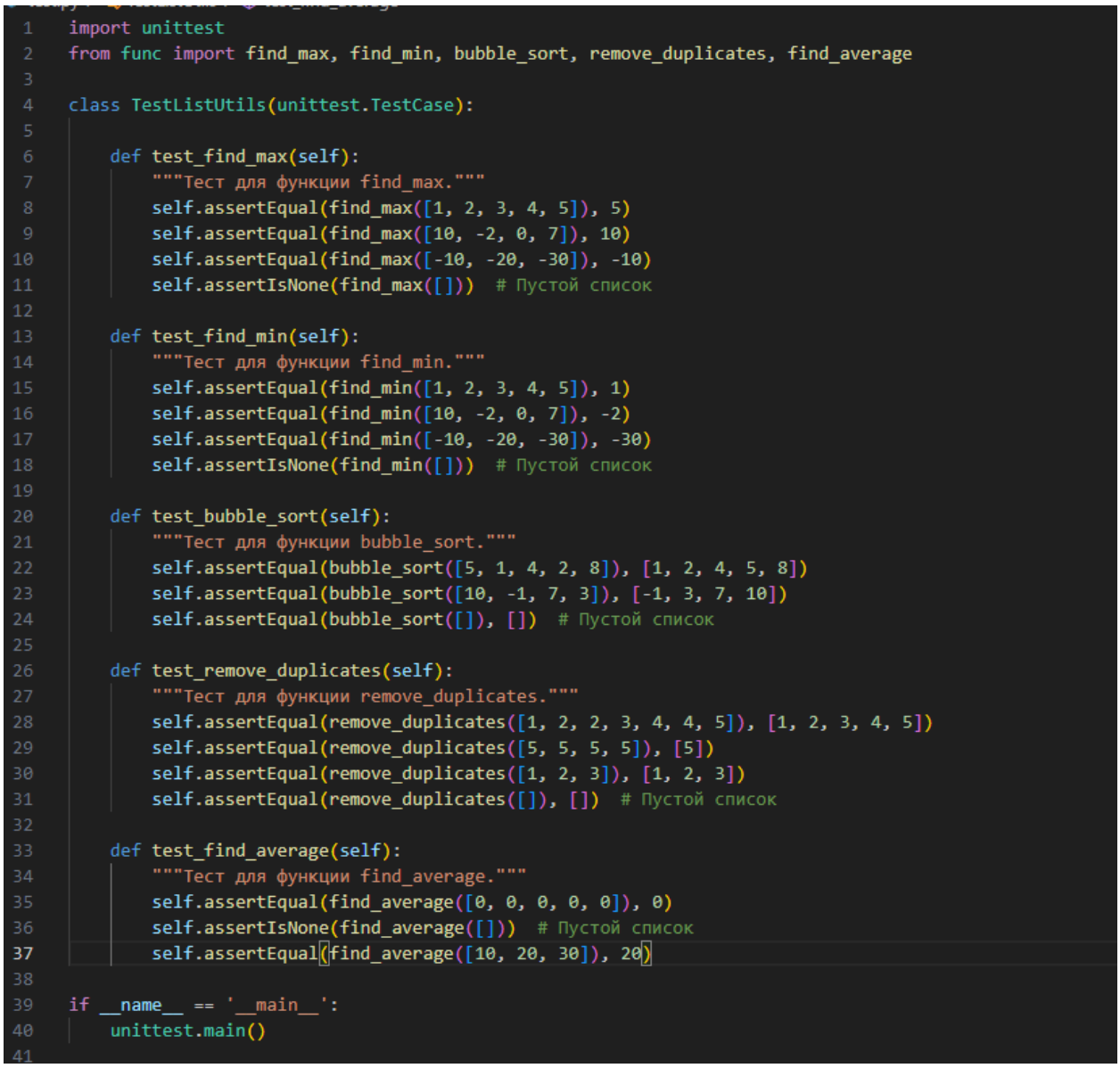
4) **Обновление оценок студента:** изменяет оценки студента, заменив его текущий массив оценок на новый.

5) **Вывод всех студентов:** выводит на экран информацию обо всех студентах в базе данных.

6) **Сортировка студентов по средней оценке:** сортирует студентов по их средней оценке (от наивысшей к наименьшей) и выводит отсортированный список.

Рисунок 1 – Код модуля «Работа с хеш-таблицей»

В качестве практики по тестированию участнику был передан модуль программы «Работа со списками» от Четина Г.М. Были написаны модульные тесты с помощью библиотеки unittest, для проверки корректности функций модуля (рис.2).



```python
import unittest
from func import find_max, find_min, bubble_sort, remove_duplicates, find_average

class TestListUtils(unittest.TestCase):

    def test_find_max(self):
        """Тест для функции find_max."""
        self.assertEqual(find_max([1, 2, 3, 4, 5]), 5)
        self.assertEqual(find_max([10, -2, 0, 7]), 10)
        self.assertEqual(find_max([-10, -20, -30]), -10)
        self.assertIsNone(find_max([]))  # Пустой список

    def test_find_min(self):
        """Тест для функции find_min."""
        self.assertEqual(find_min([1, 2, 3, 4, 5]), 1)
        self.assertEqual(find_min([10, -2, 0, 7]), -2)
        self.assertEqual(find_min([-10, -20, -30]), -30)
        self.assertIsNone(find_min([]))  # Пустой список

    def test_bubble_sort(self):
        """Тест для функции bubble_sort."""
        self.assertEqual(bubble_sort([5, 1, 4, 2, 8]), [1, 2, 4, 5, 8])
        self.assertEqual(bubble_sort([10, -1, 7, 3]), [-1, 3, 7, 10])
        self.assertEqual(bubble_sort([]), [])  # Пустой список

    def test_remove_duplicates(self):
        """Тест для функции remove_duplicates."""
        self.assertEqual(remove_duplicates([1, 2, 2, 3, 4, 4, 5]), [1, 2, 3, 4, 5])
        self.assertEqual(remove_duplicates([5, 5, 5, 5]), [5])
        self.assertEqual(remove_duplicates([1, 2, 3]), [1, 2, 3])
        self.assertEqual(remove_duplicates([]), [])  # Пустой список

    def test_find_average(self):
        """Тест для функции find_average."""
        self.assertEqual(find_average([0, 0, 0, 0, 0]), 0)
        self.assertIsNone(find_average([]))  # Пустой список
        self.assertEqual(find_average([10, 20, 30]), 20)

if __name__ == '__main__':
    unittest.main()
```

Рисунок 2 – Модульные тесты модуля «Работа со списками»

В результате проверки, один из тестов вызвал ошибку в коде.

```
----------------------------------------------------------------------
FAIL: test_find_average (__main__.TestListUtils.test_find_average)
Тест для функции find_average.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\grego\Downloads\mmm\test.py", line 35, in test_find_average
    self.assertEqual(find_average([1, 2, 3, 4, 5]), 3)
AssertionError: 75 != 3
```

Рисунок 3 – Провал теста на вычисление среднего значения элементов списка

Ошибка происходит в функции find_average(lst), т.к. неверное прописана логика вычисления. Была исправлена данная ошибка и передана участнику Четину Г.М. для итогового тестирования (рис.4).

```python
def find_average(lst):
    """Находит среднее значение списка."""
    if not lst:
        return None
    return sum(lst) / len(lst)
```

Рисунок 4 – Исправленная функция

Краткое описание ошибки: Неверное вычисление среднего значения списка.

Статус ошибки: Открыта (Open).

Категория ошибки: Серьезная (Major).

Тестовый случай: Проверка алгоритма функционирования программы.

Описание ошибки:

1. Загрузить программу.
2. Ввести список чисел, например: [1, 2, 3, 4, 5].
3. Вызвать функцию find_average().
4. Полученный результат: 75 (в результате неверного умножения).
5. Ожидаемый результат: 3 (правильное среднее значение).

Была получена документация на ошибку в модуле «Работа с хеш-таблицей» от Пашкина В.П., по которой она была найдена и исправлена (рис.5).

Краткое описание ошибки: Отсутствие проверки на наличие студента в базе данных при удалении.

Статус ошибки: закрыта («Closed»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности работы функции удаления студента.

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод removeStudent с несуществующим studentID;

3. Запустить программу;

4. Ожидаемый результат: программа должна вывести сообщение “Студент с таким ID не найден” и вернуть false.

5. Полученный результат: программа выводит сообщение “Студент с ID … удален” и возвращает true.

```
public boolean removeStudent(String studentID) {
    if (studentDB.containsKey(studentID)) { // Проверяем, есть ли студент с таким ID
        studentDB.remove(studentID);
        System.out.println("Студент с ID " + studentID + " удалён.");
        return true; // Удаление прошло успешно
    } else {
        System.out.println(x:"Студент с таким ID не найден.");
        return false; // Студент не найден
    }
}
```

Рисунок 5 – Исправленный модуль “Работа с хеш-таблицей”

Было проведено итоговое тестирование корректности работы всех функций, которое показало, что все ошибки были исправлены (рис.6).

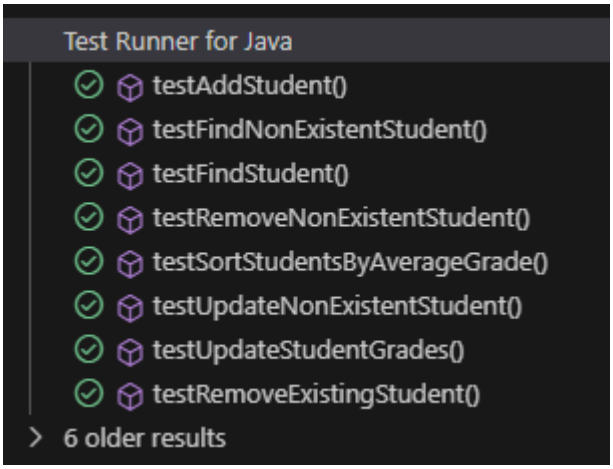


Рисунок 6 – Итоговое тестирование модуля «Работа с хеш-таблицей»

2. Четин Г.М. – модуль программы «Работа со списками»

Участник команды написал модуль программы «Работа со списками».

Была написана документация к модулю.

Модуль предназначен для выполнения различных операций над списками, таких как поиск максимального и минимального значений, сортировка, удаление дубликатов и вычисление среднего значения. Этот модуль может быть использован в различных программах для обработки данных и анализа массивов чисел (рис. 7). Данный модуль был передан Горошникову К.С. для дальнейшего тестирования.

Были реализованы следующие функции:

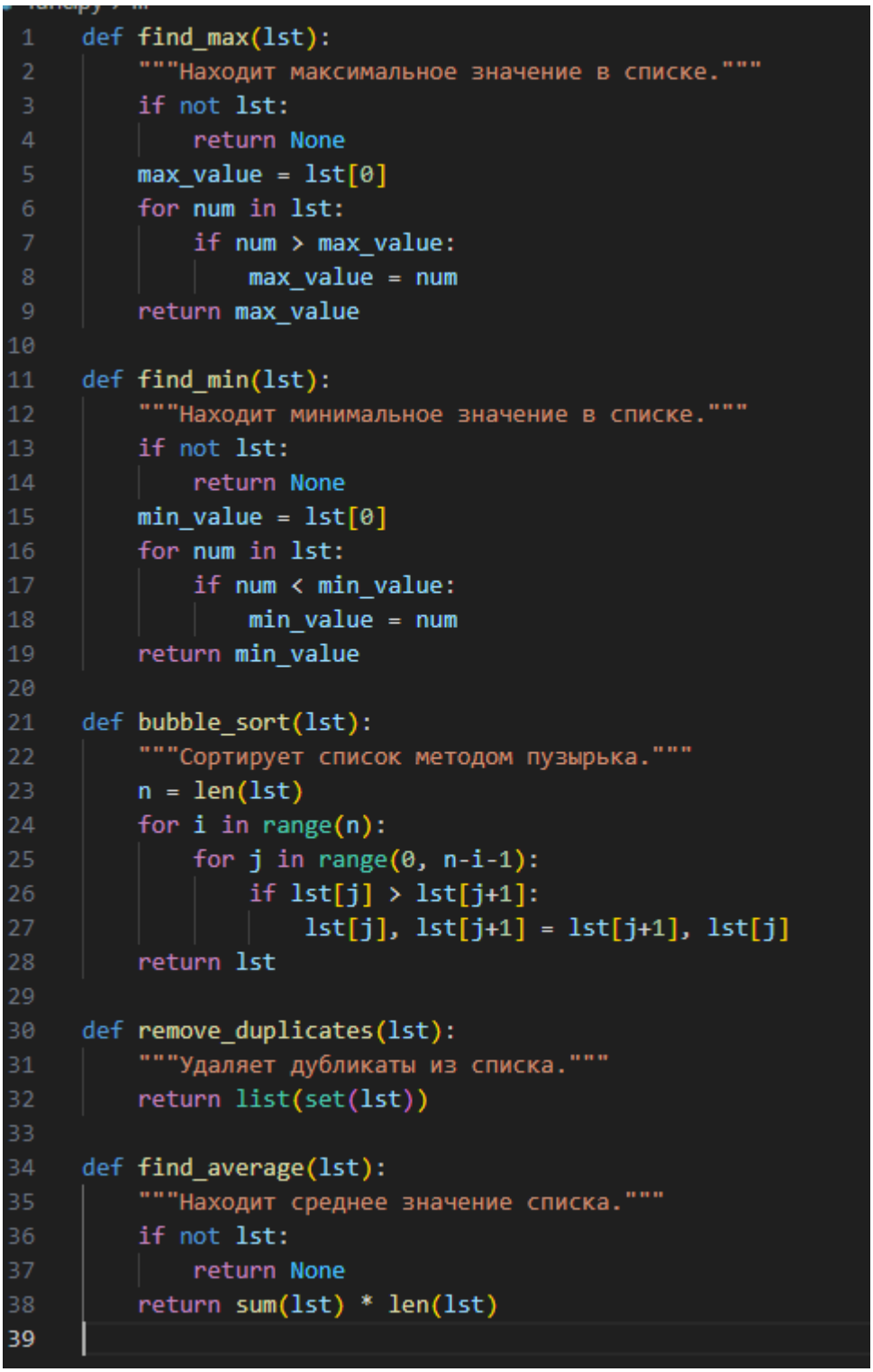
7) **find_max(lst)** - функция для поиска максимального значения в списке.

8) **find_min(lst)** - функция для поиска минимального значения в списке.

9) **bubble_sort(lst)** - функция для сортировки списка методом пузырька.

10) **remove_duplicates(lst)** - функция для удаления дубликатов из списка.

11) **find_average(lst) -** функция для нахождения среднего значения элементов списка.



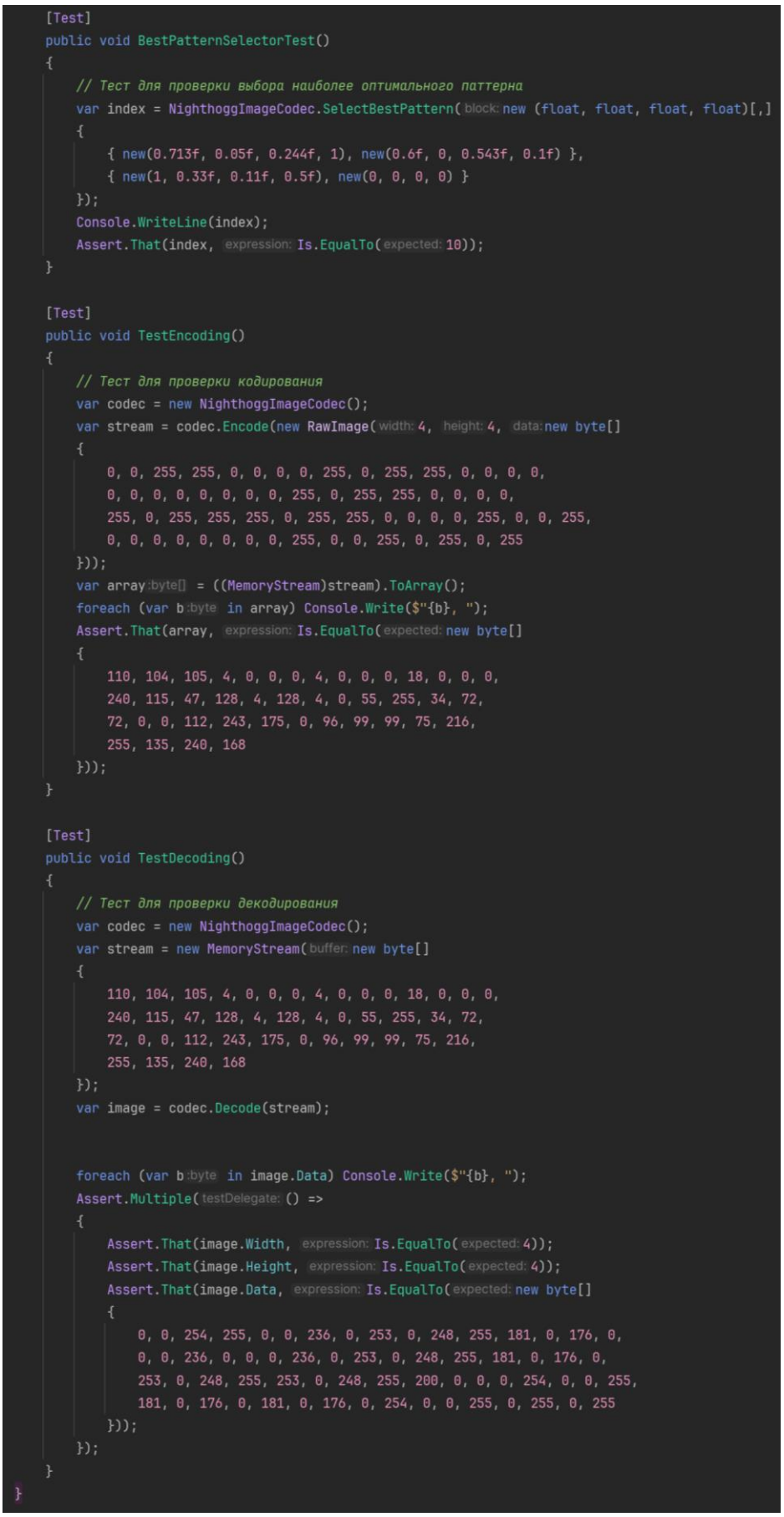
```python
def find_max(lst):
    """Находит максимальное значение в списке."""
    if not lst:
        return None
    max_value = lst[0]
    for num in lst:
        if num > max_value:
            max_value = num
    return max_value

def find_min(lst):
    """Находит минимальное значение в списке."""
    if not lst:
        return None
    min_value = lst[0]
    for num in lst:
        if num < min_value:
            min_value = num
    return min_value

def bubble_sort(lst):
    """Сортирует список методом пузырька."""
    n = len(lst)
    for i in range(n):
        for j in range(0, n-i-1):
            if lst[j] > lst[j+1]:
                lst[j], lst[j+1] = lst[j+1], lst[j]
    return lst

def remove_duplicates(lst):
    """Удаляет дубликаты из списка."""
    return list(set(lst))

def find_average(lst):
    """Находит среднее значение списка."""
    if not lst:
        return None
    return sum(lst) * len(lst)

```

Рисунок 7 – Код модуля «Работа со списками»

В качестве практики по тестированию участнику был передан модуль программы «Кодирование изображения» от Измайлова А.Р. Были написаны модульные тесты с помощью библиотеки NUnit, для проверки корректности функций модуля (рис. 8-9).

```
namespace CodecTest;

[TestFixture]
public class ImageCodecTests
{
    [Test]
    public void BitDataWriteTest()
    {
        // Тест для проверки работы побитовой записи
        using var stream = new MemoryStream();
        using var writer = new BinaryWriter(stream);
        var pos = 0;
        long buff = 0;
        NighthoggImageCodec.WriteDataWithShift(writer, 127, shift: 9, ref buff, ref pos);
        NighthoggImageCodec.WriteDataWithShift(writer, 13743, shift: 17, ref buff, ref pos);

        var array:byte[] = stream.ToArray();
        Assert.Multiple(testDelegate: () =>
        {
            Assert.That(pos, expression: Is.EqualTo(expected: 2));
            Assert.That(array, expression: Has.Length.EqualTo(expected: 3));
            Assert.That(array, expression: Is.EqualTo(expected: new byte[] { 127, 94, 107 }));
            Assert.That(buff, expression: Is.EqualTo(expected: 0));
        });
    }

    [Test]
    public void BitDataReadTest()
    {
        // Тест для проверки работы побитового чтения
        using var stream = new MemoryStream(buffer: new byte[] { 127, 94, 107, 0 });
        using var reader = new BinaryReader(stream);
        var pos = 0;
        long buff = 0;
        var r1:long = NighthoggImageCodec.ReadDataOfSize(reader, size: 9, ref buff, ref pos);
        var r2:long = NighthoggImageCodec.ReadDataOfSize(reader, size: 17, ref buff, ref pos);

        Assert.Multiple(testDelegate: () =>
        {
            Assert.That(pos, expression: Is.EqualTo(expected: 6));
            Assert.That(buff, expression: Is.EqualTo(expected: 0));
            Assert.That(r1, expression: Is.EqualTo(expected: 127));
            Assert.That(r2, expression: Is.EqualTo(expected: 13743));
        });
    }
    [Test]
    public void PatternTest()
    {
        // Тест для проверки работы построителя матрицы паттерна
        var (matrix:int[,] , max:int ) = NighthoggImageCodec.Pattern(index: 11);
        Assert.Multiple(testDelegate: () =>
        {
            Assert.That(matrix, expression: Is.EqualTo(expected: new[,] {{0, 1}, {2, 1}}));
            Assert.That(max, expression: Is.EqualTo(expected: 3));
        });
    }
```

Рисунок 8 – Модульные тесты модуля «Кодирование изображения»



```
    [Test]
    public void BestPatternSelectorTest()
    {
        // Тест для проверки выбора наиболее оптимального паттерна
        var index = NighthoggImageCodec.SelectBestPattern(block: new (float, float, float, float)[,]
        {
            { new(0.713f, 0.05f, 0.244f, 1), new(0.6f, 0, 0.543f, 0.1f) },
            { new(1, 0.33f, 0.11f, 0.5f), new(0, 0, 0, 0) }
        });
        Console.WriteLine(index);
        Assert.That(index, expression: Is.EqualTo(expected: 10));
    }

    [Test]
    public void TestEncoding()
    {
        // Тест для проверки кодирования
        var codec = new NighthoggImageCodec();
        var stream = codec.Encode(new RawImage(width: 4, height: 4, data: new byte[]
        {
            0, 0, 255, 255, 0, 0, 0, 0, 255, 0, 255, 255, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 255, 255, 0, 0, 0, 0,
            255, 0, 255, 255, 255, 0, 255, 255, 0, 0, 0, 0, 255, 0, 0, 255,
            0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 255, 0, 255, 0, 255
        }));
        var array:byte[] = ((MemoryStream)stream).ToArray();
        foreach (var b:byte in array) Console.Write($"{b}, ");
        Assert.That(array, expression: Is.EqualTo(expected: new byte[]
        {
            110, 104, 105, 4, 0, 0, 0, 4, 0, 0, 0, 18, 0, 0, 0,
            240, 115, 47, 128, 4, 128, 4, 0, 55, 255, 34, 72,
            72, 0, 0, 112, 243, 175, 0, 96, 99, 99, 75, 216,
            255, 135, 240, 168
        }));
    }

    [Test]
    public void TestDecoding()
    {
        // Тест для проверки декодирования
        var codec = new NighthoggImageCodec();
        var stream = new MemoryStream(buffer: new byte[]
        {
            110, 104, 105, 4, 0, 0, 0, 4, 0, 0, 0, 18, 0, 0, 0,
            240, 115, 47, 128, 4, 128, 4, 0, 55, 255, 34, 72,
            72, 0, 0, 112, 243, 175, 0, 96, 99, 99, 75, 216,
            255, 135, 240, 168
        });
        var image = codec.Decode(stream);


        foreach (var b:byte in image.Data) Console.Write($"{b}, ");
        Assert.Multiple(testDelegate: () =>
        {
            Assert.That(image.Width, expression: Is.EqualTo(expected: 4));
            Assert.That(image.Height, expression: Is.EqualTo(expected: 4));
            Assert.That(image.Data, expression: Is.EqualTo(expected: new byte[]
            {
                0, 0, 254, 255, 0, 0, 236, 0, 253, 0, 248, 255, 181, 0, 176, 0,
                0, 0, 236, 0, 0, 0, 236, 0, 253, 0, 248, 255, 181, 0, 176, 0,
                253, 0, 248, 255, 253, 0, 248, 255, 200, 0, 0, 0, 254, 0, 0, 255,
                181, 0, 176, 0, 181, 0, 176, 0, 254, 0, 0, 255, 0, 255, 0, 255
            }));
        });
    }
}
```

Рисунки 9 – Модульные тесты модуля «Кодирование изображения»

В результате проверки, один из тестов вызвал ошибку в коде.

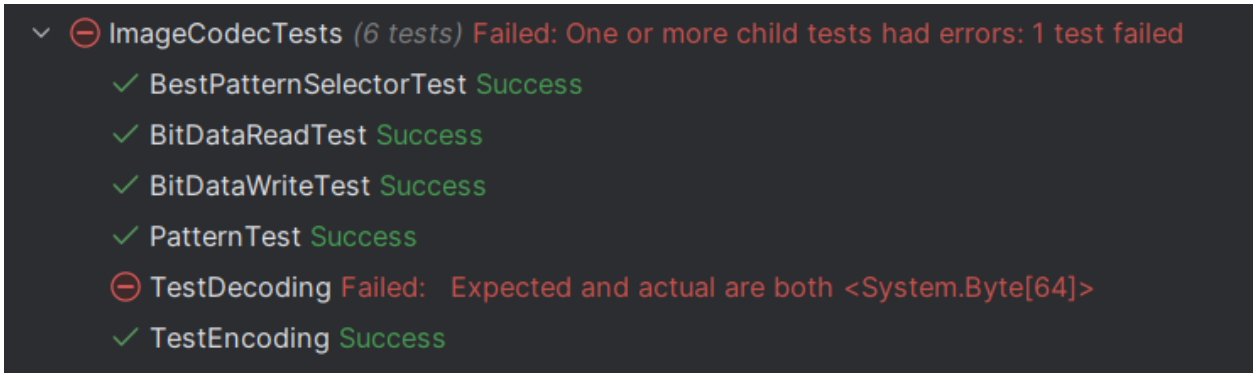


Рисунок 10 – Провал теста на декодирование изображения

Ошибка происходит в функции `Decode(Stream stream)` в процессе преобразования цветов. Была исправлена данная ошибка и передана участнику Измайлову А.Р. для итогового тестирования (рис.11).

```
public RawImage Decode(Stream stream)
{
    using var reader = new BinaryReader(stream);
    if (Verifier.Any(v:byte => reader.ReadByte() != v))
        throw new FormatException("Codec signature mismatch!");

    var width:uint = reader.ReadUInt32();
    var height:uint = reader.ReadUInt32();
    var data = new byte[4 * width * height];

    var lMul:int = (1 << LuminanceBits) - 1;
    var cMul:int = (1 << ColorBits) - 1;
    var aMul:int = (1 << AlphaBits) - 1;
    var amMul:int = (1 << AlphaMultiplierBits) - 1;
    var amFactor:float = (float) amMul / 3;

    var pos = 0;
    long buff = 0;
    var lightness = new float[2, 2];
    var colors = new (float, float, float)[3];
    for (var by = 0; by < height; by += 2)
    for (var bx = 0; bx < width; bx += 2)
    {
        for (var y = 0; y < 2; y++)
        for (var x = 0; x < 2; x++)
            lightness[x, y] = (float) ReadDataOfSize(reader, size:LuminanceBits, ref buff, ref pos) / lMul;

        var patternIndex = (int)ReadDataOfSize(reader, size:PatternBits, ref buff, ref pos);
        var (pattern:int[,], count) = Pattern(patternIndex);

        for (var i = 0; i < count; i++)
            colors[i] = (
                (float) ReadDataOfSize(reader, size:ColorBits, ref buff, ref pos) / cMul,
                (float) ReadDataOfSize(reader, size:ColorBits, ref buff, ref pos) / cMul,
                (float) ReadDataOfSize(reader, size:AlphaBits, ref buff, ref pos) / aMul
            );

        for (var y = 0; y < 2; y++)
        for (var x = 0; x < 2; x++)
        {
            if (bx + x >= width || by + y >= height) continue;
            var am:float = ReadDataOfSize(reader, size:AlphaMultiplierBits, ref buff, ref pos) / amFactor;
            var mul:float = 2f * am / 3;

            var l:float = lightness[x, y];
            var u:float = (colors[pattern[x, y]].Item1 * 2 - 1) * 0.9278f;
            var v:float = (colors[pattern[x, y]].Item2 * 2 - 1) * 0.7874f;
            var t:float = MathF.Max(0, MathF.Min(1, colors[pattern[x, y]].Item3 * mul));

            var i:long = (bx + x + (by + y) * width) * 4;

            var r:float = MathF.Max(0, MathF.Min(1, l + v));
            var g:float = MathF.Max(0, MathF.Min(1, l - (0.2126f * v + 0.0722f * u) / 0.7152f));
            var b:float = MathF.Max(0, MathF.Min(1, l + u));
            data[i] = (byte)(r * 255);
            data[i + 1] = (byte)(g * 255);
            data[i + 2] = (byte)(b * 255);
            data[i + 3] = (byte)(t * 255);
        }
    }

    return new RawImage(width, height, data);
}
```

Рисунок 11 – Исправленная функция

Краткое описание ошибки: Математическая ошибка в процессе преобразования цветового пространства YCbCr в RGB.

Статус ошибки: открыта («Open»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности преобразования цветов при декодировании изображения.

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод Decode с данными сжатого изображения;

3. Запустить программу;

4. Ожидаемый результат: программа должна вывести примерно похожее изображение (значения пикселей примерно равны с учетом погрешности кодирования).

5. Полученный результат: программа выводит изображение со значениями цветов сильно отличающимися от начальных.

Была получена документация на ошибку в модуле «Работа со списками» от Горошникова К.С по которой она была найдена и исправлена (рис.12).

Краткое описание ошибки: Неверное вычисление среднего значения списка.

Статус ошибки: Открыта (Open).

Категория ошибки: Серьезная (Major).

Тестовый случай: Проверка алгоритма функционирования программы.

Описание ошибки:

1. Загрузить программу.
2. Ввести список чисел, например: [1, 2, 3, 4, 5].
3. Вызвать функцию find_average().
4. Полученный результат: 75 (в результате неверного умножения).
5. Ожидаемый результат: 3 (правильное среднее значение).

```
def find_average(lst):
    """Находит среднее значение списка."""
    if not lst:
        return None
    return sum(lst) / len(lst)
```

Рисунок 12 – Исправленный модуль “Работа со списками”

Было проведено итоговое тестирование корректности работы всех функций, которое показало, что все ошибки были исправлены (рис.13).

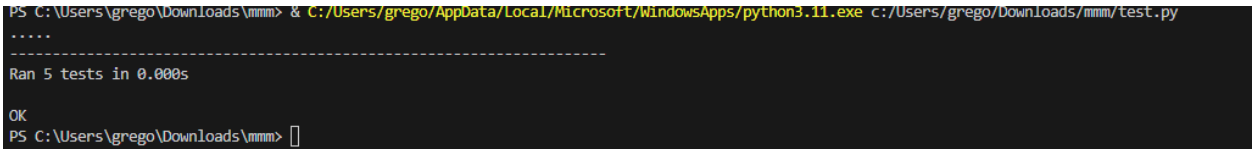


Рисунок 13 – Итоговое тестирование модуля «Работа со списками»

3. Измайлов А.Р. – модуль программы «Кодирование изображения»

Участник команды написал модуль программы «Кодирование изображения».

Была написана документация к модулю.

Описание модуля: Программа по кодированию и декодированию изображения использует несколько функций для преобразования изображения с потерями в сжатую форму бинарного представления и обратно (рис. 14-15). Этот модуль может быть использован как самостоятельный формат кодирования изображения. Данный модуль был передан Четину Г.М для дальнейшего тестирования.

Были реализованы следующие функции:

1) **Побитовая запись в поток:** записывает заданное количество бит в поток.

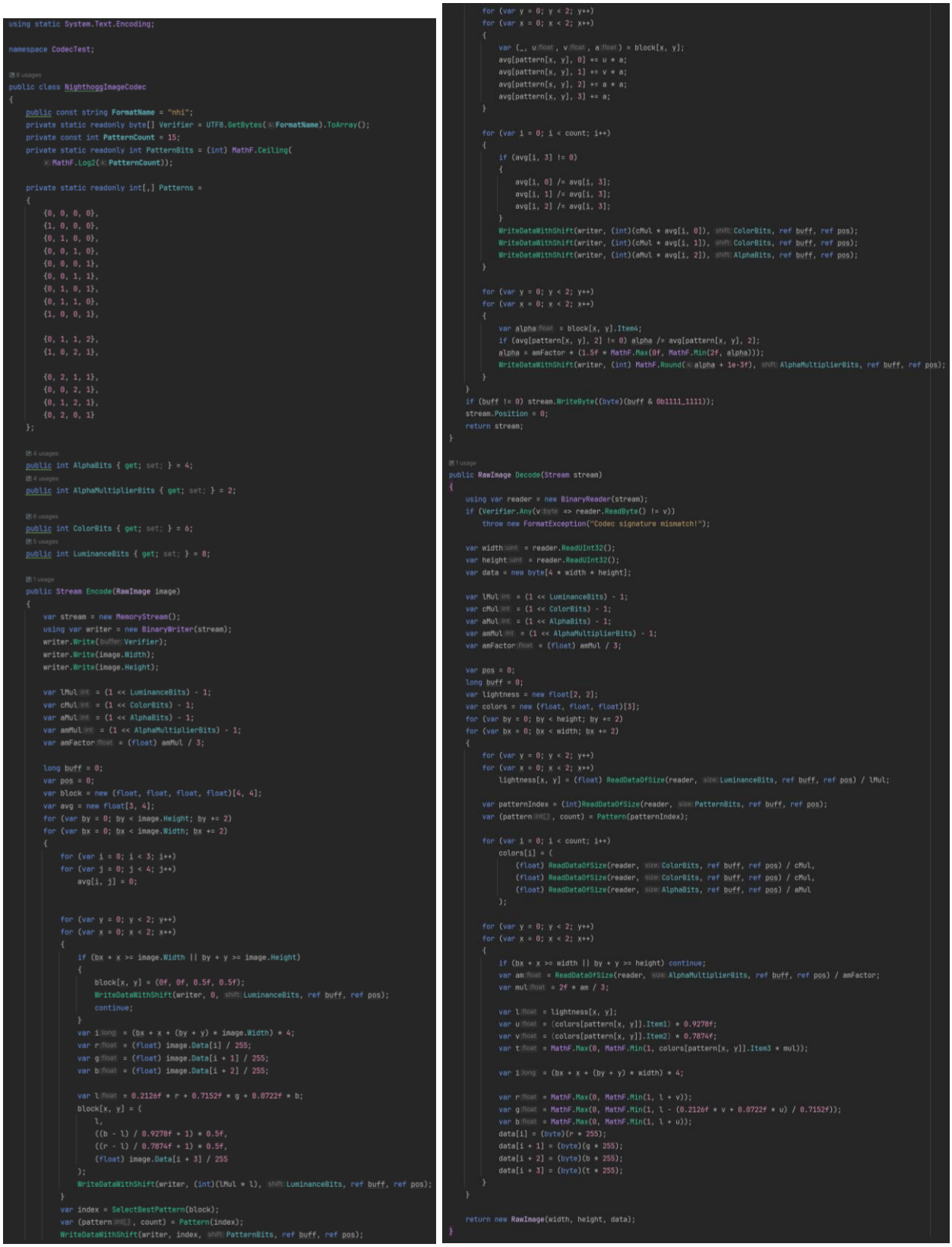
2) **Побитовое чтение из потока:** читает заданное количество бит из потока.

3) **Построитель матрицы паттерна:** позволяет построить матрицу паттерна кодирования, а также количество вариантов цветов для этого паттерна.

4) **Селектор наиболее оптимального паттерна:** ищет минимальный по потерям паттерн для набора пикселей 2 на 2.

5) **Кодировщик изображения:** кодирует RGBA изображение в бинарное представление с незначительными потерями.

6) **Декодировщик изображения:** декодирует RGBA изображение из бинарного представления.



```
using static System.Text.Encoding;

namespace CodecTest;

8 usages
public class NighthoggImageCodec
{
    public const string FormatName = "nhi";
    private static readonly byte[] Verifier = UTF8.GetBytes(s: FormatName).ToArray();
    private const int PatternCount = 15;
    private static readonly int PatternBits = (int) MathF.Ceiling(
        x: MathF.Log2(x: PatternCount));

    private static readonly int[,] Patterns =
    {
        {0, 0, 0, 0},
        {1, 0, 0, 0},
        {0, 1, 0, 0},
        {0, 0, 1, 0},
        {0, 0, 0, 1},
        {0, 0, 1, 1},
        {0, 1, 0, 1},
        {0, 1, 1, 0},
        {1, 0, 0, 1},

        {0, 1, 1, 2},
        {1, 0, 2, 1},

        {0, 2, 1, 1},
        {0, 0, 2, 1},
        {0, 1, 2, 1},
        {0, 2, 0, 1}
    };

    4 usages
    public int AlphaBits { get; set; } = 4;
    4 usages
    public int AlphaMultiplierBits { get; set; } = 2;

    6 usages
    public int ColorBits { get; set; } = 6;
    5 usages
    public int LuminanceBits { get; set; } = 8;

    1 usage
    public Stream Encode(RawImage image)
    {
        var stream = new MemoryStream();
        using var writer = new BinaryWriter(stream);
        writer.Write(buffer: Verifier);
        writer.Write(image.Width);
        writer.Write(image.Height);

        var lMul: int = (1 << LuminanceBits) - 1;
        var cMul: int = (1 << ColorBits) - 1;
        var aMul: int = (1 << AlphaBits) - 1;
        var amMul: int = (1 << AlphaMultiplierBits) - 1;
        var amFactor: float = (float) amMul / 3;

        long buff = 0;
        var pos = 0;
        var block = new (float, float, float, float)[4, 4];
        var avg = new float[3, 4];
        for (var by = 0; by < image.Height; by += 2)
        for (var bx = 0; bx < image.Width; bx += 2)
        {
            for (var i = 0; i < 3; i++)
            for (var j = 0; j < 4; j++)
                avg[i, j] = 0;


            for (var y = 0; y < 2; y++)
            for (var x = 0; x < 2; x++)
            {
                if (bx + x >= image.Width || by + y >= image.Height)
                {
                    block[x, y] = (0f, 0f, 0.5f, 0.5f);
                    WriteDataWithShift(writer, 0, shift: LuminanceBits, ref buff, ref pos);
                    continue;
                }
                var i: long = (bx + x + (by + y) * image.Width) * 4;
                var r: float = (float) image.Data[i] / 255;
                var g: float = (float) image.Data[i + 1] / 255;
                var b: float = (float) image.Data[i + 2] / 255;

                var l: float = 0.2126f * r + 0.7152f * g + 0.0722f * b;
                block[x, y] = (
                    l,
                    ((b - l) / 0.9278f + 1) * 0.5f,
                    ((r - l) / 0.7874f + 1) * 0.5f,
                    (float) image.Data[i + 3] / 255
                );
                WriteDataWithShift(writer, (int)(lMul * l), shift: LuminanceBits, ref buff, ref pos);
            }
            var index = SelectBestPattern(block);
            var (pattern: int[,], count) = Pattern(index);
            WriteDataWithShift(writer, index, shift: PatternBits, ref buff, ref pos);
            for (var y = 0; y < 2; y++)
            for (var x = 0; x < 2; x++)
            {
                var (_, u: float, v: float, a: float) = block[x, y];
                avg[pattern[x, y], 0] += u * a;
                avg[pattern[x, y], 1] += v * a;
                avg[pattern[x, y], 2] += a * a;
                avg[pattern[x, y], 3] += a;
            }

            for (var i = 0; i < count; i++)
            {
                if (avg[i, 3] != 0)
                {
                    avg[i, 0] /= avg[i, 3];
                    avg[i, 1] /= avg[i, 3];
                    avg[i, 2] /= avg[i, 3];
                }
                WriteDataWithShift(writer, (int)(cMul * avg[i, 0]), shift: ColorBits, ref buff, ref pos);
                WriteDataWithShift(writer, (int)(cMul * avg[i, 1]), shift: ColorBits, ref buff, ref pos);
                WriteDataWithShift(writer, (int)(aMul * avg[i, 2]), shift: AlphaBits, ref buff, ref pos);
            }

            for (var y = 0; y < 2; y++)
            for (var x = 0; x < 2; x++)
            {
                var alpha: float = block[x, y].Item4;
                if (avg[pattern[x, y], 2] != 0) alpha /= avg[pattern[x, y], 2];
                alpha = amFactor * (1.5f * MathF.Max(0f, MathF.Min(2f, alpha)));
                WriteDataWithShift(writer, (int) MathF.Round(x: alpha + 1e-3f), shift: AlphaMultiplierBits, ref buff, ref pos);
            }
        }
        if (buff != 0) stream.WriteByte((byte)(buff & 0b1111_1111));
        stream.Position = 0;
        return stream;
    }

    1 usage
    public RawImage Decode(Stream stream)
    {
        using var reader = new BinaryReader(stream);
        if (Verifier.Any(v: byte => reader.ReadByte() != v))
            throw new FormatException("Codec signature mismatch!");

        var width: uint = reader.ReadUInt32();
        var height: uint = reader.ReadUInt32();
        var data = new byte[4 * width * height];

        var lMul: int = (1 << LuminanceBits) - 1;
        var cMul: int = (1 << ColorBits) - 1;
        var aMul: int = (1 << AlphaBits) - 1;
        var amMul: int = (1 << AlphaMultiplierBits) - 1;
        var amFactor: float = (float) amMul / 3;

        var pos = 0;
        long buff = 0;
        var lightness = new float[2, 2];
        var colors = new (float, float, float)[3];
        for (var by = 0; by < height; by += 2)
        for (var bx = 0; bx < width; bx += 2)
        {
            for (var y = 0; y < 2; y++)
            for (var x = 0; x < 2; x++)
                lightness[x, y] = (float) ReadDataOfSize(reader, size: LuminanceBits, ref buff, ref pos) / lMul;

            var patternIndex = (int)ReadDataOfSize(reader, size: PatternBits, ref buff, ref pos);
            var (pattern: int[,], count) = Pattern(patternIndex);

            for (var i = 0; i < count; i++)
                colors[i] = (
                    (float) ReadDataOfSize(reader, size: ColorBits, ref buff, ref pos) / cMul,
                    (float) ReadDataOfSize(reader, size: ColorBits, ref buff, ref pos) / cMul,
                    (float) ReadDataOfSize(reader, size: AlphaBits, ref buff, ref pos) / aMul
                );

            for (var y = 0; y < 2; y++)
            for (var x = 0; x < 2; x++)
            {
                if (bx + x >= width || by + y >= height) continue;
                var am: float = ReadDataOfSize(reader, size: AlphaMultiplierBits, ref buff, ref pos) / amFactor;
                var mul: float = 2f * am / 3;

                var l: float = lightness[x, y];
                var u: float = (colors[pattern[x, y]].Item1) * 0.9278f;
                var v: float = (colors[pattern[x, y]].Item2) * 0.7874f;
                var t: float = MathF.Max(0, MathF.Min(1, colors[pattern[x, y]].Item3 * mul));

                var i: long = (bx + x + (by + y) * width) * 4;

                var r: float = MathF.Max(0, MathF.Min(1, l + v));
                var g: float = MathF.Max(0, MathF.Min(1, l - (0.2126f * v + 0.0722f * u) / 0.7152f));
                var b: float = MathF.Max(0, MathF.Min(1, l + u));
                data[i] = (byte)(r * 255);
                data[i + 1] = (byte)(g * 255);
                data[i + 2] = (byte)(b * 255);
                data[i + 3] = (byte)(t * 255);
            }
        }

        return new RawImage(width, height, data);
    }
```

Рисунок 14 – Код модуля «Кодирование изображения»

```
public static void WriteDataWithShift(BinaryWriter stream,
    long value, int shift, ref long buff, ref int pos)
{
    buff |= value << pos;
    pos += shift;
    while (pos >= 8)
    {
        stream.Write((byte)(buff & 0b1111_1111));
        buff >>>= 8;
        pos -= 8;
    }
}
8 usages
public static long ReadDataOfSize(BinaryReader stream,
    int size, ref long buff, ref int pos)
{
    while (pos < size)
    {
        buff |= (uint)(stream.ReadByte() << pos);
        pos += 8;
    }
    var value:long = buff & ((1 << size) - 1);
    buff >>>= size;
    pos -= size;
    return value;
}

3 usages
public static (int[,], int) Pattern(int index)
{
    var max = 0;
    var pattern = new int[2, 2];
    for (var y = 0; y < 2; y++)
    for (var x = 0; x < 2; x++)
    {
        var value:int = Patterns[index, x + y * 2];
        max = int.Max(max, value);
        pattern[x, y] = value;
    }

    return (pattern, max + 1);
}

2 usages
public static int SelectBestPattern((float, float, float, float)[,] block)
{
    var variants = new float[3, 5];

    var min = float.MaxValue;
    var iMin:int = -1;
    for (var i = 0; i < PatternCount; i++)
    {
        for (var j = 0; j < 3; j++)
        for (var k = 0; k < 5; k++)
            variants[j, k] = 0;

        for (var y = 0; y < 2; y++)
        for (var x = 0; x < 2; x++)
        {
            var pos:int = Patterns[i, x + y * 2];
            var (l:float, u:float, v:float, a:float) = block[x, y];
            variants[pos, 0] += l * a;
            variants[pos, 1] += u * a;
            variants[pos, 2] += v * a;
            variants[pos, 3] += a * a;
            variants[pos, 4] += a;
        }

        for (var j = 0; j < 3; j++)
        {
            if (variants[j, 4] == 0) continue;
            for (var k = 0; k < 4; k++)
                variants[j, k] /= variants[j, 4];
        }

        var error = 0f;
        var div = 0f;
        for (var y = 0; y < 2; y++)
        for (var x = 0; x < 2; x++)
        {
            var pos:int = Patterns[i, x + y * 2];
            var (l:float, u:float, v:float, a:float) = block[x, y];
            var dl:float = variants[pos, 0] - l;
            var du:float = variants[pos, 1] - u;
            var dv:float = variants[pos, 2] - v;
            var da:float = variants[pos, 3] - a;
            error += (dl * dl + du * du + dv * dv + da * da) * a;
            div += a;
        }

        if (div != 0) error /= div;

        if (error >= min) continue;
        min = error;
        iMin = i;
    }


    return iMin;
}
}
```

Рисунки 15 – Код модуля «Кодирование изображения»

В качестве практики по тестированию участнику был передан модуль программы «Работа с геометрическим калькулятором» от Пашкина В.П. Были написаны модульные тесты с помощью библиотеки NUnit, для проверки корректности функций модуля (рис.16).

```
using NUnit.Framework;

namespace MyProject.Tests
{
    Ссылок: 0
    public class GeometryTests
    {
        Ссылок: 6
        private Geometry _geometry;
        [SetUp]
        Ссылок: 0
        public void Setup()
        {
            _geometry = new Geometry();
        }
        [Test]
        Ссылок: 0
        public void SquarePerimeter()
        {
            double result = _geometry.SquarePerimeter(4);
            Assert.AreEqual(16, result);
        }
        [Test]
        Ссылок: 0
        public void SquareArea()
        {
            double result = _geometry.SquareArea(4);
            Assert.AreEqual(16, result);
        }
        [Test]
        Ссылок: 0
        public void CircleCircumference()
        {
            double result = _geometry.CircleCircumference(3);
            Assert.AreEqual(2 * Math.PI * 3, result, 0.0001);
        }
        [Test]
        Ссылок: 0
        public void CircleArea()
        {
            double result = _geometry.CircleArea(3);
            Assert.AreEqual(Math.PI * 3 * 3, result, 0.0001);
        }
        [Test]
        Ссылок: 0
        public void RectanglePerimeter()
        {
            double result = _geometry.RectanglePerimeter(5, 3);
            Assert.AreEqual(16, result);
        }
    }
}
```

Рисунок 16 – Модульные тесты модуля «Работа с геом. калькулятором»

В результате проверки, один из тестов вызвал ошибку в коде.

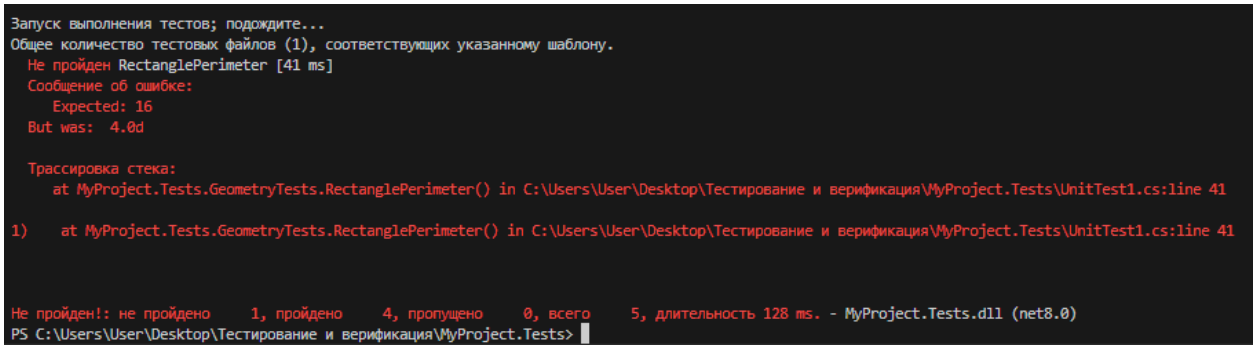


Рисунок 17 – Провал теста на получение периметра прямоугольника

Ошибка происходит в функции RectanglePerimeter(double leanth, double width), когда ей передаётся длина и ширина прямоугольника. Была исправлена данная ошибка и передана участнику Пашкину В.П. для итогового тестирования (рис.18).

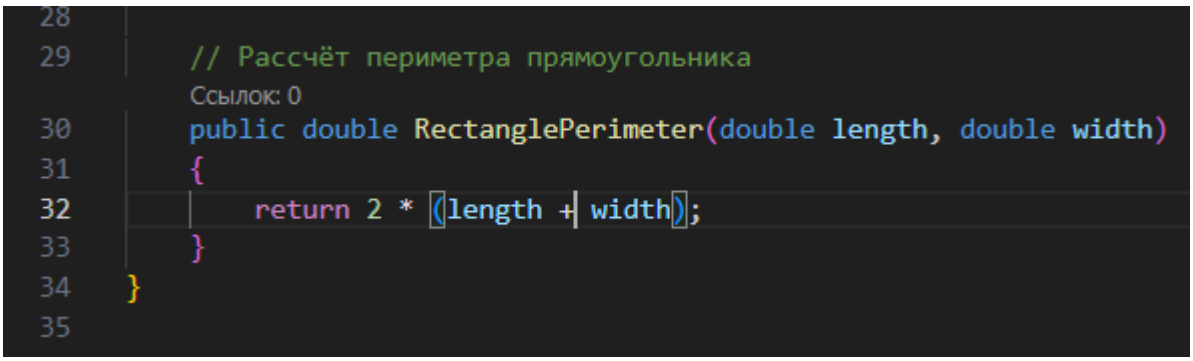


Рисунок 18 – Исправленная функция

Краткое описание ошибки: Неправильный расчет периметра прямоугольника

Статус ошибки: открыта («Open»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности работы функции расчета периметра прямоугольника.

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод RectanglePerimeter() и передать методу ширину и длину прямоугольника;
3. Запустить программу;

4. Ожидаемый результат: программа должна вывести число 16.
5. Полученный результат: программа выводит число 4.

Была получена документация на ошибку в модуле «Кодирование изображения» от Четина Г.М., по которой она была найдена и исправлена (рис.19).

Краткое описание ошибки: Математическая ошибка в процессе преобразования цветового пространства YCbCr в RGB.

Статус ошибки: закрыта («Closed»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности преобразования цветов при декодировании изображения.

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод Decode с данными сжатого изображения;
3. Запустить программу;
4. Ожидаемый результат: программа должна вывести примерно похожее изображение (значения пикселей примерно равны с учетом погрешности кодирования).
5. Полученный результат: программа выводит изображение со значениями цветов сильно отличающимися от начальных.

Было проведено итоговое тестирование корректности работы всех функций, которое показало, что все ошибки были исправлены (рис.20).

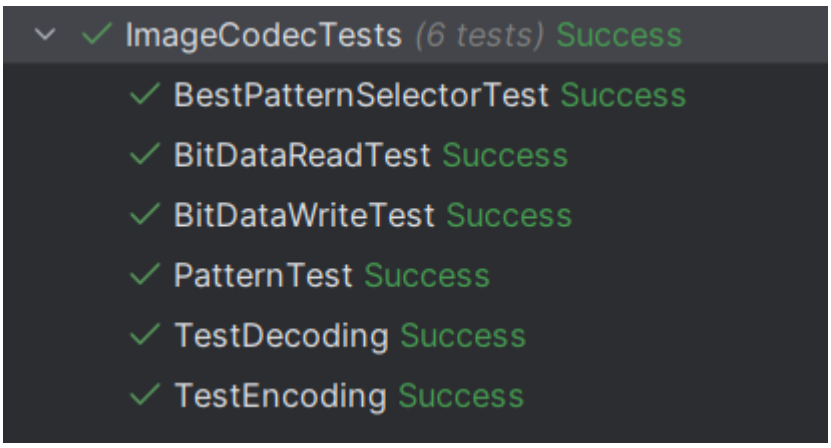


Рисунок 20 – Итоговое тестирование модуля «Кодирование изображения»

4. Пашкин В.П. – модуль программы «Работа с геометрическим калькулятором»

Участник команды написал модуль программы «Работа с геометрическим калькулятором».

Была написана документация к модулю.

Программа по работе с геометрическими фигурами использует несколько математических функций для вычисления периметров, площадей и других параметров для различных фигур (рис. 21). Каждая фигура обрабатывается с использованием методов, специально разработанных для неё. Данный модуль был передан Измайлову А.Р. для дальнейшего тестирования.

Были реализованы следующие функции:

1) **Расчёт периметра квадрата**: вычисляет периметр квадрата по длине его стороны.

2) **Расчёт площади квадрата**: вычисляет площадь квадрата по длине его стороны.

3) **Расчёт длины окружности**: вычисляет длину окружности по радиусу.

4) **Расчёт площади окружности**: вычисляет площадь окружности по радиусу.

5) **Расчёт периметра прямоугольника**: вычисляет периметр прямоугольника по длине и ширине, но в коде заложена ошибка — вместо суммы сторон происходит вычитание, что приводит к некорректному результату.

```
using System;

Ссылок: 0
public class Geometry
{
    // Рассчёт периметра квадрата
    Ссылок: 0
    public double SquarePerimeter(double side)
    {
        return 4 * side;
    }

    // Рассчёт площади квадрата
    Ссылок: 0
    public double SquareArea(double side)
    {
        return side * side;
    }

    // Рассчёт длины окружности
    Ссылок: 0
    public double CircleCircumference(double radius)
    {
        return 2 * Math.PI * radius;
    }

    // Рассчёт площади окружности
    Ссылок: 0
    public double CircleArea(double radius)
    {
        return Math.PI * radius * radius;
    }

    // Рассчёт периметра прямоугольника
    Ссылок: 0
    public double RectanglePerimeter(double length, double width)
    {
        return 2 * (length - width);
    }
}

```

Рисунок 21 – Код модуля «Работа с геометрическим калькулятором»

В качестве практики по тестированию участнику был передан модуль программы «Работа с хеш-таблицой» от Горошникова К.С. Были написаны модульные тесты с помощью библиотеки JUnit, для проверки корректности функций модуля (рис. 22).

```
public class AppTest
{
    private StudentDatabase db;
    @BeforeEach
    public void setup() {
        db = new StudentDatabase();
        db.addStudent(name:"Алексей", studentID:"101", new double[]{4.0, 3.5, 5.0});
        db.addStudent(name:"Мария", studentID:"102", new double[]{3.0, 4.0, 3.5}); }
    @Test
    public void testAddStudent() {
        db.addStudent(name:"Иван", studentID:"103", new double[]{5.0, 4.5, 5.0});
        Student student = db.findStudent(studentID:"103");
        assertNotNull(student, "Студент должен был быть добавлен и найден");
        assertEquals("Иван", student.name, "Имя студента должно быть Иван"); }
    @Test
    public void testFindStudent() {
        Student student = db.findStudent(studentID:"101");
        assertNotNull(student, "Студент должен быть найден");
        assertEquals("Алексей", student.name, "Имя студента должно быть Алексей"); }
    @Test
    public void testFindNonExistentStudent() {
        Student student = db.findStudent(studentID:"999");
        assertNull(student, "Студент не должен быть найден"); }
    @Test
    void testRemoveNonExistentStudent() {
        String nonExistentID = "999";
        boolean result = db.removeStudent(nonExistentID);
        assertFalse(result, "Удаление несуществующего студента должно возвращать false"); }
    @Test
    void testRemoveExistingStudent() {
        boolean result = db.removeStudent(studentID:"101");
        assertTrue(result, "Удаление существующего студента должно возвращать true"); }
    @Test
    public void testUpdateStudentGrades() {
        db.updateStudent(studentID:"101", new double[]{5.0, 4.5, 5.0});
        Student student = db.findStudent(studentID:"101");
        assertNotNull(student, "Студент должен быть найден после обновления");
        assertArrayEquals(new double[]{5.0, 4.5, 5.0}, student.grades, "Оценки студента должны быть обновлены");
    @Test
    public void testUpdateNonExistentStudent() {
        db.updateStudent(studentID:"999", new double[]{4.0, 3.0});
        assertNull(db.findStudent(studentID:"999"), "Студент не должен быть найден или обновлен"); }
    @Test
    public void testSortStudentsByAverageGrade() {
        db.addStudent(name:"Иван", studentID:"103", new double[]{5.0, 4.5, 5.0});
        db.sortStudentsByAverageGrade(); }
}
```

Рисунок 22 – Модульные тесты модуля «Работа с хеш-таблицой»

В результате проверки, один из тестов вызвал ошибку в коде.

```
30      @Test
31      void testRemoveNonExistentStudent() {    Expected [false] but was [true]

Expected [false] but was [true] testRemoveNonExistentStudent()

Expected                                                    Actual
-false                                                      +true
```

Рисунок 23 – Провал теста на удаление записи

Ошибка происходит в функции removeStudent(String id), когда ей передаётся несуществующий id студента. Была исправлена данная ошибка и передана участнику Горошникову К.С. для итогового тестирования (рис.24).

```
public boolean removeStudent(String studentID) {
    if (studentDB.containsKey(studentID)) { // Проверяем, есть ли студент с таким ID
        studentDB.remove(studentID);
        System.out.println("Студент с ID " + studentID + " удалён.");
        return true; // Удаление прошло успешно
    } else {
        System.out.println(x:"Студент с таким ID не найден.");
        return false; // Студент не найден
    }
}
```

Рисунок 24 – Исправленная функция

Краткое описание ошибки: Отсутствие проверки на наличие студента в базе данных при удалении.

Статус ошибки: открыта («Open»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности работы функции удаления студента.

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод removeStudent с несуществующим studentID;
3. Запустить программу;
4. Ожидаемый результат: программа должна вывести сообщение “Студент с таким ID не найден” и вернуть false.
5. Полученный результат: программа выводит сообщение “Студент с ID … удален” и возвращает true.

Была получена документация на ошибку в модуле «Работа с геометрическим калькулятором» от Измайлова А.Р. по которой она была найдена и исправлена (рис.25).

Краткое описание ошибки: Неправильный расчет периметра прямоугольника

Статус ошибки: закрыта («Open»).

Категория ошибки: серьёзная («Major»).

Тестовый случай: проверка корректности работы функции расчета периметра прямоугольника

Описание ошибки:

1. Загрузить программу;
2. Вызвать метод RectanglePerimeter() и передать методу ширину и длину прямоугольника;
3. Запустить программу;
4. Ожидаемый результат: программа должна вывести число 16.
5. Полученный результат: программа выводит число 4.

```
28
29      // Рассчёт периметра прямоугольника
        Ссылок: 0
30      public double RectanglePerimeter(double length, double width)
31      {
32          return 2 * (length + width);
33      }
34  }
35
```

Рисунок 25 – Исправленный модуль “Работа с геометрическим калькулятором”

Было проведено итоговое тестирование корректности работы всех функций, которое показало, что все ошибки были исправлены (рис.26).

```
Запуск выполнения тестов; подождите...
Общее количество тестовых файлов (1), соответствующих указанному шаблону.

Общее количество тестовых файлов (1), соответствующих указанному шаблону.


Пройден!   : не пройдено     0, пройдено     5, пропущено     0, всего     5, длительность 36 ms. - MyProject.Tests.dll (net8.0)
PS C:\Users\User\Desktop\Тестирование и верификация\MyProject.Tests>
```

Рисунок 26 – Итоговое тестирование модуля «Работа с хеш-таблицей»

Заключение

В ходе работы было изучена концепция модульного тестирования. Каждым участником был создан свой модуль программы, в который была заложена ошибка, которую необходимо было найти. Модули были переданы участниками внутри команды для тестирования. Для нахождения ошибок в модулях были написаны модульные тесты через unittest, JUnit и NUnit. После нахождения ошибок они были исправлены и переданы обратно для проверки работоспособности. По результатам работы были выполнены все задачи и удовлетворены все требования к тестированию.