Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №1.

Тема работы: «Основы автоматизации процесса

тестирования»

Наименование дисциплины: «Методы объектно-ориентированного проектирования»

Подготовил:

студент группы 23-КБ-ПИ1

Худяков Дмитрий Русланович

Проверил:

Косолапов П. А.

Краснодар

2025

Вариант 4

Сортировка Шелла (Shell sort)



Основной код:  
using System;

namespace STLab01

{

public class ShellSort

{

public static void Sort(int[] arr)

{

if (arr == null)

{

return;

}

int n = arr.Length;

for (int gap = n / 2; gap > 0; gap /= 2)

{

for (int i = gap; i < n; i++)

{

int temp = arr[i];

int j = i;

while (j >= gap && arr[j - gap] > temp)

{

arr[j] = arr[j - gap];

j -= gap;

}

arr[j] = temp;

}

}

}

}

}

Код тестов:  
using System;

using NUnit.Framework;

using STLab01;

using NUnit.Framework.Legacy;

namespace STLab01Test

{

[TestFixture]

public class ShellSortTests

{

[Test]

public void Sort\_NullArray\_DoesNotThrowException()

{

int[] arr = null;

ShellSort.Sort(arr);

ClassicAssert.IsNull(arr);

}

[Test]

public void Sort\_EmptyArray\_DoesNotChangeArray()

{

int[] arr = { };

ShellSort.Sort(arr);

ClassicAssert.AreEqual(0, arr.Length);

}

[Test]

public void Sort\_SingleElementArray\_DoesNotChangeArray()

{

int[] arr = { 5 };

ShellSort.Sort(arr);

ClassicAssert.AreEqual(new int[] { 5 }, arr);

}

[Test]

public void Sort\_AlreadySortedArray\_StaysSorted()

{

int[] arr = { 1, 2, 3, 4, 5 };

ShellSort.Sort(arr);

ClassicAssert.AreEqual(new int[] { 1, 2, 3, 4, 5 }, arr);

}

[Test]

public void Sort\_ReverseSortedArray\_BecomesSorted()

{

int[] arr = { 5, 4, 3, 2, 1 };

ShellSort.Sort(arr);

ClassicAssert.AreEqual(new int[] { 1, 2, 3, 4, 5 }, arr);

}

[Test]

public void Sort\_UnsortedArray\_BecomesSorted()

{

int[] arr = { 5, 1, 4, 2, 8 };

ShellSort.Sort(arr);

ClassicAssert.AreEqual(new int[] { 1, 2, 4, 5, 8 }, arr);

}

[Test]

public void Sort\_ArrayWithDuplicateValues\_BecomesSorted()

{

int[] arr = { 5, 1, 4, 2, 5, 1 };

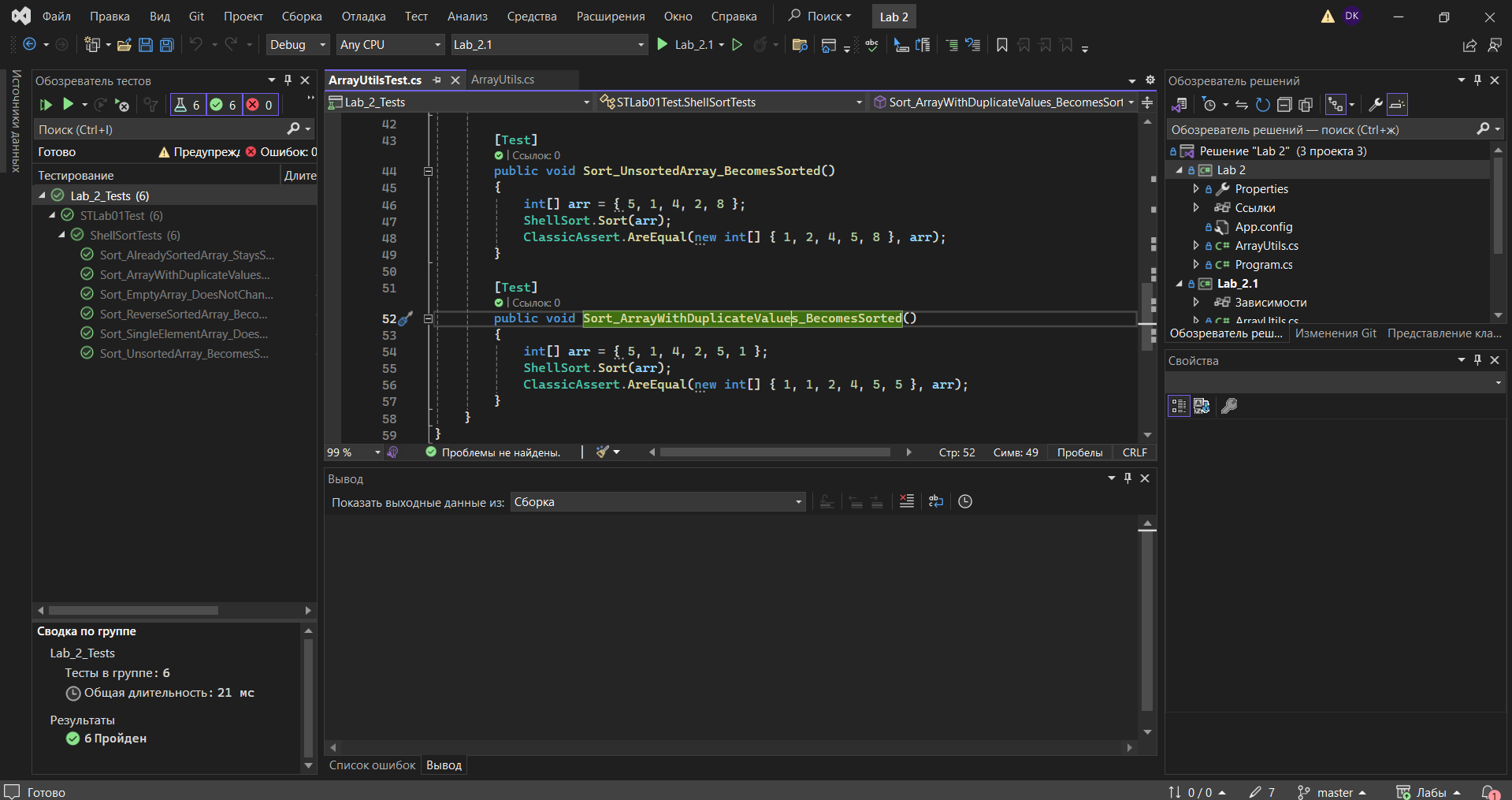
ShellSort.Sort(arr);

ClassicAssert.AreEqual(new int[] { 1, 1, 2, 4, 5, 5 }, arr);

}

}

}

Результаты тестов:  


Контрольные вопросы:  
1 **Для чего предназначена автоматизация тестирования?**  
Автоматизация тестирования предназначена для повышения эффективности проверки программного обеспечения. Она позволяет:

* Автоматически выполнять повторяющиеся тесты.
* Быстро обнаруживать дефекты.
* Сокращать время на регрессионное тестирование.
* Обеспечивать стабильность и воспроизводимость результатов тестирования.

2 **Какие преимущества даёт автоматизация тестирования?**  
Основные преимущества:

* **Экономия времени и ресурсов:** Автоматические тесты выполняются быстрее, чем ручные.
* **Повышение точности:** Исключается влияние человеческого фактора.
* **Быстрая обратная связь:** Позволяет оперативно находить и устранять ошибки.
* **Регулярное тестирование:** Легко интегрируется в процессы непрерывной интеграции (CI/CD).
* **Улучшение качества продукта:** Благодаря постоянной проверке функциональности.

3 **Что такое тестовый набор (Test Suite)?**  
Тестовый набор – это совокупность тестов, сгруппированных вместе для удобства их организации и запуска. Он может объединять тесты, проверяющие определённый модуль, функциональность или сценарий работы системы.

4 **Что такое модульное тестирование?**  
Модульное тестирование – это процесс проверки отдельных компонентов или модулей программы (например, функций, методов, классов) на корректность их работы. Цель – изолированно протестировать каждый элемент, чтобы убедиться в его правильном поведении вне зависимости от других частей системы.

5 **Какие средства предоставляет среда разработки Microsoft Visual Studio для автоматизации тестирования?**  
Visual Studio предлагает:

* **Встроенный фреймворк MSTest** для создания и запуска тестов.
* **Интеграцию с сторонними фреймворками** (например, NUnit, xUnit).
* **Инструменты для анализа покрытия кода тестами.**
* **Средства для интеграции с CI/CD-процессами.**
* **Удобный интерфейс для просмотра результатов тестирования и отладки тестовых сценариев.**

6 **Опишите NUnit Testing Framework.**  
NUnit – это популярный фреймворк для модульного тестирования в .NET, который позволяет:

* Писать тесты с использованием специальных атрибутов (например, [TestFixture] для классов и [Test] для методов).
* Использовать различные методы из класса Assert для проверки условий.
* Организовывать тесты в наборы и выполнять их как из Visual Studio, так и через консоль.
* Интегрироваться с системами CI/CD и другими инструментами разработки.

7 **Как создать новый проект модульного тестирования?**  
Для создания нового проекта модульного тестирования в Visual Studio:

* Выберите пункт **«Создать новый проект»**.
* Найдите и выберите шаблон **«Unit Test Project»** (либо соответствующий шаблон для выбранного тестового фреймворка, например, NUnit Test Project).
* Укажите имя и расположение проекта, затем нажмите **«Создать»**.

8 **Для чего используется атрибут [TestFixture()]?**  
Атрибут [TestFixture] используется для обозначения класса, содержащего тестовые методы. Он сигнализирует фреймворку (например, NUnit), что данный класс является контейнером для тестов и должен быть проанализирован и выполнен.

9 **Для чего используется атрибут [Test()]?**  
Атрибут [Test] применяется для обозначения метода, который является тестовым. Он сообщает фреймворку, что данный метод необходимо запустить в рамках тестирования, и что его результат (успех или провал) будет учитываться.

10 **Как написать тест, проверяющий наличие исключения при выполнении тестируемого метода?**  
Один из распространённых способов – использование метода Assert.Throws<T>().

[Test]

public void TestMethod\_ThrowsArgumentException()

{

// Проверяем, что метод вызывает исключение типа ArgumentException

Assert.Throws<ArgumentException>(() => {

// Вызов тестируемого метода, который должен выбросить исключение

TestedMethod();

});

}

11 **Опишите класс Assert и его основные методы.**  
Класс Assert предоставляет методы для проверки различных условий в тестах. Основные методы:

* Assert.AreEqual(expected, actual) – проверяет, что два значения равны.
* Assert.AreNotEqual(notExpected, actual) – проверяет, что два значения не равны.
* Assert.IsTrue(condition) – проверяет, что условие истинно.
* Assert.IsFalse(condition) – проверяет, что условие ложно.
* Assert.IsNull(object) – проверяет, что объект равен null.
* Assert.IsNotNull(object) – проверяет, что объект не равен null.
* Assert.Throws<TException>(TestDelegate code) – проверяет, что при выполнении делегата выбрасывается исключение указанного типа.

Эти методы помогают сравнивать ожидаемые и фактические результаты, что является основой для оценки корректности работы тестируемого кода.

12 **Какие результаты теста могут быть получены при запуске?**  
При запуске тестов могут быть получены следующие результаты:

* **Passed (Пройден):** Тест выполнен успешно, все проверки прошли.
* **Failed (Провален):** Тест выявил несоответствие между ожидаемым и фактическим результатом.
* **Inconclusive (Неопределённый):** Тест не смог определить результат (например, из-за отсутствия необходимых данных или неполной реализации).
* **Skipped/Ignored (Пропущен):** Тест не был запущен (например, если для него установлен атрибут Ignore).
* **Error (Ошибка):** При выполнении теста произошла неожиданная ошибка (например, исключение, не предусмотренное в тесте)