Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Современные технологии программирования

Лабораторная работа №2

Модульное тестирование программ на языке C# средствами Visual Studio

Вариант 1

Выполнил:

студент гр.ИП-213 Дмитриев Антон Александрович
ФИО студента

«__» _____ 2025 г.

Проверил:

Преподаватель

ФИО преподавателя

«__» _____ 2025 г. Оценка_____

Новосибирск 2025 г.

1. **Задание**

Разработайте на языке C# класс, содержащий функции в соответствии с вариантом задания.

Разработайте тестовые наборы данных для тестирования функций класса, по критерию C1.

Протестируйте созданный класс с помощью средств автоматизации модульного тестирования Visual Studio.

Проанализируйте результаты выполненных тестов по объёму покрытия тестируемого кода.

Напишите отчёт о результатах проделанной работы.

Вариант 1

Поиск минимума из двух чисел.

Функция получает двумерный массив вещественных переменных A.

Отыскивает и возвращает максимальное значение компонентов массива.

Функция получает двумерный массив вещественных переменных A.

Отыскивает и возвращает максимальное значение компонентов массива, лежащих на и выше побочной диагонали.

2. Тестовые наборы данных для тестирования класса.

1. Метод FindMin - Поиск минимума из двух чисел

№ теста	Входные данные	Ожидаемый результат	Покрываемая ветвь
1	a = 2.5, b = 3.7	2.5	a < b = true
2	a = 5.2, b = 3.1	3.1	a < b = false (b-меньше)
3	a = 4.0, b = 4.0	4.0	a < b = false (равны)

2. Метод FindMaxInArray - Поиск максимума в массиве

Тест 4: Исключение - null массив

Входные данные: array = null

Ожидаемый результат: ArgumentNullException

Покрываемая ветвь: A == null = true

Тест 5: Исключение - пустой массив

Входные данные: array = new double[0, 0]

Ожидаемый результат: ArgumentException

Покрываемая ветвь: A.Length == 0 = true

Тест 6: Массив 1×1

Входные данные:

{ 7.5 }

Ожидаемый результат: 7.5

Покрываемая ветвь: циклы не выполняются

Тест 7: Максимум в первом элементе

Входные данные:

{ 10.0, 2.0 }

{ 3.0, 4.0 }

Ожидаемый результат: 10.0

Покрываемая ветвь: A[i,j] > max = false для всех кроме первого

Тест 8: Максимум в последнем элементе

Входные данные:

{ 1.0, 2.0 }

{ 3.0, 15.0 }

Ожидаемый результат: 15.0

Покрываемая ветвь: A[i,j] > max = true для последнего элемента

Тест 9: Максимум в середине

Входные данные:

{ 1.0, 2.0, 1.0 }

{ 2.0, 20.0, 2.0 }

{ 1.0, 2.0, 1.0 }

Ожидаемый результат: 20.0

Покрываемая ветвь: A[i,j] > max = true для среднего элемента

3. Метод FindMaxAboveSecondaryDiagonal - Максимум на и выше побочной диагонали

Тест 10: Исключение - null массив

Входные данные: array = null

Ожидаемый результат: ArgumentNullException

Покрываемая ветвь: A == null = true

Тест 11: Исключение - пустой массив

Входные данные: array = new double[0, 0]

Ожидаемый результат: ArgumentException

Покрываемая ветвь: A.Length == 0 = true

Тест 12: Исключение - неквадратный массив

Входные данные:

{ 1.0, 2.0, 3.0 }

{ 4.0, 5.0, 6.0 }

Ожидаемый результат: ArgumentException

Покрываемая ветвь: rows != cols = true

Тест 13: Массив 1×1

Входные данные:

{ 5.0 }

Ожидаемый результат: 5.0

Покрываемая ветвь: !found = true, i+j ≤ rows-1 = true

Тест 14: Максимум на побочной диагонали

Входные данные:

{ 1.0, 2.0, 3.0 }

{ 4.0, 5.0, 6.0 }

{ 7.0, 8.0, 9.0 }

Область поиска (i+j ≤ 2):

Ожидаемый результат: 7.0

Покрываемая ветвь: i+j ≤ rows-1 = true для диагонального элемента

Тест 15: Максимум выше диагонали

Входные данные:

{ 1.0, 15.0, 2.0 }

{ 4.0, 5.0, 6.0 }

{ 7.0, 8.0, 9.0 }

Область поиска (i+j ≤ 2):

Ожидаемый результат: 15.0

Покрываемая ветвь: i+j ≤ rows-1 = true для элемента выше диагонали

Тест 16: Отрицательные значения

Входные данные:

{ -5.0, -2.0 }

{ -3.0, -1.0 }

Область поиска (i+j ≤ 1):

Ожидаемый результат: -2.0

Покрываемая ветвь: работа с отрицательными числами

Тест 17: Многократное изменение максимума

Входные данные:

{ 1.0, 10.0, 3.0 }

{ 4.0, 15.0, 5.0 }

{ 7.0, 8.0, 9.0 }

Область поиска (i+j ≤ 2):

Ожидаемый результат: 15.0

Покрываемая ветвь: A[i,j] > max = true несколько раз

Тест 18: Первый элемент - максимум

Входные данные:

{ 100.0, 2.0, 3.0 }

{ 4.0,   5.0, 6.0 }

{ 7.0,   8.0, 9.0 }

Область поиска (i+j ≤ 2):

Ожидаемый результат: 100.0

Покрываемая ветвь: !found = true, затем A[i,j] > max = false

3. Исходные тексты программ на языке C#.

Class.cs:

```
using System;

namespace MathOperations
{
   public static class ArrayProcessor
   {
      // 1. Поиск минимума из двух чисел
      public static double FindMin(double a, double b)
      {
         //if (a < b)
         //{
         //    return a;
         //}
         //else
         //{
         //    return b;
         //}

         return a < b ? a : b;
      }

      // 2. Поиск максимального значения во всем массиве
      public static double FindMaxInArray(double[,] A)
      {
         if (A == null)
            throw new ArgumentNullException(nameof(A), "Массив не может быть null");

         if (A.Length == 0)
            throw new ArgumentException("Массив не может быть пустым", nameof(A));

         double max = A[0, 0];
         int rows = A.GetLength(0);
         int cols = A.GetLength(1);

         for (int i = 0; i < rows; i++)
         {
            for (int j = 0; j < cols; j++)
            {
               if (A[i, j] > max)
               {
                  max = A[i, j];
               }
```

```
            }
        }

        return max;
    }

    // 3. Поиск максимального значения на и выше побочной диагонали
    public static double FindMaxAboveSecondaryDiagonal(double[,] A)
    {
        if (A == null)
            throw new ArgumentNullException(nameof(A), "Массив не может быть null");

        if (A.Length == 0)
            throw new ArgumentException("Массив не может быть пустым", nameof(A));

        int rows = A.GetLength(0);
        int cols = A.GetLength(1);

        if (rows != cols)
            throw new ArgumentException("Массив должен быть квадратным", nameof(A));

        double max = double.MinValue;
        bool found = false;

        for (int i = 0; i < rows; i++)
        {
            for (int j = 0; j < cols; j++)
            {
                // Условие для элементов на и выше побочной диагонали:
                // i + j <= rows - 1
                if (i + j <= rows - 1)
                {
                    if (!found || A[i, j] > max)
                    {
                        max = A[i, j];
                        found = true;
                    }
                }
            }
        }

        if (!found)
            throw new InvalidOperationException("Не найдено элементов на и выше
побочной диагонали");
```

```
            return max;
        }
    }
}
```

Tests.cs:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MathOperations.Tests
{
    [TestClass]
    public class ArrayProcessorC1Tests
    {
        // 1. Поиск минимума из двух чисел

        [TestMethod]
        public void FindMin_FirstLessThanSecond_ReturnsFirst()
        {
            // Ветка: a < b = true
            double a = 2.5, b = 3.7;
            double expected = 2.5;

            double result = ArrayProcessor.FindMin(a, b);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void FindMin_SecondLessThanFirst_ReturnsSecond()
        {
            // Ветка: a < b = false
            double a = 5.2, b = 3.1;
            double expected = 3.1;

            double result = ArrayProcessor.FindMin(a, b);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void FindMin_EqualValues_ReturnsAny()
        {
            // Ветка: a < b = false (при равенстве)
            double a = 4.0, b = 4.0;
            double expected = 4.0;
```

```
    double result = ArrayProcessor.FindMin(a, b);

    Assert.AreEqual(expected, result);
}

// 2. Поиск максимального значения во всем массиве

[TestMethod]
[ExpectedException(typeof(ArgumentNullException))]
public void FindMaxInArray_NullArray_ThrowsException()
{
    // Ветка: A == null = true
    double[,] array = null;

    ArrayProcessor.FindMaxInArray(array);
}

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void FindMaxInArray_EmptyArray_ThrowsException()
{
    // Ветка: A.Length == 0 = true
    double[,] array = new double[0, 0];

    ArrayProcessor.FindMaxInArray(array);
}

[TestMethod]
public void FindMaxInArray_SingleElement_ReturnsElement()
{
    // Ветки:
    // - Циклы не выполняются (массив 1x1)
    double[,] array = { { 7.5 } };
    double expected = 7.5;

    double result = ArrayProcessor.FindMaxInArray(array);

    Assert.AreEqual(expected, result);
}

[TestMethod]
public void FindMaxInArray_MaxInFirstElement_ReturnsFirst()
{
    // Ветка: A[i, j] > max = false для всех элементов кроме первого
```

```
    double[,] array = {
        { 10.0, 2.0 },
        { 3.0, 4.0 }
    };
    double expected = 10.0;

    double result = ArrayProcessor.FindMaxInArray(array);

    Assert.AreEqual(expected, result);
}

[TestMethod]
public void FindMaxInArray_MaxInLastElement_ReturnsLast()
{
    // Ветка: A[i, j] > max = true для последнего элемента
    double[,] array = {
        { 1.0, 2.0 },
        { 3.0, 15.0 }
    };
    double expected = 15.0;

    double result = ArrayProcessor.FindMaxInArray(array);

    Assert.AreEqual(expected, result);
}

[TestMethod]
public void FindMaxInArray_MaxInMiddle_ReturnsMiddle()
{
    // Ветка: A[i, j] > max = true для среднего элемента
    double[,] array = {
        { 1.0, 2.0, 1.0 },
        { 2.0, 20.0, 2.0 },
        { 1.0, 2.0, 1.0 }
    };
    double expected = 20.0;

    double result = ArrayProcessor.FindMaxInArray(array);

    Assert.AreEqual(expected, result);
}

// 3. Поиск максимального значения на и выше побочной диагонали

[TestMethod]
```

```
[ExpectedException(typeof(ArgumentNullException))]
public void FindMaxAboveSecondaryDiagonal_NullArray_ThrowsException()
{
   // Ветка: A == null = true
   double[,] array = null;

   ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);
}

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void FindMaxAboveSecondaryDiagonal_EmptyArray_ThrowsException()
{
   // Ветка: A.Length == 0 = true
   double[,] array = new double[0, 0];

   ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);
}

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void FindMaxAboveSecondaryDiagonal_NonSquareArray_ThrowsException()
{
   // Ветка: rows != cols = true
   double[,] array = new double[2, 3];

   ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);
}

[TestMethod]
public void FindMaxAboveSecondaryDiagonal_1x1Array_ReturnsElement()
{
   // Ветки:
   // - i + j <= rows - 1 = true (для единственного элемента)
   // - !found = true (первый элемент)
   double[,] array = { { 5.0 } };
   double expected = 5.0;

   double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

   Assert.AreEqual(expected, result);
}

[TestMethod]
```

```
        public void
FindMaxAboveSecondaryDiagonal_MaxOnDiagonal_ReturnsDiagonalElement()
        {
            // Ветка: i + j <= rows - 1 = true для элементов выше диагонали
            // Максимум находится на самой диагонали
            double[,] array = {
                { 1.0, 2.0, 3.0 },
                { 4.0, 5.0, 6.0 },
                { 7.0, 8.0, 9.0 }
            };
            double expected = 7.0; // Элемент на побочной диагонали

            double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void
FindMaxAboveSecondaryDiagonal_MaxAboveDiagonal_ReturnsAboveElement()
        {
            // Ветка: i + j <= rows - 1 = true для элементов выше диагонали
            // Максимум находится выше диагонали
            double[,] array = {
                { 1.0, 15.0, 2.0 },  // 15.0 - выше диагонали
                { 4.0, 5.0, 6.0 },
                { 7.0, 8.0, 9.0 }
            };
            double expected = 15.0;

            double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        [ExpectedException(typeof(ArgumentException))]
        public void FindMaxAboveSecondaryDiagonal_NonSquareArrayThrowsException()
        {
            // Ветка: Массив должен быть квадратным
            double[,] array = {
                { 1.0, 2.0, 3.0 },
                { 4.0, 5.0, 6.0 }
            };
```

```
            ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

        }

        [TestMethod]
        public void FindMaxAboveSecondaryDiagonal_NegativeValues_WorksCorrectly()
        {
            // Ветка: проверка работы с отрицательными числами
            double[,] array = {
                { -5.0, -2.0 },
                { -3.0, -1.0 }
            };
            double expected = -2.0; // Максимум среди отрицательных

            double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void
FindMaxAboveSecondaryDiagonal_MaxChangesMultipleTimes_CorrectResult()
        {
            // Ветка: A[i, j] > max = true несколько раз в цикле
            double[,] array = {
                { 1.0, 10.0, 3.0 },   // 10.0 становится максимумом
                { 4.0, 15.0, 5.0 },
                { 7.0, 8.0, 9.0 }
            };
            double expected = 15.0;

            double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

            Assert.AreEqual(expected, result);
        }

        [TestMethod]
        public void FindMaxAboveSecondaryDiagonal_FirstElementIsMax_ReturnsFirst()
        {
            // Ветка: !found = true (первый элемент), затем A[i, j] > max = false для
остальных
            double[,] array = {
                { 100.0, 2.0, 3.0 },
                { 4.0, 5.0, 6.0 },
                { 7.0, 8.0, 9.0 }
```

```
            };
            double expected = 100.0;

            double result = ArrayProcessor.FindMaxAboveSecondaryDiagonal(array);

            Assert.AreEqual(expected, result);
        }
    }
}
```

4. Результаты выполнения модульных тестов.

5. Результаты покрытия разработанного кода тестами.

1. FindMin - 100% покрытие

Условия: 1 (тернарный оператор)

Покрытые ветви: a < b = true, a < b = false

Тесты: 3 теста

2. FindMaxInArray - 100% покрытие

Условия: 4

- A == null
- A.Length == 0
- A[i,j] > max
- Циклы for

Покрытые ветви: 8 ветвей

Тесты: 6 тестов

3. FindMaxAboveSecondaryDiagonal - 100% покрытие

Условия: 7

- A == null
- A.Length == 0
- rows != cols
- i + j <= rows - 1
- !found
- A[i,j] > max
- Циклы for

Покрытые ветви: 14 ветвей

Тесты: 9 тестов

6. Выводы по выполненной работе.

В ходе выполнения лабораторной работы были успешно сформированы и закреплены практические навыки разработки модульных тестов для тестирования функций классов на языке C# с использованием средств автоматизации Visual Studio.

В ходе выполнения работы было сделано:

Полное покрытие кода тестами (критерий C1) - достигнуто 100% покрытие ветвей кода, что подтверждает корректность разработанных тестовых сценариев

Комплексное тестирование функциональности - разработаны тесты для всех трёх методов класса:

- FindMin - тестирование поиска минимального значения из двух чисел
- FindMaxInArray - тестирование поиска максимального значения в двумерном массиве
- FindMaxAboveSecondaryDiagonal - тестирование поиска максимума на и выше побочной диагонали

Покрытие граничных условий - тесты включают проверку:

- Пустых массивов и строк
- Нулевых значений
- Исключительных ситуаций
- Корректных и некорректных входных данных

Использование различных подходов тестирования:

- Тестирование нормального выполнения
- Тестирование исключительных ситуаций
- Тестирование граничных значений