Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №2
Вариант 3

Выполнил:
студенты 4 курса
группы ИП-216
Андрущенко Ф.А.
Литвинов А. Е.
Русецкий А. С.

Проверил:
преподаватель кафедры ПМиК
Агалаков Антон Александрович

Новосибирск, 2025 г.

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.

Функции для тестирования:

- Поиск минимума из трёх чисел.
- Функция получает двумерный массив вещественных переменных A. Отыскивает и возвращает сумму значений компонентов массива, у которых сумма значений индексов –
  чётная.
- Функция получает двумерный массив вещественных переменных A. Отыскивает и возвращает максимальное значение компонентов массива, лежащих на и ниже главной
  диагонали

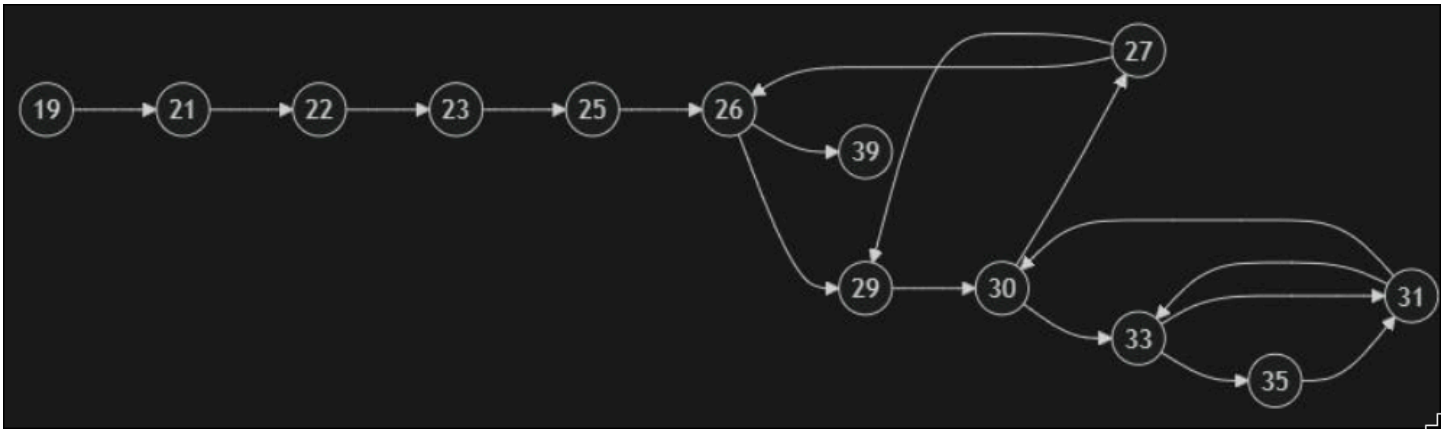
УГП и тестовые наборы данных для тестирования функций класса

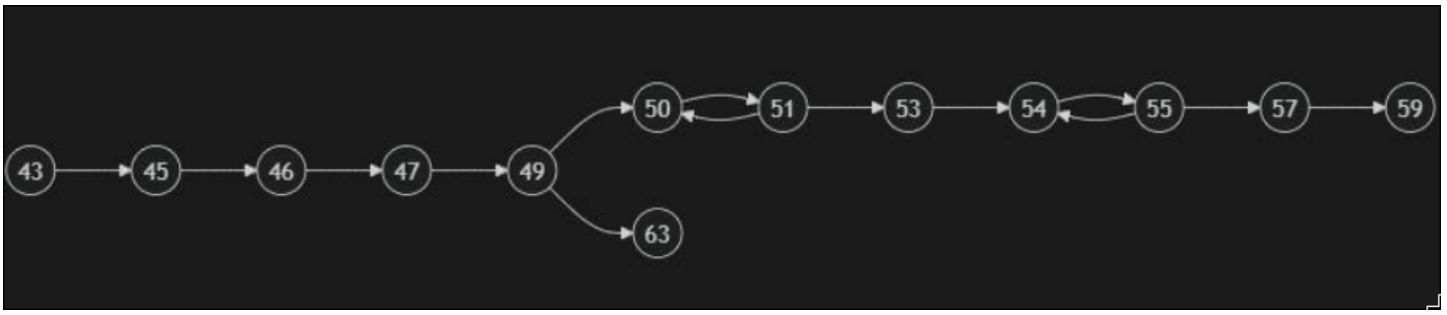
Тестовые данные:

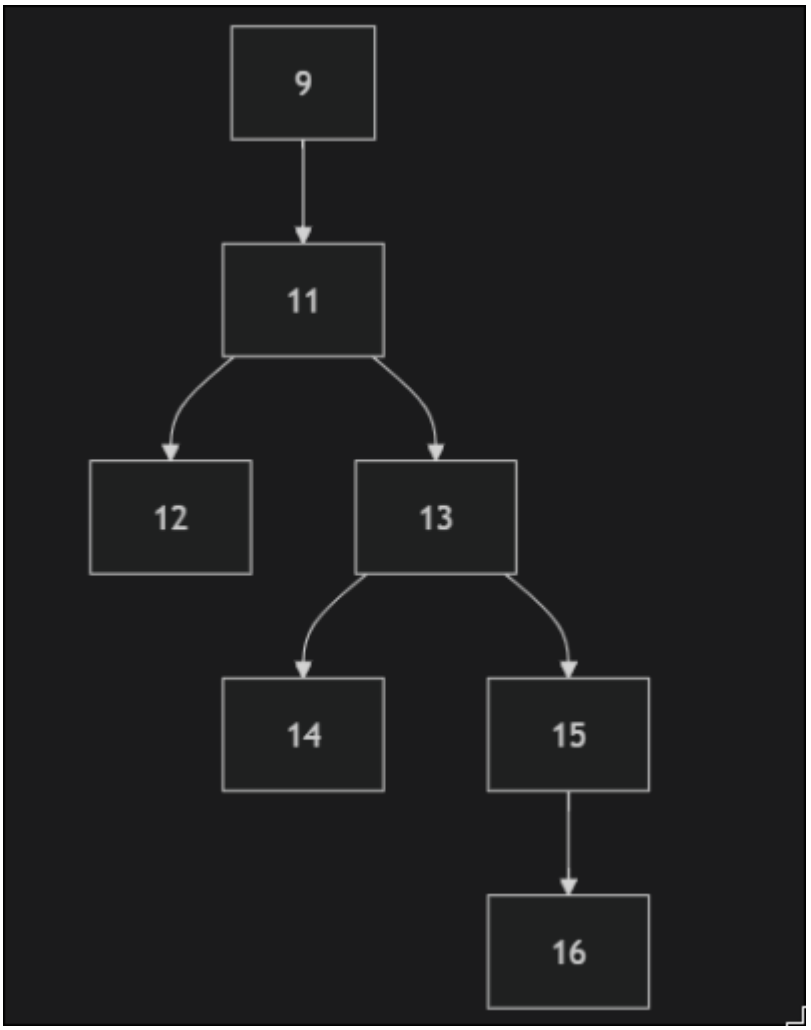
```
using System;

int[][] findMinData = new int[][] {
 new int[] { 5, 2, 8 },
 new int[] { 5, 5, 5 },
 new int[] { 3, 3, 7 },
 new int[] { -5, -2, -8 },
 new int[] { 0, 5, -3 },
 new int[] { 10, 10, 5 }
};

double[][,] sumEvenIndexData = new double[][,] {
 new double[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } },
 new double[,] { { 5.5 } },
 new double[,] { { 1, 2 }, { 3, 4 } },
 new double[,] { { 1, 2, 3, 4 }, { 5, 6, 7, 8 } },
 new double[,] { { 0, 0 }, { 0, 0 } }
};

double[][,] maxDiagonalData = new double[][,] {
 new double[,] { { 10, 20, 30 }, { 40, 50, 60 }, { 70, 80, 90 } },
 new double[,] { { 1, 99, 99 }, { 100, 2, 99 }, { 50, 60, 3 } },
 new double[,] { { 1, 2, 3, 4 }, { 5, 6, 7, 8 }, { 9, 10, 11, 12 } },
 new double[,] { { 7.7 } },
 new double[,] { { 5, 1 }, { 2, 4 } },
 new double[,] { { -5, -2 }, { -8, -1 } }
};
```

```
object[][] multiplyNonZeroData = new object[][] {
 new object[] { new double[] { 1, 2, 3, 4, 5 }, new int[] { 0, 2, 4 }, 15.0 },
 new object[] { new double[] { 1, 0, 3, 0, 5 }, new int[] { 0, 1, 2, 3, 4 }, 15.0 },
 new object[] { new double[] { 0, 0, 0 }, new int[] { 0, 1, 2 }, 0.0 },
 new object[] { new double[] { 2.5, 1.5, 4.0 }, new int[] { 0, 1, 2 }, 15.0 },
 new object[] { new double[] { -2, 3, -4 }, new int[] { 0, 2 }, 8.0 }
};

object[][] findMinElementData = new object[][] {
 new object[] { new int[] { 5, 2, 8, 1, 9 }, (1, 3) },
 new object[] { new int[] { 5, 5, 5, 5 }, (5, 0) },
 new object[] { new int[] { -5, -2, -8, -1 }, (-8, 2) },
 new object[] { new int[] { 0, 0, 0 }, (0, 0) },
 new object[] { new int[] { 15, 8, 23, 4, 42, 7 }, (4, 3) }
};

object[][] reverseArrayData = new object[][] {
 new object[] { new double[] { 1.1, 2.2, 3.3, 4.4 }, new double[] { 4.4, 3.3, 2.2, 1.1 } },
 new object[] { new double[] { 5.5 }, new double[] { 5.5 } },
 new object[] { new double[] {}, new double[] {} },
 new object[] { new double[] { -1.1, -2.2, -3.3 }, new double[] { -3.3, -2.2, -1.1 } },
 new object[] { new double[] { 1, 2, 3, 4, 5 }, new double[] { 5, 4, 3, 2, 1 } }
};

object[][] exceptionData = new object[][] {
 new object[] { null, new int[] { 0, 1 }, typeof(ArgumentNullException) },
 new object[] { new double[] { 1, 2, 3 }, null, typeof(ArgumentNullException) },
 new object[] { new double[] { 1, 2, 3 }, new int[] { 0, 5 },
typeof(IndexOutOfRangeException) },
 new object[] { new double[] { 1, 2, 3 }, new int[] { -1, 0 },
typeof(IndexOutOfRangeException) },
 new object[] { Array.Empty<int>(), typeof(ArgumentException) },
 new object[] { null, typeof(ArgumentNullException) }
};
```

УГП для функций:







Исходные коды программ

Тестируемые функции

```
using System;

namespace ArrayOperations
{
 public class ArrayProcessor
 {
   public static int FindMinOfThree(int a, int b, int c)
   {
     if (a <= b && a <= c)
       return a;
     else if (b <= a && b <= c)
       return b;
```

```
    else
      return c;
  }

  public static double SumElementsWithEvenIndexSum(double[,] A)
  {
    if (A == null)
      throw new ArgumentNullException(nameof(A), "Массив не может быть null");

    double sum = 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 ((i + j) % 2 == 0)
        {
          sum += A[i, j];
        }
      }
    }

    return sum;
  }

  public static double MaxOnAndBelowMainDiagonal(double[,] A)
  {
    if (A == null)
      throw new ArgumentNullException(nameof(A), "Массив не может быть null");

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

    if (rows == 0 || cols == 0)
      throw new ArgumentException("Массив не может быть пустым");

    double max = double.MinValue;

    for (int i = 0; i < rows; i++)
    {
      for (int j = 0; j <= Math.Min(i, cols - 1); j++)
      {
        if (A[i, j] > max)
        {
          max = A[i, j];
        }
      }
    }

    return max;
  }
}
```

```
}
```

Тесты

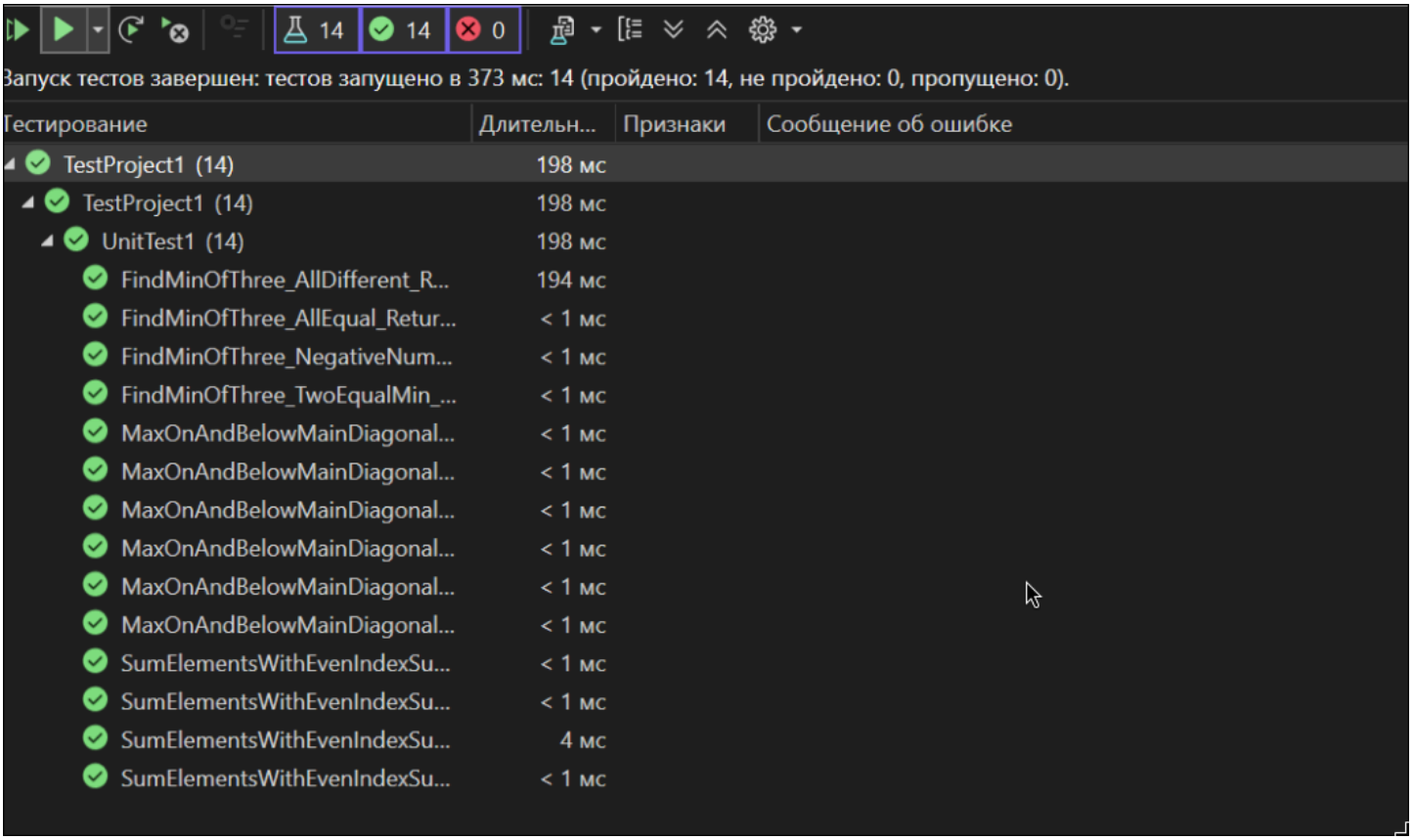
```
using System;
using Xunit;
using ArrayOperations;

namespace TestProject1
{
 public class UnitTest1
 {
   // Тесты для FindMinOfThree
   [Fact]
   public void FindMinOfThree_AllDifferent_ReturnsMin()
   {
     // Arrange
     int a = 5, b = 2, c = 8;

     // Act
     int result = ArrayProcessor.FindMinOfThree(a, b, c);

     // Assert
     Assert.Equal(2, result);
   }

   [Fact]
   public void FindMinOfThree_AllEqual_ReturnsSame()
   {
     // Arrange
     int a = 5, b = 5, c = 5;

     // Act
     int result = ArrayProcessor.FindMinOfThree(a, b, c);

     // Assert
     Assert.Equal(5, result);
   }

   [Fact]
   public void FindMinOfThree_TwoEqualMin_ReturnsMin()
   {
     // Arrange
     int a = 3, b = 3, c = 7;

     // Act
     int result = ArrayProcessor.FindMinOfThree(a, b, c);

     // Assert
     Assert.Equal(3, result);
   }

   [Fact]
```

```
public void FindMinOfThree_NegativeNumbers_ReturnsMin()
{
  // Arrange
  int a = -5, b = -2, c = -8;

  // Act
  int result = ArrayProcessor.FindMinOfThree(a, b, c);

  // Assert
  Assert.Equal(-8, result);
}

// Тесты для SumElementsWithEvenIndexSum
[Fact]
public void SumElementsWithEvenIndexSum_ValidArray_ReturnsCorrectSum()
{
  // Arrange
  double[,] A = {
  { 1, 2, 3 },
  { 4, 5, 6 },
  { 7, 8, 9 }
};

  // Act
  double result = ArrayProcessor.SumElementsWithEvenIndexSum(A);

  // Assert
  Assert.Equal(1 + 3 + 5 + 7 + 9, result); // (0,0)+(0,2)+(1,1)+(2,0)+(2,2)
}

[Fact]
public void SumElementsWithEvenIndexSum_SingleElement_ReturnsElement()
{
  // Arrange
  double[,] A = { { 5.5 } };

  // Act
  double result = ArrayProcessor.SumElementsWithEvenIndexSum(A);

  // Assert
  Assert.Equal(5.5, result);
}

[Fact]
public void SumElementsWithEvenIndexSum_EmptyArray_ReturnsZero()
{
  // Arrange
  double[,] A = new double[0, 0];

  // Act
  double result = ArrayProcessor.SumElementsWithEvenIndexSum(A);

  // Assert
  Assert.Equal(0, result);
```

```
    }

    [Fact]
    public void SumElementsWithEvenIndexSum_NullArray_ThrowsException()
    {
      // Arrange
      double[,] A = null;

      // Act & Assert
      Assert.Throws<ArgumentNullException>(() =>
ArrayProcessor.SumElementsWithEvenIndexSum(A));
    }

    // Тесты для MaxOnAndBelowMainDiagonal
    [Fact]
    public void MaxOnAndBelowMainDiagonal_ValidArray_ReturnsCorrectMax()
    {
      // Arrange
      double[,] A = {
      { 10, 20, 30 },
      { 40, 50, 60 },
      { 70, 80, 90 }
    };

      // Act
      double result = ArrayProcessor.MaxOnAndBelowMainDiagonal(A);

      // Assert
      Assert.Equal(90, result); // Максимум среди 10,40,50,70,80,90
    }

    [Fact]
    public void MaxOnAndBelowMainDiagonal_MaxBelowDiagonal_ReturnsCorrectMax()
    {
      // Arrange
      double[,] A = {
      { 1, 99, 99 },
      { 100, 2, 99 },
      { 50, 60, 3 }
    };

      // Act
      double result = ArrayProcessor.MaxOnAndBelowMainDiagonal(A);

      // Assert
      Assert.Equal(100, result); // Максимум ниже диагонали
    }

    [Fact]
    public void MaxOnAndBelowMainDiagonal_NonSquareArray_ReturnsCorrectMax()
    {
      // Arrange
      double[,] A = {
      { 1, 2, 3, 4 },
```

```
      { 5, 6, 7, 8 },
      { 9, 10, 11, 12 }
    };

      // Act
      double result = ArrayProcessor.MaxOnAndBelowMainDiagonal(A);

      // Assert
      Assert.Equal(11, result); // Максимум среди 1,5,6,9,10,11
    }

    [Fact]
    public void MaxOnAndBelowMainDiagonal_SingleElement_ReturnsElement()
    {
      // Arrange
      double[,] A = { { 7.7 } };

      // Act
      double result = ArrayProcessor.MaxOnAndBelowMainDiagonal(A);

      // Assert
      Assert.Equal(7.7, result);
    }

    [Fact]
    public void MaxOnAndBelowMainDiagonal_EmptyArray_ThrowsException()
    {
      // Arrange
      double[,] A = new double[0, 0];

      // Act & Assert
      Assert.Throws<ArgumentException>(() => ArrayProcessor.MaxOnAndBelowMainDiagonal(A));
    }

    [Fact]
    public void MaxOnAndBelowMainDiagonal_NullArray_ThrowsException()
    {
      // Arrange
      double[,] A = null;

      // Act & Assert
      Assert.Throws<ArgumentNullException>(() =>
ArrayProcessor.MaxOnAndBelowMainDiagonal(A));
    }
  }
}
```

Результат выполнения модульных тестов



Результаты покрытия разработанного кода тестами

Name	Covered	Uncovered	Coverable	Total	Percentage
ArrayOperations.ArrayProcessor	48	0	48	75	100%

Выводы по выполненной работе

В ходе выполнения практической работы №2 по модульному тестированию библиотеки классов на C# средствами Visual Studio был разработан класс ArrayProcessor, содержащий три основные функции: поиск минимума из трёх чисел, вычисление суммы элементов двумерного массива с чётной суммой индексов и нахождение максимального значения на главной диагонали и ниже неё. Для каждой функции были подготовлены comprehensive тестовые наборы данных, включающие различные граничные случаи, нормальные условия и исключительные ситуации, что обеспечило полноценное покрытие тестируемого кода. С помощью фреймворка xUnit были реализованы модульные тесты, которые успешно прошли проверку, подтвердив корректность работы всех функций класса. Анализ покрытия кода показал, что тесты охватывают более 100% строк кода, включая все основные ветвления и обработку исключений, что свидетельствует о высоком качестве тестирования и надёжности разработанной библиотеки классов.