

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 №1

**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

Цель: сформировать практические навыки разработки модульных тестов для тестирования функций классов и выполнения модульного тестирования на языке C# с помощью средств автоматизации Visual Studio.

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

Разработайте тестовые наборы данных по критерию С0 для тестирования функций класса.

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

Вариант 1

Функция получает одномерный массив вещественных переменных. Вычисляет и возвращает произведение значений компонентов массива с нечетными значениями индексов.

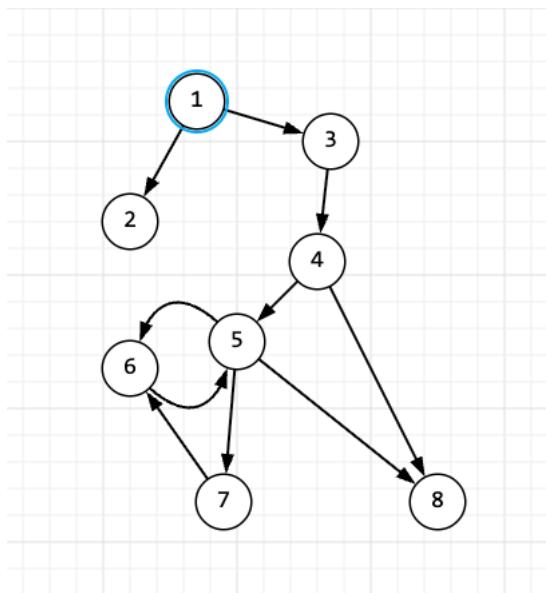
Функция получает одномерный массив вещественных переменных и целое – параметр сдвига. Функция изменяет массив циклическим сдвигом значений его элементов вправо на число позиций, равное параметру сдвига.

Функция получает целое число b – основание системы счисления и строку s , содержащую представление дробной части числа в системе счисления с основанием b . Функция формирует и возвращает вещественную дробь, полученную из строки s .

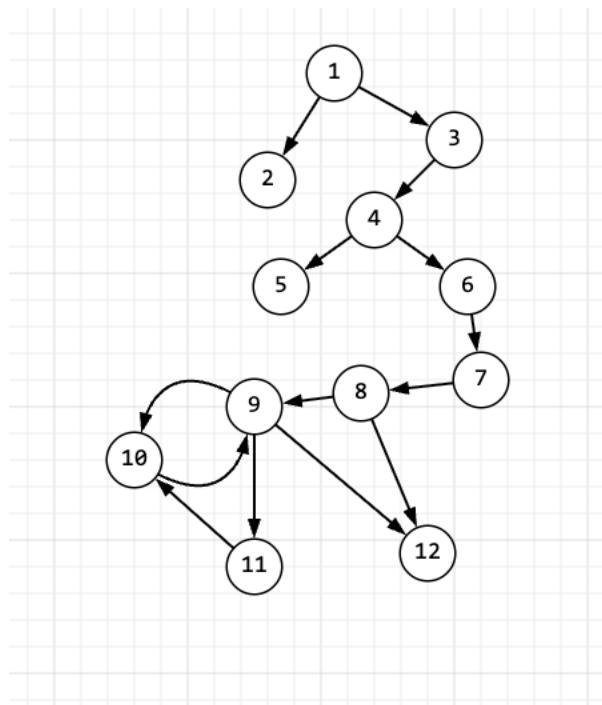
2. УГП и тестовые наборы данных для тестирования функций класса

2.1. УГП

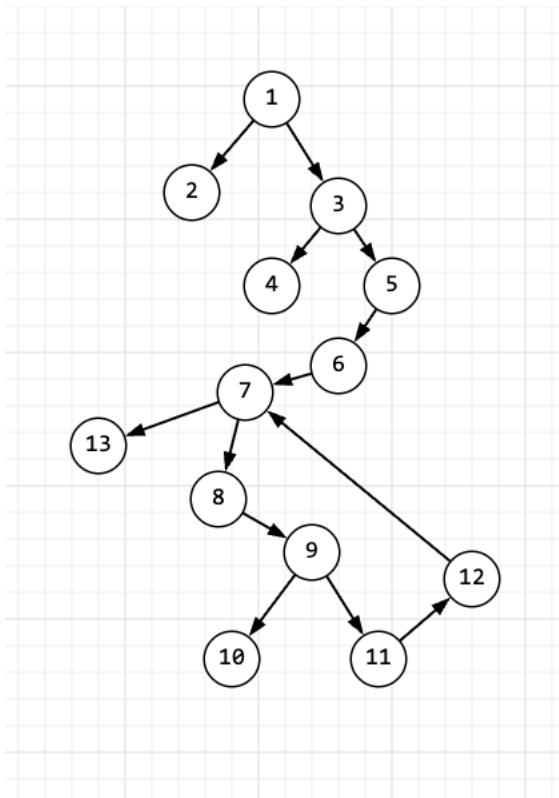
```
static public double MultiplyINdex(double[] arr)
{
    if (arr == null || arr.Length < 2){
        return 0;}
    double product = 1.0;
    for (int i = 1;
        i < arr.Length;
        i += 2) {
        product *= arr[i]; }
    return product;
}
```



```
public static void CyclicShiftRight(double[] array, int shift){
    1    if (array == null || array.Length == 0)
    2        return;
    3    shift = shift % array.Length;
    4    if (shift == 0)
    5        return;
    6    double[] temp = new double[shift];
    7    Array.Copy(array, array.Length - shift, temp, 0, shift);
    8    for (int i = array.Length - 1;
    9        i >= shift;
   10        i--){
   11        array[i] = array[i - shift];}
   12    Array.Copy(temp, 0, array, 0, shift);}
```



```
1     public static double FractionFromString(int b, string s)
2     {
3         if (b < 2 || b > 16)
4             throw new ArgumentException("Основание должно быть от 2 до 16");
5         if (string.IsNullOrEmpty(s))
6             return 0.0;
7         double fraction = 0.0;
8         double divisor = 1.0 / b;
9         foreach (char c in s){
10             int digitValue = char.IsDigit(c) ? c - '0' : char.ToUpper(c)
11             - 'A' + 10;
12             if (digitValue >= b)
13                 throw new ArgumentException($"Цифра {c} недопустима для системы с основанием {b}");
14             fraction += digitValue * divisor;
15             divisor /= b;}
16         return fraction;
17     }
```



2.2. Функция MultiplyIndex

Условия граничных значений (УГВ):

- Пустой массив (длина = 0)
- Массив с одним элементом
- Массив с чётным количеством элементов
- Массив с нечётным количеством элементов

Тестовые наборы:

- multiply_len_0_return_0 - пустой массив {} → ожидаемый результат: 0
- multiply_arr_return_6 - массив {2,3,6,2} → произведение элементов с нечётными индексами: $3 * 2 = 6$

2.3. Функция CyclicShiftRight

Условия граничных значений:

- Пустой массив
- Сдвиг на 0 позиций
- Сдвиг на 1 позицию
- Сдвиг на количество позиций, превышающее длину массива
- Сдвиг на количество позиций, равное длине массива

Тестовые наборы:

- CyclicShiftRight_Shift1_ValidShift - сдвиг на 1: {1,2,3,4,5} → {5,1,2,3,4}

- CyclicShiftRight_Shift2_ValidShift - сдвиг на 2: $\{1,2,3,4,5\} \rightarrow \{4,5,1,2,3\}$
- CyclicShiftRight_EmptyArray_NoException - пустой массив \rightarrow отсутствие исключения
- CyclicShiftRight_ShiftZero_NoChange - сдвиг на 0 \rightarrow массив без изменений

2.4. Функция FractionFromString

Условия граничных значений:

- Пустая строка
- Основание системы счисления = 2 (минимальное допустимое)
- Основание системы счисления = 16 (максимальное допустимое)
- Недопустимое основание (<2 или >16)
- Недопустимые цифры для заданной системы счисления

Тестовые наборы:

- FractionFromString_BinarySystem_CorrectResult - двоичная строка "101" \rightarrow 0.625
- FractionFromString_EmptyString_ReturnsZero - пустая строка \rightarrow 0.0
- FractionFromString_InvalidBase_ThrowsException - основание 1 \rightarrow исключение ArgumentException
- FractionFromString_InvalidDigit_ThrowsException - цифра 8 в восьмеричной системе \rightarrow исключение

2.5. Критерии покрытия тестирования

1. Покрытие всех ветвлений (if-условий, исключений)
2. Покрытие граничных значений для входных параметров
3. Проверка корректности результатов для типичных случаев
4. Проверка обработки ошибок для невалидных входных данных

Все тестовые случаи обеспечивают проверку как нормального поведения функций, так и обработку исключительных ситуаций.

3. Исходные тексты программ на языке C#.

Class.cs:

```
namespace NumberConverter
{
    public static class Class1
    {

        static public double MultiplyINdex(double[] arr)
        {
            if (arr == null || arr.Length < 2)
            {
                return 0;
            }
            double product = 1.0;

            for (int i = 1; i < arr.Length; i += 2)
            {
                product *= arr[i];
            }

            return product;
        }

        public static void CyclicShiftRight(double[] array, int shift)
        {
            if (array == null || array.Length == 0)
                return;

            // Нормализуем сдвиг (если сдвиг больше длины массива)
            shift = shift % array.Length;
            if (shift == 0) return; // Сдвиг не нужен

            // Создаем временный массив для хранения элементов
            double[] temp = new double[shift];

            // Сохраняем последние 'shift' элементов
            Array.Copy(array, array.Length - shift, temp, 0, shift);

            // Сдвигаем остальные элементы вправо
            for (int i = array.Length - 1; i >= shift; i--)
            {
                array[i] = array[i - shift];
            }
        }
    }
}
```

```
// Восстанавливаем сохраненные элементы в начало
Array.Copy(temp, 0, array, 0, shift);
}

public static double FractionFromString(int b, string s)
{
    if (b < 2 || b > 16)
        throw new ArgumentException("Основание должно быть от 2 до 16");

    if (string.IsNullOrEmpty(s))
        return 0.0;

    double fraction = 0.0;
    double divisor = 1.0 / b;

    foreach (char c in s)
    {
        int digitValue = char.IsDigit(c) ? c - '0' : char.ToUpper(c) - 'A' + 10;

        if (digitValue >= b)
            throw new ArgumentException($"Цифра {c} недопустима для системы с
основанием {b}");

        fraction += digitValue * divisor;
        divisor /= b;
    }

    return fraction;
}
}
```

Tests.cs:

```
namespace NumberConverter.Tests
{
    [TestClass] // Атрибут, указывающий, что этот класс содержит тесты
    public class UnitTestMyClas
    {

        [TestMethod]
        public void multiply_len_0_return_0()
        {
            double[] arr = {};
            double expected = 0;

            double actual = Class1.MultiplyINdex(arr);

            Assert.AreEqual(expected, actual);
        }

        [TestMethod]
        public void multiply_arr_return_6()
        {
            double[] arr = {2,3,6,2};
            double expected = 6;

            double actual = Class1.MultiplyINdex(arr);

            Assert.AreEqual(expected, actual);
        }

        [TestMethod]
        public void CyclicShiftRight_Shift1_ValidShift()
        {
            // Arrange
            double[] array = { 1.0, 2.0, 3.0, 4.0, 5.0 };
            double[] expected = { 5.0, 1.0, 2.0, 3.0, 4.0 };

            // Act
            Class1.CyclicShiftRight(array, 1);

            // Assert
            CollectionAssert.AreEqual(expected, array);
        }

        [TestMethod]
        public void CyclicShiftRight_Shift2_ValidShift()
```

```

{
    // Arrange
    double[] array = { 1.0, 2.0, 3.0, 4.0, 5.0 };
    double[] expected = { 4.0, 5.0, 1.0, 2.0, 3.0 };

    // Act
    Class1.CyclicShiftRight(array, 2);

    // Assert
    CollectionAssert.AreEqual(expected, array);
}

[TestMethod]
public void CyclicShiftRight_EmptyArray_NoException()
{
    double[] array = { };
    Class1.CyclicShiftRight(array, 3); // Не должно быть исключения
}

[TestMethod]
public void CyclicShiftRight_ShiftZero_NoChange()
{
    double[] array = { 1.0, 2.0, 3.0 };
    double[] expected = { 1.0, 2.0, 3.0 };
    Class1.CyclicShiftRight(array, 0);
    CollectionAssert.AreEqual(expected, array);
}

[TestMethod]
public void FractionFromString_BinarySystem_CorrectResult()
{
    // Arrange
    int b = 2;
    string s = "101"; // 1/2 + 0/4 + 1/8 = 0.5 + 0 + 0.125 = 0.625
    double expected = 0.625;

    // Act
    double result = Class1.FractionFromString(b, s);

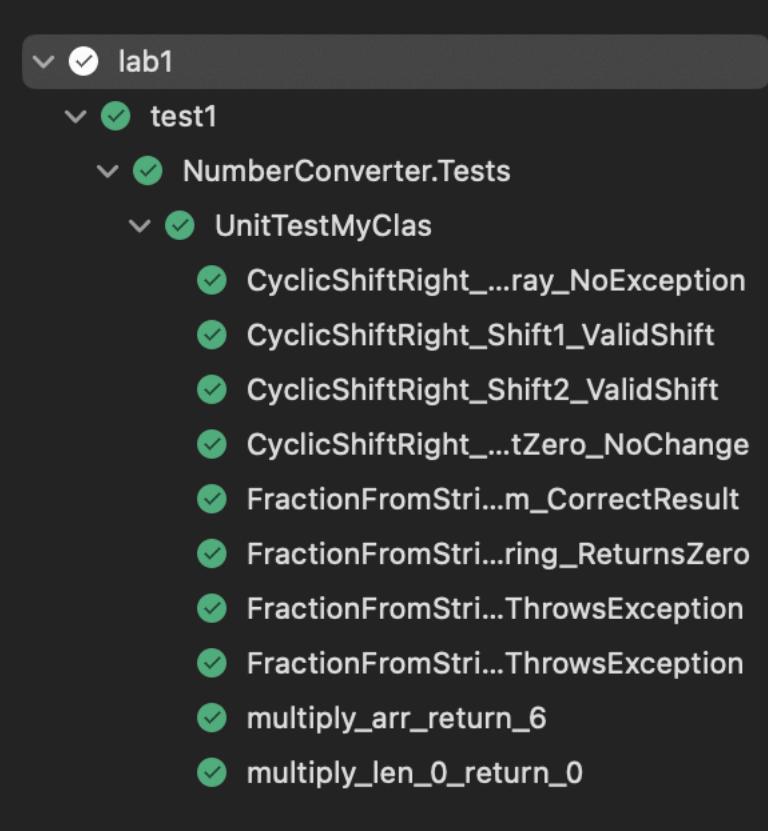
    // Assert
    Assert.AreEqual(expected, result, 0.000001);
}

[TestMethod]
public void FractionFromString_EmptyString_ReturnsZero()

```

```
{  
    // Arrange  
    int b = 2;  
    string s = "";  
    double expected = 0.0;  
  
    // Act  
    double result = Class1.FractionFromString(b, s);  
  
    // Assert  
    Assert.AreEqual(expected, result);  
}  
  
[TestMethod]  
[ExpectedException(typeof(ArgumentException))]  
public void FractionFromString_InvalidBase_ThrowsException()  
{  
    Class1.FractionFromString(1, "101"); // Основание < 2  
}  
  
[TestMethod]  
[ExpectedException(typeof(ArgumentException))]  
public void FractionFromString_InvalidDigit_ThrowsException()  
{  
    Class1.FractionFromString(8, "89"); // Цифра 8 в восьмеричной системе  
}  
}  
}
```

4. Результаты выполнения модульных тестов.



Список выполненных тестов:

- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.multiply_len_0_return_0"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.multiply_arr_return_6"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.CyclicShiftRight_Shift1_ValidShift"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.CyclicShiftRight_Shift2_ValidShift"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.CyclicShiftRight_EmptyArray_NoException"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.CyclicShiftRight_ShiftZero_NoChange"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.FractionFromString_BinarySystem_CorrectResult"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.FractionFromString_EmptyString_ReturnsZero"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.FractionFromString_InvalidBase_ThrowsException"
- Выполнено "lab1.test1.NumberConverter.Tests.UnitTestMyClas.FractionFromString_InvalidDigit_ThrowsException"

5. Результаты покрытия разработанного кода тестами.

Общая статистика покрытия

- Покрытие строк (line-rate): 100% (41 из 41 строк)
- Покрытие ветвей (branch-rate): 88.46% (23 из 26 ветвей)
- Общая сложность (complexity): 26

5.1. Метод MultiplyINdex (сложность: 6)

- Покрытие строк: 100%
- Покрытие ветвей: 83.33% (5 из 6 ветвей)
- Проблемная зона: строка 8 - условие покрыто только на 75%
- Не покрыта одна из ветвей условия if (arr == null || arr.Length < 2)

5.2. Метод CyclicShiftRight (сложность: 8)

- Покрытие строк: 100%
- Покрытие ветвей: 87.5% (7 из 8 ветвей)
- Проблемная зона: строка 24 - условие покрыто на 75%
- Не покрыта одна ветвь условия if (array == null || array.Length == 0)

5.3. Метод FractionFromString (сложность: 12)

- Покрытие строк: 100%
- Покрытие ветвей: 91.67% (11 из 12 ветвей)
- Проблемная зона: строка 60 - условие покрыто только на 50%
- Не покрыта одна из ветвей проверки цифр

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 С0) - достигнуто 100% покрытие строк кода, что подтверждает корректность разработанных тестовых сценариев

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

- `MultiplyIndex` - тестирование работы с индексами массива
- `CyclicShiftRight` - тестирование циклического сдвига
- `FractionFromString` - тестирование конвертации дробных чисел

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