Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Лабораторная работа №1

«Модульное тестирование программ на языке C# средствами Visual Studio»

Вариант №4

Выполнил: студент 4 курса
ИВТ, гр. ИП-113
Шпилев Д. И.
Проверил: старший преподаватель кафедры ПМиК
Агалаков А.А.

Новосибирск, 2024 г.

Цель

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

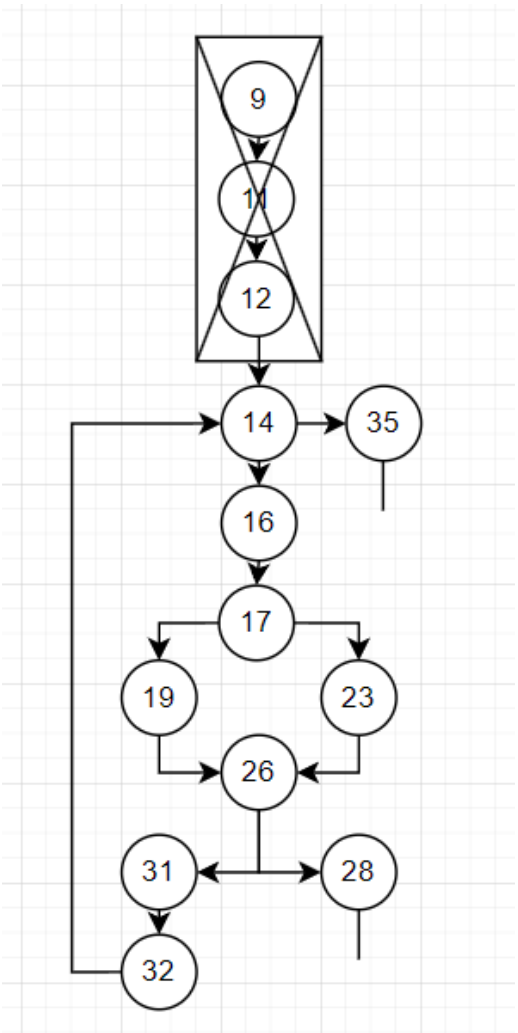
Задание

Разработайте на языке C# класс, содержащий функции в соответствии с вариантом задания. Разработайте тестовые наборы данных по критерию C0 для тестирования функций класса. Протестируйте созданный класс с помощью средств автоматизации модульного тестирования Visual Studio. Напишите отчёт о результатах проделанной работы.

1. Функция получает целое число b – основание системы счисления и строку s, содержащую представление целой части числа в системы счисления с основанием b. Функция формирует и возвращает из строки s целое число.
2. Функция получает одномерный массива целых переменных. Вычисляет и возвращает максимальный по значению элемент этого массива и номер его индекса.
3. Функция получает одномерный массив целых переменных. Вычисляет и возвращает максимальное значение среди нечётных элементов массива с нечётными значениями индекса и значение индекса (через параметр)

УГП и тестовые наборы данных для тестирования функций класса

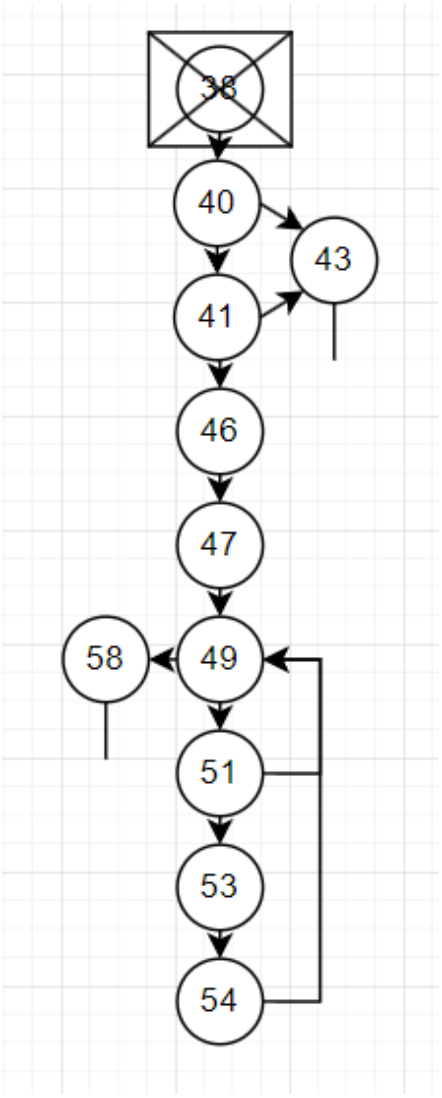
```
9   public static int ConvertToDecimal(int b, string s)
10  {
11      int result = 0;
12      int power = 1;
13
14      for (int i = s.Length - 1; i >= 0; i--)
15      {
16          int digitValue;
17          if (char.IsDigit(s[i]))
18          {
19              digitValue = s[i] - '0';
20          }
21          else
22          {
23              digitValue = char.ToUpper(s[i]) - 'A' + 10;
24          }
25
26          if (digitValue >= b)
27          {
28              throw new ArgumentException($"Цифра '{s[i]}' недопустима для системы счисления с основанием {b}");
29          }
30
31          result += digitValue * power;
32          power *= b;
33      }
34
35      return result;
36  }
```



Для полного тестирования нужно пройти пути 17-19-26 и 17-23-26, а также попасть в исключение 26-28

Для прохождения путей 17-19-26 и 17-23-26 можно взять 16-ричное число, допустим «153AF». Для попадания в исключение 26-28 берем число «1254236» и систему счисления 2.

```
38      public static (int maxValue, int index) FindMaxElement(int[] array)
39      {
40          if (array == null
41              || array.Length == 0)
42          {
43              throw new ArgumentException("Массив не должен быть null или пустым");
44          }
45
46          int maxValue = array[0];
47          int maxIndex = 0;
48
49          for (int i = 1; i < array.Length; i++)
50          {
51              if (array[i] > maxValue)
52              {
53                  maxValue = array[i];
54                  maxIndex = i;
55              }
56          }
57
58          return (maxValue, maxIndex);
59      }
```

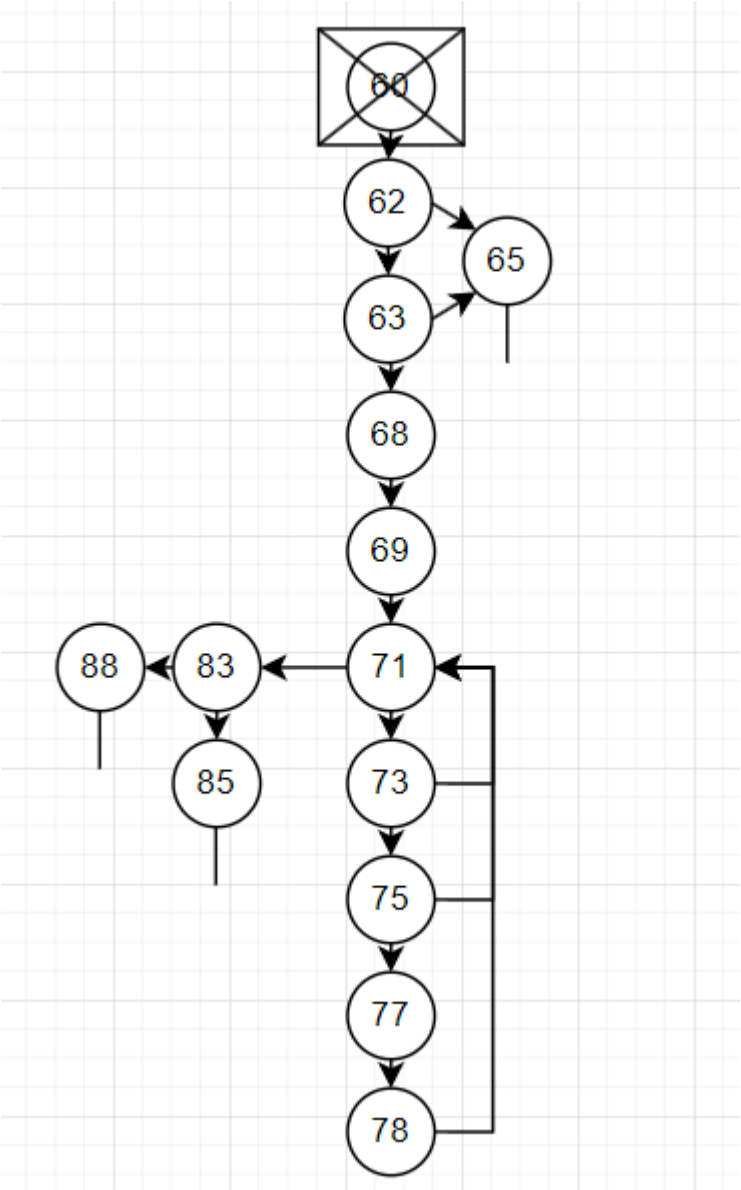


Для полного тестирования нужно выполнить исключения из пунктов 40 и 41 и попасть в пункт 43, а также попасть в путь 49-51-53-54.

Для исключения берем array = null и пустой массив array.

Для пути 49-51-53-54 можем взять `array = { 3, 6, 23, 2, 56, 7, 4, 3, 2 };`

```
60          public static int FindMaxOddElementAtOddIndex(int[] array, out int index)
61          {
62              if (array == null
63                  || array.Length == 0)
64              {
65                  throw new ArgumentException("Массив не должен быть null или пустым");
66              }
67
68              int maxValue = int.MinValue;
69              index = -1;
70
71              for (int i = 1; i < array.Length; i += 2)
72              {
73                  if (array[i] % 2 != 0)
74                  {
75                      if (array[i] > maxValue)
76                      {
77                          maxValue = array[i];
78                          index = i;
79                      }
80                  }
81              }
82
83              if (index == -1)
84              {
85                  throw new ArgumentException("Нет нечетных элементов на нечетных индексах");
86              }
87
88              return maxValue;
89          }
90      }
91  }
```



Для полного тестирования нужно выполнить исключения из пунктов 62 и 63

и попасть в пункт 65, пройти путь 71-73-75-77-78, попасть в исключение 85 и выйти из программы 85.

Для исключения берем array = null и пустой массив array

Для выхода с исключением: `array = { 3, 6, 23, 2, 56, 8, 4, 12, 2 };`

Для выхода без исключения: `array = { 3, 6, 23, 2, 56, 7, 4, 3, 2 };`

Листинг программы:

Program.cs:

```
using System;

namespace Lab1
{
    public class Program
    {
        static void Main(string[] args) { }

        public static int ConvertToDecimal(int b, string s)
        {
            int result = 0;
            int power = 1;

            for (int i = s.Length - 1; i >= 0; i--)
            {
                int digitValue;
                if (char.IsDigit(s[i]))
                {
                    digitValue = s[i] - '0';
                }
                else
                {
                    digitValue = char.ToUpper(s[i]) - 'A' + 10;
                }

                if (digitValue >= b)
                {
                    throw new ArgumentException($"Цифра '{s[i]}' недопустима для
системы счисления с основанием {b}");
                }

                result += digitValue * power;
                power *= b;
            }

            return result;
        }

        public static (int maxValue, int index) FindMaxElement(int[] array)
        {
            if (array == null || array.Length == 0)
            {
                throw new ArgumentException("Массив не должен быть null или
пустым");
            }

            int maxValue = array[0];
            int maxIndex = 0;

            for (int i = 1; i < array.Length; i++)
            {
                if (array[i] > maxValue)
                {
                    maxValue = array[i];
                    maxIndex = i;
                }
            }

            return (maxValue, maxIndex);
        }
```

```
        public static int FindMaxOddElementAtOddIndex(int[] array, out int index)
        {
            if (array == null || array.Length == 0)
            {
                throw new ArgumentException("Массив не должен быть null или
пустым");
            }

            int maxValue = int.MinValue;
            index = -1;

            for (int i = 1; i < array.Length; i += 2)
            {
                if (array[i] % 2 != 0)
                {
                    if (array[i] > maxValue)
                    {
                        maxValue = array[i];
                        index = i;
                    }
                }
            }

            if (index == -1)
            {
                throw new ArgumentException("Нет нечетных элементов на нечетных
индексах");
            }

            return maxValue;
        }
    }
}
```

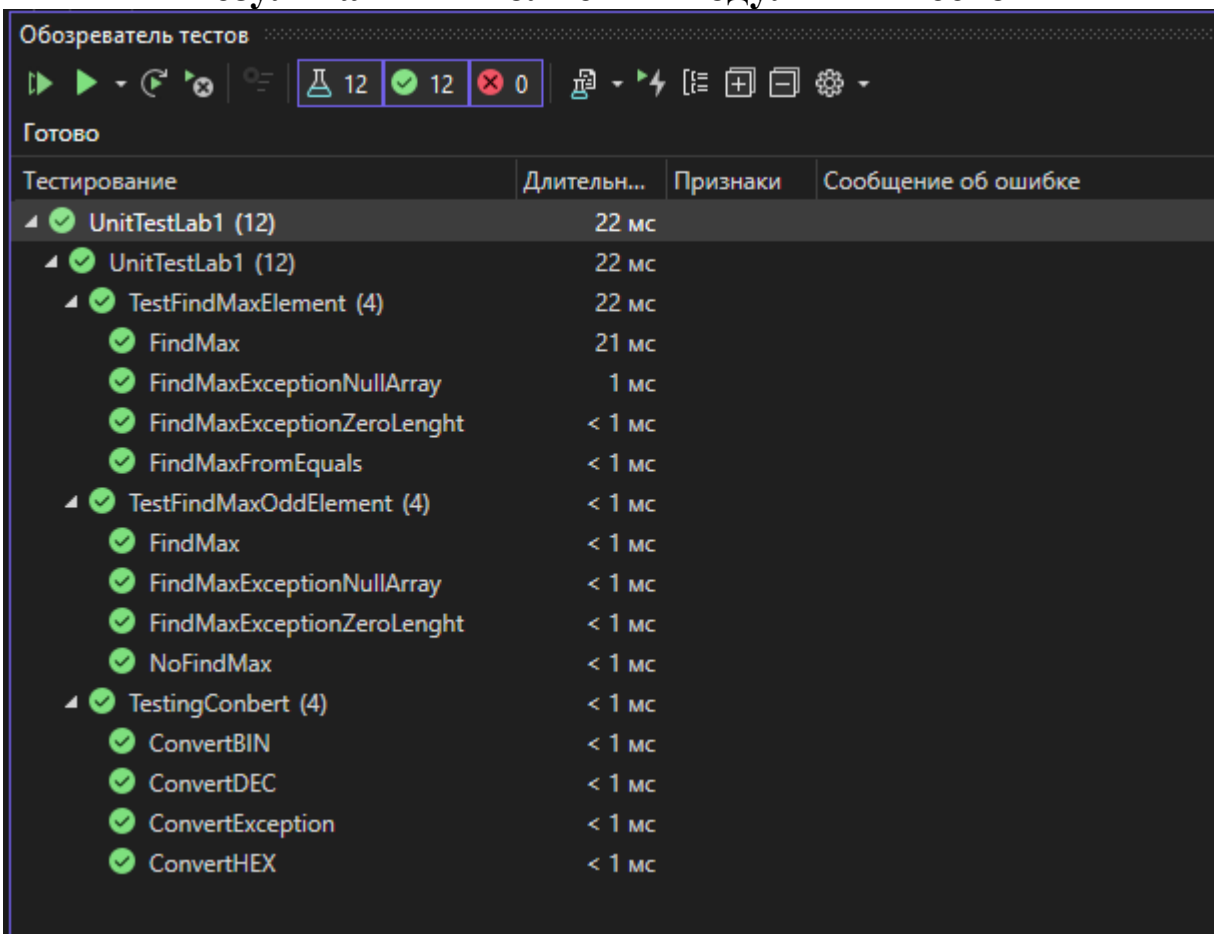
UnitTest1.cs:

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

namespace UnitTestLab1
{
    [TestClass]
    public class TestingConbert
    {
        [TestMethod]
        public void ConvertHEX()
        {
            //arrange
            int footing = 16;
            string number = "153AF";
            //act
            int result = Lab1.Program.ConvertToDecimal(footing, number);
            //assert
            Assert.AreEqual(86959, result);
        }

        [TestMethod]
        public void ConvertBIN()
        {
            //arrange
            int footing = 2;
            string number = "10001110";
            //act
            int result = Lab1.Program.ConvertToDecimal(footing, number);
            //assert
            Assert.AreEqual(142, result);
        }

        [TestMethod]
        public void ConvertDEC()
        {
            //arrange
            int footing = 10;
            string number = "1254236";
            //act
            int result = Lab1.Program.ConvertToDecimal(footing, number);
            //assert
            Assert.AreEqual(1254236, result);
        }

        [TestMethod]
        public void ConvertException()
        {
            //arrange
            int footing = 2;
            string number = "1254236";
            //act
            //assert
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.ConvertToDecimal(footing, number));
        }
    }

    [TestClass]
    public class TestFindMaxElement
    {
        [TestMethod]
        public void FindMax()
        {
```

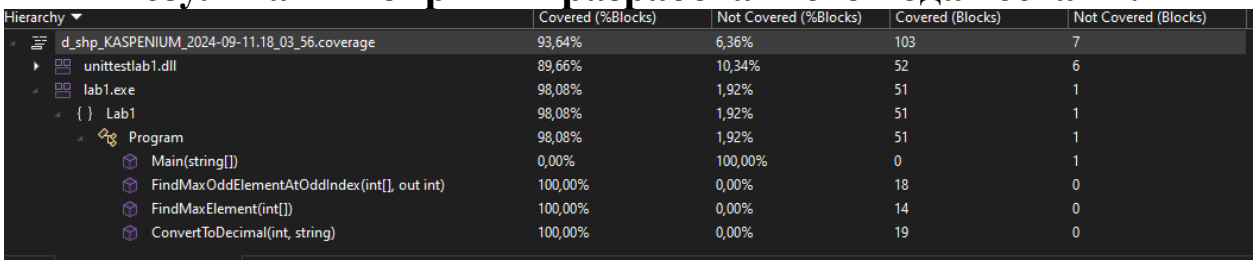
```
            //arrange
            int[] array = { 3, 6, 23, 2, 56, 7, 4, 3, 2 };
            //act
            var result = Lab1.Program.FindMaxElement(array);
            //assert
            Assert.AreEqual((56, 4), result);
        }

        [TestMethod]
        public void FindMaxFromEquals()
        {
            //arrange
            int[] array = { 3,3,3,3,3,3,3,3,3,3 };
            //act
            var result = Lab1.Program.FindMaxElement(array);
            //assert
            Assert.AreEqual((3, 0), result);
        }

        [TestMethod]
        public void FindMaxExceptionZeroLenght()
        {
            //arrange
            int[] array = { };
            //act
            //assert
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.FindMaxElement(array));
        }
        [TestMethod]
        public void FindMaxExceptionNullArray()
        {
            //arrange
            //act
            //assert
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.FindMaxElement(null));
        }
    }

    [TestClass]
    public class TestFindMaxOddElement
    {
        [TestMethod]
        public void FindMax()
        {
            //arrange
            int[] array = { 3, 6, 23, 2, 56, 7, 4, 3, 2 };
            int index;
            //act
            int value = Lab1.Program.FindMaxOddElementAtOddIndex(array, out index);
            //assert
            Assert.AreEqual((7, 5), (value, index));
        }

        [TestMethod]
        public void NoFindMax()
        {
            //arrange
            int[] array = { 3, 6, 23, 2, 56, 8, 4, 12, 2 };
            int index;
            //act
            //assert
```

```
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.FindMaxOddElementAtOddIndex(array, out index), "Нет нечетных элементов
на нечетных индексах");
        }

        [TestMethod]
        public void FindMaxExceptionZeroLenght()
        {
            //arrange
            int[] array = { };
            int index;
            //act
            //assert
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.FindMaxOddElementAtOddIndex(array, out index), "Массив не должен быть
null или пустым");
        }
        [TestMethod]
        public void FindMaxExceptionNullArray()
        {
            //arrange
            int index;
            //act
            //assert
            Assert.ThrowsException<ArgumentException>(() =>
Lab1.Program.FindMaxOddElementAtOddIndex(null, out index), "Массив не должен быть
null или пустым");

        }
    }
}
```

Результаты выполнения модульных тестов



Результаты покрытия разработанного кода тестами.



Hierarchy	Covered (%Blocks)	Not Covered (%Blocks)	Covered (Blocks)	Not Covered (Blocks)
d_shp_KASPENIUM_2024-09-11.18_03_56.coverage	93,64%	6,36%	103	7
unittestlab1.dll	89,66%	10,34%	52	6
lab1.exe	98,08%	1,92%	51	1
{ } Lab1	98,08%	1,92%	51	1
Program	98,08%	1,92%	51	1
Main(string[])	0,00%	100,00%	0	1
FindMaxOddElementAtOddIndex(int[], out int)	100,00%	0,00%	18	0
FindMaxElement(int[])	100,00%	0,00%	14	0
ConvertToDecimal(int, string)	100,00%	0,00%	19	0

Вывод

В результате работы над лабораторной работой были сформированы практические навыки разработки функций классов на языке C#, разработка модульных тестов для тестирования функций классов и выполнения модульного тестирования на языке C# с помощью средств автоматизации Visual Studio.