

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №3

Тестирование программных блоков

тема

Преподаватель

подпись, дата

А. С. Кузнецов

инициалы, фамилия

Студент КИ18-17/16 031830504

номер группы, зачетной книжки

подпись, дата

Е.В. Железкин

инициалы, фамилия

Красноярск 2021

1 Цель работы

На конкретных примерах ознакомиться с базовыми методами блочного тестирования программного обеспечения.

2 Задача работы

Продemonстрировать понимание и применение на практике ключевых понятий, рассмотренных в этой работе.

Вариант 5

Битовый вектор (реализовать в виде массива, хранящего значения true-false), при этом должны быть реализованы и протестированы следующие побитовые (поэлементные) логические операции:

- Сравнение
- И
- ИЛИ
- НЕ
- Импликация

2.1 Инструкция по запуску

Необходимо установить .Net SDK:

• Страница загрузки для Windows:
<https://dotnet.microsoft.com/download/dotnet/5.0>

- Для Linux:

```
$ sudo apt-get update; \
```

```
$ sudo apt-get install -y apt-transport-https && \
```

```
$ sudo apt-get update && \
```

```
$ sudo apt-get install -y dotnet-sdk-5.0
```

Далее на любой из двух систем выполнить в папке проекта:

```
$ dotnet test
```

```
$ dotnet test -v n (для более подробного вывода)
```

3 Ход работы

Был проведён анализ тестов из практической работы №2, результат представлен на рисунке 1. В тестах не проверяется возможность выбросов исключений (Рисунок 2). Так как не весь код покрыт тестами, то для начала исправим это. Результат представлен на рисунке №3.

Symbol	Coverage (%)	Uncovered/Total Stmts.
Total	98%	3/126
BitArrayXUnit	100%	0/87
BitArrayXUnit	100%	0/87
FixtureUnitTesting	100%	0/18
TestEqualsFirstSet(BitArray, BitArray, bool)	100%	0/3
TestCompareFirstSet(BitArray, BitArray, int)	100%	0/3
TestAndFirstSet(BitArray, BitArray, BitArray)	100%	0/3
TestOrFirstSet(BitArray, BitArray, BitArray)	100%	0/3
TestInversionFirstSet(BitArray, BitArray)	100%	0/3
TestMaterialImplicationFirstSet(BitArray, BitArray, BitArray)	100%	0/3
ParameterisedUnitTesting	100%	0/33
TestEqualsFirstSet(bool, bool, bool)	100%	0/5
TestCompareFirstSet(bool, bool, int)	100%	0/5
TestAndFirstSet(bool, bool, bool)	100%	0/6
TestOrFirstSet(bool, bool, bool)	100%	0/6
TestInversionFirstSet(bool, bool)	100%	0/5
TestMaterialImplicationFirstSet(bool, bool, bool)	100%	0/6
UnitTesting	100%	0/36
TestEqualsFirstSet()	100%	0/5
TestCompareFirstSet()	100%	0/5
TestAndFirstSet()	100%	0/6
TestOrFirstSet()	100%	0/6
TestInversionFirstSet()	100%	0/8
TestMaterialImplicationFirstSet()	100%	0/6
BitArrayExtensions	92%	3/39
BitArrayExtensions	92%	3/39
BitArrayExtension	92%	3/39
CompareTo(BitArray, BitArray)	93%	1/14
Equal(BitArray, BitArray)	92%	1/12
MaterialImplication(BitArray, BitArray)	92%	1/13

Рисунок 1 – Статистика покрытия кода из ПР2 тестами

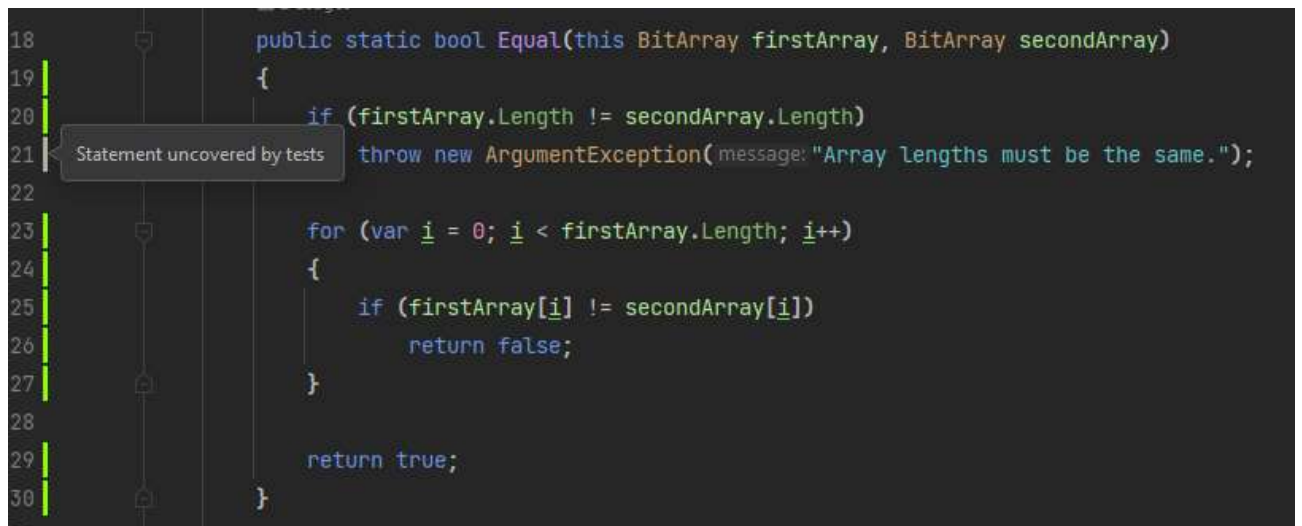


Рисунок 2 – Тестирование с помощью unit-тестов

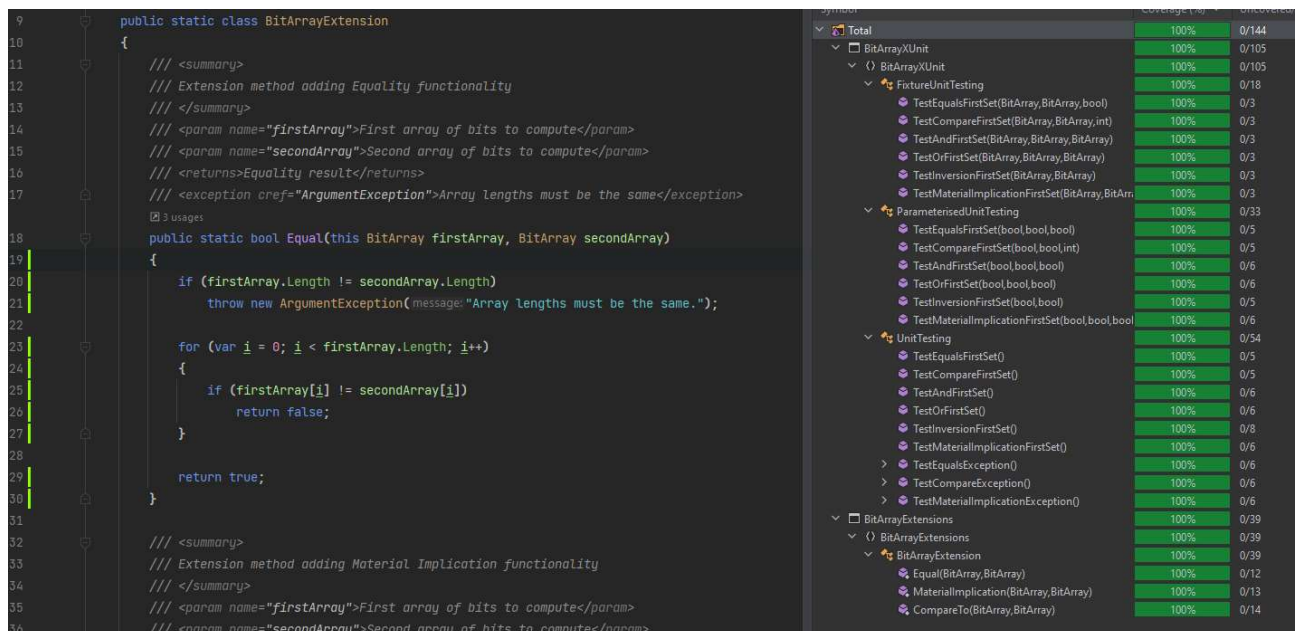


Рисунок 3 – Тестирование с помощью unit-тестов

Далее, по примеру в задании, был написан метод, конвертирующий строку в объект класса BitArray:

```

81  /// <summary>
82  /// Extension method allows translate a string to BitArray instance
83  /// </summary>
84  /// <param name="input">String to convert</param>
85  /// <returns>BitArray instance obtained as a result of conversion</returns>
86  /// <exception cref="ArgumentException">Empty or invalid input string format</exception>
87  [6 usages]
88  public static BitArray ParseFromString(string input)
89  {
90      if (input == null || input.Length == 0)
91      {
92          throw new ArgumentException(message: "Empty argument!");
93      }
94
95      input = input.Replace(oldValue: " ", newValue: "");
96      InputType type;
97
98      if (Regex.IsMatch(input, pattern: @"^[01]*$", options: RegexOptions.Compiled | RegexOptions.IgnoreCase))
99      {
100         type = InputType.Numeric;
101     }
102     else
103     {
104         if (Regex.IsMatch(input, pattern: @"^(true,)*(false,)*(true|false)?$",
105             RegexOptions.Compiled | RegexOptions.IgnoreCase))
106         {
107             type = InputType.Boolean;
108         }
109         else
110         {
111             throw new ArgumentException(message: "Incorrect argument.");
112         }
113     }
114
115     BitArray result = new BitArray(length: 0);
116
117     switch (type)
118     {
119         case InputType.Boolean:
120
121             var temp:string[] = input.Split(separator: ',');
122             var bMediator = new bool[temp.Length];
123
124             for (var i = 0; i < temp.Length; i++)
125             {
126                 if (temp[i].ToLower().Equals("false"))
127                 {
128                     bMediator[i] = false;
129                 }
130                 else
131                 {
132                     bMediator[i] = true;
133                 }
134             }

```

Рисунок 4 – Часть метода ParseFromString

```

88 |
89 |         if (input == null || input.Length == 0)
90 |         {
91 |             throw new ArgumentException(message: "Empty argument!");
92 |         }
93 |

```

Рисунок 5 – Составной условный оператор

```

97 |
98 |         if (Regex.IsMatch(input, pattern: @"^[01]*$", options: RegexOptions.Compiled | RegexOptions.IgnoreCase))
99 |         {
100 |             type = InputType.Numeric;
101 |         }
102 |         else
103 |         {
104 |             if (Regex.IsMatch(input, pattern: @"^(true,)*(false,)*(true|false)?$",
105 |                 options: RegexOptions.Compiled | RegexOptions.IgnoreCase))
106 |             {
107 |                 type = InputType.Boolean;
108 |             }
109 |             else
110 |             {
111 |                 throw new ArgumentException(message: "Incorrect argument.");
112 |             }
113 |         }

```

Рисунок 6 – Условные операторы

```
116 | switch (type)
117 | {
118 |     case InputType.Boolean:
119 |
120 |         var temp:string[] = input.Split(separator: ',');
121 |         var bMediator = new bool[temp.Length];
122 |
123 |         for (var i = 0; i < temp.Length; i++)
124 |         {
125 |             if (temp[i].ToLower().Equals("false"))
126 |             {
127 |                 bMediator[i] = false;
128 |             }
129 |             else
130 |             {
131 |                 bMediator[i] = true;
132 |             }
133 |         }
134 |
135 |         result = new BitArray(bMediator);
136 |
137 |         break;
138 |
139 |     case InputType.Numeric:
140 |
141 |         var nMediator = new bool[input.Length];
142 |
143 |         for (var i = 0; i < input.Length; i++)
144 |         {
145 |             if (input[i] == '0')
146 |             {
147 |                 nMediator[i] = false;
148 |             }
149 |             else
150 |             {
151 |                 nMediator[i] = true;
152 |             }
153 |         }
154 |
155 |         result = new BitArray(nMediator);
156 |
157 |         break;
158 | }
```

Рисунок 7 – Оператор switch - case

Я смог выделить только два класса эквивалентности битовых векторов: недействительные (пустые) и действительные (все прочие).

```
[Test]
public void Test1()
{
    var actual:BitArray = BitArrayExtension.ParseFromString(input: "110");
    var expected = new BitArray(values: new bool[] { true, true, false });

    Assert.AreEqual(expected, actual);
}

[Test]
public void Test2()
{
    var actual:BitArray = BitArrayExtension.ParseFromString(input: "true,True,false");
    var expected = new BitArray(values: new bool[] { true, true, false });

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

Рисунок 8 – Тестирование потенциальных действительных битовых векторов

```
[Test]
public void Test20()
{
    Assert.Catch( code: () => BitArrayExtension.ParseFromString(input: null));
}
```

Рисунок 9 – Тестирование потенциальных недействительных битовых векторов

▼ Total	100%	0/91
▼ BitArrayExtensions	100%	0/91
▼ BitArrayExtensions	100%	0/91
▼ BitArrayExtension	100%	0/91
Equal(BitArray, BitArray)	100%	0/12
MaterialImplication(BitArray, BitArray)	100%	0/13
CompareTo(BitArray, BitArray)	100%	0/14
ParseFromString(string)	100%	0/52

Рисунок 10 – Полное покрытие кода тестами

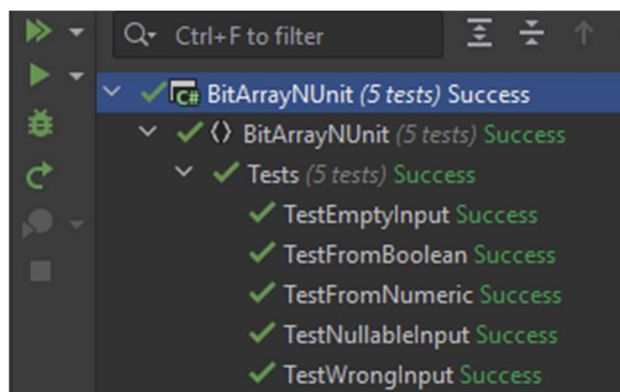


Рисунок 11 – Результаты тестов

4 Вывод

В ходе данной лабораторной работы были изучены базовые методы блочного тестирования программного обеспечения на конкретных примерах.