2 ЛАБОРАТОРНАЯ РАБОТА №2

«Исследование способов структурного тестирования программного обеспечения»

2.1 Цель работы

Исследовать основные подходы к структурному тестированию программного обеспечения. Приобрести практические навыки построения графа потоков управления и определения независимых ветвей программы.

2.2 Вариант задания

Задача 1. Дана целочисленная прямоугольная матрица. Определить количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент.

Задача 2. Дана строка, среди символов которой есть двоеточие. Получить все символы, расположенные между первым и вторым двоеточием. Если второго двоеточия нет, то получить все символы, расположенные после единственного имеющегося двоеточия.

Задача 3. Программа, которая считывает текст из файла и выводит на экран предложения, содержащие максимальное количество знаков пунктуации.

2.3 Ход выполнения работы

2.3.1 Программы, написанные в предыдущей лабораторной работе, были исследованы с помощью структурного тестирования. Код первой программы представлен в листинге 2.1.

Листинг 2.1 – Текст первой программы

```
public class MatrixCounter
```

```
{
    public int CountNegativeNumbersInRowsWithZeros(List<List<int>> data)
    {
        int result = 0;
        foreach (var row in data)
        {
            if (row.Contains(0))
            {
                var amount = row.Count(x => x < 0);
                result += amount;
            }
        }

        return result;
    }
}
```

Для этой программы был построен граф потоков управления, который показан на рисунке 2.1. Цикломатическое число для этого графа:

C(G) = 7-6+2=3.

Следует вывод, что граф имеет 3 независимых ветви.

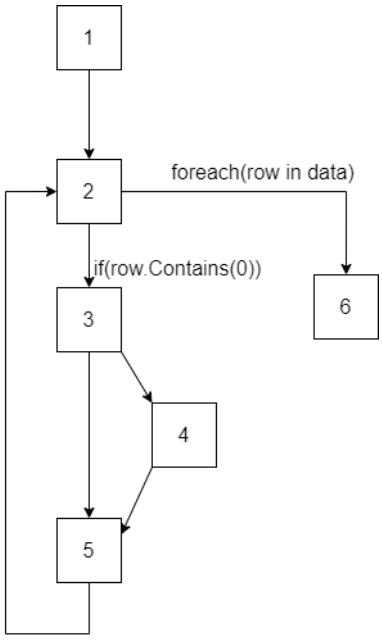

Рисунок 2.1 – Граф потоков управления первой программы

Были найдены все 3 независимых ветви этой программы:

1) 1,2,6
2) 1,2,3,5,2,6
3) 1,2,3,4,5,2,6

2.3.2 Для следующей программы были проделаны те же действия. Код программы представлен в листинге 2.2.

Листинг 2.2 – Текст второй программы

```
public static class StringExtension
{
    public static string GetStringBetweenColons(this string str)
    {
        var match = Regex.Match(str, ":[^:]*:?");
        var result = match.Groups[0].Value.Replace(":", "");
        return result;
    }
}
```

Для второй программы был построен граф потоков управления, который показан на рисунке 2.2. Цикломатическое число для этого графа:

C(G) = 2-3+2=1.

Следует вывод, что граф имеет 1 независимую ветвь.

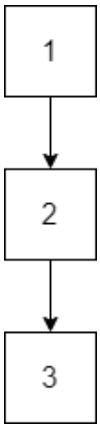

Рисунок 2.2 – Граф потоков управления второй программы

Были найдены все независимых ветви этой программы:

1) 1, 2, 3

2.3.3 Далее была исследована третья программа. Код программы представлен в листинге 2.3.

Листинг 2.3 – Текст третьей программы

```
public class SadFileReader

{

    private readonly List<char> _punctuationMarks = new List<char>(){',', '.', '!', '?', ';', ':'};


    public void WriteToConsoleStringWithMaxPunctuactionMarksCount(string fileName)

    {

        var maxStrings = new List<string>();

        int punctuationMarksMaxAmount = 0;


        var reader = new StreamReader(fileName);

        string? line;

        while ((line = reader.ReadLine()) != null)

        {

            int punctuationMarksCount = line.Count(x => _punctuationMarks.Contains(x));

            if (punctuationMarksCount > punctuationMarksMaxAmount)

            {

                maxStrings.Clear();

                maxStrings.Add(line);

                punctuationMarksMaxAmount = punctuationMarksCount;

            }

            else if (punctuationMarksCount == punctuationMarksMaxAmount)

            {

                maxStrings.Add(line);

            }

        }
```

```
            foreach (var maxString in maxStrings)
            {
                Console.WriteLine(maxString);
            }
        }
    }
```

Для третьей программы был построен граф потоков управления, который показан на рисунке 2.3. Цикломатическое число для этого графа:

C(G) = 13-10+2=5.

Следует вывод, что граф имеет 5 независимых ветвей.

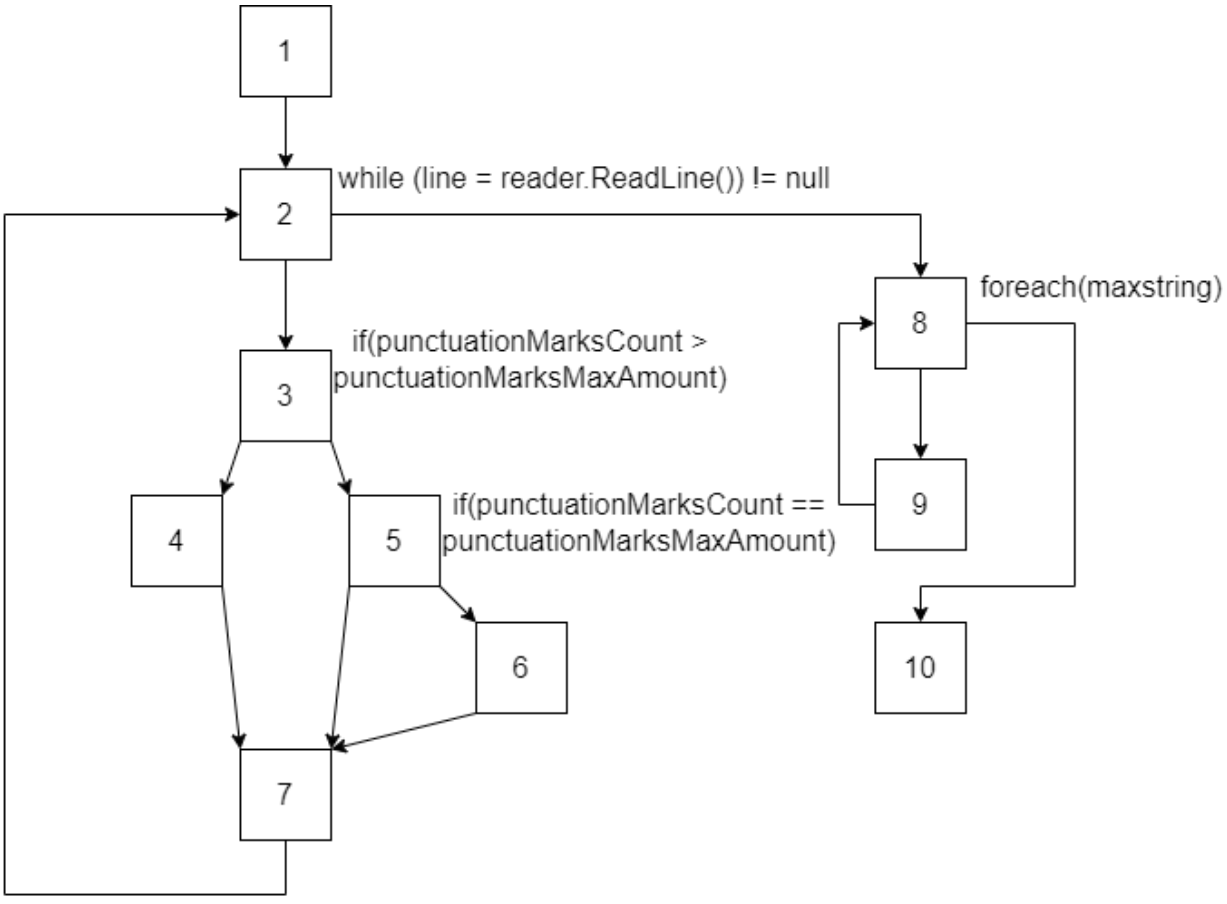

Рисунок 2.3 – Граф потоков управления третьей программы

Были найдены все 5 независимых ветвей этой программы:

1) 1, 2, 8, 10
2) 1, 2, 3, 4, 7, 2, 8, 9, 10

3) 1, 2, 3, 5, 7, 2, 8, 10
4) 1, 2, 3, 5, 6, 7, 8, 9, 10
5) 1, 2, 3, 5, 7, 2, 8, 9, 10

Выводы

В ходе лабораторной работы были исследованы основные подходы к структурному тестированию программного обеспечения. Также были приобретены практические навыки построения графа потоков управления и определения независимых ветвей программы. В конце выполнения лабораторной работы был написан отчет.