Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №1
«Встроенные типы данных в C#. Массивы. Строки. Регулярные выражения"»
по дисциплине
«ПЛАТФОРМА .NET»

Выполнил студент группы ИС/б-32о
Горбенко К. Н.
Проверил
ст. преп. Забаштанский А.К.

Севастополь
2019

1 ЦЕЛЬ РАБОТЫ

• изучить классификацию типов данных и отличительные особенности синтаксических конструкций языка C# от C++;

• изучить базовые типы: Array, String, StringBuilder, а также средства стандартного ввода/вывода и возможности форматирования вывода;

• получить понятие о регулярных выражениях и их применении для поиска, замены и разбиения текста на синтаксические лексемы.

2 ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

Для **варианта № 3** заданы следующие задания:

1. Проработать примеры программ 1-8, данные в теоретических сведениях. Создать на их основе программы. Получить результаты работы программ и уметь их объяснить. Внести их в отчет по работе с комментариями.

2. Найти номер столбца двухмерного массива целых чисел, для которого среднеарифметическое его элементов максимально.

3. Создать программу, которая будет вводить строку в переменную String. Найти в ней те слова, которые начинаются и заканчиваются одной и той же буквой.

4. Задан текст. Определить, является ли он текстом на английском языке.

3 ХОД РАБОТЫ

3.1 Примеры программ на C#

3.1.1 Пример № 1

Результат работы программы № 1:

```
Here is the array:
Саша
Маша
Олег
Света
Игорь


Игорь
Света
Олег
Маша
Саша


Cleared out all but one...
Игорь
```

Рисунок 1 – Результат работы программы из примера № 1

В примере № 1 демонстрируются статические методы Reverse и Clear класса Array.

3.1.2 Пример № 2

Результат работы программы № 2:

```
oneDimensionalArray
1234
twoDimensionalArray
1       2       3       4
5       6       7       8
```

Рисунок 2 – Результат работы программы из примера № 2

В программе демонстрируется возможность передать любой массив в функцию с помощью абстрактного класса Array. При этом индексирование массива происходит с помощью методов GetValue.

3.1.3 Пример № 3

Результат работы программы № 3:

```
strM1
Здравствуй, Мир!
World
Мир
Мир
```

Рисунок 3 – Результат работы программы из примера № 3

В данной программе исследуется тип данных char и возможность перехода от массива символов к строке и наоборот. Для перевода строки в массив символов используется метод класса String ToCharArray. Для обратной операции используется цикл for. Однако лучше это сделать с помощью конструктора класса String, принимающего массив символов.

3.1.4 Пример № 4

Результат работы программы № 4:

```
s1=ABCCDE, s2=CDE, b1=True,ch1=A,ch2=C
Lenon
```

Рисунок 4 – Результат работы программы из примера № 4

В данной программе демонстрируется класс StringBuilder, который используется для увеличения производительности при работе со строками. Основные методы: Append, Remove, Insert.

3.1.5 Примеры № 5, № 6, № 7

В программах № 5, № 6 и № 7 исследуются способы использования регулярных выражений для поиска соответствий в строке. Для данной цели могут использоваться статические методы Matches и Match, принимающие как строку, так и шаблон, а также те же методы класса Regex, инициализированного с шаблоном в конструкторе.

3.1.6 Пример № 8

```
theMatch.Length: 4
theMatch: Это
theMatch.Length: 7
theMatch: строка
theMatch.Length: 4
theMatch: для
```

Рисунок 5 – Результат работы программы из примера № 4

В данной программе в шаблон регулярного выражения включаются именованные группы.

3.2 Индивидуальное задание для работы над массивами

Была написана следующая программа:

```
1  public static class ArrayOperations
2  {
3      public static int GetMaximalAverageColumn(int[,] matrix)
4      {
5          if (matrix == null)
6              throw new ArgumentNullException($"{nameof(matrix)} instance
                   was null");
7
8          var columnsAverages = matrix
9                                    .GetColumns()
10                                   .Select(x => x.Average())
11                                   .ToArray();
12
13         return Array.IndexOf(columnsAverages, columnsAverages.Max());
14     }
15
16     public static void PrintMatrix(int[,] matrix)
17     {
18         if (matrix == null)
19             throw new ArgumentNullException($"{nameof(matrix)} instance
                   was null");
20
21         for (var i = 0; i < matrix.GetLength(0); i++)
22         {
23             for (var j = 0; j < matrix.GetLength(1); j++)
24                 Console.Write($"{matrix[i, j]} ");
```

```
25
26              Console.WriteLine();
27          }
28      }
29  }
30
31  public static class ExtensionMethods
32  {
33      public static IEnumerable<int[]> GetColumns(this int[,] array)
34      {
35          if (array == null)
36              throw new ArgumentNullException($"{nameof(array)} instance was
                    null");
37
38          for (var j = 0; j < array.GetLength(1); j++)
39          {
40              var column = new int[array.GetLength(0)];
41
42              for (var i = 0; i < array.GetLength(0); i++)
43                  column[i] = array[i, j];
44
45              yield return column;
46          }
47      }
48  }
49
50  public class Program
51  {
52      public static void CheckArrays()
53      {
54          WriteLine("Checking arrays task");
55
56          const int M = 5;
57          const int N = 10;
58          var matrix = new int[M, N];
59          var random = new Random();
60
61          for (var i = 0; i < matrix.GetLength(0); i++)
62              for (var j = 0; j < matrix.GetLength(1); j++)
63                  matrix[i, j] = random.Next(100);
64
65          ArrayOperations.PrintMatrix(matrix);
66
67          WriteLine("\nAverages:\n");
68          foreach (var item in matrix.GetColumns().Select(x => x.Average()))
69              Write($"{item} ");
70
```

```
71          WriteLine($"\nColumn with maximal average: {ArrayOperations.
                GetMaximalAverageColumn(matrix)}");
72      }
73  }
```

Результат работы программы:

```
Checking arrays task
36 62 80 10 61 8 42 20 33 65
60 40 20 38 48 9 1 60 27 19
47 72 74 23 84 23 27 44 39 68
74 9 82 75 77 85 63 59 75 51
40 96 70 40 45 33 45 33 5 8

Averages:

51,4 55,8 65,2 37,2 63 31,6 35,6 43,2 35,8 42,2
Column with maximal average: 2
```

Рисунок 6 – Результат работы программы с операциями над массивами

В данной программе активно используется библиотека System.Linq для операций над коллекциями. Также был написан extension-метод для извлечения столбцов матрицы. Для определения позиции с наибольшим средним столбцом в строке использовался статический метод IndexOf класса Array.

3.3 Индивидуальное задание для работы над строками

Была написана следующая программа:

```
1  public static class StringOperations
2  {
3      public static IEnumerable<string>
           GetWordsStartingAndEndingTheSameLetter(string line)
4          => line.Split(" ").Where(x => x.First() == x.Last());
5  }
6
7  public class Program
8  {
9      public static void CheckStrings()
10     {
11         WriteLine("Checking strings task");
12
13         WriteLine("Enter your string:");
14         var str = ReadLine();
15
16         WriteLine("Words starting and ending with the same letter:");
17         foreach (var word in StringOperations.
               GetWordsStartingAndEndingTheSameLetter(str))
18             Write($"{word} ");
19     }
```

```
20 }
```

Результат работы программы:

```
Checking strings task
Enter your string:
asdjwqjda jdawfrvd sdjfewefs revcv regergeh dkbfvksd
Words starting and ending with the same letter:
asdjwqjda sdjfewefs dkbfvksd
```

Рисунок 7 – Результат работы программы с операциями над строками

В данной программе используется метод Split класса String для разделения строки на подстроки, а также библиотека System.Linq для фильтрации полученной коллекции. Условием для фильтра является равенство результатов выполнения методов First и Last класса String, возвращающих соответственно первый и последний символы строки.

3.4 Индивидуальное задание для работы с регулярными выражениями

Была написана следующая программа:

```
1  public static class RegexOperations
2  {
3      private static string englishRegexTemplate = @"^[A-Za-z0-9 .,!?:;()]+$
           ";
4
5      public static bool IsInEnglish(string test)
6          => Regex.IsMatch(test, englishRegexTemplate);
7  }
8
9  public class Program
10 {
11     public static void CheckRegexes()
12     {
13         WriteLine("Checking regex task");
14
15         var englishText    = "A text in English with some punctuation
               marks:.,;!?()";
16         var notEnglishText = "Текст ненаанглийском  . Всякиезнакипрепинания
               :.?,!";
17
18         WriteLine(englishText);
19         WriteLine(RegexOperations.IsInEnglish(englishText));
20
21         WriteLine(notEnglishText);
```

```
22          WriteLine(RegexOperations.IsInEnglish(notEnglishText));
23      }
24  }
```

Результат работы программы:

```
Checking regex task
A text in English with some punctuation marks:.,;!?(
True
Текст не на английском. Всякие знаки препинания:.?,!
False
```

Рисунок 8 – Результат работы программы с операциями над регулярными выражениями

В данной программе создан класс, полем которого является шаблон регулярного выражения, осуществляющего проверку того, что все встреченные в строке символы являются либо буквами английскго алфавита, либо арабскими цифрами, либо пробелами, либо знаками пунктуации.

Для вызова регулярного выражения использовался статический метод Regex.IsMatch, который принимает как входную строку, так и сам шаблон регулярного выражения.

4 ВЫВОД

В ходе лабораторной работы были изучены массивы, строки и регулярные выражения языка C#, а также встроенные средства для работы с ними.

Все массивы в среде .NET являются наследниками абстрактного класса Array, что позволяет передавать в одну и ту же функцию любой массив. Кроме того, класс Array предоставляет множество статических методов и методов экземпляра для эффективной работы с массивами.

Строки в платформе .NET являются неизменяемыми, что приводит к просадкам в производительности при выполении множества операций над ними. Для избежания проблем с производительность в таких случаях используют класс StringBuilder. Класс String также содержит множество методов для работы со строками.

Регулярные выражения - мощный инструмент для работы со строками. Они позволяют проверять строку на соответствие шаблону и производить

поиск подстрок в строке. Соответствующие операции обычными методами более трудозатратны.