

Лабораторная работа №8

Тема: Создание и применение регулярных выражений при разработке программ.

Цель: Научится создавать регулярные выражения и применять их при разработке программ.

Выполнение работы:

Регулярные выражения — это часть небольшой технологической области, невероятно широко используемой в огромном диапазоне программ. Регулярные выражения можно представить себе как мини-язык программирования, имеющий одно специфическое назначение: находить подстроки в больших строковых выражениях.

Главным преимуществом регулярных выражений является использование метасимволов — специальные символы, задающие команды, а также управляющие последовательности, которые работают подобно управляющим последовательностям C#. Это символы, предваренные знаком обратного слеша (\) и имеющие специальное назначение.

В таблице 1 специальные метасимволы регулярных выражений C# сгруппированы по смыслу.

Таблица 1 - Метасимволы, используемые в регулярных выражениях C#

| Символ | Значение | Пример | Соответствует |
|------------------------|---|--------|--|
| Классы символов | | | |
| [...] | Любой из символов, указанных в скобках | [a-z] | В исходной строке может быть любой символ английского алфавита в нижнем регистре |
| [^...] | Любой из символов, не указанных в скобках | [^0-9] | В исходной строке может быть любой символ кроме цифр |
| . | Любой символ, кроме перевода строки или другого разделителя Unicode-строки | | |
| \w | Любой текстовый символ, не являющийся пробелом, символом табуляции и т.п. | | |
| \W | Любой символ, не являющийся текстовым символом | | |
| \s | Любой пробельный символ из набора Unicode | | |
| \S | Любой непробельный символ из набора Unicode. Обратите внимание, что символы \w и \S - это не одно и то же | | |

| Символ | Значение | Пример | Соответствует |
|---|--|--------------------|--|
| \d | Любые ASCII-цифры. Эквивалентно [0-9] | | |
| \D | Любой символ, отличный от ASCII-цифр. Эквивалентно [^0-9] | | |
| Символы повторения | | | |
| {n,m} | Соответствует предшествующему шаблону, повторенному не менее n и не более m раз | s{2,4} | "Press", "ssl", "progressss" |
| {n,} | Соответствует предшествующему шаблону, повторенному n или более раз | s{1,} | "ssl" |
| {n} | Соответствует в точности n экземплярам предшествующего шаблона | s{2} | "Press", "ssl", но не "progressss" |
| ? | Соответствует нулю или одному экземпляру предшествующего шаблона; предшествующий шаблон является необязательным | Эквивалентно {0,1} | |
| + | Соответствует одному или более экземплярам предшествующего шаблона | Эквивалентно {1,} | |
| * | Соответствует нулю или более экземплярам предшествующего шаблона | Эквивалентно {0,} | |
| Символы регулярных выражений выбора | | | |
| | Соответствует либо подвыражению слева, либо подвыражению справа (аналог логической операции ИЛИ). | | |
| (...) | Группировка. Группирует элементы в единое целое, которое может использоваться с символами *, +, ?, и т.п. Также запоминает символы, соответствующие этой группе для использования в последующих ссылках. | | |
| (?:...) | Только группировка. Группирует элементы в единое целое, но не запоминает символы, соответствующие этой группе. | | |
| Якорные символы регулярных выражений | | | |
| ^ | Соответствует началу строкового выражения или началу строки при многострочном поиске. | ^Hello | "Hello, world", но не "Ok, Hello world" т.к. в этой строке слово "Hello" находится не в начале |
| \$ | Соответствует концу строкового выражения или концу строки при многострочном поиске. | Hello\$ | "World, Hello" |
| \b | Соответствует границе слова, т.е. соответствует позиции между символом \w и символом \W или между символом | \b(my)\b | В строке "Hello my world" выберет слово "my" |

| Символ | Значение | Пример | Соответствует |
|--------|---|----------|---|
| | \w и началом или концом строки. | | |
| \b | Соответствует позиции, не являющейся границей слов. | \b(ld)\b | Соответствие найдется в слове "World", но не в слове "ld" |

При создании шаблона регулярного выражения с ним можно осуществить различные действия, в зависимости от того, что вам необходимо. Можно просто проверить, существует ли текст, соответствующий шаблону, в исходной строке. Для этого нужно использовать метод `IsMatch()`, который возвращает логическое значение (Пример 1).

Пример 1. Использование метода `IsMatch()`, который возвращает логическое значение:

```
using System;
using System.Text.RegularExpressions;
class Example
{
    static void Main()
    {
        // Массив тестируемых строк
        string[] test = {
            "Wuck World", "Hello world", "My wonderful world"
        };
        // Проверим, содержится ли в исходных строках слово World
        // при этом мы не укажем опции RegexOptions
        Regex regex = new Regex("World");
        Console.WriteLine("Регистрозависимый поиск: ");
        foreach (string str in test)
        {
            if (regex.IsMatch(str))
                Console.WriteLine("В исходной строке: \"{0}\" есть совпадения!", str);
        }
        Console.WriteLine();
        // Теперь укажем поиск, не зависимый от регистра
        regex = new Regex("World", RegexOptions.IgnoreCase);
        Console.WriteLine("РегистроНЕзависимый поиск: ");
```

```

        foreach (string str in test)
        {
            if (regex.IsMatch(str))
                Console.WriteLine("В исходной строке: \"{0}\" есть
совпадения!", str);
        }
    }
}

```

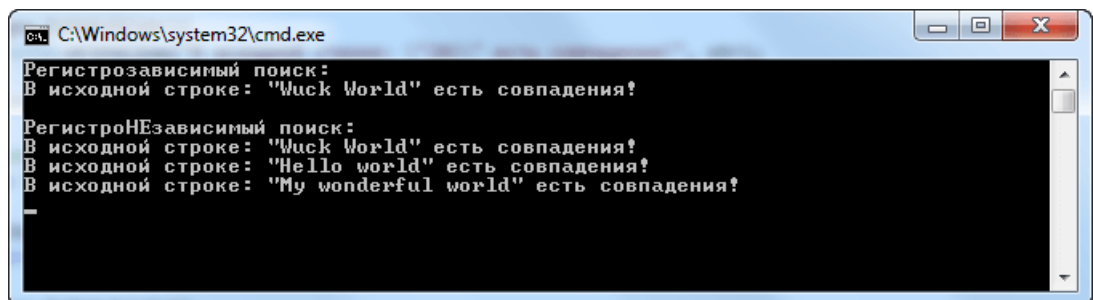


Рисунок 1 – Выполнение программы с использованием метода IsMatch()

Если нужно вернуть найденное соответствие из исходной строки, то можно воспользоваться методом Match(), который возвращает объект класса Match, содержащий сведения о первой подстроке, которая сопоставлена шаблону регулярного выражения. В этом классе имеется свойство Success, которое возвращает значение true, если найдено следующее совпадение, которое можно получить с помощью вызова метода Match.NextMatch(). Эти вызовы метода можно продолжать пока свойство Match.Success не вернет значение false (Пример 2).

Пример 2. Использование свойства Match.Success:

```

using System;
using System.Text.RegularExpressions;
class Example
{
    static void Main()
    {
        // Допустим в исходной строке нужно найти все числа,
        // соответствующие стоимости продукта
        string input = "Добро пожаловать в наш магазин, вот наши цены: " +
            "1 кг. яблок - 20 руб. " +
            "2 кг. апельсинов - 30 руб. " +

```

```

"0.5 кг. орехов - 50 руб.";

string pattern = @"\\b(\\d+\\W?руб)";
Regex regex = new Regex(pattern);

// Получаем совпадения в экземпляре класса Match
Match match = regex.Match(input);

// отображаем все совпадения
while (match.Success)
{
    // Т.к. мы выделили в шаблоне одну группу (одни круглые
    скобки),
    // ссылаемся на найденное значение через свойство Groups
    класса Match
    Console.WriteLine(match.Groups[1].Value);

    // Переходим к следующему совпадению
    match = match.NextMatch();
}
}
}

```

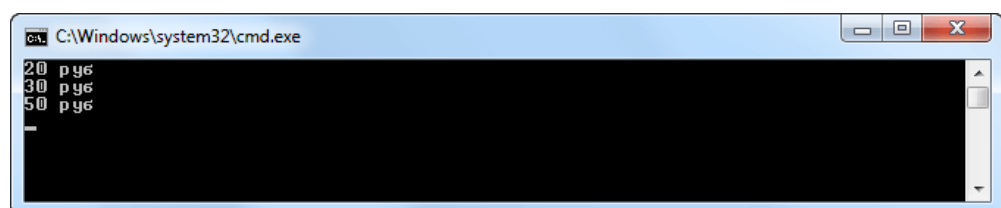


Рисунок 2 - Выполнение программы с использованием свойства Match.Success

Извлечь все совпадения можно и более простым способом, используя метод `Regex.Matches()`, который возвращает объект класса `MatchCollection`, который, в свою очередь, содержит сведения обо всех совпадениях, которые обработчик регулярных выражений находит во входной строке. Например, предыдущий пример может быть переписан для вызова метода `Matches` вместо метода `Match` и метода `NextMatch` (Пример 3).

Пример 3. Использование метода Matches:

```
using System;
using System.Text.RegularExpressions;

class Example
{
    static void Main()
    {
        // Допустим в исходной строке нужно найти все числа,
        // соответствующие стоимости продукта
        string input = "Добро пожаловать в наш магазин, вот наши цены: " +
            "1 кг. яблок - 20 руб. " +
            "2 кг. апельсинов - 30 руб. " +
            "0.5 кг. орехов - 50 руб.";

        string pattern = @"\"b(\\d+\\W?руб)";
        Regex regex = new Regex(pattern);

        // Достигаем того же результата что и в предыдущем примере,
        // используя метод Regex.Matches() возвращающий MatchCollection
        foreach (Match match in regex.Matches(input))
        {
            Console.WriteLine(match.Groups[1].Value);
        }
    }
}
```

Пример 4. Использование метода Regex.Replace():

```
using System;
using System.Text.RegularExpressions;

class Example
{
    static void Main()
    {
        // Допустим в исходной строке нужно заменить "руб." на "$",
```

```

// а стоимость переместить после знака $
string input = "Добро пожаловать в наш магазин, вот наши цены:
\n" +
    "\t 1 кг. яблок - 20 руб. \n" +
    "\t 2 кг. апельсинов - 30 руб. \n" +
    "\t 0.5 кг. орехов - 50 руб. \n";

Console.WriteLine("Исходная строка:\n {0}", input);
// В шаблоне используются 2 группы
string pattern = @"\\b(\\d+)\\W?(руб.)";

// Строка замены "руб." на "$"
string replacement1 = "$$$1"; // Перед первой группой ставится
знак $,
                                // вторая группа удаляется без замены

input = Regex.Replace(input, pattern, replacement1);
Console.WriteLine("\nВидоизмененная строка: \n" +input);
}
}

```

```

Исходная строка:
Добро пожаловать в наш магазин, вот наши цены:
    1 кг. яблок - 20 руб.
    2 кг. апельсинов - 30 руб.
    0.5 кг. орехов - 50 руб.

Видоизмененная строка:
Добро пожаловать в наш магазин, вот наши цены:
    1 кг. яблок - $20
    2 кг. апельсинов - $30
    0.5 кг. орехов - $50

```

Рисунок 3 - Выполнение программы с использованием метода
Regex.Replace()

Для закрепления темы давайте рассмотрим еще один пример использования регулярных выражений, где будем искать в исходном тексте слово «сериализация» и его однокоренные слова, при этом выделяя в консоли их другим цветом.

Пример 5. Использование методов для нахождения однокоренных слов.

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
```

```
namespace ConsoleApplication1
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            string myText = @"Сериализация представляет собой процесс
сохранения объекта на диске.
```

В другой части приложения или даже в совершенно отдельном приложении может производиться

десериализация объекта, возвращающая его в состояние, в котором он пребывал до сериализации.";

```
            const string myReg = "co";
```

```
            MatchCollection myMatch = Regex.Matches(myText,myReg);
```

```
            Console.WriteLine("Все вхождения строки \"{0}\" в исходной
строке: ",myReg);
```

```
            foreach (Match i in myMatch)
```

```
                Console.Write("\t"+i.Index);
```

```
            // Усложним шаблон регулярного выражения
```

```
            // введя в него специальные метасимволы
```

```
            const string myReg1 = @"\b[с,д]\S*ериализац\S*";
```

```
            MatchCollection
```

```
                match1
```

```
            =
```

```
Regex.Matches(myText,myReg1,RegexOptions.IgnoreCase);
```

```
            findMyText(myText,match1);
```

```
            Console.ReadLine();
```

```
        }
```



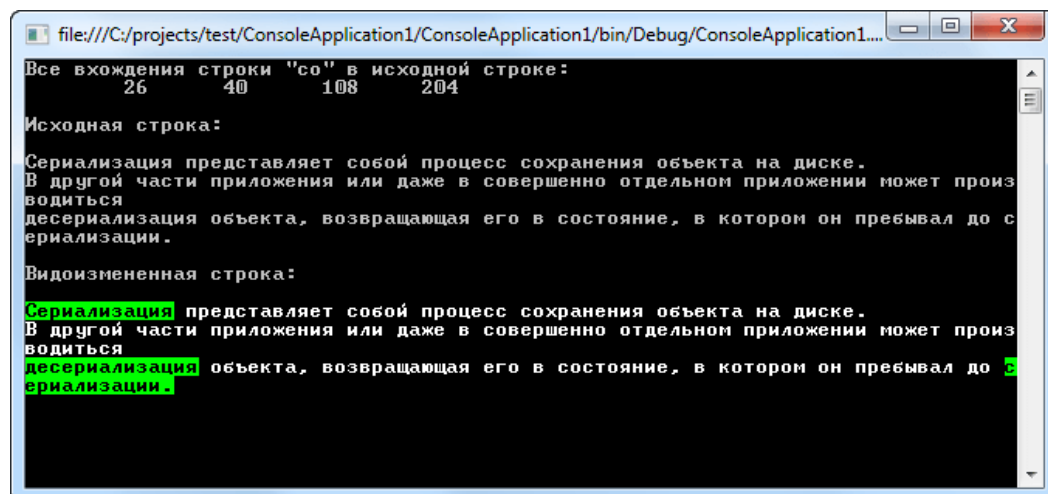
```

static void findMyText(string text, MatchCollection myMatch)
{
    Console.WriteLine("\n\nИсходная
строка:\n\n{0}\n\nВидоизмененная строка:\n",text);

    // Реализуем выделение ключевых слов в консоли другим
цветом
    for (int i = 0; i < text.Length; i++)
    {
        foreach (Match m in myMatch)
        {
            if ((i >= m.Index) && (i < m.Index+m.Length))
            {
                Console.BackgroundColor = ConsoleColor.Green;
                Console.ForegroundColor = ConsoleColor.Black;
                break;
            }
            else
            {
                Console.BackgroundColor = ConsoleColor.Black;
                Console.ForegroundColor = ConsoleColor.White;
            }
        }
        Console.Write(text[i]);
    }
}
}
}

```

Результат работы данной программы:



```
file:///C:/projects/test/ConsoleApplication1/ConsoleApplication1/bin/Debug/ConsoleApplication1....
Все вхождения строки "со" в исходной строке:
      26      40      108      204
Исходная строка:
Сериализация представляет собой процесс сохранения объекта на диске.
В другой части приложения или даже в совершенно отдельном приложении может произ-
водиться
десериализация объекта, возвращающая его в состояние, в котором он пребывал до с-
ериализации.
Видоизмененная строка:
Сериализация представляет собой процесс сохранения объекта на диске.
В другой части приложения или даже в совершенно отдельном приложении может произ-
водиться
десериализация объекта, возвращающая его в состояние, в котором он пребывал до с-
ериализации.
```

Рисунок 4 – Выполнение программы по нахождению однокоренных слов

Для проверки гибкости работы регулярных выражений, подставьте в исходный текст еще несколько слов «сериализация», вы увидите, что они будут автоматически выделены зеленым цветом в консоли.

Варианты индивидуальных заданий

Написать программу, которая:

- а) выводит текст на экран дисплея;
- б) далее – по варианту.

1. По нажатию произвольной клавиши поочередно выделяет каждое предложение текста; определяет количество предложений в тексте.
2. По нажатию произвольной клавиши поочередно выделяет каждое слово текста; определяет количество слов в тексте.
3. По нажатию произвольной клавиши поочередно выделяет каждое слово текста, оканчивающееся на гласную букву; определяет количество таких слов в тексте.
4. По нажатию произвольной клавиши поочередно выделяет каждое предложение текста в последовательности 2, 1, 3.
5. По нажатию произвольной клавиши поочередно выделяет каждое из слов текста, у которых первый и последний символы совпадают; определяет количество таких слов в тексте.
6. По нажатию произвольной клавиши поочередно выделяет каждое слово текста, начинающееся на гласную букву; определяет количество таких слов в тексте.

7. Определяет количество символов в самом длинном слове; по нажатию произвольной клавиши поочередно выделяет каждое слово текста, содержащее максимальное количество символов.

8. Определяет количество символов в самом коротком слове; по нажатию произвольной клавиши поочередно выделяет каждое слово текста, содержащее минимальное количество символов.

9. Определяет в каждом предложении текста количество символов, отличных от букв и пробела; по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении – поочередно все символы, отличные от букв и пробела.

10. Определяет количество предложений текста и количество слов в каждом предложении; по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении – поочередно все слова.

11. Определяет количество букв „а“ в последнем слове текста; по нажатию произвольной клавиши выделяет последнее слово текста, а в выделенном слове – поочередно все буквы „а“.

12. Определяет самую длинную последовательность цифр в тексте (считать, что любое количество пробелов между двумя цифрами не прерывает последовательности цифр); по нажатию произвольной клавиши поочередно выделяет каждую последовательность цифр, содержащую максимальное количество символов.

13. Определяет порядковый номер заданного слова в каждом предложении текста (заданное слово вводится с клавиатуры); по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении – заданное слово.

14. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводить с клавиатуры); выводит текст на экран дисплея ещё раз, выкидывая из него заданное слово и удаляя лишние пробелы.

15. По нажатию произвольной клавиши поочередно выделяет в тексте заданные слова, которые нужно поменять местами (заданные слова вводить с клавиатуры); выводит текст на экран дисплея ещё раз, меняя в нём местами заданные слова и удаляя лишние пробелы.

16. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводить с клавиатуры); выводит текст на экран дисплея ещё раз, заключая заданное слово в кавычки, и поочередно выделяет заданное слово вместе с кавычками.

17. Выводит текст на экран дисплея ещё раз, вставляя в каждое предложение в качестве последнего заданное слово, введенное с клавиатуры в качестве исходных данных; по нажатию произвольной клавиши поочередно выделяет в тексте вставленное слово.

18. По нажатию произвольной клавиши поочередно выделяет в тексте лишние пробелы между словами; выводит текст на экран дисплея ещё раз, удаляя лишние пробелы между словами и начиная каждое предложение с новой строки.

19. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводится с клавиатуры); выводит текст на экран дисплея ещё раз, заменяя в заданном слове строчные буквы прописными.

20. Определяет наибольшее количество подряд идущих пробелов в тексте; по нажатию произвольной клавиши поочередно выделяет каждую из последовательностей пробелов максимальной длины.

21. Определяет в каждой строке текста количество прописных букв; по нажатию произвольной клавиши поочередно выделяет каждое слово, начинающееся с прописной буквы, а в выделенном слове – прописные буквы.

22. По нажатию произвольной клавиши поочередно выделяет в тексте слово с заданной буквой; выводит на экран дисплея ещё раз те слова, в которых заданная буква встречается более одного раза.

23. По нажатию произвольной клавиши поочередно выводит фрагменты текста, отделенные знаками препинания; выводит на экран дисплея сведения о знаках препинания по строкам в виде: знак препинания – количество.

24. По нажатию произвольной клавиши поочередно выводит построчно фрагменты текста, разделенные символом горизонтальной табуляции; выводит на экран дисплея общее количество символов табуляции в тексте.

25. Выводит текст на экран дисплея ещё раз, разделяя знаками переноса каждое слово на слоги; по нажатию произвольной клавиши поочередно выделяет в каждой строке текста слово с наибольшим количеством слогов.

26. По нажатию произвольной клавиши поочередно выделяет в тексте слова, после которых стоит знак препинания; выводит текст на экран ещё раз, выделяя знаки препинания.

27. По нажатию произвольной клавиши выводит количество десятичных чисел по строкам; выводит текст на экран дисплея ещё раз, заменяя десятичные числа на шестнадцатеричные.

28. По нажатию произвольной клавиши поочередно выделяет каждое число в тексте; выводит текст на экран дисплея ещё раз, заменяя числа пробелами.

29. По нажатию произвольной клавиши поочередно выделяет в тексте слова с заданной буквой (вводится с клавиатуры); выводит на экран дисплея ещё раз те слова, в которых нет заданной буквы.

30. По нажатию произвольной клавиши поочередно выделяет в тексте каждые первое и второе слова с первыми строчными гласными буквами.

Шкала оценивания индивидуальных заданий

| Примеры | 1-5 балла |
|---|-------------|
| Примеры + Индивидуальное задание (реализация программы без использования регулярных выражений) | 1-8 баллов |
| Примеры + Индивидуальное задание (реализация программы с помощью регулярных выражений) | 1-10 баллов |

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. Скриншоты выполнения примеров
5. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
6. Вывод по лабораторной работе