Практическая работа 24
Использование регулярных выражений

Цель занятия: Получить практические навыки использования регулярных выражений

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio

Краткие теоретические сведения

Регулярные выражения – это один из способов поиска подстрок (соответствий) в строках. Осуществляется это с помощью просмотра строки в поисках некоторого шаблона. Поддержка регулярных выражений в .Net выполняется классами пространства имен System.Text.RegularExpressions.

Основные классы:

Regex - постоянное регулярное выражение.

Match - предоставляет результаты очередного применения всего регулярного выражения к исходной строке.

MatchCollection - предоставляет набор успешных сопоставлений, при итеративном применении образца регулярного выражения к строке.

Capture - предоставляет результаты отдельного захвата подвыражения.

Group - предоставляет результаты для одной регулярной группы.

GroupCollection - предоставляет коллекцию найденных групп и возвращает набор групп как одно соответствие.

CaptureCollection - предоставляет последовательность найденных подстрок и возвращает наборы соответствий отдельно для каждой группы.

Для того чтобы использовать Regex в своих программах необходимо в список используемых пространств имен добавить:

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

Далее, в коде самой программы необходимо создать экземпляр Regex:

```
Regex newReg = new Regex(pattern, options);
```

Все найденные соответствия в тексте помещаются в тип MatchCollectionMatchCollection matches;

И далее в этот объект поместить текст, в котором необходимо произвести поиск:

matches = optionRegex.Matches(text);

В результате в matches появляются все результаты парсинга. Мы можем посмотреть сколько их (matches.Count), можем узнать значение конкретного элемента (matches[N].Value)

pattern – образец или условие для поиска, например, если необходимо найти слово“не” в строке “не может быть”, то pattern для поискабудет выглядеть так: @"не". В простейшем случае, можно обойтись без условия поиска, тогда найдено будет значение точно повторяющее pattern. Можно получить как значение найденного совпадения так и их количество, как видно из примера ниже.

Пример 1:

```
string pattern = @"не";
string text = "неможетбыть";
Regex newReg = new Regex(pattern);
MatchCollection matches = newReg.Matches(text);
foreach(Match mat in matches)
{
   Console.WriteLine("Значениенайденногообьекта {0}",mat.Value);
}
Console.WriteLine("Числонайденныхсовпадений{0}",matches.Count);
```

Теперь остановимся на условиях поиска, или options. Для начала необходимо создать условие поиска:

RegexOptions option = RegexOptions.<условие поиска>;

Ниже приведены условия поиска.

IgnoreCase – находит совпадения независимо от регистра, т.е. прописными или строчными буквами в строке написано слово.

IgnorePatternWhitespace – устраняет из шаблона неизбежные пробелы и включает комментарии помеченные «#».

Compiled – указывает что регулярное выражение скомпилировано в сборку. Это порождает более быстрое исполнение но увеличивает время запуска.

CultureInvariant – указывает игнорирование региональных языковых различий.

ExplicitCapture – указывает что единственные допустимые записи являются явно поименованными или пронумерованными группами в форме(?<name>…)

Multiline – Многострочный режим. Изменяет значения символов “^” и“$” так что они совпадают соответственно в начале и конце каждой строки, а не только в начале и конце целой строки.

RightToLeft – указывает что поиск будет выполнен справа на лево, а не слева на право.

Singleline – однострочный режим.

None – указывает на отсутствие заданных параметров.

Пример 2:

```
string pattern = @"не";
string text = "Не может быть совсем не может быть";
RegexOptions option=RegexOptions.IgnoreCase;
Regex newReg = new Regex(pattern,option);
MatchCollection matches = newReg.Matches(text);
foreach(Match mat in matches)
{
    Console.WriteLine("Значениенайденногообьекта {0}",mat.Value);
}
Console.WriteLine(“Числонайденныхсовпадений{0}”,matches.Count);
```

В этом случае будет осуществлен поиск слова “не” не зависимо от того какими буквами (прописными или строчными) оно написано. И в результате будут найдены оба слова в строке. При желании и опыте, код может быть оптимизирован.

Используя специальные символы можно создавать более сложные шаблоны для поиска:

^ - указывает на то, что поиск должен начинаться с начала строки, например шаблон (@”^не”) найдет “не” в строке, если она начинается с него: “Не может быть совсем не может быть не”.

$ -указывает на то что поиск должен производиться в конце строки, шаблон (@”не$”) найдет “не” в той же самой строке, только если она заканчивается им.

{n} – указывает точное число вхождений в строку, например шаблон (@"не{2}") найдет слово “нее” в строке.

{n,} – указывает число вхождений не менее n, т.е. шаблон (@"не{2,}") найдет слова “нее”, “неее” и т.д.

{n,m} – указывает число вхождений-n и количество символов вхождения-m, то есть шаблон (@"не{2,4}") найдет все слова где количество букв “е” больше 2, но определит только количество букв ”е” равное 4. Т.е. в слове “нееееееееее” он найдет только “неееe”.

+ - Соответствует 1 или более предшествующих выражений. Например, "не+" соответствует "не" и "нее", но не соответствует "н".

- Соответствует 0 или более вхождений предшествующего выражения. Например, 'не' соответствует "н" и "нее".

? - Соответствует 0 или 1 предшествующих выражений. Например, 'бы(ло)?' соответствует "бы" в "бы" или "было".

В квадратных скобках можно указать диапазон букв или цифр для поиска, например [A-Z] или [0-9]

Пример:

Поиска времени в формате 00:00:00

```
string s1 = "Не время для драконов 00:00:00";
Regex reg = new Regex(@"[0-9]+:[0-9]+:[0-
9]+",RegexOptions.IgnoreCase);
MatchCollection mc = reg.Matches(s1);
        foreach (Match mat in mc)
{
            Console.WriteLine(mat.ToString());
        }
        Console.WriteLine(mc.Count.ToString());
```

Задания

1 Изучить теоретические сведения и задание к работе

2 В соответствии с вариантом задания составить отлаженную программу.

Порядок выполнения работы

Создадим текстовый файл, в котором неправильно расставлены пробелы, после знаков препинания они не стоят:

Компания Avast опубликовала результаты опроса,проведенного среди российских пользователей.Оказалось,что 42% россиян сталкивались с фишингом,при этом жертвами таких атак стали 27% пользователей.

Две трети респондентов,подвергшихся фишинговой атаке,пострадали,занимаясь личными вопросами,одна треть—решая рабочие задачи.

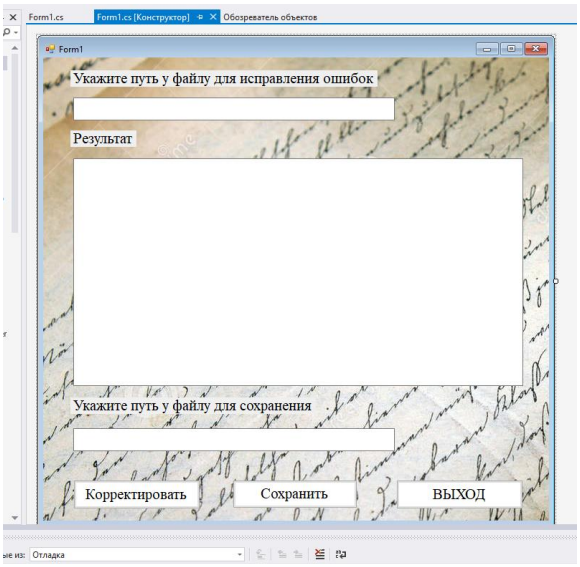
Чуть больше четверти опрошенных (27%) заявили,что им пришлось сменить пароли от аккаунтов,13% заявили,что у них украли деньги, и у 11% украли личные данные.Еще 11% жертв пришлось аннулировать кредитные или дебетовые карты.

Среди тех,кто понес финансовые потери,43% потеряли до 3500 рублей,каждый пятый (20%) потерял от 3500 до 6999 рублей,11% потеряли от 7000 до 13 999 рублей,5%—от 14 000 до 20 999 рублей и один из пяти (20%) более 21 000 рублей.

Интересно,что трое из пяти (61%) россиян,пострадавших от фишинга,никому не сообщили о мошенничестве.Основные причины,по которым люди не уведомляют о таких инцидентах:думают,что атака не стоит хлопот (30%),не знают,кому сообщать об этом (29%),уверены в том,что все равно ничего не изменится,если они сообщат (29%),считают,что полученная мошенниками информация ничего не стоит (23%).

Из жертв фишинга,которые сообщили об атаке,почти половина (49%) заяви ли о мошенничестве в полицию,43%—в компанию,сотрудником которой притворялся злоумышленник,26% рассказали о произошедшем своим коллегам.

Создадим форму для запуска исправлений текста.

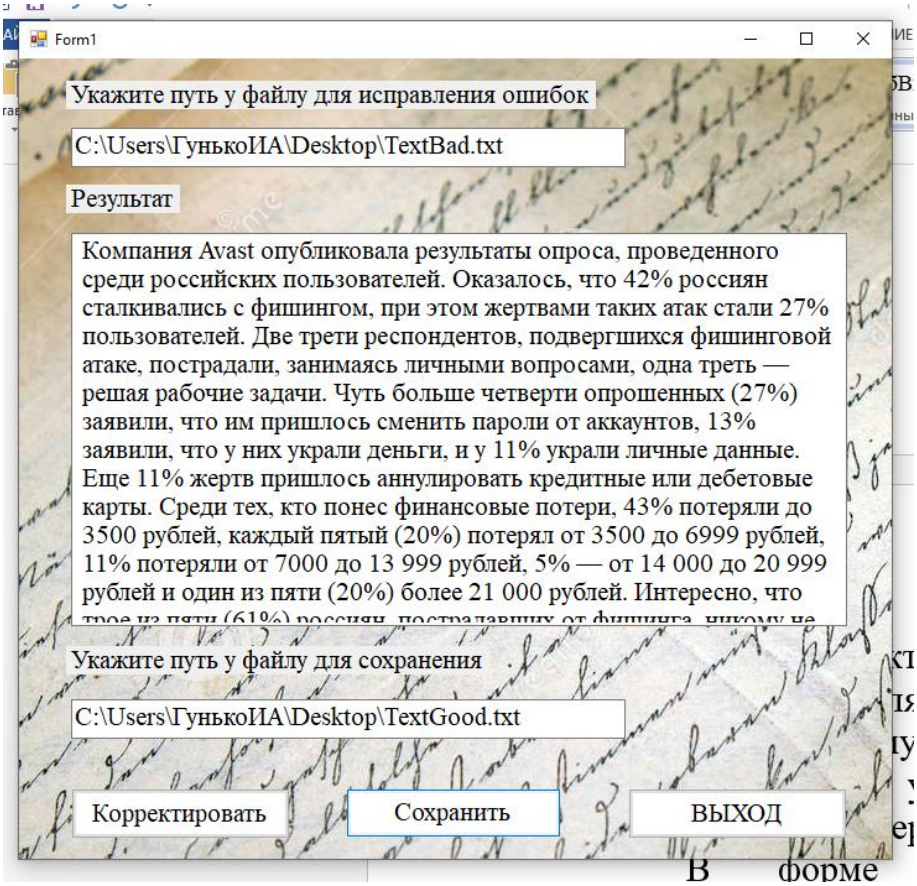


Нажатию кнопки Корректировать запишем код:

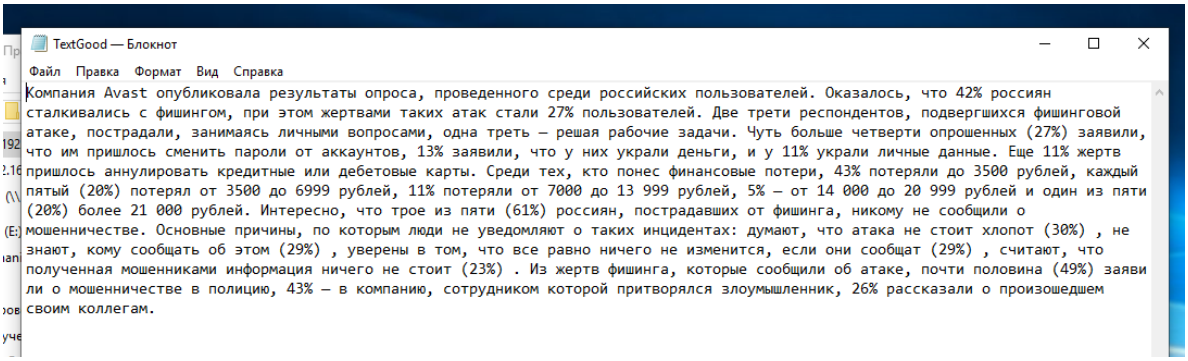
```
}
public string line = null;
ссылка: 1
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (textBox1.Text == "") MessageBox.Show("Укажите путь к файлу");
            else
            {
            using (StreamReader reader = File.OpenText(textBox1.Text))
            {
                line = reader.ReadToEnd();
                string line1 = null;
                int k = 0;
                for (int i = 0; i < line.Length - 1; i++)
                {
                    line1 += line[i];
                    if (char.IsPunctuation(line[i]))
                    {
                        k++;
                        line1 += " ";
                    };
                    k++;
                }
                line = new Regex("–").Replace(line1, " –");
                line = new Regex(@"\s+").Replace(line, " ");
                line = new Regex(@" \)").Replace(line, @"\)" );
                line = new Regex(@"\( ").Replace(line, @"\(" );
                line = new Regex(@"\\").Replace(line, "");
                //MessageBox.Show(line);
                textBox2.Text = line;
            }
        }
    }
    catch (FileNotFoundException)
    {
        MessageBox.Show("Файл не найден");
    }
```

Здесь корректируем текст в два приема, сначала к каждому знаку препинания добавляется пробел. Но если пробел уже присутствовал, то их станет два, поэтому вторым этапом с помощью Regex заменяем несколько пробелов на один, у скобок удаляем пробелы, где они не нужны, а к тире добавим пробел спереди.

В форме укажем путь к файлу с ошибками (C:\Users\ГунькоИА\Desktop\TextBad.txt), нажмем кнопку Корректировать, убедимся на экране в правильности текста, пропишем путь для исправленного файла (C:\Users\ГунькоИА\Desktop\TextGood.txt) и сохраним результат.



В текстовом файле TextGood.txt



Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Листинг программы
7 Вывод

Варианты заданий

1 Дана строка, проверить является ли она электронным адресом.

2 Дан текстовый файл, состоящий из нескольких строк. Найти в нем слово «мир», в другой текстовый файл записать это слово и количество встречающихся экземпляров.

3 Дан текст, найти в нем слово «авто» и однокоренные с ним слова, и выделить их цветом.

4 Дан текстовый файл, состоящий из нескольких строк. Проверить начинается ли он с введенного слова. Выдать сообщение о результате проверки

5 Дан текст, найти в нем слова, заканчивающиеся на букву «а», и вывести эти слова.

6 Дан текстовый файл, состоящий из нескольких строк. Проверить заканчивается ли он введенным словом. Выдать сообщение о результате проверки

7 Дан текст и слово, вывести строки, в которых оно встречается и заканчивающиеся «?».

8 Дан текстовый файл, состоящий из нескольких строк. Найти в нем введенное слово, организуя поиск с конца в начало.

9 Дана строка, проверить является ли она номером телефона.

10 Дан текст и слово, вывести все слова из текста не абсолютно совпадающие с введенным словом.

11 Дана строка, проверить является ли она IP-адресом.

12 Дан текст, найти в нем введенное слово, написанное прописными буквами.

13 Дан текстовый файл, состоящий из нескольких строк. Найти в нем слово «куб». В другой файл записать строки, в которых оно встречается и заканчивающиеся «!».

14 Дан текст, найти в нем слова, содержащие введенные буквы, и вывести их количество.

15 Дан текстовый файл, состоящий из нескольких строк. Проверить начинается ли он введенным словом. Выдать сообщение о результате проверки

16 Дан текстовый файл, состоящий из одной строки, проверить является ли она IP-адресом.

17 Дан текст, найти в нем слово «пароль» и однокоренные с ним слова, и выделить их цветом.

18 Дана строка, проверить является ли она электронным адресом.

19 Дан текстовый файл, состоящий из нескольких строк. Найти в нем слово «мир», в другой текстовый файл записать это слово и количество встречающихся экземпляров.

20 Дан текст, найти в нем слово «мега» и однокоренные с ним слова, и выделить их цветом.

21 Дан текстовый файл, состоящий из нескольких строк. Найти в нем слово «Ростов». В другой файл записать строки, в которых оно встречается.

22 Дан текстовый файл и слово, вывести все слова из файла, не абсолютно совпадающие с введенным словом.

23 Дан текст, найти в нем введенное слово, написанное прописными буквами.

Контрольные вопросы

1. Для чего предназначены регулярные выражения?
2. Какое пространство имен необходимо использовать для работы с регулярными выражениями?
3. Какой класс является центральным при работе с регулярными выражениями?
4. Основная функциональность регулярных выражений.
5. Что такое метасимволы и квантификаторы?

Используемая литература

1 Гниденко, И. Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.

2 Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.

3 Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.

4 Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.

5 http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.

6 https://metanit.com/sharp/net/1.2.php

7 https://vscode.ru/prog-lessons/kak-uznat-svoy-host-i-ip-c-sharp.html