

# Регулярные выражения

<https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions>

# Регулярные выражения

*System.Text.RegularExpressions*

RegExp, RegEx

- ▶ формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

строка-образец (англ. pattern, «шаблон», «маска»), состоящая из символов и метасимволов и задающая правило поиска.

<https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions>

# 1) Подключить

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

# 2) Создать экземпляр Regex:

```
Regex newReg = new Regex(pattern, RegexOptions.None);
```

RegexOptions option =  
RegexOptions.<условие поиска>;

- **IgnoreCase** игнорирует регистр символов
  - **ExplicitCapture** буквальное соответствие
  - **CultureInvariant** игнорировать национальные установки
  - **IgnorePatternWhitespace** удаляет из строки пробелы и разрешает комментарии, начинающиеся со знака #
  - **Multiline** текст надо рассматривать в многострочном режиме: ^ и \$ символы начала и конца к каждой линии
  - **None**
  - **RightToLeft**
  - **Singleline** однострочный режим
  - **Compiled** компилируется в сборку

3) Все найденные соответствия в тексте помещаются в тип MatchCollection

```
MatchCollection matches;
```

4) Поместить текст, в котором необходимо произвести поиск:

```
matches= newReg.Matches(textoriginal);
```

в matches появляются все результаты парсинга

5) сколько их

```
int i = matches.Count;
```

6) узнать значение конкретного элемента

```
String s= matches[N].Value;
```

## 7) Проверка на существование

```
if (newReg.IsMatch(textoriginal))
```

## 8) Просмотр найденных

```
Regex regex = new Regex(pattern);
```

```
Match match = regex.Match(textoriginal);
```

```
while (match.Success)
```

```
{
```

```
    int index = match.Index;
```

```
    String findStr = match.Value;
```

```
    match = match.NextMatch();
```

```
}
```

```
Group allgroup = match.Groups[1];
```

вернуть найденное соответствие из  
исходной строки

к следующему совпадению

В шаблоне обращение к группе (одни  
круглые скобки) - ссылаемся на найденное  
значение через свойство Groups

# Варианты использования

1) Подходит ли строка под регулярное выражение - Regex.IsMatch () - true, false

IsMatch(string input, int startat)

строка,

позиция для поиска

IsMatch(string input)

IsMatch(string input, string pattern,  
System.Text.RegularExpressions.RegexOptions options)

IsMatch(string input, string pattern)

## 2) Замена текста

Regex.Replace ()

Replace (string input, string pattern, string replacement)

Replace(string input, string replacement)

Replace(string input, string replacement, int count)

Replace(string input, string pattern, string replacement,

System.Text.RegularExpressions.RegexOptions options)

### 3) Разделение одной строки на массив строк

`string[]`

`Split(string input, string pattern)`

`Split(string input, int count)`

# Элементы языка регулярных выражений

- ▶ <http://msdn.microsoft.com/ru-ru/library/az24scfc.aspx>
- ▶ система сжатого описания некоторого множества строк с помощью шаблонов

- ▶ Обычные символы (литералы)
- ▶ специальные символы (метасимволы)
- ▶ Escape-символы
- ▶ Классы символов
- ▶ Привязки
- ▶ Конструкции группирования
- ▶ Кванторы
- ▶ Конструкции обратных ссылок
- ▶ Конструкции изменения
- ▶ Подстановки
- ▶ Прочие конструкции

# Escape-символы

\t – символ табуляции

\r – символ возврата каретки

\n – новая строка

\e – символ escape

## Метасимволы

[группа\_символов] - один из группы

[^группа\_символов] – отрицание

[первый-последний] - диапазон символов

\w - любой алфавитно-цифровой знак

\W - не символ

\d - любая десятичная цифра

\D - не цифра

\s - пробел Unicode \S не пробел Unicode

[^aei] - laetr → | t r

[0-9a-fA-F]

# Привязки (якоря)

атомарные утверждения нулевой ширины, приводят к успеху или сбою сопоставления, в зависимости от текущей позиции в строке

- ▶ ^ или \A Начало строки
- ▶ \$ или \z Конец строки
- ▶ \b Граница слова
- ▶ \B Не граница слова
- ▶ И т.д.

$^{\backslash d\{1\}}$	$\rightarrow$	948basjfb	$\rightarrow$	9
$\backslash d\{3\}\$$	$\rightarrow$	basjfb948	$\rightarrow$	948

# Конструкции группирования

- ▶ часть выражений регулярных выражений (часть\_выражения)  
"захвата" и сохранения их во встроенных переменных \$1, \$2, ..., \$9 или имя

(on.) → one ->\$1

(?:one) → группировка (не запоминает символы, соответствующие этой группе)

# Квантор или множители

количество вхождений предшествующего элемента (знака, группы или класса знаков)

- ▶ \* пред. шаблон , 0 или любое число раз ( $\{0,\}$ ).
- ▶ + пред. хотя бы 1 раз ( $\{1,\}$ ). «о+z»
- ▶ ? пред. 0 или 1 раз  $\{0,1\}$ .

be+ beennn → bee, bent → be

\d\* → любое количество цифр, идущих подряд

ru?n → run или rn

- ▶  $\{m,n\}$  - повторений может быть от  $m$  до  $n$  включительно.
- ▶  $\{m,\}$   $m$  и более повторений.
- ▶  $\{,n\}$  не более  $n$  повторений.
- ▶ и т.д.

```
\d{3,5} → 234 2345 23456
```

# Конструкции изменения сопоставление по принципу “либо-либо”

| Соответствует любому элементу,  
разделенному вертикальной чертой (|)

th(e|is|at)

# Приоритеты

<b>Наименование</b>	<b>Обозначение</b>
Круглые скобки (группа)	( )(?:...)
Множители	?, +, *, {m,n}
Последовательность и фиксация	abc, \A, \Z
Дизъюнкция	

# Примеры:

Чтение даты в формате YYYY-MM-DD:

```
(\d{4})-(\d\d)-(\d\d)
```

YYYY - в \$1, MM - в \$2, DD - в \$3

```
\s\d+\.\d+\.
```

Пробел цифры точка цифры точка

```
@"[0-9]+:[0-9]+:[0-9]+" ??????????
```

\w+

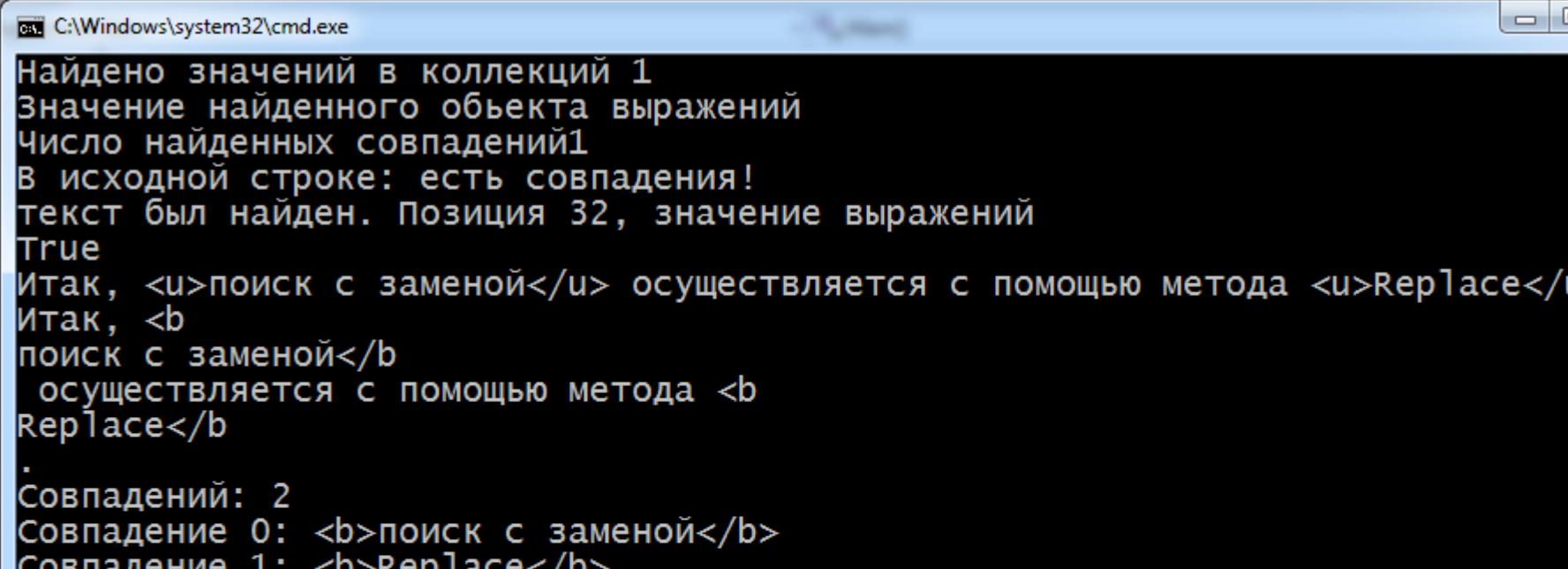
Любой текстовый символ, не являющийся пробелом, символом табуляции - слово (цифра или буква)

\b\w+([\.\w]+)\*\w@\w(([\.\w]\*\w+)\*[.\w]{2,3})

e-mail

- ▶ ^[a-zA-Z0-9\_]{8,20}\$
- ▶ \.(?:jp(?:e?g|e|2)|gif|png|tiff?|bmp|ico)\$

```
string text = "Итак, <b>поиск с заменой</b> осуществляется с помощью  
метода <b>Replace</b>.";  
  
bool matchb = Regex.IsMatch(text, "<b>(.*)</b>");  
  
string textOut = Regex.Replace(text, "<b>(.*)</b>", @"<u>$1</u>",  
    RegexOptions.IgnoreCase);  
  
string[] splitStr = Regex.Split(text, ">");  
  
MatchCollection matchesn = Regex.Matches(text, "<b>(.*)</b>",  
    RegexOptions.IgnoreCase);  
}
```



```
C:\Windows\system32\cmd.exe  
Найдено значений в коллекции 1  
Значение найденного объекта выражений  
Число найденных совпадений1  
В исходной строке: есть совпадения!  
текст был найден. Позиция 32, значение выражений  
True  
Итак, <u>поиск с заменой</u> осуществляется с помощью метода <u>Replace</u>  
Итак, <b>  
поиск с заменой</b>  
осуществляется с помощью метода <b>  
Replace</b>  
.Совпадений: 2  
Совпадение 0: <b>поиск с заменой</b>  
Совпадение 1: <b>Replace</b>
```