# Регулярные выражения

Сайт для тестирования: regexstorm.net/tester

*Регулярные выражения* (Regular Expressions/Regex/RegExp) – формальный язык поиска и осуществления манипуляций над строками, основанный на использовании метасимволов.

*Патерн*(pattern) – это строка образец, состоящий из символов и метасимволов. (создает правило поиска)

Результаты работы регулярного выражения:

* Проверка наличия шаблона в заданном тексте,
* Получение подстроки, соответствующей шаблонам,
* Определение группы символов соответствующей части шаблона,
* Удаление найденных подстрок,
* Замена найденных подстрок по определённому шаблону.

Работа с Regex в C#:

* Для работы нужен using System.Text.RegularExpressions;
* Для объявления Regex имяШаблона = new(Шаблон);
* имяШаблона.ИмяМетода(…);

Особенности:

* Синтаксис шаблона в основном **одинаковый** в разных языках.
* Строка замены используется для манипуляции с текстом, и может содержать в себе спец. символы для замены.
* Первым параметром в конструкторе предаётся **шаблон**.
* У Rerex **есть статические методы**

*Match* – находит совпадения.

Класс MatchCollection в C# представляет собой набор успешных совпадений, найденных при итеративном применении шаблона регулярного выражения к входной строке.

*Group* – это результата совпадения в подстроке для одной буквы.

*Класс GroupCollection* – это коллекция найденных букв в одном совпадении.

Методы Regex:

|  |  |
| --- | --- |
| Методы |  |
| IsMatch(string input) | возвращает true, если шаблон соответствует входной строке. |
| Match(string input) | возвращает первое вхождение шаблона в входной строке. |
| Matches(string input) | возвращает все вхождения шаблона в входной строке |
| Replace(string input, string replacement) | заменяет все вхождения шаблона в входной строке строкой замены. |
| Split(String) | разделяет входную строку на массив подстрок в позициях, определённых шаблоном регулярного выражения. |

Алгоритм работы с регулярным выражением:

1. Создать входную строку и строку шаблона

string input = “…”;

string pattern = @“…”;

1. Создать объект

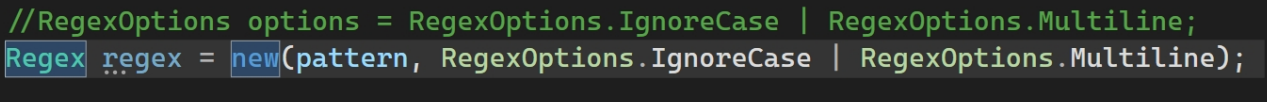
Regex regex = new(pattern);

В конструкторе можно указать параметр типа RegexOptions.

Примеры опций:

|  |  |
| --- | --- |
| Опция |  |
| RegexOptions.Compiled | при установке этого значения регулярное выражение компилируется в сборку, что обеспечивает более быстрое выполнение |
| RegexOptions.CultureInvariant | при установке этого значения будут игнорироваться региональные различия |
| RegexOptions.IgnorePatternWhitespace | удаляет из строки пробелы и разрешает комментарии, начинающиеся со знака # |
| RegexOptions.Singleline | при данном режиме символ "." соответствует любому символу, в том числе последовательности "\n", которая осуществляет переход на следующую строку |
| RegexOptions.IgnoreCase | при установке этого значения будет игнорироваться регистр |
| RegexOptions.Multiline | указывает, что текст надо рассматривать в многострочном режиме. При таком режиме символы "^" и "$" совпадают, соответственно, с началом и концом любой строки, а не с началом и концом всего текста |
| RegexOptions.RightToLeft | приписывает читать строку справа налево |

Пример реализации:



1. Применение регулярного выражения для поиска и замены совпадения.

Применение поиска:

//1

string input = “Hello world 123 456”;

string pattern = @“o”;

//2

Regex regex =new(pattern, RegexOptions.IgnoreCase)

//3

if (regex.IsMatch(input))

Console.WriteLine(“совпадение найдено”);

Получение совпадений:

//3

Match match = regex.IsMatch(input)

if (match.Succses)

Console.WriteLine($“совпадение найдено {match.Value}”);

Перебор всех групп совпадений:

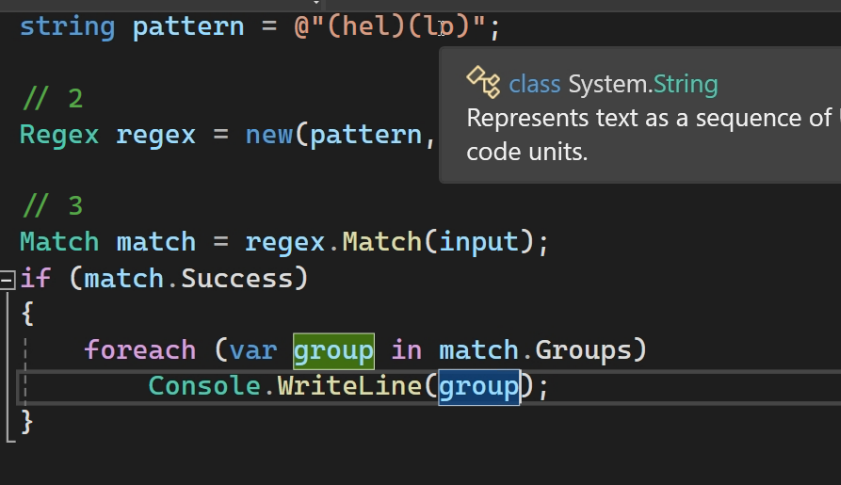
Группы записываются в шаблоне в круглых скобках.

*Всё совпадение* – это нулевая группа. Номер остальных нумеруется с единицы.

//match.Groups

//match.Group[0] = match.Value;

//match[i] i-ая группа



Перебор всех совпадений соответствующего шаблона **через Match**

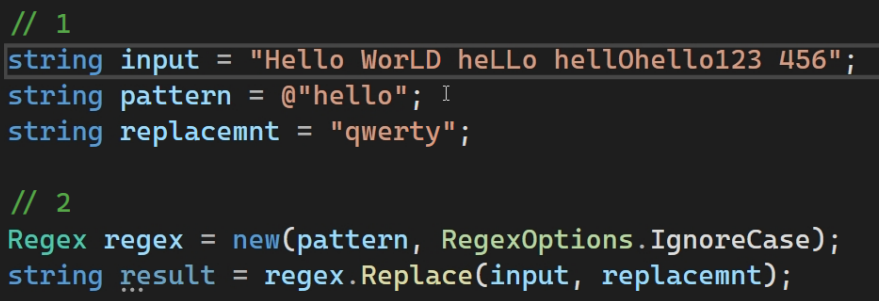


через MatchCollection



Пример замены подстрок:

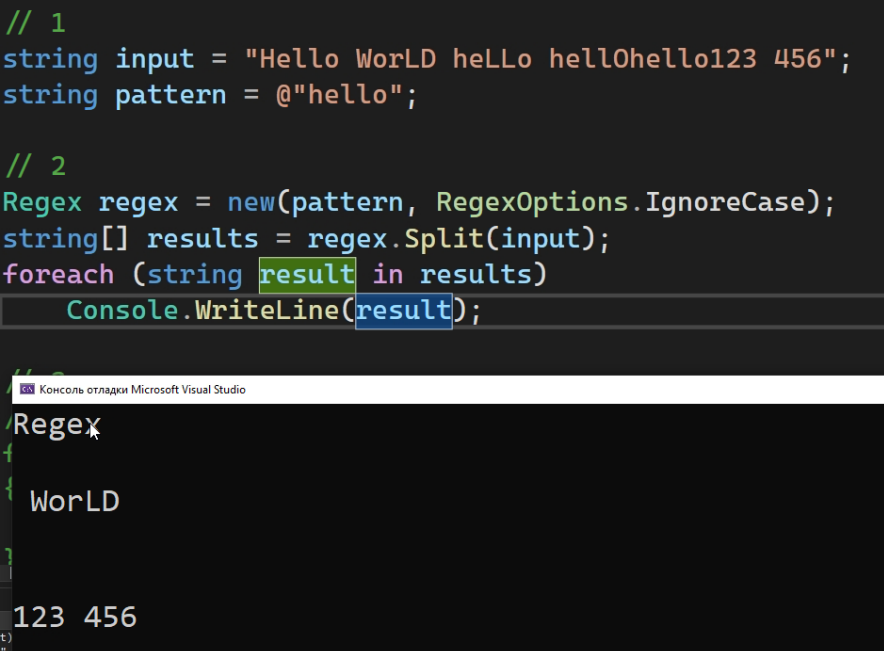
1. Создать входную строку, строку шаблона и создание строки с шаблоном замены
2. Вызвать Replace у регулярного выражения.



Разделение на массив строк:

Разделителем могут быть регулярные выражения.

1. Создать входную строку и строку шаблона.
2. Вызвать Split у регулярного выражения.



Пример создания объекта для match:

result = Regex.Split(input, pattern);

## Классы символов

{n} – ищем количество повторений n

{n, m} – количество повторений от n до m

{n,} – ищем количество повторений от n до максимума

{,n} – количество повторений от 0 до n

|  |  |
| --- | --- |
| Символ |  |
| […] | это любой из символов в скобках, для поиска мы указываем символ или диапазон |
| ^ | соответствие должно начинаться в начале строки (например, выражение @"^пр\w\*" соответствует слову "привет" в строке "привет мир") |
| $ | конец строки (например, выражение @"\w\*ир$" соответствует слову "мир" в строке "привет мир", так как часть "ир" находится в самом конце) |
| . | знак точки определяет любой одиночный символ (например, выражение "м.р" соответствует слову "мир" или "мор") |
| \* | предыдущий символ повторяется 0 и более раз |
| + | предыдущий символ повторяется 1 и более раз |
| ? | предыдущий символ повторяется 0 или 1 раз |
| \s | соответствует любому пробельному символу |
| \S | соответствует любому символу, не являющемуся пробелом |
| \w | соответствует любому алфавитно-цифровому символу |
| \W | соответствует любому не алфавитно-цифровому символу |
| \d | соответствует любой десятичной цифре |
| \D | соответствует любому символу, не являющемуся десятичной цифрой |
| \G | совпадение начинается там, где закончилось предыдущее |
| \b | граница слова между символами \w и \W |
| \B | совпадения не до границы |

Пример:

*[0-9]* – это любая из цифр

*[а-яё]* – это любая из русских букв

*[.,!?-*] – это любой из знаков применения

Для поиска всего КРОМЕ используется **^**

Пример:

[^.,!?-] – всё кроме знаков препинания

[^a-zA-z] – все кроме английских букв

*Квантификаторы* – это количество элементов в последовательности.

\d? – все цифры у которых 0 или 1 повторение

\d\d? – ищем одну или 2 цифры\d?

\d\* – все цифры где 0 и более совпадений

\d+ – все цифры где 1 и более повторений

\w+ – поиск слов

\w+ (- \w+)\*

*Якоря* – это привязки или позиция внутри строки

^\w+ – поиск первого слова в строке

^\+7 – поиск строк начинающихся с +7

\G[a-zA-Z]+\s – ищем слова разделённые пробелом

\bпрограмм\B

Чувствительности к регистру можно указать через параметр или (?i) (без учёта регистра)

(?i)^[a-z]+

**Группировка**

|  |  |  |
| --- | --- | --- |
|  | Пример | |
| (выражение) – захватывает строку соответствующую выражению | | \7(-\d{3}){2}(-\d{2}){2} – пример поиска номера телефона |
| (? ‘имя’выражение) или (? <имя>выражение) – именованная группа | | \7(?<first>(?<second>-\d{3})){2}(-\d{2}){2}) |
| (выражение1| выражение2) | |  |
| (?=выражение) – позитивный просмотр вперёд | | .+(?=group) – получает строку до слова group  (?=group).\* – получает строку после слова group |
| (?!=выражение) – негативный просмотр вперёд | | regex(?!.Match) – поиск regex после которого нет .Match |
| (?<=выражение) –позитивный просмотр назад | | (?<=.\*group).\* – вернёт то ,что после слова group |
| (?<!выражение) –негативный просмотр назад | | (?<!group).\* – ищет подстроки не после слова group |

**Replacement**

|  |  |  |
| --- | --- | --- |
|  |  | Пример: |
| $номерГруппы | замещает часть строки соответствующее указанной группы | (\d+)(\.)(\d+)  Замена: $1,$3 – заменяет точку на запятую |
| ${названиеГруппы} | замещает часть строки соответствующее названию группы | (?<целое>\d+)(?<разделитель>\.)(?<вещественное>\d+)  Замена: ${целое},${вещественное} – заменяет точку на запятую |