*Увлекательный C# - регулярные выражения*

*Стич Назар ИВТ-22 Объектно Ориентированное Программирование*

*https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions*

Регулярные выражения предоставляют мощный, гибкий и эффективный способ обработки текста. Комплексная нотация сопоставления шаблонов регулярных выражений позволяет быстро анализировать большие объемы текста в следующих целях:

поиск определенных шаблонов символов;

проверка текста на соответствие предопределенному шаблону (например, адресу электронной почты);

извлечение, изменение, замена или удаление текстовых подстрок;

добавление извлеченных строк в коллекцию для создания отчета.

Для многих приложений, которые работают со строками или анализируют большие блоки текста, регулярные выражения — незаменимый инструмент.

Принцип работы регулярных выражений

Главный компонент обработки текста с помощью регулярных выражений — это механизм регулярных выражений, представленный в .NET объектом [System.Text.RegularExpressions.Regex](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex). Как минимум, для обработки текста с использованием в регулярных выражений механизму регулярных выражений необходимо предоставить два следующих элемента:

Шаблон регулярного выражения для определения текста.

В .NET шаблоны регулярных выражений определяются специальным синтаксисом или языком, который совместим с регулярными выражениями Perl 5 и добавляет дополнительные возможности, например сопоставление справа налево. Дополнительные сведения см. в разделе [Элементы языка регулярных выражений. Краткий справочник](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expression-language-quick-reference).

Текст, который будет проанализирован на соответствие шаблону регулярного выражения.

Методы класса [Regex](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex) позволяют выполнять следующие операции:

Определить, входит ли шаблон регулярного выражения во входной текст, с помощью метода [Regex.IsMatch](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.ismatch). Пример, в котором для проверки текста используется метод [IsMatch](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.ismatch), см. в статье [Практическое руководство. Проверка строк на соответствие формату электронной почты](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/how-to-verify-that-strings-are-in-valid-email-format).

Получить один или все экземпляры текста, соответствующего шаблону регулярного выражения с помощью метода [Regex.Match](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.match) или [Regex.Matches](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.matches). Первый метод возвращает объект [System.Text.RegularExpressions.Match](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.match), который предоставляет сведения о соответствующем тексте. Второй метод возвращает объект [MatchCollection](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.matchcollection), содержащий один объект [System.Text.RegularExpressions.Match](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.match) для каждого соответствия, обнаруженного в обработанном тексте.

Заменить текст, соответствующий шаблону регулярного выражения, с помощью метода [Regex.Replace](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.replace). Примеры использования метода [Replace](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.replace) для изменения форматов даты и удаления недопустимых символов из строки см. в статьях [Практическое руководство. Исключение недопустимых символов из строки](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/how-to-strip-invalid-characters-from-a-string) и [Пример. Изменение форматов даты](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expression-example-changing-date-formats).

Обзор объектной модели регулярных выражений см. в разделе [Объектная модель регулярных выражений](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/the-regular-expression-object-model).

Дополнительные сведения о языке регулярного выражения см. в разделе ["Язык регулярных выражений" — краткий справочник](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expression-language-quick-reference) или скачивание и печать одного из следующих брошюр:

[Краткий справочник в формате Word (DOCX)](https://download.microsoft.com/download/D/2/4/D240EBF6-A9BA-4E4F-A63F-AEB6DA0B921C/Regular%20expressions%20quick%20reference.docx);

[Краткий справочник в формате PDF (PDF)](https://download.microsoft.com/download/D/2/4/D240EBF6-A9BA-4E4F-A63F-AEB6DA0B921C/Regular%20expressions%20quick%20reference.pdf).

Примеры регулярных выражений

Класс [String](https://learn.microsoft.com/ru-ru/dotnet/api/system.string) включает в себя методы поиска строк и замены, которые можно использовать при поиске литеральных строк в более крупной строке. Регулярные выражения максимально полезны, если требуется найти одну из нескольких подстрок в длинной строке или определить шаблоны в строке, как показано в следующих примерах.

Пример 1. Замена подстроек

Предположим, что список рассылки содержит имена, в которые иногда входит обращение (Mr., Mrs., Miss или Ms.) в дополнение к имени и фамилии. Предположим, вы не хотите включать названия при создании меток конверта из списка. В этом случае можно использовать регулярное выражение для удаления заголовков, как показано в следующем примере:

C#Копировать

using System;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

string pattern = "(Mr\\.? |Mrs\\.? |Miss |Ms\\.? )";

string[] names = { "Mr. Henry Hunt", "Ms. Sara Samuels",

"Abraham Adams", "Ms. Nicole Norris" };

foreach (string name in names)

Console.WriteLine(Regex.Replace(name, pattern, String.Empty));

}

}

// The example displays the following output:

// Henry Hunt

// Sara Samuels

// Abraham Adams

// Nicole Norris

Шаблон (Mr\.? |Mrs\.? |Miss |Ms\.? ) регулярного выражения соответствует любому вхождениям "г-н", "г-жа", "Миссис", "Мисс", "г-жа", или "г-жа". После вызова метода [Regex.Replace](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.replace) сопоставленная строка заменяется на [String.Empty](https://learn.microsoft.com/ru-ru/dotnet/api/system.string.empty" \l "system-string-empty); другими словами, она удаляется из исходной строки.

Пример 2. Определение повторяющихся слов

Случайный повтор слов — это распространенная ошибка при написании текстов. Используйте регулярное выражение для идентификации повторяющихся слов, как показано в следующем примере:

C#Копировать

using System;

using System.Text.RegularExpressions;

public class Class1

{

public static void Main()

{

string pattern = @"\b(\w+?)\s\1\b";

string input = "This this is a nice day. What about this? This tastes good. I saw a a dog.";

foreach (Match match in Regex.Matches(input, pattern, RegexOptions.IgnoreCase))

Console.WriteLine("{0} (duplicates '{1}') at position {2}",

match.Value, match.Groups[1].Value, match.Index);

}

}

// The example displays the following output:

// This this (duplicates 'This') at position 0

// a a (duplicates 'a') at position 66

Шаблон регулярного выражения \b(\w+?)\s\1\b интерпретируется следующим образом:

Развернуть таблицу

| Расписание | Интерпретация |
| --- | --- |
| \b | Начало на границе слова. |
| (\w+?) | Соответствует одному или нескольким символам слова (как можно меньшему количеству). Вместе они формируют группу, к которой можно обращаться как к \1. |
| \s | Соответствует пробелу. |
| \1 | Соответствует подстроке, равной группе с именем \1. |
| \b | Соответствует границе слова. |

Метод [Regex.Matches](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.matches) вызывается с параметрами регулярного выражения [RegexOptions.IgnoreCase](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regexoptions" \l "system-text-regularexpressions-regexoptions-ignorecase). Поэтому операция сопоставления учитывает регистр, а пример указывает, что подстрока "This this" является повтором.

Входная строка содержит подстроку "this? This". Тем не менее, из-за пересекающегося знака препинания он не определяется как дублирование.

Пример 3. Динамическое создание регулярного выражения с учетом языка и региональных параметров

Следующий пример демонстрирует преимущества использования регулярных выражений с гибкими возможностями глобализации .NET. В примере объект [NumberFormatInfo](https://learn.microsoft.com/ru-ru/dotnet/api/system.globalization.numberformatinfo) применяется для определения формата денежных значений в текущих региональных стандартах системы. Затем эти данные используются для динамического создания регулярного выражения, которое извлекает денежные значения из текста. Для каждого совпадения извлекается подгруппа, содержащая только числовые строки, которая преобразуется в значение [Decimal](https://learn.microsoft.com/ru-ru/dotnet/api/system.decimal), после чего рассчитывается промежуточный итог.

C#Копировать

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Text.RegularExpressions;

public class Example

{

public static void Main()

{

// Define text to be parsed.

string input = "Office expenses on 2/13/2008:\n" +

"Paper (500 sheets) $3.95\n" +

"Pencils (box of 10) $1.00\n" +

"Pens (box of 10) $4.49\n" +

"Erasers $2.19\n" +

"Ink jet printer $69.95\n\n" +

"Total Expenses $ 81.58\n";

// Get current culture's NumberFormatInfo object.

NumberFormatInfo nfi = CultureInfo.CurrentCulture.NumberFormat;

// Assign needed property values to variables.

string currencySymbol = nfi.CurrencySymbol;

bool symbolPrecedesIfPositive = nfi.CurrencyPositivePattern % 2 == 0;

string groupSeparator = nfi.CurrencyGroupSeparator;

string decimalSeparator = nfi.CurrencyDecimalSeparator;

// Form regular expression pattern.

string pattern = Regex.Escape( symbolPrecedesIfPositive ? currencySymbol : "") +

@"\s\*[-+]?" + "([0-9]{0,3}(" + groupSeparator + "[0-9]{3})\*(" +

Regex.Escape(decimalSeparator) + "[0-9]+)?)" +

(! symbolPrecedesIfPositive ? currencySymbol : "");

Console.WriteLine( "The regular expression pattern is:");

Console.WriteLine(" " + pattern);

// Get text that matches regular expression pattern.

MatchCollection matches = Regex.Matches(input, pattern,

RegexOptions.IgnorePatternWhitespace);

Console.WriteLine("Found {0} matches.", matches.Count);

// Get numeric string, convert it to a value, and add it to List object.

List<decimal> expenses = new List<Decimal>();

foreach (Match match in matches)

expenses.Add(Decimal.Parse(match.Groups[1].Value));

// Determine whether total is present and if present, whether it is correct.

decimal total = 0;

foreach (decimal value in expenses)

total += value;

if (total / 2 == expenses[expenses.Count - 1])

Console.WriteLine("The expenses total {0:C2}.", expenses[expenses.Count - 1]);

else

Console.WriteLine("The expenses total {0:C2}.", total);

}

}

// The example displays the following output:

// The regular expression pattern is:

// \$\s\*[-+]?([0-9]{0,3}(,[0-9]{3})\*(\.[0-9]+)?)

// Found 6 matches.

// The expenses total $81.58.

На компьютере с региональными параметрами "English - United States (en-US)" пример динамически создает регулярное выражение \$\s\*[-+]?([0-9]{0,3}(,[0-9]{3})\*(\.[0-9]+)?). Шаблон регулярного выражения интерпретируется следующим образом:

Развернуть таблицу

| Расписание | Интерпретация |
| --- | --- |
| \$ | Выполняется поиск одного вхождения символа доллара ($) во входной строке. Строка шаблона регулярного выражения содержит обратную косую черту, что говорит о том, что символ доллара интерпретируется буквально, а не как привязка регулярного выражения. Символ $ только указывает, что обработчик регулярных выражений должен попытаться начать его совпадение в конце строки. Чтобы убедиться, что символ валюты текущего языка и региональных параметров не интерпретируется как символ регулярного выражения, в примере вызывается [Regex.Escape](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex.escape) метод для экранирования символа. |
| \s\* | Поиск нуля или нескольких вхождений пробела. |
| [-+]? | Найдите ноль или одно вхождение положительного или отрицательного знака. |
| ([0-9]{0,3}(,[0-9]{3})\*(\.[0-9]+)?) | Внешние скобки определяют это выражение как записывающую группу или подэкспрессию. Если найдено соответствие, сведения об этой части строки можно получить из второго объекта [Group](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.group) в объекте [GroupCollection](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.groupcollection), который возвращается свойством [Match.Groups](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.match.groups). Первый элемент в коллекции представляет все совпадение. |
| [0-9]{0,3} | Поиск 0-3 вхождений десятичных цифр (0-9). |
| (,[0-9]{3})\* | Поиск нуля или нескольких вхождений разделителя группы, за которыми следуют три десятичные цифры. |
| \. | Поиск одного вхождения десятичного разделителя. |
| [0-9]+ | Поиск одной или нескольких десятичных цифр. |
| (\.[0-9]+)? | Поиск нуля или одного вхождения десятичного разделителя, за которым следует по крайней мере одна десятичная цифра. |

Если каждый подпаттерн найден в входной строке, совпадение завершается успешно, а [Match](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.match) объект, содержащий сведения о совпадении, добавляется в [MatchCollection](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.matchcollection) объект.

Связанные статьи

Развернуть таблицу

| Заголовок | Description |
| --- | --- |
| [Элементы языка регулярных выражений — краткий справочник](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expression-language-quick-reference) | Сведения о наборе символов, операторов и конструкций, которые можно использовать для определения регулярных выражений. |
| [Объектная модель регулярных выражений](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/the-regular-expression-object-model) | Сведения об использовании классов регулярных выражений и примеры кода. |
| [Подробные сведения о поведении регулярных выражений](https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/details-of-regular-expression-behavior) | Сведения о возможностях и поведении регулярных выражений платформы .NET. |
| [Использование регулярных выражений в Visual Studio](https://learn.microsoft.com/ru-ru/visualstudio/ide/using-regular-expressions-in-visual-studio) |  |

Справочные материалы

[System.Text.RegularExpressions](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions)

[System.Text.RegularExpressions.Regex](https://learn.microsoft.com/ru-ru/dotnet/api/system.text.regularexpressions.regex)

[Краткий справочник по регулярным выражениям (скачать в формате Word)](https://download.microsoft.com/download/D/2/4/D240EBF6-A9BA-4E4F-A63F-AEB6DA0B921C/Regular%20expressions%20quick%20reference.docx)

[Регулярные выражения — краткий справочник (загрузить в формате PDF)](https://download.microsoft.com/download/D/2/4/D240EBF6-A9BA-4E4F-A63F-AEB6DA0B921C/Regular%20expressions%20quick%20reference.pdf)