**Паттерн модель формального регулярного языка**

Паттерн язык регулярных выражений — это модель узкоспециализированного языка программирования, встроенного в C#,.NET [1], Python [2] и другие языки программирования. Паттерны регулярных выражений определяются специальным синтаксисом или языком, который совместим с регулярными выражениями Perl 5 [3] и добавляет некоторые дополнительные функции. Паттерн язык относительно невелик и ограничен, поэтому не все возможные задачи обработки строк могут быть выполнены с использованием регулярных выражений.

Модель предоставляют мощный, гибкий и эффективный метод обработки текста. Нотация сопоставления текста с паттерном позволяет быстро анализировать большие объемы текста, чтобы:

* Найти определенные паттерны символов.
* Проверить текст, чтобы убедиться, что он соответствует предварительно определенному паттерну (например, адресу электронной почты).
* Извлечь, редактировать, заменить или удалить текстовые подстроки.
* Добавить извлеченные строки в коллекцию, чтобы сгенерировать отчет.

**Примеры регулярных выражений**

Поддерживается синтаксис немного отличный от PCRE (Perl Compatible Regular Expressions) [3]. Регулярные выражения по умолчанию чувствительны к регистру.

Следующие примеры иллюстрируют использование и построение простых регулярных выражений. Каждый пример включает тип текста для сопоставления, одно или несколько регулярных выражений, соответствующих этому тексту, и примечания, поясняющие использование **специальных** символов и форматирование.

1. Совпадение только с точной фразой
2. Сопоставьте слово или фразу в списке
3. Сопоставьте слова с разными вариантами написания или специальными символами
4. Совпадение с любым адресом электронной почты из определенного домена
5. Совпадение с любым IP-адресом в диапазоне
6. Соответствие буквенно-цифровому формату
7. Действительный номер мобильного телефона..
8. Проверка пользовательского ввода
9. Разбивает строку на основе паттерна

**1. Совпадение только с точной фразой**

| пример | Сопоставление фразы "stocks tips" |
| --- | --- |
| 1 pattern | **(\W|^)stock\stips(\W|$)** |
| 2 pattern | **(\W|^)stock\s{0,3}tips(\W|$)** |
| 3 pattern | **(\W|^)stock\s{0,3}tip(s){0,1}(\W|$)** |
| Примечания | * \W соответствует любому символу, который не является буквой, цифрой или символом подчеркивания. Это предотвращает регулярное выражение из совпадающих символов до или после фразы. * В примере 2 \s соответствует символу пробела, а {0,3} указывает, что от 0 до 3 * между словами stock и tip могут быть пробелы. * ^ соответствует звездочке новой строки. Позволяет регулярному выражению соответствовать фразе, если она появляется в начале строки без символов перед ним. * $ соответствует концу строки. Позволяет регулярному выражению соответствовать фразе, если она появляется в конце строки без символов после него. * В примере 3 (s) соответствует букве s, а {0,1} указывает, что буква может встречаются 0 или 1 раз после слова tip. Следовательно, регулярное выражение соответствует stock tip и stock tips. * Кроме того, вы можете использовать символ ? вместо {0,1} |
|  | Создайте шаблон для слова, которое начинается с буквы «М». |
| pattern | **string** pattern = @"\b[M]\w+";  **string** authors = "Mahesh Chand, Raj Kumar, Mike Gold, Allen O'Neill, Ma";  Regex rg = **new** Regex(pattern)  // Получить все совпадения  MatchCollection matchedAuthors = rg.Matches(authors);  // Распечатать всех подходящих авторов  **for** (**int** count = 0; count < matchedAuthors.Count;count++)  Console.WriteLine(matchedAuthors[count].Value); |
|  |  |
| pattern |  |

**2. Сопоставьте слово или фразу в списке**

| пример | **Сопоставьте любое слово или фразу из следующего списка:** |
| --- | --- |
|  | * aloney * darn * drat * fooey * gosh darnit * heck |
| pattern | **(?i)(\W|^)(baloney|darn|drat|fooey|gosh\sdarnit|heck)(\W|$)** |
| Примеча-ния | • (...) группирует все слова таким образом, что класс символов \W применяется ко всем словаv в скобках.  • (?i) делает соответствие содержимого нечувствительным к регистру.  • \W соответствует любому символу, который не является буквой, цифрой или символом подчеркивания. Он не позволяет регулярному выражению сопоставлять символы до или после слов или фраз в списке.  • ^ соответствует звездочке новой строки. Позволяет регулярному выражению сопоставлять слово, если оно появляется в начале строки без символов перед ним.  • $ соответствует концу строки. Позволяет регулярному выражению сопоставлять слово, если оно находится в конце строки, без символов после него  • | указывает «или», поэтому регулярное выражение соответствует любому слову в списке.  • \s соответствует пробелу. Используйте этот символ для разделения слов во фразе. |
|  |  |

**3.Сопоставьте слова с разными вариантами написания**

**или специальными символами**

| пример | Сопоставьте слово "гравий" и некоторые запутывания, которые используют спамеры, например: |
| --- | --- |
|  | **gra@via**  **grav1ia**  **gr@av1ia**  **gra@via** |
| pattern | v[i!1][g@]ra[vi@a] |
|  | • \W не включен, так что другие символы могут стоять до или после любого из вариантов "гравий". Например, регулярное выражение по-прежнему соответствует "гравий" в следующем тексте:  **gravia!! or \*\*\*gravia\*\*\*** |
|  |  |

**4.Совпадение с любым адресом электронной почты из**

**определенного домена. Проверка нескольких писем**

| пример | Совпадение с любым адресом электронной почты из доменов yahoo.com, hotmail.com и gmail.com |
| --- | --- |
| 1 pattern | **(\W|^)[\w.\-]{0,25}@(yahoo|hotmail|gmail)\.com(\W|$)** |
| Примеча-ния | • \W соответствует любому символу, который не является буквой, цифрой или символом подчеркивания. Это предотвращает регулярное выражение из совпадающих символов до или после адреса электронной почты.  • ^ соответствует звездочке новой строки. Позволяет регулярному выражению сопоставляться с адресом, если он появляется в начале строки без символов перед ним.  • $ соответствует концу строки. Позволяет регулярному выражению сопоставляться с адресом, если он находится в конце строки без символов после него.  • [\w.\-] соответствует любому символу слова (a-z, A-Z, 0-9 или знак подчеркивания), точке или дефису. Это наиболее часто используемые допустимые символы в первой части адреса электронной почты. Символ \- (обозначающий дефис) должен стоять последним в списке символов в квадратных скобках.  • Знак \ перед тире и точкой «экранирует» эти символы, т. е. указывает, что тире и точка сами по себе не являются специальными символами регулярных выражений. Нет необходимости экранировать точку в квадратных скобках.  • {0,25} указывает, что от 0 до 25 символов в предыдущем наборе символов могут стоять перед символом @. Настройка электронной почты соответствия содержания поддерживает сопоставление до 25 символов для каждого набора символов в регулярном выражении.  • Форматирование (...) группирует домены, а | символ, который их разделяет, указывает на «или». |
|  | **Проверка нескольких писем** |
| 2 pattern | **^((\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*)\s\*[;]{0,1}\s\*)+$** |
|  | **разделение писем с помощью разделителя ';'** |
| 3 pattern | **^((\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*)\s\*[,]{0,1}\s\*)+$** |
|  | **Если вы хотите использовать разделитель ',' используйте этот** |
| 4 pattern | **^((\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*)\s\*[;,.]{0,1}\s\*)+$** |
|  | **и если вы хотите использовать оба разделителя ',' и ';'** |

**5.Совпадение с любым IP-адресом в диапазоне**

| пример | Соответствует любому IP-адресу в диапазоне от 192.168.1.0 до 192.168.1.255. |
| --- | --- |
| 1 pattern | **192\.168\.1\.** |
| 2 pattern | **192\.168\.1\.\d{1,3}** |
| Примеча-ния | • Знак \ перед каждой точкой «убегает» от точки, т. е. указывает, что период сам по себе не является специальным символом регулярного выражения.  • В примере 1 за последней точкой не следует никаких символов, поэтому регулярное выражение соответствует любому IP-адресу, начинающемуся с 192.168.1., независимо от следующего за ним числа.  • В примере 2 \d соответствует любой цифре от 0 до 9 после последней точки, а {1,3} указывает, что цифры от 1 до 3 могут появляться после этой последней точки. В этом случае регулярное выражение соответствует любому полному IP-адресу, начинающемуся с 192.168.1. Это регулярное выражение также соответствует недопустимым IP-адресам, таким как 192.168.1.999. |
|  |  |

**6.Соответствие буквенно-цифровому формату. Замена нескольких пробелов, специальные символы. Разделить строку на буквенный символ.**

| пример | Сопоставьте номера заказов на покупку. Этот номер имеет различные возможные форматы, например: |
| --- | --- |
|  | * PO nn-nnnnn * PO nn-nnnn * PO# nn nnnn * PO#nn-nnnn * PO nnnnnn |
| 1 pattern | **(\W|^)po[#\-]{0,1}\s{0,1}\d{2}[\s-]{0,1}\d{4}(\W|$)** |
|  | Замена нескольких пробелов |
| 2 pattern | **string** badString = "a string with ton of white space.";  **string** CleanedString = Regex.Replace(badString,"\\s+", " ");  Console.WriteLine($"Cleaned String: {CleanedString}"); |
|  | Заменить специальные символы |
| 3 pattern | Regex.Replace(your String, @"[^0-9a-zA-Z]+", "") |
|  | Этот код удаляет все специальные символы, но если вы не хотите удалять некоторые специальные символы, например. запятая","и двоеточие":" то:  Regex.Replace(Your String, @"[^0-9a-zA-Z:,]+", "") |
|  | Разделить строку на буквенный символ |
| 4 pattern | **string** azpattern = "[a-z]+";  **string** str = "Asd2323b0900c1234Def5678Ghi9012Jklm";  **string**[] result = Regex.Split(str, azpattern,  RegexOptions.IgnoreCase, TimeSpan.FromMilliseconds(500));  **for** (**int** i = 0; i < result.Length; i++){  Console.Write("'{0}'", result[i]);  **if** (i < result.Length - 1)  Console.Write(", ");  } |

**7. Действительный номер мобильного телефона..**

| пример | Сопоставьте номер **мобильного телефона**, например: |
| --- | --- |
|  | **string**[] str = { "+91-9678967101", "9678967101", "+91- 9678-967101", "+91-96789-67101", "+919678967101"}; |
| 1 pattern | @"^\+?\d{0,2}\-?\d{4,5}\-?\d{5,6}" |
| Примеча-ния | **using** System; **using** System.Collections.Generic; **using** System.Linq; **using** System.Text; **using** System.Text.RegularExpressions;  **namespace** RegularExpression1 { **class** Program {  **static void** Main(**string**[] args){  Regex r = **new** Regex(@"^\+?\d{0,2}\-?\d{4,5}\-?\d{5,6}");  **string**[] str = { "+91-9678967101", "9678967101", "+91-  9678-967101", "+91-96789-67101", "+919678967101" }; //Введите строки для совпадения с действительным номером // мобильного телефона.  **foreach** (**string** s **in** str) {  Console.WriteLine(" {0} {1} a valid mobile number.", s,  r.IsMatch(s) ? "is" : "is not"); //The IsMatch method is used to validate a string or //to ensure that a string conforms to a particular part  } // **foreach**  } // Main  } // class  } // **namespace** |
| 2 pattern |  |
|  |  |
| 3 pattern |  |

**8. Проверка пользовательского ввода**

| пример | **Пользовательский ввод** |
| --- | --- |
| 1 pattern  2 pattern  3 pattern  4 pattern  5 pattern | **If**(!Regex.Match(firstNameTextBox.Text,"^[A-Z][a-zAZ]\*$").Success) {...}  **if** (!Regex.Match(cityTextBox.Text, @"^([a-zA-Z]+|[a-zA-Z]+\s[a-zAZ]+)$").Success) {...}  **if** (!Regex.Match(stateTextBox.Text, @"^([a-zA-Z]+|[a-zA-Z]+\s[azA-Z]+)$").Success) {...}  **if** (!Regex.Match(zipCodeTextBox.Text, @"^\d{5}$").Success)  {...}  **if** (!Regex.Match(phoneTextBox.Text, @"^[1-9]\d{2}-[1-9]\d{2}-\d{4}$").Success) {...} |
| Примеча-ния |  |
| 2 pattern |  |
|  |  |
| 3 pattern |  |

**9. Разбивает строку на основе паттерна**

| пример | **Разбить цифры от строк** |
| --- | --- |
|  | **string** Text = "1 One, 2 Two, 3 Three is good."; |
| 1 pattern | @"\D+" |
| Примеча-ния | **string**[] digits = Regex.Split(Text, @"\D+");  **foreach** (**string** value **in** digits) {  **int** number;  **if** (**int**.TryParse(value, **out** number)) {  Console.WriteLine(value);  }  } |
| 2 pattern |  |
|  |  |
| 3 pattern |  |

Регулярные выражения со следующими специальными символами не поддерживаются, так как они могут привести к задержкам в обработке вашей электронной почты:

• \* (звездочка)

• + (знак плюс)

[1] [.NET Regular Expressions | Microsoft Docs](https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions)

[2] [HOWTO по регулярным выражениям | Python 3 (digitology.tech)](https://digitology.tech/docs/python_3/howto/regex.html)

[3] [Perl Compatible Regular Expressions - Wikipedia](https://en.wikipedia.org/wiki/Perl_Compatible_Regular_Expressions)