Обучающий материал по регулярным выражениям

Что такое регулярные выражения?

Регулярные выражения (или regex, от regular expressions) — это мощный инструмент для поиска и обработки текста. Они представляют собой последовательности символов, которые задают шаблон, соответствующий строкам текста. Регулярные выражения используются для выполнения операций поиска, замены, извлечения и валидации данных. Они позволяют эффективно работать с текстом и извлекать или изменять его содержимое.

В языке программирования Python регулярные выражения поддерживаются через модуль re.

В текстовых редакторах (например, Notepad++ и Sublime Text) регулярные выражения используются для поиска и замены текста.

В языке программирования Python регулярные выражения поддерживаются через модуль re.

В текстовых редакторах (например, Notepad++ и Sublime Text) регулярные выражения используются для поиска и замены текста.

Мощность: они могут использоваться для выполнения сложных операций, таких как замена текста, извлечение данных и анализ структуры строки.

Экономия времени: регулярные выражения позволяют автоматизировать процессы обработки текстовых данных, значительно ускоряя выполнение задач.

Портативность: регулярные выражения поддерживаются во многих языках программирования и текстовых редакторах.

Основные области применения регулярных выражений

- Программирование: для поиска и манипуляции текстом (например, поиск чисел, обработка строк).

- Веб-разработка: для валидации данных (например, проверка email-адресов, телефонных номеров), а также для парсинга данных с веб-страниц.

- Текстовые редакторы: для поиска и замены текста по шаблону, извлечения данных.

- Анализ данных: для обработки больших объемов текстовых данных, извлечения информации.

Основы синтаксиса регулярных выражений

Точные символы

Регулярные выражения могут содержать *точные символы*, которые будут искать совпадения с этими символами в строке.

Пример: a — найдет символ "a" в строке.

Квантификаторы

* — 0 или более повторений (например, a* — 0 или более символов "a").

+ — 1 или более повторений (например, a+ — 1 или более символов "a").

{n} — ровно n повторений (например, a{3} — ровно три символа "a").

{n,} — минимум n повторений (например, a{2,} — два и более символа "a").

{n,m} — от n до m повторений (например, a{2,4} — от двух до четырех символов "a").

Символьные классы

\d — цифра (аналог [0-9]).

\w — буквенно-цифровой символ (аналог [a-zA-Z0-9_]).

\s — пробельный символ (аналог [\t\n\r\f\v]).

Примеры:

\d+ — одно или более чисел.

\w+ — одно или более буквенно-цифровых символов.

[ ] – символьный класс. Допустимыми считаются символы внутри скобочек.

[^] – исключающий символьный класс. Недопустимы символы в скобочках.

\n – тот же символ что и в n-ой группе. Например ‘[a-d]\1’ найдет ‘aa’, ‘bb’, ‘cc’, ‘dd’

(?!...) – негативный просмотр вперед. Проверяет, что дальше не будет шаблона «…»

(?<!...) Негативный ретроспективный просмотр. Проверяет, что **раньше не было** шаблона «…»

(?=...) – позитивный просмотр. Проверяет, что дальше будет шаблон «...»

(?<=...) – позитивный ретроспективный просмотр. Проверяет, что раньше был шаблон.

Группировка и альтернация

Группировка используется для объединения нескольких символов в одну сущность. Например: (abc) — группа символов "abc".

Альтернация позволяет искать одно из нескольких возможных значений. Например: a|b — найдет либо "a", либо "b".

Примеры:

(abc|def) — найдет либо "abc", либо "def".

(a|b)+ — найдет одну или более "a" или "b".

Метасимвол	Значение
.	Соответствует любому *только* 1 символу
[ ]	Соответствует одному символу из содержащихся в квадратных скобках
-	Граница последовательности символов в квадратных скобках
[^]	Соответствует одному символу из *не* содержащихся в квадратных скобках
^	Соответствует началу строки
$	Соответствует концу строки
*	Соответствует 0 или более предыдущих элементов
?	Соответствует 0 или одному предыдущему элементу
+	Означает, что предшествующий символ присутствует и может повторяться несколько раз
|	Операция ИЛИ. Соответствует либо выражению до операции, либо выражению после нее
\	Символ литерализации. Позволяет использовать для сопоставления любой метасимвол как литерал (экранирование специальных символов)
( )	Группирует символы в подстроки
{ }	Количество повторений предыдущего символа

Практическое использование регулярных выражений

Регулярные выражения часто используются в программировании для поиска, замены и извлечения данных. Рассмотрим пример на Python и PascalABC.NET:

```python
import re

pattern = r'\d+'  # шаблон для поиска всех чисел
text = "У меня есть 123 яблока и 456 апельсинов."
matches = re.findall(pattern, text)
print(matches)  # Output: ['123', '456']
```

```
##
var pattern := '\d+';
var s := 'У меня есть 123 яблока и 456 апельсинов';

var matches := Regex.Matches(s, pattern);
print(matches)
```

В этом примере регулярное выражение \d+ находит все последовательности цифр в строке.

Применение в текстовых редакторах

Многие текстовые редакторы, такие как Notepad++ и Sublime Text, поддерживают регулярные выражения для поиска и замены текста. Например, в Notepad++ можно использовать регулярное выражение для поиска всех email-адресов:

[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}

Это регулярное выражение находит все email-адреса в тексте.

Использование в валидации данных

Регулярные выражения также часто используются для валидации данных, введенных пользователем. Например, для проверки правильности ввода email-адреса можно использовать следующее регулярное выражение:

[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}

Это регулярное выражение проверяет, что введенный текст является корректным email-адресом.

Регулярные выражения в PascalABC.NET

В языке PascalABC.NET имеются следующие основные методы для работы с классом Regex:

- *Regex.IsMatch* – возвращает Boolean (true/false) в зависимости от того найдено ли выражение в строке

```
##
var s := 'Маша мыла раму';

var b := Regex.IsMatch(s, '\d+');

println(b); // output: False
```

Объяснение: регулярное выражение \d+ находит какую-нибудь последовательность чисел. Чисел в строке нет, значит результат – ложь.

- Regex.Match – возвращает первое найденное выражение

```
##
var s := 'Маша мыла раму';

var b := Regex.Match(s, 'м[б-я^у]');

println(b); // output: мы
```

Объяснение: регулярное выражение «м[б-я^у]» ищет сочетание из двух букв, из которых первая – *м*, а вторая – какая то из букв промежутка от *б* до *я*, кроме у (исключающий класс).

- Regex.Matches – возвращает все найденные выражения

```
##
var s := 'Мама мыла маму';

var b := Regex.Matches(s, '(м[а-я])+', RegexOptions.IgnoreCase);

println(b); // output: [мама, мы, маму]
```

Объяснение: это регулярное выражение ищет последовательности из пар букв «м» с какой то любой другой буквой от «а» до «я».

- Regex.Split – разбивает строку на фрагменты; разделители фрагментов определяются регулярным выражением

```
##
var s := 'sk123ibidi do11p dop ye033s';

var b := Regex.Split(s, '\d+', RegexOptions.IgnoreCase);

println(b); // output: [sk,ibidi do,p dop ye,s]

```

Объяснение: это регулярное выражение находит последовательности из цифр. Таким образом, эти последовательности служат разделителями.

- Regex.Replace – заменяет найденные выражения

```
##
var s := 'sk123ibidi do11p dop ye033s';

var b := Regex.Replace(s, '\d+', '', RegexOptions.IgnoreCase);

println(b); // output: skibidi dop dop yes

```

Объяснение: это регулярное выражение находит последовательности из цифр. Эти цифры затем заменяются на ‘’ – то есть отсутствие символа

Решение задач ЕГЭ с помощью RegEX

Некоторые задачи из ЕГЭ можно очень удобно и быстро решать регулярными выражениями. Ниже будут приведены 18 задач типа 24. Обычно идеи в таких задачах не слишком замудренные и нужно лишь привести данные в удобный вид и применить соответствующее вопросу задачи регулярное выражение.

Все семплы данных можно скачать по этой ссылке: https://drive.google.com

В конце документа вы сможете найти их решения на PascalABC.NET.

№1 В текстовом файле **k7a-2.txt** находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите длину самой длинной подцепочки, состоящей из символов A, C, D (в произвольном порядке).

Разбор решения

- Нам подходят из всех символов только A C и D. Значит нам нужно обозначить их в символьном классе.

- наше регулярное выражение выглядит так '[ACD]+'. Теперь просто найдем все нужные нам цепочки и выберем самую длинную.

```
var pattern := '[ACD]+';

var ms := Regex.Matches(s, pattern);

var maxLen := 0;

for var i:=0 to ms.Count-1 do
begin
  if(ms[i].Length > maxLen) then
    maxLen := ms[i].Length;
end;

println(maxLen);
```

№2 В текстовом файле **k7a-6.txt** находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите длину самой длинной подцепочки, не содержащей гласных букв.

№3 В текстовом файле **k7c-6.txt** находится цепочка из символов латинского алфавита A, B, C, D, E, F. Найдите количество цепочек длины 3, в которых символы не совпадают.

Разбор решения

- Первым делом необходимо составить регулярное выражение. Первый символ должен быть от A до F, а следующий за ним не должен быть таким же как данный символ. Что здесь удобно использовать?

- Негативный просмотр вперед! Теперь выражение выглядит так «([A-F])(?\1)». Теперь нужно продолжить его составление для 2 оставшихся символов тройки.
- «([A-F])(?!\1)([A-F])(?!\1|\2)([A-F])» - искомое выражение.

```
4  var pattern := '([A-F])(?!\1)([A-F])(?!\1|\2)([A-F])';
5
6  var ms := Regex.Matches(s, pattern);
7
8  println(ms);
9
10
11
```

№4 В текстовом файле **k8-18.txt** находится цепочка из символов, в которую могут входить заглавные буквы латинского алфавита A…Z и десятичные цифры. Найдите длину самой длинной подцепочки, состоящей из одинаковых символов. Если в файле несколько подходящих цепочек одинаковой длины, нужно взять первую из них. Выведите сначала символ, из которого строится эта подцепочка, а затем через пробел – длину этой подцепочки.

№5 Текстовый файл **24-1.txt** состоит не более чем из 10^6 символов. Определите максимальное нечётное число, записанное в этом файле. Под числом подразумевается последовательность цифр, ограниченная другими символами (не цифрами).

Важно понимать, что не все задачи типа 24 можно решить регулярными выражениями. Рассмотрим следующую задачу:

Текстовый файл **24.txt** содержит последовательность из строчных и заглавных букв английского алфавита и цифр, всего не более 10^6 символов. Определите длину наибольшей возрастающей подпоследовательности. **Возрастающей подпоследовательностью** будем называть непрерывную последовательность символов, расположенных в порядке увеличения их номера в кодовой таблице символов ASCII.

Регулярки позволяют выбирать строки подходящие шаблону, однако этот *шаблон должен быть фиксированным*. В данном случае с возрастающей последовательностью мы не можем придумать какой-либо паттерн.

Эту задачу легче решить обыкновенным перебором строки. Вот пример решения на Python:

```
# Читаем файл
with open("24.txt", "r") as file:
    content = file.read().strip()

max_length = 1
current_length = 1

# Проходим по строке
for i in range(1, len(content)):
    if ord(content[i]) > ord(content[i - 1]):  # Проверяем по ASCII-коду
        current_length += 1  # Продолжаем последовательность
        max_length = max(max_length, current_length)  # Обновляем максимум
    else:
        current_length = 1  # Начинаем новую последовательность

# Выводим результат
print(max_length)
```

№6 Текстовый файл **24-J1.txt** состоит не более чем из 10^6 кириллических символов К, О, Т. Определите максимальное количество подряд идущих комбинаций КОТ.

№7 Текстовый файл **24-J4.txt** состоит не более чем из 10^6 символов J, O, B, S. Сколько раз встречаются комбинации «BOSS» при этом до и после этого слова нет символа «J». Например, комбинации «JBOSS», «BOSSJ» и «JBOSSJ» не должны учитываться.

№8 Текстовый файл **24-j7.txt** состоит не более чем из 10^6 десятичных цифр. Найдите максимальную длину последовательности, которая состоит из цифр одинаковой четности. Например, в последовательности 1533244622185452354, 5 последовательностей с нечетными цифрами – 1533, 1, 5, 5, 35 – и 5 с четными – 244622, 8, 4, 2, 4. Следовательно, искомая

последовательность – 244622. В качестве ответа укажите максимальную длину найденной последовательности.

№9 Текстовый файл **24-j9.txt** состоит не более чем из 10^6 символов английского алфавита. Определите количество палиндромов (последовательностей, которые читаются в обе стороны одинаково) длиной 5 символов.

№10 Текстовый файл **24-153.txt** содержит строку из заглавных букв A, B, C, D, E, F, всего не более 10^6 символов. D-подстроками назовём последовательности идущих подряд символов D, ограниченные иными символами и/или границами строки. Определите минимальную длину D-подстроки.

№11 Текстовый файл **24-153.txt** содержит строку из заглавных букв A, B, C, D, E, F, всего не более 10^6 символов. AF-подстроками назовём непустые последовательности идущих подряд символов A, B, C, D, E, F, ограниченные в начале символом A, а в конце символом F (граничные символы входят в подстроку). Определите количество AF-подстрок длиной от 7 до 10 символов.

№12 Текстовый файл **`24-180.txt`** содержит строку из десятичных цифр, всего не более чем из 10^6 символов. Файл образовался в результате последовательной записи «таймкодов» некоторых событий в формате HHMM (часы и минуты слитно по две цифры, т.е. всего 4 цифры на «таймкод», от 0000 до 2359) и прочих случайных данных. Найдите максимально возможное количество подряд идущих «таймкодов» между фрагментами случайной информации. Например, в строке 4212231135414447 можно выделить таймкоды тремя способами: 4[2122]3[1135]4[1444]7, 42[1223,1135]4[1444]7 или 421[2231,1354,1444]7. В последнем случае получилось наибольшее количество таймкодов подряд (3), это число и нужно ввести в ответе.

№13 Текстовый файл **24-181.txt** содержит строку из заглавных латинских букв и точек, всего не более чем из 10^6 символов. Определите максимальное количество идущих подряд символов, среди которых нет букв Y, а количество точек не превышает 5.

№14 Текстовый файл **24-181.txt** содержит строку из заглавных латинских букв и точек, всего не более чем из 10^6 символов. Определите максимальное количество идущих подряд символов, среди которых нет точек, а количество гласных (букв A, E, I, O, U, Y) не превышает 7.

№15 Текстовый файл **24-191.txt** содержит строку из заглавных латинских букв, всего не более чем из 10^6 символов. Определите количество подстрок длиной не более 12 символов, которые начинаются и заканчиваются буквой A и не содержат других букв A (кроме первой и последней) и букв B.

№16 Текстовый файл **24-191.txt** содержит строку из заглавных латинских букв, всего не более чем из 10^6 символов. Определите количество подстрок длиной не более 15 символов, которые начинаются буквой A, содержат букву F, заканчиваются буквой B и не содержат других букв A и B, кроме первой и последней.

№17 Текстовый файл **24-197.txt** содержит строку из заглавных латинских букв X, Y и Z, всего не более чем из 10^6 символов. Определите максимальное количество идущих подряд троек символов ZXY или ZYX.

№18 Текстовый файл **24-204.txt** содержит строку из заглавных латинских букв A, B и C, всего не более чем из 10^6 символов. Найдите максимальное количество подряд идущих пар символов AA или CC. Искомая подстрока может включать только пары AA, только пары CC или содержать одновременно как пары AA, так и пары CC.