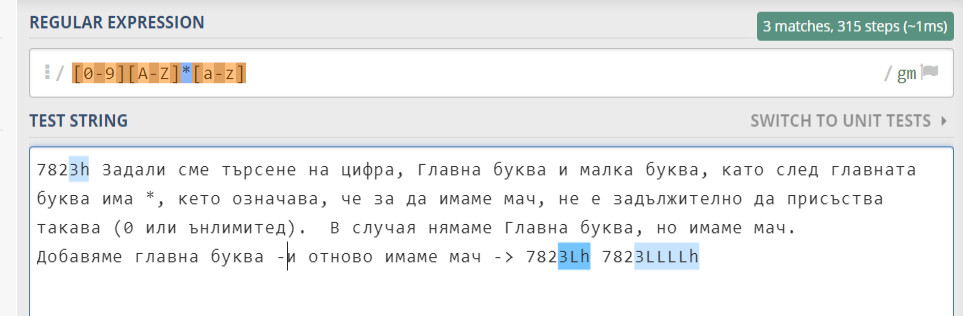
**Regular expressions a.k.a Regex**

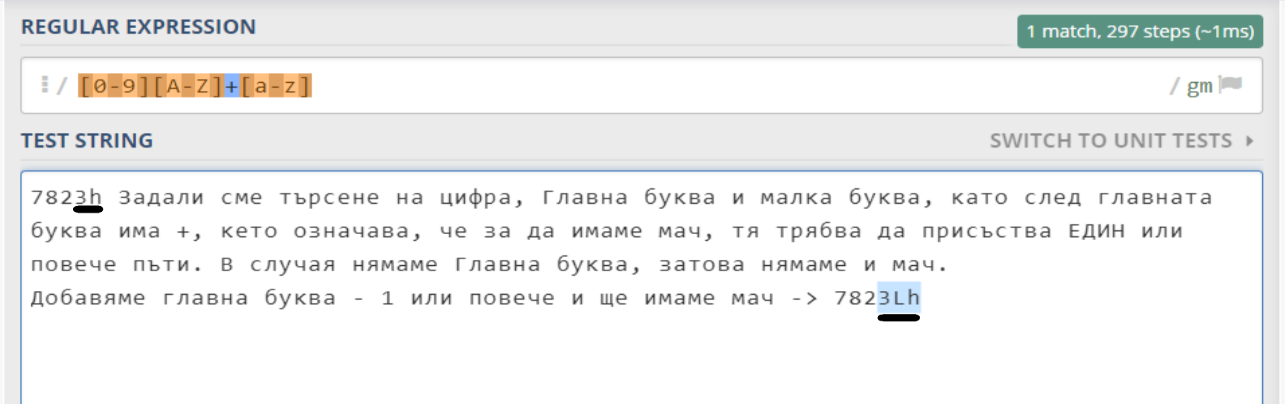
* Most common flags :  
  - i : case insensitive  
  - g : global (doesn't stop after first match)  
  - m : multi-line
* Most common anchors :  
  - ^ : Start of string  
  - $ : End of string  
  - \A : Start of string (not affected by multi-line content)  
  - \Z : End of string (not affected by multi-line content)
* Most common quantifiers :  
  - {n} : Exactly n times  
  - {n,} : At least n times  
  - {m,n} : Between m and n times  
  - ? : Zero or one time  
  - + : One or more times  
  - \* : Zero, one or more times
* Most common meta sequences :  
  - . : Any character but \n and \r  
  - \w | \W : Any word character | Any non-word character  
  - \d | \D : Any digit character | Any non-digit character  
  - \s | \S : Any whitespace character | Any non-whitespace character
* Character set :  
  — [abc] : Will match either a, b or c  
  - [1-9] : Will match any digit from 1 to 9  
  - [a-zA-Z] : Will match any letter
* Match any character but :  
  - [^abc] : Matches anything but a, b or c
* Escape a character :  
  - \character (example : escaping + => \+)
* Refer to a group (also used for capturing groups, look further):  
  — (group of characters) (example : /(he)+/ will match 'hehehe'
* One group or another :  
  - | : /^h((ello)|(ola))$/ will match both 'hello' and 'hola'

**Quantifiers**:

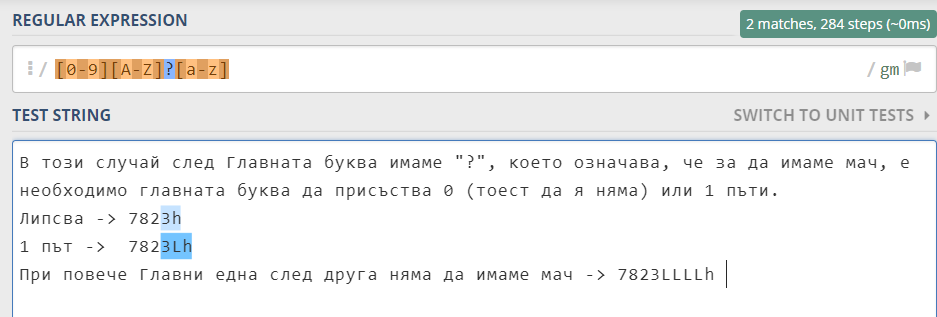
**\***  - matches the previous element zero or more times



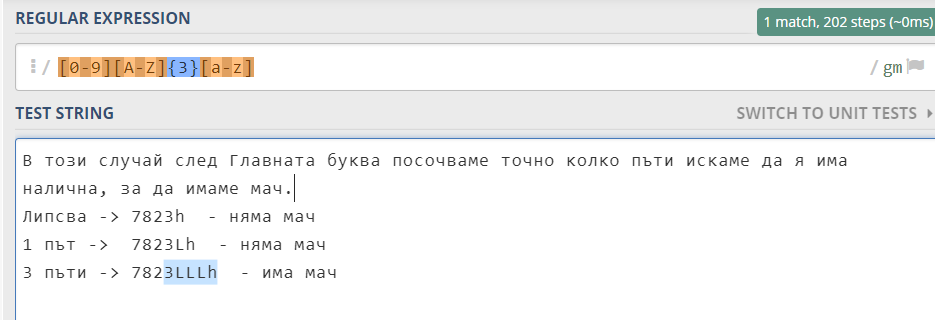
**+** - matches the previous element one or more times



**?** - matches the previous element zero or one times



**{3}** - matches the previous element exactly 3 times

****

**{3,5}** - matches the previous element between 3 and 5 times

**{3, }** – minimum 3 elements for match

**Greedy** – лаком, алчен

- ^ : Start of string края на реда!  
- $ : End of string края на реда!

\b – beginnnig or end of string -> \b word \b

**Predefined classes:**

* \w - matches any word character (a-z, A-Z, 0-9, \_)
* \W - matches any non-word character (the opposite of \w)
* \s - matches any white-space character
* \S - matches any non-white-space character (opposite of \s)
* \d - matches any decimal digit (0-9)
* \D - matches any non-decimal character (the opposite of \d)

(\d{2})-([A-Z][a-z]{2})-(\d{4}) - Мачване на дата във формат 25-06-1991

**[A-Za-z] – Или малка, или главна буква**

**Grouping Constructs:**

* **(subexpression) - captures the matched subexpression as numbered group – групиране**

\d{2}-(\w{3})-\d{4} – поставяне в скоби за обособяване на подгрупа

* **(?:subexpression) - defines a non-capturing group – изключване на подгрупа, която не ни е нужна ?: -**

Например мачнали сме Pesho Peshov, но ни е необходима само фамилията му.

* **(?<name>subexpression)** - defines a named capturing group – наименоване на подгрупа
* **|** - boolean or (text1 | text2)

(\d{2} | \d{4}) – така няма да работи, тъй като ще открие 2 елемента и ще спре мачването, затова 🡪 (\d{4} | \d{2}) !!!

. мачва всичко, затова трябва да я искейпнем с \.

Ако не искам да търси greedy, трябва да му сложа една ? въпросителна

Иначе ще продължи да търси до края на реда и ще хване затварящата скоба на някой друг таг.

Например в задачата с търсенето на тагове <[a-z]+.\*?> Точката ще мачне абсолютно всичко, затова слагаме \* и след това въпросителна

/ - този знак трябва да се изскейпне – \/ peece!!!

\1 – търси същото като в група 1

using System.Text.RegularExpressions;

var regex = new Regex(“some expresion…”)

(?<=\s) – Проверяваме само дали има начало на стринг или празно място.

Look ahead and look Behinde

**\k<nameOfTheGroup> - repeat the same group**