

Διάλεξη 1η

RegEx

Regular Expressions

Δημητριάδης Βασίλης

vdimitriadis@uth.gr

Αργυρίου Ιωάννης

iargyriou@uth.gr

Τι είναι τα Regular Expressions;

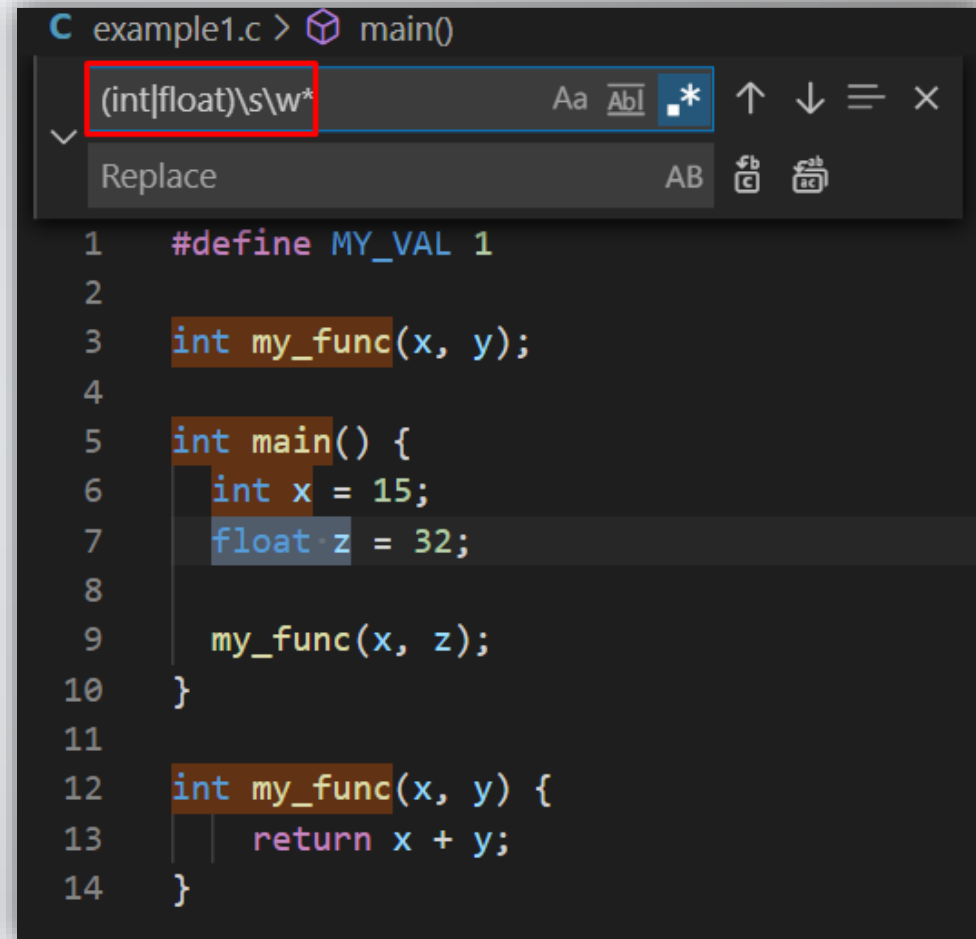
- Ακολουθία χαρακτήρων
- που ψάχνει **μοτίβα** μέσα σε κείμενα



Πότε χρησιμοποιούμε Regular Expressions;

- Search
- Search and Replace
- Web Scraping
- Parsing

Συγκεκριμένα εμείς θα κάνουμε
Parsing



The screenshot shows a code editor window titled 'example1.c > main()'. A search bar at the top contains the regular expression '(int|float)\s\w*' which is highlighted with a red rectangle. Below the search bar is a 'Replace' field. The code below is a C program with the following lines:

```
1  #define MY_VAL 1
2
3  int my_func(x, y);
4
5  int main() {
6      int x = 15;
7      float z = 32;
8
9      my_func(x, z);
10 }
11
12 int my_func(x, y) {
13     return x + y;
14 }
```

Πόσο **καλές γνώσεις** θα χρειαστούμε;

- **Βασικές γνώσεις** RegEx
- Το πολύ **4-5 σύμβολα**
- Flex και Bison δεν υποστηρίζουν όλο το υποσύνολο του RegEx
- Ωστόσο, εμείς θα δείξουμε ένα σημαντικό υποσύνολο του RegEx

Γιατί;

- Για να σας **ταλαιπωρήσουμε**
- Learn once. **Use Anywhere**
- Γλώσσες που το υποστηρίζουν:

C, C++, Python, Ruby, PHP, Java
VB/C#/JavaScript και πολλές ακόμη!

(και το Flex φυσικά'... ˘˘(˘˘)˘˘)

Εργαλεία που θα χρησιμοποιήσουμε:

- Testing: <https://regexr.com/>
- Cheat sheet: <https://regexcheatsheet.com/>
- Cookbook:
 - <https://medium.com/factory-mind/regex-cookbook-most-wanted-regex-aa721558c3c1>
 - <https://www.regexplanet.com/cookbook/index.html>
- Tutorial: https://regexone.com/lesson/introduction_abcs

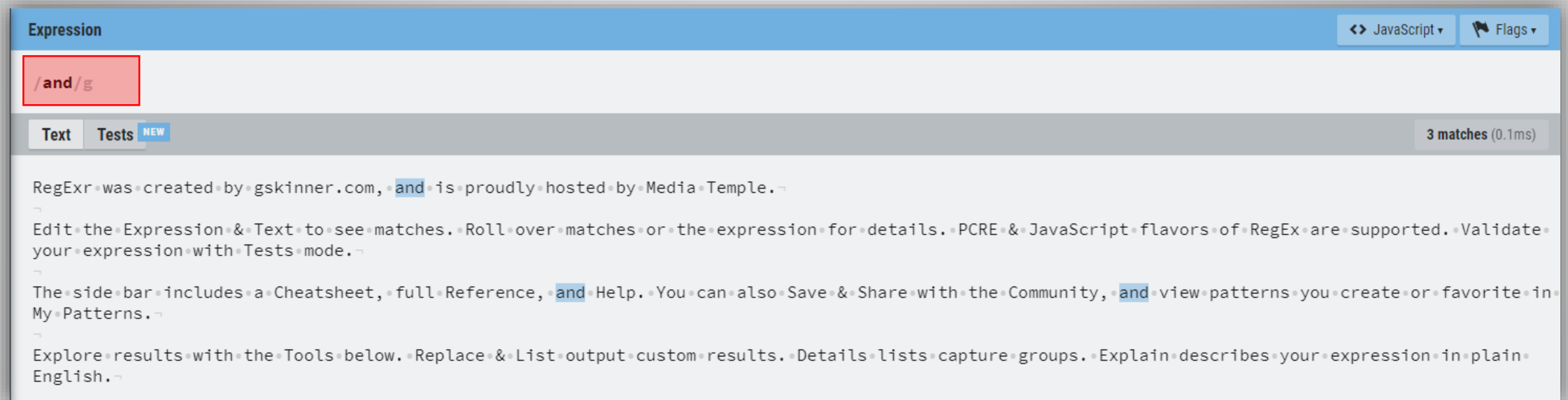
Απλή, κλασσική αναζήτηση

supercomputer

The "Answer to the Ultimate Question of Life, the Universe, and Everything", calculated by an enormous supercomputer named "Deep Thought " over a period of 7.5 million years is 42.

Απλή, κλασσική αναζήτηση

Δοκιμάστε το στο <http://regexr.com>



Anchors – Αρχή και τέλος

^ – Συμβολίζει την αρχή ενός string

`^This` **This** is the end

\$ – Συμβολίζει το τέλος ενός string

`end$` This is the **end**

^...\$ – Ακριβές string-match

`^This is the end end$` **This is the end**

Διαφορές από το να το γράφαμε με κλασσική αναζήτηση;

anchors – Αρχή και τέλος

int

^int

^int\$

```
int main() {  
  int x = 15;  
  float z = 32;  
  
  my_func(x, z);  
}
```

Πιστεύετε θα χρησιμοποιήσουμε Anchors?

Quantifiers

***** – 0...infinity

`abc*`

ab
abc
abcccc

+ – 1...infinity

`abc+`

ab
abc
abcccc

? – 0...1

`abc?`

ab
abc
abcccc

{start, end}

`abc{2,4}`

ab
abc
abcccc

abcc
abccc
abccccccc

{start, }

`abc{2,}`

ab
abc
abcccc

abcc
abccc
abccccccc

{start, end}

`abc{2}`

ab
abc
abcccc

abcc
abccc
abccccccc

Ποια λέξη ψάχνει αυτό;

`Vasi+lis?`

Sequences

Αν όμως θέλουμε **συνδυασμό χαρακτήρων**;

$a(bc)^*$

a
abc
abcbc
abcbcbcbc
abcbcbcbcd

$a(bc)\{2,5\}$

a
abc
abcbc
abcbcbcbc
abcbcbcbcbc

babcbc
babcbcbcbcb
abcbcbcd

Πάμε να κάνουμε δοκιμές!

Or Operators

Αν όμως θέλουμε **συνδυασμό χαρακτήρων**;

`a(b|c)`

a
ab
ac
 abc
 acb
abcabcacb

`ab(cd|ef)`

ab
abcd
abef
 abce

`a[bc]`

a
ab
ac
 abc
 acb
abcabcacb

`ab[cdef]`

ab
abcd
abef
abce

Πάμε να κάνουμε δοκιμές!

Bracket Expressions

[abc] – a or b or c

Άμα θέλαμε ένα **μεγάλο or**? Πχ, το **αγγλικό αλφάβητο**;

[a-z] = [abcdefghijklmnopqrstuvwxyz]

Άμα θέλαμε να συμπεριλάβουμε **και τα κεφαλαία**;

[a-zA-Z] = [abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ]

Συνεπώς **τι κάνουν αυτά**;

[a-zA-Z0-9]

[a-zA-Z0-9_]

[a-zA-Z0-9_+^]

Bracket Expressions

[a-zA-Z0-9]

a
?
-
abc
abc?_ac

[a-zA-Z0-9]+

a
?
-
abc
abc?_ac

[a-zA-Z0-9_?]

a
?
-
abc
abc?_ac

[a-zA-Z0-9_?]+

a
?
-
abc
abc?_ac
abc?_ac+abd36_12

Bracket Expressions - Άρνηση

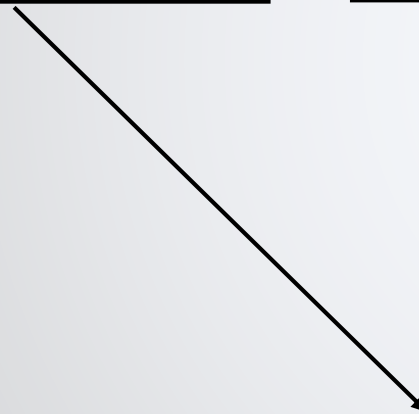
Τι γίνεται άμα θέλαμε όμως το αντίθετο;

Δηλαδή, **να μην περιέχει έναν από αυτούς τους χαρακτήρες;**

[^a-zA-Z0-9]

[^a-zA-Z0-9_]

[^a-zA-Z0-9_+^]



a
?
_
abc
abc?_ac
abc?_ac+abd36_12

Συμβολισμός **οποιοδήποτε χαρακτήρα**

Τι γίνεται άμα θέλαμε να πούμε... «**οποιοδήποτε χαρακτήρα**»;

Χρησιμοποιούμε τον χαρακτήρα «.»

.og

.at

ab.

ab...

ab.+

ab.*

Mr J.+n

Escaping

Έστω ότι όντως θέλαμε να ψάξουμε όμως την «.»;
Θα κάναμε **escape** τον χαρακτήρα!

Χρησιμοποιούμε το πρόθεμα «\»

```
www\[a-zA-Z0-9]\.com
```

Τι ψάχνουν αυτά;

```
\[a-zA-Z0-9\]
```

```
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

Predefined classes

\d – Any digit = [0-9]

\D – NO digit = [^0-9]

\w – Word character = [a-zA-Z0-9_]

\W – NO Word character = [^a-zA-Z0-9_]

\s – Space character = [\t\n]

\S – NO Space character = [^\t\n]

`\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`

FLEX Patterns

RegEx που δουλεύουν στο FLEX

<https://www.cs.virginia.edu/~cr4bd/flex-manual/Patterns.html>