

Εισαγωγή στα Λειτουργικά Συστήματα



SET ΔΙΑΦΑΝΕΙΩΝ 6

ΚΑΝΟΝΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

ΑΝΤΩΝΗΣ ΣΙΔΗΡΟΠΟΥΛΟΣ

Κανονικές Εκφράσεις (Regular Expressions)

2

- Μια κανονική έκφραση είναι ένας σύντομος και σαφής τρόπος έκφρασης οποιουδήποτε συνδυασμού χαρακτήρων.
- Οι κανονικές εκφράσεις (RE) αποτελούνται από συνδυασμό κανονικών χαρακτήρων με έναν ή περισσότερους μεταχαρακτήρες. Οι μεταχαρακτήρες είναι χαρακτήρες με ειδική σημασία.
- Οι κανονικές εκφράσεις χρησιμοποιούνται κυρίως για ΕΛΕΓΧΟ strings.

Κανονικές Εκφράσεις (Regular Expressions)

3

- Οι Κανονικές εκφράσεις δεν είναι μόνο δυνατότητα του Unix. Στο Unix χρησιμοποιούνται ιδιαίτερα, αλλά είναι θέμα να τις υποστηρίξουν οι εφαρμογές που χρησιμοποιούμε.
- ΟΛΕΣ οι γλώσσες προγραμματισμού έχουν υποστήριξη για κανονικές εκφράσεις.
 - Είτε μέσω βιβλιοθηκών, πχ: C, C++
 - Είτε είναι ενσωματωμένες στην γλώσσα, πχ: perl, php, Javascript

Κανονικές Εκφράσεις (Regular Expressions)

4

Προτάθηκαν για πρώτη φορά το 1956 (S. Kleene)

Ο Ken Thomson (1968) τις χρησιμοποίησε στην εντολή **grep** (global **r**egular **e**xpression **p**rint)

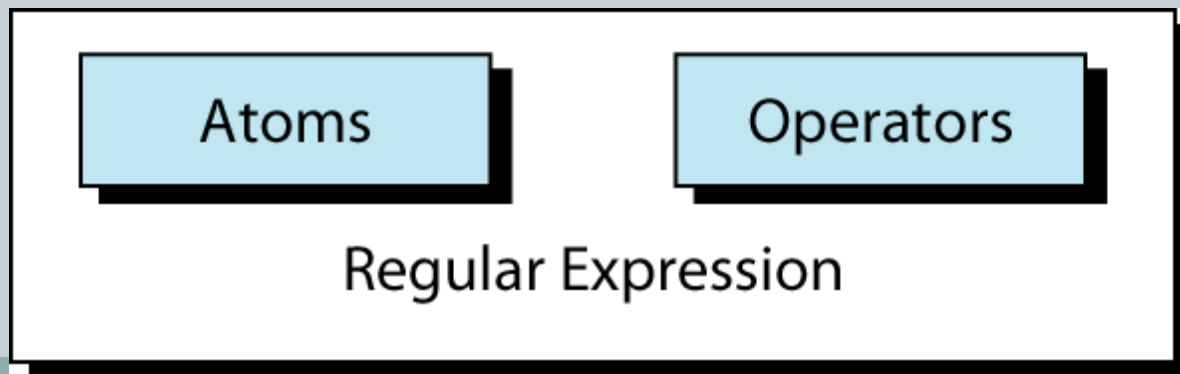
Η εντολή **grep** αναζητά μια δεδομένη κανονική έκφραση μέσα σε ένα αρχείο. Διαβάζει το αρχείο γραμμή-γραμμή και ελέγχει αν στην τρέχουσα γραμμή ταιριάζει η κανονική έκφραση. Αν ναι, τότε τυπώνει την γραμμή, αν όχι τότε προχωρά στην επόμενη

Μερικές από τις πλέον ισχυρές εντολές (λειτουργίες) του UNIX, όπως **grep** και **sed**, χρησιμοποιούν κανονικές εκφράσεις.

Κανονικές Εκφράσεις

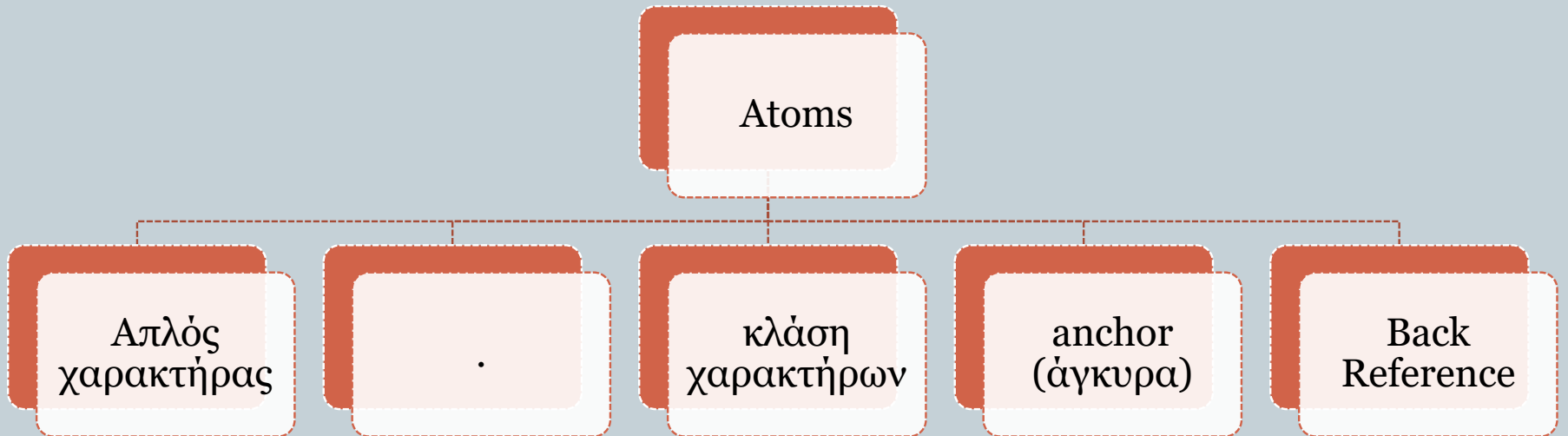
5

- Μια κανονική έκφραση είναι παρόμοια με μια μαθηματική έκφραση.
- Μια μαθηματική έκφραση αποτελείται από τελεστέους (operands) και τελεστές (operators).
- Μια κανονική έκφραση αποτελείται από atoms και operators.
- Το **atom** προσδιορίζει αυτό που αναζητούμε και το σημείο του κειμένου όπου υπάρχει αντιστοιχία.
- Ένας **operator** συνδυάζει atoms σε σύνθετες εκφράσεις.



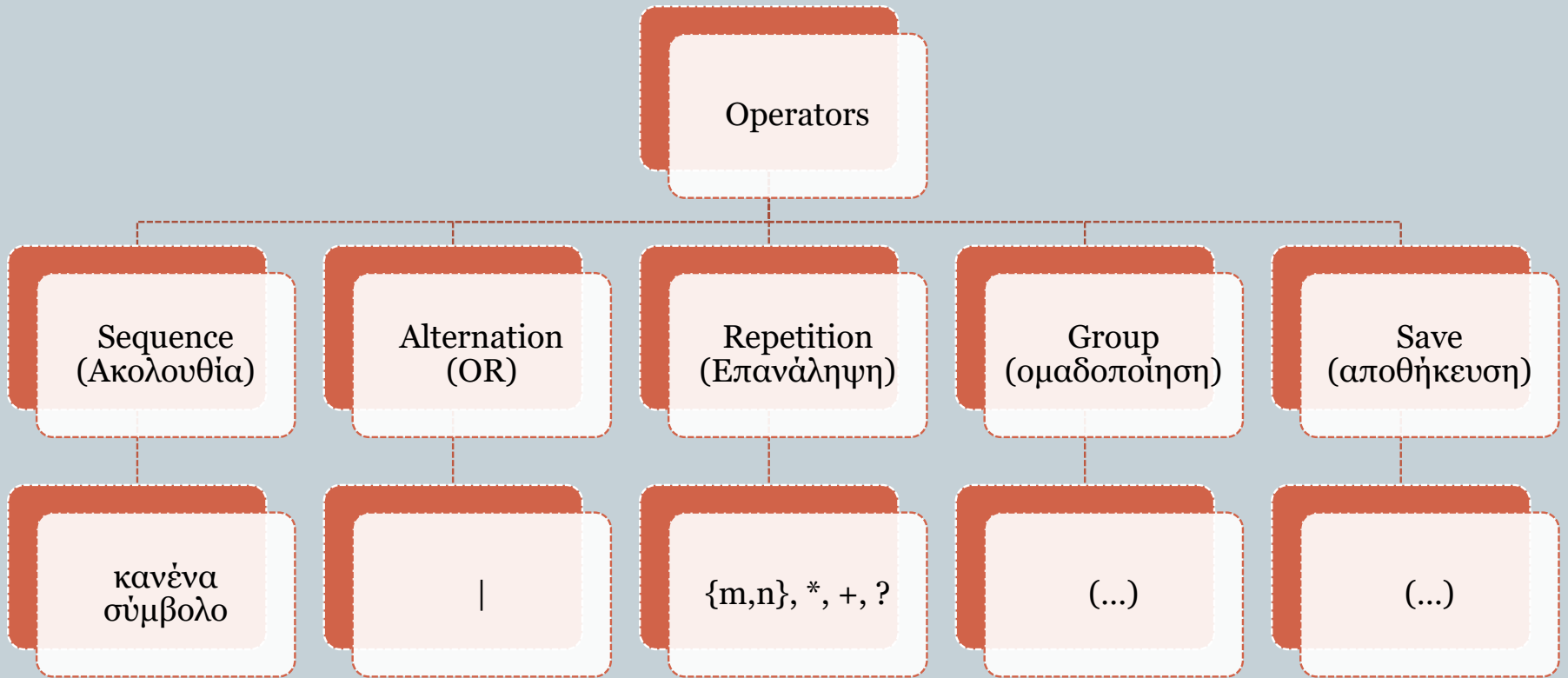
Atoms

6



Operators

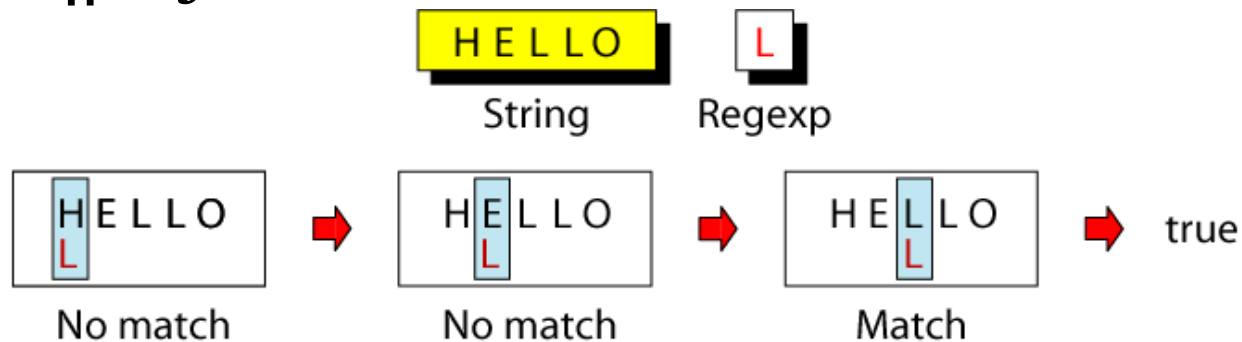
7



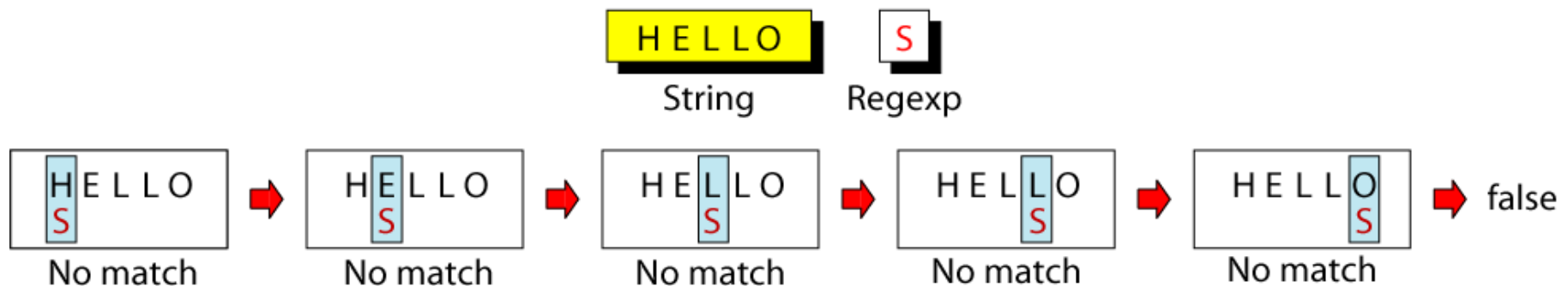
Παράδειγμα απλής RE

8

- Η απλούστερη μορφή RE είναι ένας απλός χαρακτήρας:



(a) Successful Pattern Match



(b) Unsuccessful Pattern Match

Παράδειγμα απλής RE

9

- μια ακολουθία απλών χαρακτήρων:

HELLO

String

ELL

RegExp

HELLO
ELL

No match



HELLO
ELL

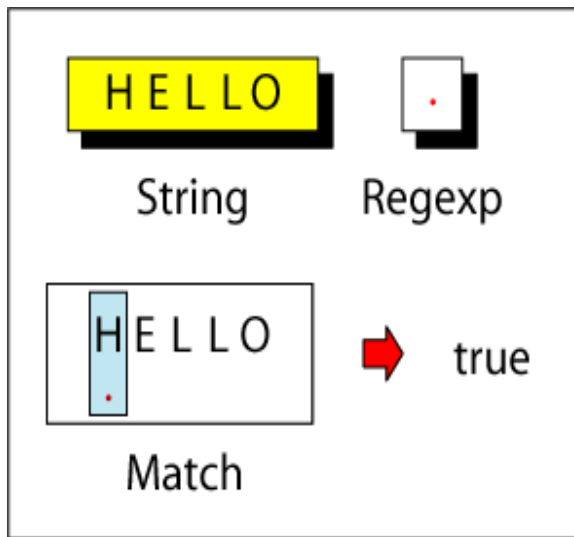
match

Εφόσον δεν υπάρχει κάποιο σύμβολο πράξης, εννοείται η πράξη της ακολουθίας: αναζητούμε τον χαρακτήρα E, ο οποίος ακολουθείται από τον χαρακτήρα L, ο οποίος ακολουθείται από τον χαρακτήρα L

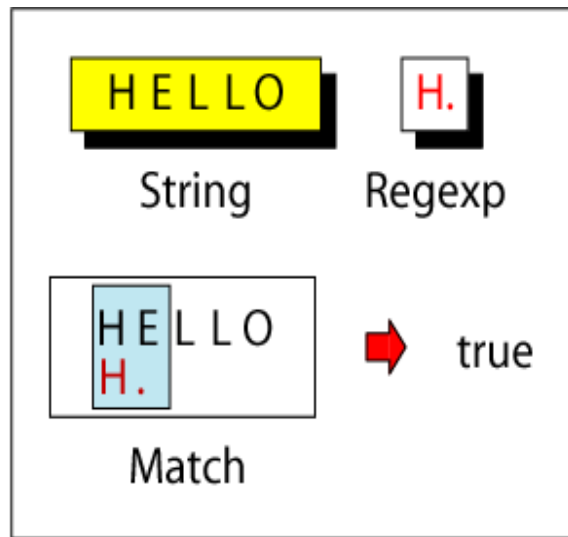
Παράδειγμα RE

10

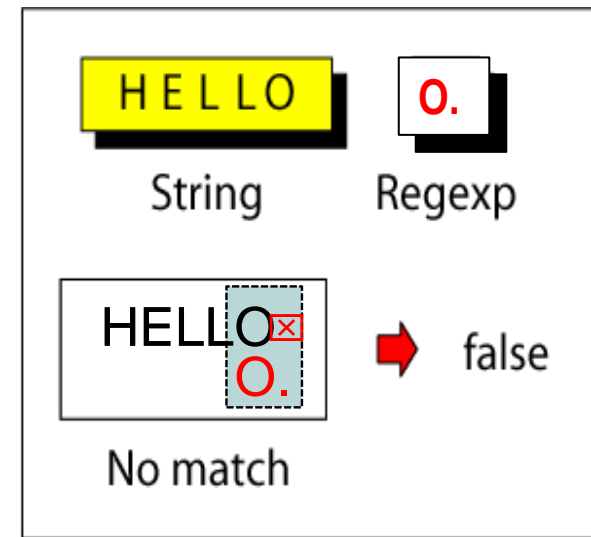
- Η τελεία (.) σημαίνει ένας οποιοσδήποτε χαρακτήρας:



(a) Single-Character



(b) Combination-True



(c) Combination-False

11

-
- The diagram illustrates the difference between String and Regexp matching. It shows three scenarios:
- String**: HELLO
 - Regexp**: [ABL]
- The scenarios are shown in a sequence of three boxes, each with a red arrow pointing to the next:
- No match**: The string "HELLO" is shown with the regex "[ABL]" below it. The string is highlighted in blue, and the regex is highlighted in red. The result is "No match".
 - No match**: The string "HELLO" is shown with the regex "[ABL]" below it. The string is highlighted in blue, and the regex is highlighted in red. The result is "No match".
 - Match**: The string "HELLO" is shown with the regex "[ABL]" below it. The string is highlighted in blue, and the regex is highlighted in red. The result is "Match".
- The final result of the match is "true".

Κλάση χαρακτήρων

12

- Υπάρχουν και pre-defined σύνολα χαρακτήρων:
 - `[:alnum:]` → `o-9a-zA-Z`, eg: `[[:alnum:],+]` ⇔ `[o-9a-zA-Z,+]`
 - `[:alpha:]` → `a-zA-Z`
 - `[:cntrl:]`, `[:print:]` → Κοντρόλ χαρακτήρας (<31 στον πίνακα ASCII), εκτυπώσιμος χαρακτήρας (visible and spaces).
 - `[:digit:]`, `[:xdigit:]` → `o-9` , `o-9A-F`
 - `[:graph:]` → εκτυπώσιμος χαρακτήρας εκτός spaces/tabs.
 - `[:lower:]`, `[:upper:]`, → πεζός, κεφαλαίος
 - `[:punct:]`, `[:space:]` → στίξη (`\`) `[!'"#$%&'()*+,-./:;<=>?@\^_`{|}~-]`

Κλάση χαρακτήρων

13












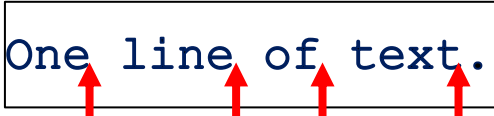
- Υπάρχουν και pre-defined σύνολα χαρακτήρων:
 - `[:alnum:]` → `o-9a-zA-Z`, eg: `[[:alnum:],+]` ⇔ `[o-9a-zA-Z,+]`
 - `[:alpha:]` → `a-zA-Z`
 - `[:cntrl:]`, `[:print:]` → Κοντρόλ χαρακτήρας (<31 στον πίνακα ASCII), εκτυπώσιμος χαρακτήρας (visible and spaces).
 - `[:digit:]`, `[:xdigit:]` → `o-9`, `o-9A-F`
 - `[:graph:]` → εκτυπώσιμος χαρακτήρας εκτός spaces/tabs.
 - `[:lower:]`, `[:upper:]` → μικρός, κεφαλαίος
 - `[:punct:]`, `[:space:]` → `!"#$%&'()*+,-./:;<=>?@\^_`{|}~ -`

Πλεονέκτημα: Περιλαμβάνονται και Α-Ωα-ω και όλα τα γράμματα όλων των εθνικών αλφάβητων όταν το σύνολο χαρακτήρων είναι UTF. Όταν το σύνολο χαρακτήρων είναι ASCII τότε ισχύει
`[:alpha:] == a-zA-Z`

ἄγκυρες

14

- **Anchors** : είναι atoms που χρησιμοποιούνται για να αντιστοιχήσουν το πρότυπο σε ένα συγκεκριμένο τμήμα
- Τα **Anchors** δεν αντιστοιχούνται σε χαρακτήρες αλλά καθορίζουν ένα σημείο.

Anchor		Means	Example
		Beginning of string	
		End of string	
		Beginning of word	
		End of word	

ἄγκυρες

15

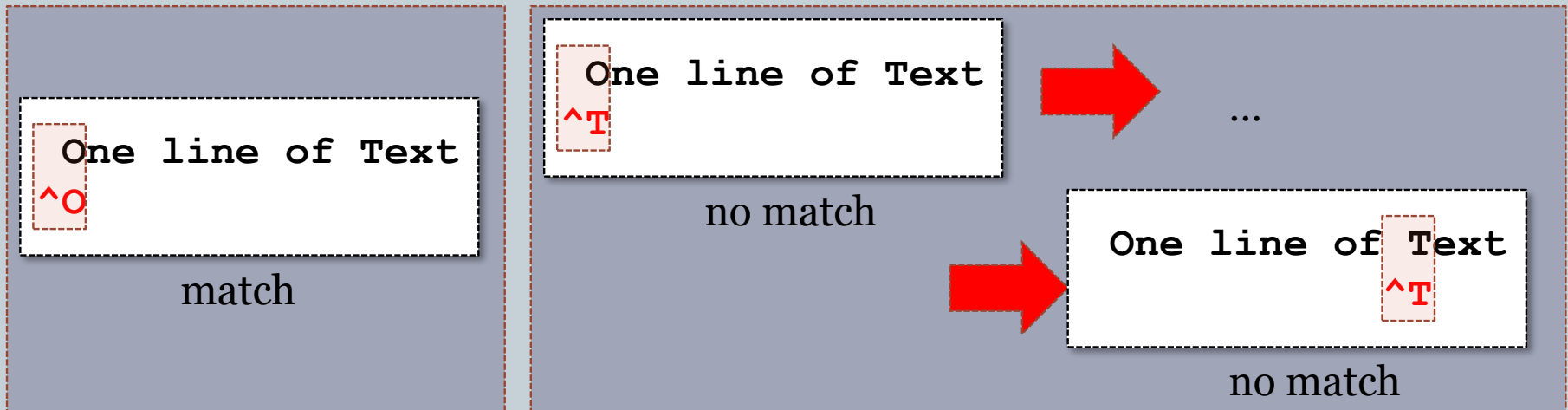
- **Anchors** : είναι atoms που χρησιμοποιούνται για να αντιστοιχήσουν το πρότυπο σε ένα συγκεκριμένο τμήμα
- Τα **Anchors** δεν αντιστοιχούνται σε χαρακτήρες αλλά καθορίζουν ένα σημείο.

One line Of Text

String

^O

RegExp



ἀγκυρες

16

title Of text

String

t\$

RegExp

title Of text

t\$

no match



...

title Of text

t\$

no match



...

title Of text

t\$

no match



...

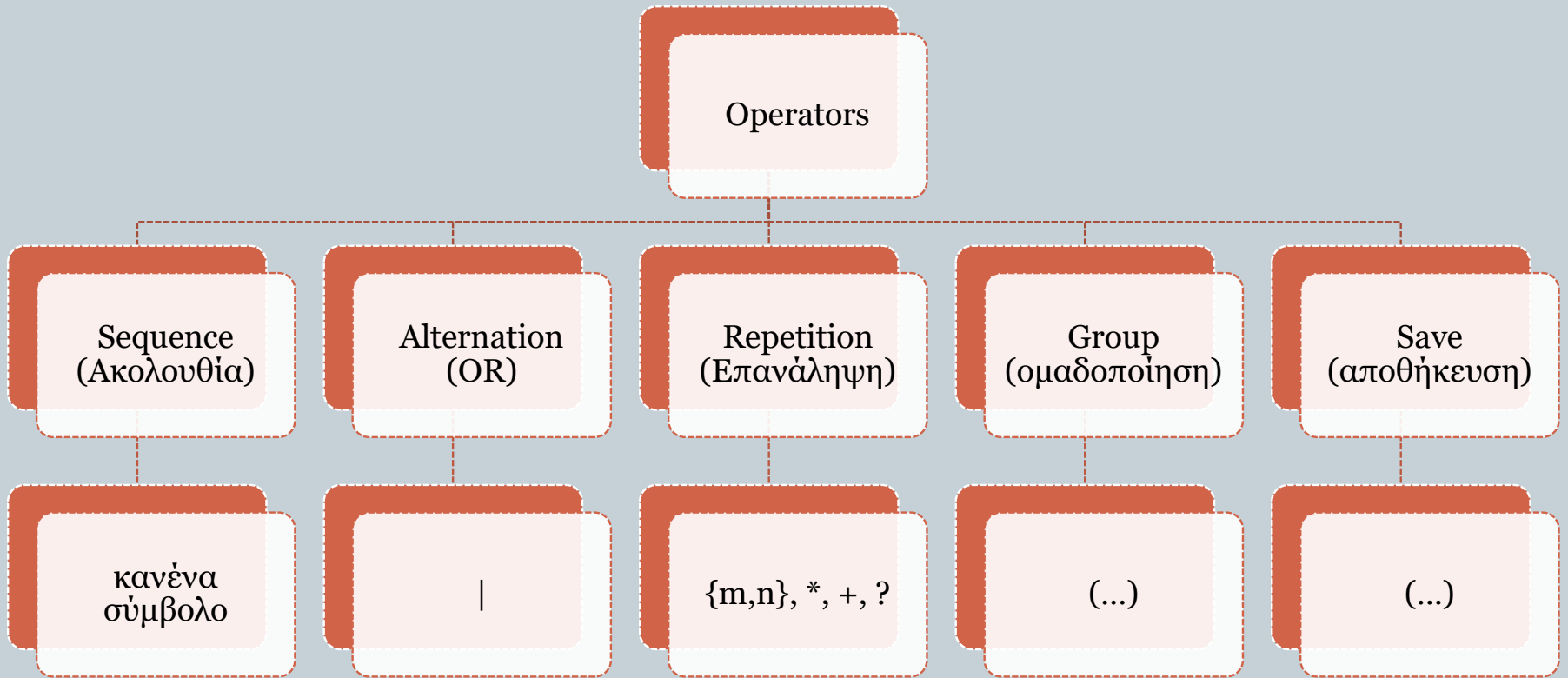
title Of text

t\$

match

Operators

17



Τελεστής Ακολουθίας

18

- Ο τελεστής ακολουθίας (sequence operator) δεν εκφράζεται με σύμβολο.
- Αυτό σημαίνει ότι αν μια σειρά από atoms φαίνονται σε μια κανονική έκφραση, υποδηλώνεται η παρουσία ενός αόρατου sequence operator ανάμεσά τους.

dog



matches the pattern "dog"

a..b



matches "a", any two characters, and "b"

[2-4][0-9]



matches a number between 20 and 49

[0-9][0-9]



matches any two digits

^\$



matches a blank line

^.\$



matches a one-character line

[0-9]-[0-9]



matches two digits separated by a "-"

Τελεστής Ακολουθίας

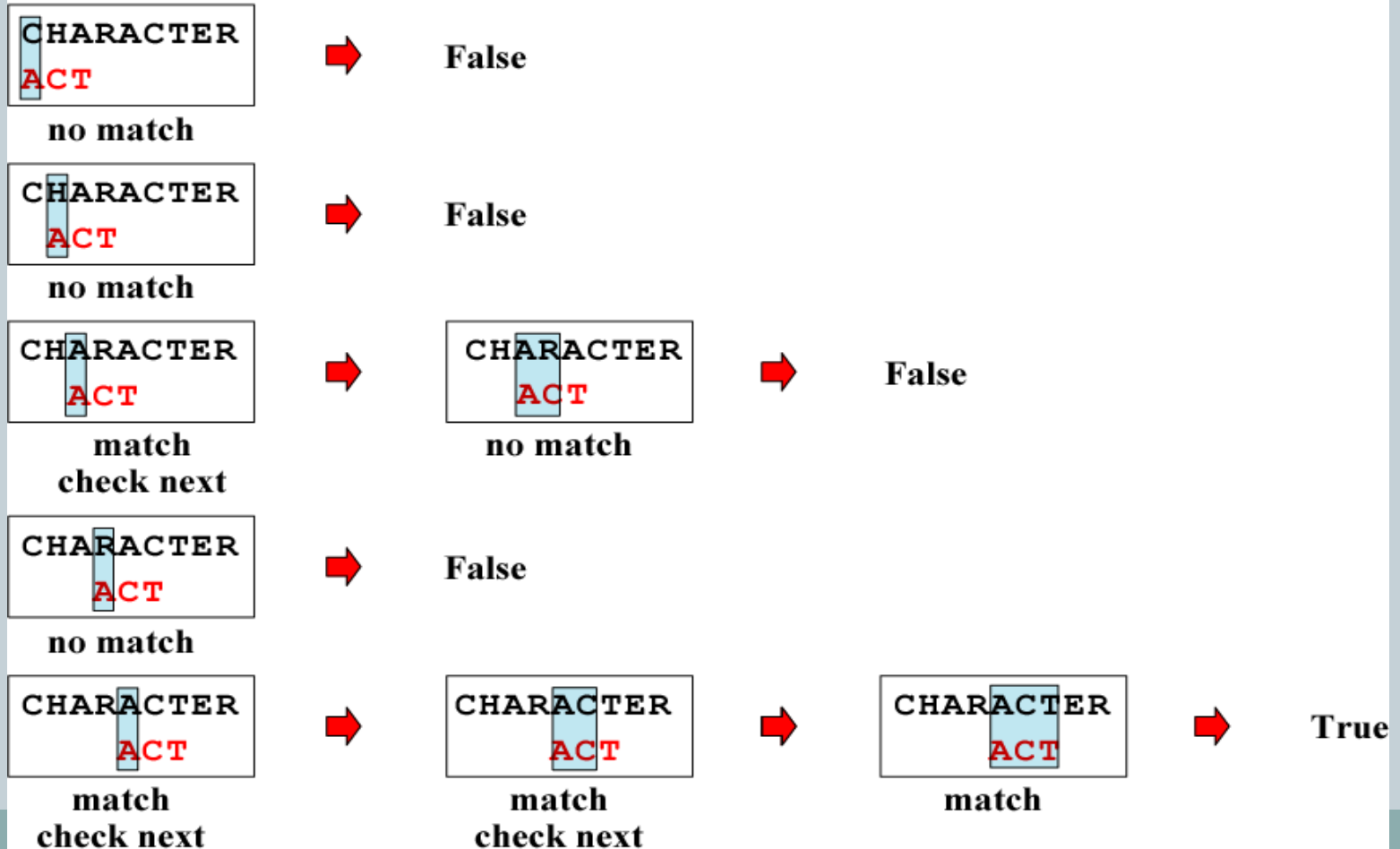
19

CHARACTER

String

ACT

Regexp



Τελεστής OR

20

- Ο τελεστής εναλλαγής (alternation operator) χρησιμοποιείται για να ορίσει μια ή περισσότερες εναλλακτικές περιπτώσεις

UNIX|unix



matches "UNIX" or "unix"

Ms|Miss|Mrs



matches "Ms" or "Miss" or "Mrs"

Τελεστής επανάληψης (Repetition Operator)

21

- Ο τελεστής επανάληψης (repetition operator) καθορίζει για το atom που υπάρχει ακριβώς πριν από την επανάληψη πόσες φορές πρέπει να επαναληφθεί.

σύμβ.	εξήγηση
{n}	Το προηγούμενο atom ακριβώς n φορές
{n,m}	Το προηγούμενο atom από n έως m φορές
{n,}	Το προηγούμενο atom n ή περισσότερες φορές
* ή {0,}	Το προηγούμενο atom 0 ή περισσότερες φορές
+ ή {1,}	Το προηγούμενο atom 1 ή περισσότερες φορές
? ή {0,1}	Το προηγούμενο atom 0 ή 1 φορές

Τελεστής επανάληψης - Παραδείγματα

σύμβ.	εξήγηση
$A\{3\}$	AAA
$B\{2, 4\}$	BB ή BBB ή BBBB \Leftrightarrow BB BBB BBBB
$AB\{1, 3\}$	AB ή ABB ή ABBB
$A\{1, 3\}B$	AB, AAB, AAAB
$A\{1, 3\}B\{1, 3\}$	AB, ABB, ABBB, AAB, AABBB, AABBBB, AAAB, AAABBB, AAABBBB
$AB\{4, \}C$	ABBBBC ή ABBBBBC ή ABBBBBBC ή κτλ.
AB^*C	AC, ABC, ABBC, ABBBC, κτλ.
$[zxc]\{2\}$	$[zxc][zxc]$ (Τα 2 atoms μπορούν να ταιριάζουν σε διαφορετικούς χαρακτήρες, πχ: "zc") s zxc styf, ... zz,zx,zc,xz,xx,xc,cz,cx,cc
$A.\{1, 3\}$	A. ή A.. ή A... \rightarrow AZ, AC, A#, ACC, AXC, ACVB, ACDESR
$A[^A]\{1, 3\}A$	$A[^A]A \mid A[^A][^A]A \mid A[^A][^A][^A]A$ θα ταιριάζουν τα ABA, ABHA, AOIPA, A ABA , ABCBC ABA , "A A" αλλά όχι τα ARTYUA, AA, AAA, ...

Τελεστής επανάληψης - Παραδείγματα

23

σύμβ.	εξήγηση
A^*B	B ή AB ή AAB ή AAAB ή κτλ. , XCXC B ,
$A+B$	AB ή AAB ή AAAB ή κτλ., XCXC B , XT AAB TE
$BA?B$	BB ή BAB, BAAB , BAAB BAB , BXB , BX BB
$B.?B$	BB ή B.B πχ: BAB, BΔB, BOB, B)B, B B, XX B-B
$B.*B$	BB ή B.B ή B..B ή B...B ή B....B ή κτλ. BADSADB , BHGDB B BSG, FSDG B BTF
$^B.*B$	$^BB ^B.B ^B..B $ κτλ. BADSADB , BHGDB B BSG, FSDG B BTF
$^B.*B\$$	$^BB\$ ^B.B\$ ^B..B\$ $ κτλ. BADSADB , BHGDB B BSG, FSDG B BTF
$B.+B$	B.B ή B..B ή B...B ή B....B ή κτλ.
$[0-9]^*[,]?[0-9]^+$	Μηδέν ή περισσότερα νούμερα, «,» μια ή καμία φορές και μετά ένα ή περισσότερα νούμερα

Τελεστής επανάληψης - Παραδείγματα

24

ABBCCCDD

String

BC*D

Regexp

ABBCCCDD
BC*D

No Match



Retry

ABBCCCDD
BC*D

B Matches, Cs?



ABBCCCDD
BC*D

Zero Cs Match



ABBCCCDD
BC*D

No Match For D



Retry

ABBCCCDD
BC*D

B Matches



ABBCCCDD
BC*D

Four Cs Match



ABBCCCDD
BC*D

D Matches



Successful
Pattern Found Is
BCCCD

Τελεστής ομαδοποίησης (Group Operator)

25

- Ο **group operator** είναι ένα ζεύγος παρενθέσεων που ανοίγουν και κλείνουν.
- Όταν μια ομάδα χαρακτήρων περικλείεται σε παρενθέσεις ο επόμενος τελεστής εφαρμόζεται σε όλη την ομάδα.

σύμβ.	εξήγηση
$AB\{2\}$	ABB
$(AB)\{2\}$	ABAB
$A(AB)\{1,3\}XY$	AABXY, AABABXY, AABABABXY
$A((XZ)\{2\}P\{3\})\{3\}$	→ A(XZXZPPP){3} → AXZXZPPPXZXZPPPXZXZPPP

Τελεστής αποθήκευσης (Save) & Back Reference

26

- Με την ομαδοποίηση δίνεται συγχρόνως και «εντολή αποθήκευσης σε buffer» του string που ταίριαξε στο τμήμα της RE που είναι στις παρενθέσεις
- Υπάρχουν 9 buffers που μπορούν να χρησιμοποιηθούν για αποθήκευση.
- Οι buffers συμβολίζονται με \1, \2, \3, ..., \9

σύμβ.	εξήγηση
(.)\1	Θα ταίριαξει στα AA,BB,ΓΓ,...,00,11,22 ... οποιουδήποτε 2 χαρακτήρες που επαναλαμβάνονται. YGDF//GHSFD, hg as f
([0-9])[0-9]*\1	Θα ταίριαξει στα 55, 58765, 1675566551, 4 3 5 2 31, 4 543, 1 2 2334, 3 4567, asdd22tf, sdf 45 drr 5, a2as5235....
([0-9]{3})\1	Θα ταίριαξει στα 346346, 987987, 123123, 51231237 ...
([0-9]{3})\1[A-Z]*\1	Θα ταίριαξει στα 346346346, 987987ADS987, 123123TPEE123, A123123A4B123, ...
([0-9])\1{3}	➔ ([0-9])\1\1\1 . Θα ταίριαξει στα 5555, 9999, 2222, 233222243, 34222222456, ...
([0-9]{2})\1{2}	➔ ([0-9][0-9])\1\1. Θα ταίριαξει στα 555555, 828282, 161616, ...
(.)(.){2}\2\1	➔ (.)(..){2}\2\1 Θα ταίριαξει στα ABXYBA, EP98PE, 6DYRERE, A6DYREYD6R
(.){3}\1	AXCDA, XCXADSX

- $([0-9])[0-9]\{8\}\backslash 1 \rightarrow \mathbf{012345678045}$
89712345678700
- $^([0-9])[0-9]\{8\}\backslash 1\$$
- $\backslash <([0-9])[0-9]\{8\}\backslash 1\backslash > \rightarrow \text{hello } \mathbf{0123456780} \text{ hello}$
- $\backslash <([0-9])[0-9]\{8\}\backslash 1\backslash > \rightarrow \text{hello } \mathbf{0123456780}$

Backslash

28

- Ο χαρακτήρας \ (backslash) χρησιμοποιείται είτε για να αποδώσει ειδική σημασία στον χαρακτήρα που ακολουθεί, είτε για να αναιρέσει την ειδική σημασία του (αν είναι ειδικός χαρακτήρας).

σύμβ.	εξήγηση
*	Τελεστής επανάληψης
*	Ο χαρακτήρας «*»
\[Ο χαρακτήρας «[»
...	...

Ειδικοί χαρακτήρες

29

- Ο χαρακτήρας \ (backslash) χρησιμοποιείται είτε για να αποδώσει ειδική σημασία στον χαρακτήρα που ακολουθεί, είτε για να αναιρέσει την ειδική σημασία του (αν είναι ειδικός χαρακτήρας).

σύμβ.	εξήγηση
*	τελεστής
+	τελεστής
?	τελεστής
[κλάση
{	τελεστής
(,)	ομαδοποίηση
	or

σύμβ.	εξήγηση
]	μόνο μετά από [
}	μόνο μετά από {
^	στην αρχή της RE (αρχή string), όταν είναι πρώτος χαρακτήρας μέσα σε κλάση (άρνηση), σε άλλα σημεία δεν έχει ειδική σημασία.
-	μέσα σε κλάση (εύρος). Αλλιώς ο χαρ. «-».
\$	στο τέλος της RE (τέλος string). Αλλιώς ο χαρ. «\$».
.	οποιοσδήποτε χαρακτήρας. Μέσα σε [] ο χαρακτήρας «.».

- μέσα σε μια κλάση χαρακτήρων, (σχεδόν) όλοι οι ειδικοί χαρακτήρες χάνουν την ειδική σημασία τους.

πχ: `[{}*() /+*?|.]` σημαίνει ένας χαρακτήρας από τους `{ } * () / + * ? | .`

Ειδικοί χαρακτήρες

30

- Ο χαρακτήρας `\` (backslash) χρησιμοποιείται είτε για να αποδώσει ειδική σημασία στον χαρακτήρα που ακολουθεί, είτε για να αναιρέσει την ειδική σημασία του (αν είναι ειδικός χαρακτήρας).

σύμβ.	εξήγηση
<code>\1..\9</code>	back reference
<code>\<, \></code>	anchors
<code>\w</code>	word character ⇔ <code>[a-zA-Z0-9_]</code>
<code>\d</code>	digit ⇔ <code>[0-9]</code>
<code>\s</code>	spaces ⇔ spaces and tabs and lines breaks
<code>\w, \d, \s</code>	άρνηση των παραπάνω

- μέσα σε μια κλάση χαρακτήρων, (σχεδόν) όλοι οι ειδικοί χαρακτήρες χάνουν την ειδική σημασία τους.

πχ: `[{}*()/+*?|.]` σημαίνει ένας χαρακτήρας από τους `{ } * () / + * ? | .`

τύποι κανονικών εκφράσεων

31

- Υπάρχουν διάφοροι τύποι κανονικών εκφράσεων (όχι σε όλα συμβατοί μεταξύ τους)
 - IEEE POSIX
 - ✦ Basic Regular Expressions (BRE)
 - ✦ Extended Regular Expressions (ERE)
 - Perl
 - και άλλοι νεώτεροι (ανάλογα με την γλώσσα προγραμματισμού)
 - ✦ πχ: η `phpr` είχε αναπτύξει δικό της μηχανισμό κανονικών εκφράσεων, αλλά πλέον καταργήθηκε και χρησιμοποιείται της `perl` (`preg`)

BRE vs. ERE

32

- In basic regular expressions the meta-characters `?`, `+`, `{`, `|`, `(`, and `)` lose their special meaning; instead use the back slashed versions `\?`, `\+`, `\{`, `\|`, `\(`, and `\)`.

ERE	BRE
<code>?</code>	<code>\?</code>
<code>+</code>	<code>\+</code>
<code>{ }</code>	<code>\{ \}</code>
<code>()</code>	<code>\(\)</code>
<code> </code>	<code>\ </code>

Χρήσεις των Κανονικών Εκφράσεων

33

- Οι κανονικές εκφράσεις δεν είναι δυνατότητα του UNIX, αλλά μπορούν να χρησιμοποιηθούν μέσα από οποιαδήποτε γλώσσα προγραμματισμού (perl, php, javascript, java, VB, C++, C#, C, python, SQL,...)
- Χρήσεις:
 - Αναζήτηση προτύπων και εξόρυξη πληροφορίας από αρχεία ή ροές δεδομένων. (grep, vi, ...)
 - Αντικατάσταση προτύπων σε αρχεία ή ροές δεδομένων. (sed, vi, ...)
 - Έλεγχος μορφής (σύνταξης) ενός string.

Χρήσεις των Κανονικών Εκφράσεων (2)

34

- Παράδειγμα:
- Σε μια φόρμα (ή σε οποιοδήποτε πρόγραμμα) ζητούμε από τον χρήστη να εισάγει αριθμό τηλεφώνου. Δεν θέλουμε όμως να πληκτρολογήσει spaces, παύλες, τελείες ή άλλους χαρακτήρες εκτός από αριθμούς. Επίσης, το μήκος τηλεφώνου πρέπει να είναι 10.
- Για να γίνει αυτός ο έλεγχος μέσα από μια γλώσσα προγραμματισμού χωρίς την χρήση κανονικών εκφράσεων απαιτείται να γράψουμε πάνω από 10 γραμμές κώδικα.
- Με χρήση κανονικών εκφράσεων αρκεί να ελέγξουμε ότι το string που έδωσε ο χρήστης ταιριάζει στην κανονική έκφραση:

`^[0-9]{10}$`

`^[26][0-9]{9}$`

Χρήσεις των Κανονικών Εκφράσεων (3)

35

- Php:
 - `if(preg_match('/^[0-9]{10}$/', $input_string)) { ... }`
- Perl:
 - `If(/^[0-9]{10}$/ =~ $input_string) { ... }`
- Javascript:
 - `var myreg = /^[0-9]{10}$/ ;`
 - `if (myreg.exec(input_string)) { ... }`
- Java:
 - `String input_string;`
 - `...`
 - `if(input_string.match("^[0-9]{10}$")) { ... }`
- C#:
 - `If (Regex.Match(input_string, @"^[0-9]{10}$").Success) { ... }`

Χρήσεις των Κανονικών Εκφράσεων (4)

36

- Σε περίπτωση που θέλουμε να δεχόμαστε και την διεθνή μορφή τηλεφώνου, πχ: +302310999888, θα πρέπει απλά να επεκτείνουμε την κανονική έκφραση:

```
(^[0-9]{10}$|^\\+[0-9]{11,13}$)
```