Προγραμματισμός Διαδικτύου

Δρ. Μηνάς Δασυγένης
mdasygenis@uowm.gr

Τμήμα της παρουσίασης δημιουργήθηκε από τον κ. Παναγιώτη Συμεωνίδη (ICTE/UOWM, 2008-2009)

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Σκοπός ενότητας

- ✓ Καλή γνώση των κανονικών εκφράσεων
- ✓ Δημιουργία κλάσεων
- ✓ Εξοικείωση με τη χρήση τελεστών

Προγραμματισμός διαδικτύου

Κανονικές εκφράσεις

Περιεχόμενα

Κανονικές εκφράσεις (Regular Expressions-RE)

- 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

Παραδείγματα

- Οι κανονικές εκφράσεις χρησιμοποιούνται κυρίως για την αναζήτηση αλφαριθμητικών γενικής μορφής είτε σε αρχεία είτε για επικύρωση των δεδομένων φόρμας!!!!

  - Για την επικύρωση δεδομένων που περιέχουν ένα από τα παρακάτω:

    - Τη λέξη Unix,
    - Τη λέξη UNIX,
    - Ένα πρότυπο (pattern) που αποτελείται από 4 ψηφία
    - Έναν ταχυδρομικό κωδικό,
    - Ένα όνομα,
    - Όλα τα φωνήεντα σε αλφαβητική σειρά.

Κανονικές εκφράσεις

Προτάθηκαν για πρώτη φορά το 1956 (S. Kleene)

- Ο Ken Thomson (1968) τις χρησιμοποίησε στην εντολή grep (global regular expression print)

Μια κανονική έκφραση είναι ένα πρότυπο αποτελούμενο από μια ακολουθία χαρακτήρων που αντιστοιχούνται με το προς εξέταση κείμενο.

Ο διερμηνευτής εξετάζει το κείμενο ως προς το πρότυπο για να αποφασίσει αν το πρότυπο (pattern) και το κείμενο (text) ταιριάζουν (αντιστοιχούνται).

Αν υπάρχει αντιστοιχία η έκφραση είναι αληθής και η εντολή που χρησιμοποιεί το πρότυπο εκτελείται.

Κανονικές εκφράσεις

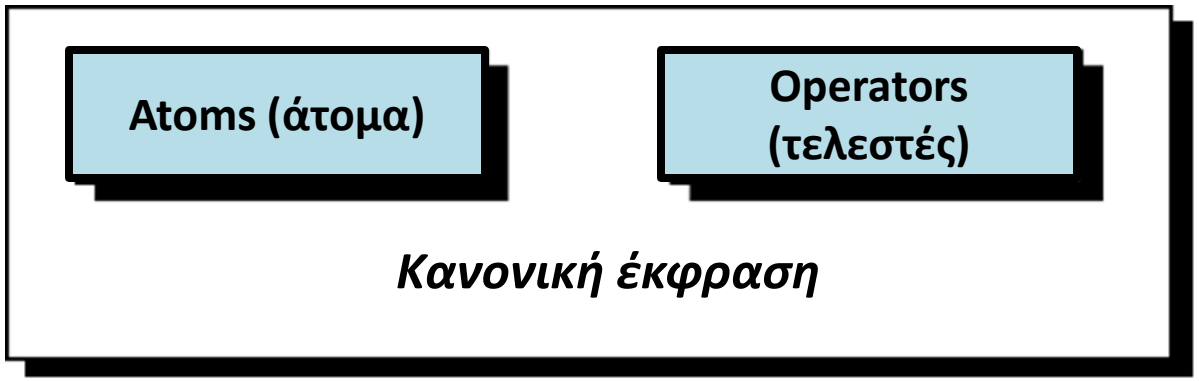
- Μια κανονική έκφραση είναι μια σειρά από κανονικούς χαρακτήρες και ειδικούς τελεστές.

- Οι κανονικοί χαρακτήρες περιλαμβάνουν το σύνολο των πεζών και κεφαλαίων γραμμάτων, τα ψηφία, και άλλους συχνά χρησιμοποιούμενους χαρακτήρες : ~, ', !, @, #, _, -, =, :, ;, ,,/

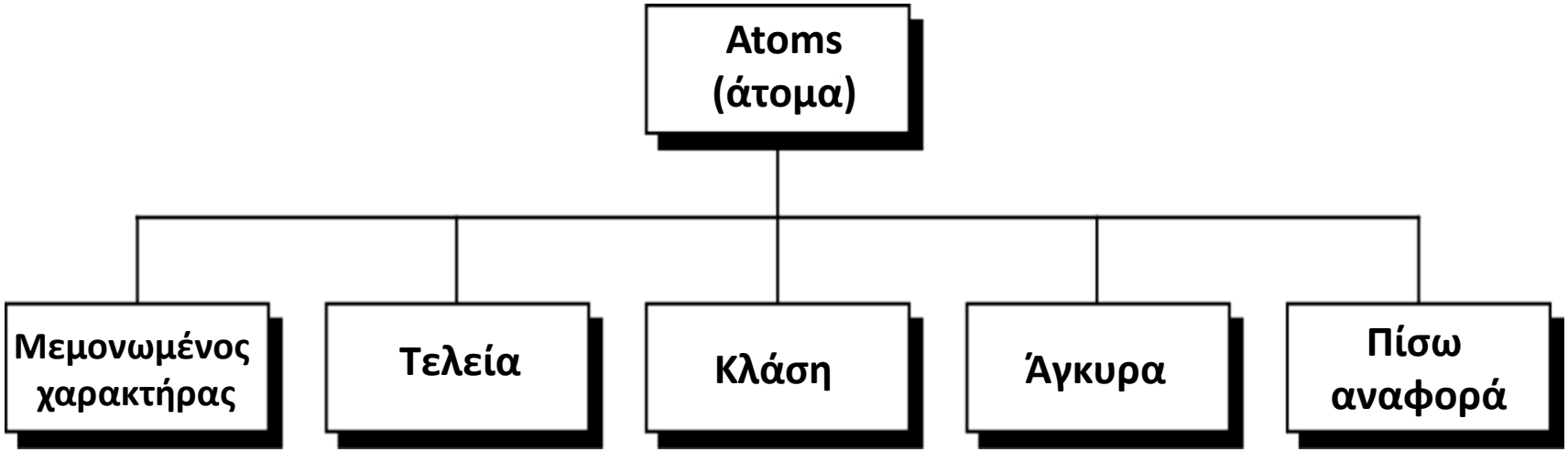
- Οι ειδικοί τελεστές είναι \, ., *, [, ^, $,].

Κανονική έκφραση

- Μια κανονική έκφραση είναι πανόμοια με μια μαθηματική.
- Μια μαθηματική έκφραση αποτελείται από τελευταίους (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.

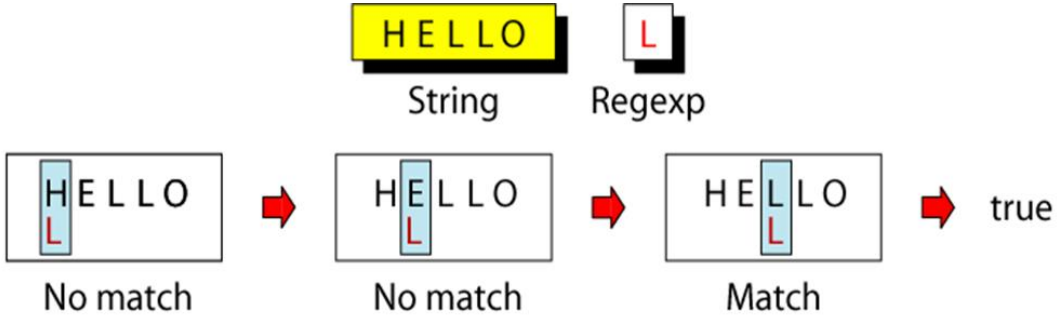


Atoms (άτομα)

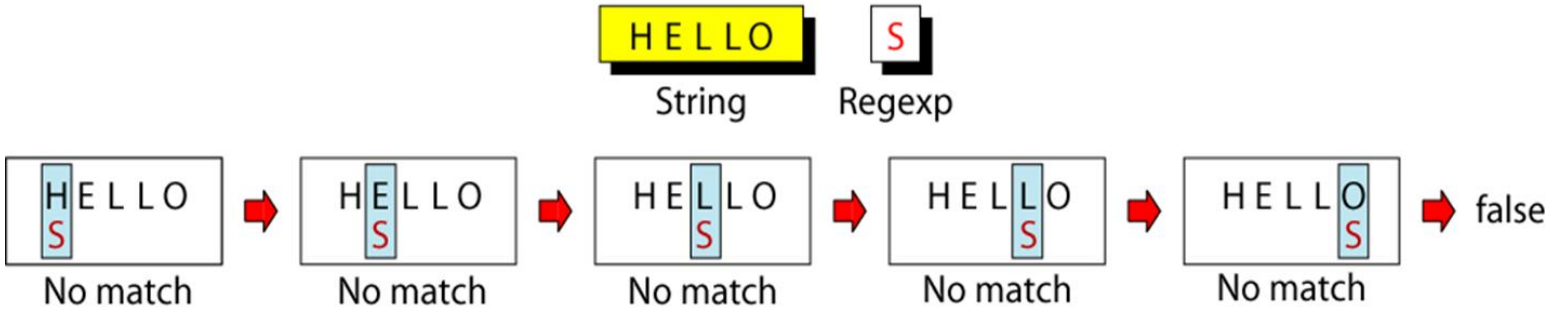

Atoms
(άτομα)
Μεμονωμένος χαρακτήρας
Τελεία
Κλάση
Άγκυρα
Πίσω αναφορά

Παράδειγμα προτύπου ενός χαρακτήρα

Το απλούστερο atom είναι ένας χαρακτήρας



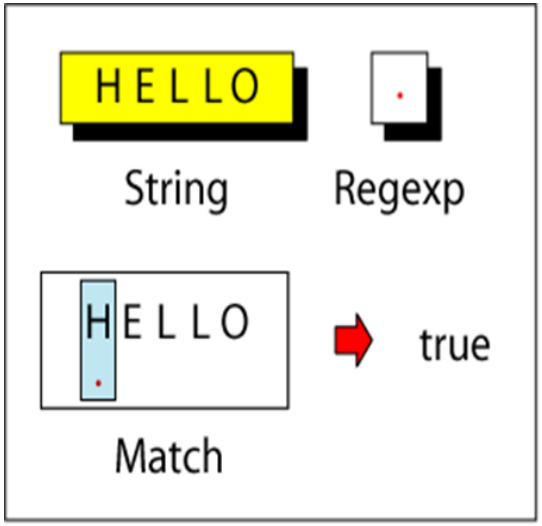
(a) Επιτυχημένο ταίριασμα



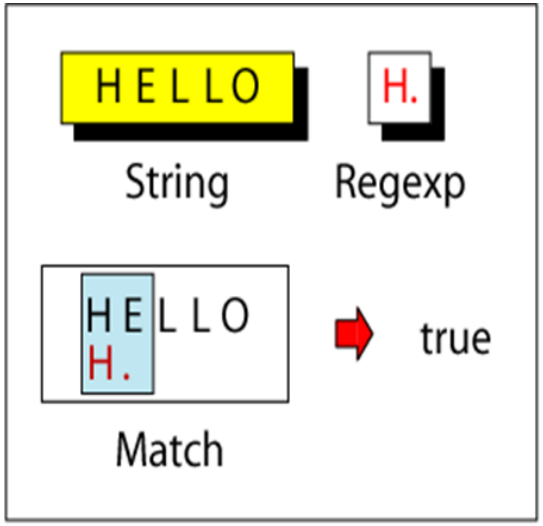
(b) Ανεπιτυχές ταίριασμα

Παράδειγμα Dot Atom

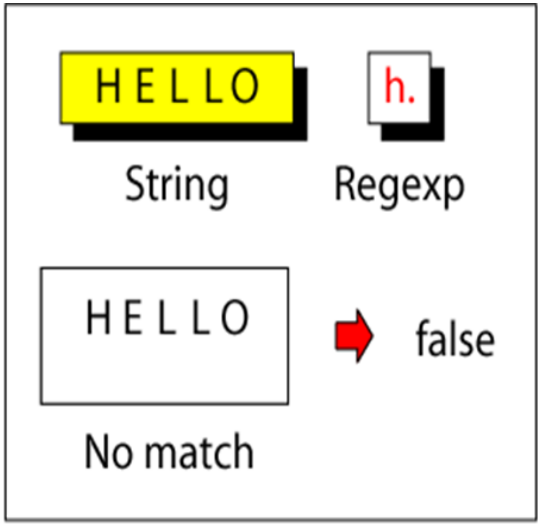
Μια τελεία (dot) αντιστοιχεί με οποιοδήποτε απλό χαρακτήρα εκτός από τον new line character (\n).



(a) Μεμονωμένος χαρακτήρας



(b) Συνδυασμός - true



(c) Συνδυασμός - false

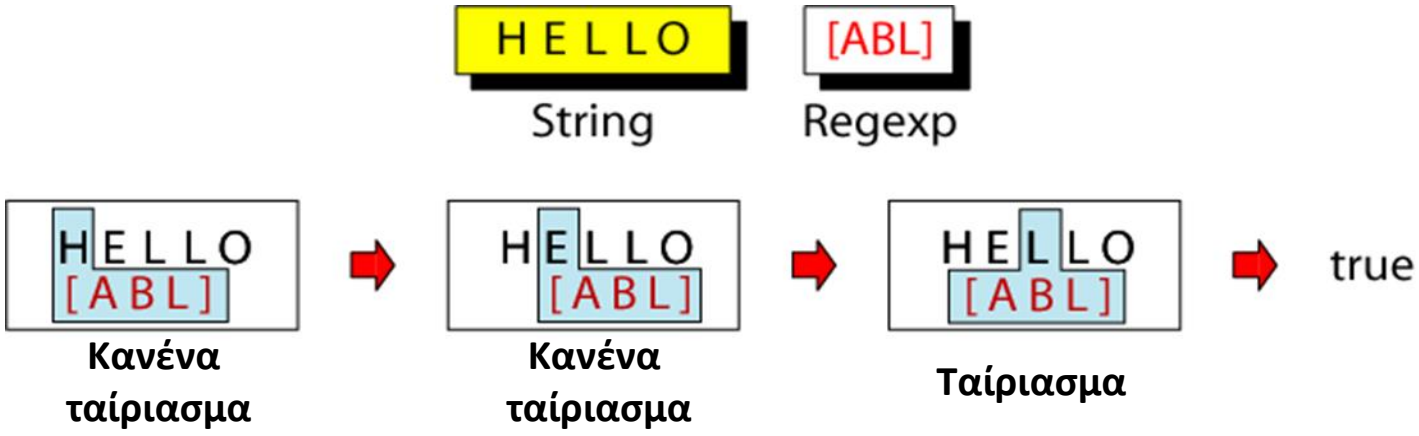
Παράδειγμα Class Atom

Μια class atom ορίζει ένα σύνολο ASCII χαρακτήρων, καθένας από τους οποίους μπορεί να αντιστοιχεί με οποιουσδήποτε χαρακτήρες του κειμένου.

Το σύνολο χαρακτήρων που χρησιμοποιείται στη διαδικασία αντιστοίχισης (matching process) περικλείεται σε αγκύλες.

Μια περιοχή (range) χαρακτήρων κειμένου ορίζεται με μια παύλα (-) π.χ. [a-d]

Το ^ είναι ένας χαρακτήρας εξαίρεσης π.χ. κάθε χαρακτήρας εκτός από φωνήεν, χρησιμοποιείται στην έκφραση [^aeiou].



Παραδείγματα κλάσεων

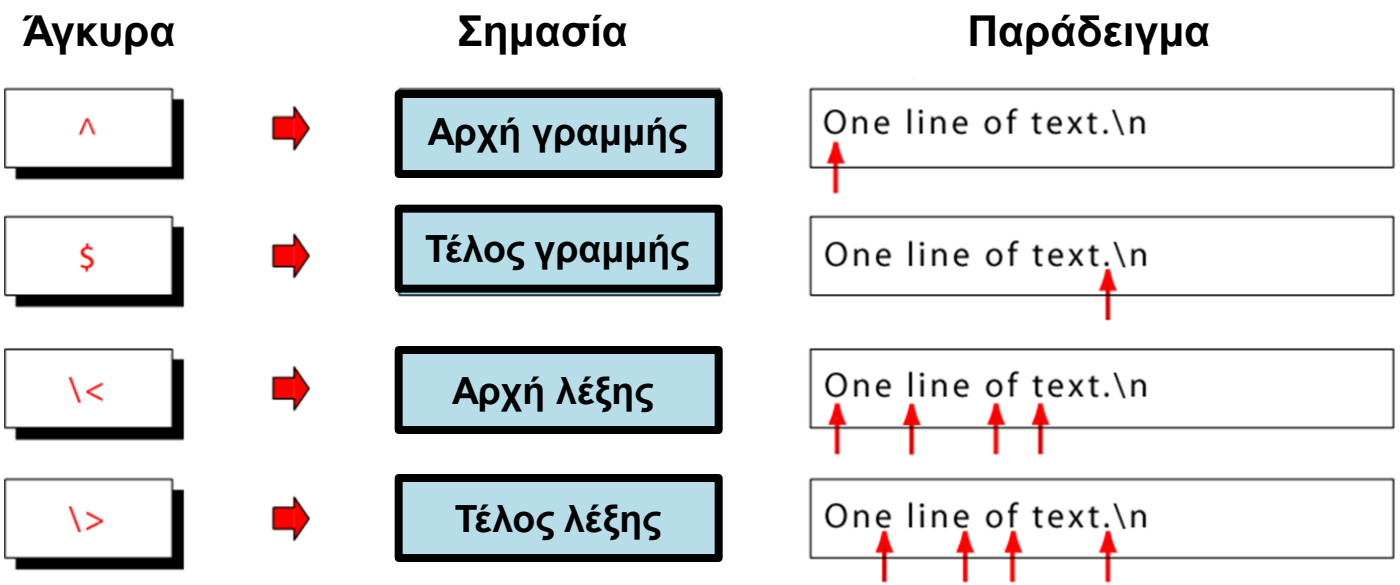
Ο escape character (\) χρησιμοποιείται όταν ο αντιστοιχούμενος χαρακτήρας είναι ένας από τους **–** και **^**

Κοινή έκφραση	Σημασία
[A-H]	[ABCDEFGH]
[A-Z]	Κάθε κεφαλαίο γράμμα
[0-9]	Κάθε ψηφίο
[[a]	[ή a
[0-9\-]	Ψηφίο ή ενωτικό
[^AB]	Κάθε χαρακτήρας εκτός A ή B
[A-Za-z]	Κάθε γράμμα
[^0-9]	Κάθε γράμμα εκτός των ψηφίων
[]a]	] ή a
[^\^]	Όλα εκτός ^

Anchors (Άγκυρες)

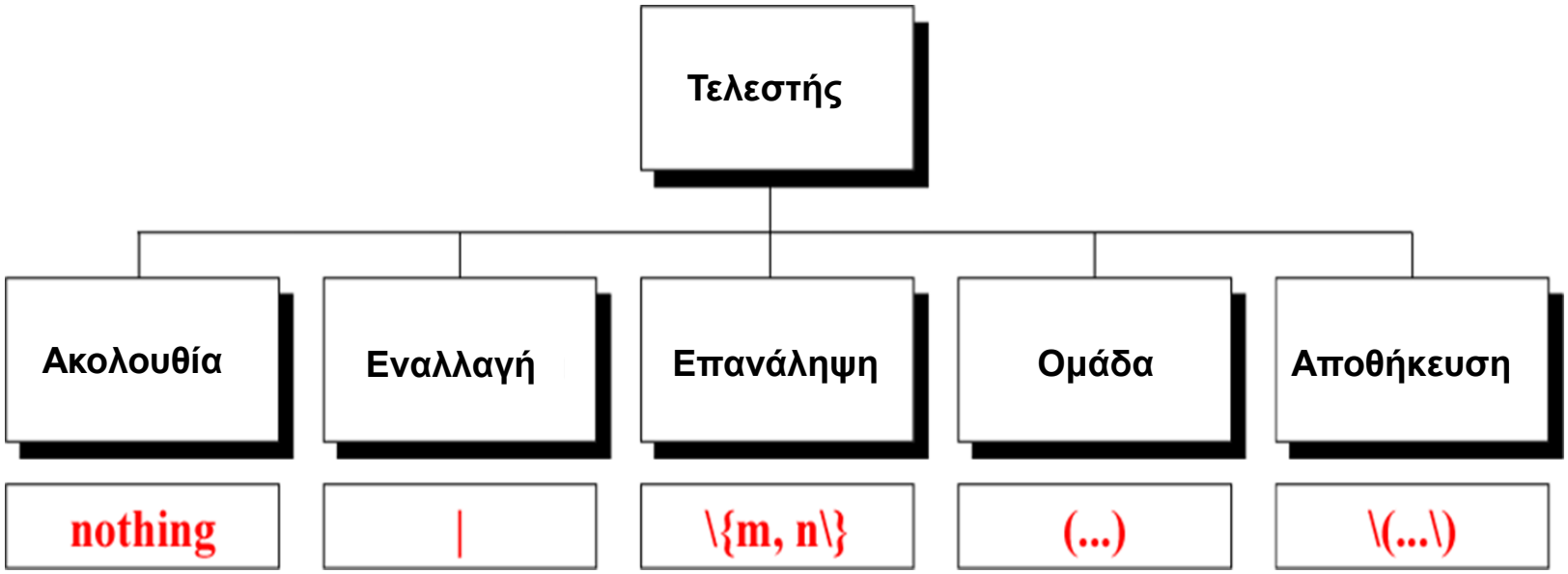
Anchors : είναι atoms (άτομα) που χρησιμοποιούνται για να αντιστοιχήσουν το πρότυπο σε ένα συγκεκριμένο τμήμα.

Τα Anchors δεν αντιστοιχούνται στο κείμενο αλλά καθορίζουν το σημείο όπου πρέπει να βρεθεί ο επόμενος χαρακτήρας στο πρότυπο.



Τελεστές

Μπορούμε να συνδυάσουμε atoms με operators



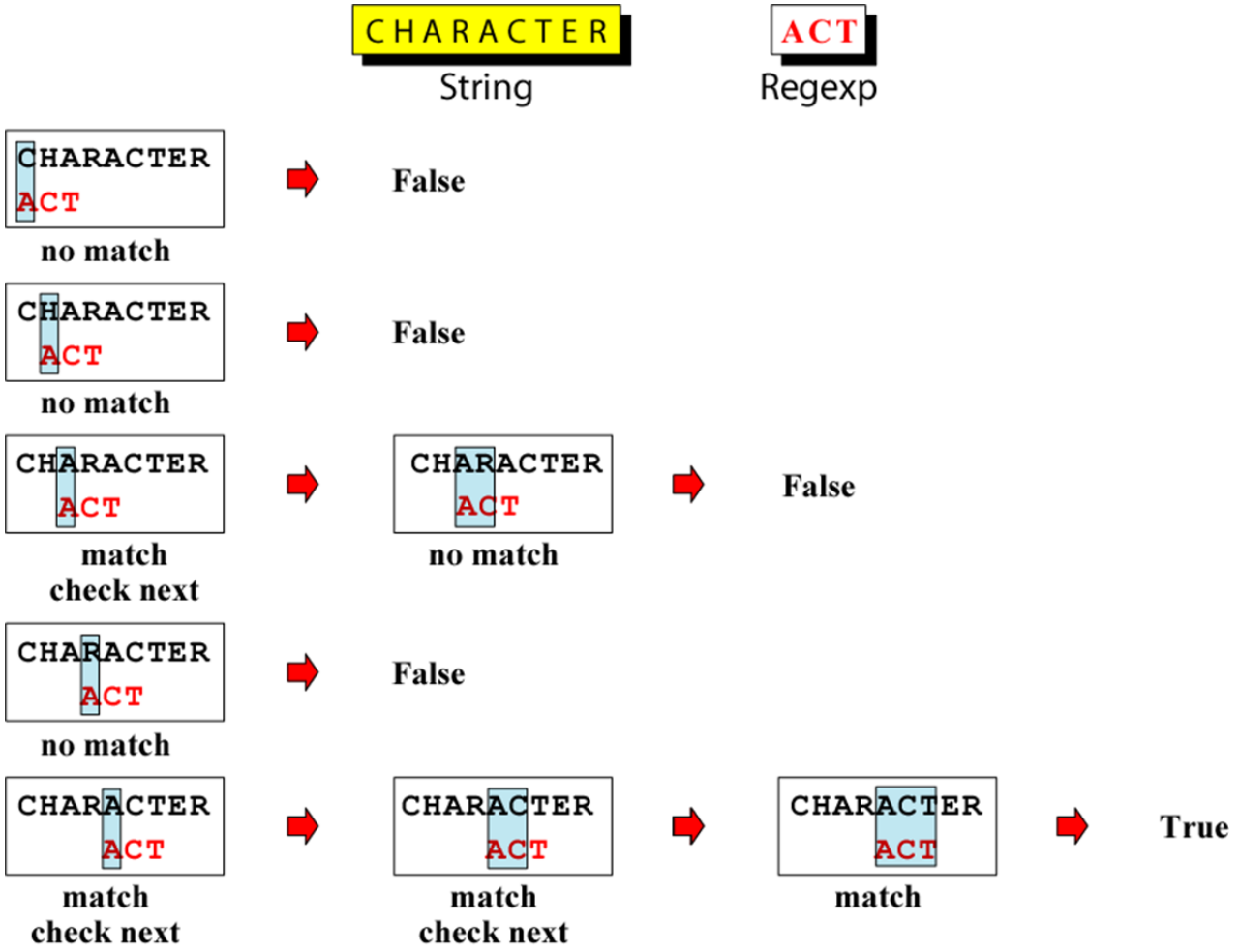
Παραδείγματα τελεστών ακολουθίας

Ο τελεστής ακολουθίας (sequence operator) δεν υπάρχει.

Αυτό σημαίνει ότι αν μια σειρά από atoms φαίνονται σε μια κανονική έκφραση, υποδηλώνεται η παρουσία ενός αόρατου sequence operator ανάμεσά τους.

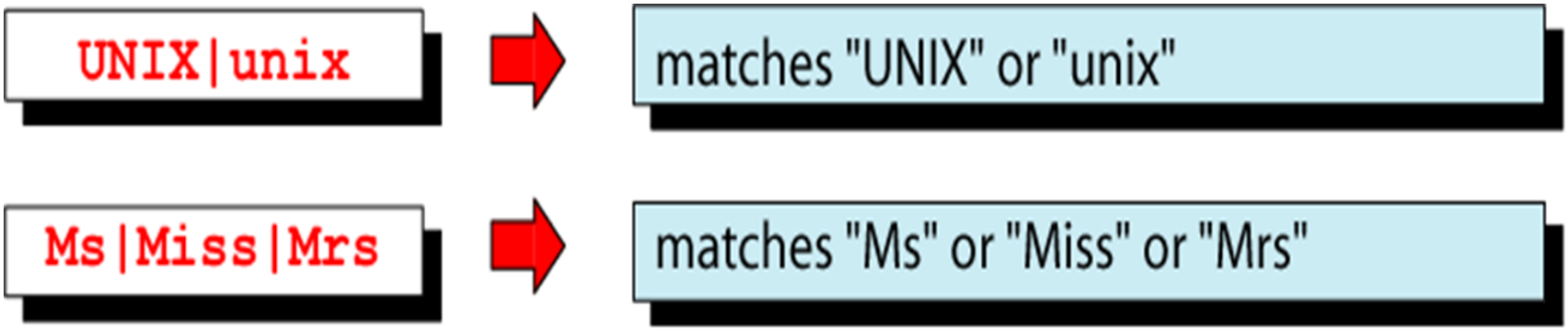
dog	Ταιριάζει με το πρότυπο “dog”
a…b	Αντιστοιχεί το “a”, άλλους δυο χαρακτήρες, και “b”
[2-4] [0-9]	Αντιστοιχεί νούμερο μεταξύ 20 και 49
[0-9] [0-9]	Αντιστοιχεί κάθε δυο αριθμούς
^$	Αντιστοιχεί μια λευκή γραμμή
^.$	Αντιστοιχεί σε μια γραμμή χαρακτήρων
[0-9] – [0-9]	Αντιστοιχεί δυο αριθμούς που χωρίζονται από “-”

Παράδειγμα χρήσης τελεστής ακολουθίας

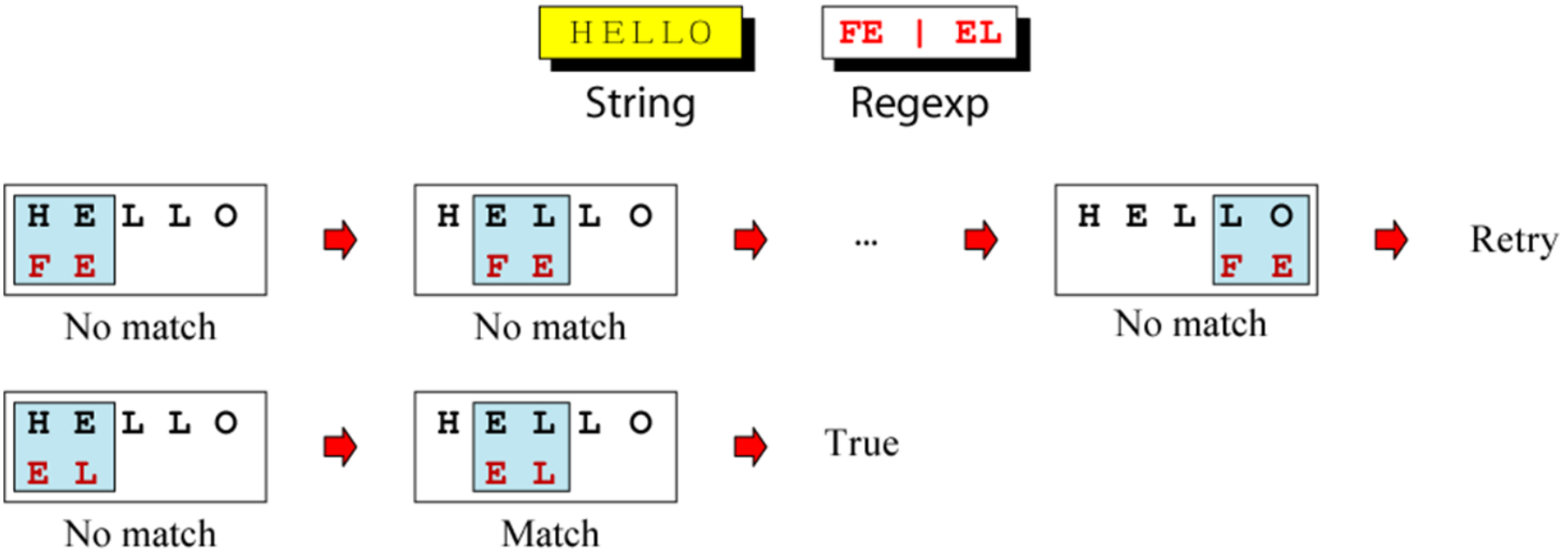


Τελεστής εναλλαγής (Alternation Operator)

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



Αντιστοίχιση Alternation Operators



Τελεστής επανάληψης (Repetition Operator)

Ο τελεστής επανάληψης (repetition operator) καθορίζει ότι το atom ή η έκφραση που υπάρχει ακριβώς πριν από την επανάληψη μπορεί να επαναληφθεί.

m είναι ο ελάχιστος αριθμός επαναλήψεων.

n είναι ο μέγιστος αριθμός επαναλήψεων.

```
\{m , n\}
```

Αντιστοιχεί τον προηγούμενο χαρακτήρα m σε n φορές

```
A\{3 , 5\}
```

→ **Αντιστοιχεί “AAA”, “AAAA”, ή “AAAAA”**

```
BA\{3 , 5\}
```

→ **Αντιστοιχεί “BAAA”, “BAAAA” ή “BAAAAA”**

Βασικές φόρμες επανάληψης

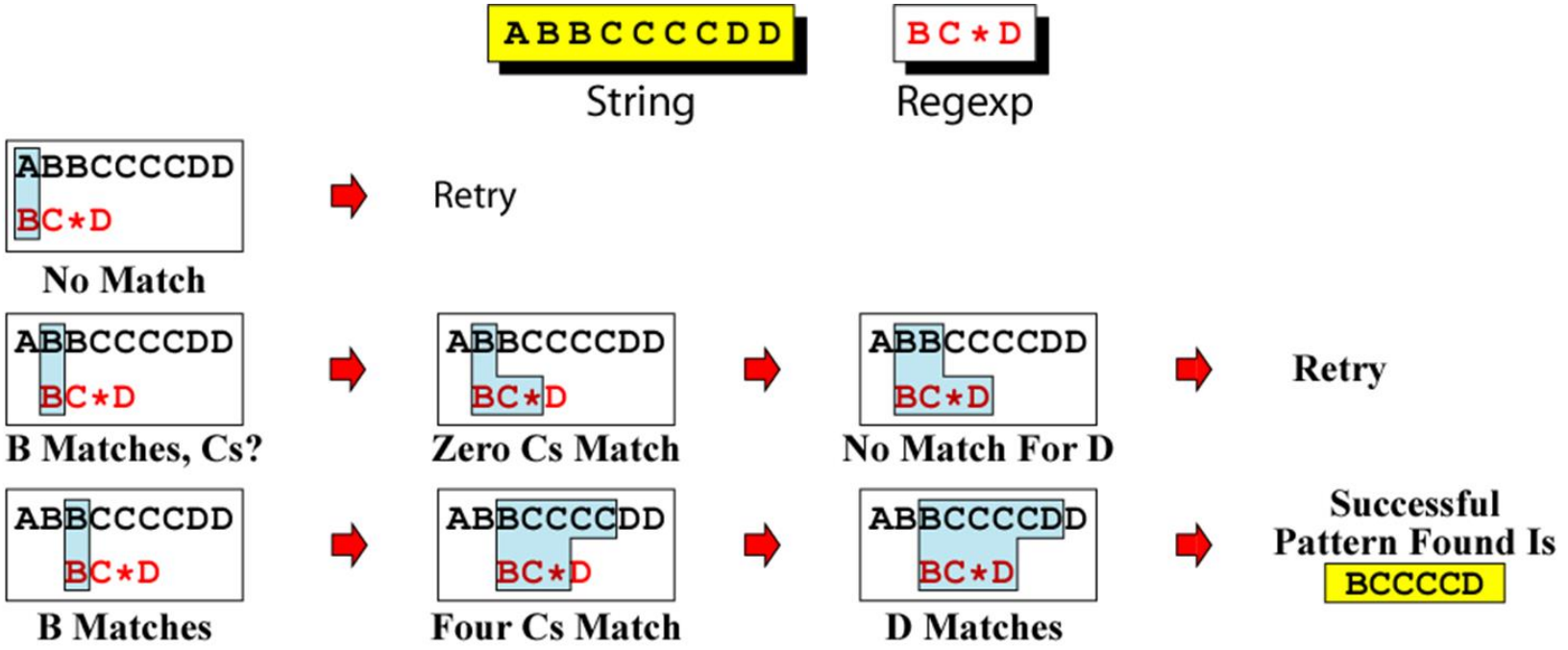
Μορφές

`\{m\}`	Αντιστοιχεί προηγούμενα άτομα ακριβώς m φορές
`\{m, \}`	Αντιστοιχεί προηγούμενα άτομα m ή περισσότερες φορές
`\{, n\}`	Αντιστοιχεί προηγούμενα άτομα n ή λιγότερες φορές

Παραδείγματα

`CA\{5\}`	`CAAAAA`
`CA\{3,\}`	`CAAA, CAAAA, CAAAAA, …`
`CA\{,2\}`	`C, CA, CAA`

Επαναλαμβανόμενη αντιστοίχιση προτύπων



Τελεστής ομαδοποίησης (Group Operator)

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.

Κοινή έκφραση	**Αντιστοιχία**
`A(BC)\{3\}`	ABCBCBC
`(F(BC)\{2\}G)\{2\}`	FBCBCGFBCBCG

Κανόνες δημιουργίας & αντιστοίχισης κανονικών εκφράσεων

- Οι χαρακτήρες : **.** ,*, **[** ,**** είναι ειδικοί χαρακτήρες εκτός όταν εμφανίζονται ανάμεσα σε [ ].

- Ο χαρακτήρας **^** είναι ειδικός χαρακτήρας όταν είναι ο πρώτος που εμφανίζεται σε μια κανονική έκφραση και όταν είναι ο πρώτος χαρακτήρας μέσα σε [ ].

- Ο χαρακτήρας **$** είναι ειδικός χαρακτήρας όταν είναι τελευταίος.

- Ένας ειδικός χαρακτήρας που έχει μπροστά **** είναι ο ίδιος ο χαρακτήρας (escape chars).

Πίνακας συμβολισμού κανονικών εκφράσεων

Συμβολισμός	Σημασία
c	Αντιστοιχεί στον χαρακτήρα c
\c	Αναγκάζει το c να διαβάσει μόνον ως το γράμμα c
^	Ταυτοποιείται κατά την αναζήτηση με την αρχή της γραμμής
$	Ταυτοποιείται κατά την αναζήτηση με το τέλος της γραμμής
.	Ταυτοποιείται με ένα και μόνο χαρακτήρα
[xy]	Κάθε μεμονωμένος χαρακτήρας που περιέχεται στο σύνολο
[^xy]	Κάθε μεμονωμένος χαρακτήρας που δεν περιέχεται στο σύνολο (exclusion set)

Πίνακας συμβολισμού κανονικών εκφράσεων (συνέχεια)

Συμβολισμός	Σημασία
c*	Καμία ή περισσότερες εμφανίσεις του χαρακτήρα c
*	0 ή περισσότερες εμφανίσεις της έκφρασης που προηγείται
+	1 ή περισσότερες εμφανίσεις της έκφρασης που προηγείται
?	0 ή 1 εμφανίσεις της έκφρασης που προηγείται

Παραδείγματα

- Όλες οι γραμμές που δεν ξεκινούν από κεφαλαίο αγγλικό χαρακτήρα :

‘^[^A-Z]'

- Όλες οι γραμμές που περιέχουν !,& ,* :

'([\!*\&])‘

- Όλες οι γραμμές που περιέχουν την τιμή $1.99 :

'\$1\.99‘

- Όλες οι γραμμές με μήκος 2 χαρακτήρες :

'^..$‘ ή ^.\{2\}$

Παραδείγματα

- Όλες οι γραμμές που έχουν μήκος ακριβώς 17 χαρακτήρες:

'^.\{17\}$'

- Όλες οι γραμμές που έχουν μήκος τουλάχιστον 25 χαρακτήρες:

'^.\{25,\}$'

Παραδείγματα

- Όλες οι γραμμές που ξεκινούν με * :
  - **^***
- Όλες οι γραμμές που δεν περιέχουν αριθμούς :
  - **^[^0–9]*$**
- Όλες οι γραμμές που περιέχουν τα έτη 1991 έως 1995 :
  - **199[1-5]**
- Οποιαδήποτε ακολουθία χαρακτήρων δεν περιέχει ψηφία:
  - **[A-Za-z][A-Za-z]***

Παραδείγματα

- Οποιοσδήποτε προσημασμένος ακέραιος :
  - **[+\-][0-9][0-9]***

- Οποιαδήποτε ακολουθία χαρακτήρων :
  - **.* (ιδιωματισμός!)**

- Οποιοδήποτε αναγνωριστικό (identifier) :
  - **[a-zA-Z_][a-zA-Z_0-9]***

- Οποιοσδήποτε πραγματικός αριθμός χωρίς πρόσημο:
  - **[0-9]+[.][0-9]+**
  - **[0-9]*[.][0-9]***

Παραδείγματα

- Οποιοδήποτε πλήθος από * ακολουθούμενο από οποιοδήποτε πλήθος \ :
  - ‘***’

- Ώρα σε 12ωρη βάση:
  - ‘1*[012]*[1-9]*:[0-5][0-9]’

Κανονική έκφραση για ημερομηνία

Υποθέτουμε ότι οι ημερομηνίες είναι στη μορφή : Feb 9, 1999 .

Ποια είναι η μορφή της κανονικής έκφρασης;

‘[A-Z][a-z]* [0-9]\{1,2\}, [0-9]\{4\}’

‘[A-Z][a-z]\{3,\} [0-9]\{1,\}, [0-9]\{4\}’

Κανονική έκφραση για τηλέφωνο

- Ποια είναι η μορφή της κανονικής έκφρασης για τηλεφωνικούς αριθμούς της περιοχής Θεσσαλονίκης; (πρόθεμα : 2310, 2391,2392, …, 2396 και αριθμός εξαψήφιος στη μορφή :

  23XX-XXX-XXX)

  - **‘23[19][0-6]-[0-9]\{3\}-[0-9]\{3\}’**

Επεκτεταμένες RE

- Είναι υπερσύνολο της grep. Αναζητεί και αντιστοιχεί όλες τις περιπτώσεις κανονικών εκφράσεων της grep και επιπλέον :
  - Μια RE που ακολουθείται από το **+** αντιστοιχεί μία ή περισσότερες ταυτίσεις της RE.
  - Μια RE που ακολουθείται από το **?** αντιστοιχεί καμία ή μία ταυτίσεις της RE.
  - Πολλαπλές RE διαχωριζόμενες με **|** αντιστοιχούνται με οποιαδήποτε από τις RE που περιέχονται.
  - Μια RE μπορεί να περικλείεται σε παρενθέσεις για ομαδοποίηση.

Παραδείγματα επεκτεταμένων εκφράσεων

- Παραδείγματα :
  - ’19|23’
  - ‘[A-Z][A-Z]+’
  - ‘239(4|6)-(0|1)
  - ‘(Sally|Fred) (Smith|Jones|Parker)

Παραδείγματα επεκτεταμένων

όλες οι γραμμές που ξεκινούν με
'(^#)'

όλες οι γραμμές που ΔΕΝ ξεκινούν με
'(^[^#])'

όλες οι γραμμές που ξεκινούν με e, f ή g.
'(^[efg])'

όλες οι γραμμές που ξεκινούν με κεφαλαίο γράμμα
'(^[A-Z])'

Επεκτεταμένες RE

```
# όλες οι γραμμές που ΔΕΝ ξεκινούν με κεφαλαίο γράμμα
 '(^[^A-Z])'

# όλες οι γραμμές που περιέχουν ! * &
 '([\!\*\&])'

# όλες οι γραμμές που περιέχουν ! * & αλλά δεν ξεκινούν με #
 '([^#][\!\*\&])'
```

Αποθήκευση (save)

- Ο τελεστής αποθήκευσης \(\) αντιγράφει το αλφαριθμητικό που περικλείεται σε μια προσωρινή μνήμη (buffer ή register) για μεταγενέστερη χρήση. Υπάρχουν συνολικά εννέα (9) buffers που αριθμούνται από \1 έως \9.

- \1 : είναι το τμήμα κειμένου που αντιστοιχεί στην πρώτη κανονική έκφραση σε παρένθεση.

- \2 : είναι το τμήμα κειμένου που αντιστοιχεί στην δεύτερη κανονική έκφραση σε παρένθεση.

- …..

Παραδείγματα

- '^\([a-z]\)\1'
  - Αναζητεί και εμφανίζει τις γραμμές που ξεκινούν από ένα ζεύγος όμοιων γραμμάτων.

- '\([a-z][a-z]*\)\1\1'
  - Αναζητεί και εμφανίζει τις γραμμές που περιέχουν τουλάχιστον τρία αντίγραφα ενός συνόλου χαρακτήρων που αποτελείται από πεζά γράμματα.

Παραδείγματα

Bill\(.\)Gates\1

Αναζητεί ένα πρότυπο αλφαριθμητικού που ξεκινά με τη λέξη **Bill**, ακολουθούμενη από ένα χαρακτήρα (πλην του χαρακτήρα αλλαγής γραμμής), ακολουθούμενο από τη λέξη **Gates**, ακολουθούμενη από τον ίδιο ένα χαρακτήρα.

- Η παρακάτω ακολουθία αντιστοιχεί στο πρότυπο:
  Bill!Gates! **Bill-Gates-**

αλλά όχι η ακολουθία:

Bill?Gates! **Bill-Gates**

Παραδείγματα

“a\(.\)b\(.\)c\2d\1”

- Το παράδειγμα αφορά ένα πρότυπο που ξεκινά με **a**, ένα χαρακτήρα (#1), ακολουθούμενο από το **b**, ακολουθούμενο από έναν άλλο χαρακτήρα (#2), **c**, το χαρακτήρα #2, **d**, και το χαρακτήρα #1.

- Η παρακάτω ακολουθία αντιστοιχεί στο πρότυπο:

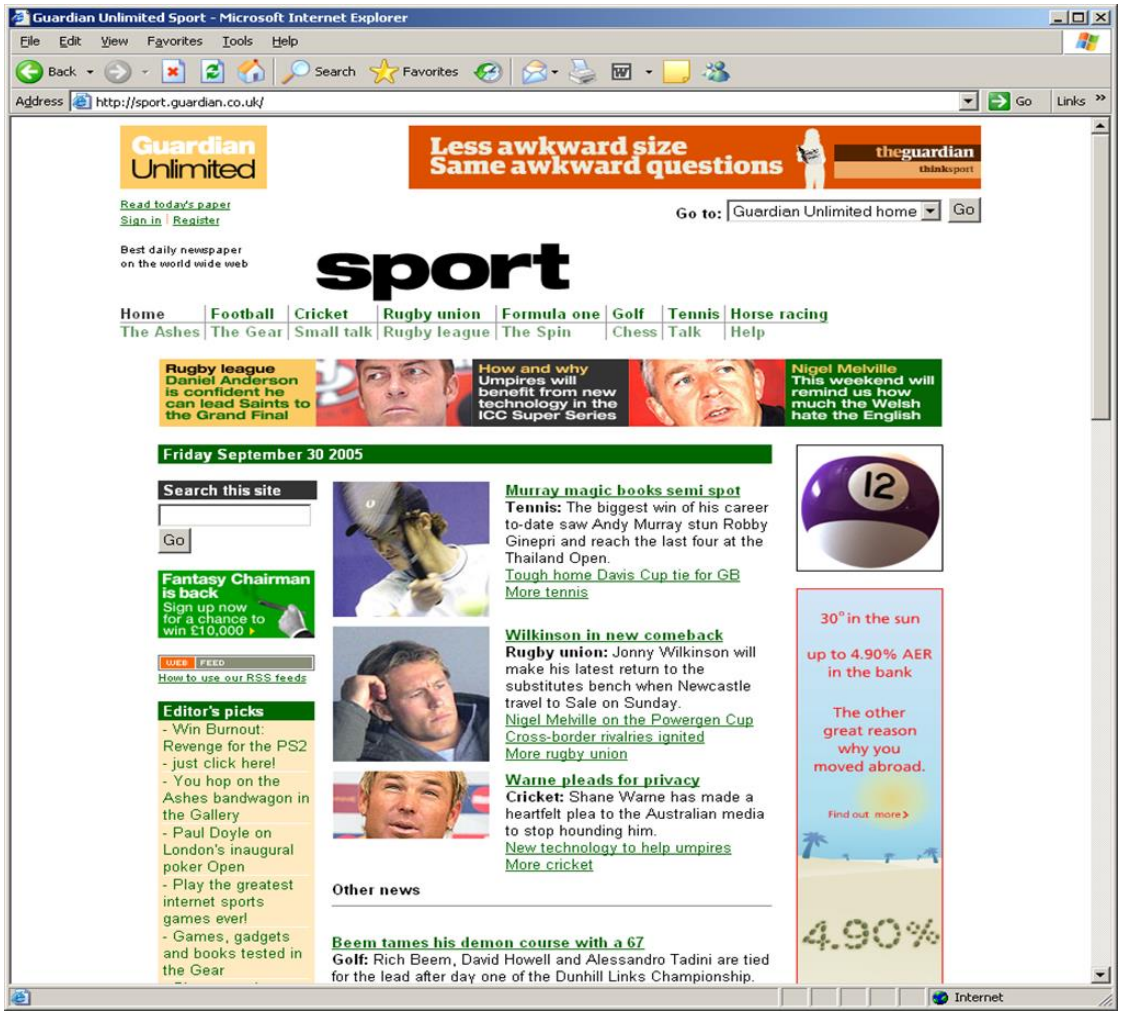
  a-b!c!d-

Ένα παράδειγμα χρήσης RE

Ας υποθέσουμε ότι θέλετε να δημιουργήσετε μια web-σελίδα που θα περιέχει πάντα τους τελευταίους τίτλους αθλημάτων που συλλέχθηκαν από πολλές ιστοσελίδες εφημερίδων.

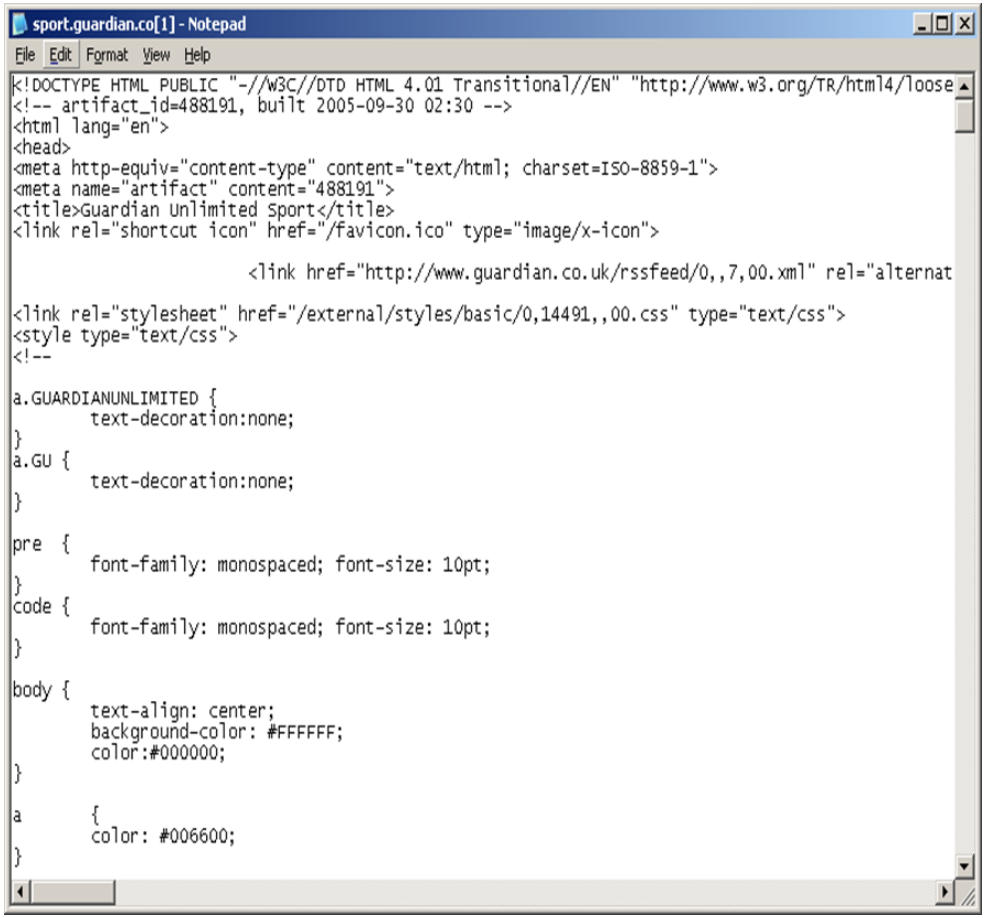
Μπορεί, για παράδειγμα, να θέλετε να συμπεριλάβετε το Guardian’s sports στη σελίδα σας.

Ένα παράδειγμα χρήσης RE



Προσθήκη αυτών των τίτλων χωρίς αυτόματο τρόπο

Θα έπρεπε να είχατε πρόσβαση στην πηγή της σελίδας.

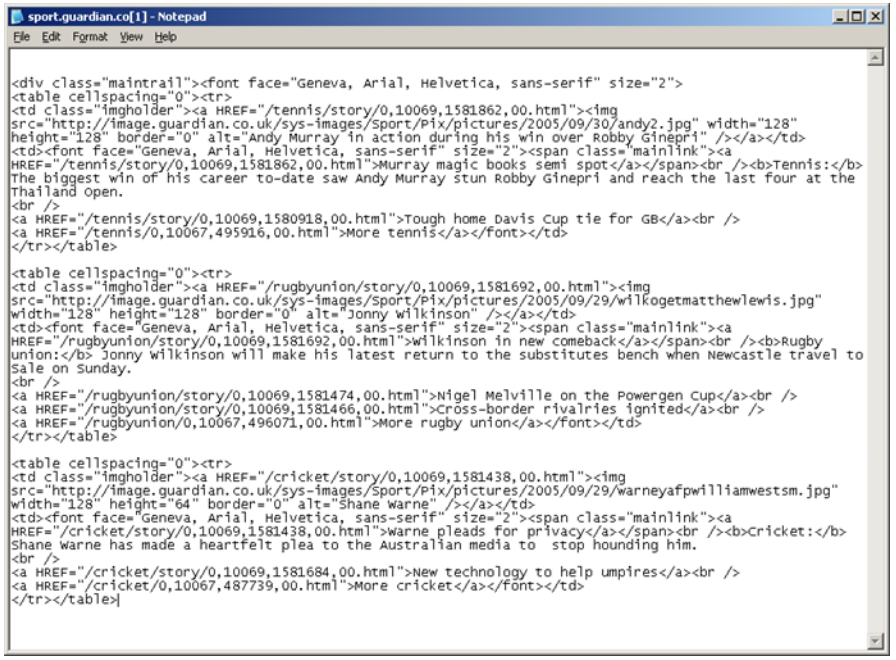


Προσθήκη αυτών των τίτλων χωρίς αυτόματο τρόπο

Θα πρέπει βρείτε το κείμενο
το οποίο καθορίζει τους τίτλους .

Να το αναλύσετε

Και να αντιγράψετε το σχετικό
bits στο HTML για τη δική
σας ιστοσελίδα.



Πίνακας πρωτοσέλιδων της προηγούμενης σελίδας

Εξετάζοντας το, διαπιστώνουμε ότι η πηγή περιέχει ένα HTML πίνακα για κάθε άθλημα στη λίστα των κορυφαίων ιστοριών.
Εδώ είναι ο πίνακας για τα πρωτοσέλιδα του τένις στη σελίδα που είδατε νωρίτερα :

```
<table cellspacing="0"><tr>
<td class="imgholder"><a HREF="/tennis/story/0,10069,1581862,00.html"><img
src="http://image.guardian.co.uk/sys-images/Sport/Pix/pictures/2005/09/30/andy2.jpg"
width="128" height="128" border="0" alt="Andy Murray in action during his win over
Robby Ginepri" /></a></td>
<td><font face="Geneva, Arial, Helvetica, sans-serif" size="2"><span
class="mainlink"><a HREF="/tennis/story/0,10069,1581862,00.html">Murray magic
books semi spot</a></span><br /><b>Tennis:</b> The biggest win of his career to-date
saw Andy Murray stun Robby Ginepri and reach the last four at the Thailand Open.
<br />
<a HREF="/tennis/story/0,10069,1580918,00.html">Tough home Davis Cup tie for
GB</a><br />
<a HREF="/tennis/0,10067,495916,00.html">More tennis</a></font></td>
</tr></table>
```

Κείμενο καθορισμού κύριου τίτλου προηγούμενης σελίδας

Εδώ είναι το κείμενο που καθορίζει τον κύριο τίτλο του τένις στη σελίδα που είδατε νωρίτερα :

```
<span class="mainlink">
<a HREF="/tennis/story/0,10069,1581862,00.html">
Murray magic books semi spot
</a></span>
<br />
<b>Tennis:</b> The biggest win of his career to-date saw Andy Murray stun Robby Ginepri and reach the last four at the Thailand Open
```

Πως θα το κάνετε εσείς

Για να αποκτήσετε αυτή την ιστορία στη δική σας ιστοσελίδα, μπορείτε να αντιγράψτε το σχετικό τμήμα HTML στην πηγή του κώδικα για την ιστοσελίδα σας.

Αλλά …

... το χειροκίνητο αυτό είναι πολύ μεγάλη εργασίας

Θα έπρεπε να αυτοματοποιήσουμε την πλήρη εργασία.

Αυτόματη πρόσθεση τίτλων

Για να προσθέσετε αυτόματα τίτλους, πρέπει να γράψετε ένα πρόγραμμα που θα το κάνει :

- Κατεβάστε τον πηγαίο κώδικα για την σελίδα Guardian.
- Ανάλυση αυτόν τον πηγαίο κώδικα για να εξαγάγετε το κατάλληλο κείμενο.
- Προσθέστε το σχετικό κείμενο του πηγαίο κώδικα για τη δική σας ιστοσελίδα.

Αυτόματη πρόσθεση τίτλων

- ✓ Αργότερα, θα δούμε πώς να κατεβάσετε πηγές από άλλους ιστότοπους.
- ✓ Τώρα, θα επικεντρωθούμε στο θέμα της ανάλυσης του κειμένου.

Κανονικές εκφράσεις

Η τεχνολογία κανονικής έκφρασης παρέχει ένα βολικό τρόπο αναζήτησης συμβολοσειρών για ενδιαφέρον πρότυπα.

Κανονικές εκφράσεις (συνέχεια)

Παράδειγμα κανονικής έκφρασης :

/ab*c/

Αυτό αναζητά τη συμβολοσειρά string για τις υποκλάσεις που συγκρίνουμε

«Ένα a που ακολουθείται από μηδέν ή ένα b ακολουθείται από το c»

Θα ταίριαζε με οποιοδήποτε από τις παρακάτω υποκατηγορίες:

ac
abc
abbc
abbbc
….

Χρησιμοποιώντας κανονικές εκφράσεις στην PHP

Οι κανονικές εκφράσεις υποστηρίζονται σε αρκετές γλώσσες, συμπεριλαμβανομένης της PHP.

- Η PHP παρέχει μια ομάδα προκαθορισμένων λειτουργιών για τη χρήση τους.

- Για τώρα, θα επικεντρωθούμε σε ένα μόνο από αυτά, τη συνάρτηση preg_replace.

Η συνάρτηση preg_replace

Μορφή κλήσης:

preg_replace (regexp, replacement, subject [, int limit])

Αυτή η συνάρτηση επιστρέφει το αποτέλεσμα της αντικατάστασης των substrings στο θέμα που ταιριάζει με την αντικατάσταση της regexp.

- Ο αριθμός των substrings τα οποία αντικαθίστανται ελέγχεται από προαιρετικό όριο παραμέτρου.
- Το παράδειγμα εφαρμογής βρίσκεται στην επόμενη διαφάνεια.

Κανονικές εκφράσεις (συνέχεια)

Κώδικας PHP :

```
<?php
$myString = "xyzacklmabbcpqrabbbbbcstu";
echo "myString is $myString <br>";
$myString = preg_replace("/ab*c/","_",$myString);
echo "myString is now $myString";
?>
```

Με έξοδο :

myString is xyzacklmabbcpqrabbbbbcstu
myString is now xyz_klm_pqr_stu

Χρησιμοποιώντας την οριακή παράμετρο στο preg_replac

Κώδικας PHP :

```
<?php
$myString = "xyzacklmabbcpqrabbbbbcstu";
echo "myString is $myString <br>";
$myString = preg_replace("/ab*c/","_",$myString,1);
echo "myString is now $myString";
?>
```

Με έξοδο :

myString is xyzacklmabbcpqrabbbbbcstu
myString is now xyz_klmabbcpqrabbbbbcstu

Μετα-χαρακτήρες

Έχουμε δει ότι ορισμένοι χαρακτήρες έχουν μια ειδική σημασία στις κανονικές εκφράσεις:

Το παράδειγμα στις τελευταίες διαφάνειες χρησιμοποίησε το * χαρακτήρα πράγμα που σημαίνει :

«0 ή περισσότερες παρουσίες του προηγούμενου χαρακτήρα ή μοτίβου»

Αυτοί ονομάζονται μετα-χαρακτήρες.

Άλλοι μετα-χαρακτήρες εμφανίζονται στην επόμενη διαφάνεια.

Μετα-χαρακτήρες

Ο χαρακτήρας ***** που σημαίνει ‘0 ή περισσότερες παρουσίες των προηγουμένων’.

Ο χαρακτήρας **+,** που σημαίνει ‘1 ή περισσότερες παρουσίες των προηγουμένων’.

Ο χαρακτήρας **?**, που σημαίνει ‘0 ή 1 παρουσία των προηγούμενων’.

Ο χαρακτήρας **{** and **}** **,** οριοθετεί μια έκφραση καθορίζοντας μια σειρά από αποδεκτές εμφανίσεις του προηγούμενου χαρακτήρα.

Παραδείγματα:

{m} σημαίνει ακριβώς **m** εμφανίσεις των προηγουμένων χαρακτήρων / pattern
{m,} σημαίνει τουλάχιστον **m** εμφανίσεις των προηγουμένων char / pattern
{m, n} σημαίνει τουλάχιστον **m**, αλλά όχι περισσότερο από **n**, εμφανίσεις των προηγούμενων char/pattern

Μετα-χαρακτήρες

Περαιτέρω μετα-χαρακτήρες είναι:

Ο χαρακτήρας **^** , που ταιριάζει με την έναρξη μιας σειράς

Ο χαρακτήρας $, που ταιριάζει με το τέλος μιας σειράς

Ο χαρακτήρας **.** , που αντιστοιχεί σε οτιδήποτε εκτός από έναν χαρακτήρα νέας γραμμής.

Ο χαρακτήρας **[** and **]**, ξεκινά μια τάξη ισοδυναμίας των χαρακτήρων, κάποιο από το οποίο μπορεί να ταιριάζει με έναν χαρακτήρα στη συμβολοσειρά στόχου.

Ο χαρακτήρας **(** and **)**, οριοθετεί μια ομάδα sub-patterns.

Ο χαρακτήρας **|**, χωρίζει εναλλακτικά πρότυπα.

Κανονικές εκφράσεις (συνέχεια)

Παράδειγμα έκφρασης:

/^a.*d$/

Αυτό αντιστοιχεί σε ολόκληρη τη συμβολοσειρά που παρέχει τον στόχο Συμβολοσειράς, ξεκινά με ένα a, που ακολουθείται από μηδέν ή περισσότερους ίδιας γραμμής χαρακτήρες, και τελειώνει με ένα d.

Ένα παράδειγμα βρίσκεται στην επόμενη διαφάνεια.

Παράδειγμα κώδικα PHP

```
<?php
$myString1 = ”abcdefghijklmnopqrstuvd";
echo "myString1 is $myString1 <br>";
$myString1 = preg_replace(”/^a.*d$/","_",$myString1);
echo "myString1 is now $myString1 <br>";
$myString2 = ”xabcdefghijklmnopqrstuvd";
echo "myString2 is $myString2 <br>";
$myString2 = preg_replace(”/^a.*d$/","_",$myString2);
echo "myString2 is now $myString2";
?>
```

Resultant output is

```
myString1 is abcdefghijklmnopqrstuvd
myString1 is now _
myString2 is xabcdefghijklmnopqrstuvd
myString2 is now xabcdefghijklmnopqrstuvd
```

Κανονικές εκφράσεις (συνέχεια)

Παρόλο που ορισμένοι χαρακτήρες έχουν ιδιαίτερες έννοιες στις κανονικές εκφράσεις, μπορούμε, μερικές φορές, να θέλουμε να τις χρησιμοποιήσουμε για να ταιριάζουν με το string.
Το κάνουμε αυτό για να αποφύγουμε την κανονική έκφραση, στην οποία προηγείται μια ανάστροφη κάθετος ****

Παράδειγμα κανονικής έκφρασης:

/^a\^+.*d$/

Αυτό αντιστοιχεί σε ολόκληρη τη συμβολοσειρά στόχου, υπό τον όρο ότι ο στόχος συμβολοσειρά ξεκινά με ένα a, που ακολουθείται από έναν ή περισσότερους χαρακτήρες, που ακολουθείται από μηδέν ή περισσότερους ίδιας σειράς χαρακτήρες και τελειώνει με ένα d.

Ένα παράδειγμα βρίσκεται στην επόμενη διαφάνεια.

Κανονικές εκφράσεις (συνέχεια)

Όπως αναφέρθηκε προηγουμένως, οι **[** and **]** χαρακτήρες έχουν μια ιδιαίτερη σημασία σε κανονικές εκφράσεις οριοθετούν μια ισοδυναμία χαρακτήρων, οποιαδήποτε από τις οποίες μπορεί να χρησιμοποιείται για να ταιριάζει με ένα χαρακτήρα στη συμβολοσειρά στόχου.

Παράδειγμα κανονικής έκφρασης:

/a[KLM]b/

Αντικαθιστά οποιαδήποτε υποσυμβολοσειρά που περιλαμβάνει «το γράμμα a που ακολουθείται από ένα από τα τρία γράμματα KLM, ακολουθούμενο από το γράμμα b» .

Κανονικές εκφράσεις (συνέχεια)

Ο χαρακτήρας **^** έχει ιδιαίτερη σημασία, όταν χρησιμοποιείται ως πρώτος χαρακτήρας μεταξύ **[** and **]** χαρακτήρων; αυτή η έννοια είναι διαφορετική από την ειδική σημασία του όταν χρησιμοποιείται εκτός του **[** and **]** χαρακτήρα.

Όταν χρησιμοποιείται ως πρώτος χαρακτήρα μεταξύ των **[** and **]** χαρακτήρων, ο χαρακτήρας **^** προσδιορίζει το συμπλήρωμα της ισοδυναμίας της κλάσης που θα είχε διευκρινιστεί αν δεν υπήρχε.

Παράδειγμα κανονικής έκφρασης:

/a[^KLM]b/

Αντικαθιστά οποιαδήποτε υποσυμβολοσειρά που περιλαμβάνει «το γράμμα a που ακολουθείται από ένα μόνο γράμμα που δεν είναι ένα από KLM, που ακολουθείται από το γράμμα b».

Κανονικές εκφράσεις (συνέχεια)

Ο χαρακτήρας **-** έχει επίσης ιδιαίτερη σημασία, όταν χρησιμοποιείται μεταξύ **[** and **]** χαρακτήρων.

Χρησιμοποιείται για να ενώσει την αρχή και το τέλος μιας ακολουθίας χαρακτήρων, ο καθένας από τα οποία μπορεί να χρησιμοποιηθεί για να ταιριάζει με έναν χαρακτήρα στη συμβολοσειρά στόχου.

Παράδειγμα κανονικής έκφρασης:

/a[0-9]b/

Ταιριάζει με οποιαδήποτε υποσυμβολοσειρά που περιλαμβάνει «το γράμμα a που ακολουθείται κατά ένα ψηφίο, ακολουθούμενο από το γράμμα b.

Κανονικές εκφράσεις (συνέχεια)

Ορισμένες ακολουθίες διαφυγής έχουν επίσης ιδιαίτερη σημασία στα κανονικά εκφράσεις. Καθορίζουν ορισμένες κοινώς χρησιμοποιούμενες κλάσεις ισοδυναμίας χαρακτήρων:

\w είναι ισοδύναμο με [a-zA-Z0-9_]
\W είναι ισοδύναμο με [^a-zA-Z0-9_]
\d είναι ισοδύναμο με [0-9]
\D είναι ισοδύναμες με [^ 0-9]
\s είναι ισοδύναμο με [\ n \ t \ f \ r]
\S είναι ισοδύναμο με [^ \ n \ t \ f \ r]
\b υποδηλώνει ένα όριο λέξης
\B υποδηλώνει ένα όριο όχι λέξης

Σημειώστε τους χαρακτήρες SP κατά την έννοια του \ s και \ S, που είναι το λευκό- η ισοδυναμία χώρου περιλαμβάνει SP.
Το \ f είναι formFeed και το \ r είναι carriageReturn .

Κανονικές εκφράσεις (συνέχεια)

Παράδειγμα κανονικής έκφρασης:

/ %\d\d\d\D/

Αντιστοιχεί σε οποιοδήποτε υποσυμβολοσειρά που περιλαμβάνει ‘ένα % ακολουθούμενο από τρία δεκαδικά ψηφία, ακολουθούμενα από ένα μη ψηφίο’.

Παράδειγμα κανονικής έκφρασης:

/ \s\w\w\s/

Αντιστοιχεί σε οποιοδήποτε υποσύνολο περιλαμβάνει ‘ένα λευκό χώρο χαρακτήρα, ακολουθούμενο από δύο χαρακτήρες λέξεων, που ακολουθούνται από ένα άλλο χαρακτήρα’.

Κανονικές εκφράσεις (συνέχεια)

Κώδικας PHP :

```
<?php
$myString = ”This is not an apple";
echo "myString is $myString <br>";
$myString = preg_replace("/\s\w\w\s/","_",$myString);
echo "myString is now $myString";
?>
```

Με έξοδο:

myString is This is not an apple
myString is now This_not_apple

Παράδειγμα RE

Ας υποθέσουμε ότι έχουμε την παρακάτω HTML

wine
f12
cheese

Ας υποθέσουμε ότι θέλουμε να απαλλαγούμε από τη λίστα και οποιοδήποτε στοιχείο της λίστας του οποίου το περιεχόμενο περιλαμβάνει μόνο μη ψηφία.

Δηλαδή, θέλουμε να γίνει η HTML

f12

Παράδειγμα RE

```
<?php
$myString = ”<ul><li>wine</li><li>f12</li><li>cheese</li></ul>";
echo "myString is $myString <br>";
$myString = preg_replace(”/<li>\D+<\/li>/","",$myString);
echo "myString is now $myString <br>";
?>
```

Με έξοδο:

myString is
wine
f12
cheese

myString is now
f12

Βλέποντας το raw-HTML

Υποθέστε ότι θέλουμε να δούμε ακατέργαστη HTML στην έξοδό μας.

Δηλαδή, ας υποθέσουμε ότι θέλαμε να δούμε :

myString is wine
f12
cheese
myString is now
f12

Θα πρέπει να αντικαταστήσετε όλες τις εμφανίσεις του < with <

Θα μπορούσαμε να χρησιμοποιήσουμε τις κανονικές εκφράσεις γι 'αυτό, αλλά, η συμβολοσειρά που πρόκειται να αντικατασταθεί είναι μια σταθερά.

Έτσι μπορούμε να χρησιμοποιήσουμε μια απλούστερη τεχνολογία.

Βλέποντας το raw-HTML

```
<?php
$myString = ”<ul><li>wine</li><li>f12</li><li>cheese</li></ul>";
echo "myString is ".str_replace(“<“,”&lt;”,$myString).”<br>";
$myString = preg_replace("/<li>\D+<\/li>/",”x",$myString);
echo "myString is now ".str_replace(“<“,”&lt;”,$myString);
?>
```

Με έξοδο:

myString is wine
f12
cheese
myString is now f12

Κανονικές εκφράσεις (συνέχεια) - θυμόμαστε τους υποπληθυσμούς

Όταν ένα <pattern> συνδυάζεται με μια σειρά στόχων, τα substrings που ταιριάζουν με τα sub-patterns μπορούν να θυμούνται και να επαναχρησιμοποιούν αργότερα στο ίδιο pattern.

Τα sub-patterns τα οποία πρέπει να ταιριάζουν με τα substrings που θυμόμαστε, περιλαμβάνονται στις παρενθέσεις.

Τα sub-patterns είναι αριθμημένα σιωπηρώς, ξεκινώντας από το 1 και οι αντίστοιχες υποκατηγορίες τους μπορούν να συνεχίσουν να επαναχρησιμοποιούνται αργότερα.

Το pattern χρησιμοποιεί back-references ς όπως \ 1 ή \ 2 ή \ 3

Ωστόσο, για να πάρετε την ανάστροφη κάθετο, πρέπει να το ξεπεράσετε, έτσι πρέπει να πληκτρολογήσετε \\ 1 ή \\ 2 ή \\ 3 σε τυπικές εκφράσεις.

Χρησιμοποιώντας τα back-references

```
<?php
$myString1 = ”klmAklmAAklmABklmBklmBBklm";
echo "myString is $myString <br>";
$myString1 = preg_replace(”/([A-Z])\\1/",”_",$myString1);
echo "myString1 is now $myString1 ";
?>
```

Με έξοδο:

myString1 is klmAklmAAklmABklmBklmBBklm
myString1 is now klmAklm_klmABklmBklm_klm