

## Forelæsning 6

### Streng-matching og indhentning af data fra fremmede web-sites

- Regulære udtryk (mønstre)
- Tchkommandoen `REGEXP`
- Bill Gates Personal Wealth Clock
- Valutahandel — check af form variable

### Syntaks for regulære udtryk, del 1

Et mønster  $m$  kan blandt andet tage følgende former:

- $c$  matches af ethvert tegn
- $m_1m_2$  matches af tegnet  $c$ ; tegnet  $\backslash$ .
- $m^*$  sekventiel match af  $m_1$  og  $m_2$ . Eks: strengen 'abc' matcher mønsteret 'a.c'
- $m^+$  matches af 0 eller flere efterfølgende forekomster af tegnsekvenser som matcher mønsteret  $m$ . Eks: strengene 'abbbbbb' og 'aa' matcher mønsteret 'ab^a'
- $m?$  matches af strenge som matcher  $m$ . Eks: strengen 'cababc' matcher mønsteret 'c(ab)^c'
- $m^+$  matcher et eller flere efterfølgende forekomster af tegnsekvenser som matcher mønsteret  $m$ . Eks: mønsteret 'ca+b' matches af strengen 'caab' men ikke af strengen 'cb'
- $m?$  matches af 0 eller 1 forekomst af tegnsekvenser som matcher mønsteret  $m$ . Eks: strengene 'abc' og 'ab' matcher begge mønsteret 'abc?'

### Regulære udtryk (mønstre)

Sprog til at beskrive strenge der har en bestemt form, tilfældes.

To vigtige brug af mønstre:

- Sikring af at data fra brugere har den rigtige form
  - hvis et tal forventes i en HTML-form må web-programmet ikke godtage andre tegn end talcifre
- Indhentning af data fra fremmede web-sites
  - dollarkursen
  - dagens nyheder fra Reuters
  - vejret
  - ...
  - XML

### Syntaks for regulære udtryk, del 2

Et mønster  $m$  kan blandt andet tage følgende former:

- $m_1 | m_2$  matches af tegnsekvenser som matcher enten  $m_1$  eller  $m_2$ . Eks: mønsteret '(gris|ko)' matches af tegnsekvensen 'gris' og tegnsekvensen 'ko'
- [...] matches af tegn i klassen. Eks: mønsteret '[abc1-4]' matches af tegnsekvenser bestående af tegnene a, b, c, 1, 2, 3, 4
- [^...] matches af alle andre tegn end dem i klassen. Eks: mønsteret '[abc1-4]' matches af tegnsekvenser bestående af alle tegn bortset fra a, b, c, 1, 2, 3, 4. Hatten '^' betyder altså, alt andet end de tegn der følger

### Eksempler på mønstre

- `[A-Za-zÆØÅæøå]` : matcher alle tegn i det danske alfabet
  - `[0-9][0-9]` : matcher to cifrede tal der kan starte med 0
  - `(ko|gris)` : matcher de to strenge ko og gris
  - `((a|b)a)*` : matcher aa, ba, aaaa, baaa, ...
  - `(0|1)+` : matcher binære tal, dvs 0, 1, 01, 11, 011101010, ...
  - . . . : matcher to vilkårlige tegn.
  - `([1-9][0-9]+)/( [1-9][0-9]+)` : matcher helkatsbrøker, f.eks. 1/8, 32/5645, 45/6, ...
- Matches strengen 012/54 af mønstret?
- `<html>.*</html>` : matcher en HTML side.
  - `www\.\(((it-cl|itu)\.dk)|(\.it\.edu))` : matcher `www.itu.dk`, `www.it-c.dk` og `www.it.edu`.
  - `http://hug.it.edu:8034/oplevelse2/(.*)\.\.ccl` : matcher alle Tol filer på hug i katalog `oplevelse2` for den bruger der anvender portnummer 8034.

### Eksempler på brug af regex-kommandoen

#### Eksempel: Telefonnummer

```
% set s "Name: Hans Hansen Tlf: 66 66 66 66"
Name: Hans Hansen Tlf: 66 66 66 66
% regex {Name: ([a-zA-Z ]+) Tlf: ([0-9 ]+)} %s match name tlf
1
% set match
Name: Hans Hansen Tlf: 66 66 66 66
% set name
Hans Hansen
% set tlf
66 66 66 66
%
```

#### Eksempel: Email

```
proc valid_email { e } {
  set pattern {^[a-zA-Z][0-9a-zA-Z\._]*@[0-9a-zA-Z\._]+}$}
  return [regex $pattern $e]
}
% valid_email name@company.com
1
% valid_email name@company@com
0
```

Opgave: Klassificer mønstret ovenfor strenge som ikke er email-adresser?

### Tcl-kommandoen regex

Kan bruges til at matche en streng mod et mønster:

```
regex pattern string ?match ?arg1 ... ?argN
```

Kommandoen returnerer 1 hvis der findes en delstreng i strengen *string* som matcher mønstret *pattern*. Ellers returneres 0.

Hvis *pattern* startes med `"` må ingen tegn forekomme før delstrengen i *string*.

Tilsvarende, hvis *pattern* sluttes med `$` må ingen tegn forekomme efter delstrengen i *string*.

Eksempel:

```
% regex {[0-9]+} "aa99AA"
1
% regex {^[0-9]+$} "aa99AA"
0
% regex {[0-9]+} "99AA"
1
% regex {[0-9]+$} "99AA"
0
% regex {^[0-9]+$} "aa99"
0
```

Hvis et variabelnavn gives for *?match* bindes variabelen til delstrengen som matcher mønstret.

Hvis variabelnavne gives for *arg1 ... argN* bindes variablerne til delstrenge matchende delmønstre givet i parenteser i *pattern*.

### Flere eksempler på brug af regex-kommandoen

#### Eksempel: URL (Http)

```
% set pattern {^http://[a-zA-Z\._]+(:[0-9]+)?(/[0-9a-zA-Z\._]*)?$}
^http://[a-zA-Z\._]+(:[0-9]+)?(/[0-9a-zA-Z\._]*)?$
% regex $pattern http://www.it.edu
1
% regex $pattern http://hug.itu.dk:8077/oplevelse5/
1
% regex $pattern attp://hug.itu.dk:8077/oplevelse5/
0
```

#### Eksempel: Dansk Cpr-nummer

```
% set pattern {[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]}
[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]
% regex $pattern "234543-1143"
1
% regex $pattern "23543-1143"
0
% regex $pattern "2354831143"
0
% regex $pattern "13-54-83-1143"
0
```

Der er ikke noget semantisk check af cpr-nummeret – kun et syntaktisk check.

### Bill Gates Personal Wealth Clock

Vi har brug for kursen på Microsoft aktier, hvilket vi får fra følgende url:  
`http://gs.secapi.com/cgi-bin/gs?ticks=MSFT.`

HTML siden indeholder bl.a. strengen:

```
Laest Tradedd at</a></td><td align=right><strong>55 13/16</strong>
```

Vi er interesserede i kursen 55 13/16, som vi matcher ud med følgende `regexp` kommando:

```
regexp {Laest Tradedd at</a></td><td align=right><strong>([^-a-z]*)</strong>} \  
$quote_html match raw_quote
```

Variablen `raw_quote` vil indeholde 55 13/16.

For at regne med kursen skal 55 13/16 omformes til et kommat. Dette gøres igen med regulære udtryk:

```
regexp {(.*) (.*)} "55 13/16" match whole fraction  
regexp {(.*)/(.*)} $fraction match num denom  
set extra [expr double($num) / $denom]  
set kurs [expr $whole + $extra]
```

Først binder vi `whole` til 55 og `fraction` til 13/16.

Derefter binder vi 13 til variablen `num` og 16 til variablen `denom`.

Givet `whole`, `num` og `denom` beregner vi brøken, dvs. kursen, som et kommat: `kurs = 55.8125`.

Trøjaskolen

Databasestøttet Webpublicering, forår 2002

Side 6-9

**Bill Gates Personal Wealth Clock, fortsat**

På url `http://www.census.gov/cgi-bin/popclock` finder vi befolkningsantallet i USA. HTML koden indeholder bl.a.:

```
<center><h1> 283,655,630</h1>
```

Følgende regulære udtryk matcher ud de tre tal adskilt af kommaer:

```
regexp {<h1>[^-0-9]*([0-9]+),([0-9]+)*</h1>} \  
$population_html match millions thousands units
```

Variablen `millions` sættes lig 283, `thousands` lig 655 og `units` lig 630.

Befolkningsantallet beregnes således

```
set population [expr ($millions * 1000000) + ($thousands * 1000) + $units]
```

For at beregne Gates rigdom estimerer vi hans beholdning af Microsoft aktier. Dit bidrag beregnes herfter nært idet vi kender befolkningsstallet.

```
set gates_shares_pre_split [expr double(141159990)]  
set gates_shares [expr $gates_shares_pre_split * 2]  
set gates_wealth [expr $gates_shares * $msft_stock_price]  
set personal_share [expr $gates_wealth / $population]
```

Trøjaskolen

Databasestøttet Webpublicering, forår 2002

Side 6-10

### Opgaver til regulære udtryk og regexp-kommandoen

1. Opskriv `regexp` kommando som for strengen "DDMMYY:DDDD" binder DD til variablen `dag`, MM til `maaned`, YY til `aar` og DDDD til `k`.

```
set pattern _____  
regexp $pattern "280370-3213" _____
```

2. Opskriv `regexp` kommando som for strengen "SE49584832AB34" binder 49584832 til variablen `nr1` og AB34 til `nr2`.

```
set pattern _____  
regexp $pattern "SE49584832-AB34" _____
```

3. Opskriv `regexp` kommando som for strengen "ISBN 1-55860-534-7" binder 1 til variablen `nr1`, 55860 til `nr2`, 534 til `nr3` og 7 til `nr4`.

```
set pattern _____  
regexp $pattern "ISBN 1-55860-534-7" _____
```

Trøjaskolen

Databasestøttet Webpublicering, forår 2002

Side 6-11

**Valutahandel — check af form variable**

Valutahandelservicen består af to filer: `valutahandel.html` og `valutahandel.tcl`

```
valutahandel.html  
<html>  
<head>  
<title>Valutahandel</title>  
</head>  
<body bgcolor=white>  
<h2>Valutahandel</h2>  
<form method=post action=valutahandel.tcl>  
Hvor mange kroner vil du veksle til dollars? <p>  
<input type=text size=10 name=kroner>  
<input type=submit value="Beregn" >  
</form>  
</body>  
</html>
```

`http://hug.it.edu:8002/F2002/lec6/valutahandel.html`

Trøjaskolen

Databasestøttet Webpublicering, forår 2002

Side 6-12

### Valutahandel — fortsat (valutahandel.tcl del 1)

```
set gebyr 20
set dollarkurs 8.34

proc return_page { b } {
    ns_return 200 text/html "
    <html><head><title>Valutahandel</title></head>
    <body bgcolor=white><h2>Valutahandel</h2> $b </body>
    </html>"
}
```

### Valutahandel — fortsat (valutahandel.tcl del 2)

```
set form_variablen
if { ! [info exists kroner] } {
    return_page "Fejl: form variabel 'kroner' forventet"
    return
}
if { ! [regexp {^[0-9]{0-9}*$} $kroner] } {
    return_page "Fejl: heltal forventet"
    return
}
if { $kroner < $gebyr } {
    return_page "Fejl: du kan ikke veksle et beløb mindre end kr. $gebyr"
    return
}
set dollars [expr ($kroner - $gebyr) / $dollarkurs]

return_page "For $kroner kroner modtager du \$$dollars <p>
<a href=\"valutahandel.html\">Tilbage</a>"
```