Programmering

1. Analys av problemet
2. Design
3. Algoritmformulering
4. Kodning
5. Testning och felsökning
6. Underhåll

Definition:

En algoritm för ett problem är en metod för att lösa problemet i fråga i ett **ändligt antal steg**. Metoden består av ett antal **elementära operationer** och en anvisning om i vilken **ordning** dessa ska utföras.

Proffs:
- flambera räkorna

Amatör:
- häll över spriten
- tänd på
- skaka om
- släck elden
- servera genast

- Perl är att braåhaspråk
  - från script till program
- Andra perlar
  - ActivePerl (PC)
  - MacPerl (Mac)
  - Amiga, etc.

Till Unix/Linux finns ett stort antal portar

Länkar:

Perl och CPAN
- http://www.perl.com/pub
- http://www.perl.com/pub/language/info/software.html
- ftp://ftp.du.se/pub/CPAN/README.html

Perl till PC och bra organiserad dokumentation
- http://www.activestate.com/Products/ActivePerl/

Perl till Mac
- http://www.macperl.com/

Annat skoj
- http://www.snerk.net/quotes/geekquotes.txt
- http://www.wall.org/~larry/
- http://www.perl.com/pub/au/Christiansen_Tom

Perl är ett syntaktiskt språk:

- saker och ting måste stå på rätt ställen
- olika konstruktioner har olika syntax
- ”vita tecken” är insignifikanta. Skriv luftig kod (dvs, använd mycket vita tecken!!!)
- Alla yttranden/satser avslutas med semikolon
- Kommentarer inleds med #

Perl är:

• Tolkande (ej kompilerande)
• Plattformsoberoende

Perl har:

• datatyper
• operatorer
• funktioner

Perls datatyper:

- Skalär data ($xxx)
  - tal (4, 5.6, -234, -6.5e24)
  - strängar (’hej’,’a’,”hur står det till med Perlkunskaperna egentligen!!!”, ’’)
- Listor (@xxx)
- Hash (%xxx)

Medlemskap i lista – använd hash!
Variabelnamn börjar alltid med en bokstav (a-z)

Sant / Falskt:

• Falskt är:
  • 0
  • ’’ (tomma strängen/listan)
  • undef (odefinierat värde)

• Allt annat är sant

Variabler är symboler som kan ändra värden över tiden. Namnet förblir dock oförändrat.

Skalära variablers namn börjar med $.
Variabler tilldelas värden med tilldelningsoperatorn ’=’

```
$a = 1; # Nu ha variabeln $a värdet 1

$a = 1111; # Nu har variabeln $a värdet 1111

$a = ’hej på dig din gamla räv!!’;

# Nu har $a värdet ’hej på dig din gamla räv!!’

$a = ’’; # Nu har variabeln $a värdet av en tom sträng
```


Kontexten är viktig!!!

- $a = $a[0];
- @b = @a;
- $b[-1] = $a{$a};
- %b = %a;

Slutsats: Använd vettiga variabelnamn!!!

Specialvariabler

- mansidan perlvar
- $_ är en viktig variabel som innehåller det senaste resultatet av ett utvärderat uttryck

Flera funktioner tar $_ som defaultargument

Aritmetiska operatorer

plus	+	$a = 2 + 3; # $a är 5
minus	-	$a = 2 – 3; # $a är -1
gånger	*	$a = 2 * 3; # $a är 6
delat med	/	$a = 2 / 3; # $a är 0,666666666
modulus	%	$a = 2 % 3; # $a är 2 (heltalsresten)
upphöjt till	**	$a = 2 ** 3; # $a är 8


```
$b = $a / 2;                    $a = $a/2;
print $b;                       print $a;

men helst:
$a /= 2;
print $a;          eller        print $a /= 2;
```

Detta sista kallas för "binär tilldelning" och kan användas med det flesta aritmetiska operatorerna.
+=, -=, *=, /=, **=, %=

Om man ska addera eller minska med ett (increment/decrement) finns det ännu kortare sätt:
$a++ och $a--

Strängoperatorer:

Konkateneringsoperatorn (’.’ (punkt!)):
$a = 'perl' . 'kurs'; # $a blir 'perlkurs'

Upprepningsoperatorn:
$a = 'perl' x 2 ; # $a blir 'perlperl'

Även dessa har sina binära tilldelningsvarianter:
$a = 'perl';
$a x= 2; # $a är nu 'perlperl'
$a .= 'kurs'; # $a är nu 'perlperlkurs'

Jämförelse	**Numerisk**	**Strängar**
lika med	==	eq (**eq**ual)
inte lika med	!=	ne (**n**ot **e**qual)
mindre än	<	lt (**l**ess **t**han)
mindre eller lika med	<=	le (**l**ess or **e**qual)
större än	>	gt (**g**reater **t**han)
större eller lika med	>=	ge (**g**reater or **e**qual)

OBS!

$a = 7 < 30; # $a blir sant, har ett värde skilt från noll oftast 1
$a = 7 lt 30; # $a blir falsk, får värdet noll.
ASCII-värdet för 7 är större än värdet för för 3!

Utskrifter 1

print och strängar med dubbla citationstecken

```
print "hej du!";          >hej du!>

print "hej du!\n";        >hej du!
                          >

$a = 'hej du!';           >hej du!
print "$a\n";             >
```

print skriver som default till <STDOUT>

print

```
$hejsan = 'hej då';
print "Jag vill skriva ut ett bakvänt snedstreck och ett n:
 \\n och variabelnamnet \$hejsan!";
ger:
>Jag vill skriva ut ett bakvänt snedstreck och ett n: \n och
/…/ variabelnamnet $hejsan!

print "Jag vill skriva ut ett bakvänt snedstreck och ett n:
\n och variabelnamnet $hejsan!";
ger:
>Jag vill skriva ut ett bakvänt snedstreck och ett n:
 och variabelnamnet hej då!
```

\n radmatning
\t tabb
\\ bakvänt snedstreck
\” dubbla citationstecken
\u nästa bokstav versal
\l nästa bokstav gemen
\U versaler fram till nästa stopptecken
\L gemener fram till nästa stopptecken
\E stoppteckenet

Utskrifter 2

print och strängar med enkla citationstecken

```
$a = 'Kalle';
$b = '$a Nilsson';
print ’$a $b\n';          >$a $b\n>

print "$a $b\n";          >Kalle $a Nilsson
                          >
```


Inläsning:

```
print ”Ge mig en siffra: \n”;
$a = <STDIN>;
print ”Du valde siffran $a”;

>Ge mig en siffra:
> 15
>Du valde siffran 15
>
```

<> Kollar först kommandoraden, sedan STDIN

Program (Inleds alltid med en ”Shebang”!)

```
#! /usr/local/bin/perl -w
print "Hej! Nu ska jag visa dig allt jag kan göra med siffrorna 2 och 3!\n"   #hälsningsmeddelande
# Summa
$a = 3 + 2;
print "Summan av 2 och 3 är $a! Kul !\n";
# Konkatenering
$a = 3.2;
print "Konkatenering av 3 och 2 är $a! Kul!\n";
```


Programkörning

perl-program måste göras körbara!

```
>chmod a+x litet-prog
>ls -l
-rwx--x--x     litet-prog
> litet-prog
Hej! Nu ska jag visa dig allt jag kan göra med siffrorna 2 och 3!
Summan av 2 och 3 är 5! Kul !
Konkatenering av 3 och 2 är 3.2! Kul!
>
```


Funktioner

chop & chomp

```
$a = 'hej';
$b = chop($a); # $b är 'j' och $a är 'he'
$b = chop($a); # $b är 'e' och $a är 'h'

$a = <STDIN>;
chomp($a);
Nu är $a bara det du skrev minus radslut (\n).
```

Funktioner

Ascii

$a = chr(NUMMER);

$a är tecknet vars asciinummer är NUMMER.

$b = ord(TECKEN);

$b är TECKEN:s asciinummer

OBS Dessa funktioner finns inte med i Learning Perl!!!!!

Strängfunktioner

h	e	j
0	1	2

$sträng = ’hej’;

$position = index($sträng1,$sträng2);

$pos = index($sträng, ’e’); -> $pos blir 1

$position = rindex(’hejhej’,$sträng); -> $position blir 3

$delsträng = substr($sträng1,$start,$längd)

$del = substr($sträng,0,2); -> $del blir ’he’

$del = substr($sträng,-2,2); $del blir ’ej’

Matchning och tr///

tr/listA/listB/

t.ex. tr/AX/BY/ (alla A blir B och alla X blir Y)

gemener -> versaler

$a =~ tr/a-zåäö/A-ZÅÄÖ/;

versaler -> gemener

$a =~ tr/A-ZÅÄÖ/a-zåäö/;

Matchning och tr///

programmet:

$a = ’AX’;

print ”Först är variabeln \$a $a, bra eller hur? \n”;

$a =~ tr/AX/BY/;

print ”nu är variabeln \$a $a, bra eller hur? \n”;

kommer att skriva ut:

> Först är variabeln $a AX, bra eller hur?

nu är variabeln $a BY, bra eller hur?

Matchning och s///

s/A/B/
som tr/// men där A och B kan vara flerteckensuttryck. A byts ut mot B.

```
$a = ’hejdå på dig’;
$a =~ s/hej/tja/;     # $a är ’tjadå på dig’
$a = ~ s/på dig//;    # $a är ’tjadå ’
$a =~ s/å/aa/;        # $a är ’tjadaa ’
```


Styrning av s///

s/// styrs med ’switchar’, t.ex. g och i

T.ex.
$stringy =~ s/kalle/pelle/gi;

Förklaring:
g globalt, annars endast första matchningen
i ej skiftlägeskänslig matchning

I exemplet ovan matchar *kalle* även mot *KallE* och ersätter dessutom alla instanser