Datalogi A

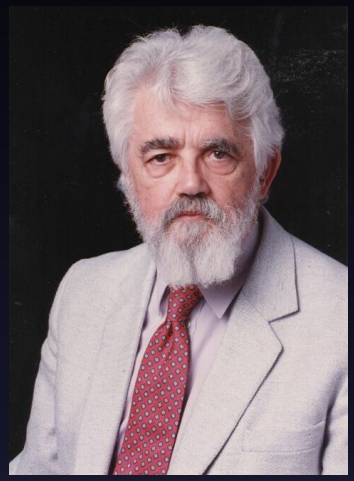
Lektion 4

2010-11-27

Välkommen till lektion 4

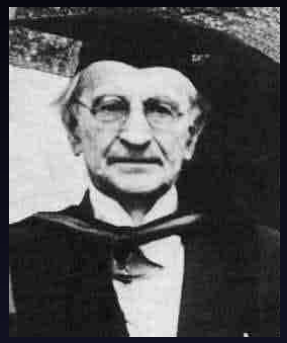
- Lisp
- Lite polsk notation
- Scheme
- Variabler
- Funktioner
- Rekursion
- Listor

Lisp



- John McCarthy skrev en artikel 1958 som beskrev ett nytt sätt att beskriva algoritmer
- Steve Russel läste artikeln, skrev Lisp
- Användes mycket inom AI-forskning
- Är extremt kraftfullt trots få byggstenar

Lite polsk notation



- Uppfanns kring 1920 av den polska matematikern Jan Łukasiewicz

Infix notation	Polsk notation	I Scheme
1 + 2	+ 1 2	`(+ 1 2)`
5 · 2	· 5 2	`(* 5 2)`
1 + 2 - 7	+ 1 - 2 7	`(+ 1 (- 2 7))`
((2 + 3) / (5 - 1))	/ + 2 3 - 5 1	`(/ (+ 2 3) (- 5 1 ))`

Scheme

- Uppfanns av Guy Steele och Gerald Jay Sussman på 70-talet
- Dialekt av Lisp
- Gjord för att vara enkelt


Bullpaus

Variabler

- Skapas med `define`

```
(define a 20)
(display a)
-> 20

(define b 16)
(display (+ a b))
-> 36

(define c (+ a (/ b 2)))
(display c)
-> ?
```

Funktioner

- Skapas med `define`

```
(define (skrivHej)
  (display "hej"))

(skrivHej)
-> hej
```

Funktioner, fortsättning

```
(define (plus a b)
  (+ a b))

(plus 1 2)      -> 3
(plus 5 7)      -> 12
(plus 521 845) -> 1366
```

Rekursion

- När en funktion kallar på sig själv
- Ett vanligt exempel: fakultet
- 5! = 5×4×3×2×1 = 120
- Vad är fel här:

```
(define (fakultet tal)
  (* tal (fakultet (- tal 1))))
```

- Vi behöver avbrottsvillkor! När ska rekursionen avslutas?

Rekursion, fortsättning

- Fel:

```
(define (fakultet tal)
  (* tal (fakultet (- tal 1))))
```

- Rätt:

```
(define (fakultet tal)
  (if (<= tal 1)
    1
    (* tal (fakultet (- tal 1)))))
```

Listor

- Skapas med `list`

```
(define minaTal (list 2 32 1 76 124))
(define minaDjur (list "Ko" "Häst" "Cykel"))
```

- `car` hämtar första platsen, `cdr` hämter resten:

```
(display (car minaDjur)) -> Ko
(display (cdr minaDjur)) -> Häst Cykel
```