ERHVERVSAKADEMI
AARHUS

Programmering modul 05

CSIK E2025

ERHVERVSAKADEMI
AARHUS

I dag

Oplæg om vektorer	08:30
Oplæg om array references	10:30
Oplæg om strings	11:45
Frokost	12:00

Arbejde i teams:

- Bliv færdige med "Imperative programming" side 40-72 og "Stack machine" side 35-62
- "Imperative programming" side 73-98
- Beslut selv, hvornår I holder pauser

Vektor

- Lineær sekvens af *n* dataelementer
- Man kan slå det enkelte element op per indeks
- Indices er de naturlige tal fra 0 (inklusiv) til *n* (eksklusiv)
- Nogle vektorer kan ændres, efter de er dannet

ERHVERVSAKADEMI
AARHUS

Tekststreng = vektor af characters

- Lineær sekvens af *n* characters*, fx `'hello'`
- Man kan slå den enkelte character op per indeks
- Indices er de naturlige tal fra 0 (inklusiv) til *n* (eksklusiv)
- Tekststrenge er uforanderlige og kan ikke ændres, efter de er dannet

```
let s = 'hello'
```

```
write len s // 5
```

```
write s[0] // 'h'
write s[1] // 'e'
write s[2] // 'l'
write s[3] // 'l'
write s[4] // 'o'
write s[5] // fejl
```

```
set s[1] = 'a' // fejl
```

*) Tænk bogstaver, cifre, tegnsætnings-tegn, emojis mv. for nu

Array = vektor af variabler

- Lineær sekvens af *n* variabler
- Man kan slå den enkelte variabel op per indeks
- Indices er de naturlige tal fra 0 (inklusiv) til *n* (eksklusiv)
- Længden af et array er fastlagt ved dannelsen, men de *n* variabler kan ændres senere

```
let a = [42, true, [87, nil]]
```

```
write len a // 3
```

```
write a[0]    // 42
write a[1]    // true
write a[2]    // [87, nil]
write a[2][0] // 87
write a[3]    // fejl
```

```
set a[1] = false
set a[2][1] = 88
write a // [42, false, [87, 88]]
```

Arrays

Syntaks:

```
array[e]
```

array constructor udtryk

Semantik:

1. Evaluér *e*
   Lad os sige, resultatet er *n*
2. Opret et array af længde *n* i arbejdslageret
3. Indsæt værdien **`nil`** på alle pladser i det nye array
4. Værdien af hele udtrykket er en reference til det nye array

```
[e1,…,en]
```

array literal udtryk

1. Evaluér $e_1,\ldots,e_n$ i den rækkefølge
   Lad os sige, resultatet er værdierne $v_1,\ldots,v_n$
2. Opret et array af længde *n* i arbejdslageret
3. Kopiér værdierne $v_1,\ldots,v_n$ ind i det nye array
4. Værdien af hele udtrykket er en reference til det nye array

Arrays

Syntaks:

```
array[e]
```

array constructor udtryk

Semantik:

1. Evaluér *e*
   Lad os sige, resultatet er *n*
2. Opret et array af længde *n* i arbejdslageret
3. Indsæt værdien **`nil`** på alle pladser i det nye array
4. Værdien af hele udtrykket er en reference til det nye array

Sideeffekt

```
[e1,…,en]
```

array literal udtryk

1. Evaluér $e_1,\ldots,e_n$ i den rækkefølge
   Lad os sige, resultatet er værdierne $v_1,\ldots,v_n$
2. Opret et array af længde *n* i arbejdslageret
3. Kopiér værdierne $v_1,\ldots,v_n$ ind i det nye array
4. Værdien af hele udtrykket er en reference til det nye array

Sideeffekt

Arrays

Syntaks:

```
e1[e2]
```

array indexing udtryk

Semantik:

1. Evaluér e_1
   Lad os sige, resultatet er en reference til et array af længde *n*
2. Evaluér e_2
   Lad os sige, resultatet er et heltal *idx*, hvor $0 \leq idx < n$
3. Værdien af hele udtrykket er den værdi, der er gemt på indeks *idx* i arrayet

```
set e1[e2] = e3
```

array indexing udtryk
på venstresiden i set-kommando

1. Evaluér e_1
   Lad os sige, resultatet er en reference til et array af længde *n*
2. Evaluér e_2
   Lad os sige, resultatet er et heltal *idx*, hvor $0 \leq idx < n$
3. Evaluér e_3
   Lad os sige, resultatet er værdien *v*
4. Overskriv variablen på indeks *idx* i arrayet med *v*

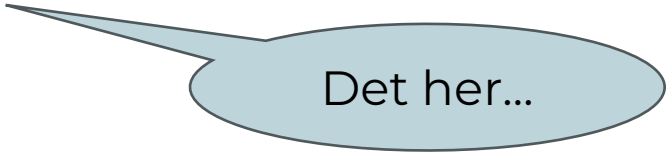
```
len e
```

array length udtryk

1. Evaluér *e*
   Lad os sige, resultatet er en reference til et array af længde *n*
2. Værdien af hele udtrykket er *n*

Arrays

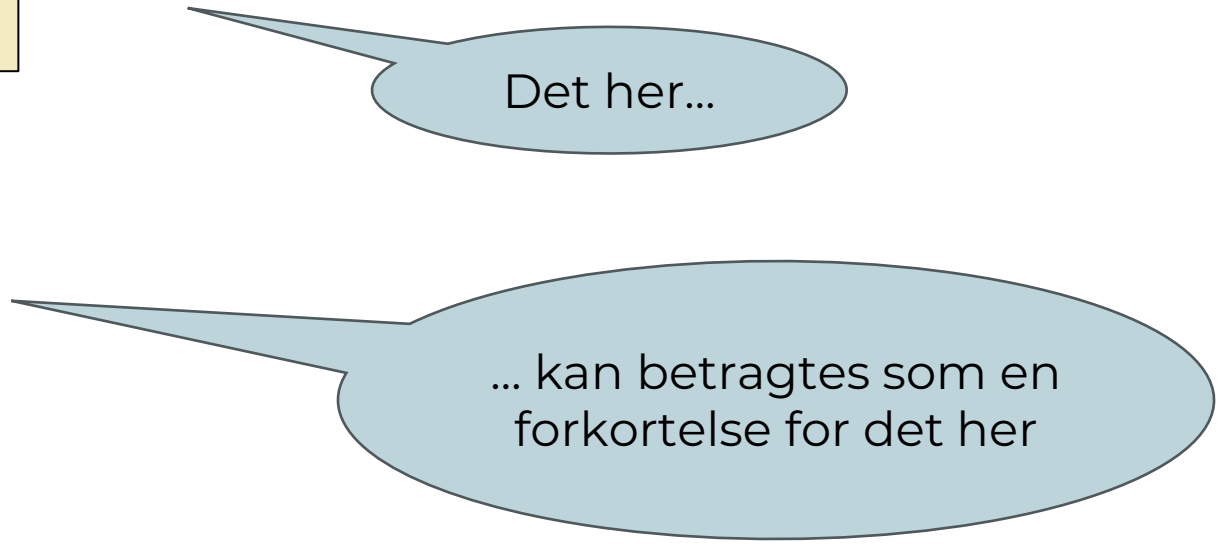
```
let a = [42, true, [87, nil]]
  // do something with a
```



Arrays

```
let a = [42, true, [87, nil]]
  // do something with a
```

```
let a = array[3]
  set a[0] = 42
  set a[1] = true
  set a[2] = [87, nil]
  // do something with a
```



codelabby.com

Med
skuldermakker
og i teams

Canvas:
samarbejdsstrukturer05.pdf

Array references

Hvordan forklarer vi programmets opførsel?

```
let a = array[3]
let b = a
  set b[2] = 99
  write a[2]
```

Array references

Hvordan forklarer vi programmets opførsel?

```
let a = array[3]
let b = a
  set b[2] = 99
  write a[2]
```

I programmeringssproget i "Imperative programming":

- Fem slags værdier: **`nil`**, sandhedsværdier, heltal, tekststrenge, array referencer
- Værdier er uforanderlige, når først de er dannet
- Enhver variabel indeholder præcist én værdi
- En variabel kan sættes til at indeholde en anden værdi med en **`set`**-kommando

Array references

Hvordan forklarer vi programmets opførsel?

```
let a = array[3]
let b = a
  set b[2] = 99
  write a[2]
```

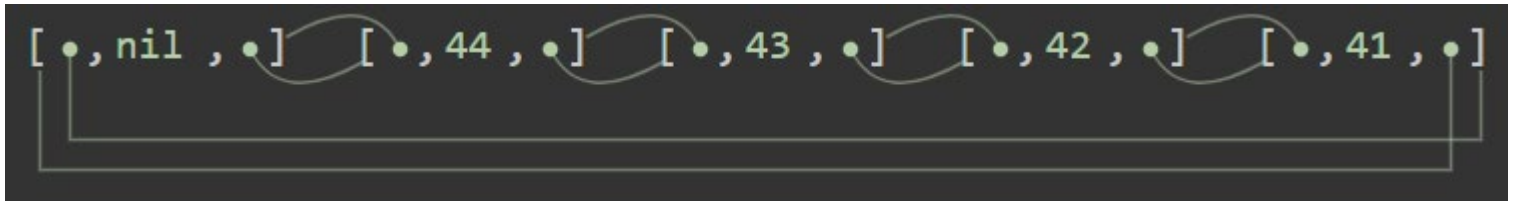
I programmeringssproget i "Imperative programming":

- Fem slags værdier: **`nil`**, sandhedsværdier, heltal, tekststrenge, array referencer
- Værdier er uforanderlige, når først de er dannet
- Enhver variabel indeholder præcist én værdi
- En variabel kan sættes til at indeholde en anden værdi med en **`set`**-kommando
- Arrays er ikke værdier (for arrays kan ændres)
- Variabler indeholder ikke arrays (for variabler indeholder kun værdier)

Hægtede strukturer

Hægtet liste / linked list:

Dobbelt-hægtet liste / doubly-linked list:

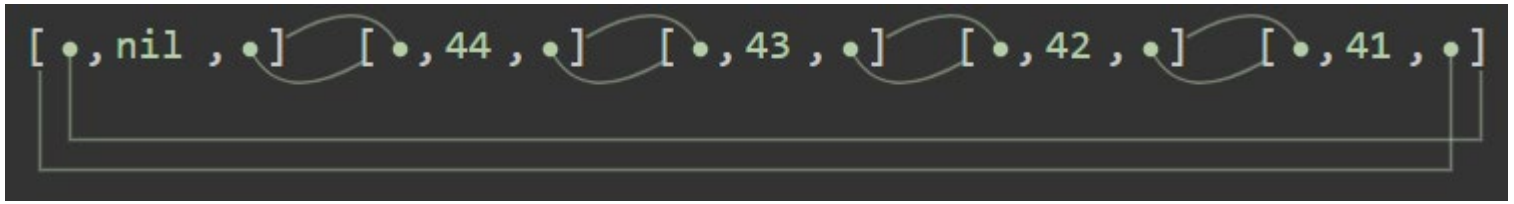


Hægtede strukturer

Hægtet liste / linked list:

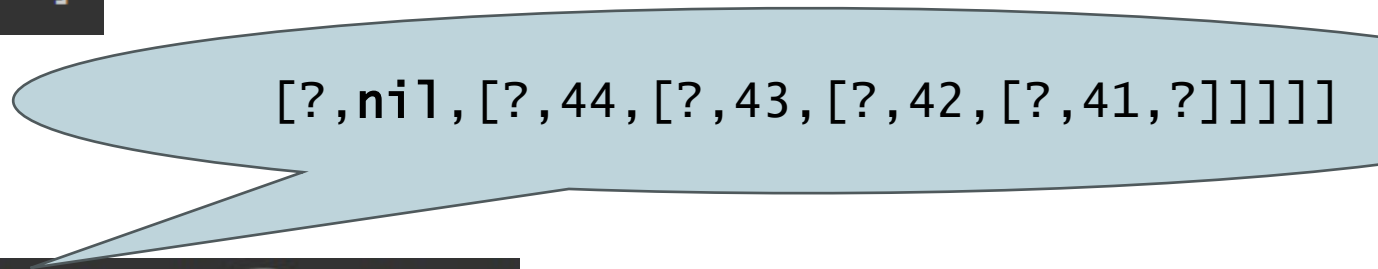
[44,[43,[42,[41,nil]]]]

Dobbelt-hægtet liste / doubly-linked list:

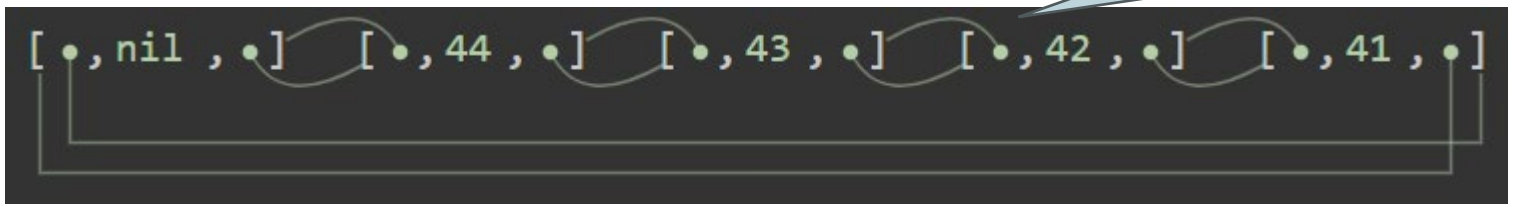


Hægtede strukturer

Hægtet liste / linked list:



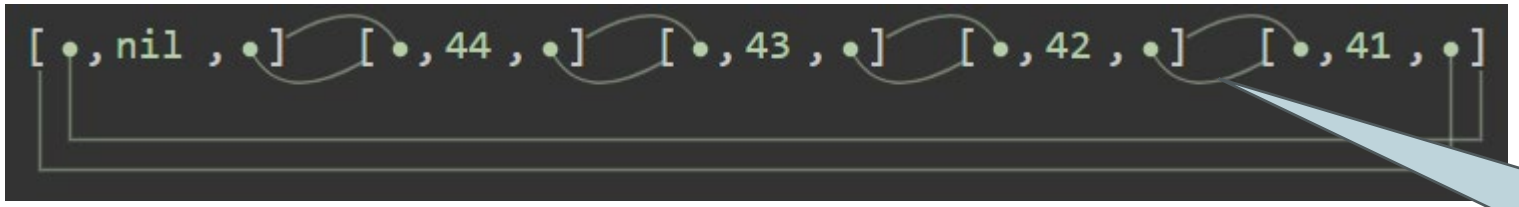
Dobbelt-hægtet liste / doubly-linked list:



Hægtede strukturer

Hægtet liste / linked list:

Dobbelt-hægtet liste / doubly-linked list:



Array reference
= abstrakt lageradresse

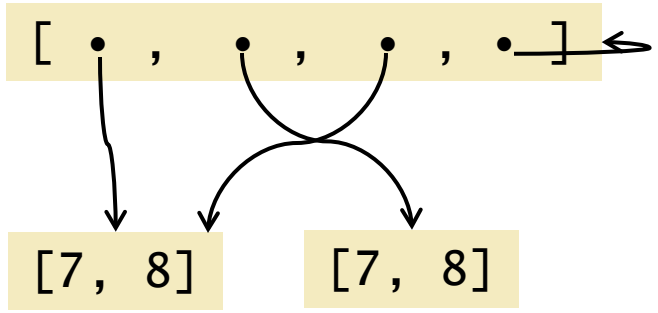
Tekst-repræsentation af arrays

```
given [[7, 8], [7, 8], @1, @0] then [false, true, true, @0]

let a = read
let b = [a[0] = a[1], a[0] = a[2], a = a[3], nil]
  set b[3] = b // create the expected cyclic structure
  write b
```

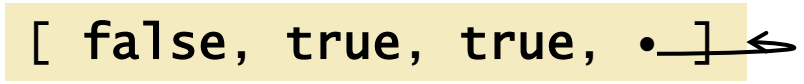
Tekst-repræsentation af arrays

```
given [[7, 8], [7, 8], @1, @0] then [false, true, true, @0]

let a = read
let b = [a[0] = a[1], a[0] = a[2], a = a[3], nil]
  set b[3] = b // create the expected cyclic structure
  write b
```

Input:



Output:



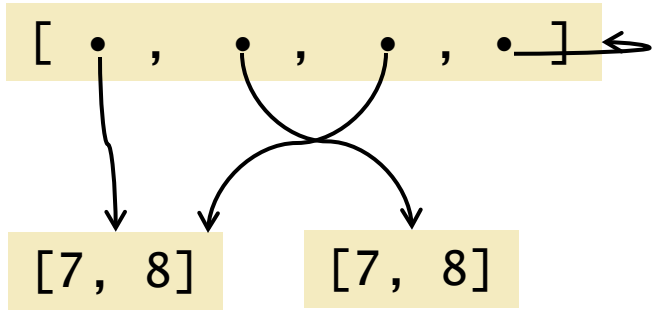
ERHVERVSAKADEMI
AARHUS

Tekst-repræsentation af arrays

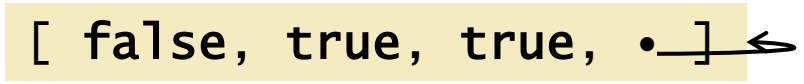
@-notationen er ikke vigtig

```
given [[7, 8], [7, 8], @1, @0] then [false, true, true, @0]

let a = read
let b = [a[0] = a[1], a[0] = a[2], a = a[3], nil]
  set b[3] = b // create the expected cyclic structure
  write b
```

Input:



Output:



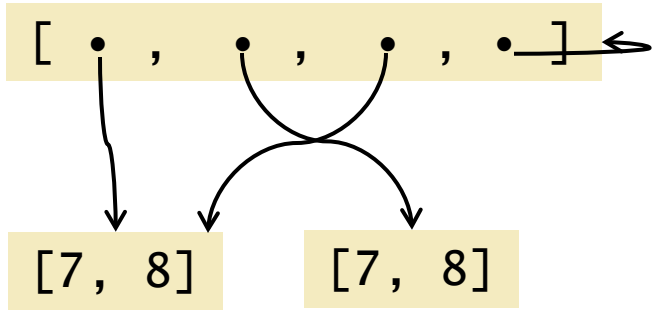
Tekst-repræsentation af arrays

```
given [[7, 8], [7, 8], @1, @0] then [false, true, true, @0]

let a = read
let b = [a[0] = a[1], a[0] = a[2], a = a[3], nil]
  set b[3] = b // create the expected cyclic structure
  write b
```

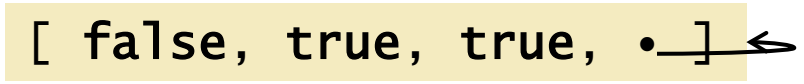
@-notationen er ikke vigtig

I praksis er det ofte en fejl at forsøge at lave en cyklisk struktur om til tekst

Input:



Output:



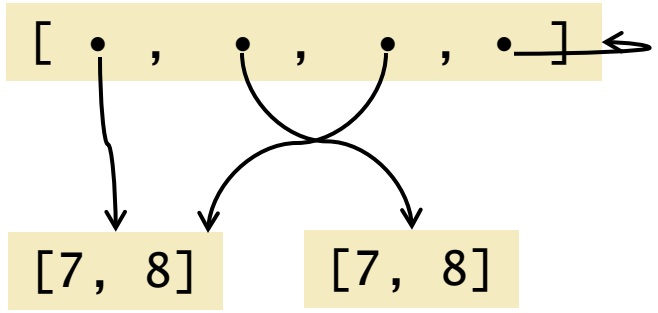
Tekst-repræsentation af arrays

```
given [[7, 8], [7, 8], @1, @0] then [false, true, true, @0]

let a = read
let b = [a[0] = a[1], a[0] = a[2], a = a[3], nil]
  set b[3] = b // create the expected cyclic structure
  write b
```

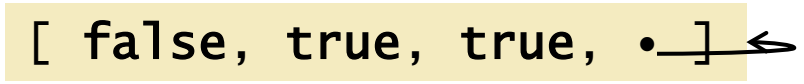
@-notationen er ikke vigtig

I praksis er det ofte en fejl at forsøge at lave en cyklisk struktur om til tekst

Input:



Output:



I praksis bruger man skitser som disse til at arbejde med cykliske strukturer

codelabby.com

Med
skuldermakker
og i teams

Canvas:
samarbejdsstrukturer05.pdf

ERHVERVSAKADEMI
AARHUS

Tekst

Uformelt: tekst er en sekvens af skrifttegn

Tekst

Uformelt: tekst er en sekvens af skrifttegn

Formelt:

Tekststreng / string	= uforanderlig vektor af Unicode code points
Character	= tekststreng af længde 1
Glyf / glyph	= den grafiske form af et skrifttegn (afhænger af skriftsnit / font)

ERHVERVSAKADEMI
AARHUS

Tekst

Uformelt: tekst er en sekvens af skrifttegn

Formelt:

Tekststreng / string	= uforanderlig vektor af Unicode code points
Character	= tekststreng af længde 1
Glyf / glyph	= den grafiske form af et skrifttegn (afhænger af skriftsnit / font)

Eksempel:

- Tre glyffer: a *a* *a*
- Samme character, `'a'`
- Unicode navn: "Latin Small Letter A"
- Unicode code point: 97

https://www.compart.com/en/unicode/U+0061
https://www.compart.com/en/unicode/U+1F600

Strings

Syntaks:

Semantik:

`'naive'`

string literal udtryk

Allerede en værdi: en tekststreng af længde 5
Unicode code points: 110, 97, 105, 118, 101

`ch e`

code point conversion udtryk

1. Evaluér *e*
   Lad os sige, resultatet er et Unicode code point
   dvs. et heltal *n* med $0 \leq n \leq 1114111$
2. Værdien af hele udtrykket er den tilsvarende character
   dvs. en tekststreng af længde 1

`cp e`

code point conversion udtryk

1. Evaluér *e*
   Lad os sige, resultatet er en character
   dvs. en tekststreng af længde 1
2. Værdien af hele udtrykket er det tilsvarende Unicode code point
   dvs. et heltal *n* med $0 \leq n \leq 1114111$

Strings

Syntaks:

```
e1[e2]
```

string indexing udtryk

Semantik:

1. Evaluér e_1
   Lad os sige, resultatet er en tekststreng af længde *n*
2. Evaluér e_2
   Lad os sige, resultatet er et heltal *idx*, hvor $0 \leq idx < n$
3. Værdien af hele udtrykket er den character, der befinder sig på indeks *idx* i tekststrengen

```
len e
```

string length udtryk

1. Evaluér *e*
   Lad os sige, resultatet er en tekststreng af længde *n*
2. Værdien af hele udtrykket er *n*

Strings

Syntaks:

```
'abc{e1}def{e2}ghi{e3}jkl'
```

string interpolation udtryk

Semantik:

1. Evaluér e_1, e_2, e_3 i den rækkefølge
   Lad os sige, resultatet er tekststrengene s_1, s_2, s_3
2. Værdien af hele udtrykket er sammensætningen
   `'abc'` s_1 `'def'` s_2 `'ghi'` s_3 `'jkl'`

```
e1 join e2
```

join udtryk

1. Evaluér e_1
   Lad os sige, resultatet er et array af længde n
   og at elementerne kan konverteres til tekststrengene $s_1,\ldots,s_n$
2. Evaluér e_2
   Lad os sige, resultatet er tekststrengen s
3. Værdien af hele udtrykket er sammensætningen
   $s_1\ s\ s_2\ s\ \ldots\ s\ s_n$

ERHVERVSAKADEMI
AARHUS

Hjemmearbejde

Se på Canvas under Plan for modul 05

ERHVERVSAKADEMI
AARHUS

Reaktioner på i dag

Spørgsmål?

Bekymringer?

Protester?

Kritik?