# Imperativ Programmering og Datastrukturer

## Rekursion

René Rydhof Hansen

24. november 2008

# Mål

- At kunne forklare/definere hvad rekursion er (som begreb)
- At kunne implementere en given rekursiv funktion
- At kunne forklare hvordan rekursion kan bruges som problemløsningsstrategi

# For at forstå rekursion... skal man først forstå rekursion

## Rekursion

- Rekursion er en problemløsningsstrategi baseret på løsning af delproblemer af samme slags som hovedproblemet.
- Der skal være et basistilfælde der ikke involverer rekursion for at sikre terminering.

## Example

En folder kan indeholde

- Filer
- 

## Rekursion vs. Iteration

- Løkker kan altid formuleres som rekursion
- Rekursion kan, med noget besvær, formuleres som iteration

# For at forstå rekursion... skal man først forstå rekursion

## Rekursion

- Rekursion er en problemløsningsstrategi baseret på løsning af delproblemer af samme slags som hovedproblemet.
- Der skal være et basistilfælde der ikke involverer rekursion for at sikre terminering.

## Example

En folder kan indeholde

- Filer
-

## Rekursion vs. Iteration

- Løkker kan altid formuleres som rekursion
- Rekursion kan, med noget besvær, formuleres som iteration

# For at forstå rekursion... skal man først forstå rekursion

## Rekursion

- Rekursion er en problemløsningsstrategi baseret på løsning af delproblemer af samme slags som hovedproblemet.
- Der skal være et basistilfælde der ikke involverer rekursion for at sikre terminering.

## Example

En folder kan indeholde

- Filer
- ... og *andre foldere*

## Rekursion vs. Iteration

- Løkker kan altid formuleres som rekursion
- Rekursion kan, med noget besvær, formuleres som iteration

# Tilbage til øen

## Example

Fakultetsfunktionen (!):

$$n! = n \cdot (n-1) \cdot \cdots \cdot 2 \cdot 1 = \begin{cases} 1 & n = 0 \\ n \cdot (n-1)! & \text{ellers} \end{cases}$$

```
public static int fact(int i) {
  if (i == 0) {
    return 1;
  } else {
    return i * fact(i-1);
  }
}
```

# Endnu et eksempel: Fibonacci-tallene

## The Rabbit Connection

A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?

## Svar

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 \ldots$$

## Definition

$$\text{fib}(n) = \begin{cases} 1 & \text{hvis } n = 1 \\ 1 & \text{hvis } n = 2 \\ \text{fib}(n-1) + \text{fib}(n-2) & \text{ellers} \end{cases}$$

# Endnu et eksempel: Fibonacci-tallene

## The Rabbit Connection

A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?

## Svar

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 \ldots$$

## Definition

$$\mathrm{fib}(n) = \begin{cases} 1 & \text{hvis } n = 1 \\ 1 & \text{hvis } n = 2 \\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2) & \text{ellers} \end{cases}$$

# Endnu et eksempel: Fibonacci-tallene

## The Rabbit Connection

A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?

## Svar

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 \ldots$$

## Definition

$$\mathrm{fib}(n) = \begin{cases} 1 & \text{hvis } n = 1 \\ 1 & \text{hvis } n = 2 \\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2) & \text{ellers} \end{cases}$$

# Fibonacci-tallene i C#

```
public static int fib(int n) {
  if(n == 1) {
    return 1;
  } else if (n == 2) {
    return 1;
  } else {
    return fib(n-1) + fib(n-2);
  }
}
```

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
  x
 xxx
xxxxx
  0     1     2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
  xxx
 xxxxx         x
   0     1     2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

**En gang for længe siden...**

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
xxxxx   xxx    x
  0      1     2
```

**Rekursiv Løsning**

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
        x
xxxxx  xxx
  0     1     2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
          x
         xxx    xxxxx
   0      1       2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
x      xxx    xxxxx
0       1       2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
                  xxx
 x               xxxxx
 0        1        2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
                  x
                 xxx
                xxxxx
  0       1       2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Rekursion som generel strategi: Hanoi's Tårne

## En gang for længe siden...

- Skiverne skal flyttes fra pæl 0 til pæl 2
- En skive kan flyttes til en tom pæl
- En skive kan flyttes over på en *større* skive

```
                x
               xxx
              xxxxx
  0      1      2
```

## Rekursiv Løsning

- Flyt øverste 2 skiver fra 0 til 1 via 2
- Flyt sidste skive direkte fra 0 til 2
- Flyt øverste 2 skiver fra 1 til 2 via 0

# Opsummering og Næste gang

Opsummering

- Rekursion som begreb

Næste gang

- Rekursion i programmering (rekursive datastrukturer)

## Repetition af basal programmering

- Pr. gruppe
- Send repetitionsemner til `rrh@cs.aau.dk`