Introduktion til Programmering og Problemløsning (PoP)

Jon Sporring
Department of Computer Science
2021/10/07

UNIVERSITY OF COPENHAGEN

Hvor langt er I kommet med materialet?

https://tinyurl.com/mtx48bbp

Hvad er værdien og typen

https://tinyurl.com/4ethtv6t

Arrays vs. lists

Hvad skriver programmet?

arr.fsx

```
let arr = [| 3.0; 4.0 |]
printfn "%A" arr
for i = 0 to 1 do
  arr.[i] <- float(i)
  printfn "%A" arr
printfn "%A" arr
```

-:--- arr.fsx All L1

lst.fsx

```
let lst = [ 3.0; 4.0 ]
printfn "%A" lst
for i = 0 to 1 do
  let lst = [0 .. i]
  printfn "%A" lst
printfn "%A" lst
```

-:--- lst.fsx All L5

https://tinyurl.com/549mne7x

Funktioner som argumenter og returværdi

fctArgument.fsx

```
let dbl x = 2 * x
let apply dbl x = dbl x
printfn "%A, %A" (dbl 3) (apply dbl 3)
```

```
U:---  fctArgument.fsx   All L4     (fsharp)
```

map.fsx

```
let lst = [0..5]
let sqr x = x * x
let lst2 = List.map sqr lst
printfn "All elements in a %A squared is %A" lst lst2
```

```
U:---  map.fsx        All L5     (fsharp)
```

fctReturn.fsx

```
let mul a =
    let f x = a * x
    f
let f = mul 2
printfn "%A, %A" (mul 3) (f 3)
```

```
-:---  fctReturn.fsx   All L6     (fsharp)
End of buffer
```

mapAnonymous.fsx

```
let lst = [0..5]
let lst2 = List.map (fun x -> x * x) lst
printfn "All elements in a %A squared is %A" lst lst2
```

```
U:---  mapAnonymous.fsx   All L2     (fsharp)
Wrote /Users/jrh630/repositories/PoP.git/lectures/05Lists/sr
c/mapAnonymous.fsx
```

List.fold og List.foldBack

Hvad er typen?

```
% fsharpi

Microsoft (R) F# Interactive version 11.0.0.0 for F# 5.0
Copyright (c) Microsoft Corporation. All Rights Reserved.

For help type #help;;

> List.fold;;
val it : (('a -> 'b -> 'a) -> 'a -> 'b list -> 'a)
> List.foldBack;;
val it : (('a -> 'b -> 'b) -> 'a list -> 'b -> 'b)
```

Vilkårlig type ‘a og ‘b

List.fold tager en funktion f a b -> a, a, en liste af b og returnerer a typer.

fold.fsx

```
let ctf str v = str + ", " + (string v)
let lst = [0 .. 2]
let aString = List.fold ctf "" lst
printfn "%A vs. %A" lst aString
```

-:--- fold.fsx All L5 (fsharp)

List.fold: ctf (ctf (ctf "" 0) 1) 2

List.foldBack tager en funktion f a b -> b, en liste af a, b og returnerer b.

foldBack.fsx

```
let ctb v str = str + ", " + (string v)
let lst = [0 .. 2]
let aString = List.foldBack ctb lst ""
printfn "%A vs. %A" lst aString
```

-:--- foldBack.fsx All L1 (fsharp)

List.foldBack: ctb 0 (ctb 1 (ctb 2 ""))

List.fold og List.foldBack

Hvordan kan man bruge fold hhv. foldBak men undgå det foranstillede “, “?

fold.fsx

```
let ctf str v = str + ", " + (string v)
let lst = [0 .. 2]
let aString = List.fold ctf "" lst
printfn "%A vs. %A" lst aString
```

-:--- fold.fsx All L5 (fsharp)

foldBack.fsx

```
let ctb v str = str + ", " + (string v)
let lst = [0 .. 2]
let aString = List.foldBack ctb lst ""
printfn "%A vs. %A" lst aString
```

-:--- foldBack.fsx All L1 (fsharp)

List.fold og List.foldback med anonyme funktioner

Sum en liste af heltal:

v1 let add a b = a+b
List.fold **add** 0 [1..5]

v2 let add a b = a+b **in** List.fold add 0 [1..5]

v3 List.fold **(fun acc elm -> acc + elm)** 0 [1..5]

v4 List.foldBack (fun elm acc -> acc + elm) [1..5] 0

Debugging med printfn og sekvenser af udsagn:

List.fold (fun acc elm -> **printfn "%d" acc;** acc + elm) 0 [1..5]

List.foldBack (fun elm acc -> printfn "%d" acc; acc + elm) [1..5] 0

Hvor lang tid tager det?

time.fsx

```
open System.Diagnostics

let lst = [1UL..1000000UL]
let timer = Stopwatch()
let N = 3;

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let af = List.fold (fun acc elm -> acc + elm) 0UL lst
  printfn "fold sum: %A" af
  timer.Stop()
  printfn "fold took: %d ms" timer.ElapsedMilliseconds

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let ab = List.foldBack (fun elm acc -> acc + elm) lst 0UL
  printfn "foldback sum: %A" ab
  timer.Stop()
  printfn "foldback took: %d ms" timer.ElapsedMilliseconds
```

```
U:---  time.fsx        All L5     (fsharp)
Wrote /Users/jrh630/repositories/PoP.git/lectures/05Lists/src/time.fsx
```

```
% fsharpi time.fsx
fold sum: 500000500000UL
fold took: 37 ms
fold sum: 500000500000UL
fold took: 4 ms
fold sum: 500000500000UL
fold took: 2 ms
foldback sum: 500000500000UL
foldback took: 17 ms
foldback sum: 500000500000UL
foldback took: 11 ms
foldback sum: 500000500000UL
foldback took: 13 ms

% fsharpc time.fsx && mono time.exe
Microsoft (R) F# Compiler version 11.0.0.0 for F# 5.0
Copyright (c) Microsoft Corporation. All Rights Reserved.
fold sum: 500000500000UL
fold took: 162 ms
fold sum: 500000500000UL
fold took: 5 ms
fold sum: 500000500000UL
fold took: 2 ms
foldback sum: 500000500000UL
foldback took: 15 ms
foldback sum: 500000500000UL
foldback took: 12 ms
foldback sum: 500000500000UL
foldback took: 9 ms
```

Hvor lang tid tager det?

timeArrPostPrepen...

```
open System.Diagnostics

let timer = Stopwatch()
let N = 3;
let M = 100000;

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let mutable arr = [||]
  for i = 1 to M do
      arr <- Array.append [|i|] arr
  printfn "arr length: %A" arr.Length
  timer.Stop()
  printfn "prepend took: %d ms" timer.ElapsedMilliseconds

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let mutable arr = [||]
  for i = 1 to M do
      arr <- Array.append arr [|i|]
  printfn "arr length: %A" arr.Length
  timer.Stop()
  printfn "postpend took: %d ms" timer.ElapsedMilliseconds
```

-:--- timeArrPostPrepend.fsx All L19 (fsharp)

timeLstPostPrepen...

```
open System.Diagnostics

let timer = Stopwatch()
let N = 3;
let M = 10000;

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let mutable lst = []
  for i = 1 to M do
      lst <- i::lst
  printfn "lst length: %A" lst.Length
  timer.Stop()
  printfn "prepend took: %d ms" timer.ElapsedMilliseconds

for i = 1 to N do
  timer.Reset()
  timer.Start()
  let mutable lst = []
  for i = 1 to M do
      lst <- lst @ [i]
  printfn "lst length: %A" lst.Length
  timer.Stop()
  printfn "postpend took: %d ms" timer.ElapsedMilliseconds
```

-:--- timeLstPostPrepend.fsx All L16 (fsharp)

Spørgsmål