Introduktion til Programmering og Problemløsning (PoP)

Håndkøring

Jon Sporring
Department of Computer Science
2022/09/12

UNIVERSITY OF COPENHAGEN

Funktioner

Leksikografisk virkefelt

Organisering = nemmere at forstå og vedligeholde

```
let greetings (name : string) : string =
    "Hello " + name
```

Indryk angiver funktionskroppen

```
let str = greetings "Jon"
printfn "%A" str
printfn "%A" (greetings "World")
```

```
> let greetings (name : string) : string
=
-   "Hello " + name;;
val greetings : name:string -> string
```

```
let greetings name =
  "Hello " + name
```

```
let greetings name = "Hello " + name
```

```
let greetings name : string = "Hello " + name
```

```
let greetings (name : string) = "Hello " + name
```

Dokumentation - simpel

3 = 2 + 1 linjekommentar

```
/// Write a simple greeting to someone
let greetings (name : string) : string = "Hello " + name // name is without space in front
(* This function needs to be tested! *)
```

Almindelig kommentarer udenfor dokumentationsstandarden

Løs en andengradsligning (i 8 trin)

1. Note omhvad skal funktionen gøre
   /// Given parameters a, b, and c, solve for x
   /// when a*x*x + b*x + c = 0

2. Find på et navn
   solve

3. Skriv et eksempel på dens brug
   let a = 1.0
   let b = 0.1
   let c = -1.0
   let x = solve a b c
   printfn "0 = %Ax^2 + %Ax + %A => x = %A" a b c x

4. Beslut typen
   let solve (a: float) (b: float) (c: float) : float*float = (-3.4, 2.1)

Mockup funktion

5. Lav en implementation
   let solve (a: float) (b: float) (c: float) : float*float =
     let sqrtd = sqrt (b ** 2.0 - 4.0 * a * c)
     ((-b - sqrtd) / (2.0 * a), (-b + sqrtd) / (2.0 * a))

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Lav en funktion i 8-trin

1. Note omhvad skal funktionen gøre
2. Find på et navn
3. Skriv et eksempel på dens brug
4. Beslut typen
5. Lav en implementation
6. (Intern test)
7. Kør eksemplet fra 3 (og 6)
8. Skriv dokumentation

Løs en andengradsligning (i 8 trin)

```
/// Given parameters a, b, and c, solve for x
/// when a*x*x + b*x + c = 0
let solve (a: float) (b: float) (c: float) : float*float =
  let sqrtd = sqrt (b ** 2.0 - 4.0 * a * c) // We assume that d >= 0
  ((-b - sqrtd) / (2.0 * a), (-b + sqrtd) / (2.0 * a))

// Example use of solve
let a = 1.0
let b = 0.1
let c = -1.0
let x = solve a b c
printfn "%Ax^2+%Ax+%A=0 => x=%A" a b c x
```

Dokumentationsstandarden

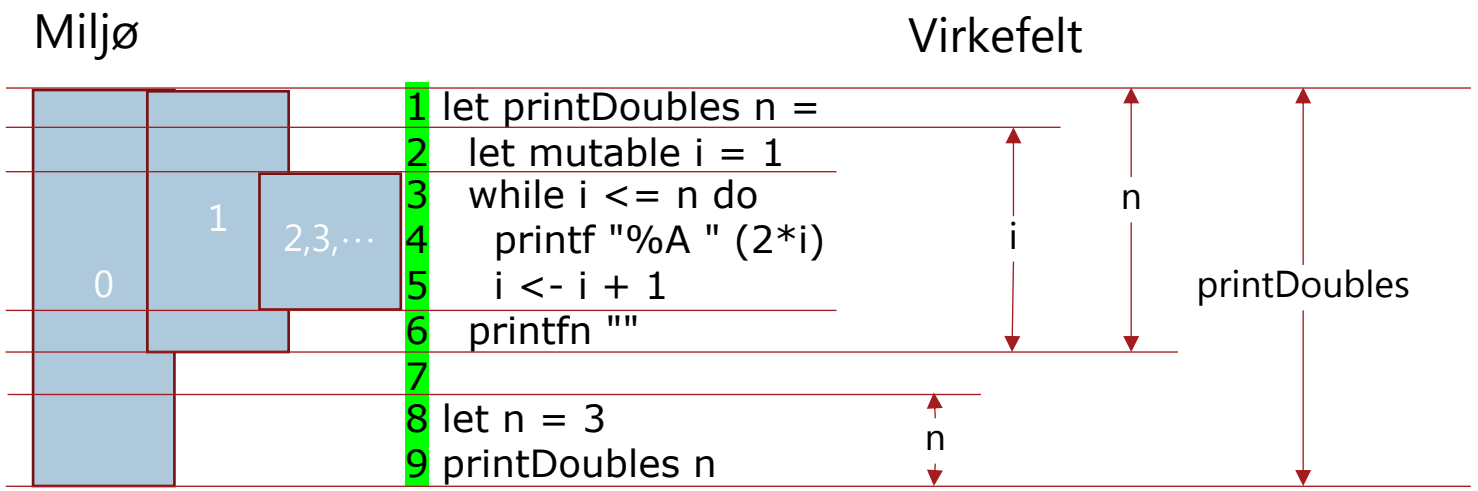
```
/// <summary>
/// Given parameters a, b, and c, solve for x
/// when a*x*x + b*x + c = 0
/// </summary>
/// <param name="a">Quadratic coefficient.</param>
/// <param name="b">Linear coefficient.</param>
/// <param name="c">Constant coefficient.</param>
/// <param name="sgn">+1 or -1 determines the solution.</param>
/// <returns>The solution to x.</returns>
let solve (a: float) (b: float) (c: float) : float*float =
  let sqrtd = sqrt (b ** 2.0 - 4.0 * a * c) // We assume that d >= 0
  ((-b - sqrtd) / (2.0 * a), (-b + sqrtd) / (2.0 * a))

// Example use of solve
let a = 1.0
let b = 0.1
let c = -1.0
let x = solve a b c
printfn "%Ax^2+%Ax+%A=0 => x=%A" a b c x
```

Virkefelt (scope) vs. miljø (environment)

En værdi/variable/funktions **virkefelt/scope** er den del af et program, hvor den er synlig

Et **miljø/environment** er en samling af værdier/variable/funktioner tilgængeligt på et givent sted i et program




```
% dotnet fsi printDoubles.fsx
2 4 6
```

Håndkøring (tracing)

simple.fsx

```
1 let a = 3
2 printfn "%A" a
```

Trin	Linje	Miljø	Bindinger osv.
0	-	E0	-
1	1	E0	a=3
2	2	E0	output = “3”

Initialle oprettelse kan udelades

Håndkøring (tracing)

function.fsx

```
1 let double n =
2   2*n
3
4 let result = double 5
5 printfn "%A" result
```

Trin	Linje	Miljø	Bindinger osv.
1	1	E0	double = ((n), 2*n, ())
2	4	E0	result = double 5?
3	1	E1	((n=5), 2*n, ())
4	2	E1	return = 10
5	4	E0	result = 10
6	5	E0	output = “10”

Opret nyt miljø, kopier og indsæt argumentværdi

Luk indlejret miljø, vend tilbage til kaldemiljø, og erstat ukendt værdi.

Funktioner er værdier (closures):
fkt = ((argument navne), udtryk, miljøtilstand)

Håndkøring (tracing)

loop.fsx

```
1 let printDoubles n =
2   let mutable i = 1
3   while i <= n do
4     printf "%A " (2*i)
5     i <- i + 1
6   printfn ""
7
8 let n = 2
9 printDoubles n
```

Trin	Linje	Miljø	Bindinger osv.
1	1	E0	printDoubles = ((n), body, ())
2	8	E0	n = 2
3	9	E0	result = printDoubles 2?
4	1	E1	((n=2), body, ())
5	2	E1	i = α_1, i <- 1
6	3	E2	(), while-body, (n=2, i=α_1))
7	3	E2	exit = false
8	4	E3	(), 2*i, (n=2, i=α_1))
9	4	E3	result = 2
10	4	E2	output = “2”
11	5	E2	i <- 2
12	3	E2′	(), while-body, (n=2, i=α_1))
13	3	E2′	exit = false
14	4	E3	(), 2*i, (n=2, i=α_1))
15	4	E3	result = 4
16	4	E2′	output = “4”
17	5	E2′	i <- 3
18	3	E2′′	(), while-body, (n=2, i=α_1))
19	3	E2′′	exit = true
20	6	E1	output = <newline>
21	9	E0	result = ()

Luk while-miljø og opret et nyt

Bunken (heap)

Trin	Bindinger osv.
5	α_1 = 1
11	α_1 = 2
17	α_1 = 2

Resumé

I denne video hørte du om:

- Håndkøring (tracing) af
  - et simpelt program
  - en funktion
  - en while-løkke med mutérbar værdi