Programmering og Problemløsning

4.2: Moduler og afprøvning

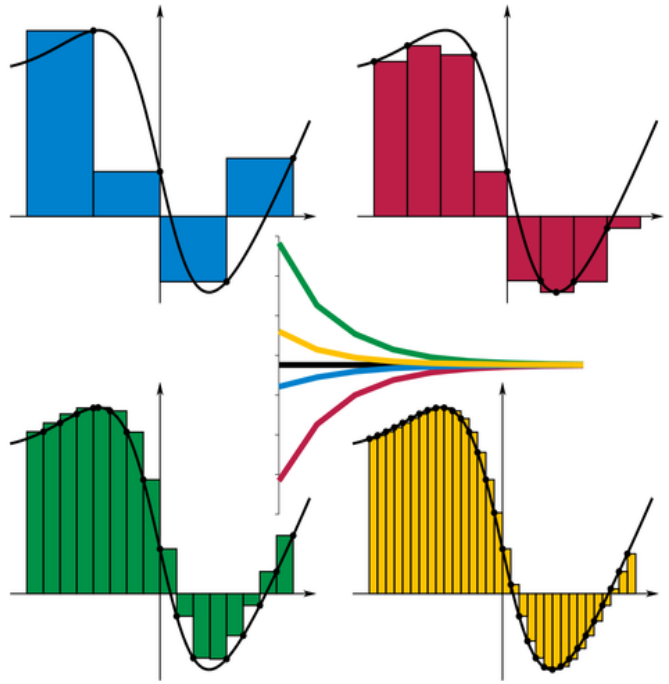
Repetition af Nøglekoncepter

- Tupler
- Betingelser

- Stakken og bunken
- Referenceceller
- Højere-ordens funktioner
- Anonyme funktioner

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum
```

```
let a = 0.0
let b = 1.0
let d = 1e-5
let result = integrate (fun x -> x * exp(x)) a b d
printfn "Int_%g^%g f(x) dx = %g" a b result
```



By I, KSmrq, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=2347919

Moduler og biblioteker

```
$ fsharpc -a integrate.fsi integrate.fs
$ fsharpc -r integrate.dll application.fsx
```

Bibliotek (library)

```
/// Estimate the integral of f
/// from a to b with stepsize d
let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum
```

Signatur (.fsi)

```
module metaFunctions

/// Estimate the integral of f
/// from a to b with stepsize d
val integrate : (float -> float) -> float -> float -> float -> float
```

Implementation (.fs)

```
module metaFunctions

let integrate f a b d =
  let mutable sum = 0.0
  let mutable x = a
  while x < b do
    sum <- sum + d * (f x)
    x <- x + d
  sum
```

Application (.fsx)

```
let a = 0.0
let b = 1.0
let d = 1e-5
let f x = x * exp(x)
let result = metaFunctions.integrate f a b d
printfn "Int_%g^%g f(x) dx = %g" a b result
```

Applikation/program

```
let a = 0.0
let b = 1.0
let d = 1e-5
let result = integrate (fun x -> x * exp(x)) a b d
printfn "Int_%g^%g f(x) dx = %g" a b result
```

Biblioteksvarianter

Åbning

Pas på namespace polution

```
open metaFunctions

let a = 0.0
let b = 1.0
let d = 1e-5
let f x = x * exp(x)
let result = integrate f a b d
printfn "Int_%g^%g f(x) dx = %g" a b result
```

Filsuffikser

.fsx – script fil
.fsscript – script fil
.fs – implementaitons fil
.fsi – signatur fil
.dll – oversat bibliotek
.exe – oversat og linket program

Adgangskontrol

Signatur variant (.fsi)

```
module metaFunctions
```

Oversættelse:

```
$ fsharpc --nologo -a integrateVar.fsi integrate.fs
$ fsharpc --nologo -r integrate.dll application.fsx
```

.../application.fsx(5,28): error FS0039: The value, constructor, namespace or type 'integrate' is not defined.

Krav til Software

- Funktionalitet: Kompilerer det, løser det opgaven?
- Pålideligt: Hvad vis internettet falder ud?
- Brugsvenligt: Er det nemt at bruge?
- Effektivitet: Tager det lang tid at bruge, er det langsomt?
- Vedligeholdelse: Er det net at rette bugs, at tilføje ny funktionalitet?
- Portérbart: Kan det nemt flyttes til en ny computer, telefon, etc.?

Black-box testing

1. Beslut et interface
2. Find grænsetilfælde

let dec2bin n = ?

Unit	Case	Expected output	Comment
dec2bin n	n = -1	"Illegal value"	negative tal
	n = 0	"0b0"	grænsetilfælde
	n = 1	"0b1"	1 bit
	n = 2	"0b10"	2 bit
	n = 10	"0b1010"	stort lige tal (venstre bit sat min ikke højre)
	n = 11	"0b1011"	stort ulige tal (venstre og højre bit sat)

Black-box (unit) testing

Unit	Case	Expected output	Comment
dec2bin n	n = -1	"Illegal value"	negative tal
	n = 0	"0b0"	grænsetilfælde
	n = 1	"0b1"	1 bit
	n = 2	"0b10"	2 bit
	n = 10	"0b1010"	stort lige tal (venstre bit sat min ikke højre)
	n = 11	"0b1011"	stort ulige tal (venstre og højre bit sat)

```
// Unit : dec2bin
let dec2bin n =
  if n < 0 then
    "Illegal value"
  elif n = 0 then
    "0b0"
  else
    let mutable v = n
    let mutable str = ""
    while v > 0 do
      str <- (string (v % 2)) + str
      v <- v / 2
    "0b" + str

printfn "Black-box testing of dec2bin.fsx"
printfn "  n < 0  - %b" (dec2bin -1 = "Illegal value")
printfn "  n = 0  - %b" (dec2bin 0 = "0b0")
printfn "  n = 1  - %b" (dec2bin 1 = "0b1")
printfn "  n = 2  - %b" (dec2bin 2 = "0b10")
printfn "  n = 10 - %b" (dec2bin 10 = "0b1010")
printfn "  n = 11 - %b" (dec2bin 11 = "0b1011")
```

```
$ fsharpi dec2binBlackTest.fsx
Black-box testing of dec2bin.fsx
  n < 0  - true
  n = 0  - true
  n = 1  - true
  n = 2  - true
  n = 10 - true
  n = 11 - true
```

White-box (unit) testing

1. Beslut hvilke units, der skal afprøves
2. Identificer forgreningspunkter
3. Lav inputeksempler for alle units, som afprøver hver forgreningsvej, og notér det forventede output
4. Skriv et program, som kører koden med alle inputeksempler, og sammenlign resultatet med det forventede output

```
let dec2bin n =
  if n < 0 then                (* WB: 1 *)
    "Illegal value"
  elif n = 0 then              (* WB: 2 *)
    "0b0"
  else
    let mutable v = n
    let mutable str = ""
    while v > 0 do             (* WB: 3 *)
      str <- (string (v % 2)) + str
      v <- v / 2
    "0b" + str
```

Unit	Branch	Condition	Input	Expected output	Comment
dec2bin	1	n < 0			
	1a	true	-1	"Illegal value"	
	1b	branch 2			Fall through
	2 (n>=0)	n = 0			
	2a	true	0	"0b0"	
	2b	branch 3			Fall through
	3 (n>0)				
	3a	true	1	"0b1"	
	3b	false	1	"0b1"	

White-box (unit) testing

```
// Unit : dec2bin
let dec2bin n =
  if n < 0 then            (* WB: 1 *)
    "Illegal value"
  elif n = 0 then          (* WB: 2 *)
    "0b0"
  else
    let mutable v = n
    let mutable str = ""
    while v > 0 do         (* WB: 3 *)
      str <- (string (v % 2)) + str
      v <- v / 2
    "0b" + str
```

Unit	Branch	Condition	Input	Expected output	Comment
dec2bin	1	n < 0			
	1a	true	-1	"Illegal value"	
	1b	branch 2			Fall through
	2 (n>=0)	n = 0			
	2a	true	0	"0b0"	
	2b	branch 3			Fall through
	3 (n>0)				
	3a	true	1	"0b1"	
	3b	false	1	"0b1"	

```
printfn "White-box testing of dec2bin.fsx"
printfn "  Unit: dec2bin"
printfn "    Branch: 1a - %b" (dec2bin -1 = "Illegal value")
printfn "    Branch: 2a - %b" (dec2bin 0 = "0b0")
printfn "    Branch: 3a - %b" (dec2bin 1 = "0b1")
```

```
$ fsharpi dec2binWhiteTest.fsx
White-box testing of dec2bin.fsx
  Unit: dec2bin
    Branch: 1a - true
    Branch: 2a - true
    Branch: 3a - true
```