Programmering og Problemløsning

4.1: Tupler, Moduler og afprøvning

Fibonacci

For-løkke

```
let mutable m = 1
let mutable n = 1
let N = 5
for i = 3 to N do
  let p = m + n
  m <- n
  n <- p
printfn "%d: %d" N n
```

While-løkke

```
let mutable m = 1
let mutable n = 1
let mutable i = 3
let N = 5
while i <= 5 do
  let p = m + n
  m <- n
  n <- p
  i <- i + 1;
printfn "%d: %d" N n
```

Tupler

```
$fsharpi
...
> let a = (1, 1.0);;
val a : int * float = (1, 1.0)
```

Produktype

Funktioner til at indicerer i par

```
> printfn "%A %A" (fst a) (snd a);;
1 1.0
val it : unit = ()
```

Parentes unødvendig men anbefalelses

```
> let b = 1, "en", '\049'
val b : int * string * char = (1, "en", '1’)
```

Venstre side af en binding kan have navngivne tuple-elementer

```
> let (b1, b2, b3) = b;;
val b3 : char = '1'
val b2 : string = "en"
val b1 : int = 1
```

Hele typen - ikke enkelt - elementer kan være mutérbare

```
> let mutable c = (1,2)
- c <- (2,3)
- printfn "%A" c;;
(2, 3)
val mutable c : int * int = (2, 3)
val it : unit = ()
```

Fibonacci

For-løkke

```
let mutable m = 1
let mutable n = 1
let N = 5
for i = 3 to N do
  let p = m + n
  m <- n
  n <- p
printfn "%d: %d" N n
```

While-løkke

```
let mutable m = 1
let mutable n = 1
let mutable i = 3
let N = 5
while i <= 5 do
  let p = m + n
  m <- n
  n <- p
  i <- i + 1;
printfn "%d: %d" N n
```

Tupple + for-løkke

```
let mutable pair = (1,1)
let N = 5
for i = 3 to N do
  pair <- (snd pair, fst pair + snd pair)
printfn "%d: %d" N (snd pair)
```

```
let fib N =
  if N < 3 then 1
  else
    let mutable pair = (1,1)
    for i = 3 to N do
      pair <- (snd pair, fst pair + snd pair)
    snd pair
let N = 5
printfn "%d: %d" N (fib N)
```

Moduler og biblioteker

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

Program

```
let fib N =
  if N < 3 then 1
  else
    let mutable pair = (1,1)
    for i = 3 to N do
      pair <- (snd pair, fst pair + snd pair)
    snd pair
```

```
let N = 5
printfn "%d: %d" N (fib N)
```

Signatur (.fsi)

```
module Library

/// Calculate the n'th Fibonacci number
val fib : int -> int
```

Application (.fsx)

```
let N = 5
printfn "%d: %d" N (Library.fib N)
```

Implementation (.fs)

```
module Library

let fib N =
  if N < 3 then
    1
  else
    let mutable pair = (1,1)
    for i = 3 to N do
      pair <- (snd pair, fst pair + snd
pair)
    snd pair
```

Biblioteksvarianter

Åbning

Pas på namespace polution

```
open Library

let N = 5
printfn "%d: %d" N (fib N)
```

Filsuffikser

.fsx – scriptfil
.fsscript – scriptfil
.fs – implementationsfil
.fsi – signaturfil
.dll – oversat bibliotek
.exe – oversat og linket program

Uden implementationsfil

Sammenblanding af signatur og implementation

```
$ fsharpc -a library.fs
$ fsharpc -r library.dll application.fsx
```

Implementationsfil giver adgangskontrol

empty.fsi

```
module Library
// This file is intentionally empty
```

Oversættelse:

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

```
.../applicationOpen.fsx(4,21): error FS0039: The value or constructor 'fib' 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.?

Decimal til Binær

Program (.fsx)

```
/// Convert a non-negative integer into its
/// binary form. E.g., dec2bin 3 =  "0b11"
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
let N = 116
printfn "%d_10 = %s_2" N (dec2bin N)
```

Implementation (.fs)

```
module convert

/// Convert a non-negative integer into its
/// binary form. E.g., dec2bin 3 =  "0b11"
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
```

Application (.fsx)

```
open convert

let N = 116
printfn "%d_10 = %s_2" N (dec2bin N)
```

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)

```
open convert

printfn "Black-box testing of dec2bin.fsx"
printfn "  %5b: n < 0" (dec2bin -1 = "Illegal value")
printfn "  %5b: n = 0" (dec2bin 0 = "0b0")
printfn "  %5b: n = 1" (dec2bin 1 = "0b1")
printfn "  %5b: n = 2" (dec2bin 2 = "0b10")
printfn "  %5b: n = 10" (dec2bin 10 = "0b1010")
printfn "  %5b: n = 11" (dec2bin 11 = "0b1011")
```

```
$ fsharpc -a dec2bin.fs
$ fsharpc -r dec2bin.dll dec2binBlackTest.fsx
$ mono dec2binBlackTest.exe
Black-box testing of dec2bin.fsx
  true: n < 0
  true: n = 0
  true: n = 1
  true: n = 2
  true: n = 10
  true: n = 11
```

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

```
module convert

/// Convert a non-negative integer into its
/// binary form. E.g., dec2bin 3 =  "0b11"
let dec2bin n =
  if n < 0                          (* 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	false			-> Branch 2
	2	n = 0			n>=0
	2a	true	0	"0b0"	
	2b	false			-> Branch 3
	3	v > 0			n>0
	3a	true	1	"0b1"	1 or more
	3b	false			0 times, impossible.

White-box (unit) testing

Unit	Branch	Condition	Input	Expected output	Comment
dec2bin	1	n < 0			
	1a	true	-1	"Illegal value"	
	1b	false			-> Branch 2
	2	n = 0			n>=0
	2a	true	0	"0b0"	
	2b	false			-> Branch 3
	3	v > 0			n>0
	3a	true	1	"0b1"	1 or more
	3b	false			0 times, impossible.

```
open convert

printfn "White-box testing of dec2bin.fsx"
printfn "  Unit: dec2bin"
printfn "    %5b: Branch 1a" (dec2bin -1 = "Illegal value")
printfn "    %5b: Branch 2a" (dec2bin 0 = "0b0")
printfn "    %5b: Branch 3a" (dec2bin 1 = "0b1")
```

```
$ fsharpc -a dec2binWhite.fs
$ fsharpc -r dec2binWhite.dll dec2binWhiteTest.fsx
$ mono dec2binWhiteTest.exe
White-box testing of dec2bin.fsx
  Unit: dec2bin
     true: Branch 1a
     true: Branch 2a
     true: Branch 3a
```

DIKU Bits

TUESDAY LECTURES
BLOCK 1, 2019

24 SEPTEMBER

Pocket-Size Life Quality: Are You Ready for a Call?

Katarzyna Wac

Associate professor in the Human-Centred Computing section at DIKU

12.15 - 13.00 in Small UP1
Read more at diku.dk/diku-bits