Programmering og Problemløsning

3.1: Præcedens, association og virkefelter

Nøglekoncepter

- Fortolker vs. oversætter
- Talsystemer (decimal, binær, octal, hexadecimal)

- Heltal, flydende tal, tegn, strenge
- Typer og operatorer

Type	int	float	char	string	float	float
Tre	3	3.0	‘3’	“3”	3e0	3.0e0

Operatorer og præcedens

Operatorer og typer

3 + 4

3.0 + 4.0

~~3 + 4.0~~

5 / 2

5 % 2

2 * (5 / 2) + 5 % 2

2.0 ** 3.0

pown 2 3

"hej " + "med " + "dig"

Præcedens og association

Operator	Associativity	Description
`+<expr>, -<expr>, ~~~<expr>`	Left	Unary identity, negation, and bitwise negation operator
`f <expr>`	Left	Function application
`<expr> ** <expr>`	Right	Exponent
`<expr> * <expr>, <expr> / <expr>, <expr> % <expr>`	Left	Multiplication, division and remainder
`<expr> + <expr>, <expr> - <expr>`	Left	Addition and subtraction binary operators
`<expr> ^^^ <expr>`	Right	bitwise exclusive or
`<expr> < <expr>, <expr> <= <expr>, <expr> > <expr>, <expr> >= <expr>, <expr> = <expr>, <expr> <> <expr>, <expr> <<< <expr>, <expr> >>> <expr>, <expr> &&& <expr>, <expr> ||| <expr>` ,	Left	Comparison operators, bitwise shift, and bitwise 'and' and 'or'.
`<expr> && <expr>`	Left	Boolean and
`<expr> || <expr>`	Left	Boolean or

Operatorer og præcedens

Operatorer og typer	Præcendens og association	Typecasting	Unære operatorer
3 + 4	exp 0.0	float 3	2 - 3
3.0 + 4.0	exp 1.0	int 3.2	-3
3 + 4.0	exp 0.0 + 1.0	int 3.6	char (int 'c' + -int 'a' + int 'A')
5 / 2	2.0 **(3.0 ** 4.0)	int (3.2 + 0.5) = 3	char (int 'c' -int 'a' + int 'A')
5 % 2	(2.0 / 3.0)/ 4.0	int (3.6 + 0.5) = 4	
2 * (5 / 2) + 5 % 2		int 'a'	
2.0 ** 3.0		int 'A'	
pown 2 3		char 65	
"hej " + "med " + "dig"		char (int 'c' - int 'a' + int 'A')	

String slicing,boolske værdier og operatorer

Slicing	Boolske værdier og operatorer	Sammenligninger
"abcdefghijkl".[1] = 'b'	true = 1	3 < 4
"abcdefghijkl".[1..4] = "bcde"	false = 0	3 > 4
"abcdefghijkl".[..4] = "abcde"	a && b	3 <> 4
"abcdefghijkl".[4..] = "efghijkl"	a || b	3 = 4
"abcdefghijkl".Length = 12	not a	~~not 3 = 4~~
"abcdefghijkl".[0..11] = "abcdefghijkl"		not (3 = 4)

Bindinger af værdier

verbose syntax

```
let name = "World” in do printfn "Hello %A" name
```

Lightweight syntax

```
let name = "World”
do printfn "Hello %A" name
```

Optional ‘do’

```
let name = "World”
printfn "Hello %A" name
```

Sekvenser

```
let name = "World" in do printfn "Hello %A" name; do printfn "Goodbye %A" name
```

Nøgleord kan ikke bruges som navne

Type	Keyword
Regular	`abstract`, `and`, `as`, `assert`, `base`, `begin`, `class`, `default`, `delegate`, `do`, `done`, `downcast`, `downto`, `elif`, `else`, `end`, `exception`, `extern`, `false`, `finally`, `for`, `fun`, `function`, `global`, `if`, `in`, `inherit`, `inline`, `interface`, `internal`, `lazy`, `let`, `match`, `member`, `module`, `mutable`, `namespace`, `new`, `null`, `of`, `open`, `or`, `override`, `private`, `public`, `rec`, `return`, `sig`, `static`, `struct`, `then`, `to`, `true`, `try`, `type`, `upcast`, `use`, `val`, `void`, `when`, `while`, `with`, and `yield`.
Reserved	`atomic`, `break`, `checked`, `component`, `const`, `constraint`, `constructor`, `continue`, `eager`, `fixed`, `fori`, `functor`, `include`, `measure`, `method`, `mixin`, `object`, `parallel`, `params`, `process`, `protected`, `pure`, `recursive`, `sealed`, `tailcall`, `trait`, `virtual`, and `volatile`.
Symbolic	`let!`, `use!`, `do!`, `yield!`, `return!`, `|`, `->`, `<-`, `.`, `:`, `(`, `)`, `[`, `]`, `[<`, `>]`, `[|`, `|]`, `{`, `}`, `'`, `#`, `:?>`, `:?`, `:>`, `..`, `::`, `:=`, `;;`, `;`, `=`, `_`, `?`, `??`, `(*)`, `<@`, `@>`, `<@@`, and `@@>`.
Reserved symbolic	`~` and `` ` ``

Table 6.1: Table of (possibly future) *keywords* and symbolic keywords in F#.

Virkefelter (scope)

Navne (i yderste virkefelt) kan ikke overskrives

```
let name = "World"
let name = "Jon"
do printfn "Hello %s" name
```

Virkefelter via parenteser

```
let greeting = "Hello"                          0
let name = "Jon"
do printfn "%s %s" greeting name
(
    let name = "Anders"                         1
    do printfn "%s %s" greeting name
)
do printfn "%s %s" greeting name
```