

# F# - Funktionell programmering i Visual Studio

Pär Nordström  
Valtech

# I tidernas begynnelse (eller strax efter... 60-talet)

Algol

Lisp

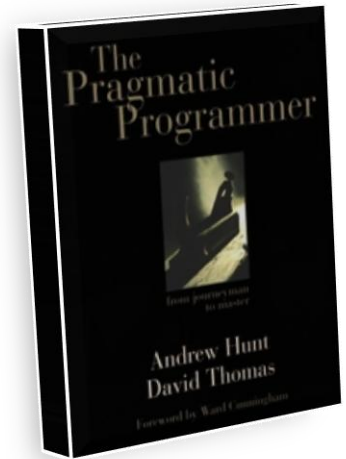
...till idag



- **Multiparadigm**
  - Funktionellt
  - Imperativt, objektorienterat
- **Stark typat**
  - Type inference
- **Kompilerat**

# Varför?

- ***”Lär dig ett nytt språk per år”***
  - Andy Hunt, Dave Thomas
  - The Pragmatic Programmer
- **Formelltunga eller formelcentrerade domäner**
  - Bank/Finans
  - Forskarvärlden
- **Parallell programmering**
  - Mycket gratis i språket
- **Fullvärdig medlem i VS 2010**



# Syntax - rekursion

```
let rec factorial n =  
  match n with  
  | 0 -> 1  
  | _ -> n * factorial (n - 1)
```

```
printfn "Factorial = %d" (factorial 5 )
```

```
val factorial : int -> int
```

```
| Factorial = 120  
| val it : unit = ()
```

# Syntax - listor

```
let rec quicksort list =  
  match list with  
  | []          -> []  
  | head :: tail -> quicksort (List.filter (fun e -> e < head) tail) @  
                      [head] @  
                      quicksort (List.filter (fun e -> e >= head) tail)
```

```
let a = [9;13;23;3;6;8;2;3;5;3;7;8;3;1000;2;47]  
printfn "Unsorted : %A\nSorted : %A" a (quicksort a)
```

```
Unsorted : [9; 13; 23; 3; 6; 8; 2; 3; 5; 3; 7; 8; 3; 1000; 2; 47]  
Sorted   : [2; 2; 3; 3; 3; 3; 5; 6; 7; 8; 8; 9; 13; 23; 47; 1000]
```

```
val a : int list = [9; 13; 23; 3; 6; 8; 2; 3; 5; 3; 7; 8; 3; 1000; 2; 47]
```

# Syntax - pipeline



```
let (|>) x f = f X
```

```
[1;2;3] |> List.map (fun x -> x*x*x)
```

```
List.map (fun x -> x*x*x) [1;2;3]
```

```
val it : int list = [1; 8; 27]
```

# Syntax - async



```
let isPrime (n:int) =  
    let bound = int (System.Math.Sqrt(float n))  
    seq {2 .. bound} |> Seq.exists (fun x -> n % x = 0) |> not
```

```
let primeAsync n =  
    async { return (n, isPrime n) }
```

```
let primes m n =  
    seq {m .. n}  
    |> Seq.map primeAsync  
    |> Async.Parallel  
    |> Async.RunSynchronously  
    |> Array.filter snd  
    |> Array.map fst  
  
primes 100000 100200  
    |> Array.iter (printfn "%d")  
  
1000003  
1000033  
1000037  
1000039  
1000081  
1000099  
1000117 1001953  
1000121 1001977  
1000133 1001981  
1000151 1001983  
1000159 1001989  
val it : unit = ()
```

## Mer info...



- [cs.hubfs.net](http://cs.hubfs.net)
- [msdn.microsoft.com/fsharp](http://msdn.microsoft.com/fsharp)