

COMPSCI 734 – Recap F# 335

Radu Nicolescu

Department of Computer Science
University of Auckland

6 Mar 2019

- ① Factorial
- ② Compiling
- ③ Main method
- ④ Pipelines and Compositions

Factorial – F# Naïve

- Naïve imperative solution – “impure”

```
1  let fact n:int =  
2      let mutable m = n  
3      let mutable f = 1  
4      while (m >= 1) do  
5          f <- f * m  
6          m <- m - 1  
7      f  
8  printfn "Imperative: %A" (fact 5)
```

- Naïve recursive solution – “pure” but inefficient

```
1  let rec fact ' n =  
2      if n <= 0 then 1  
3      else (fact ' (n-1)) * n  
4  printfn "Recursive: %A" (fact ' 5)
```

Factorial – F# Tail recursive

- Tail recursive solution – “pure” and efficient

```
1 let fact ' ' n =  
2     let rec fact_tailrec n acc =  
3         if (n <= 0) then acc  
4         else fact_tailrec (n-1) (n*acc)  
5     fact_tailrec n 1  
6  
7 printfn "Tail Recursive: %A" (fact ' ' 5)
```

Factorial – F# Tail recursive – reverse engineered in C#

- Tail recursive solution – “pure” and efficient (const stack size)

```
1 public static class Factorial {  
2     internal static int fact_tailrec (int n, int acc)  
3         while (n > 0) {  
4             acc = n * acc;  
5             n = n - 1;  
6         }  
7     return acc;  
8 }  
9 public static int fact (int n) {  
10     return Factorial.fact_tailrec (n, 1);  
11 }  
12 }
```

- The main method with printfn was omitted

Compiling – F#

- Linqpad – a lightweight interactive editor, excellent for learning and developing small snippets (free version)
<http://www.linqpad.net/>
- F# command-line interpreter FSI
- F# command-line compiler FSC
- F# compiler and batch (.BAT, .CMD) or shell (.SH) script file

```
1  if exist factorial.exe del factorial.exe
2  fsc factorial.fs
3  factorial.exe
4  pause
```

Main method – F#

```
1 open System
2
3 ...
4
5 [<EntryPoint >]
6 let Main args =
7     try
8         Console.WriteLine ("... ProcessorCount = {0}",
9                             Environment.ProcessorCount)
10
11     with
12     | ex ->
13         Console.WriteLine ("*** {0}", ex.Message)
14     0
```

Pipeline – F#

- Pipeline usage

```
1 let z = x |> f |> g |> h // let z = h (g (f x))
```

- Pipeline definition

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

Composition – F#

- Composition usage

```
1 let h = f >> g // let h = g << f  
2 let y = h x // let y = g (f x)
```

- Composition definition

```
1 let (>>) f g = (fun x -> g (f x))
```

- The keyword **fun** introduces a **lambda** (anonymous inline function)