ITT8060
Advanced Programming

In F#

Juhan Ernits, Hendrik Maarand, Ian Erik Varatalu

Department of Software Science

Welcome to Advanced Programming (in F#)!

- Course team:
  - Juhan Ernits
  - Hendrik Maarand
  - Ian Erik Varatalu
- Course web page
  - https://fsharp.pages.taltech.ee
  - Contact: juhan.ernits@taltech.ee
  - Online meetings: Teams team (with Taltech Uni-ID), Team ID for joining **wir5fe1**:
    https://teams.microsoft.com/l/team/19%3A189vuW9lKcwVkPySM8d-NsIpnRNBXvzuDVIRplPYjLM1%40thread.tacv2/conversations?groupId=e5089253-ad39-4288-b2a1-c3ae878f9b30&tenantId=3efd4d88-9b88-4fc9-b6c0-c7ca50f1db57

    Moodle (gradebook, links to videos, coursework feedback, ...), enrollment key **itt8060-2024**: https://moodle.taltech.ee/enrol/instances.php?id=33950

Textbooks

- Main textbook
  - Michael R. Hansen and Hans Rischel: Functional Programming using F# (paper copies in libraries and online access from Taltech network: Functional Programming Using F# (cambridge.org)
- Additional textbook
  - Tomas Petricek with Jon Skeet: Real-world functional programming with examples in F# and C#
    - 10 copies at Taltech: http://tallinn.ester.ee/record=b2780259~S1*eng
    - Several copies available in Tartu
- Additional textbook
  - Don Syme: Expert F# 3.0 and Expert F# 4.0
    - 5 copies at Taltech: http://tallinn.ester.ee/record=b2994544~S1*eng
    - Several copies available in Tartu
- More materials at https://fsharp.org/learn/

Structure of the course

- The course runs for 16 weeks

- Lectures
  - Room U05-103, lectures get recorded
- Lab sessions
  - Rooms
    - Wed 12:00: ICT-122 (IVSM (in English), we will record broadcast sessions)
    - Thu 8:00: ICT-401 (IAPM (in Estonian))

Structure of the assessment

- Coursework 45% of the final mark
  - 9 courseworks, each counting for 5%. Courseworks are mapped to concepts.
  - The coursework should be your own work.
  - You should be able to explain your work to the lab assistant upon request.
  - Your mark will be cancelled if you are not able to explain your own solutions to courseworks.
  - There may be some bonus courseworks available.
- An in class test in week 9, 5% of the final mark. (October 30, 2024)
  - You need to be there to get the 5%!
  - An indication of your progress.
- Exam 50% of the final mark
  - Written
  - You need to get at least 50% of exam total in order to pass.
    - (Strictly enforced)
    - Exam times:
      - Wednesday, Jan 8, 2025. 11:00 Tallinn, Tartu.
        Wednesday, Jan 15, 2025. 11:00 Tallinn
        Wednesday, Jan 22, 2025. 11:00 Tallinn

Advanced Programming (in F#)

You can all write programs!

- What is your first programming language?

Elixir
Lean
Clojure
C++
C#
C
Javascript
Python
PHP
Objective C
Erlang

Swift
F#
Haskell
OCaml
Scala
Idris
Whitespace
Prolog
TypeScript
Kotlin
Agda

Why F# of many functional languages

- Kotlin
- Scala
- Elixir
- Erlang
- Common Lisp (Emacs)
  - Many dialects, Clojure, Racket, Scheme, etc
- Haskell
- Agda
- Idris
- Ocaml
- C++ STL
- ...

F#

- F# is an *industrially supported* **functional first** .Net language
  - Belongs to eagerly evaluated ML family of languages
  - Bears similarities with Ocaml
- Allows easy integration into existing .Net projects.
- F# is well designed (did you read the paper by Don Syme?)
  - The Early History of F# (acm.org)

Some concepts touched upon in ITT8060

- functions and modules including higher order functions
- pipelines and composition
- lists, arrays, sequences
- pattern matching
- active patterns
- type inference
- recursive functions including tail recursion
- quotations
- record types, discriminated union types
- option types
- units of measure
- object programming
- asynchronous programming
- computation expressions
- type providers

Sample quote

- F# was the first language to introduce an async modality to allow the localized reinterpretation of the existing control constructs of the language. This meant that converting a piece of code from synchronous to asynchronous involved nothing more than wrapping async { ... } around the code and marking up the await points (let! in F#). This directly influenced the async/await mechanism added to C# 5.0 in 2012. the F# version was first presented to the C# designers in 2007 and many discussions were held in between. The C# async/await feature has been influential on TypeScript, Kotlin, Python 3.5, Java, JavaScript and other languages.

Don Syme, The Early History of F# (acm.org), page 75:50

Imperative programming style

```
IEnumerable<string> GetExpensiveProducts() {
    List<string> filteredInfos = new List<string();
    foreach(Product product in Products) {
        if(product.UnitPrice > 75.00M) {
            filteredInfos.Add(String.Format("{0} - ${1}",
               product.ProductName, product.UnitPrice));
        }
    }
    return filteredInfos;
}
```

Imperative programming style

```
IEnumerable<string> GetExpensiveProducts() {
    List<string> filteredInfos = new List<string();
    foreach(Product product in Products) {
        if(product.UnitPrice > 75.00M) {
            filteredInfos.Add(String.Format("{0} - ${1}",
               product.ProductName, product.UnitPrice));
        }
    }
    return filteredInfos;
}
```

State changing operations

Object oriented approach involves thinking about collections of objects that pass messages

Declarative programming style

```
IEnumerable<string> GetExpensiveProducts() {
    return from product in Products
         where product.UnitPrice > 75.0M
            select String.Format("{0} - ${1}",
               product.ProductName, product.UnitPrice);
        }
```

Declarative programming style

```
IEnumerable<string> GetExpensiveProducts() {
    return from product in Products
        where product.UnitPrice > 75.0M
            select String.Format("{0} - ${1}",
                product.ProductName, product.UnitPrice);
    }
```

Declarative style focuses on what a solution is.

Some advantages:

- Fast prototyping based on abstract concepts
- More advanced applications are within reach
- Supplement modelling and problem solving techniques
- Execute in parallel on multi-core platforms

Example: convenient parallelisation

```
var updated =
  from m in monsters
  let nm = m.PerformStep()
  where nm.IsAlive select nm;
```

LINQ

Example: convenient parallelisation

```
var updated =
  from m in monsters
  let nm = m.PerformStep()
  where nm.IsAlive select nm;
```

LINQ

```
var updated =
 from m in monsters.AsParallel()
 let nm = m.PerformStep()
 where nm.IsAlive select nm;
```

PLINQ

Course goals

- To give a generalised perspective to programming.
- To give an understanding how to think and program functionally and develop new skills for writing well structured code.
- To identify problems and domains that lend themselves to be thought about in functional ways.
- Functional techniques are now commonplace in mainstream programming languages.

Course goals cont.

- To show that real world business and scientific computing tasks often have a natural functional structure.
- To show how to test functional programs.
- To give an overview of various applied techniques, such as asynchronous and parallel programming in the functional context.

Why I use F#?

// Wrote this code yesterday:

```
[<Literal>]
let eventUri=__SOURCE_DIRECTORY__ + "\\events24s.json"
type Event = JsonProvider<eventUri>
let event = Event.Load(eventUri)

let eventList = event.Data |> Array.toList |> List.map (fun j -> j.Schedule |> Array.toList) |> List.collect id

printfn "\"alg_kp\";\"lopp_kp\";\"isikukood\";\"eesn\";\"peren\";\"fk_ruum_id\";\"tunniplaan\";\"ainekood\";\"pohjus\";"
for r in eventList do
    if (roomIds.Contains r.RoomId) && (r.Start > startTime) then
        let re = RegularExpressions.Regex($"[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9]",RegexOptions.NonBacktracking)
        let m = re.Match((r.Subject))
        printfn "%A;%A;\"isikk\";\"eesn\";\"perekn\";%A;\"jah\";%A;\"pohjus\";"
          (r.Start.ToString("yyyy-MM-dd HH:mm:ss"))
          (r.End.ToString("yyyy-MM-dd HH:mm:ss"))
          (roomMap (getRoomNo r.RoomId).RoomNo)
          //((r.Content))
          (if m.Success then
            m.Value
           else
            "Tunniplaani sundmus")
    else
        ()
```

Can anything exciting be done in F#?

- E.g. worlds fastest regular expression engine RE# (Ian)
  - https://cs.taltech.ee/staff/iavara/regex
- The fastness argument is explained here:
  - https://arxiv.org/abs/2407.20479

A bit of history

- The model of computation in functional programming is the application of functions to arguments. **No side effects**

Introduction of λ –calculus around 1930 by Church and Kleene when investigating function definition, function application, recursion and computable functions. For example,

f(x) =x+2 is represented by λ x.x+2

Curry Howard isomorphism

- 1934 Haskell Curry observes that the types of the combinators could be seen as axiom-schemes for intuitionistic implicational logic.
- In 1958 he observes that a certain kind of proof system (Hilbert style deduction system), coincides on some fragment to the typed fragment of a standard model of computation known as combinatory logic.
- In 1969 Howard observes that a proof system referred to as natural deduction, can be directly interpreted in its intuitionistic version as a typed variant of the model of computation known as lambda calculus.

from Wikipedia

LISP

- Introduction of the type-less functional-like programming: language LISP was developed by McCarthy in the late 1950s.
  - Used for solving various AI problems

A bit of history cont.

- Introduction of the ”variable-free” programming language FP (Backus 1977), by providing a rich collection of functionals (combining forms for functions)
- Introduction of functional languages with a strong type system like ML (by Milner) and Miranda (by Turner) in the 1970s.

Some background of the SML family

- Standard Meta Language (SML) was originally designed for theorem proving Logic for Computable Functions (Edinburgh LCF) Gordon, Milner, Wadsworth (1977)
- High quality compilers, e.g.
  - Standard ML of New Jersey and
  - Moscow ML
- based on a formal semantics Milner, Tofte, Harper, MacQueen 1990 & 1997

Some background of the SML family

- SML-like systems (SML, OCAML, F#,. ..) have now applications far away from its origins
  - Compilers,
  - Artificial Intelligence,
  - Data analysis,
  - Web-applications, Financial sector,
  - iOS application development
  - Android application development …

- F# is a .net language:
  - The .NET Language Strategy | .NET Blog (microsoft.com)
  - Declarative aspects are sneaking into more ”main stream“ languages
- Often used to teach high-level programming concepts

Quick F# motivation pitch by Ian