ITT8060
Advanced Programming

In F#

James Chapman
Juhan Ernits
Wolfgang Jeltsch

Welcome to Advanced Programming (in F#)!

- Teachers:
  - James Chapman
  - Juhan Ernits
  - Wolfgang Jeltsch
- Course web page
  - http://courses.cs.ttu.ee/pages/ITT8060
  - Contact: itt8060@cs.ttu.ee
  - Forum for Q&A:
    Details will be available on the course web page

Textbooks

- Main textbook
  - Tomas Petricek with Jon Skeet: Real-world functional programming with examples in F# and C#
    - 10 copies at TUT: http://tallinn.ester.ee/record=b2780259~S1*eng
    - Several copies available in Tartu
- Additional textbook
  - Don Syme: Expert F#
    - 5 copies at TUT: http://tallinn.ester.ee/record=b2994544~S1*eng
    - Several copies available in Tartu
- Recommended additional reading
  - Michael R. Hansen and Hans Rischel: Functional Programming using F#

Structure of the course

- The course runs for 16 weeks
- Lectures
  - Room ICT-A1
- Lab sessions
  - Rooms
    - All groups: ICT-401
    - Additional seating available in: ICT-637

- ITT8060 is compulsory in the IVSM curriculum
- People not from IVSM curriculum: you need to contact itt8060@cs.ttu.ee to get a place. First come – first serve as long as you have fulfilled the prerequisites.
- The ones not with IVSM code who have not sent an e-mail may get their registrations rejected in ÕIS!

Structure of the assessment

- Coursework 45% of the final mark
  - 9 courseworks, each counting for 5%
  - 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.
  - You need to be there to get the 5%!
  - An indication of your progress.
- Exam 50% of the final mark
  - Written

Advanced Programming (in F#)

You can all write programs!

- What is your first programming language?

Java
C++
C#
C
Javascript
Python
PHP

F#
Haskell
OCaml
Scala
ML
Whitespace
Prolog

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

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

PLINQ

C++ STL language

```
tempate<int N>
constexpr int fac()
{
   return N*fac<N-1>();
}
```

- STL is a Turing complete compile time functional language

C++14

- Has functional features like automatic type inference, e.g.:

```
auto lambda = [](auto x, auto y) {return x + y;};
```

Course goals

- To give a generalised perspective to programming.
- To give an understanding how to think and program functionally and achieve 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.

Quick F# tutorial

A wind down with some history

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

A bit of history cont.

- Introduction of the type-less functional-like programming: language LISP was developed by McCarthy in the late 1950s.

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,
  - Web-applications, Financial sector,
  - iOS application development
  - Android application development …

- F# is now integrated in the .net environment
- Declarative aspects are sneaking into more ”main stream languages”
- Often used to teach high-level programming concepts