**1. Course Description**

**1.1. Title of a Course**

Functional and Logic Programming

**1.2. Pre-requisites**

The course presents new programming paradigm, so no prior knowledge of programming is strictly required. Some knowledge of discrete mathematics and lambda-calculus would be a plus.

**1.3. Course Type**

Optional

**1.4. Abstract**

The course presents two programming paradigms: functional programming and logic programming. Most of the attention is given to functional programming, as more useful in practice. The importance of learning other programming paradigms cannot be underestimated: it helps students to look at problems from different viewpoints, to attempt different ways of problem decomposition, as well as to use more appropriate tools (eg. functional and logic programming languages) for some of the problems.

Students learn the mathematical basics and underlying algorithmic model of functional and logic languages, thus understanding deeper relationship between mathematical models and programming languages and their semantics. Most importantly, they also get experience in practical functional programming using F# programming language, which gives them useful practical tool for many data processing tasks.

The course is designed for students of the bachelor program "Software Engineering" at the Faculty of Computer Science, HSE.

**2. Learning Objectives**

* Students should get practical experience using most widely used functional and logic programming languages: F# and Prolog
* Students will understand different programming paradigms and the relationship between programming paradigm and underlying mathematical computational model
* Students will understand different approaches to solving problems: functional decomposition and declarative programming.

**3. Learning Outcomes**

After taking the course, student should be able to:

* Use functional/logic programming languages for solving practical problems in the areas where it is appropriate
* Identify those problems and estimate the appropriateness of using functional/logic programming; design software systems using multi-paradigm approach
* Understand the relationship between first-order predicate logic / lambda-calculus with programming languages and the process of computation
* Use more declarative and pure programming style with higher order abstractions, effectively use functional features of modern programming languages (eg. LINQ).

**4. Course Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Topic | Total hrs | In-class hrs | | Self-study |
| Lectures | Seminars/Labs |
| **Module 1 (32 hrs**) | | | | | |
| 1 | Introduction to Functional Programming | 16 | 4 | 2 | 10 |
| 2 | Algebraic Data Types – Lists and Trees | 20 | 4 | 5 | 11 |
| 3 | Lambda-Calculus as a Computational Model | 13 | 2 | 1 | 10 |
| 4 | Lambda-Calculus as a Programming Language | 12 | 1 | 0 | 11 |
| 5 | Functional Programming Techniques | 16 | 2 | 4 | 10 |
| 6 | Monads, Metaprogramming, Parallel and Async Programming | 18 | 3 | 4 | 11 |
| **Module 2 (32 hrs)** | | | | | |
| 7 | Object-Oriented and Imperative Features of F#. | 11 | 1 | 2 | 8 |
| 8 | Functional Aspects of Modern Programming Languages | 10 | 2 | 1 | 7 |
| 9 | Overview of Popular Functional Programming Languages: Haskell, Lisp, Erlang | 11 | 2 | 1 | 8 |
| 10 | Type Providers and Data Processing in F# | 11 | 1 | 2 | 8 |
| 11 | Introduction to Logic Programming | 14 | 4 | 2 | 8 |
| 12 | Resolution. Definite Clause Logic. Logic Programming with and without Negation using SLD Resolution | 10 | 2 | 0 | 8 |
| 13 | Logic Programming Techniques. Solving Logical Problems. | 14 | 2 | 4 | 8 |
| 14 | Typing in Functional and Logic Programming. Programming Language Semantics for Functional and Logic Languages | 10 | 2 | 0 | 8 |
|  | **Total:** | **190** | **32** | **32** | **126** |

**5. Reading List**

**5.1. Required.**

* Д.Сошников. Функциональное программирование на языке F#. – М.: ДМК Пресс, 2011.
* Братко И. Алгоритмы искусственного интеллекта на языке Prolog. – М.: Вильямс, 2004.
* Pickering, R. F# Succinctly, 2014. -- <https://www.syncfusion.com/ebooks/fsharp>
* Clifton, M. Imperative to Functional Programming Succinctly, 2014. -- <https://www.syncfusion.com/ebooks/imperative>
* Harrison, J. Introduction to Functional Programming. Lecture Notes, Cambridge University, 1997. -- <https://www.cl.cam.ac.uk/teaching/Lectures/funprog-jrh-1996/>
* Petricek, T. Analyzing and Visualizing Data with F#. O’Reilly, 2015. <https://www.oreilly.com/programming/free/analyzing-visualizing-data-f-sharp.csp>
* F# Programming WikiBook: <https://en.wikibooks.org/wiki/F_Sharp_Programming>

**5.2. Optional.**

* Peyton-Jones, Simon. Implementing functional languages: a tutorial, Prentice-Hall, 1992. -- <https://www.microsoft.com/en-us/research/publication/implementing-functional-languages-a-tutorial/>
* Purely Functional Data Structures. Chris Okasaki. September 1996. CMU-CS-96-177. School of Computer Science. Carnegie Mellon University. Pittsburgh, PA - <https://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>
* E. Chailloux, P. Manoury, B.Pagano. Programming Objective Caml. O’Reilly. <https://caml.inria.fr/pub/docs/oreilly-book>, русский перевод: <http://shamil.free.fr/comp/ocaml/>
* Ulf Nilsson and Jan Mauszynski. Logic, Programming and Prolog (2nd edition). John Wiley & Sons Ltd, 1995. (<http://www.ida.liu.se/~ulfni/lpp>)

**6. Grading System**

Knowledge of students is assessed throughout the course using two homework assignments, mid-term test (after the first module) and final exam.

**7. Guidelines for Knowledge Assessment**

Students will be formally graded (in the range 1-10) on the following:

* Homework Assignments (H1, H2)
* Mid-Term Test (M)
* Final Exam (E)

The final grade will be calculated as E\*0,4+M\*0,3+(H1+H2)\*0,15.

**8. Special Equipment and Software Support**

* A projector for lectures
* Whiteboard with markers
* Internet access
* Computer class for Labs/Seminars with computers running Windows
* The following free software will be used in the course:
  + Visual Studio Code or Visual Studio 2015 Community Edition
  + GNU Prolog