CSCE 4430: Programming Languages - Fall 2016

Instructor: Paul Tarau, Professor - see my home page for contact info and office hours.

Grader: TBD - see his/her course page here.

Description and Objectives:
A comprehensive programming language course, with emphasis on programming paradigms and language processors - and some of their formal models like Predicate Logic and Lambda Calculus. Hands-on work with implementations of key concepts (recursion, inheritance, unification, backtracking, type inference, infinite and lazy data objects, threads. The course also provides a glimpse at salient features of modern object oriented languages and an overview of language implementation techniques, run-time systems, garbage collection, interpreters, compilers with emphasis on addressing and memory management in efficient procedural languages like C.

Syllabus (L1..Ln) indicate number of the lecture

Basics

* From Turing machines and Lambda calculus to today's programming languages L1
* Programming Paradigms: logic, functional, object oriented, imperative L1
* Language Processors: Interpreters and Compilers, Domain Specific Languages (DSLs) L1

Scripting and Web Programming Languages

* Javascript and JSON: some simple client-server Web programming architectures - L2
* Warm-up case study with a DSL: Javascript Minecraft Modding with ScriptCraft L2

Logic Programming Languages (Prolog)

* Intro to Prolog: Facts and Rules, Recursion, Lists and Compound Terms, Arithmetic L3
* Propositional and Predicate logic L3
* Unification and Horn Clause Resolution in Prolog L3
* Non-determinism and Backtracking L4
* Case study: solving cryptarithmetic puzzles L4
* Definite Clause Grammars, Parsing and Generation L5
- Combinatorial Generation in Prolog L6
- Problem solving with Prolog L7
- Runtime Code Generation: the dynamic database L8
- Higher order Predicates and Meta-interpreters L8

**Functional Programming Languages (Haskell)**

- Intro to Haskell L9
- Programming with Recursive Functions and Lists L9
- Higher Order Functions L10
- Working with Fold, Map, Zip L10
- Polymorphism, Type inference L11
- Recursive Data Types and Pattern Matching in Haskell L11
- List Comprehensions L12
- Lazy Evaluation, Computing with Infinite Lists L12
- Lambda Calculus, De Bruijn Indices L13
- Case study: Generating Lambda Terms in Prolog L14
- Simply-typed Lambda Terms, Type Checking vs. Inference L15
- Evaluation of Lambda Terms, Normalization, L18

**Object Oriented Programming (Java+Scala+Swift)**

- Static vs. Dynamic Aspects L19
- Classes, Instances, Objects L19
- Overview of Java 8 L20
- Overview of Scala L21
- Overview of Swift L22
- Collections, Iterators and IO operations L23
- Reflection and Serialization L23

**Low Level Imperative Programming (C)**

- Basics: assignment, function calls, lexical scoping, memory representations, stack and heap L24
- Implementing dynamic memory management and garbage collection L24
- Implementing high-level programming languages in C L24

**Concurrent Programming**

- Event driven programming, Coroutining, Futures, Multi-threading L25
Distributed Programming, Message Passing, Coordination L25

Future Trends in Programming Language Design L26

Prerequisites: Data Structures

Recommended books and online materials:

- Tucker & Noonan: Programming Languages, Principles and Paradigms, McGraw Hill
- The Art of Prolog by Sterling and Shapiro, MIT Press
- Doets & van Eijck: The Haskell Road to Logic, Math and Programming
- Harper’s draft PL book
- Java for Students: Douglas Bell & Mike Parr, Prentice Hall

Evaluation:

- 2 Individual Exams: 30%+30%
- Team Project and Assignments (groups of 2-3): 40%

Software, tutorials and related links:

- The Eclipse Open Software Development Platform
- SWI Prolog tutorial.
- Haskell compiler GHC
- Haskell tutorials.
- Java interpreter/compiler JDK

Outcomes:

1. Understand the concepts of programming languages, with emphasis on programming paradigms and language processors.
2. Have a practical understanding of commonalities and differences between major programming paradigms.
3. Understand the key object oriented, logic and functional programming concepts.
4. Understand the key concepts of event driven and concurrent programming.
5. Understand the use and implementation of modern programming language concepts like recursion, inheritance, reflection, unification, backtracking, type inference, infinite data objects, and threads.
6. Have some familiarity with domain-specific languages with emphasis on Internet
programming languages.