CSCE 4430: Programming Languages - Spring 2011

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

Teaching Assistant: TBD - see his/her home page for project submissions, contact info and office hours

E-mail : *t a r a u@cs.unt.edu*
WWW : http://www.cs.unt.edu/~tarau
Address: Department of Computer Science, University of North Texas, P.O. Box 311366, Denton, Texas 76203, USA
Phone : Tel : +1-940-565-2806, +1-940-565-2767
Fax : +1-940-565-2799

Description and Objectives:

An advanced programming language course, with emphasis on programming paradigms and language processors - and some of their formal models like Predicate Logic and Lambda Calculus and exhibiting actual implementations of key concepts (recursion, inheritance, unification, backtracking, type inference, infinite and lazy data objects, threads, event-driven and concurrent/distributed programming). 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
  - Models of Computation, Computability, Turing Equivalence L1
  - Evolution of Programming Languages L2
  - Programming Paradigms: logic, functional, object oriented, imperative L3
  - Language Specification: Syntax and Semantics L4
  - Language Processors: Interpreters and Compilers L5
- Logic Programming Languages (Prolog) L6
  - Unification and Horn Clause Resolution L7
  - Non-determinism and Backtracking with application to Problem Solving L8
  - Practical Prolog: IO, File Operations, GUI programming L9
  - Definite Clause Grammars L10
  - Parsing and Generation L11
  - Equivalence Preserving Program Transformations L12

    - Meta-Interpreters, Universal Machines L13
- Functional Programming Languages (Haskell)
    - Lambda Calculus and Recursion Theory L14
    - Higher Order Functions L15
    - Working with Fold/Unfold, Map, Zip L16
    - Lazy Evaluation, Computing with Infinite Lists L17
    - Polymorphism, Type inference, Type Classes L18
    - Emulating Imperative Programming with Monads L19
    - Monadic IO, Interactivity, File Operations L20
- Object Oriented Programming (Java and similar languages)
    - Types: Static vs. Dynamic Type Checking L21
    - Classes and Instances, Type vs. Implementation Inheritance L22
    - Reflection and Serialization L23
    - Object Oriented Exception Handling L24
    - Iterators and *yield/return* based coroutining constructs L25
- Low Level Imperative Programming (C and Go)
    - basics: assignment, function calls, lexical scoping, memory representations L26
    - implementing dynamic memory management and garbage collection L27
    - implementing high-level programming languages L28
- Concurrent Programming L29
    - Multi-threaded programming in Java and Prolog
    - Distributed Programming, Client/Server Message Passing and Coordination
    - Service Architectures and Interactivity
- Future trends in Programming Language Design L30

Prerequisites: mandatory (Data Structures)

Recommended books:

- A. Tucker & R. Noonan: Programming Languages, Principles and Paradigms, McGraw Hill
- R. Harper's draft PL book at: http://www.cs.cmu.edu/~rwh/plbook/
- Java for Students: Douglas Bell & Mike Parr, Prentice Hall
- The Art of Prolog by Sterling and Shapiro, MIT Press

Evaluation:

- 2 Individual Exams: 50%
- Team Project and Assignments (groups of 2-3): 50%

Resources:

- Haskell compiler GHC and Prelude library sources

- Java interpreter/compiler on silo and CAS network (java, javac), Java from www.javasoft.com
- Go: a very interesting minimalist language from Google – and its library
- Java based Prolog compiler
- The Eclipse Open Software Development Platform
- Programming Languages Popularity/Impact Index
- 99 Haskell Problems based on 99 Prolog Problems Various Haskell Coding Styles
- An Objective-C tutorial
- Programmers Competency Matrix
