Topics in Programming Languages **CISC 860*** **Fall 2014**

The Structure and Design of Programming Languages

An advanced course in the design and evaluation of programming languages: history and evolution of programming languages, relation to programming methodology; principles of programming language design, psychology of notation, uniformity, linearity and locality; objective criteria for the evaluation of programming languages with relation to design goals; case studies of specific languages. Students will undertake independent research studies and present results in the form of class seminars.

Text:

None.

References:

Wegner, *Milestones in the History of Programming Languages.*
Weinberg, *The Psychology of Computer Programming.*
Horowitz, *Programming Languages: A Grand Tour.*

Instructor:

J.R. Cordy

Prerequisites:

At least one undergraduate level course in the study of programming languages.
Some knowledge of automata theory and formal semantics (axiomatic or denotational).
Thorough knowledge of at least three distinct and preferably diverse modern programming languages.

Outline:

1. History of Programming Languages (3 weeks)

   A series of lectures and discussions led by the instructor on the early history and comparative study of programming languages, emphasizing the evolution of algorithmic notation and its relation to programming methodologies. This section of the course will be based on material from Horowitz's Programming Languages: A Grand Tour, in particular Wegner's paper Milestones in the History of Programming Languages. The section will end with proposals from students for new milestones in the spirit of Wegner from more recent history.

2. Design of Programming Languages (3 weeks)

   Lectures and discussions led by the instructor on the general principles of programming language design and the criteria by which a programming language can be objectively evaluated. This section will be based on relevant material from other disciplines including linguistics, the psychology of memory and perception, industrial engineering and mathematics as well as the psychology of computer programming. A primary reference for this section of the course will be Weinberg's Psychology of Computer Programming. The section will end with student presentations on specific design issues drawn from the literature.

3. Language Case Studies (3 weeks)

   Case studies of specific programming languages and language features with an aim to understanding the goals of the language and how the design attempts to meet those goals. The section will end with student case studies for more recent programming languages and features.

4. Student Project Seminars (3 weeks)

   The remainder of the course will consist of seminars led by individual students reporting the results of independent research projects in programming language structure and design. Projects may include new language designs, comparative or evaluative studies of languages for particular paradigms or domains, proposals for new features or paradigms, and so on.

Goals:

All too often programming languages are considered only from the point of view of vehicles to express algorithms for execution by a computer. In point of fact, computer programs are read more by people than by machines, and thus their major role is as a notation for communication between programmers. The goal of this course is foster the study of programming languages in this more fundamental role, with an aim to understanding the criteria by which programming languages can be designed and objectively evaluated.

Bibliography on Programming Language Structure and Design:

As part of the course, students will be expected to contribute to an ongoing annotated bibliography on programming language structure and design. Each member of the class will research and submit written reports on at least six new articles of significant interest in the area, assigned as two reviews for each part of the course. One of the two submitted in each case will be formally presented to the class for open discussion.

Sources for this research will include relevant scientific journals such as Communications of the ACM, ACM Transactions on Programming Languages and Systems, Proceedings of the SIGPLAN Symposia on Principles of Programming Languages, Proceedings of the IFIP Conference on System Implementation Languages, Proceedings of the ACM Conference on Language Design for Reliable Software, Proceedings of the ACM Conference on Language Issues in Software Environments, IEEE Transactions on Software Engineering, Proceedings of the ACM Conference on Artificial Intelligence and Programming Languages, Proceedings of the International Conference on Software Language Engineering, and so on.

Course Project:

Each member of the class will be expected to undertake a formal written independent language design project on some relevant aspect of programming language design and structure, and to make an oral presentation of their findings in the form of a seminar of approximately one hour's length. If the class is large then projects will be undertaken in teams of two. Projects involving significant original research will be encouraged, as will projects that consist of simply researching the literature and intelligently distilling key ideas for presentation.

Marking:

Marks will be assigned on the basis of a short midterm examination on the lecture material, bibliography contributions, oral and written presentation of the course projects and class participation. The midterm examination will be given in the first class following completion of the two initial lecture series by the instructor. Students will be expected to participate actively in seminars and class discussions.

Tentatively, marks will be divided in roughly the following proportions :

Midterm Examination	20%
Bibliography Contributions	20%
Course Project	50%
Class Participation	10%