

1. COS 301 Programming Languages Fall 2011

2. Credits and Contact Hours: 3 cr.

3. **Instructor:** Curtis Meadow 230 Neville Hall curtis.meadow@umit.maine.edu

4. **Textbook:** *Concepts of Programming Languages (9th ed)*. Robert Sebesta, Addison-Wesley 2009. ISBN 0-13-607347-6. The web site also has links to a substantial body of supplemental reading

5. Specific Course Information

5a. **Catalog Description:** This course examines the syntax, semantics and implementation of modern programming languages. Topics include the history of programming languages, BNF and context-free grammars, lexical analysis, operational semantics, typing systems, subprograms, control structures and support for abstraction. Languages from the imperative, functional, object-oriented and logic paradigms are examined. The course emphasizes critical understanding of the trade-offs in language design and the selection of languages for different programming domains. Students are expected to learn a language with which they have no previous experience.

5b. **Prerequisites:** COS 140 (Foundations of Computer Science) and COS 226 (Data Structures).

5c. **Required Course**

6a. Explicit Goals for the course:

- The student will be able to explain the major programming paradigms: imperative, object-oriented, functional and declarative.
- The student will be able to recognize and explain major programming language constructs.
- The student will be able to read BNF and regular grammars, regular expressions, and graphic representations of finite state machines and will be able to write at least simple grammars, regular expressions, and FSMs.
- The student will be able to analyze a computing problem and select a language or languages that are appropriate for solving the problem
- The student will be able to recognize and describe the major historical languages and identify their influence on modern languages
- The student will learn one additional programming language
- The student will be able to apply knowledge of programming language paradigms and programming constructs to learning new languages without formal instruction
- The student will learn basic professional written communication skills and improve skills already present.
- The student will be able to write a formal research paper following a specified format with title page, table of contents, list of figures, abstract, narrative, appendices and references.
- The student will be able to use the IEEE citation standard in writing,

6b. Student Outcomes Addressed

(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline. Students are introduced to context-free and regular grammars, finite-state automata, the fundamentals of lexical and syntactic analysis, static semantic analysis, algorithms for expression evaluation.

(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.

Students are expected to understand how various programming languages are used in different problem domains by understanding the trade-offs between expressivity, efficiency, readability and security. Students are expected to be able to apply this knowledge to select appropriate programming languages for solving particular problems.

(c) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs.

Students are expected to learn a new programming language and implement solutions to given problems in that language.

(f) An ability to communicate effectively with a range of audiences.

This is a writing intensive course; students write five papers covering aspects of a programming language and are expected to write in a formal, professional manner with appropriate citations and an annotated bibliography. First draft papers are reviewed with comments and students then rewrite the papers, correcting issues addressed by the comments.

(h) Recognition of the need for and an ability to engage in continuing professional development.

Students are expected to understand the historical development of programming languages from early beginnings to modern and emerging languages. Throughout the course the ability and need to learn new languages quickly is emphasized. Examples are presented in a number of languages with which students are not familiar.

7. Brief List of Topics

Programming Language Paradigms

Evolution of the Major Programming Languages

Syntactic and Semantic Analysis: BNF grammars, attribute grammars, regular grammars, regular expressions, parse trees, recursive descent parsing

Names, Binding and Scope

Data Types

Expression and Assignment Statements

Statement Level Control Structures

Subprograms and Functions

Implementing Subprograms

Abstract Data Types and Encapsulation Concepts

Object Oriented Paradigm

Logic and Declarative Languages