

COURSE DESCRIPTION

Dept., Number	CS 361	Course Title	Programming Languages and Implementation
Semester hours	4	Course Coordinators	Scharff, Badii

2004-2006 Catalog Description

Study of the principles of language design, semantics, and implementation. Emphasis will be put on formal syntax and interpretation, including an introduction to automata, lexical analysis, parsing, error diagnosis and recovery, and code generation. Students will work on several weekly exercises during the first weeks of the semester and on a substantial project that will apply the material covered in the second part of the term.

Textbook Faculty may choose the latest edition of one of the following:

R. Sethi; *Programming Languages, Concepts and Constructs*; Addison Wesley

A. Tucker and R. Noonan; *Programming Languages: Principles and Paradigms*;
McGraw Hill.

T. W. Pratt and M. V. Zelkowitz; *Programming Languages: Design and Implementation.*;
Prentice Hall.

and The following standard ML text is to be used by all sections:

Jeffrey D. Ullman; *Elements of ML Programming*, Prentice Hall

References

Papers on Aspect-oriented programming, comparisons between Java and C#, logic programming, and XML will be distributed.

Course Goals

Objective 1: To introduce students to new programming paradigms (including imperative, functional and logical) and the associated thought processes and specificities for each

Outcomes: Students will demonstrate the ability to:

- ♦ List and define the major programming paradigms (with features) and the associated thought processes for each
- ♦ Explain how language features contribute to programming practices
- ♦ Compare programming paradigms/languages ((e.g. compare Java with C++/C#)

Objective 2: To outline the language translation process

Outcomes: Students will demonstrate the ability to:

- ♦ Differentiate interpreters and compilers
- ♦ List the different phases (with their features) of the compilation process

Objective 3: To introduce students to low and high level topics in programming languages

Outcomes: Students will demonstrate the ability to:

- ♦ Read and modify grammars (BNF, EBNF, regular expressions, syntax diagrams)
- ♦ Differentiate abstract and concrete syntax
- ♦ Master different parsing algorithms
- ♦ Draw parse trees
- ♦ Give at least three examples how abstraction is implemented in modern languages

Objective 4: To improve Java programming skills and develop programming skills in other programming languages (including imperative, functional, and logical languages)

Outcomes: Students will demonstrate the ability to:

- ◆ Read language specification manuals
- ◆ Implement the scanner and parser of an experimental programming language using the Recursive Descent Parsing algorithm in Java
- ◆ Understand memory management
- ◆ Write small programs/functions in imperative (e.g. C), functional (e.g. SML/Haskell), and logical languages (e.g. Prolog)

Objective 5: To develop the ability to learn new programming languages and awareness in new trends in programming languages

Outcome: Students will demonstrate the ability to:

- ◆ learn and research new topics in programming languages independently

Prerequisites by Topic

Programming ability in one high level language (Pascal, Java, C or C++) at the level of the "experienced novice"; ability to write programs of a few hundred lines as specified by the instructor.

Students should be fully competent in developing data structures and simple algorithms as well as in analyzing the run-time complexity of simple algorithms, including recursive algorithms.

Major Topics Covered in the Course

- ◆ History of programming languages
- ◆ Overview of the language translation process (Interpreters and compilers, lexical analysis, syntactical analysis, semantic analysis, code generation, optimization)
- ◆ Language principles including both low and high level topics (Syntax and semantics, grammars, parse trees, language specifications, abstraction, and scanning and parsing algorithms)
- ◆ Memory management (Static, run-time and heap memories, parameter passing, garbage collection algorithms, pointers)
- ◆ Imperative programming (Variables and their scope, types and type checking, expressions, casting, parameters, memory management)
- ◆ Object-oriented programming (Inheritance, polymorphism, encapsulation, abstraction, field and method access, reflexivity, memory management)
- ◆ Functional programming in SML/Haskell (types, tuples, lists, recursion, high-order functions, introduction to program correctness)
- ◆ Logic programming with Prolog (Facts, clauses, SLD resolution, lists)
- ◆ New topics in programming languages (e.g. aspect-oriented programming, C#)

Laboratory Projects

Implementing a scanner and a parser for an experimental language which may be provided or designed by the students, depending on the instructor. Additionally, 2 to 5 smaller projects, depending on the instructor, will exemplify programming in different languages.