CSE 425S
Programming Systems and Languages

- *Formal semantics*
- *Language implementation*
- *Programming paradigms*

A systematic study of the principles, concepts, and mechanisms of computer programming languages: their syntax, semantics, and pragmatics; the processing and interpretation of computer programs; programming paradigms; and language design. Illustrative examples will be selected from a variety of programming language paradigms.

The study of languages is central to the computer science field. This course addresses key issues regarding language definition and implementation techniques. Formal specification of languages requires an understanding of methods for expressing their syntax (e.g., BNF, attribute grammars) and semantics (e.g., operational, denotational, and axiomatic). Language implementation deals with the run-time structures needed to support various language features such as data types, operations, flow of control constructs, exception handling, visibility rules, subprograms, concurrency, etc. Block structured languages (e.g., Ada, Java, C++) provide a vehicle for studying the evolution of modern language features and their implementation techniques. Functional programming (e.g., pure Lisp), logic programming (e.g., Prolog), and object-oriented programming (e.g., Smalltalk) are used to explore implementation requirements for alternate programming paradigms. Students entering this course must have considerable exposure to at least one major programming language and a basic knowledge of discrete mathematics, lambda calculus, and predicate calculus. Upon completion of this course students must be able to specify formally the syntax and semantics of simple programming languages and must be able to design interpreters for such languages.

Offered:	Every fall semester
Prerequisites:	CSE 132, CSE 240, and CSE 241
Credit:	3 units

August 20, 1995

Syllabus

TEXTBOOK

K. C. Louden, *Programming Languages—Principles and Practice*, 2003 Brooks/Cole, Thomson Learning Inc.

A. INTRODUCTION

1. Language design issues (Ch. 1-3)

B. SYNTAX SPECIFICATION

2. Context-free grammars, BNF, attributed grammars (Ch. 4)
3. Discussion

C. SEMANTIC MODELS

4. Operational model (Ch. 13.2)
5. ...
6. Denotational model (Ch. 13.3)
7. ...
8. Axiomatic model (Ch. 13.4-5)
9. ...
10. Discussion

Project 1. Language design, syntax, axiomatic semantics.

D. IMPLEMENTATION

11. Variables, representation, biding, tuples, garbage collection (Ch. 5)
12. Data types (Ch. 6)
13. Sequence control, exception handling, subprograms (Ch. 7-8)
14. ...
15. Abstract data types (Ch. 9)
16. ...
17. Object-oriented programming (Ch. 10)
18. ...
19. Discussion

Project 2. Denotational semantics, run-time system design.

E. CONCURRENT PROGRAMMING

20. Shared variables, message passing (Ch. 14)
21. Multitasking
22. Discussion

Project 3. Operational semantics, run-time system design.

F. ALTERNATE PARADIGMS

23. Functional programming, Lisp, ML, Miranda, Scheme, Actors (Ch. 11)
24. ...
25. Logic programming, resolution, unification, Prolog, Parlog (Ch. 12)
26. ...
27. Data flow
28. HyperMedia
29. Event-based programming
30. Discussion

Final Exam

Homework Format

A. Cover Page

- class
- project number
- project name
- date
- name

B. Language Syntax

- brief overview
- formal syntax
- additional constraints
- example

C. Language Semantics

- brief overview identifying the model type and the general modeling strategy
- definitions and notation
- formal model organized in some logical form

D. Interpreter Design

- brief overview of the design
- logical program organization
- pictorial representation of the data structures design
- principal operations on the data structures

E. Implementation

- pictorial representation of the code organization and its relation to the design
- documented code (Java or C++)