

## COURSE DESCRIPTION

**Department and Course Number**                      CMPS 450                      **Course Coordinator**    Mark G. Radle

**Course Title**                      Programming Languages                      **Total Credits**                      3

**URL**                                      www.cacs.louisiana.edu/~mgr/450

### **Current Bulletin Description:**

Programming Languages. Formal, functional, and practical issues of design and implementation of imperative, functional, and declarative languages; denotational semantics; data types and abstraction, control abstraction, separate compilation units, concurrency. Fa. Prereq: CMPS 440 with a grade of C or better.

### **Textbook:**

*Programming Languages: Principle and Practice Second Edition*, Kenneth C. Loudon, PWS Publishing Company, 2004.

References : Various readings on specific programming languages are provided.

### **Course Goals:**

1. To study programming language constructs and features.
2. To become familiar with language design principles.
3. To become familiar with the different semantic issues of programming languages.
4. To become competent in problem solving using different language paradigms.
5. To become proficient in implementing a lexical analyzer for a programming language based on its specification.
6. To become familiar with the principal methods of formal semantics: operational, denotational, and axiomatic.
7. To experience a diverse range of programming languages, constructs, and implementation issues.
8. To introduce students to programming language theory including predicate calculus, the lambda calculus, and semantic algebras.

### **Course Outcomes:**

1. Students will be knowledgeable of a wide range of programming language constructs and features.
2. Students will have an appreciation of the implications of language design principles.
3. Students will have experience problem solving using different language paradigms.
4. Students will be familiar with the principal methods of formal semantics.
5. Students will minimally competent in formal language theory.
6. Students will have experience in learning programming languages in a relatively short period of time.

### **Prerequisites by Topic:**

1. In-depth knowledge of at least one imperative programming language.
2. Basic data types, basic control structures, procedural abstraction, and data abstraction.
3. Recursion.
4. Data structures and algorithms.
5. Regular language and context free language theory

### Major Topics Covered in the Course:

1. Language design and implementation: Influences on language design; language design criteria and trade-offs. Compilation, interpretation. (2 hours)
2. Overview of the major programming languages: Environment, contributions, motivation for development. (1 hour)
3. Logic programming languages: Predicate calculus, Prolog, resolution (5 hours)
4. Semantics: Denotational semantics of expressions and assignment, conditional, and loop statements. Introduction to axiomatic and operational. (3 hours)
5. Variables and their attributes: Binding, scope (static and dynamic), lifetime, referencing environment, type. (2 hours)
6. Data types: Design and implementation of various data types, including primitive types, enumerated types, arrays, records, discriminated unions, and pointers. Type equivalence – structural, name. Type checking, type inference. (4 hours)
7. (2 hours)
8. Data abstraction and abstract data types  
Encapsulation, generics (2 hours)
9. Expressions and operators: Overloading, coercion, short-circuit operators, functional side effects. (1.5 hours)
10. Functional programming languages: Lambda-calculus, ML, Scheme (5 hours)
11. Statements: Design issues and implementation of assignment selection, repetition. (1.5 hours)
12. Subprograms: Design issues, implementation, overloaded, generic, local and non-local referencing environments, static scope, dynamic scope, side effects. (4 hours)
13. Parameter passing: Methods (pass by value, result, etc.), implementations, and functions as parameters. (2 hours)
14. Concurrency: Monitors, tasks, synchronization, CSP, Oocam. (3 hours)
14. Tests (2 hours)
15. Final Exam (2 hours)

### Laboratory projects (specify number of weeks on each) :

“Typical programming assignments include:”

1. Write a meta-circular interpreter for Scheme in Scheme. (5 weeks)
2. Write a program in language “A” that recognizes a legal program in language “B”. (semester long project)
3. Design a set of predicates in Prolog to solve a problem requiring a state space search. (3 weeks)

4. Implement a program that changes the functionality of a programming language construct. (5 weeks)

### **Oral and Written Communications**

Every student is required to submit at least \_\_0\_\_ written reports (not including exams, tests, quizzes, or commented programs) of typically \_\_0\_\_ pages and to make \_\_0\_\_ oral presentations of typically \_\_0\_\_ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

### **Social and Ethical Issues**

Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

N.A.

### **Theoretical Content**

Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

Finite automata, context-free grammars and type inferencing.

### **Problem Analysis**

Please describe the analysis experiences common to all course sections.

### **Solution Design**

Please describe the design experiences common to all course sections.

See laboratory projects above.