

Clemson University
Department of Computer Science
COURSE DESCRIPTION

CP SC 428 Design and Implementation of Programming Languages 3(3,0)

Current Catalogue Description

Overview of programming language structures and features and their implementation. Control and data structures found in various languages are studied. Runtime organization and environment and implementation models are also included. Preq: CP SC 231 and 360 with a C or better.

Course Type

Required for BS in Computer Science.
Elective for BS in Computer Information Systems.
Elective for BA in Computer Science.

Textbooks

Terrence W. Pratt and Marvin V. Zelkowitz. *Programming Languages: Design and Implementation*. 4th Ed. Prentice-Hall, 2001.
Kenneth C. Louden, *Programming Languages: Principles and Practice*, Thompson. 2003.

Course Objectives and Outcomes

1. To understand the basic aspects of various programming paradigms.
 - 1.1 Students will be able to discuss principles of the three major paradigms: functional, imperative, and logic.
 - 1.2 Students will understand how to implement OOPS and non-OOPS paradigms.
2. To be able to implement basic data and data structures in programming languages.
 - 2.1 Students will implement Boolean data type.
 - 2.2 Students will implement Integers data type.
 - 2.3 Students will implement Floating Point data type.
 - 2.4 Students will implement Strings data type.
 - 2.5 Students will implement Arrays composites.
 - 2.6 Students will implement Structures composites.
 - 2.7 Students will implement Objects composites.
3. To be able to implement stack and heap management in executing programs using various language features.

- 3.1 Students will implement local variables of the above types.
- 3.2 Students will implement pass-by-value parameter passing.
- 3.3 Students will be able to generate code for recursive functions.
- 4. To be able to discuss, compare, and contrast various language features implementations and the effects of the decision on program performance.
 - 4.1 Students will be able to discuss difference in imperative, functional, and logical paradigms.
 - 4.2 Students will be able to discuss the relationship of structures and objects in OO paradigm.
 - 4.3 Students will implement scope for compiler.
- 5. To participate in a capstone experience.
 - 5.1 Students will implement a small compiler of their own design.
 - 5.2 Students will develop their own test cases to verify that their compiler meets specifications.

Prerequisites by Topic

- 1. Experience with programming languages [CPSC 101, 102, 210, 212, 215].
- 2. Advanced data structures and algorithms [CPSC 212, 350]
- 3. Computer Architecture and Assembler-level programming [CPSC 330, 231]
- 4. Formal computability models: FSM, CFL [CPSC 350]

Major Topics Covered in the Course

This course is taught in a problem-learning style. The following topics are covered:

- 1. Linguistics. Semiotics: Syntax (lexical and syntactic structure; surface structure), Semantics (denotational and axiomatic); Pragmatic use (learning and using *gforth*).
- 2. Implementation of programming languages in 8 milestones.
- 3. Project management through 8 milestones.
- 4. General capstone issues: design and implementation of a large project.
- 5. Advanced programming structures: inheritance/variant records. Testing. Each student is responsible for implementing and testing (unit and system).

Class/Laboratory Schedule

Lecture: 3 hours/week

Course Contribution to Program Objectives

1. General Educational Objective 1: programming languages are the fundamental mechanism for describing programs and data structures and are a major component of a “broad-based coverage of the discipline of computing.”
2. General Educational Objective 2: programming languages are so pervasive that one cannot be called a computer science without the ability to understand current and future programming languages and hence a fundamental aspect of “producing graduates who are prepared to work in computing.”
3. BS in CS Specific 1: this course provides practice in project management and the formalisms of programming that are needed to support “emerging and future information infrastructure.”
4. BS in CIS Specific 1: this course provides practice in project management and the formalisms of programming that are needed to support “emerging and future information infrastructure.”

Estimate CSAB Category Content

Category	Core	Advanced	Category	Core	Advanced
Data Structures	0	3/5	Computer Organization and Architecture	3/5	0
Algorithms	0	3/5	Concept of Programming Languages	0	3/5
Software Design	3/5	0			

Theoretical Foundations

CPSC 350 is a putative prerequisite. Students are expected to understand finite state automata, finite state machines, context free languages, and pushdown automata. These are fundamental theoretical foundations and are an integral part of the course.

Since denotational semantics is a central focus, students must be able to follow a discussion of recursive descent, recursive functions, define recursive functions, and test recursive functions.

Axiomatic semantics are emphasized as the design maxim for testing the compiler.

Communication (Oral & Written) Skills

Several students are called upon every day, and often are required to go to the board. The size of the class virtually guarantees that a student will only be called upon once per semester. However, there are many “read-out” opportunities from the group work. These are graded as participation grades.

Every day, students must turn in a written homework assignment, which is also part of their participation grade.

All milestones have a written report that is worth 25% of the milestone grade.

Social and Ethical Implications of Computing

The discussion comes up, often in connection with work ethics or marketing ploys, and almost always is due to student disaffection with something. Discussions are not purposefully scheduled unless the topic comes up and can be fit into the class. The students are placed in four-person groups; they are graded by their peers at the end of the semester.

Problem Analysis and Design

The students must complete eight milestones to receive an A in the course. The general structure of the milestones is developed for them; they must solve certain milestone-specific problems. Class time is for discussion of design issues.

Students are expected to be on time and to complete the milestone specification. Students are expected to design and demonstrate that their code passes their own tests. The faculty/graders have their own test.

Students are expected to show, if not develop, a personal software process.

Program Implementation

See above. Students are expected to implement, in the language of their choice, the project outlined in the eight milestones. Each program must be submitted with user test data and with the full project report.

Coordinator/Prepared by

Prof. Steve Stevenson, 07/27/04