Course Information

SER 221 Programming Languages and Their Execution Environment

Catalog Description

Introduces the fundamental programming language concepts of data, type, control, abstraction, and structure; software development and execution environments; programming language paradigms.
Credit Hours: 3

Prerequisites by Topic, Course

- CST200

Textbooks and Other Resources

- Problem Solving with C++, by Walter Savitch
- Concepts of Programming Languages by Roberta W. Sebesta
- Elements of ML Programming, by Jeffrey D. Ullman
- SWI Prolog (www.swi-prolog.org) for Prolog programming and the Standard ML of New Jersey (www.smlnj.org) for ML programming. You can download personal versions of SWI Prolog and Standard ML of NJ for free from the respective websites.

Course Description and Goals

Description: This course acquaints the student with different classes of programming languages, criteria for language selection in various application areas, and the fundamentals of language design. Formal notations for language definition will be introduced. Run time support for each class of language will be examined. Students will gain exposure to the unique features of C++, object-oriented design and general features of object-oriented languages. In addition to traditional imperative/object-oriented languages, aspects of functional programming, including ML, and of logic programming, including Prolog, will be presented.

Major Topics Covered

- Overview, C++ classes (1 week).
- Object-Oriented Programming: Operator overloading, inheritance, polymorphism, pointers (2 weeks)
- Templates, exception handling (1 week)
• Language design issues (1 week)
• Impact of machine architecture (1 week)
• Syntax, types, objects, composite types, files, structured data types, subprograms, and parameter passing (4 weeks)
• Logic programming (2 weeks)
• Functional programming (2 weeks)

Course Coordinator
Timothy Lindquist

Sample Course Activities

1. Programming projects
   Description
   Three programming projects require that students construct programs in three different languages. One (using gcc) is a C-based language that is compiled to object code (C, C++, or Objective-C). Another is a functional language that is interpreted (Scheme). The third is a declarative language (XSLT, SQL, or Prolog).
   Activity Type
   Individual programming projects.
   Grading Information
   The three projects make up 25% of the course grade. Programming projects are graded based on correct solution of the problem, and proper programming language usage.

2. Program Analysis, Implementation and Execution.
   Description
   Each team will be given a different program, in possibly different programming languages. Teams are to analyze programs and elaborate the implementation of the program. The team constructs a document gives an oral presentation that elaborates intermediate data and flow of recognition, translation, and execution (run-time structure) of the program as it is executed with sample input data. For a compiled language, this includes the output of the scanner, parser, intermediate code generator, and snap-shot of the run-time environment at a significant point in execution.
   Activity Type
   Team-based activity with written document and oral presentation.
   Grading Information
   The team project counts 10% of the course grade based on: Team effectiveness, individual contribution to team, team presentation format and organization, team program analysis document format and organization, and accuracy of technical solution.

3. Language Characterization
   Description
   Students are provided a programming language for which they develop a 15 minute tutorial/wiki style presentation. Presentations include language history, language
characteristics and code examples, examples of usage (by projects and domain), implementation characterization, and references. Each student will be responsible for a different language. The languages assigned include special purpose and historically significant languages, such as snobol, jovial, ada, algol, ML, PL-1, lucid, bcpl.

**Activity Type**
Research and oral presentation

**Grading Information**
The presentation accounts for 5% of the course grade and is based on: format, organization, and delivery of the presentation, accuracy and completeness of the language characterization. All students will rate and critique each presentation.

**Course Outcomes**

1. **CO1** Explain design and implementation considerations for data types, expressions, control structures, and subprograms in modern programming languages

   **Student Outcomes Supported**
   Technical Competence

2. **CO2** Compare top-down and bottom-up methodologies for problem solving using features of imperative and object-oriented languages.

   **Student Outcomes Supported**
   Problem Solving

3. **CO3** Explain design and implementation considerations for scope, binding, abstract data types, objects, and exception handling in modern programming languages.

   **Student Outcomes Supported**
   Technical Competence

4. **CO4** Design and implement solutions to problems using imperative features of C++.

   **Student Outcomes Supported**
   Design, Technical Competence

5. **CO5** Design and implement solutions to problems using object-oriented features of C++.

   **Student Outcomes Supported**
   Design, Technical Competence
6. CO6 Design and implement solutions to problems using declarative features of ML/Prolog.

Student Outcomes Supported
Design, Technical Competence