

CSE 240 Introduction to Programming Languages

Syllabus and Course Information

Course web page in: www.asu.edu/myasu/

Catalog Description

Introduces the procedural (C/C++), applicative (Scheme/LISP), and declarative (Prolog) languages. Lecture, lab. Prerequisite: CSE 205.

Text

Y. Chen, W.T. Tsai, Introduction to Programming Languages: Programming in C, C++, Scheme, Prolog, C#, and SOA, second edition, Kendall/Hunt Publishing Company, 2006, ISBN 0-7575-2974-7.

Reference Books and On-Line Materials

There are many books and on-line materials that are related to the course. Followings are a few of them that might be useful. Other similar books could be used as reference books too. The first two books were used as the textbooks for this course in the past a few years.

- *Programming Languages Essentials*, H. Bal and D. Grune, Addison-Wesley, 1994.
This books focuses on the language principles. Little programming is taught.
- *MiniManuals in PC SCHEME* and *PROLOG*, McGraw-Hill, 1991.
The prolog part of this book is imported into the current text.
- *The C Programming Language*, B. Kernighan and D. Ritchie, Prentice-Hall, 1978.
- *C++ Programming with Design Patterns Revealed*, T. Mueldner, Addison Wesley, 2002.
- *C++: How to Program*, Third Edition, Harvey and Paul Deitel, Prentice-Hall, 2001.
- *The C++ Programming Language*, B. Stroustrup, Addison-Wesley, 1997.
- *The Schematics of Computation*, V. Manis and J. Little, Prentice Hall, 1995.
- *Programming in Prolog*, W.F. Clocksin, and C.S. Mellish, Third, revised and extended edition, Springer-Verlag
- *On-line Materials* are listed under the link "Course Documents" in the course web page.

Objectives and Outcomes

Prior programming experience is required. The objectives and outcomes of the course are

1. To provide computer science students with an exposure to different programming paradigms
 - 1) Students will understand strong vs. weak typing in computer programming languages
 - 2) Students will understand the control structures of functional, logic, and imperative programming languages.
 - 3) Students will understand the execution of functional, logic, and imperative programming languages.
 - 4) Students will understand the recursion mechanism of functional, logic, and imperative programming languages.
2. To develop an introductory understanding of an applicative programming language (Scheme)

- 1) Students will work with the Scheme interpreter to evaluate simple functions.
- 2) Students will write and execute simple Scheme functions.
- 3) Students will write and execute Scheme programs requiring multiple functions.
3. To develop an introductory understanding of a declarative programming language (Prolog)
 - 1) Students will create a simple Prolog factbase and provide queries to obtain information from the factbase.
 - 2) Students will create Prolog programs that use recursive rules to provide a problem solution.
 - 3) Students will create Prolog programs that use multiple rules to solve a problem.
4. To develop an introductory understanding of a procedural programming language (C/C++)
 - 1) Students will write C/C++ programs using pointers.
 - 2) Students will write C/C++ programs using multiple functions/procedures.
 - 3) Students will write C/C++ programs using dynamic memory allocation.
 - 4) Students will write C/C++ programs that allocate and de-allocate static, stack and heap memory.
 - 5) Students will design C/C++ programs applying object-oriented features such as inheritance, polymorphism and class hierarchy.

Requirement

The official prerequisite is CSE205 (Concepts of Computer Science and Data Structures). By topics I expect you to have

- understood basic concepts of computer organization, including registers, memory, arithmetic and logic units (ALU), processor, input and output.
- been familiar with object-oriented design, static and dynamic data structures like Integer, Floating-point numbers, Arrays, Strings, Stacks, and data abstraction techniques.
- understood programming techniques and control structures like branching, iteration and recursion.
- commanded a high level programming language like Java or C++ and the environment in which a program is developed, e.g., editor, compiler/interpreter, linker, source code, executable code, debugging tool, etc.

If you don't meet the official prerequisites but are admitted into the course because you did courses "equivalent" to the official prerequisite courses, it is your responsibility to make sure you do understand the necessary background material.

Major Topics Covered in the Course (Tentative)

1. Aspects of programming languages (2 weeks)

- 1) Different paradigms of programming languages
- 2) Introduction to the structures of programming languages
- 3) Program processing: interpretation, compilation and macro processing
- 4) Strong versus weak typing
- 5) Orthogonality

2. Introduction to procedural programming languages (C and C++) (3 weeks)

- 1) Basic data types and data declarations, scope rule and forward declaration
- 2) Complex data types: array, pointer, string, constants, enumeration, struct types

- 3) Data structures: stack, linked list, and tree
- 4) Recursion: concept and programming
- 5) Functions and parameter passing

3. Introduction to object-oriented programming languages (C++) (2 weeks)

- 1) Concept of object-oriented programming
- 2) Class definition and members of class, constructor and destructors
- 3) Memory management: static, stack and heap memory, and garbage collection
- 4) Inheritance and class hierarchies, polymorphism, virtual functions and dynamic binding
- 5) containment versus inheritance

4. Introduction to applicative programming language (Scheme) (3.5 weeks)

- 1) Arithmetic expression and prefix notation
- 2) Basic Scheme procedures, defining your own procedures
- 3) Scheme environment, global and local variables
- 4) Recursive procedures
- 5) Programming with data structures, number, character, strings, symbol, pairs and lists

5. Introduction to declarative programming language Prolog (3.5 weeks)

- 1) Facts, rules, and goals
- 2) Structured facts and rules
- 3) Scope of variables
- 4) Arithmetic operations
- 5) Recursion and recursive rules
- 6) Lists and list manipulation

Assessment and Grading

Your performance will be assessed by assignments, programming projects, quizzes, a Mid-Term Exam and a Final Exam. Their weights are:

Assignments and Projects	40%
Quizzes	20%
Mid-Term	18%
Final Exam	22%
Total	100%

The final letter grade is decided according to the percentage points obtained as follows:

A-, A, A+	90-92, 93-95, 96-100%
B-, B, B+	80-82, 83-86, 86-89%
C, C+	70-75, 76-79%
D	60-69%
E	less than 60%

The grade of “I” (incomplete) can be given ONLY when a student, who is doing otherwise acceptable work (passing grade), is unable to complete a part of work (e.g., the final exam) because of documented illness or other conditions beyond the student’s control. In the latter case, the student must discuss with the instructor and complete an application form from the department before the part of work is due or as soon as the circumstances are known. Please see ASU grading policies at: <http://students.asu.edu/grades-grading-policies>

Extra Credit and Alternative Activity

Missing a graded activity will be given zero credit. In-class exercises and quizzes may not be made up. One additional quiz will be arranged to override one missing or poor quiz score.

No extra credit-activities will be given to any individual. Extra credit-activities may be given to the entire class. An alternative to the assignment and exam may be arranged if a student misses the activity and the absence is caused by documented illness or personal emergency that made the completion/attending impossible. A written explanation (including supporting documentation) must be submitted to the instructor before the part of work is due or as soon as the circumstances are known.

Grading Appeals

Any inquires or appeals on grades of homework, projects, or tests must be done in writing by completing the "Grade Inquiry Form" within a week from the day the assignment was returned or comments were published on-line. State the problem and the rationale for any change in grade in your appeal.

Cooperation

You are encouraged to cooperate in study group on preparing assignments, projects, tests and exams where permitted. However, anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem.

The instructor and the TA will **CAREFULLY** check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity) to check any assignment that you submitted for grading. The Ira A. Fulton School of Engineering and the Department of Computer Science and Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:

http://www.asu.edu/studentaffairs/studentlife/judicial/academic_integrity.htm

ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.