

# CS3360: Design and Implementation of Programming Languages

Spring 2015

**Instructor:** Nigel Ward, nigel@utep.edu  
Computer Science 3.0408, 747-6827  
Office Hours: Wednesdays 2:30-3:30, or by appointment,  
and generally whenever the door is open

**TA:** Gabriel Arellano, grarellano@utep.edu  
Office Hours: Fridays 2:00-3:00 in CCSB 1.0706, and by appointment

**Class Time:** TTh 12:00-1:20 in CCSB 1.0202

**Textbook:** *Concepts of Programming Languages*, 10<sup>th</sup> edition, Robert W. Sebesta, Addison Wesley, 2012. (required; bring to class every day)

**Recommended:** *The C Programming Language*, 2<sup>nd</sup> edition, Brian W. Kernighan and Dennis Ritchie, Prentice Hall 1988.

**Course Description:** Design features of modern programming languages, including flow control mechanisms and data structures; techniques for implementation of these features.

**Goals:** Acquire the knowledge and skills needed to rapidly learn and program effectively in new programming languages.

**Major Topics:**

Principles of programming languages, programming paradigms, and language trade-offs. Scope and bindings, data types, subprograms, semantics, syntax and its specification. Programming in representative languages.

**Course Policies**

The prerequisite for this class is CS2302 with a C or better.

Assigned readings are to be done before class.

Assignments are to be submitted in hardcopy in class, unless otherwise specified. After a 1 minute grace period at the start of class, assignments will be considered late. Late assignments will be accepted at the end of class or before or after any subsequent class session, and will be penalized at least 10% per day or partial day of lateness, for up to five days. Depending on the circumstances the penalty may be higher, for example, if an assignment is received after the solution has been discussed in class. For some assignments the code will also need to be submitted separately. Email submissions of assignments are not accepted unless otherwise specified.

Assignments are to be done individually unless specifically designated as group assignments. While you may discuss assignments with others, your solutions should be designed, written, and tested by you alone. If you need help, consult the TA or the instructor.

All code should be your own, unless the assignment explicitly permits the use of borrowed code or found code; in such cases you must acknowledge your sources and state specifically what you used.

Programming assignments will be graded primarily on functionality, design quality, thoroughness of testing, and readability. In addition style and other factors will be considered when appropriate. Some of these factors inevitably involve subjective judgments; if you have questions about these or any other aspect of the grading, please see the TA or the instructor.

Tests will be closed-book, except that one single-sided page of hand-written notes may be brought in for the first test, two for the second test, and three for the final. If you leave the classroom for any reason, your test will be graded on only what you did up until that time.

Grades will be based on four components, weighted approximately as follows: assignments (50%), the final examination (20%), tests (25%), and other factors (5%), including quizzes, in-class exercises, and general participation. The instructor reserves the right to adjust the final grades upwards in cases where the performance on both the assignments and the tests is solid.

Assignments and tests will be challenging. Grading will be on a points-earned basis (points above zero), rather than a points-off basis (points below expectation). Letter grades will be assigned accordingly; the A/B break will probably be around 75% and the B/C break around 65%.

### **General Policies**

Students are expected to be punctual, and, as always, to conduct themselves professionally and courteously.

If you have or suspect a disability and need accommodation you should contact the Center for Accommodations and Support Services at 747-5148 or at [cass@utep.edu](mailto:cass@utep.edu) or visit Room 106 Union East.

No make-up exams or assignments will be given except under the conditions set forth in the Catalog.

Students are free to attend class or not, bearing in mind that absence may annoy other students, interfere with learning, and result in a lower grade.

### **Likely Topics, Readings and Major Assignments**

|                                    |                    |          |                        |
|------------------------------------|--------------------|----------|------------------------|
| Introduction                       | Chapter 1          | (2 days) |                        |
| Syntax                             | Chapter 3          | (2 days) |                        |
| Scripting, Web languages           | Section 2.18       | (4 days) | PHP Assignment         |
| Semantics                          | Section 3.5        | (1 day)  |                        |
| Scope and Binding                  | Chapter 5          | (2 days) |                        |
|                                    |                    |          | Bash Assignment        |
| Data Types                         | Chapter 6          | (2 days) |                        |
| Logic Programming                  | Chapter 16         | (2 days) | Prolog Assignment      |
| Subprograms                        | Chapters 9 and 10  | (2 days) |                        |
| Expressions and Control Structures | Chapters 7 and 8   | (1 day)  |                        |
| Object-Oriented Languages          | Chapters 11 and 12 | (3 days) | Objective-C Assignment |
| Functional Programming             | Chapter 15         | (3 days) | Haskell Assignment     |
| Review, Tests, etc.                | Chapter 2          | (3 days) |                        |

**Course Website:** <http://www.cs.utep.edu/nigel/pl/>

### **Important Dates**

January 29: Class begins  
February 17: Test 1  
March 9-13: Spring Break  
March 26: Test 2  
May 12: Final Exam, 1:00-3:45

### **Target Learning Outcomes**

#### **Level 1: Knowledge and Comprehension**

Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. The material has been presented only at a superficial level.

Upon successful completion of this course, students will be able to:

- 1.a. Explain the broad outlines in the history of development of programming languages.
- 1.b. Explain the stages of programming language interpretation and compilation.
- 1.c. Understand data and control abstractions of programming languages.

#### **Level 2: Application and Analysis**

Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.

Upon successful completion of this course, students will be able to:

- 2.a. Define syntax of a small context-free grammar in BNF and EBNF
- 2.b. Show the syntax of a small subset of a programming language in BNF and EBNF.
- 2.c. Use attribute grammars to describe the static semantics of small programming languages.
- 2.d. Describe dynamic semantics of small subsets of programming languages, e.g., expressions and control structures.
- 2.e. Compare different approaches to naming, storage bindings, typing, scope, and data types.
- 2.f. Analyze design dimensions of subprograms, including parameter passing methods, subprograms as parameters, and overload subprograms.

#### **Level 3: Synthesis and Evaluation**

Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery.

Upon successful completion of this course, students will be able to:

- 3.a. Evaluate modern, representative programming languages critically w.r.t. design concepts, design alternatives and trade-offs, and implementation considerations for scope, binding, data types, expressions, control structures, subprograms, abstract data types, objects, concurrency structures, and exception handling.
- 3.b. Choose a suitable programming paradigm and language for a given problem or domain.