CS3360: Design and Implementation of Programming Languages

Spring 2024

Instructor:  Nigel Ward, nigel@utep.edu
Computer Science 3.0408, 747-6827
Office Hours: Tue 1-2, Fri 3-4, and by appointment,
and generally whenever the door is open

TA:  Tosin Ige, toige@miners.utep.edu
Office Hours: TBD

Class Time:  Mon Wed 10:30 – 11:50 in CCSB 1.0202

Textbook:  Concepts of Programming Languages, 11th edition, Robert W. Sebesta, Addison Wesley, 2016. (The 10th and 12th editions are also acceptable if you can map sections and assignments across editions.) Required: bring to class, in hardcopy or pdf, every day.


Course Description

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


Course Policies

The prerequisite for this class is CS 2302 with a C or better. CS 3432 (Architecture) is recommended. Assigned readings are to be done before class.

Assignment submission is in hardcopy at the start of class, by default. Programming assignments will, when specified, also be submittable in Blackboard. Email submissions of assignments are not accepted unless otherwise specified.

Late assignments are those received, for hardcopy submission, after a 1-minute grace period at the start of class. Such late assignments will be accepted at the end of class or before or after any subsequent class session. Blackboard assignments will be late if received after the specified deadline. Late assignments 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.

Discussion of assignments with peers is encouraged, but your solutions should be designed, written, and tested by you alone. All assignments are to be submitted individually unless designated as group assignments.

Use of code fragments from others will be allowed for a few of the larger assignments, when so specified. In such cases you must acknowledge your sources and state specifically what you used. If you need help, consult the TA or the instructor.
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 final grades upwards in cases where the performance on both the assignments and the tests is solid.

Grading standards are high: assignments and tests will be challenging, and 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 80% and the B/C break around 70%.

Grading criteria for programming assignments will be primarily functionality, design quality, thoroughness of testing, and readability. Style and other factors will also have some weight. 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.

Quizzes will be frequent. The lowest quiz grade will be dropped.

General Policies

The class format will be lectures, in-class design exercises, discussions, student presentations, project activities, and possibly guest speakers. There is no remote option.

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

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 or visit Room 106 Union East.

Students are expected to be punctual, to conduct themselves professionally and courteously, and to follow the spirit and letter of the UTEP Standards of Student Conduct and Academic Integrity policy https://www.utep.edu/student-affairs/osccr/student-conduct/academic-integrity.html. Suspected violations will be referred to the Office of Student Conduct.

Topics, Readings and Major Assignments, tentative

<table>
<thead>
<tr>
<th>Component</th>
<th>Chapter(s)</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Chapter 1</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Syntax</td>
<td>Chapter 3</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Scripting, Web languages</td>
<td>Section 2.18</td>
<td>(4 days)</td>
</tr>
<tr>
<td>Semantics</td>
<td>Section 3.5</td>
<td>(1 day)</td>
</tr>
<tr>
<td>Scope and Binding</td>
<td>Chapter 5</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Data Types</td>
<td>Chapter 6</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Logic Programming</td>
<td>Chapter 16</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Subprograms</td>
<td>Chapters 9 and 10</td>
<td>(2 days)</td>
</tr>
<tr>
<td>Functional Programming</td>
<td>Chapter 15</td>
<td>(3 days)</td>
</tr>
<tr>
<td>Object-Oriented Languages</td>
<td>Chapters 11 and 12</td>
<td>(3 days)</td>
</tr>
<tr>
<td>Expressions, Control Structures</td>
<td>Chapters 7, 8, 14</td>
<td>(1 day)</td>
</tr>
<tr>
<td>Review, Tests, etc.</td>
<td>Chapter 2</td>
<td>(3 days)</td>
</tr>
</tbody>
</table>
Course Website: http://www.cs.utep.edu/nigel/pl/

Important Dates (tentative)
   January 17: Class begins
   February 14: Test 1
   March 11 – 15: Spring Break
   March 27: Test 2
   May 10, Friday: 10:00 - 12:45 : Final Exam

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. Describe broad trends 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.
   1.d. Understand how attribute grammars describe static semantics.
   1.e. Describe ways to formally specify the dynamic semantics of small subsets of programming languages, such as expressions and control structures.
   1.f. Understand code snippets written in a paradigm beyond imperative, object-oriented, and functional, e.g., algebraic, aspect-oriented, logic, or probabilistic 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
   2.b. Define the syntax of a small subset of a programming language using BNF
   2.c. Compare different approaches to naming, storage bindings, typing, scope, and data types.
   2.d. Analyze design dimensions of subprograms, including parameter passing methods, subprograms as parameters, and overloaded subprograms.
   2.e. Be able to write programs to solve simple problems in a purely functional language.
   2.f. Be able to write programs to solve simple problems in a scripting language.

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 with respect to design concepts, design alternatives, and implementation considerations for variables, types, expressions, control structures, and program modules.
   3.b. Choose a suitable programming paradigm and language for a given problem or domain.