CS3360: Design and Implementation of Programming Languages

Fall 2022

Instructor: Nigel Ward, nigel@utep.edu
Computer Science 3.0408, 747-6827
Office Hours: Wed 2:30 – 3:30, Fri 1:30 – 2:30, or by appointment,
and generally whenever the door is open

IA: TBD
Office Hours: TBD

Class Time: TuTh 1:30 – 2:50 in Liberal Arts 222

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

Recommended: *The C Programming Language,*2nd 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 CS 2302 with a C or better. CS 3432 (Architecture) is also recommended.

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. 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 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 functionally, design quality, thoroughness of testing, and readability. Style and other factors will also 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.

There will be frequent quizzes. The lowest quiz grade will be dropped.

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.

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 80% and the B/C break around 70%.

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.

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 or visit Room 106 Union East.

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

Topics, Readings and Major Assignments, tentative

Introduction	Chapter 1	(2 days)	
Syntax	Chapter 3	(2 days)	
Scripting, Web languages	Section 2.18	(4 days)	Assignment 1
Semantics	Section 3.5	(1 day)	
Scope and Binding	Chapter 5	(2 days)	Assignment 2
Data Types	Chapter 6	(2 days)	
Logic Programming	Chapter 16	(2 days)	Assignment 3
Subprograms	Chapters 9 and 10	(2 days)	
Functional Programming	Chapter 15	(3 days)	Assignment 4
Object-Oriented Languages	Chapters 11 and 12	(3 days)	Assignment 5
Expressions, Control Structures	Chapters 7, 8, 14	(1 day)	
Review, Tests, etc.	Chapter 2	(3 days)	

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

Important Dates (tentative)

August 23: Class begins
September 22: Test 1
October 27: Test 2
November 24: Thanksgiving
??? : 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, sub-programs 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.