

CSE 3341 (Approved): Principles of Programming Languages

Course Description

Formal languages and grammars; recursive descent parsing; data types, expressions, control structures, parameter passing; compilers and interpreters; memory management; functional programming principles.

Prior Course Number: CSE 655

Transcript Abbreviation: Prin Prgrmnng Langs

Grading Plan: Letter Grade

Course Deliveries: Classroom

Course Levels: Undergrad

Student Ranks: Junior

Course Offerings: Autumn, Spring

Flex Scheduled Course: Never

Course Frequency: Every Year

Course Length: 14 Week

Credits: 3.0

Repeatable: No

Time Distribution: 3.0 hr Lec

Expected out-of-class hours per week: 6.0

Graded Component: Lecture

Credit by Examination: No

Admission Condition: No

Off Campus: Never

Campus Locations: Columbus

Prerequisites and Co-requisites: CSE 2231 and (CSE 2331 or CSE 680) and CSE 2421 and (CSE 3901 or CSE 3902 or CSE 560)

Exclusions: Not open to students with credit for CSE 5341 or CSE 655

Cross-Listings:

The course is required for this unit's degrees, majors, and/or minors: No

The course is a GEC: No

The course is an elective (for this or other units) or is a service course for other units: Yes

Subject/CIP Code: 14.0901

Subsidy Level: Baccalaureate Course

Programs

Abbreviation	Description
BS CSE	BS Computer Science and Engineering

Course Goals

Master using syntax-directed parsing, printing, execution, and compilation for simple imperative language constructs.
Master distinguishing between compile-time vs. run-time activities.
Be competent with using syntax-related concepts including regular expressions and context-free grammars to describe the structure of languages.
Be competent with analyzing programming language design issues related to data types, expressions and control structures, parameter passing.
Be competent with principles of object-oriented languages.

Be competent with implementing object-oriented languages.
Be familiar with memory management techniques for imperative languages, including object-oriented languages.
Be familiar with using functional programming languages.
Be exposed to analyzing variable bindings and scope rules.

Course Topics

Topic	Lec	Rec	Lab	Cli	IS	Sem	FE	Wor
Overview of types of languages; language design and evaluation criteria; implementation methods (compilers and interpreters)	3.0							
Grammars (regular expressions, CFGs), abstract & concrete parse trees; recursive descent parsing;	6.0							
Recursive descent interpretation, compilation;	3.0							
Principles of OO languages (data abstraction, encapsulation, single and multiple inheritance, polymorphism (single and multiple dispatch), etc.)	7.0							
Implementation of imperative languages including OO languages (stack, heap management; activation records etc; runtime dispatch in OO languages)	10.0							
Functional programming principles (Scheme), including some discussion of implementation techniques	7.0							
More advanced ideas (ML types/inferencing, dynamically-typed languages, ...?)	7.0							

Representative Assignments

Implement a complete recursive descent interpreter for a simple imperative language.
Implement an application that involves use of one or two important patterns such as Visitor and Observer.
Implement a set of functions in Scheme to manipulate lists in a variety of ways (or, possibly: implement an application that does some symbolic computation such as a chess playing program).
Other homework assignments focusing on such things as CFGs.

Grades

Aspect	Percent
Homeworks	15%
Programming projects	35%
Midterms, Finals	50%

Representative Textbooks and Other Course Materials

Title	Author
<i>Programming language pragmatics</i>	M. Scott

ABET-EAC Criterion 3 Outcomes

Course Contribution	College Outcome
***	a An ability to apply knowledge of mathematics, science, and engineering.

Course Contribution		College Outcome
**	b	An ability to design and conduct experiments, as well as to analyze and interpret data.
***	c	An ability to design a system, component, or process to meet desired needs.
*	d	An ability to function on multi-disciplinary teams.
***	e	An ability to identify, formulate, and solve engineering problems.
*	f	An understanding of professional and ethical responsibility.
*	g	An ability to communicate effectively.
*	h	The broad education necessary to understand the impact of engineering solutions in a global and societal context.
**	i	A recognition of the need for, and an ability to engage in life-long learning.
*	j	A knowledge of contemporary issues.
***	k	An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

BS CSE Program Outcomes

Course Contribution		Program Outcome
***	a	an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering;
**	b	an ability to design and conduct experiments, as well as to analyze and interpret data;
***	c	an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints such as memory, runtime efficiency, as well as appropriate constraints related to economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability considerations;
*	d	an ability to function on multi-disciplinary teams;
***	e	an ability to identify, formulate, and solve engineering problems;
*	f	an understanding of professional, ethical, legal, security and social issues and responsibilities;
*	g	an ability to communicate effectively with a range of audiences;
*	h	an ability to analyze the local and global impact of computing on individuals, organizations, and society;
**	i	a recognition of the need for, and an ability to engage in life-long learning and continuing professional development;
*	j	a knowledge of contemporary issues;
***	k	an ability to use the techniques, skills, and modern engineering tools necessary for practice as a CSE professional;
***	l	an ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
***	m	an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;
***	n	an ability to apply design and development principles in the construction of software systems of varying complexity.

Prepared by: Neelam Soundarajan