The University of Georgia

Department of Computer Science

Course Information Sheet
CSCI 1301
Introduction to Computing and Programming

Brief Course Description
(50-words or less)

Algorithms, programs, and computing systems. Fundamental techniques of program development and supportive software tools. Programming projects and applications in a structured computer language.

Extended Course Description / Comments

This course is a rigorous introduction to problem solving using fundamental programming techniques: variables, operators, expressions, decision statements, loops, nested statements, arrays, methods, objects, classes, inputs, and outputs. This course includes programming projects incorporating algorithm design and implementation with a structured computer language and hands-on experience creating, testing, and debugging software. This course is typically the first major-related course taken by computer science majors or anyone interested in learning how to program.

Pre-Requisites and/or Co-Requisites

MATH 1113
PreCalculus

Approved Textbooks
(If more than one is listed, the textbook used is up to the instructor's discretion)

Author(s): Walter Savitch and Frank M. Carrano
Title: *Java: An Introduction to Problem Solving and Programming*
Edition: 5th Edition
ISBN-13: 978-0136072256

Author(s): Walter Savitch
Title: *Java: An Introduction to Problem Solving and Programming*
Edition: 6th Edition
ISBN-13: 978-0132162708

Specific Learning Outcomes (Performance Indicators)

This course presents fundamental programming topics in a structured programming language. At the end of the semester, all students will be able to do the following:

1. Apply knowledge of compiling, running, testing, and debugging programs.
2. Design and implement algorithms to solve problems.
3. Write programs with a structured programming language that utilize variables, operators, expressions, decision statements, loops, nested statements, arrays, methods, objects, classes, inputs, and outputs to solve problems.
4. Write the output of a program by tracing through its source code.
5. Use an integrated development environment for programming.
6. Generate program documentation.

Relationship Between Student Outcomes and Learning Outcomes

		Student Outcomes										
		a	b	c	d	e	f	g	h	i	j	k
Learning Outcomes	1	•	•	•						•		
	2	•	•	•						•		•
	3	•	•	•						•		•
	4	•										
	5	•		•						•		
	6	•					•					

Student Outcomes

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
d. An ability to function effectively on teams to accomplish a common goal.
e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
f. An ability to communicate effectively with a range of audiences.
g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
h. Recognition of the need for and an ability to engage in continuing professional development.
i. An ability to use current techniques, skills, and tools necessary for computing practice.
j. 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.
k. An ability to apply design and development principles in the construction of software systems of varying complexity.

Major Topics Covered (Approximate Course Hours)

Hardware/Software Basics (1-hour)
Algorithms (0.5-hours)
Compiling and Running Programs (0.5-hours)
Basic Input and Output (0.5-hours)
Integrated Development Environment (0.25-hours)
Variables and Constants (0.5-hours)
Data types (0.5-hours)
Assignment Statements (1.5-hours)
Operators (1.5-hours)
Expressions (2-hours)
Modular Arithmetic (0.5-hours)
Math functions (0.5-hours)
Strings (1-hour)
If Statements (3-hours)
Switch Statements (0.5-hours)
Enumerations (0.5-hours)
Programming Style and Documentation (1-hour)
Loops (3.5-hours)

Scope and Block Statements (0.5-hours)
Assertion Checks (0.25-hours)
Program Tracing, Testing, and Debugging (2-hours)
Methods (2-hours)
Classes (2-hours)
Objects (2-hours)
Constructors (0.5-hours)
Static Variables and Methods (0.5-hours)
Overloading (0.25-hours)
Packages (0.25-hours)
Information Hiding and Encapsulation (1-hour)
Public and Private Modifiers (0.5-hours)
Arrays (3-hours)
Sequential Search of Arrays (0.25-hours)
Selection and Bubble Sort of Arrays (0.75-hours)

Assessment Plan for this Course

The course instructor takes the results of exam questions and a sampling of programming assignments corresponding to course outcomes, and reports these results to the ABET committee. If necessary, the instructor also writes a recommendation to the ABET committee for better achieving the course outcomes the next time the course is offered.

How Data is Used to Assess Program Outcomes

Each course Learning Outcome, listed above, directly supports one or more of the Program Outcomes, as is listed in "Relationships between Program Outcomes and Learning Outcomes". For CSCI 1301, Program Outcomes (a), (b), (c), (f), (i), and (k) are supported.

Course Master

Course History