COURSE DESCRIPTION

Dept., Number  CSC 10  Course Title  Introduction to Programming Logic
Semester hours  3  Course Coordinator  Meiliu Lu
URL (if any):  http://gaia.ecs.csus.edu/~mei/

Catalog Description

An introduction to computer science with an emphasis on programming concepts and methodology. Intended to assist students with little or no programming experience to understand the basic principles of programming logic. Topics include: computer hardware and software, programming concepts and methodology including problem solving and algorithm development, flow of control, modular design using techniques that can be applied to common programming languages. Lecture two hours, technical activity and laboratory, two hours. Prerequisite: At least a C- grade in Intermediate Algebra (Math 11 or equivalent).

Textbook


Course Goals

1. Read and analyze problem descriptions.
2. Analyze the input and output needs for a specified problem.
3. Develop or choose appropriate algorithms for solving problems.
4. Design algorithms using the control structures of structured programming.
5. Represent algorithms using an approach acceptable in the modern software development industry.
6. Simulate the execution of algorithms using a systematic desk-checking approach.
7. Describe the relationship between algorithm design and computer programming in modern high-level programming languages.
8. Describe the relationship between simulated execution of algorithms (desk-checking) and the actual computer execution of programs that implement those algorithms.
9. Develop a modular design for a software implementation to solve a problem.
10. Describe several approaches for communication of data within a modular software design.
11. Use preconditions and postconditions to describe the behavior of a “black-box” module.

Prerequisites by Topic

Thorough understanding of:

- Intermediate Algebra.
Major Topics Covered in the Course

1. Introduction to programming, how computers work (2 hours).
2. Pseudocode (3 hours).
3. The Software Development Life Cycle (2 hours).
4. Structured programming (1 hour).
5. Desk checking (2 hours).
6. Developing algorithms using the Sequential Control Structure (1 hour).
7. Precedence of arithmetic operators (1 hour).
8. Developing algorithms using the Conditional Control Structure (4 hours).
9. Developing algorithms using the Loop Control Structure (4 hours).
10. Designing and using modules. Introduction to communication of data between modules, flow of control, rationale for using modules (subroutines and functions) (3 hours).
11. Communication between modules (2 hours).
12. Arrays (3 hours).
13. Exams (2 hours).

Laboratory Projects

1. Introduction to the environment which will be used for execution of algorithms (1 week).
2. Using variables and assignments statements and labeled output (1 week).
3. Using input and various data types (2 weeks).
4. Performing arithmetic operations (2 weeks).
5. Using conditional control structure (2 weeks).
6. Using loops (2 weeks).
7. Using single-dimensional arrays (2 weeks).
8. Using subroutines (2 weeks).

Note: In addition to these projects, guided design and other programming-related activities are conducted during most lab activity sessions.

Estimated Curriculum Category Content (Semester hours)

<table>
<thead>
<tr>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>0.3</td>
<td></td>
<td>Data Structures</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td>1.0</td>
<td></td>
<td>Prog. Languages</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Comp. Arch.</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Oral and Written Communications

The Lecture/Discussion assignments will require algorithms, represented in pseudocode and, later in the semester, structure charts, along with desk checks for the algorithms. These require statements as to the purpose of the algorithm as well as comments throughout the algorithm.
Social and Ethical Issues

No significant component.

Theoretical Content

No significant component.

Problem Analysis

Design is the primary focus of all lecture material. All but the most introductory laboratory programming assignments include requirement for design, represented by pseudocode and/or a graphical representation of the logic.

Solution Design

Design is the primary focus of all lecture material. All but the most introductory laboratory programming assignments include requirement for design, represented by pseudocode and/or a graphical representation of the logic.

/aa