CSC 202 - COMPUTER PROGRAMMING PRINCIPLES

CREDIT HOURS: 3
PREREQUISITES: CSC 102
GRADE REMINDER: Must have a C or better in each prerequisite course.

CATALOG DESCRIPTION

Problem solving and algorithm design, program structures, data types, software development methods, and programming style.

PURPOSE OF COURSE

To introduce a disciplined approach to problem solving methods and algorithm development; to introduce procedural and data abstraction; to teach program design, coding, debugging, testing, and documentation using good programming style; to teach a block-structured high-level programming language; and to provide a foundation for further studies in computer science.

EDUCATIONAL OBJECTIVES

Upon successful completion of the course, students should be able to:

1. Apply a disciplined approach to problem solving and algorithm design.

2. Use the following: strategies for problem solving, techniques for analyzing problems and defining requirements, tools for representing algorithms, and methods for verifying and validating algorithms and programs.

3. Write programs in a modern block-structured procedural programming language.

4. Design and, by means of the programming language being learned, implement imperative solutions to moderately complex problems.

5. Demonstrate through artifact creation and testing, a solid knowledge of and an ability to properly use these programming features and facilities: data types, fundamental data structures (arrays, records, and arrays of records) control structures, procedures, functions, parameters, text files, and binary files.

6. Demonstrate through artifact creation, familiarity with abstract data types, pointers, and recursion.

7. Use operating system tools (command system, editor, compiler, linker, and loader) in single and multi-user environments.

8. Write cooperatively on software development projects.

CONTENT

<table>
<thead>
<tr>
<th>Description</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Terminal or Microcomputer Skills Review</td>
<td>1</td>
</tr>
<tr>
<td>Use of operating system and editor command languages</td>
<td></td>
</tr>
<tr>
<td>Problem Solving and Algorithm Design</td>
<td>10</td>
</tr>
<tr>
<td>Strategies for problem solving--problem decomposition, solution by analogy</td>
<td></td>
</tr>
<tr>
<td>Problem analysis and requirements definition--understanding the problem, describing the output requirements, identifying the input data</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm representation--pseudocode and graphical techniques including structure charts and flowcharting
Algorithm verification--desk checking with and without test data

Program Structures
Control structures--sequential, iterative, selective
Subprograms--procedures and functions, parameters, scope of identifiers, subprogram nesting, and introduction to recursion

Data Types, Operations, and Storage
Standard scalar types--integer, real, boolean, character
Structured types--arrays, character strings, records, arrays of records
Standard user-defined types--subrange, enumerated
Introduction to abstract data types
Files--text files for data, source programs, and operating system commands; binary files for data, object programs, and load modules

Program Development--Methods and Style
Design--procedural abstraction, data abstraction, top-down design and stepwise refinement, modular design, block structure, information hiding
Coding--use of structured control statements and modern programming style including proper indentation and choice of appropriate descriptive identifiers
Program debugging and verification--generation of test data, debugging techniques including manual and built-in tracing as well as use of stubs and drivers, top-down versus bottom-up testing
External and internal program documentation techniques

Exams (Plus Final)

REFERENCES