COURSE DESCRIPTION

Dept., Number  CSC 25  Course Title  Introduction to C Programming
Semester Hours  3  Course Coordinator  Weide Chang
URL (if any):  http://gaia.csus.edu/~changw/

Catalog Description

Topics include: types, operators, control structures, input/output, arithmetic operations, the C library and preprocessor, functions and parameters, arrays, strings, pointers, and structures. Program design and style will be emphasized. Students will use a microcomputer C compiler. Note: Students with significant programming experience should take CSC 60 rather than CSC 25. Lecture two hours, technical activity and laboratory two hours.

Textbook


References


Course Goals

To have students learn the fundamentals of programming for beginners:
1. Basic program structure, syntax, and notations.
2. Decision-making: if-else, switch-case, and short-hands of these.
3. Loops and nested loops: while, do-while, and for.
4. Basic modularization: defining subroutines, calling subroutines, argument passing.
5. Basic I/O operations: printf, scanf, open, read, write, and close disk files.
6. Arrays: single and multiple dimensions, and association with loops.
7. Special C features: addresses, pointers, and association with arrays.

Prerequisites by Topic

Basic Understanding of:
- Use of a personal computer: navigating in Windows, typing and editing words, creation of files and folders.

Exposure to:
- A programming language or scripting tool and experience on making a computer carry out a series of specified command sequences.
Major Topics

1. First exposure to programming: compiling and running a program (1).
2. Getting started with C: functions, statements, expressions (1).
3. Variables, arithmetic operators, includes, defines (1).
4. Use of simple input/output functions (1).
5. Number bases: decimal, binary, and hexadecimal formats (1).
6. Relational and logical operators (1).
7. Decision-making (3).
8. Loops (6).
9. Arrays (3).
10. Functions and arguments (3).
11. Modular use of functions (3).
12. Sorting algorithms (3).
13. Searching algorithms (3).
14. Strings (1).
15. Pointers (6).
16. Pointers related to arrays and functions (3).
17. User-defined data structures (2).
18. Midterm exam and quizzes (3).

Outcomes

*Thorough understanding of:*
- ANSI C programming syntax.
- Syntax errors.
- Built-in data types.
- Use of basic I/O function.
- Assignments, use of arithmetic operators, precedence issues.
- Decision-making, use of logical and relational operators.
- Loops - single and double (nested).
- Single-dimensional arrays and the use of loops to access them.

*Basic Understanding of:*
- Compiler-linker-loader usage.
- Debugger usage.
- Semantic errors.
- Single pointers to addresses of built-in data types.
- User-defined data types.
- Break and continue in nested loops.
- Multi-dimensional arrays and the use of nested loops to access them.
- Number/counting systems.
Exposure to:

- Compound data structures.
- Arrays being pointers.
- Pointers to functions.
- Double pointers to addresses of built-in data types.
- Single pointers to addresses of user-defined data types.
- Sorting and searching algorithms.
- Multi-file source code.
- Run-time logical errors.
- Definition of a programming problem.
- Illustration of algorithms and data structures for solving programming problems.

Laboratory Projects

Altogether there are about 20 programming assignments conducted in the lab, once or twice a week, depending on the progress of learning the specific topics. The following is a list of the lab assignments (total 15 lab hrs).

1. Use of the compiler tools (0.5 lab hr).
2. Basic arithmetic calculations, including output functions (0.5 lab hr).
3. Use of special mathematic functions to solve an engineering problem, including input functions (0.5 lab hr).
4. Decision-making – basic relational operators, use of "if" and "if-else" (0.5 lab hr).
5. Formulation of programming statements for various logical and relational tests for decision-making (0.5 lab hr).
6. Use of simple while loops to count and sum a series of numbers (1 lab hr).
7. Use of simple for loops to count and sum a series of numbers (0.5 lab hr).
8. Giving a loop, tracing the execution sequence of statements (1 lab hr).
9. Use of simple and nested while loops to display different patterns of symbols (1 lab hr).
10. Use of nested for loops to display different patterns of symbols (1 lab hr).
11. Reading and writing files (1 lab hr)
12. Single-dimensional arrays, declaration, initialization, reading, and updating them with simple loops (0.5 lab hr).
13. Multi-dimensional arrays, use of nested loops to access them (1 lab hr).
14. Pointers to simple data types (1 lab hr).
15. Character arrays as pointers (0.5 lab hr).
16. Defining functions as subroutines (0.5 lab hr).
17. Passing arguments to functions (1 lab hr).
18. Use of struct and typedef to define new data types (0.5 lab hr).
19. Pointers to user-defined data types (1 lab hr).
20. Pointers to functions (1 lab hr).
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>0.5</td>
<td></td>
<td>Data Structures</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td></td>
<td></td>
<td>Prog. Languages</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Comp. Arch.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Oral and Written Communications
None.

Social and Ethical Issues
None.

Theoretical Content
None.

Problem Analysis
Students learn to understand and use basic specifications to define programming problems.

Solution Design
Students learn basic programming skills to implement simple solutions.