EME5 Introduction to Computer Programming for Engineering Applications

Harry H. Cheng, Professor

Course Description

Structured programming in C for solving problems in engineering. Introduction to MATLAB and comparison study of C/C++ with MATLAB.

Expanded Course Outline

The presentation is cumulative so that each topic builds on information presented in the earlier lectures. A single topic is presented first and then reinforced in the subsequent lectures.

1. (0.5 week) Information for the Course; Introduction to Computer Hardware and Software (Chapter 1)

2. (0.5 week) Getting Started in programming in C, Cross-platform commands (such as ls, pwd, cd, chmod) for handling text files and C compiler commands for C/C++ programs in Windows, Mac OS X, and Linux (Chapter 2)

3. (0.7 week) Number Systems; Internal representations of binary, octal, decimal, and hexadecimal numbers; binary two's complementary representation; Floating point numbers; Selection of Scalar Data Types (char, short, int, long, long long, signed, unsigned, float, double, float complex, complex, double complex) for applications, Limitations and numerical accuracy of different data types; The 32-bit and 64-bit programming models, precautions and issues in writing programs for 32-bit and 64-bit machines across different platforms of Windows, Mac, and Linux (such as the size of long and pointer types); Formatted Input/Output of integral types and floating point numbers (Chapter 3)

4. (0.4 week) Expressions; Operands; Unary, binary, and ternary operators; Operator Precedence; Order of Operations using Expression Evaluation Tree. (Chapter 4)

5. (1.1 weeks) Selection and Iterative Statements (if, else-if, switch, while-loop, for-loop, do-while loop, break, continue) for Structured Programming; Algorithm Development Using Flowchart, Pseudocode, and Procedures; Machine Epsilon; Random Number Generation and its Applications (Chapter 5)

6. (0.7 week) Functions for Modular Programming and Code Reuse, Function Prototypes; Develop a statistics analysis library. (Chapter 6)

7. (0.3 week) Preprocessing Directives, Macros, header files, cross-platform development (Chapter 7)

8. (0.4 week) Storage Classes and Program Structure; Communication between functions; Static and automatical duration of variables; Global variables, static variables in file scope and static variables in function scope (Chapter 8)

9. (1 week) Introduction of vector and matrix for linear algebra (teaching math); One- and Two-Dimensional Arrays and Variable Length Arrays for matrix computation, Linear Algebra, and Data Processing (Chapter 10)

10. (0.8 week) Pointers, Functions Pass by Value versus Functions Pass by Reference, Relation between Pointer and Array, Dynamical Memory Allocation and Deallocation, Memory leak and dangling memory, Pointer to Pointers, Pointer to functions, and Interfacing Hardware Using Pointers (Chapter 11)

11. (0.6 week) ASCII Code; Characters and Strings for processing characters and strings using arrays and pointers; Application of pointers in string processing (Chapter 12)

12. (1.2 week) Structures and Enumerations; Top-down and bottom-up design of large-scale software project; development of libraries and Application Programming Interface (API); Develop a GPA library with testing and GUI application programs for layered software development and deployment; On-line documentation and development of user's s guide (Chapter 13)

13. (0.4 week) File Processing; Writing into and Reading from Files. (Chapter 14)

14. (0.3 week) Computational Arrays for matrices and linear algebra for engineering applications (Chapter 21)

15. (0.1 week) Introduction to Object-Oriented Design in C++, Two- and Three-Dimensional Plotting Using a Plotting Class (Chapter 20)

16. (1 week) Introduction to MATLAB and Comparison Study with C (Chapter 23)

    - Comparison study of type-less versus typed languages. Style of MATLAB programming.
    - Variables and arrays in MATLAB
    - Matrix computation and linear algebra in MATLAB.
    - Operators in MATLAB.
    - Selection and iterative statements in MATLAB.
    - Formatted input and output in MATLAB.
    - File processing in MATLAB.
    - Two- and three-dimensional plotting in MATLAB.
    - Functions and function files (M-files) in MATLAB.
    - Functions passing arrays in the arguments and functions returning arrays in MATLAB.

17. (Self-study) Introduction to Fortran and Comparison Study with C (Chapter 24)

Algorithm Development

The course is focused on structured programming and software design in C. Numerous algorithms for computer-aided problem solving are developed throughout the course to solve practical problems in engineering and science. Students will learn how to write computer programs in C/C++ to solve the following carefully selected representative problems numerically with algorithm development in flowchart and pseudocode either in lectures or homework assignments.

1. Numerical solutions for quadratic equations.

2. Solving a mechanical system with the position, velocity, acceleration, and force. Present solutions graphically.

3. Solutions of the second order differential equation with applications in vibration, oscillation, and electric circuits for the overdamped, critically damped, and underdamped systems. Presenting the solutions graphically.

4. Numerical solution for nonlinear equations using Newton's method.

5. Applying Newton's method for solving nonlinear equations to calculate the square root and cubic root numerically.

6. Numerical issues in algorithm development and numerically calculating the machine Epsilon ϵ with $1 + \epsilon = 1$.

7. Efficient recursive evaluation of Taylor series for trigonometric and exponential functions without using arrays.

8. Numerical computing of Fourier series for piecewise continuous functions. Present summation of Fourier series with different number of terms in comparison with the piecewise continuous function graphically.

9. Comparison study of algorithm development using recursive functions versus recursive evaluations usig loops.

10. Numerical solution for nonlinear equations using the bisection method, with a comparison study of the numerical efficiencies of different algorithms, such as recursive versus non-recursive algorithms, and selection of different convergence criteria.

11. Efficient and recursive evaluation of polynomials using arrays, to avoid the sensitivity of computing high-order polynomials numerically.

12. Linear interpolation and their applications. Present solutions graphically.

13. Linear regression and their applications. Present solutions graphically.

14. Random number generation and their applications. Using the Monte Carlo method to calculate the approximate value of π. Present solutions graphically.

15. Vector and matrix operations using nested loops.

16. Solving a system of linear equations $\mathbf{Ax}=\mathbf{b}$ and their applications in engineering.

17. Bubble sort and selective sort, and their applications.

18. Efficiently interfacing hardware and accessing memory using pointers.

19. Developing a statistics library (for modular programming and code reuse for a function) with algorithms for calculating the mean, median, standard deviation, maximum, minimum, sorting, and searching, as well as their applications in engineering such as the product quality control.

20. Developing a relatively large-scale program with data structures for user-friendly processing of student information stored in data files. Separating the GPA library as API, based on the previously developed statistics library (code reuse for a library), from the main application program. The program uses software design principles in C for developing large-scale application programs and includes all major programming features, scuh as data structures, enumerated types, global and static variables, loops, decision making, functions in libraries, recursive functions, array processing, and files.

Projects on Physical Computing

1. Project 1: Introduction to Physical Computing
   Introduction to Raspberry Pi
   Install Linux Operating System on Pi
   Introduction to Linux

2. Project 2: Blinking an LED
   Breadboard for interfacing electronic components
   General purpose input and output (GPIO)
   GPIO viewer to interact with GPIO
   Data types and Variables
   Using Wiring C functions for interacing GPIO
   Debugging for Physical Computing

3. Project 3: Push-Button for Traffic Light Control
   Multiple digital input and output
   The while-loop
   The if-else statement

4. Project 4: Analog Input with Photo-resistor
   Serial Peripharal Interface (SPI)
   I2C
   Functions

5. Project 5: Graphing Light
   Data Acquisition and Visualization
   Arrays
   Plotting