CDS 351 - CSI 501
Introduction to Scientific Programming
Syllabus

Instructor: Dr. Fernando Camelli
Contact Info: fcamelli@gmu.edu

Date: Wednesday 4:30 pm – 7:10 pm
Place: Innovation Hall, room 336, Fairfax Campus
Office Hour: Wednesday from 3:00 pm to 4:00 pm
Prerequisites: Permission of instructor

Description:
This course focuses on elements of programming using C/C++ and Fortran languages. The main goal of this class is to familiarize students with basic concepts of programming in computational sciences. Students who complete this course should be able to develop their own code, program algorithms, manage the input and output of data, as well as compile and link codes with other libraries (i.e. LAPACK).

The following topics will be covered: basics of programming, prototyping, and algorithm programming. The programming concepts will be applied to basic problems in computational sciences, including computational geometry and computational mathematics.

In order to take this class, students must be familiar with basic concepts of analytic geometry and calculus, matrix algebra, and elementary differential equations.

Programming topics:

- Introduction to programming, pseudo code, basic logic for programming. Concept of variable and array in C/C++ and Fortran 90 (C++&F90). Basics of compilation.
- C/C++&F90 programming structures and styles.
- Operators and expressions, variable names and constants in C/C++&F90.
- First program “hello world” in C/C++&F90.
- Control of flow and logical expressions in C/C++&F90. Statements and blocks, if-else, else-if, switch, loops - while and for, loops - do - while, break and continue (Taylor’s expansion of functions, numerical integration of functions, and calculation of series).
- Functions in C/C++&F90. Basic of functions, return variables, external variables, scope, static variables (trigonometry functions and computational geometry problems).
- Pointers and arrays in C/C++&F90. One-dimensional arrays vs. multidimensional arrays (vectors and matrices, spares matrix basics, conjugate-gradient methods).
- Data structures and types in C/C++&F90. Derived data structures (application to complex numbers, data structures: arrays, linked lists, binary trees, and heap data structure).
- Input/output in ASCII and binary format in C/C++&F90. File processing.
- C/C++&F90 preprocessor. Linking with other libraries (LAPACK, basic MPI).
- Debugging and profiling tools. Management of projects using make.
- Calling F90 subroutines from C/C++. Calling C/C++ subroutine from F90. Linking C/C++ and F90 modules.

Science applications:

- Computational Geometry: polygon triangulation and partitioning, search and intersection, point location, geometric data structures (quadtrees, octrees, interval trees, segment trees, multi-level trees, kd-trees, range trees), distance fields, etc.
- Computational Mathematics: system of equations (Gauss elimination, LU factorization, partial pivoting, sparse matrices), interpolation, numerical differentiation and integration (finite difference, trapezoid’s rule, Simpson’s rule), ordinary differential equations.

Homework and Projects:
The homework/projects include developing codes to solve common scientific problems in computational mathematics and computational geometry. Some of these projects will be focused on the input/output of data commonly use in computational sciences.

The challenge of the homework assignments will be different for undergraduate and graduate students. The following list of assignments provides the differences between undergraduate and graduate requirements:

1) Week 1: Simple programs and data types
    a. For undergraduate and graduate students:
    Write your first program in Fortran and C, hello project and Makefile.
    b. For undergraduate and graduate students:
    Data types in C and Fortran and flow control assignment: basic loops and if statement programs.

2) Week 2: Arrays
    a. For undergraduate and graduate students:
    Write a program to allocate arrays in Fortran and C. Use of data structures. Write a program to read real number, calculate the average and standard deviation. Use arrays.

3) Week 3: I/O
    a. For undergraduate and graduate students:
    Write and I/O program to read and write data from an ASCII file with a given format. Differences between Fortran and C.
    b. For graduate students:
    Write a program to read VTK and NetCDF files using the API provided by VTK and NetCDF. The program will use external or third party libraries. Using a Makefile to manage and compile the project.

4) Week 4 and 5: Functions in Fortran and C, interfaces and prototypes
    a. For undergraduate and graduate students:
    Write a program that includes functions to calculate convert temperature from Fahrenheit to Celsius; and from Celsius to Fahrenheit. The temperature and correspondent unit are input from the console, and the answer should output to the console. The students have to write a version in Fortran and C of the assignment.
    b. For graduate students:
    Write a code to build a quadtree and binary tree for a given set of points. Implement functions to insert points, search points around points with a given radius, and save the data structure for the given points. This assignment can be written in Fortran or C.

5) Week 6 and 7: Recursion, Taylor's series, Simpson rule for integration, and linear systems
    a. For undergraduate and graduate students:
    Write a code that uses recursion to calculate the factorial number of an integer. This program must be written in Fortran and C.
    b. For undergraduate and graduate students:
    Write a program that will integrate a polynomial function of order n with coefficients a_i between the integration limits a and b.
    c. For undergraduate:
    Write a program to solve a linear system of n equations and n unknowns. Use Gauss elimination. This program can be written in either Fortran or C.
    d. For graduate students:
    Write a program to solve a linear system of n equations and n unknowns. Use sparse matrices. This program can be written in either Fortran or C.

6) Week 8 and 9: Splines, and Matrix decomposition
    a. For undergraduate and graduate students:
    Write a program to handle spline interpolation. This program should be written in Fortran and C.

b. For graduate students:
Write a module/function that calculates the singular value and the Cholesky decompositions.

7) Week 10: OpenGL and C
   a. For undergraduate and graduate students:
   Write a program that generates a Sierpinski gasket. The graphics modules/subroutines to handle the visualization will be provided as external libraries to the students.

8) Week 11: Ordinary differential equations
   a. For graduate students:
   Write a fourth order with adaptive step size Runge-Kutta solver for ODE's.

9) Week 12: Calling Fortran subroutines from C and vice versa, and LAPACK
   a. For undergraduate and graduate students:
      i. Write a program in C that calls a subroutine in Fortran that initialize an array to zero. The arrays is defined in the C portion and passed to the Fortran module.
      ii. Write a program in Fortran that calls a function in C that initialize an array to zero. The array is defined in the Fortran portion of the code and passed to the C function.
   In both cases write the array from console.
   b. For graduate students:
   Write a code that uses LAPACK to solve a system of linear equations. Examples of systems of equations to be solved will be provided from the instructor.

Exams: final project

Note:
1) Projects for undergraduate students will be related to computational geometry problems and provided by the instructor. It will be one common project for all the undergraduate students.
2) Projects for graduate students will be related to their area of interest or research and they must be approved by the instructor.

Grades: homework and projects (70%), final project (30%)

Class URL: http://www.cds.gmu.edu/~fcamelli/academics/csi501.html
Note: Presentations in PDF format will be posted online after lectures for students.

Text Book (not required but suggested):
1. *"C Programming Language"*, by B. W. Kernighan, and D. M. Ritchie.
2. *"Fortran 95/2003 Explained"*, by M. Metcalf, J. Reid, and M Cohen.
3. *"The Fortran 2003 Handbook"*, by J. C. Adams, W. S. Brainerd, J. T. Martin, R. A. Hendrickson, and R. E Maine.
4. *"Computational Geometry in C"*, by J. O'Rourke.
5. *"Computational Geometry: Algorithms and Applications"*, by M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf.

Supplement Reference Books:
6. *"Algorithms in C"*, by R. Sedgewick.
7. *"Memory as a Programming Concept in C and C++"*, by F. Franek.
8. *"Geometric Data Structures for Computer Graphics"*, by E. Langetepe, and G. Zachmann
9. *"C A Reference Manual"*, by S. P. Harbison III, and G. L. Steele Jr.
10. *"Numerical Analysis"*, by T. Sauer.