MPATE-GE 2618: C Programming for Music Technology

Syllabus

Instructor

Dr. Schuyler Quackenbush
schuyler.quackenbush@nyu.edu

Lab Teaching Assistant

TBD

Description

MPATE-GE 2618: C Programming for Music Technology is an intensive, graduate-level introductory course in programming concepts and computer science with a focus on software design, algorithms, and data representation for digital signal processing and audio applications. Assignments consist of extensive programming in C.

Topics include:

- C programming: syntax, primitive types, iteration, conditional expressions, functions, arrays, pointers, dynamic memory allocation, standard libraries.
- Software development: problem decomposition, abstraction, data structures, implementation, debugging, testing.
- Algorithms: design, specification, and analysis.
- Data representations for signal processing and audio applications.
- Introduction to audio APIs

Prerequisites

No prior programming experience is required.

Class Times

Lectures (MPATE-GE 2618) meets on Tuesday and Thursday 1:40-2:55 in room TBD of the Education building (35 W. 4th St).

Lab (MPATE-GE 2617) meets Thursday 3:30-4:45 in room TBD of the Education building (35 W. 4th St).

NOTE: All students enrolled in MPATE-GE 2618 must also take MPATE-GE 2617.

Office Hours

Weekly office hours will be in the time between the Thursday lecture and the lab. Location TBD by student and instructor.

Class Materials

All class materials are available at the NYU Classes course website:

- Resources
  - Lecture slides as PDF
  - Any other course material
- Assignments
  - Problem description and any associated source code and data
  - Final project information

Problem Sets

Problem sets will be distributed via NYU Classes Assignments. Each will have a posted due data. Students should post completed assignments via the same NYU Classes Assignment mechanism.

Quizzes

The course will have at least one quiz. The quiz will be "closed-book." However, you may utilize during each quiz one two-sided page (8.5" × 11") of notes, typed or written, but nothing else.

Final Project

The final project will be your opportunity to put your programming skills to use and implement your own software application. As long as your project is written in C, the nature of your project is entirely up to you, albeit subject to the instructor's approval. You are welcome to utilize external libraries and hardware provided that the instructor has access to all of these things and you are able to present it in class at the end of the semester.

Final projects include a written report, the actual program code, and a presentation to the class.

Students in the music technology program are encouraged to choose a project topics that involve processing of audio signals, including real-time audio I/O.

Final Exam

The course does not have a final exam.

Grades

Final grades for the lecture class and lab will be determined using the following weights:

Problem Sets:	70%
Quizzes:	10%
Final Project:	20%

The instructor will post grades for each assignment, quiz and final project on the NYU Classes Gradebook.

Books

Required text:

Programming in C, Fourth Edition by Stephen Kochan

Optional text for students less comfortable with programming:
Absolute Beginner's Guide to C, Second Edition by Greg Perry

Optional recommended audio programming text:
The Audio Programming Book, R. Boulanger and V. Lazzarini (Eds.)

Platforms

Assignments are designed to be implemented on a UNIX-based system such as Mac OS X or Cygwin on Windows.

Course Outline

- Introduction
    - Book
    - Problem sets, Final project and Lab
    - Course grading
    - About instructor

- Programming problem statement
    - Algorithms
- Terminal
    - Bash commands
    - History of Unix
    - Directories
    - Unix $PATH
    - Redirection and pipes
- Compiler, Assembler, Machine code
- Basic C program
    - Hello world
    - Comments in code
    - Code structure

- Binary numbers
    - 2s complement signed integer
    - ASCII for characters
    - float, double
- Versions of C
    - ANSI
    - C99
- C is Strongly typed language
- Statement syntax
- Variables
    - char, int, short, long, long long, float, double
    - variable

        - declaration
        - assignment
    - Expressions
        - Integer division
        - Promotion
        - cast
    - Unsigned
    - Bool
    - #define for defining constants
    - typedef
        - typedef unsigned int counter
- Operators
    - Arithmetic
        - Precedence
        - Cast
        - Promotion in expression evaluation
    - ++, += and other variants
    - Relational
    - Boolean
    - Logical
    - Bit operations

- Flow of execution
    - Conditional
        - If else
        - ? :
        - switch
    - Looping
        - For
        - Break
        - While
        - Do while
    - Style
        - Brace position (K&R or Microsoft)
    - Conditional
        - switch
        - ? :
        -
- Logical operators

- Arrays
    - Integer
        - Declaration
        - Initialization

  - Array access and assignment
    - A[i]
    - Indexed from zero
    - “off-end” access or assignment
  - Character
    - Null terminated
    - String functions (“n” variant)

- Functions
  - Args
    - Call by value (is the default)
    - Call by reference
  - Return values
    - Int, double
    - Char * (example of “status strings”);
  - Function prototypes
    - When do you need them within a single file
    - *.h in multiple files
  - Variable scope
    - Local automatic (on stack)
    - Within {} scope e.g. for (int i=0; …) loop
    - Global (in heap)
    - static
- Command line arguments and main()
  - Parsing command line
    - Pointer to string
  - Useful functions
    - atoi, atof,
  - #define and enum {}
  - Array of strings
    - Const modifier, e.g. for “message” strings
- Command line processing
  - Designating options
  - Processing of optional arguments
  - Qualifying arguments

- File I/O
  - Fopen
  - Fclose
  - Character I/O – ASCII
    - fscanf, fprintf
    - getc, putc
  - File I/O – Binary
    - fread/fwrite

  - Function to read a line of text

- Enum, const, typedef
- Pre-processor
  - #if () #else #endif
- Structures
  - Example: Student
  - Accessing structures
    - Direct
    - Pointer to struct
  - Typedef and structures
    - Example
  - Arrays of structures
- Pointers
  - Arguments to functions
  - Arrays
    - Index
    - Pointer
    - Pointer offset and pointer arithmetic
  - Pointers to structures
  - Pointers to functions
- Bit operators
  - Bit operators
    - and, or
    - shift
  - Bit test

- Characters
  - Characters
  - Strings

- Using random numbers
  - Numerical integration
- Recursion
  - Base case
  - Recursive call
  - Example programs
- Memory allocation
  - Why: when array size is unknown at compile time
    - Read in a wav file
  - Malloc, calloc and Free
  - Multi-dimensonal arrays
  - Arrays of structures
  - Memory leaks – malloc() without free()

- Memory usage
    - Virtual memory
    - Pages
    - Protection
    - Caching
    - Calling and Stack frame
    - Variables
        - Global
        - Automatic local
        - Static local
- Computational efficiency
    - Big O
    - Linear Log, Polynomial, Exponential
- DFT, FFT
- Searching algorithms
    - Linear
    - Binary
    - Hash
- Hash tables
    - For searching
- Sorting: quicksort overview
- Multi-file programs
- Linked lists
    - Alternative to arrays
    - Grow
    - Re-order
    - Singlely and doubly linked lists
- Binary Trees
    - Data representation
    - Huffman code
- Search
    - Exhaustive
    - Greedy
- Example problems
    - Traveling Salesman
    - Shortest Path
    - 0-1 Packpack
- Convolution
    - Time Domain
    - Frequency Domain
- Examples
    - Reverberation
- Using Libraries

- Real-Time operation
    - latency
- Multi-threaded code
    - Callbacks
- PortAudio
- LibSndFile
    - Open, Read, Write, Seek, Close
- NCurses
    - Non-blocking keyboard input
    - Non-tty output
- OpenGL
    - Graphics window
    - Plot waveform
    - Plot Spectrum
- C++ Classes
    - scoped data and member functions