Welcome to
CS 241
Data Organization using C

Instructor: Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/
Office: Electrical and Computer Engineering (ECE)
Room 233

1/17/2017

Computer Science Course Sequence

1. CS-105: Intro programming JavaScript and HTML5, currently optional.
2. CS-152: Intro programming Java. More rigorous than 105.
3. CS-251: Intermediate programming Java.
3. CS-241: Data Organization using C.

2. CS 357: Declarative Programming.
2. CS-351: Design of Large Programs.
2. CS-361: Algorithms I.
Course Description

CS-241 is an introduction to the C Programming language, an introduction to using a command-line interface in the Linux operating system, and an introduction to machine level data organization and memory allocation.

Students taking this course should already be familiar with basic concepts of computer programming such as variables, conditional control flow and loops and have a solid base in Algebra I.

Developing mastery of these fundamental concepts is one of the goals of CS-241.

Students in CS-241 author many C programs:

- Lab assignments will be short and simple.
- Projects will be more interesting and touch on a wide range of computer applications including encryption, numerical analysis, databases, artificial intelligence, genetic algorithms and games.

Quiz: Write a function in Java, C, C++, JavaScript, MATLAB or Python that...

1) Takes no arguments.
2) Does not returns a value.
3) Uses a nested loop to display to the console a 15 x 15 multiplication table.
4) The multiples of 1 (1 through 15) must print on the first line with at least one space between each number.
5) The multiples of 2 (2, 4, 6, ... 30) must print on the second line with at least one space between each number.
6) The same must be true for each of the other multiples.
Course Resources

Class website:
http://cs.unm.edu/~joel/cs241/
- Syllabus
- Projects
- Lecture Notes
- Supplemental readings

Blackboard Learn:
- Assignment Drop-box
- Assignment discussions
- Grades

Course Goals

1. Read and apply the C syntax covered in the textbook (The C Programming Language by Kernighan and Ritchie).

2. Without a computer, determine the output of C language source code involving triply nested loops, conditional control flow, function calls, pointers, arrays, arithmetic, logical and bit operators, structures and memory allocation.

3. Use a Linux command-line environment to manipulate files, and directories, and to edit, compile, run and debug C programs. This includes the use of simple makefiles and a low level debugger such as valgrind.

4. Implement, in C, any given algorithm with a complexity level equivalent to that of quicksort or a doubly linked list with accuracy, efficiently and clarity.

5. A small taste of using CUDA / OpenCL.
Course Grading

- 45% Programming Projects.
- 25% Exams (midterm and final)
- 20% Laboratory Programming Assignments.
  - Attendance is required. After missing two labs, -2 points per missed lab from final course grade.
  - The minimum grade on a lab turned in with reasonable effort will be a 10/20 if the student attended the associated lab class. Does not apply to projects.
- 10% Lecture quizzes
  - Approximately 30, i-clicker, attendance required.
  - Missed quizzes cannot be made up – *for any reason!*

Turning In Assignments

- All assignments must be submitted into Blackboard Learn.
- e-mailing code is useful for getting help with debugging, but an assignment turned in by e-mail counts for nothing 😞.
- Late projects/assignments receive a 5% per day penalty.
- Assignments are not accepted after the next class period past the due date (since solutions will be posted and explained).
- An assignment is “turned in” on the Blackboard Learn date stamp of the final version submitted.
- Every project will have extra-credit options that can be used to cover missed work - or to boost your grade.
- If you do extra stuff that you think is cool, *show me!*
- Each student's lowest lab grade (not project) will be dropped.
Assignment Grading: 1 of 2

- In general, assignments are graded by your lab instructor.
- When you have a question about how something was graded, **first**, ask the person who graded it.
- The lab instructors must follow strict grading rubrics to score your assignments.
- Sometimes the rubrics miss-fire and cause more points to be taken off than should be for an error I did not think of. Such miss-fires need to be brought to my attention so that I can let all the graders know of any change for uniform grading.

Assignment Grading: 2 of 2

Work the system:

- Assignments have grading rubrics and are graded by compiling them and running them on given data files.
- If your program does not compile or expects the wrong number of spaces or commas, then your program will miss most of the points on the rubric.
- Therefore, you get a much higher grade if you turn in a program that works perfectly with some of the input data than a program that works almost correctly with all of the input.
- Some of the tests are easy to pass and some are hard. Some tests just test different requirements.
We will use i-clicker for quizzes in lecture only.

We will use i-clicker every lecture.

If you have already registered your i-clicker on the web for a different class *this semester*, then you do not need to register again.

Register your iclicker in Blackboard Learn under CS-241.

Some classes may require you to register in a different way or with a different ID number. If so, register again for this class in this way.

One i-clicker can be registered to more then one person (as long as no two of them are in the same i-clicker class at the same time).

---

**Quiz Question #1:**

Do you have your i-clicker?

a) Yes – I am ready to go.

b) I bought one from the bookstore, but forgot it.

c) My dog ate it.

d) No – I did not get one yet.

e) What is an i-clicker anyway?
Forgotten i-clicker?

- If you forget your i-clicker, you may borrow one of my loner i-clickers.
- Each loner i-clicker has an animal picture.
- If you borrow an i-clicker, then to get credit, you must **e-mail me on the same day:**
  - **Subject:** CS-241 borrowed i-clicker
  - **Body:** Name, Date, and Animal.
- There is a limit to how often an i-clicker can be borrowed.

Working Together but do not Cheat

- Working together and helping one another on all projects is highly encouraged. This includes discussion of project:
  - *specification*,
  - *algorithms*,
  - *data structures*,
  - and *test cases*.
- Do **not** share code.
- It is considered cheating to leave your code (paper or electronic copies) where others can find it. You responsible for the security of your intellectual property.
Computer Access

- Need to work on a CS Linux machine.
- See: Instructions for obtaining a CS account and its Terms of Service (TOS) on the class website.
- Use Putty (or some other SSH) to connect: moons.cs.unm.edu
- Putty is free and runs from a single executable.
- Therefore, in addition to your UNM account, you will need a cs.unm.edu computer account.
- With a CS computer account you can access *.cs.unm.edu and use the CS Linux lab on the third floor of FEC.

Programming vs Natural Language

- The entire C vocabulary consists under 40 reserved words.
- There are many libraries, such as math and stdio. However, these are the proper nouns of the language.
- A person can be fluent in a language without knowing the vast majority of its proper nouns.
- Proper nouns are learned as needed, and can be forgotten when no longer needed.
- Like natural languages, programming languages have punctuation and syntax rules (e.g. In C, every statement is ended with a semicolon). Programming languages, however, have much fewer rules than natural languages.
Small Language with Complex Usage

- Programming Languages are much smaller and easier to learn than natural languages.
- However, programming languages are primarily used to express complex branchings of conditional logic that far surpass common uses of natural languages.
- Logic skills have strong carryover from one programming language to another.

The C language is:

**Strongly Typed:** Identifiers must have strictly defined and invariant types before being used.

**Imperative:** Statements change the program's state.

**Structured:** Makes extensive use of block and loop structures - in contrast to using simple tests and jumps such as the goto statement.

**Procedural:** Based upon the concept of breaking a complex program into multiple, named, functional units call procedures, routines, subroutines, or functions.
Isn't C Old & Outdated?

C is widely used in industry for:

**Embedded Systems and Firmware:**
C is very lightweight and it's way easier to port a C compiler to a small platform than it is to port a C++ or Java compiler.

**Compilers and Device Drivers:**
- C is "close to the machine".
- Need to control every bit of every byte.
- Need to specifically use a particular level of cache, main memory, graphics memory, ....
- Need to specifically do a "hard write" verses a "buffered write".