A First Course in Computing for ALL Engineering Students

Yale Patt
The University of Texas at Austin

What a Computer is to an Engineer
(and in particular, a non-Computer Engineer)

* ***A Tool used to solve problems***
  (MatLab, for example)

    -- Computers process algorithmically
    -- Computers process numbers

* ***An embedded processor that controls a system***
  (airplane, factory, heart monitor, traffic flow, etc.)

    -- Sensors
    -- Actuators
    -- Functions
    -- Concept of State

What an Engineer needs to know
-- to Use the tool
-- to design the system

* ***How the computer works***

* ***How the numbers are represented***

* ***How an algorithm "works" on a computer***

* ***From sensors (inputs)via "programs" to actuators (output)***

What an Engineer does NOT need in this course

* ***Excel***
* ***Word***
* ***Web browsing***
* ***Rote learning of programming***

"Problem solving is programming."

-- Dr. Nick Tredennick
(in Microprocessor Report, May 3, 2004)

* ***Engineers have ALWAYS solved problems. (That's what engineers do.)***

  -- We don't describe our engineering problem to a Mathematician, and expect him/her to come up with the equations that specify the problem.

  -- We expect the engineer to be able to describe the problem mathematically.

  -- In those cases where we do enlist the help of a mathematician, we make sure there is meaningful dialogue between engineer and mathematician.

* ***TODAY'S problems are solved by computer programs.***

  -- Can we entrust the problem solving task to one who knows nothing about the base technology?

  -- Or, should we expect the engineer to be able to describe the problem algorithmically?

  -- In those cases where we do enlist the help of a programmer, should we make sure there is meaningful dialogue between engineer and programmer?

Why "Intro to Computing" is Essential to ALL Engineering Curricula (and deserves more than token exposure to programming)

* ***A Core Competency (like physics, calculus)***
  -- Use the tool, design the embedded processor

* ***Engineering is about design***
  -- Students can have a meaningful design experience IF exposed correctly (modified bottom-up) to Intro to Computing

* ***Engineering is about Tradeoffs***
  -- Lots of examples of tradeoffs in programming (e.g., recursion vs. iteration)

* ***Engineering is about State***
  -- Lots of examples of state in computer hardware and software

* ***An ACTIVE learning experience in the freshman year***
  -- The student programs from scratch
  -- The student debugs his own program
  -- The student succeeds.

* ***Engineers want causal, deterministic systems***
  -- Everything should make sense
  -- Matlab, etc. become obvious next steps

Why Intro to Programming in X is the Wrong Answer

* ***Approach is almost always top-down***
  -- Results in memorizing, not understanding

* ***Effects of Memorizing***
  -- Students don't ever get it.
  -- If not 100%, they can't figure out their mistakes
  -- Cookbook education

* ***Provides no insight into the important tools (Matlab, etc.)***

* ***Provides no insight into how the embedded processor interacts with their system***

* ***Provides no real insight into Tradeoffs, State***

What is Important?

* ***Top-down design,***
  Bottom-up learning for understanding

* ***Abstraction is vital, but…***

* ***Not bottom-up,***
  but "motivated" bottom-up

* ***Engineering is about DESIGN,***
  first understand the components

* ***From Concrete to Abstract***
  (Dijkstra notwithstanding)

* ***Cut through protective layers***

* ***Memorizing is not understanding***

* ***Students do better working in groups***