Programming in the small

vs.

BIG Picture

Lenny Pitt*
Director, Ugrad Programs
University of Illinois

* [Theory; Informatics; K-12 programming]
The prompt....

Do we need to teach detailed programming in a first course to majors, and what we should teach in a non-major course to convey computing and computational thinking?
### Programming in the small

<table>
<thead>
<tr>
<th>Topic</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to computing concepts.</td>
<td>2</td>
</tr>
<tr>
<td>Constants, variables, expressions, precedence of operators, assignment, basic input and output, built-in functions.</td>
<td>3</td>
</tr>
<tr>
<td>Structured programming ideas</td>
<td>2</td>
</tr>
<tr>
<td>Conditionals</td>
<td>3</td>
</tr>
<tr>
<td>Iteration</td>
<td>4</td>
</tr>
<tr>
<td>Data files, ASCII</td>
<td>4</td>
</tr>
<tr>
<td>Modular programming: Functions with and without return values; parameters.</td>
<td>4</td>
</tr>
<tr>
<td>Arrays, aggregate data types, collections.</td>
<td>4</td>
</tr>
<tr>
<td>Operations on aggregate data types</td>
<td>4</td>
</tr>
<tr>
<td>Recursion</td>
<td>3</td>
</tr>
<tr>
<td>Object-oriented programming</td>
<td>4</td>
</tr>
</tbody>
</table>
Variations

• Non-majors: science
  – tool use [mathematica; matlab; unix]
  – focused MPs [numeric computation…]

• Non-majors: general
  – media [recruiting; retention]
  – tool use [DB, spreadsheet macros, …]

• Majors
  – breadth of application
  – infusion of CS into CS1
  – aimed at recruiting and/or retention
  – a challenge, given constraints, directions
Programming in the LARGE

• *Using* abstractions instead of *creating* them
• APIs... familiarity, use.
• UML  [see “UML: The Positive Spin”]
• other software engineering tools

“Nahhh”

• concrete before the abstract; wait for CS2
• machine proximity for understanding, empowerment
• lost in layers of abstraction
• client/cs communication issue
Programming in the LARGE

- Using abstractions instead of creating them
- APIs... familiarity, use.
- UML [see “UML: The Positive Spin”]
- other software engineering tools

“Sure...”

- Only trivial programs are in a vacuum
- CS1 Programming in the MEDIUM since dawn of man
  - MPs that build
  - Instructor-supplied packages
The Big Picture

- intro programming
- data structures
- discrete math
- programming languages
- automata, decidability
- algorithms
- operating systems
- systems programming
- compilers
- digital logic
- computer architecture
- numerical methods
- software engineering
- networking
- verification
- computational complexity
The Big Picture

- intro programming
- data structures
- discrete math
- programming languages
- automata, decidability
- algorithms
- operating systems
- numerical methods
- software engineering
- verification
- computational complexity

CS0: CS for all
choices must be made...

• Use “Computational Thinking” to guide choice?
Computational Thinking

- intro programming
- data structures
- discrete math
- programming languages
- automata, decidability
- algorithms
- operating systems
- systems programming
- compilers

- digital logic
- computer architecture
- numerical methods
- software engineering
- networking
- verification
- computational complexity
choices must be made…

- Use “Computational Thinking” to guide choice?
- Use intended audience/use to guide choice.
choices must be made...

- Use "Computational Thinking" to guide choice?
- Use intended audience/use to guide choice.

• ACM/IEEE CC 2001
  CS1/breadth approach extends intro sequence. Other CS1 approaches discussed
  • Variants already mentioned
  • Can "put off" key skill development and concepts - what’s the hurry.
• Retain and/or Recruit
choices must be made…

- Use “Computational Thinking” to guide choice?
- Use intended audience/use to guide choice.

Examples:
- Hardware/software hierarchy: YES
- TMs: YES
- Historical/Ethical: YES
- Programming: SOME
choices must be made…

• Use “Computational Thinking” to guide choice?
• Use intended audience/use to guide choice.

CS0:
• less depth than CS1-breadth
• less focus on programming
• inclusion of broad application areas
<table>
<thead>
<tr>
<th>TOPIC DESCRIPTION</th>
<th>CONTACT HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data and Representation</strong>, including binary numbers and addition, representation of text, sound, images, and motion pictures. Elementary compression techniques.</td>
<td>3</td>
</tr>
<tr>
<td><strong>Logic gates and simple circuits</strong>; incl. truth-tables, flip-flops, storage devices</td>
<td>3</td>
</tr>
<tr>
<td><strong>Components of computing system</strong>, stored program, von Neumann architecture, fetch/execute cycle, memory hierarchy, typical machine code.</td>
<td>3</td>
</tr>
<tr>
<td><strong>Operating systems, system software</strong> Š abstracting away hardware, user interfaces, file hierarchy and structure, scheduling and multi-tasking</td>
<td>3</td>
</tr>
<tr>
<td><strong>Programming languages</strong>, hierarchy (machine..assemblyÉ) imperative languages, typical high-level primitives/pseudocode, iteration.</td>
<td>4</td>
</tr>
<tr>
<td><strong>Algorithms</strong>, basic concepts, conditional and compound statements, iteration, loop control, flow charts, search, sort, pattern matching, elementary data structures, notions of efficiency and correctness</td>
<td>5</td>
</tr>
<tr>
<td>Computer as an experimental tool - discrete event simulation, queuing, data visualization.</td>
<td>2</td>
</tr>
<tr>
<td><strong>Databases</strong>: fields, records, tables, keys, queries Š select, project, set operations, join, views.</td>
<td>3</td>
</tr>
<tr>
<td><strong>Networks</strong> and <strong>WWW</strong>, numerous topics incl. history, topology, packets, protocols, security, encryption, viruses, html, xml, queries and search engines</td>
<td>4</td>
</tr>
<tr>
<td><strong>Artificial Intelligence</strong>: philosophical issues, natural language, machine learning, neural networks, expert systems, genetic algorithms, robotics</td>
<td>6</td>
</tr>
<tr>
<td><strong>Applications</strong> across fields: info storage and retrieval, decision support, data visualization, communications, modeling/design and simulation, art-music-video-entertainment including editing of media, education, e-commerce, embedded and real-time systems</td>
<td>2</td>
</tr>
<tr>
<td><strong>Limitations of computation</strong>: uncomputability and halting problem, intractability, other limitations</td>
<td>2</td>
</tr>
<tr>
<td>Miscellaneous topics, review, history, ethics.</td>
<td>1</td>
</tr>
<tr>
<td>Exams</td>
<td>2</td>
</tr>
<tr>
<td>TOTAL</td>
<td>43</td>
</tr>
</tbody>
</table>
choices must be made…

• Use “Computational Thinking” to guide choice?
• Use intended audience/use to guide choice.

Don’t need to recruit
Don’t need to retain
Don’t need to help appreciate
No ethics / history

physical, life, social sciences
Three key areas

1. Simulation (continuous/dynamic, discrete event, stochastic).

2. Data analysis (visualization, clustering, mining)

3. Discrete algorithms (discrete models, complexity, bag ’o algorithms)
Discrete algorithms (discrete models, complexity, bag ’o algorithms)

• Typically two courses in CS curriculum:
  – Discrete mathematics
  – Algorithms

• Focus of two courses is wrong:
  Discrete mathematics: proof, properties of discrete structures, reasoning about them
  Algorithms: proofs of correctness, complexity of existing algorithms; design and analysis of new ones; algorithmic approaches
Discrete algorithms (discrete models, complexity, bag 'o algorithms)

- Survey of discrete structures and use to model problems
- Notion of algorithm and complexity
- Survey of algorithms on discrete structures and complexity.

**OUTCOME:** students should have the skills to take a problem, consider the appropriateness of various discrete representations for the problem, and reformulate the problem in a natural way, determining applicability of various algorithms.
Summary

- Three models:
  - for majors.... stay the course(s)
  - for users .... simulation/data/algorith
  - for “other”.... breadth & appreciation