

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

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

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 Programmng 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, complexity, algorithms, operating systems, systems programming, compilers
- digital logic
- computer architecture
- numerical methods
- software engineering
- networking
- verification
- computational complexity

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

CS0: CS for all

choices must be made...

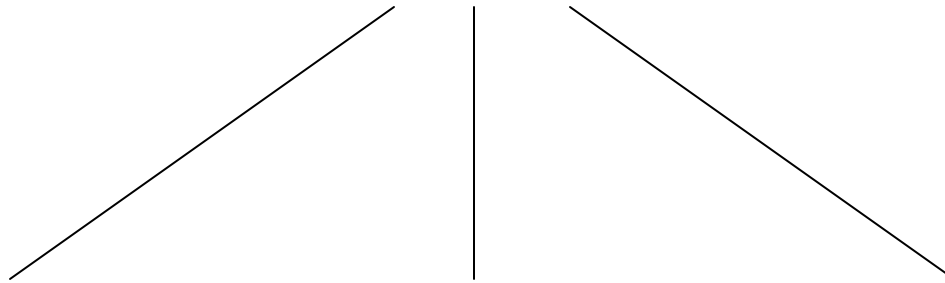
- Use “Computational Thinking” to guide choice?

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.



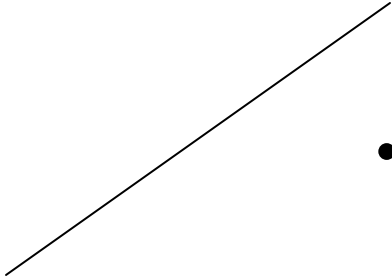
QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.
major

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.
user

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.
other

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:

- Hdwe/software hierarchy: YES
- TMs: YES
- Historical/Ethical: YES
- Programming: SOME

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

major

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

QuickTime™ and a
TIFF (Uncompressed) decoder or
are needed to see this picture.

other

TOPIC DESCRIPTION	CONTACT HOURS
Data and Representation , including binary numbers and addition, representation of text, sound, images, and motion pictures. Elementary compression techniques.	3
Logic gates and simple circuits ; incl. truth-tables, flip-flops, storage devices	3
Components of computing system , stored program, von Neumann architecture, fetch/execute cycle, memory hierarchy, typical machine code.	3
Operating systems, system software Š abstracting away hardware, user interfaces, file hierarchy and structure, scheduling and multi-tasking	3
Programming languages , hierarchy (machine..assemblyÉ) imperative languages, typical high-level primitives/pseudocode, iteration.	4
Algorithms , basic concepts, conditional and compound statements, iteration, loop control, flow charts, search, sort, pattern matching, elementary data structures, notions of efficiency and correctness	5
Computer as an experimental tool - discrete event simulation, queuing, data visualization.	2
Databases : fields, records, tables, keys, queries Š select, project, set operations, join, views.	3
Networks and WWW , numerous topics incl. history, topology, packets, protocols, security, encryption, viruses, html, xml, queries and search engines	4
Artificial Intelligence : philosophical issues, natural language, machine learning, neural networks, expert systems, genetic algorithms, robotics	6
Applications 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	2
Limitations of computation : uncomputability and halting problem, intractability, other limitations	2
Miscellaneous topics, review, history, ethics.	1
Exams	2
TOTAL	43

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

user

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/algorithm
- for “other”.... breadth & appreciation