Central ideas
- Computer science is a discipline
- It aims at solving problems
- The tools include
  - analysis (understanding and breaking down problem)
  - algorithms (step-by-step plans)
  - interaction
- CS has a rational and an empirical side

CS I introduces a curriculum that includes:
- Hardware (architecture, organization)
- Operating systems and networks
- Theory (logic, statistics, discrete math, automata, AI)
- Programming and software engineering

The chronic software crisis
- The global market demands new ways be found to work and do business
- New software is required rapidly
- Software development has tended to be behind schedule
- Software is often unreliable
- Solutions: design, documentation, readable code, structured techniques, object-oriented technology

Social and professional issues
- History of computing
- Social context of computing
- Methods and tools of analysis
- Professional and ethical responsibilities
- Risks and liabilities of computer-based systems
- Intellectual property
- Privacy and civil liberties
[Source: Computing Curricula 2001]

Models of computation
- A model is an abstract description that lets us focus on essentials
- The languages we discuss assume the Random Access Machine model, with stored programs
- Other models: neural, finite automata, Turing machine, parallel random access machine, cellular automata
Computing power

- Brain:
  - $10^{11}$ neurons
  - $10^{15}$ synapses (connections)
  - $10^{16}$ firings/sec (in parallel, 200 mph)
- PC:
  - 1 processor
  - $10^9$ registers (very fast storage)
  - $10^9$ bits RAM (fast memory)
  - $10^{12}$ bits hard drive storage (slow)
  - $10^9$ operations per second
    (serial; 72 million mph)

The stored-program model

- The computer user chooses software and input data:
- A program in effect defines a machine:
- An algebraic formula is a simple program:

The stored-program model

- The computer user chooses software and input data:
- A program in effect defines a machine:
- An algebraic formula is a simple program:

Kinds of abstraction in CS I

- Variables and other expressions with values
- Data types
- Control structures (e.g., loops)
- Programs, functions
- Subprograms
- Classes

Computation and symbol manipulation

- Symbol: an abstraction by which we may represent parts of the real world
- Example: $\{0,1\}$ is a set of symbols
- This set may be used to build symbols of any complexity (numerals, words, pix…)
- We can operate on symbols (add, concatenate, reorder, etc.)

Number (numeral) systems

- Any natural number may be represented in many ways; e.g., twelve:
  - Unary: 111111111111
  - Decimal: 12
  - Octal (base-8): 14
  - Hexadecimal: 0c
  - Binary: 1100
- In all but the unary system, the value represented by a digit depends on its place in the numeral

Expression evaluation is algebraic

- Problem:
  - Let $a = 4$
  - Let $b = 2$
  - Find $a^2 - b + 5$
- Solution: $a^2 - b + 5$
  - $= 16 - 2 + 5$
  - $= 14 + 5$
  - $= 19$
- Algebra: A set of values and a set of operators
Problems
• Write an expression whose value is the area of the surface of a 2' x 3' x 5' box
• ... a box \( w \) feet wide, \( h \) feet high, and \( d \) feet deep
• Write an expression whose value is the 5% sales tax on a purchase of a $199 hard drive and a $79.95 software package
• ... A hard drive costing \( h \) dollars and software costing \( s \) dollars, taxed at \( t \) %

Some sets we will use
• Truth values: \{0,1\} = \{true, false\}
• Digits: \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}
• Natural numbers (G. Peano):
  1. 0 is a natural number
  2. Every natural number has a successor natural number
  3. There are no other natural numbers
• Other sets: real numbers, integers, characters, strings, keywords, rules, operations

Computations may or may not have input
• Problem: Display interest charged when buying a $20,000 car over 3 years at 10%
• Solution:
  \( Y1 = 20,000 \times 0.1 \)
  \( Y2 = Y1 + Y1 \times 0.1 \)
  \( Y3 = Y2 + Y2 \times 0.1 \)
  \( Total = Y1 + Y2 + Y3 \)
  Display Total
• Describe a version of above with inputs

Computations may be interactive
• Typical menu- or command-driven user environment:
  Repeat
    input a command
    execute command
  until command = “quit”
• With interaction, input may depend on previous output
• Almost all computing today is interactive
• Internet computing presents new challenges related to interaction

Operating-system and application software
• The operating system runs at all times
• It is started by a ROM “boot” process
• The OS may support multiple application programs concurrently
• The OS links applications to the hardware
• Both application programs and the OS have user interfaces
• OS examples: Win XP, Mac OS, UNIX

3 models of computing
• Algorithmic
• Sequential interactive
• Multi-stream interactive
Introduction

Features of graphical user environments (e.g., Windows)

- Program loading (Start menu; icons, Run)
- Task switching (task bar)
- Device control (Start / Settings / Control panels)
- Disk directory tree structure
  - program files
  - data files
  - folders
- Menus, icons, dialog boxes
- Drag and drop
- Double-click to execute commands

Objects in a graphical user interface

- A window is an object with position, size, contents
- A window opens, closes, moves, resizes
- Other objects: icons, menus, buttons
- In Windows, clicking on an object with right mouse button gives user access to object’s properties and methods (operations)

Networks connect PCs

- Cables and network cards connect local area networks (LANs)
- Each workstation PC runs its own software
- In client-server computing, the software (e.g., in response to database query) runs on the server

Internet computing

- The Internet is a network of computer networks
- It is connected by service providers and the telephone and TV-cable systems
- Email and Web data travels in packets (each with a small amount of data)
- We access Web pages by supplying a uniform resource locator (URL)
- The web-page data, in Hypertext Markup Language (HTML) form, is communicated to the user’s computer

Guiding ideas

- Algorithm design
- Documentation
- Modular decomposition
- Objects and classes of objects

Steps in problem solving

- Repeat until problem is solved:
  1. Specify the desired results
  2. Design a solution
  3. Code solution in programming language
  4. Compile, test, debug

CS I topics

- Hardware (1)
- Software design (2)
- C and C++ (3)
  - Input/output (3)
  - Simple data types (4)
  - Control structures (5)
  - Subprograms (6)
  - Structure types/classes (7)