

Undergraduate Curriculum and Accreditation Advances

Computing Curricula 2001

Eric Roberts, Stanford University

Snowbird 2002

July 15, 2002



Computing Curricula 2001 (CC2001)



Charter: To review the Joint ACM and IEEE/CS Computing Curricula 1991 and develop a revised and enhanced version for the year 2001 that will match the latest developments of computing technologies in the past decade and endure through the next decade.

Final version of CS report released on December 15, 2001

<http://www.computer.org/education/cc2001/>

<http://www.acm.org/sigcse/cc2001/>



CC2001 Task Force



ACM

IEEE Computer Society

Education Board chair:

Peter Denning

Task Force co-chairs:

Eric Roberts (*editor*)

Russ Shackelford

Steering committee members:

Richard Austing

Fay Cover

Gordon Davies

Andrew McGettrick

Michael Schneider

Ursula Wolz

VP for Education:

Carl Chang

Task Force co-chairs:

James Cross

Gerald Engel (*editor*)

Steering committee members:

Doris Carver

Richard Eckhouse

Willis King



Francis Lau

Susan Mengel

Robert Sloan (*secretary*)

Pradip Srimani

CC2001 Volumes





Computing Curricula 2001

Computer Science

The Joint Task Force
on Computing Curricula

IEEE Computer Society
Association for Computing Machinery





Computing Curricula 2001

Computer Engineering

The Joint Task Force
on Computing Curricula




IEEE Computer Society
Association for Computing Machinery



Computing Curricula 2001

Software Engineering



The Joint Task Force on
Software Engineering Education
Project
(SWEEP)



Computing Curricula 2001

Information Systems

Association for Computing Machinery
IEEE Computer Society
Association for Information Systems





Computing Curricula 2001

Two-Year Colleges

The Joint Task Force
on Computing Curricula

IEEE Computer Society
Association for Computing Machinery



Computing Curricula 2001

Overview

The Joint Task Force
on Computing Curricula

IEEE Computer Society
Association for Computing Machinery

History of Curriculum Reports

- 1967 COSINE report (Commission on Engineering Education)
- 1968 *Curriculum '68* (ACM)
- 1977 *A Curriculum in CS and Engineering* (IEEE-CS)
- 1978 *Curriculum '78* (ACM)
- 1983 *Model Program in CS and Engineering* (IEEE-CS)
- 1989 *Computing as a Discipline*
- 1991 *Computing Curricula '91* (IEEE-CS + ACM)
- 2001 *Computing Curricula 2001* (IEEE-CS + ACM)

Problems with Earlier Curricular Efforts

- The curriculum gave institutions too little guidance.
- Knowledge units are often not as useful as courses.
- The set of common requirements was too large.
- The structure made it difficult to incorporate new areas into the curriculum.
- The curriculum emphasized specific pedagogical approaches for which there had been inadequate testing and development.
- The process did not provide sufficient opportunity for external review.

CC2001 Accomplishments

- Established a strong partnership between ACM and IEEE-CS
- Completed a survey and evaluation of the impact of CC'91
- Articulated a set of principles to guide our work
- Created a structure (KFGs and PFGs) for broad participation
- Secured funding from NSF to support a review meeting
- The process did not provide opportunity for external review.
- Developed a body of knowledge (BOK) for undergraduate CS
- Defined a set of core topics for all CS undergraduates
- Outlined learning objectives for each unit in the BOK
- Specified a list of over 80 courses for undergraduate CS
- Identified desired characteristics for CS graduates
- Implemented web resources to complement written report
- Integrated feedback from conference sessions into the report
- Published three public drafts on the way to the final report

CC2001 Principles

1. Computing is a broad field that extends well beyond the boundaries of computer science.
2. Computer science draws its foundations from a wide variety of disciplines.
3. The rapid evolution of computer science requires an ongoing review of the corresponding curriculum.
4. Development of a computer science curriculum must be sensitive to changes in technology, new developments in pedagogy, and the importance of lifelong learning.
5. CC2001 must go beyond knowledge units to offer significant guidance in terms of individual course design.

CC2001 Principles

6. CC2001 should seek to identify the fundamental skills and knowledge that all computing students must possess.
7. The required body of knowledge must be made as small as possible.
8. CC2001 must strive to be international in scope.
9. The development of CC2001 must be broadly based.
10. CC2001 must include professional practice as an integral component of the undergraduate curriculum.
11. CC2001 must include discussions of strategies and tactics for implementation along with high-level recommendations.

CC2001: Sample Knowledge Unit

DS1. Functions, relations, and sets [core]

Minimum core coverage time: 6 hours

Topics:

Functions (surjections, injections, inverses, composition)

Relations (reflexivity, symmetry, transitivity, equivalence relations)

Sets (Venn diagrams, complements, Cartesian products, power sets)

Pigeonhole principle

Cardinality and countability

1. Explain with examples the basic terminology of functions, relations, and sets.
2. Perform the operations associated with sets, functions, and relations.
3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
4. Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.

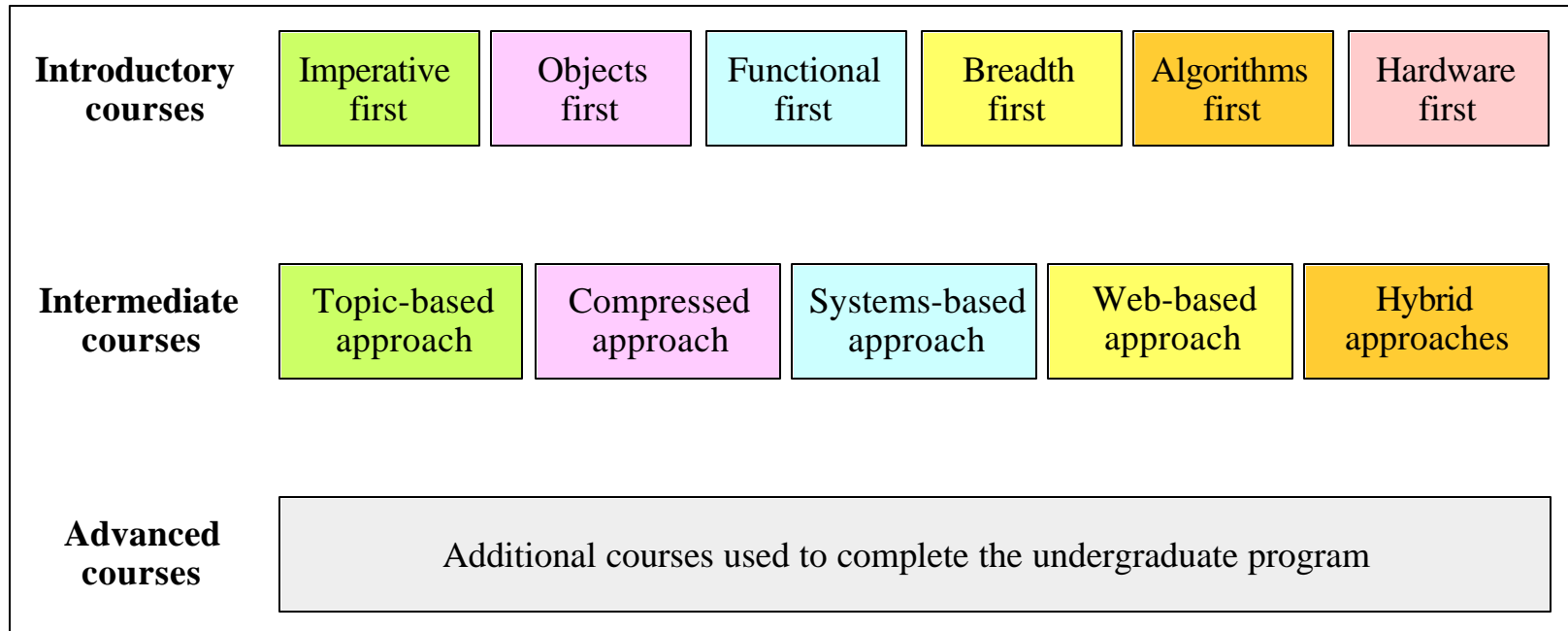
Points to Remember about the Core

- The core is not a complete curriculum.
 - All undergraduate programs must include additional material
- Core units are not intended to be taught in a set of core courses early in the undergraduate curriculum.
 - Some core material will come in the junior or senior year
 - Introductory courses will often include elective topics
- Hours indicate “lecture” hours, not credit hours.
 - Hour estimates do not include preparation or study time
- Hours listed for units indicate the minimum time.
 - You can always include more
- Hours are not as important as learning objectives.

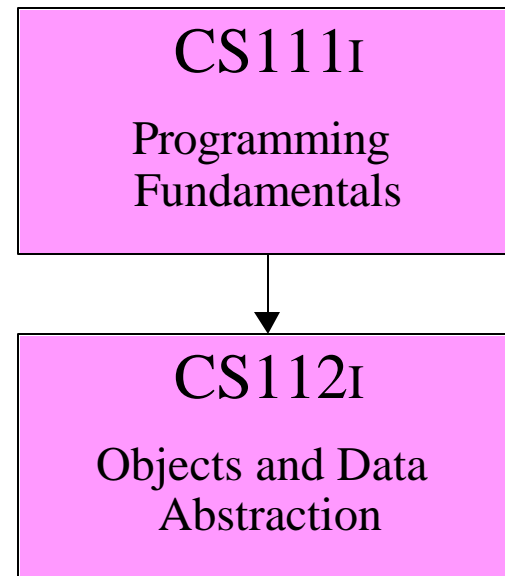
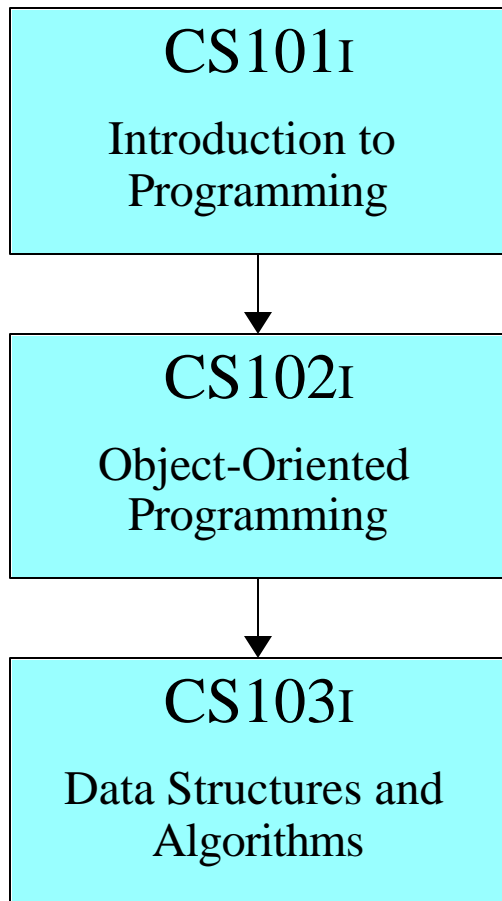
The Undergraduate CS Core

DS. Discrete Structures	43 core hours
PF. Programming Fundamentals	38 core hours
AL. Algorithms and Complexity	31 core hours
PL. Programming Languages	21 core hours
AR. Architecture and Organization	36 core hours
OS. Operating Systems	18 core hours
NC. Net-Centric Computing	15 core hours
HC. Human-Computer Interaction	8 core hours
GR. Graphics and Visualization	3 core hours
IS. Intelligent Systems	10 core hours
IM. Information Management	10 core hours
SE. Software Engineering	31 core hours
SP. Social and Professional Issues	16 core hours
<hr/>	
Total	280 core hours

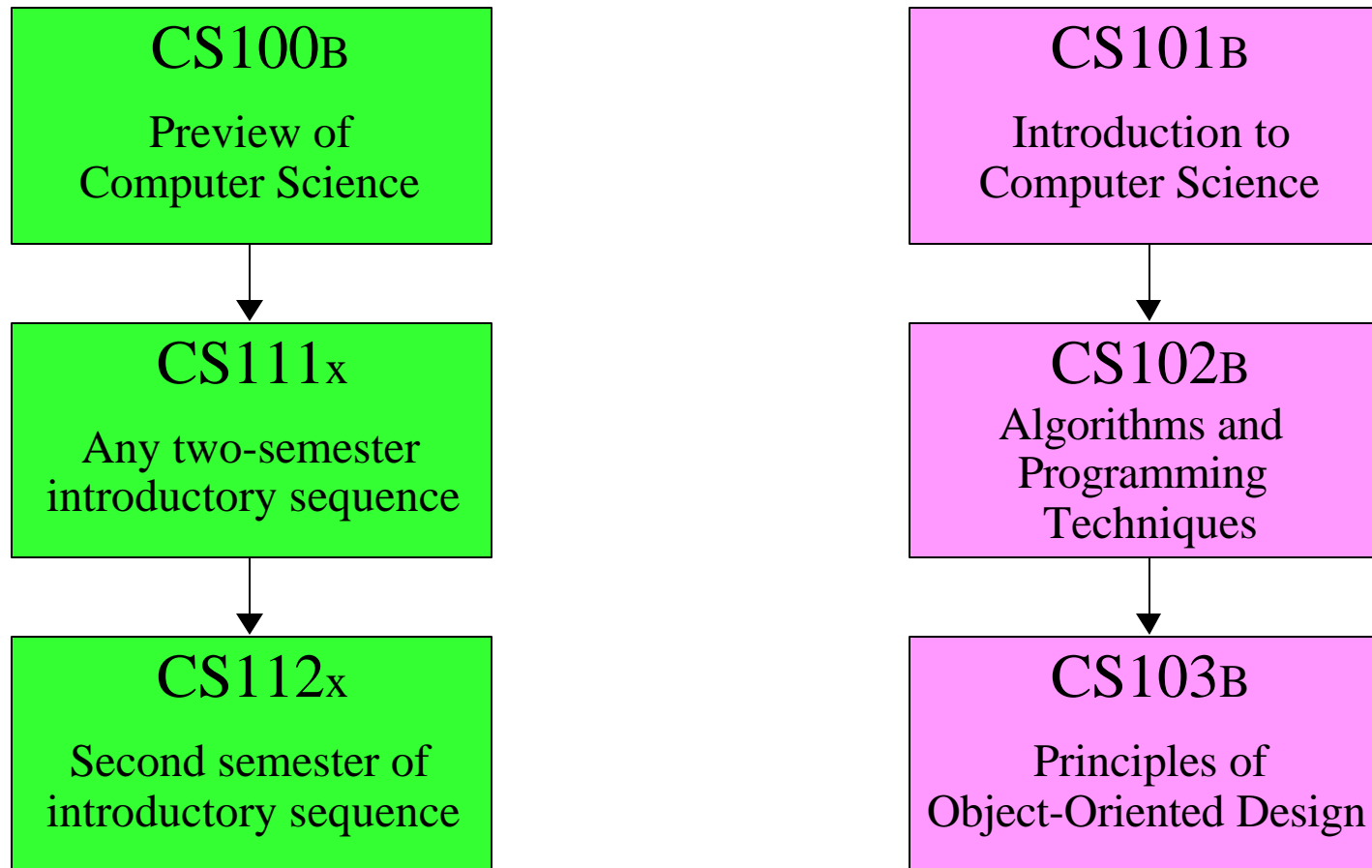
Structure of the Curriculum



Two- and Three-Semester Intro Tracks



Structures for the Breadth-First Model



Sample University Curriculum (US)

CS101i. Programming Fundamentals Calculus I	CS102i. The Object-Oriented Paradigm CS115. Discrete Structures for Computer Science Calculus II
CS103i. Data Structures and Algorithms Science course I	CS120. Introduction to Computer Organization Science course II Probability and Statistics
CS210r. Algorithm Design and Analysis CS220r. Computer Architecture Advanced mathematics elective	CS225r. Operating Systems CS280r. Social and Professional Issues CS elective Undergraduate research project
CS230r. Net-centric Computing CS262r. Information and Knowledge Management CS290r. Software Development Undergraduate research project	CS490. Capstone Project CS elective CS elective

Sample Small College Curriculum (US)

CS111o. Object-Oriented Programming CS105. Discrete Structures I	CS112o. Object-Oriented Design and Methodology CS106. Discrete Structures II
CS210c. Algorithm Design and Analysis CS220c. Computer Architecture	CS226c. Operating Systems and Networking Mathematics elective
CS262c. Information and Knowledge Management CS elective	CS292c. Software Development and Prof. Practice CS elective
CS elective	CS490. Capstone Project

Sample Discipline-Based Curriculum (Three-Year UK Model)

<p>CS101o. Intro to Object-Oriented Programming CS105. Discrete Structures I CS120. Introduction to Computer Organization</p>	<p>CS102o. Objects and Data Abstraction CS106. Discrete Structures II Probability and statistics</p>
<p>CS103o. Algorithms and Data Structures CS210s. Algorithm Design and Analysis CS220s. Computer Architecture CS271s. Information Management</p>	<p>CS226s. Operating Systems and Networking CS240s. Programming Language Translation CS255s. Computer Graphics CS291s. Software Dev. and Systems Programming</p>
<p>CS260s. Artificial Intelligence CS380. Professional Practice CS elective CS491. Capstone Project I</p>	<p>CS326. Concurrent and Distributed Systems CS393. Software Engineering and Formal Spec. CS elective CS492. Capstone Project II</p>

Sample Course Description

CS105. Discrete Structures I

Introduces the foundations of discrete mathematics as they apply to computer science, focusing on providing a solid theoretical foundation for further work. Topics include functions, relations, sets, simple proof techniques, Boolean algebra, propositional logic, digital logic, elementary number theory, and the fundamentals of counting.

Prerequisites: Mathematical preparation sufficient to take calculus at the college level.

Syllabus:

- Introduction to logic and proofs: Direct proofs; proof by contradiction; mathematical induction
- Fundamental structures: Functions (surjections, injections, inverses, composition); relations (reflexivity, symmetry, transitivity, equivalence relations); sets (Venn diagrams, complements, Cartesian products, power sets); pigeonhole principle; cardinality and countability
- Boolean algebra: Boolean values; standard operations on Boolean values; de Morgan's laws

Sample Course Description (continued)

- Propositional logic: Logical connectives; truth tables; normal forms (conjunctive and disjunctive); validity
- Digital logic: Logic gates, flip-flops, counters; circuit minimization
- Elementary number theory: Factorability; properties of primes; greatest common divisors and least common multiples; Euclid's algorithm; modular arithmetic; the Chinese Remainder Theorem
- Basics of counting: Counting arguments; pigeonhole principle; permutations and combinations; binomial coefficients

Units covered:

DS1	Functions, relations, and sets	9 hours (6 core + 3)
DS2	Basic logic	5 core hours (of 10)
DS3	Proof techniques	4 core hours (of 12)
DS4	Basics of counting	9 hours (5 core + 4)
AR1	Digital logic and digital systems	3 core hours (of 6)
	Elementary number theory	5 hours
	Elective topics	5 hours

Sample Course Description (continued)

Notes:

This implementation of the Discrete Structures area (DS) divides the material into two courses. CS105 covers the first half of the material and is followed by CS106, which completes the core topic coverage. Because the material is stretched over two courses—as opposed to CS115 which covers the material in a single course—many of the units are given more coverage than is strictly required in the core. Similarly, the two-course version includes additional topics, reducing the need to cover these topics in more advanced courses, such as the introductory course in algorithmic analysis (CS210).

Although the principal focus is discrete mathematics, the course is likely to be more successful if it highlights applications whose solutions require proof, logic, and counting. For example, the number theory section could be developed in the context of public-key cryptography, so that students who tend to focus on the applications side of computer science will have an incentive to learn the underlying theoretical material.

CC2001: General Requirements

- Mathematics
 - Discrete mathematics
 - Additional mathematics is required, but not constrained to calculus
- Science
 - Students must be exposed to the scientific method
 - Science training can come from a wide variety of fields
- Applications of computing
 - All students must study some area that uses computing in a substantive way
- Communications skills
 - Writing
 - Oral presentation
 - Critiquing
- Working in teams
 - Team work should begin early in the curriculum
 - All students should engage in a significant team project

CC2001: Characteristics of CS Graduates

<i>Cognitive capabilities</i>	Knowledge and understanding Modeling Requirements Critical evaluation and testing Methods and tools Professional responsibility
<i>Technical capabilities</i>	Design and implementation Evaluation Information management Human-computer interaction Risk assessment Tools
<i>Transferable skills</i>	Operation Communication Teamwork Numeracy Self management Professional development

Strategies and Tactics

- The curriculum must be adapted for the local environment.
- The curriculum must reflect the integrity and character of computer science as an independent discipline.
- The curriculum must respond to rapid technical change and encourage students to do the same.
- Curriculum design must be guided by the outcomes you hope to achieve.
- The curriculum as a whole should maintain a consistent ethos that promotes innovation, creativity, and professionalism.

Strategies and Tactics (Continued)

- The curriculum must be accessible to a wide range of students and appeal to their individual strengths.
- The curriculum must provide students with a capstone experience that gives them a chance to apply their skills and knowledge to solve a challenging problem.
- Computer science programs must have adequate computing resources.
- Attracting and retaining faculty will often be a critical challenge for computer science programs.

CC2001: Relationship to Accreditation

The curriculum should be consistent with widely recognized models and standards.

—Guidance for CAC Criteria (2002-03)

Under the ABET 2000 structure, programs seeking accreditation must define concrete objectives and measurable outcomes for the program, and then demonstrate that those goals are being met. We believe that the CC2001 curriculum provides several curricular models that are appropriate for accreditation under the ABET-CAC guidelines. The CC2001 report, moreover, offers both a rationale and an assessment structure for ensuring compliance.