International Society For Technology In Education (ISTE)
Educational Computing and Technology Programs
Standards for Computer Science Initial Endorsement

Introduction

Computer Science (CS)--Initial Endorsement Standards
Computer Science Education programs meeting ISTE standards will prepare candidates to teach secondary computer science. It is anticipated that these standards can be implemented through a teacher preparation *endorsement* program (not a major in computer science education). Candidates completing this program will exhibit knowledge, skills, and dispositions equipping them to teach application usage, computer concepts, information technology fluency, and computer programming.

Who should complete the *ISTE Computer Science* program standards?
Secondary computer science teachers need to be prepared to meet the instructional needs of two audiences. As there are general math and general science courses for students who may not study those disciplines beyond high school, there will likely be general computer science courses that use a literacy or a fluency approach to computing. Such a course provides non-specialists a foundation for living and adapting in a technology-rich culture. The second audience includes those who will be served by various courses designed to introduce students more specifically to computer science. They are likely to involve, or build on, computer programming. Prospective computer science teachers must be prepared to address both kinds of courses.

The program standards and performances described in this document employ a broad definition of computer science education. Just as mathematics includes arithmetic, basic skills, and understandings necessary to effectively use mathematics, computer science is viewed as including the basic skills and understandings necessary for life-long learning relative to technology use. Also as in mathematics, computer science educators must also deal with the more advanced aspects of the discipline--e.g., programming and algorithm design; computer system components, connections, and operations; data and problem modeling; and social interactions, implications, and issues. Institutions with teacher preparation programs should provide their prospective computer science teachers with preparation in all these aspects of computing. Several types of experiences are likely to contribute to such preparation. The formal study of computer science is critical and it is anticipated that prospective teachers will complete preparation approximating a minor in computer science (at least 18 semester hours are recommended).

Typically, prospective teachers will have their primary preparation in a discipline other than computer science. That work should provide some general skills and practice that can be adapted to teaching computer science. Professional teaching preparation specific to computer science (a methods course) is also necessary to ensure that general teaching skills can be appropriately applied to computer science and to deal with discipline-specific practice.

Computer Science Standard I. (CS-I) Specialty Content Preparation in Computer Science.

Professional study in computer science education for secondary teachers provides experiences selected to develop a breadth and depth of knowledge of computer science. Courses and performances fulfilling these requirements must include experiences beyond the beginning level in computer science. It is anticipated that study approximately equivalent to a minor in computer science will provide the necessary specialty content in computer science (the equivalent of at least 18 semester hours of instruction is recommended).

CS-I.A. Programming and Algorithm Design

Much of computer science involves programming. Any comprehensive study of the discipline includes (and probably begins with) developing programming skill and an understanding of what happens when programs are executed. Basic skill in programming is essential; it is not, however, sufficient. Languages and tools change. It is important that students be prepared for change and that they understand that different approaches to programming exist and new ones will be developed.

CS-I.A.1. Laboratory-based Programming Experiences

Candidates will perform laboratory-based activities that demonstrate programming proficiency in a modern high-level programming language. (A two or three semester introductory sequence of courses is recommended. A connected, sequential approach to computer science during the initial study of the discipline is expected.)

a. demonstrate knowledge of and skill regarding the syntax and semantics of a high level programming language, its control structures, and its basic data representations
b. demonstrate knowledge of and skill regarding common data abstraction mechanisms (e.g., data types or classes such as stacks, trees, etc.)
c. demonstrate knowledge of and skill regarding program correctness issues and practices (e.g., testing program results, test data design, loop invariants)
d. design, implement, and test programs of sufficient complexity to demonstrate knowledge and skills included in CS-I.A.1.a - CS-I.A.1.c

CS-I.A.2. Multiple Paradigms.

Candidates will demonstrate an understanding of and flexibility with differing approaches/paradigms in programming (e.g., imperative, functional, object-oriented)

a. describe program design processes used in at least two different programming paradigms
b. design, implement, and test programs in languages from two different programming paradigms in a manner appropriate to each paradigm

CS-I.B. Computer Systems--Components, Organization, and Operation.

In-depth knowledge of how computer systems work individually and collectively is essential for adapting to new technological developments and understanding the implications of computers. It is also useful in programming. Prospective teachers will know how computer systems work.

1. effectively use a variety of computing environments (e.g., single- and multi-user systems and various operating systems)
2. describe the operation of a computer system--CPU & instruction cycle, peripherals, operating system, network components, and applications--indicating their purposes and interactions among them

CS-I.C. Data Representation and Information Organization.

Data is crucial to computing. Understanding it is necessary at a variety of levels--machine level representation (for program correctness); data structures (for program implementation); problem representation (for solution design); files and databases (for general applications); and interactions among systems and people (for overall system design and effectiveness). This kind of knowledge is also important for non-specialist understanding of the capabilities and limitations of computers.

1. describe how data is represented at the machine level (e.g., character, boolean, integer, floating point)
2. identify and provide usage examples of the various data structures and files provided by a programming language (e.g., objects, various collections, files)
3. describe the elements (people, hardware, software, etc.) and their interactions within information systems (database systems, the Web, etc.)

CS-I.D. Social Aspects of Computing.

We live within a cultural environment and interact daily with other people. Computing specialists need to communicate and work with each other and with non-specialists. Specialists and non-specialists alike need to be cognizant of issues and risks related to computing in our society and to learn independently as new developments in technology arise. Candidates will demonstrate skills and understanding relative to social aspects of computing that are appropriate for specialists and non-specialists.

CS-I.D.1. Societal Impact and Issues.

It is important that high school graduates be able to make informed decisions regarding computing in their personal lives and with respect to societal laws and norms. Doing so requires knowledge of computing and potential issues. It also requires skill at recognizing, researching, and analyzing issues to reach defensible conclusions.

a. demonstrate awareness of social issues related to the use of computers in society and principles for making informed decisions regarding them (e.g., security, privacy, intellectual property, limits of computing, rapid change)
b. analyze various social issues involving computing, producing defensible conclusions
c. demonstrate an understanding of significant historical events relative to computing

CS-I.D.2. Independent Learning and Communication

Computer science teachers should help students develop their ability to learn independently about computing and to communicate their newly developed understanding. Prospective teachers should be able to learn independently and to effectively communicate that learning.

a. conduct independent learning on specific, unfamiliar topics in general areas central to computer science
b. produce and present reports of substantial independent learning achieved in CS-I.D.2.a

CS-I.D.3. Collaborative Software Development.

Software development is generally a collaborative effort. Special attention to developing such skills in students is necessary. Prospective teachers should possess knowledge and experience in collaborative software development.

a. participate in substantive team software development projects that apply sound software engineering principles
b. describe behaviors and activities that enhance and detract from successful collaborative efforts

Computer Science Standard II. (CS-II). Professional Preparation.

Professional studies culminating in computer science education endorsements provide studies of and experiences in the methods, techniques, and strategies related to teaching computer science at the secondary level. (It is recommended that these experiences be equivalent in depth to at least the level achieved in three or more semester hours of instruction, i.e., a methods course. However, the specific number of hours recommended should not be construed as a requirement.) Teaching involves at least the activities of planning, delivering and managing, and assessing instruction. Prospective teachers should prepare to do each of these. They should also be prepared for the role of professional computer science educator.

CS-II.A. Planning Instruction.

Teaching is more than presenting information to students. As with programming, there are common approaches to common tasks. Teachers need to learn the tasks and approaches and be able to apply and evaluate them with respect to the students in their computer science classes.

1. Identify resources, strategies, activities, and manipulatives appropriate to teaching secondary computer science
2. Plan lessons/modules/courses related to each of:
    - programming process
    - knowledge/concepts
    - issue examination
3. Develop assessment strategies appropriate to lesson goals and the need to provide student feedback
4. Perform course and lesson planning that addresses student population characteristics (e.g., academic ability, cultural experience)

CS-II.B. Classroom and Field Experiences in Computer Science--Delivering Instruction

Teaching computer science is similar to teaching mathematics, social studies, language arts, etc. It is also different from each of them. Ideally, all candidates would experience practice-teaching in computer science. Realistically, that will not always be possible. Still, teacher preparation should strive to get as close to that ideal as is possible. Experiencing teaching-like activity and reflecting on the implications to full-fledged teaching of computing is a minimal requirement for preparation to teach computer science.

1. Observe and discuss the teaching of secondary computer science
2. Participate in the teaching of secondary computer science (lab assistant, tutoring, mini-teaching, etc.)
3. Plan and deliver a unit of instruction (see CS-II.A.4)

CS-II.C. Classroom & Course Management.

Managing students and instruction in computer science courses is very much like managing students in other disciplines. However, there are some differences. At least two differences are obvious. Sometimes it is useful or necessary to have students work at computers with all students doing essentially the same thing (without good planning, teachers answer the same question over and over while students wait their turn for teacher attention). Additionally, computer science homework often differs from that in other disciplines in its duration and need for specialized computer access. Prospective computer science teachers need to develop plans for dealing with these and other discipline specific management issues.

1. plan direct instruction involving simultaneous use of computing facilities by students (e.g., holding class in the lab, closed labs)
2. plan instruction involving students independently using computing facilities

CS-II.D. Instructional Assessment.

Reflection upon one's own performance as a teacher is essential for improving that performance. Thus, prospective teachers should be taught to examine and work to improve their teaching practice.

1. develop a personal plan for evaluating their own practice of teaching
2. make use of their plan for self-evaluation in the instructional delivery activities alluded to in CS-II.B.3

CS-II.E. Professional Development.

Computer science teachers often have no colleagues in the discipline in their school or even in their district. The discipline and pedagogical practice within it are constantly changing. Additionally, computer science offers secondary students an opportunity for a successful and productive career. Prospective teachers should prepare for dual professional roles in education and computer science.

1. discuss guidance roles and possible enrichment activities for secondary computer science students (e.g., computing career guidance, preparation for college, and extracurricular activities such as computer clubs and organized competitions)
2. plan for professional growth after identifying professional computer science and computer science education societies, organizations, groups, etc. that provide professional growth opportunities and resources