Computer Science at Seaver College

Philosophy of the curriculum

The Computer Science majors at Seaver College are joint majors with Mathematics and with Philosophy. The college also offers a computer science minor as well as the possibility of setting up a contract major that combines computer science with other academic disciplines. Historically, Seaver College has emphasized quality teaching at the undergraduate level while encouraging scholarly activity by its faculty, especially when it can have a positive impact on the undergraduate experience. The computer science curriculum is based on three themes – abstraction, integration, and languages and paradigms.

Abstraction

Abstraction is based on the concept of layers in which the details of one layer of abstraction are hidden from layers at a higher level. A computer scientist uses abstraction as a thinking tool to understand a system, to model a problem, and to master complexity. The ability to abstract cannot be acquired in a single course, but must be developed over several years. Consequently, all courses in the curriculum emphasize the abstraction process, not only as a framework to understand the discipline but also as a tool to solve problems.

Integration

The curriculum focuses on how well the courses are integrated as opposed to how many courses it has to offer. There are two aspects of integration in the curriculum – integration between courses and the integration of theory and practice. Both aspects of integration are important. Without integration between courses the curriculum becomes simply a collection of unrelated facts with no unity based on fundamental principles. The integration of theory and practice not only serves to re-enforce students' understanding of abstract concepts but also provides them with insight and appreciation of the practical solutions at hand.

Languages and paradigms

Because of the continued evolution of programming languages and paradigms we would do our students a disservice by emphasizing only one programming language or paradigm throughout the curriculum. Students should be multilingual and should experience multiple paradigms in their undergraduate careers. Our curriculum seeks to strike the proper balance between breadth and depth. Too much breadth will not equip students with the detailed skills necessary to solve realistic problems. Too much depth in one language or paradigm will give students a narrow vision that makes it difficult to consider multiple approaches to a problem.

Breadth is achieved by giving students a choice of programming languages and associated courses during the first semester. Depth is achieved by using a different single industrial-strength language during the second and third semesters. Courses in the following semesters introduce other programming paradigms based on different languages.

The language choices for the first two years are driven by both pedagogical and practical industry concerns. Pedagogical concerns are important during the first two years, because this is when students begin to form algorithmic thinking patterns and develop problem-solving skills. The criteria are that the programming environment should be simple to learn yet powerful enough to illustrate fundamental concepts of computing. Skill in a practical language is necessary for students to be well equipped for their post graduate careers. The languages for the third and fourth years are chosen for the variety of programming paradigms on which they are based.

The Computer Science / Mathematics major

The Computer Science / Mathematics major leading to a B.S. degree is satisfied by completing a core of 16 courses plus one elective. Students who intend to pursue graduate education in computer science should take both electives.

Intended outcomes

Upon successful completion of the program, the student should possess:

- A mathematical foundation that underpins all scientific endeavors and especially the discipline of computer science. This foundation is sufficient for graduate work in computer science but not in mathematics.
- A working knowledge of programming paradigms and software design principles, and programming languages that are used to implement them.
- Knowledge of fundamental structures in computer science such as computer architecture/organization, operating systems, and computer networks

The standard sequence

The courses are divided into a first year core, a second year core, and an upper division curriculum. Following is a list of the courses that are taken in the normal sequence. Courses that are offered alternate years should be taken when they are offered, even though the sequence will differ from what is listed below.

First year

Math 220, Formal Methods
CoSc 101, Programming Principles I with JavaScript
or CoSc 105, Programming Principles I with R
Math 150, Calculus I
Math 221, Discrete Structures
CoSc 121, Programming Principles II

Second year

Math 151, Calculus II
CoSc 320, Data Structures
Math 250, Calculus III
CoSc 330, Computer Systems
Phys 210, Physics I

Third year

Math 260, Linear Algebra
CoSc 450, Programming Paradigms
Math 365, Automata Theory (offered alternate years)
CoSc 465, Operating Systems (offered alternate years), elective *

Fourth year

Math elective (Math 316, Math 340, Math 345, or Math 350)
CoSc 475, Computer Networks
CoSc 490, Senior Capstone
CoSc 425, Computer Organization (offered alternate years), elective *

*Note: Only one computer science elective required.

Computer Science / Mathematics first year

Fall semester

Math 220, Formal Methods (3)
CoSc 101, Programming Principles I with JavaScript (3)
or
CoSc 105, Programming Principles I with R (3)

The courses in the first year of a computer science curriculum must serve the purpose of laying the foundation for the curriculum as a whole. Formal Methods is one such course. It is designed to teach a sound understanding of proof and a skill in formal manipulation. Its immediate application is to programming methodology. Students discover that to know how and why a program works, they need to view a program beyond its operational semantics. Understanding how and why a program works is equivalent to proving its correctness. Proof of correctness of a program requires the program itself to be defined as a mathematical entity. Formal Methods provides students with a methodology to set up a program as a mathematical object, build it, prove its correctness, and in the process understand it. It also teaches students to think rigorously, a skill that is valuable in all courses.

Students need a combination of operational and mathematical thinking to solve programming problems. For the Formal Methods course to be effective, the curriculum complements it with a course in programming where students learn the operational aspect of programs. This role is fulfilled by the Programming Principles I courses, either CoSc 101 with the JavaScript language, or CoSc 105 with the R language. Both languages are object-oriented and present data types, control structures, functions, and elementary data structures. Students use objects as opposed to designing objects with either of these languages.

Spring semester

Math 150, Calculus I (4)
Math 221, Discrete Structures (3)
CoSc 121, Programming Principles II (3)

Calculus is a fundamental mathematical tool for the sciences. The calculus course begins a three-semester sequence whose goal is proficiency in using this tool.

Discrete Structures presents the abstract mathematical view of fundamental data structures that form the building blocks of an algorithm. The course uses the methodology of its prerequisite course, Formal Methods, to perform complexity analysis and proof of correctness on the algorithms that manipulate data structures. It complements the concurrent Programming Principles courses, which teach the concrete implementation of data structures and algorithms in a specific programming language.

The purpose of Programming Principles II is to make the transition from the procedural paradigm to the object-oriented paradigm. The steps include recursion, dynamic storage allocation, inheritance, and polymorphism. Students learn abstraction by experiencing layers of abstraction, starting at the lower procedural level and progressing to the higher object-oriented level. The programming language is C++ regardless of the language taken in the Principles I course. Using different languages for Principles I and II furthers our goal of students being multilingual.

The content and purpose of Programming Principles II and Formal Methods are integrated. Such integration is designed to provide students with a concrete framework on which they can build and develop their problem solving skills. One area of integration is the connection between run-time analysis in the Programming Principles courses and asymptotic analysis in the Formal Methods/Discrete Structures sequence. Another is the connection between the assert function of C++ and pre- and post-conditions of Hoare triples from Formal Methods/Discrete Structures.

Computer Science / Mathematics second year

Fall semester

Math 151, Calculus II (4)
CoSc 320, Data Structures (4)

Data Structures continues the object-oriented paradigm from Computer Science II. The course presents data structures that are essential to programming, such as hash tables, stacks, queues, trees, and graphs, as systems of cooperating objects. This approach gives more depth to object-oriented design principles. The course continues to use an industry mainstream language, C++. Using the same language as that in Programming Principles II furthers our goal of students having an in-depth working knowledge of a single language.

Spring semester

Math 250, Calculus III (4)
CoSc 330, Computer Systems (3)
Phys 210, Physics I (5)

Computer Systems presents a unified picture of system architecture based on four layers of abstraction: high-order, assembly, operating system, and machine. Each layer has its own language: Java, assembly language, operating system calls, and machine language respectively. The course emphasizes the relationship between the layers by exploring the translation process using finite state machines. A software project to do a translation using the Java programming language enhances students' programming skill. The switch from C++ to Java in Computer Systems furthers our goal of students learning multiple languages in the curriculum.

Physics I introduces students to the most fundamental of the experimental sciences and to the scientific method. The laws of physics provide an understanding of how physical systems work and are the basis of computer modeling of the physical world.

Computer Science / Mathematics third year

Fall semester

Math 260, Linear Algebra (4)
CoSc 450, Programming Paradigms (4)

Linear algebra presents matrix algebra, vector spaces, and linear transformations. Its goal is to provide students with the skill to apply these mathematical tools to problems in computer science such as graphics, computer modeling, and numerical methods.

Programming Paradigms introduces three major programming models that complement the procedural and object-oriented approaches: the functional, declarative, and concurrent models. Each model is presented in the context of an associated programming language: Lisp, Prolog, and Java respectively. This course furthers the goal of providing experiences in multiple paradigms and languages in the undergraduate curriculum.

Spring semester

Math 365, Automata Theory (3)
CoSc 465, Operating Systems (3)

The goal of Automata Theory is to answer the question, What is computability, and what are its limits? Students learn the abstract models that help to answer these questions – finite automata, pushdown automata, and Turing Machines. The course prepares students for more advanced work in theoretical computer science and serves as a basis for applied work in language design and compiler construction.

Operating Systems is a continuation in more depth of the topic introduced in the Computer Systems course. Operating systems are presented as multi-layer components that hide the details of hardware implementation in order to master the complexity of resource management. The course furthers the theme of abstraction in the curriculum and serves as an example of the power of abstraction in software design.

Computer Science / Mathematics fourth year

Fall semester

Math elective (Math 316, Math 340, Math 345, or Math 350)
CoSc 475, Computer Networks (4)

The math elective includes one course from Math 316 Biostatistics (3), Math 340 Differential Equations (4), Math 345 Numerical Methods (4), or Math 350 Mathematical Probability (4). Each course provides the student with the mathematical tools to apply computation to problem domains outside of computer science.

Computer Networks presents data communication and networking principles based on the seven-layer OSI reference model. It provides students with software and hardware tools to simulate and test various networking environments. It includes the use of mathematical tools from the probability course to construct queueing models for performance analysis of network protocols, furthering the goal of integration between courses in the curriculum. That Computer Networks is a required course in the curriculum is a reflection of the growing importance of networks in our society as exemplified by the ubiquity of the Internet.

Spring semester

CoSc 490, Senior Capstone (4)
CoSc 425, Computer Organization (3)
CoSc 495, Internship in Computing (1 to 4)

The Senior Capstone course centers on a large team-based software project that requires students to integrate their knowledge and experience from previous courses. One component of the course will be content-based whose topic will vary from year to year. Its purpose is to give students an experience that represents what they are likely to encounter in the computer industry.

Computer Organization is a continuation in more depth of the topic introduced in the Computer Systems course. The hardware organization of computers is presented in multiple layers of abstraction and includes the logic gate layer, the microprogramming layer, and the machine layer. The course is yet another example of the pervasive theme of abstraction throughout the curriculum.

Internship in Computing provides students with a practicum in industry. It serves as a bridge between the academic world and the professional world.

Computer Science / Mathematics Major
Course requirements

Core courses (all required)

_____ CoSc 101 or 105, Programming Principles I (3)
_____ CoSc 121, Programming Principles II (3)
_____ CoSc 320, Data Structures (4)
_____ CoSc 330, Computer Systems (3)
_____ CoSc 450, Programming Paradigms (4)
_____ CoSc 475, Computer Networks (4)
_____ CoSc 490, Senior Capstone (4)
_____ Phys 210, Physics I (5)
_____ Math 220, Formal Methods (3)
_____ Math 221, Discrete Structures (3)
_____ Math 150, Calculus I (4)
_____ Math 151, Calculus II (4)
_____ Math 250, Calculus III (4)
_____ Math 260, Linear Algebra (4)
_____ Math 365, Automata Theory (3)
_____ Math elective (Math 316, 340, 345, or 350)

Elective (one required)

_____ CoSc 425, Computer Organization (3)
_____ CoSc 465, Operating Systems (3)

General education requirements

General Studies

_____ GS 199, First-Year Seminar (3)

English and Literature

_____ Eng 101, English Composition (3)
_____ Literature Choice (4)

Christianity and Culture

_____ Rel 101, Old Testament in Context (3)
_____ Rel 102, New Testament in Context (3)
_____ Rel 301, Christianity and Culture (3)

Western and Non-Western Heritage

_____ Hum 111, Western Heritage I (3)
_____ Hum 212, Western Heritage II (3)
_____ Hum 313, Western Heritage III (3)
_____ Non-Western Heritage choice (4)

Fine Arts

_____ Fine Arts Choice (2)

Communication

_____ 251-level foreign language (4)
_____ Spe 180, Public Speaking (4)

Human Institutions and Behavior (two required)

_____ Econ 200, Economic Principles (4)
_____ Psyc 200, Introduction to Psychology (3)
_____ Soc 200, Introduction to Sociology (3)

American Experience

_____ PoSc 104, The American People and Politics (4)
_____ Hist 304, History of the American Peoples (4)

Notes

Math 220 and Phys 210 meet the General Education mathematics and science requirements.
A total of 128 units, including 40 upper-division units are required to graduate.

The Computer Science / Philosophy major

The Computer Science / Philosophy major leading to a B.A. degree is satisfied by completing a core of 14 courses plus one elective.

Intended outcomes

Upon successful completion of the program, the student should possess:

- A working knowledge of logic with application to philosophical argumentation and to program analysis.
- A working knowledge of programming paradigms and software design principles, and programming languages that are used to implement them.
- Knowledge of fundamental structures in computer science such as computer architecture, artificial languages, and computational models.
- An understanding of key issues in contemporary philosophy and the history of philosophy—including theories of mind, language, and knowledge—and the ability to apply that understanding to real-world problems.

The standard sequence

The courses are divided into a first year core, a second year core, and an upper division curriculum. Following is a list of the courses that are taken in the normal sequence. Course scheduling may vary from the following sequence. For example, courses that are offered alternate years should be taken when they are offered. A student's major advisor may be from philosophy or computer science, but academic advice is available from both disciplines.

First year

Math 220, Formal Methods
CoSc 101, Programming Principles I with JavaScript
or CoSc 105, Programming Principles I with R
Math 221, Discrete Structures
CoSc 121, Programming Principles II
Phil 200, Introduction to Philosophy

Second year

CoSc 320, Data Structures
CoSc 330, Computer Systems
Phil 290, Logic
One upper-division Philosophy elective.*

Third year

CoSc 450, Programming Paradigms
Math 365, Automata Theory (offered alternate years)
Phil 300, Ancient Philosophy
Phil 310, Modern Philosophy

Fourth year

Phil 420, Epistemology
Phil 480, Major Philosophical Problems Seminar

*Note: May be taken second, third, or fourth year.

Computer Science / Philosophy Major
Course requirements

Core courses (all required)

_____ CoSc 101 or 105, Programming Principles I (3)
_____ CoSc 121, Programming Principles II (3)
_____ CoSc 320, Data Structures (4)
_____ CoSc 330, Computer Systems (3)
_____ CoSc 450, Programming Paradigms (4)
_____ Math 220, Formal Methods (3)
_____ Math 221, Discrete Structures (3)
_____ Math 365, Automata Theory (3)
_____ Phil 200, Introduction to Philosophy (4)
_____ Phil 290, Logic (4)
_____ Phil 300, Ancient Philosophy (4)
_____ Phil 310, Modern Philosophy (4)
_____ Phil 420, Epistemology (4)
_____ Phil 480, Major Philosophical Problems Seminar (4)

Elective (one required)

_____ Upper-division (300- 400-level) Philosophy course (4)

General education requirements

General Studies

_____ GS 199, First-Year Seminar (3)

English and Literature

_____ Eng 101, English Composition (3)
_____ Literature Choice (4)

Christianity and Culture

_____ Rel 101, Old Testament in Context (3)
_____ Rel 102, New Testament in Context (3)
_____ Rel 301, Christianity and Culture (3)

Western and Non-Western Heritage

_____ Hum 111, Western Heritage I (3)
_____ Hum 212, Western Heritage II (3)
_____ Hum 313, Western Heritage III (3)
_____ Non-Western Heritage choice (4)

Fine Arts

_____ Fine Arts Choice (2)

Communication

_____ 251-level foreign language (4)
_____ Spe 180, Public Speaking (4)

Human Institutions and Behavior (two required)

_____ Econ 200, Economic Principles (4)
_____ Psyc 200, Introduction to Psychology (3)
_____ Soc 200, Introduction to Sociology (3)

American Experience

_____ PoSc 104, The American People and Politics (4)
_____ Hist 304, History of the American Peoples (4)

Natural Science

_____ Laboratory Science choice (4)

Notes

Math 220 meets the General Education mathematics requirement.
A total of 128 units, including 40 upper-division units are required to graduate.

The Computer Science minor

The Computer Science minor is satisfied by completing a core of five courses plus one elective.

Minor core

Math 220, Formal Methods
CoSc 101, Programming Principles I with JavaScript
or CoSc 105, Programming Principles I with R
Math 221, Discrete Structures
CoSc 121, Programming Principles II
CoSc 320, Data Structures

Minor elective, choose one

CoSc 330, Computer Systems
Math 365, Automata Theory
CoSc 450, Programming Paradigms

The Digital Humanities minor

The minor prepares students for graduate study in art history, history, literature, religion, or library sciences as well as any of the fields which serve public humanities, including cultural art centers and museums. The minor emphasizes the digital humanities' values of open access, public accountability, collaboration, interdisciplinary cooperation, and non-hierarchical organizational structures.

A total of 18 units are required for the digital humanities minor.

Minor core, 6 units

Eng 225, Introduction to Digital Humanities
CoSc 101, Programming Principles I with JavaScript
or CoSc 105, Programming Principles I with R

Minor elective, 12 units

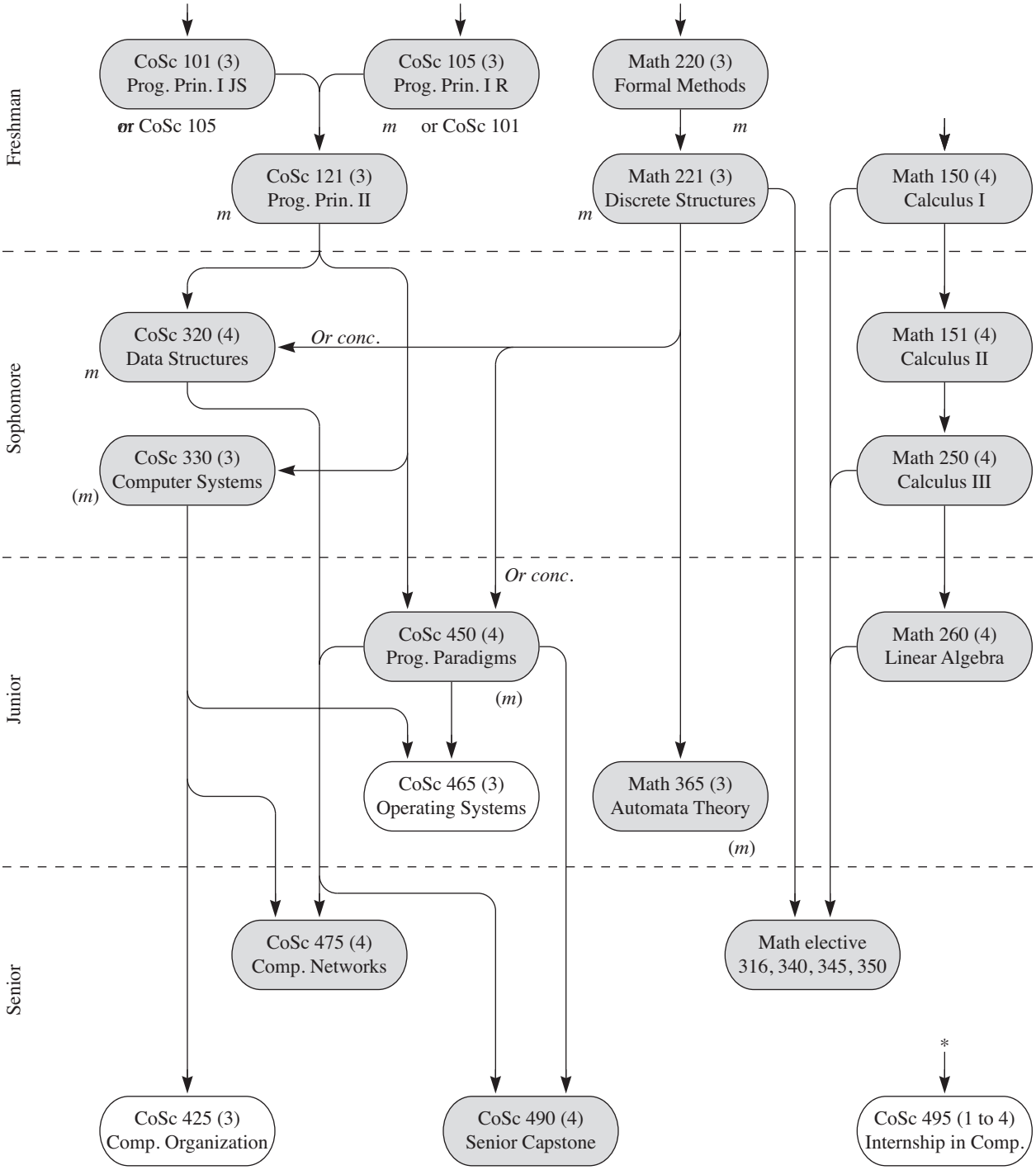
CoSc 121, Programming Principles II
CoSc 330, Computer Systems
DH 292, Selected Topics
DH 299, Directed Studies
DH 492, Selected Topics
DH 495, Digital Humanities Internship
DH 499, Directed Studies
Eng 310, Library and Archival Sources in a Digital Environment
Eng 460, Principles of Writing with Technology

Or another course approved by the HUTE divisional dean with consent of the instructor.

Computer science students who minor in Digital Humanities may not include CoSc 330 as one of their minor elective courses.

Prerequisites

The following diagram shows the prerequisite structure of the curriculum. Semester hours are in parentheses. Required courses are shaded with one additional computer science elective plus physics required for the major. The letter *m* indicates courses required for a computer science minor. One additional course must be chosen from the set of courses labeled (*m*).



* Note: Prerequisite for Internship in Computing is 90 units completed with a minimum 3.0 GPA and committee approval.