Computer Science Implementation Framework

Grade 9-12

2026

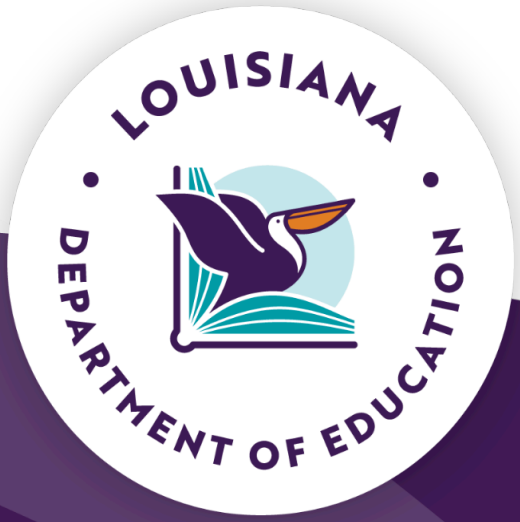

Table of Contents

Background

Louisiana Act 541 (2022), the Computer Science Education Act, aimed to establish a comprehensive and seamless statewide computer science education program across all educational levels to meet the demands of a dynamic economy. To achieve this goal, Act 541 created the Computer Science Education Advisory Commission (CSEAC). CSEAC was responsible for advising the State Board of Elementary and Secondary Education (BESE) through the Louisiana Department of Education (LDOE) to develop and implement a state action plan for delivering education in computer science in all public schools.

The first key action within the Louisiana K-12 Computer Science Education Plan calls for establishing content standards for computer science. From May to August 2024, the K-12 Computer Standards Writing Committee drafted student-centered and developmentally appropriate computer science standards within defined grade bands: elementary (K-5), middle school (6-8), and high school (9-12). These standards were carefully designed to reflect the increasing complexity and depth of understanding expected as students progress through each educational stage, with a focus on analytical and critical thinking skills, digital literacy, digital responsibility, and technology skills. In October 2024, BESE approved the K-12 Louisiana Student Standards for Computer Science. These standards now serve as a guiding framework to support high-quality computer science instruction in Louisiana.

To support standards implementation, the Louisiana K-12 Computer Science Education Plan also calls for the development of computer science frameworks for standards implementation, which show the overlap with existing content standards.

Purpose of the Frameworks

The frameworks support Key Action 1 of the Louisiana K-12 Computer Science Education Plan by providing guidance for integrating the K-12 Louisiana Student Standards for Computer Science into existing or new high school coursework. This framework also supports Key Actions 2 and 3 by intentionally focusing on connections to Industry-Based Credentials (IBCs) within coursework and on a coordinated progression of learning through Jumpstart Pathways.

Connections between Computer Science, Mathematics, and Science

A key feature of the Computer Science Implementation Frameworks is highlighting the connections between the K-12 Louisiana Student Standards for Computer Science and mathematics and science instruction. High school courses designed to meet the computer science graduation requirements provide foundational knowledge of core computer science concepts while emphasizing mathematical, scientific, and engineering practices. Computer science concepts, including sequencing, patterns, and logic, are aligned with mathematical reasoning and scientific inquiry. Computational thinking strategies, such as decomposition and algorithmic thinking, support problem-solving approaches in math and science.

Finding Connections to Core Content Areas

In high school, computer science implementation emphasizes a deeper understanding of computer science concepts and skills, while also reinforcing how computer science can be applied within and across content areas. Computer science standards provide in-depth knowledge of core concepts applicable to a wide range of careers and real-world applications.

Unpacking the Standards

Computer science standards are essential for effective computer science implementation. When considering which computer science standards best fit a lesson, summarize the standard's ultimate purpose or goals by asking, “What should students accomplish by the end of grade 12 to master this standard?” Identifying the underlying skill or content required to master the standard helps in determining its best fit within a specific content area.

Finding Connections

Connecting standards to the real world makes standards more meaningful and relevant to students. Making connections between computer science standards and content or skills in other content standards demonstrates the real-world applications of computer science. Connections may be found between computer science core practices and practices of another content area. Computer science skills also reinforce skills emphasized in other content areas. For a full explanation of the core concepts and core practices of the computer science standards, see the K-12 Louisiana Student Standards for Computer Science.

Computational Thinking Skills

Computational thinking involves breaking down complex problems into smaller, manageable parts (decomposition), recognizing patterns (pattern recognition), creating step-by-step solutions (algorithms), and generalizing solutions to other problems (abstraction). These skills are not exclusive to computer science; they are applicable in other content areas, including math and science. By understanding computational thinking, teachers can identify opportunities to apply these problem-solving strategies in multiple contexts.

Computational Thinking Practice	In Mathematics	In Science
Abstraction	Represent concepts using visualizations such as graphs, charts, or diagrams.	Create simplified simulations of complex scientific phenomena.
Algorithmic Thinking	Create step-by-step instructions for solving problems or geometric constructions.	Create models of scientific processes like the water cycle or food chain.
Decomposition	Break down multi-step word problems into smaller, manageable parts.	Break down complex investigations into smaller, testable hypotheses.
Pattern Recognition	Identify number patterns, geometric patterns, or patterns in data sets.	Identify patterns in data, such as weather, animal behavior, or plant growth.

Core Concept 1: Computing Systems (CS)

Overview

Students interact with a wide variety of computers each day. Computers are devices that can collect, store, analyze, and act upon information in ways that can positively and negatively affect human capabilities. Humans (users) operate, program, and maintain the physical components (or hardware) and instructions (or software) that make up a computer. The hardware, software, and users are collectively called computing systems. Users leverage their understanding of hardware and software to resolve problems within computing systems in a process called troubleshooting.

High School Standards for Core Concept 1: Computing Systems

Subconcepts and Core Practices	High School (9-12)
Subconcept 1: **Hardware and Software** *Core Practices:* *Collaborating Around Computing;* *Recognizing and Defining Computational Problems;* *Communicating about Computing*	H.CS.1A. Analyze the levels of interactions between application software and system software as well as the hardware layers.
	H.CS.1B. Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday things.
Subconcept 2: **Troubleshooting** *Core Practices:* *Collaborating Around Computing;* *Recognizing and Defining Computational Problems;* *Testing and Refining Computation Artifacts;* *Communicating about Computing*	H.CS.2A. Generate guidelines that convey systematic troubleshooting strategies that other users can utilize to identify and fix errors.

Core Concept 2: Networks and the Internet (NI)

Overview

Computing systems typically do not operate in isolation. Networks connect computers to share information and resources and are integral to computer and data science. Networks enable critical communication for the computing systems that drive our economy and career sectors. The increased level of connectivity brought about by the internet provides fast and secure communication that facilitates innovation.

High School Standards for Core Concept 2: Networks and the Internet

Subconcepts and Core Practices	High School (9-12)
Subconcept 1: Hardware and Network Communication *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.NI.1A. Evaluate a network's scalability, reliability, and appropriateness by describing the relationship between routers, switches, devices, topology, and addressing (MAC, IP, Subnet, Gateway).
	H.NI.1B. Illustrate how to trace data through a network model, explaining the interactions that occur throughout the process.
	H.NI.1C. Describe and evaluate the internet as a digital public infrastructure (DPI) from the highest level to the private service provider level.
Subconcept 2: Cybersecurity *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.NI.2A. Recommend security measures to address factors that create trade-offs between the usability and security of a computing system.
	H.NI.2B. Interpret and analyze mechanisms through which malware and other types of cyber attacks can impact hardware, software, and sensitive data.
	H.NI.2C. Compare and contrast how software developers protect computing systems and information from unauthorized user access.

Core Concept 3: Data and Analysis (DA)

Overview

Computing systems function by processing and storing data. Data is abundant due to the growing number of connected devices worldwide. As the volume of data expands, so does the demand for accurate and efficient data analysis methods. Data science is the cross-disciplinary use of data to inform decision-making, test hypotheses, predict trends, and develop precise models that drive innovation across industries.

High School Standards for Core Concept 3: Data and Analysis

Subconcepts and Core Practices	High School (9-12)
Subconcept 1: Data Representation *Core Practices:* *Collaborating around computing;* *Recognizing and defining computational problems;* *Creating computational artifacts*	H.DA.1A. Evaluate data representations, propose strategies to reconstruct the data, and visualize data in a variety of ways.
	H.DA.1B. Define and describe database structures to optimize the search and retrieval of data.
Subconcept 2: Data Collection *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.DA.2A. Explain and describe the impacts of uncertainty and the limitations of data collection technology and tools.
	H.DA.2B. Describe the personal and legal impacts of accumulated data, both collected and derived, for given scenarios. Propose tools and techniques to manage the accumulated data appropriately.
Subconcept 3: Data Storage *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.DA.3A. Justify choices on how data elements are organized and where data is stored, considering cost, speed, reliability, accessibility, privacy, and integrity.
	H.DA.3B. Explain and utilize the appropriate data structural organization system to collaborate and communicate data within a team or user group in given scenarios.

Subconcept 4: Visualizations and Transformations *Core Practices:* *Creating computational artifacts;* *Communicating about computing*	H.DA.4A. Create interactive data visualizations using software tools that explain complex data to others.
	H.DA.4B. Utilize data analysis tools to ingest (extract, transform, and load) and process data into relevant information.
Subconcept 5: Inference and Models *Core Practices:* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.DA.5A. Create a model utilizing data with the appropriate simulated variables to develop predictions for real-world phenomena
	H.DA.5B. Apply and evaluate data analysis techniques to identify patterns represented in complex systems.
	H.DA.5C. Analyze patterns in data visualizations, then select a collection tool to test a hypothesis and communicate the relevant information to others.
	H.DA.5D. Evaluate the impacts of the variables and the model on the performance of a simulation to refine a hypothesis.

Core Concept 4: Algorithms and Programming (AP)

Overview

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing systems. Algorithms and programs control all computing systems, empowering people to communicate with the world in novel ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use, how to process the data, and how to store the information. The decomposition of more significant problems into simpler ones, combined with recombining existing solutions and analyzing various solutions, helps determine the most appropriate solution to a problem.

High School Standards for Core Concept 4: Algorithms and Programming

Subconcepts and Core Practices	High School (9-12)
Subconcept 1: Variables and Algorithms *Core Practices:* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.AP.1A. Assess variables, then classify the scope and type of variable.
	H.AP.1B. Design algorithms that can be adapted to express an idea or solve a problem.
	H.AP.1C. Use and adapt classical algorithms to solve computational problems.
	H.AP.1D. Explain what computer memory is and how variables are stored and retrieved.
	H.AP.1E. Identify and explain how a derived data type can be utilized in a real-world scenario.
Subconcept 2: Control Structures *Core Practices:* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.AP.2A. Justify the selection of control structures to balance implementation complexity, maintainability, and program performance.
	H.AP.2B. Design and iteratively develop computational artifacts using events to initiate instructions.

Subconcept 3: Modularity *Core Practices:* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.AP.3A. Decompose problems into smaller components using constructs such as procedures, modules, and/or objects.
	H.AP.3B. Create computational artifacts using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
Subconcept 4: Program Development *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Recognizing and defining computational problems;* *Developing and using abstractions;* *Creating computational artifacts;* *Testing and refining computational artifacts;* *Communicating about computing*	H.AP.4A. Utilize the Software Development Life Cycle (SDLC) to create software that is a minimum viable product.
	H.AP.4B. Develop and utilize test cases to verify that a program performs according to the program's design specifications.
	H.AP.4C. Design and develop programs by working in team roles using version control systems, integrated development environments (IDEs), and collaborative tools and practices.
	H.AP.4D. Evaluate licenses that limit or restrict the use of computational artifacts when utilizing resources such as libraries.
	H.AP.4E. Apply the appropriate documentation techniques to make programs more accessible to debug and to be maintained by others.
	H.AP.4F. Iteratively evaluate and modify an existing program to add functionality and discuss intended and unintended implications.
	H.AP.4G. Use a standard library and/or application programming interface (API) to create reusable code components to design simple programs and enhance existing programs.

Core Concept 5: Impacts of Computing (IC)

Overview

The impacts of computing can be positive and negative, enabling innovation, communication, and access to information, while also raising concerns around ethics, privacy, and fairness. Individuals and communities not only shape computing systems through interactions, behaviors, industry practices, and laws, but are also shaped by interactions with computing systems. Computer and data science create new means of communication by accelerating information exchange and establishing cyberspace as a dynamic workspace for individuals.

High School Standards for Core Concept 5: Impacts of Computing

Subconcepts and Core Practices	High School (9-12)
Subconcept 1: Intellectual Achievements *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Testing and refining computational artifacts;* *Communicating about computing*	H.IC.1A. Analyze the key milestones of computer science, historical events influenced by computer science, and the people connected to these achievements.
	H.IC.1B. Explain how innovations in computer science and technology enable advancements in other fields of study.
Subconcept 2: Social Interaction *Core Practices:* *Fostering cyber responsibility; Collaborating around computing;* *Testing and refining computational artifacts;* *Communicating about computing*	H.IC.2A. Evaluate the adoption and adaptation of social norms from the physical world to the cyber world.
	H.IC.2B. Describe how cyberspace is becoming a universal medium for connecting humans, the economy, business, and computing.
	H.IC.2C. Describe and critique how algorithmic feedback loops can shape perceptions, reinforce a limited data set, and limit the sources of information that may inform the individual user.
Subconcept 2: Laws, Safety, and Industry Practices *Core Practices:* *Fostering cyber responsibility;* *Collaborating around computing;* *Testing and refining computational artifacts;* *Communicating about computing*	H.IC.3A. Debate laws and industry regulations that impact the development and use of computational artifacts.
	H.IC.3B. Describe and analyze the motives of online threat actors to a user's personal safety, privacy, and well-being.
	H.IC.3C. Compare and contrast the varied approaches to govern data, intellectual property, control information access, and various ways for users to be aware of guidance.

	H.IC.3D. Explain how the interconnectedness of cyberspace can lead to physical and digital vulnerabilities.
	H.IC.3E. Debate the ethical considerations of creating and publishing computational artifacts.
	H.IC.3F. Analyze the data provenance of computational artifacts.
	H.IC.3G. Explain how individuals and organizations can exert influence on personal and societal perceptions and practices through computing technologies.

Considerations for Computer Science Implementation

Implementing computer science in grades 9-12 requires strategic planning to guarantee long-term success. This approach ensures a learning progression that aligns high school expectations and provides access for all students. When determining the best implementation model to fit the needs of schools and students, systems should consider the following factors:

- Cost and Funding
    - What is the total cost for teachers to prepare for and deliver quality computer science experiences? Costs may include:
        - Professional learning or development in a curriculum or towards certification
            - Teacher access licenses
            - Training registration costs
            - Teacher expenses and stipends
    - What is the total cost of materials needed for the number of students participating in the quality computer science experiences? Materials may include:
        - Hands-on student materials
            - Support items required to connect with digital programs (e.g., a calculator or micro:bits, using some form of hands-on device that connects with the program)
            - Consumables and non-consumables
        - Digital devices or other technology
            - Student access codes
            - Platform usage
    - How much money is available to purchase high-quality materials for both students and the teacher? Is there a plan to maintain and expand the program after the purchase of initial materials is complete? Funding sources may include:
        - Federal or grant funding
        - Classroom/school donations
        - Partnerships with local industry and the community
- Data Privacy
    - Does the program meet federal/state/district laws to protect student privacy? Applicable laws can include:

        - FERPA (Family Educational Rights and Privacy Act)
        - COPPA (Children's Online Privacy Protection Act)
        - ACT 837
        - Parental Agreements/Consents
    - What is necessary to ensure all work with vendors complies with state and system policies? This may include:
        - Vendor Agreements or data protection contracts
        - LDOE Data Sharing Agreements
    - Does the vendor provide additional enhanced data security measures? Data security measures may include:
        - MultiFactor Authentications (MFA)
        - Breach response plans
- Additional Resource Considerations
    - What resources are available to set up these quality computer science experiences for students?
        - Human resources - computer science teacher, teacher/staff members, parent volunteers
        - Local resources - industry and business partners through Career and Technical Education (CTE)/Career Opportunities/District Technology Centers
        - Teachers must ensure that local school system networks permit access to the resources and create website-required teacher accounts.

Computer Science Courses

Courses that meet the graduation requirement or equivalent

Many computer science concepts, such as sequencing, patterns, and logic, align with mathematical reasoning and scientific inquiry. Computational thinking strategies, such as decomposition and algorithmic thinking, support problem-solving approaches in math and science. Integrating computer science within mathematics and science helps students see connections between the content areas, encouraging students to view technology as a tool for solving real-world problems in a variety of professional fields.

Mathematics

The following courses meet the computer science graduation requirement and offer foundational knowledge of computer science concepts, with an emphasis on mathematical practices and concepts.

- (061175) AP Computer Science A
    - AP Computer Science A introduces students to computer science through programming using Java. This course includes the design of solutions to problems, the use of data structures to organize large sets of data, the development and implementation of algorithms to process data and discover new information, the analysis of potential solutions, and the implications of computing systems.
- (165031) Statistical Reasoning
    - Statistical Reasoning courses introduce the study of likely events and the analysis, interpretation, and presentation of quantitative data. Course topics generally include basic probability and statistics, such as discrete probability theory, odds and probabilities, probability trees, populations and samples, frequency tables, measures of central tendency, and data presentation (including graphs). Course topics may also include normal distribution and measures of variability.
- (061141) Introduction to Computational Thinking
    - Introduction to Computational Thinking courses introduce students to computational thinking and its applications in STEM. Courses utilize the core components of computational thinking, including abstraction, decomposition, and algorithmic thinking, helping students model problems and visualize mathematical concepts concretely. Courses also emphasize mathematical approaches to contemporary problems, handling of data, and data optimization using basic concepts from algebra, geometry, and discrete mathematics.
- (061810) Data Science
    - Data science courses combine mathematical, quantitative, and computational concepts and strategies to empower students to live skillfully and productively in our data-driven society. Course topics present students with a more realistic picture of how these concepts and strategies are used to address relevant, real-world problems. The framework includes course design principles and student learning outcomes, which, when applied together, can help students see how mathematics can provide the opportunity for success in further education and the workforce.

- (061820) Quantitative Reasoning
    - Quantitative reasoning courses support students' use of mathematical practices and, when faced with unfamiliar contexts, provide them with strategies to solve these new problems. Courses use collaborative learning techniques to help students recognize the need for data-driven decision-making and bring to light the dangers inherent in basing decisions solely on anecdotal evidence. Mathematical concepts in Quantitative Reasoning include numeracy, mathematical modeling, basic statistical reasoning, and analysis of complex information using graphical strategies.

Science

The following courses meet the computer science graduation requirement and offer foundational knowledge of computer science concepts, with an emphasis on the science and engineering practices.

- (061177) AP Computer Science Principles
    - AP Computer Science Principles is an introductory college-level computing course that introduces students to the breadth of the field of computer science. Students learn to design and evaluate solutions and apply computer science to solve problems through the development of algorithms and programs. They incorporate abstraction into programs and use data to discover new knowledge. Students also explain how computing innovations and computing systems — including the internet — work, explore their potential impacts, and contribute to a computing culture that is collaborative and ethical.
- (061102) Computer Science I
    - Computer Science I presents students with the conceptual foundations of computer science through an exploration of human-computer interaction, web design, computer programming, data modeling, and robotics. While this course includes programming, the focus is on the computational practices associated with computer science, rather than just a narrow focus on coding, syntax, or tools. Computer Science 1 courses teach students the computational practices of algorithm design, problem-solving, and programming within a context that is relevant to their lives.
- (061103) Computer Science II
    - Computer Science II presents a deeper understanding of computer science with greater emphasis on programming and application development. This course builds on the foundation of Computer Science I and provides students with the knowledge and skills necessary to construct computer programs in one or more languages, including object-oriented languages. Courses should include the development of applications for multiple platforms or operating systems. An emphasis is placed on user-centered design and current industry practices.
- (080530) Data Manipulation and Analysis
    - Data Manipulation and Analysis courses introduce students to the emerging field of Data Science. Topics include the standard practices for effective data manipulation, analysis, and interpretation, as well as necessary concepts in the four disciplines involved (data science, computing, mathematics, and statistics). Students engage with a variety of tools, such as spreadsheets, programming environments, and databases, focusing on the application of the concepts within a real-world context rather than theory. Students have opportunities to

work with large datasets and create models.

- (060200) Modeling and Simulation
    - Modeling and Simulation courses focus on the study of the forces and laws of nature and their application to modern technology. Equilibrium, motion, momentum, energy conversion, electromagnetism, and optical phenomena are presented in the context of current, real-world applications. Demonstrations, math labs, and applied laboratory experiments are an integral part of the classroom experience. These courses enable students to gain a solid foundation for careers in electronics, robotics, telecommunications, and other technological fields.

Foreign Language

Act 502 of the 2022 Regular Legislative Session allows certain computer science courses to satisfy the foreign language credit requirement for students who graduate with a Taylor Opportunity Program for Students (TOPS) University diploma. These courses must be taken in sequence and aligned to a coding language.

- Option 1
    - (061175) AP Computer Science A and (061102) Computer Science I
- Option 2
    - (1221300) Computer Coding as a Foreign Language I and (121301) Computer Coding as a Foreign Language II