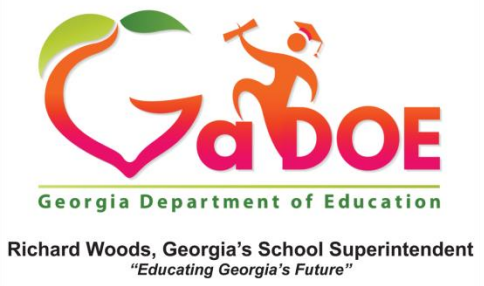
Georgia
Standards of Excellence

Computer Science

Standards

Middle School Computer Science I



Georgia Standards of Excellence for K-8 Computer Science

Georgia Standards of Excellence (GSE) for Computer Science (CS) were created in response to the growing ubiquity of computing devices and their impact on every aspect of society. If Georgia's students are to participate effectively in society, a shift in K-12 education must correspond. In Georgia, Computer Science is understood as the study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society. The standards blend the core concepts of computer science (i.e., what students should know) and computer science practices (i.e., what students should do). These core concepts and practices should be taught in an integrated way to provide authentic learning experiences for students.

The GSE for Computer Science immerse students in the practices of Computer Science from Kindergarten through grade 12, effectively transitioning Computer Science from a high school elective to a comprehensive K-12 discipline for all students. Some skills or concepts are emphasized more in particular grade bands in conjunction with research on how students learn and other knowledge and skills taught at those levels. Any curriculum aligned to these GSE should revisit domains and concepts over time as students apply their learning by creating computational artifacts. Creating computational artifacts can be as simple as writing socially responsible electronic messages (e.g., email and social media posts) and as complex as designing an app for a drone or a self-driving vehicle.

The standards are organized in grade bands rather than grade levels to afford schools flexibility in presenting the content while maintaining a structured, developmental progression from one band to another. Teachers can scaffold instruction from simple familiarization in the K-2 grade band to deeper involvement in the 3-5 and more thorough treatment in the 6-8 grade band. In addition, the 6-8 grade band standards are designed to feed directly into the high school CS pathways which are, in turn, designed to meet the dynamic needs of industry and post-secondary study of computer science.

Georgia-owned and Georgia-grown, the GSE for Computer Science relate broadly to national and international frameworks. The grade bands follow the structure set forth by the K12 CS Framework; they develop a comprehensive conceptual framework that grows over the years. The K-8 GSE for Computer Science also correspond to the ISTE standards for students as organizational domains. These domains are intended to be cross-curricular. The ISTE domains (e.g. Empowered Learner) define a high-level perspective on the characteristics of a 21st century student. These characteristics are couched in a digital society but are not restricted to computer science content. Likewise, the GSE for Computer Science can be integrated into other content areas and support enduring characteristics for learning (e.g., collaborative, communicative, creative, and critical thinking). Ultimately, the GSE for Computer Science support and inspire Georgia's students as they grow and learn, empowering students to be successful, responsible, and engaged citizens.

Georgia Standards of Excellence for K-8 Computer Science

The Standards are written in the following format:

CSS = Computer Science Standard
EL = Empowered Learner (Domain)
6-8 = Grade band 6 through 8
1 = is the standard number
1… = Element of the standard

Cluster 6-8
Empowered Learner
CSS.EL.6-8.1
Use technology resources to increase self-direction and self-regulation in learning, including for problem solving and collaboration (e.g., using the Internet to access online resources, edit documents collaboratively)

1. Understand the difference between editing a shared document and suggesting edits (e.g. track changes)
2. Use digital tools or platforms to organize, display, annotate, and/or share a curated collection
3. Complete an individual project (e.g., research or design) using technology resources

Georgia Standards of Excellence for K-8 Computer Science

Table of Contents

Empowered Learner
CSS.EL.6-8.1
Use technology resources to increase self-direction and self-regulation in learning, including for problem solving and collaboration (e.g., using the Internet to access online resources, edit documents collaboratively).

1. Understand the difference between editing a shared document and suggesting edits (e.g. track changes).
2. Use digital tools or platforms to organize, display, annotate, and/or share a curated collection.
3. Complete an individual project (e.g., research or design) using technology resources.

Digital Citizen
CSS.DC.6-8.2
Understand benefits and risks of digital citizenship and practices safe, responsible, legal, and ethical behavior while using technology tools and resources, especially related to personal information.

1. Understand that digital content is permanent and cannot be deleted.
2. Discuss the benefits and risks of using password management systems and storing personal information online.
3. Understand how browser settings such as cookies collect data and track personal information.

CSS.DC.6-8.4
Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

CSS.DC.6-8.5
Understand the pervasiveness and tradeoffs of computers and computing in daily life.

1. Find and adjust privacy settings for online accounts; discuss the positive and negative effects that social media can have on one's life.
2. Recognize how the overuse of technology can impact mental, physical, and emotional health.
3. Identify phishing emails, insecure websites (e.g., not https), and risky links when on the Internet; build a positive digital footprint (e.g., blog, website, social media).

CSS.DC.6-8.6
Apply strategies for troubleshooting hardware and software problems that may occur during use.

1. Choose the correct settings for printing and troubleshoot common printing issues (printer not turned on, printer not connected).
2. Connect devices (including internet/WIFI and Bluetooth) and troubleshoot common connection issues (e.g., disrupted Internet connection, website that is not loading).
3. Manage and deploy software updates and troubleshoot simple software issues (i.e., look up solutions to issues).

CSS.DC.6-8.7
Explore the relationship between computer hardware and software.

1. Develop a working vocabulary of embedded computing including digital, analog, events, microcontrollers, sensors, LED, switch, servo, cloud computing, and internet of things.
2. Investigate how software interacts with hardware in the boot process.
3. Analyze and explain how computers communicate information with simple hardware inputs and outputs.
4. Create a product that analyzes how simple computer hardware can be used to develop innovative new products that interact with the physical world.
5. Design a computer program that senses something in the real world and changes an output based on the input.

Knowledge Constructor
Conceptual Category: Data and Analysis
CSS.KC.6-8.14
Evaluate the accuracy, relevance, appropriateness, comprehensiveness, and biases that occur in electronic information sources.

1. Demonstrate an understanding of the credibility, bias, accuracy, relevance, age appropriateness, and comprehensiveness of electronic information sources.
2. Evaluate and discuss of the credibility, bias, accuracy, relevance, age appropriateness, and comprehensiveness of electronic information sources.
3. Apply strategies for determining the reliability of information found on the Internet.

CSS.KC.6-8.15
Gather, manipulate, and analyze data using a variety of digital tools to identify solutions and make informed decisions.

1. Gather data and calculate numerical equations using spreadsheet formulas and functions.
2. Use spreadsheet data to create tables, charts, and graphs.
3. Use spreadsheets and databases to make predictions, solve problems, and draw conclusions.

CSS.KC.6-8.16
Traverse online environments using critical thinking to find valid sources of information.

CSS.KC.6-8.17
Analyze various ways to visually represent data.

1. Interpret tables, charts, and graphs created by someone else.
2. Discuss design decisions in choosing between text, tables, charts, and graphs.
3. Discuss design decisions to make visualizations of data clear and concise.

Innovative Designer and Creator
CSS.IDC.6-8.18
Recognize that there may be multiple approaches to solving a problem.

CSS.IDC.6-8.19
Approach problem solving iteratively, using a cyclical process.

CSS.IDC.6-8.20
Design, develop, debug and implement computer programs.

1. Develop a working vocabulary of programming including flowcharting and/or storyboarding, coding, debugging, user interfaces, usability, variables, lists, loops, conditionals, programming language, events.
2. Utilize the design process to brainstorm, implement, test, and revise an ide
3. Cite evidence on how computers represent data and media (sounds, images, video, etc.).
4. Design a user interface and test with other users using a paper prototype.
5. Implement a simple algorithm in a computer program.
6. Develop an event driven program.
7. Create a program that accepts user and/or sensor input and stores the result in a variable.
8. Create a computer program that implements a loop.
9. Develop a program that makes a decision based on data or user input.
10. Debug a program with an error.

CSS.IDC.6-8.29
Create digital artifacts to address a current issue requiring resolution.

1. Summarize ethical, privacy, and legal issues of a digital world using current case studies.
2. Collaborate as a team to develop an artifact that represents multiple perspectives regarding a global crisis.
3. Analyze and explain the functionality and suitability (or appropriateness) of a computational artifact.
4. Develop a program for creative expression or to satisfy personal curiosity which may have visual, audible, and/or tactile results.
5. Develop a program specifically with the goal of solving a problem, creating new knowledge, or helping people, organizations, or society.

Computational Thinker
Conceptual Category: Recognizing and Defining Computational Problems
CSS.CT.6-8.30
Identify sub-problems to consider while addressing a larger problem.

CSS.CT.6-8.31
Recognize when it is appropriate to solve a problem computationally; Make sense of computational problems and persevere in solving them.

CSS.CT.6-8.32
Develop through application, logical observations relative to computational thinking procedures to analyze and solve problems current to everyday life.

1. Identify characteristics of computational thinking (decomposition, pattern recognition, algorithmic thinking and abstraction).
2. Explain issues and analyze routine hardware and software problems current to everyday life.
3. Apply troubleshooting concepts to issues regarding compatibility, data, and identity.
4. Describe ways to resolve operational problems caused by hardware errors.
5. Explain how technology can create ethical and legal issues in the business world and a technology-based society and how it can be used to solve & manage those issues.

CSS.CT.6-8.33
Utilize computational thinking to solve problems.

1. Make observations and organize the concepts of modularity, including functions and methods, as it relates to programming code reusability and cloud computing in the software industry.
2. Develop a working vocabulary of computational thinking including sequences, algorithms, binary, pattern matching, decomposition, abstraction, parallelization, data, automation, data collection, data analysis, boolean, integer, branches (if...then...else), and iteration {loops (For, While)}.
3. Analyze the problem-solving process, the input-process-output-storage model of a computer, and how computers help humans solve problems.
4. Develop an algorithm to decompose a problem of a daily task.

CSS.CT.6-8.34
Recognize when to use the same solution for multiple problems.

Conceptual Category: Data and Information
CSS.CT.6-8.35
Evaluate the storage and representation of data; Analyze how data is collected with both computational and non-computational tools and processes.

1. Discuss binary numbers, logic, sets, and functions and their application to computer science.
2. Explain that searches may be enhanced by using Boolean logic (e.g., using “not”, “or”, “and”).

Conceptual Category: Algorithms
CSS.CT.6-8.36
Understand and use the basic steps in algorithmic problem solving in computing and other authentic applications.

1. Select basic steps to solve algorithmic problems.
2. Evaluate basic steps of algorithmic problem solving to design solutions.
3. Solve algorithmic problems of increasing complexity.

Conceptual Category: Programming
CSS.CT.6-8.37
Use and compare simple coding control structures (e.g., if-then, loops)

1. Use a visual block-based and/or text-based programming language individually and collaboratively to solve problems of increasing complexity.
2. Create a program individually and collaboratively using a text-based programming. language; Identify variables and compare the types of data stored as variables.

Conceptual Category: Creating Computational Artifacts
CSS.CT.6-8.38
Consider the purpose of computational artifacts for practical use, personal expression, and/or societal impact.

1. Compare and contrast examples of high level and low-level programming languages.
2. Investigate the notion of hierarchy in computing including high level languages, translations, instruction sets, and logic circuits.
3. Develop problem solutions using a programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.

Conceptual Category: Testing and Refining Computational Artifacts
CSS.CT.6-8.39
Test computational artifacts systematically by considering multiple scenarios and using test cases.

Conceptual Category: Human Computer Interaction
CSS.CT.6-8.40
Describe how humans and machines interact to accomplish tasks that cannot be accomplished by either alone.

1. Identify what distinguishes humans from machines focusing on human intelligence versus machine intelligence (e.g., robot motion, speech and language understanding, and computer vision); Explain why some tasks can be accomplished more easily by computers.
2. Describe ways in which computers use models of intelligent behavior (e.g., robot motion, speech and language understanding, and computer vision) and how they differ (e.g., emotional decision making versus logical decisions, common sense, literal versus abstract).
3. Design and demonstrate the use of a device (e.g., robot, e-textile) to accomplish a task, individually and collaboratively.

Creative Communicator
Conceptual Category: Collaborating Around Computing
CSS.CC.6-8.41
Use online resources to participate in collaborative activities for the purpose of developing solutions or products.

CSS.CC.6-8.42
Improve teamwork and collaboration skills: providing useful feedback, integrating feedback, understanding, and accepting multiple perspectives.

1. Understand the difference between CC and BCC as well as Reply and Reply All and when to use each appropriately.

CSS.CC.6-8.43
Collaborate productively and recognize the value of working with individuals of varying perspectives, skills, and backgrounds.

1. Set and implement equitable expectations and workloads when working in teams.

CSS.CC.6-8.44
Demonstrate correct keyboarding techniques while increasing speed and maintaining accuracy.

CSS.CC.6-8.45
Use productivity technology tools (e.g. word processing, spreadsheet, presentation software) for individual and collaborative writing, communication, and publishing activities.

Global Collaborator
Conceptual Category: Fostering an Inclusive Computing Culture
CSS.GC.6-8.46
Recognize that equitable access to computing benefits society as a whole.

CSS.GC.6-8.47
Consider others' perspectives as well as one's own perspective when developing computational solutions.

CSS.GC.6-8.48
Consider the needs of a variety of end users regarding accessibility and usability.

CSS.GC.6-8.49
Use software applications to collaborate and create authentic products.

1. Identify and utilize the appropriate software application for productivity.
2. Use various applications in a professional manner to share and communicate with peers and teachers.
3. Share documents created using word processing, presentation, and spreadsheet software.
4. Create original works using software applications in a collaborative manner.
5. Collaborate in small groups to create and edit online documents in real time.
6. Identify and use appropriate file sharing strategies (e.g., copy and paste, links, posts, and attachments).

Glossary of Computer Science Terms

These terms are used throughout the standards. They are content-specific vocabulary for Computer Science.

Term	Definition	Example
Abstraction	The process of taking away or removing characteristics from something in order to make it less complex. The product should be a new representation of essential characteristics. The new representation hides details that are irrelevant to the problem at hand.	To represent a person, an abstraction may include two arms, two legs, a head, and a torso but no hair or toes. This representation gives enough information to show a person without being too complex.
Algorithm	Detailed, step-by-step instructions for solving a problem or completing a task.	The set of steps used to solve a long division problem is an example of an algorithm.
Analog	A defining characteristic of data; analog data are stored in a continuous transmission of a signal. It is often contrasted with digital, which is how computers store and process data as a set of individual symbols.	A compact disc is digital; a vinyl record is analog.
Artifact (computational)	Anything created by a human using a computer.	A word processing document, an app, and a webpage are all computational artifacts.
Binary	A number system using only on the numerals 0 and 1.	The binary number 01011 converts to 11 in decimal numbers.
Biometric	The measurement and analysis of unique physical or behavioral characteristics (such as fingerprint or voice patterns) especially as a means of verifying personal identity.	Fingerprint scanners utilize a biometric evaluation to grant access.

Block-based programming language	A visual representation of common sets of instructions for coding that can be organized to create computer programs; block-based programing is often used to teach coding to younger or novice learners.	A popular block-based coding language is Blockly.
Coding	Creating a computer program.	Writing directions for a computer using a computer language such as Java, Python, or Blockly.
Computational Thinking	A problem-solving process used to formulate problems in a way that a computer and other tools could be used to help solve.	

Conditional	A programming statement, often starting with "if", in which one half expresses something that depends on the other half.	If student's grade is greater than or equal to 60
		Print "passed"
		else
		Print "failed"
		endif

Curate	Collect, organize, and present information typically using professional or expert knowledge.	Selecting a set of pictures to share or add to a photo album.
Debug	The process of finding and removing errors from computer programs.	Correcting errors.
Decomposition	Specific to computer science, decomposition means breaking a complex problem or system into parts that are easier understand.	To create an app that calculates an ideal heart rate, the program would break down the process to input of information from a patient, calculation of that information, and output of the ideal heart rate.
Design Process	A formal approach used by architects, engineers, and scientists for breaking down a large project into manageable chunks.	

Drone	A remote-controlled pilotless aircraft or missile.	A photographer can use a drone to take aerial pictures.
Event	An action or occurrence recognized by software, often originating from the external environment, that may be handled by the software.	Accepting input from a user is an event that may be followed up by some processing activity.
Hexadecimal (Hex)	Relating to or using a system of numerical notation that has 16 digits rather than 10 as its base.	The number 15 in our common base ten decimal system is represented with the letter 'F' in hexadecimal.
Ideate	The process of generating ideas and solutions.	Sketching, prototyping, or brainstorming can be processes for ideation.
Loop	A sequence of instructions that is continually repeated until a certain condition is reached.	An action that is performed again and again by a computer program.
Model	Constructing a representation of some part of a problem or system.	A budget is a model for how money is spent and earned.
Ordinal	Relating to an ordinal number; representing a position in a series.	1st, 2nd, 3rd, 4th, ...
Phishing	The fraudulent practice of sending emails purporting to be from reputable sources in order to entice individuals to reveal personal information, such as passwords and credit card numbers.	A phishing email is a fake message from a place like the Internal Revenue Service requesting a social security number.
Prototype	A model of something from which other forms are developed or copied.	At an auto show, a "concept car" is a prototype of a car that may go into mass production.
Remix	To change a set of code by adding or rearranging smaller code segments to create a different outcome.	A computer program that uses segments of other programs to solve a problem.

Scratch	A block-based programming platform commonly used for novice programmers.	
Sequence	An ordered, step-by-step process of an action or event proceeding in a pattern.	5, 10, 15, 20 is a sequence that relies on a pattern of +5.
Unplugged	Activities used for teaching computational thinking or computer science without a computing device.	Using playing cards to teach sorting is an unplugged activity to teach how computers sort data.