Public Schools of North Carolina
State Board of Education | Department of Public Instruction

2020 North Carolina K12 Computer Science Standards with Descriptions.

This document is designed to help North Carolina educators teach the NC Standard Course of Study for Computer Science.

This document provides more detailed descriptions of each standard in the 2020 NC K12 Computer Science Standards which are based on the 2017 Computer Science Teachers Association Computer Science Standards.

Introductory Computer Science

ICS-CS-01	**Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.**	**Computing Systems** Devices
	Computing devices are often integrated with other systems, including biological, mechanical, and social systems. A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The creation of integrated or embedded systems is not an expectation at this level. Students might select an embedded device such as a car stereo, identify the types of data (radio station presets, volume level) and procedures (increase volume, store/recall saved station, mute) it includes, and explain how the implementation details are hidden from the user.	
ICS-CS-02	**Compare levels of abstraction and interactions between application software, system software, and hardware layers.**	**Computing Systems** Hardware & Software
	At its most basic level, a computer is composed of physical hardware and electrical impulses. Multiple layers of software are built upon the hardware and interact with the layers above and below them to reduce complexity. System software manages a computing device's resources so that software can interact with hardware. For example, text editing software interacts with the operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. For example, students may explore the progression from voltage to binary signal to logic gates to adders and so on. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level.	
ICS-CS-03	**Explain the roles of operating systems including memory management, data storage/retrieval, process management, and access control.**	**Computing Systems** Hardware & Software
	Examples of roles could include memory management, data storage/retrieval, processes management, and access control.	
ICS-CS-04	**Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.**	**Computing Systems** Troubleshooting
	Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is similar to one they have seen before or adapt solutions that have worked in the past. Examples of complex troubleshooting strategies include resolving connectivity problems, adjusting	

	system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another. Students could create a flow chart, a job aid for a help desk employee, or an expert system.	
ICS-NI-01	**Evaluate the relationship between routers, switches, servers, and topology with regard to networks.**	**Networks & the Internet** Network Communication & Organization
	Each device is assigned an address that uniquely identifies it on the network. Routers function by comparing IP addresses to determine the pathways packets should take to reach their destination. Switches function by comparing MAC addresses to determine which computers or network segments will receive frames. Students could use online network simulators to experiment with these factors.	
ICS-NI-02	**Identify examples to illustrate how sensitive data can be affected by malware and other attacks.**	**Networks & the Internet** Cybersecurity
	Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, present threats to sensitive data. Students might reflect on case studies or current events in which governments or organizations experienced data leaks or data loss as a result of these types of attacks.	
ICS-NI-03	**Recommend cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.**	**Networks & the Internet** Cybersecurity
	Security measures may include physical security tokens, two-factor authentication, and biometric verification. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be securely stored and transmitted. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures. Students should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.	
ICS-NI-04	**Compare various security measures and consider tradeoffs between the usability and security of a computing system.**	**Networks & the Internet** Cybersecurity
	Security measures may include physical security tokens, two-factor authentication, and biometric verification, but choosing security measures involves tradeoffs between the usability and security of the system. The needs of users and the sensitivity of data determine the level of security implemented. Students might discuss computer security policies in place at the local level that present a tradeoff between usability and security, such as a web filter that prevents access to many educational sites but keeps the campus network safe.	

ICS-DA-01	**Compare different binary representations of data, including text, sound, images, and numbers.**	**Data & Analysis** Storage
	For example, convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation, and logic gates.	
ICS-DA-02	**Evaluate the tradeoffs in how data elements are organized and where data is stored.**	**Data & Analysis** Storage
	People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity. Students should evaluate whether a chosen solution is most appropriate for a particular problem. Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud.	
ICS-DA-03	**Create interactive data visualizations using software tools to help others better understand real-world phenomena.**	**Data & Analysis** Collection Visualization & Transformation
	People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Students should model phenomena as systems, with rules governing the interactions within the system and evaluate these models against real-world observations. For example, flocking behaviors, queueing, or life cycles. Google Fusion Tables can provide access to data visualization online.	
ICS-DA-04	**Create computational models that represent the relationships among different elements of data collected.**	**Data & Analysis** Inference & Models
	Computational models make predictions about processes or phenomena based on selected data and features. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and ability to understand a system. Predictions or inferences are tested to validate models. Students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.	
ICS-AP-01	**Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.**	**Algorithms & Programming** Algorithms

	A prototype is a computational artifact that demonstrates the core functionality of a product or process. Prototypes are useful for getting early feedback in the design process, and can yield insight into the feasibility of a product. The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm.	
ICS-AP-02	**Explain the use of artificial intelligence within computing systems.**	**Algorithms & Programming** Algorithms
	Artificial Intelligence (AI) has become a critical component of nearly all aspects of technology in our society. Students should be able to explain how AI is used within technology. For example, students might explain how a store makes recommendations based on search and buying history.	
ICS-AP-03	**Utilize lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.**	**Algorithms & Programming** Variables
	Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (arrays) to account for the differences.	
ICS-AP-04	**Justify the selection of specific control structures, considering implementation, readability, and program performance.**	**Algorithms & Programming** Control
	Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion. For example, students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence.	
ICS-AP-05	**Iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.**	**Algorithms & Programming** Control
	In this context, relevant computational artifacts include programs, mobile apps, or web apps. Events can be user-initiated, such as a button press, or system-initiated, such as a timer firing. At previous levels, students have learned to create and call procedures. Here, students design procedures that are called by events. Students might create a mobile app that updates a list of nearby points of interest when the device detects that its location has been changed.	

ICS-AP-06	**Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.**	**Algorithms & Programming** Modularity
	At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).	
ICS-AP-07	**Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.**	**Algorithms & Programming** Modularity
	Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.	
ICS-AP-08	**Systematically design programs for broad audiences.**	**Algorithms & Programming** Modularity
	Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences. Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.	
ICS-AP-09	**Refine programs by incorporating feedback from users.**	**Algorithms & Programming** Modularity
	Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences. Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.	
ICS-AP-10	**Evaluate licenses that limit or restrict use of computational artifacts when using resources such as software libraries.**	**Algorithms & Programming** Program Development

	Examples of software licenses include copyright, freeware, and the many open-source licensing schemes. At previous levels, students adhered to licensing schemes. At this level, they should consider licensing implications for their own work, especially when incorporating libraries and other resources. Students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license.	
ICS-AP-11	**Evaluate computational artifacts for usability.**	**Algorithms & Programming** Program Development
	Testing and refinement is the deliberate and iterative process of improving a computational artifact. Students should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts. For example, students could solicit feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.	
ICS-AP-12	**Modify computational artifacts to increase usability and accessibility.**	**Algorithms & Programming** Program Development
	The testing and refinement process of improving a computational artifact includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students modify artifacts by incorporating feedback from a variety of end users to improve usability.	
ICS-AP-13	**Develop computational artifacts working in team roles using collaborative tools.**	**Algorithms & Programming** Program Development
	Collaborative tools could be as complex as a source code version control system or as simple as a collaborative word processor. Team roles in pair programming are driver and navigator but could be more specialized in larger teams. As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students. Students might work as a team to develop a mobile application that addresses a problem relevant to the school or community, selecting appropriate tools to establish and manage the project timeline; design, share, and revise graphical user interface elements; and track planned, in-progress, and completed components.	
ICS-AP-14	**Explain design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.**	**Algorithms & Programming** Program Development
	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program.	

ICS-IC-01	**Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.**	**Impacts of Computing** Culture
	Computing may improve, harm, or maintain practices. Equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people who lack access to broadband or who have various disabilities. Students should also begin to identify potential bias during the design process to maximize accessibility in product design.	
ICS-IC-02	**Elaborate how computational innovations have and may continue to impact society.**	**Impacts of Computing** Culture
	Our world has been transformed by computational innovations. Students should be able to articulate how these innovations have impacted our society. For example, students might identify examples from education, healthcare, art/entertainment, or energy and create a presentation describing the impact.	
ICS-IC-03	**Evaluate how equity, access, and influence impact distribution of computing resources in a global society.**	**Impacts of Computing** Culture
	The distribution of computing resources is not uniform across all parts of our global society. Students should evaluate how equity and access are impacted by factors such as wealth, geographic location, governmental programs. For example, students might explore the effectiveness of the US E-rate program for increasing access to technology in schools.	
ICS-IC-04	**Test computational artifacts to reduce bias and equity deficits.**	**Impacts of Computing** Culture
	Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities. Students should begin to identify potential bias during the design process to maximize accessibility in product design and become aware of professionally accepted accessibility standards to evaluate computational artifacts for accessibility.	
ICS-IC-05	**Demonstrate ways a given algorithm applies to problems across disciplines.**	**Impacts of Computing** Culture
	Computation can share features with disciplines such as art and music by algorithmically translating human intention into an artifact. Students should be able to identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and that can be solved computationally.	

ICS-IC-06	**Utilize tools and methods for collaboration on a project to increase connectivity of peers.**	**Impacts of Computing** Social Interactions
	Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers. Students should explore different collaborative tools and methods used to solicit input from team members, classmates, and others, such as participation in online forums or local communities. For example, students could compare ways different social media tools could help a team become more cohesive.	
ICS-IC-07	**Explain the beneficial and harmful effects that intellectual property laws can have on innovation.**	**Impacts of Computing** Safety Law & Ethics
	Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing. For example, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship. Students should be aware of intellectual property laws and be able to explain how they are used to protect the interests of innovators and how patent trolls abuse the laws for financial gain.	
ICS-IC-08	**Explain privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.**	**Impacts of Computing** Safety Law & Ethics
	Data can be collected and aggregated across millions of people, even when they are not actively engaging with or physically near the data collection devices. This automated and non-evident collection can raise privacy concerns, such as social media sites mining an account even when the user is not online. Other examples include surveillance video used in a store to track customers for security or information about purchase habits or the monitoring of road traffic to change signals in real time to improve road efficiency without drivers being aware. Methods and devices for collecting data can differ by the amount of storage required, level of detail collected, and sampling rates.	
ICS-IC-09	**Evaluate the social and economic implications of privacy in the context of safety, law, and ethics.**	**Impacts of Computing** Safety Law & Ethics
	Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing. Students might review case studies or current events which present an ethical dilemma when an individual's right to privacy is at odds with the safety, security, or wellbeing of a community.	