Computer Science Model Curriculum: Public Comment DRAFT

Introduction

The Ohio Department of Education's Office of Learning and Instructional Strategies is in the process of revising Ohio's Model Curriculum for Computer Science. The following is a draft of the revised model curriculum highlighted with changes from the first round of public comments.

The State Board of Education adopted Ohio's Learning Standards and Model Curriculum for Computer Science in December 2018. Implemented in the 2019-2020 (FY2020) school year, the purpose of Ohio's Learning Standards and Model Curriculum for Computer Science is to provide guidance to schools and districts. Ohio educators, many of whom engage daily with Ohio students, led the process to create Ohio's Learning Standards and Model Curriculum for Computer Science.

The review and revision process presents an opportunity for families, computer science professionals, community members and other educators, to provide suggestions for improving the model curriculum through a public comment survey. The second public comment survey **is open March 2 through March 18.**

In the Fall of 2021, the public was invited to provide comments on the current standards and model curriculum. Advisory group members, who have an educational background or professional experience in computer science, met several times to discuss the public comment and make suggestions for revisions. The advisory group then provided direction and guidance to the working groups of Ohio educators, who worked to write the revisions.

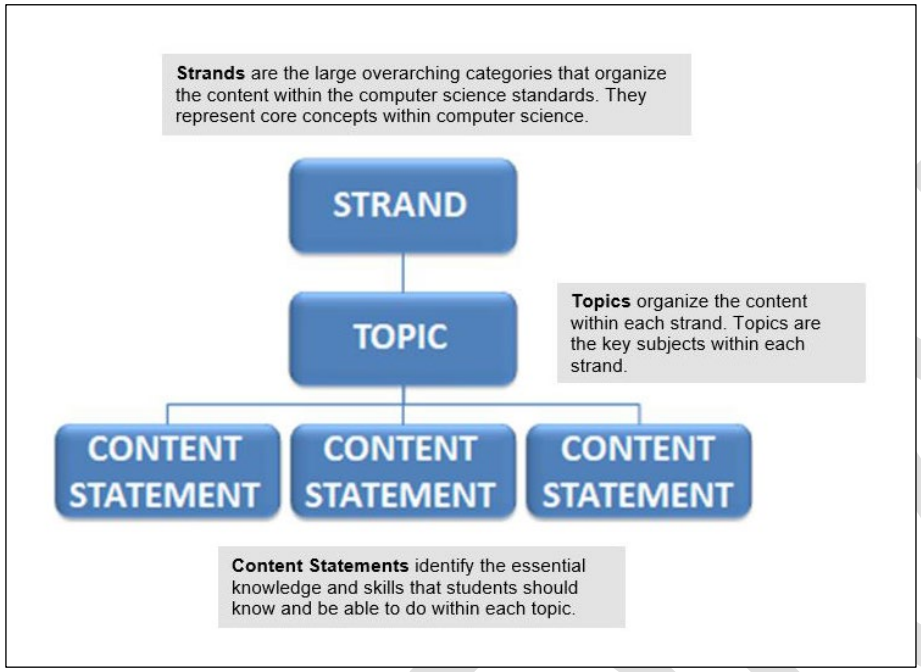
In your review, please focus on the content of the proposed model curriculum changes. All materials will be going through additional technical edits, but because of the timeline outlined in HB110, the Department wanted to make these drafts available for public comment at this time.

To see the model curriculum adopted by the Ohio State Board of Education in 2018 please visit the Model Curriculum for Computer Science.

Computer Science Model Curriculum: Public Comment DRAFT

OVERVIEW OF THE COMPUTER SCIENCE STANDARDS FRAMEWORK

Ohio's Computer Science Standards are organized by strands, topics and content statements.



Kindergarten through Grade 8-Content statements are organized *by grade level*. Below is an example of a content statement for kindergarten and its corresponding content statement code. This content statement addresses the topic of *Devices* within the *Computing Systems* strand.

COMPUTING SYSTEMS

Devices

CS.D.K.a With guidance, identify and label commonly used devices and their components, explaining their connection to different tasks, to perform a variety of tasks.

Grades 9-12 - Content statements are organized by grade band into two levels – Foundational and Advanced. Foundational level standards are targeted at students who have limited or no experience in Computer Science. Foundational level courses ask students to demonstrate competency of one or more subsets of the standards. Advanced standards are targeted at students demonstrating mastery at the foundational level. See an example of a content statement for high school and its corresponding content statement code below. This content statement addresses the topic of Networking within the Networks and the Internet strand, at the Foundational Level.

NETWORKS AND THE INTERNET

Networking

NI.N.9-12.F.a Evaluate and select networking devices to establish scalable communications.

Computer Science Model Curriculum: Public Comment DRAFT

Strand	Computing Systems
Topic	**Hardware/Software**
CS.HS.K.a With guidance and support, identify and use hardware and software necessary for accomplishing a task.	**Expectations for Learning** **LEARNING PROGRESSION** In kindergarten, with teacher support, students name and use hardware and software to perform functions. In grade 1, students will begin comparing hardware and software and their functions. **IMPORTANT CONCEPTS** • Hardware and software identification and the function they perform **KEY SKILL/PROCEDURES** • With guidance and support, identify the differences between hardware and software, their functions and purpose • Use hardware and software to perform functions **Content Elaborations** **CLARIFICATIONS** Students will classify and sort hardware and software. They will label software and hardware and their functions. **CONTENT FOCUS** The focus is on the identification of hardware and software and their function. **COMPUTER SCIENCE PRACTICES** *Practice 7.* Communicating About Computing 2. Describe, justify and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**

ATP.A.7.a Select and modify pseudocode for a multi-step process to solve a problem.

Expectations for Learning

LEARNING PROGRESSION

In previous grades, students should have developed a basic understanding of what goes into an algorithm and learned how to write pseudocode. As students create an algorithm, they should be able to evaluate the solution and make modifications in their pseudocode to solve the problem. At the end of 7th grade, students should be able to identify the parts of a program's pseudocode (input, output, decisions). In future grades, students will write their own pseudocode and be able to justify the most efficient solution for a multi-step process.

IMPORTANT CONCEPTS

- Understand the flow of a program
- Compare different sets of pseudocode and determine the most efficient solution

KEY SKILL/PROCEDURES

- Identify the inputs, outputs, processes and decisions
- Use proper symbols to create flow charts to represent pseudocode

Content Elaborations

CLARIFICATIONS

Students should be able to write pseudocode to model an algorithm. This could be writing a recipe for making a sandwich, writing directions between two locations, or writing the steps necessary to compute a semester grade. At this level, the expectation is that looping is done utilizing iteration, not recursion.

CONTENT FOCUS

Students should be able to identify the inputs, outputs, and decision steps within the algorithms they create in pseudocode. Students should be able to create a flow chart representation of their pseudocode, using proper symbols for inputs, outputs, and decisions.

Computer Science Model Curriculum: Public Comment DRAFT

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**
	COMPUTER SCIENCE PRACTICES *Practice 3.* Recognizing and Defining Computational Problems 2. Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures. *Practice 4.* Developing and Using Abstractions 4. Model phenomena and processes and simulate systems to understand and evaluate potential outcomes. *Practice 5.* Creating Computational Artifacts 3. Modify an existing artifact to improve or customize it. *Practice 6.* Testing and Refining Computational Artifacts 1. Systematically test computational artifacts by considering all scenarios and using test cases.

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**

ATP.A.8.a Create multiple pseudocode to solve a multi-step process and justify the most efficient solution.

Expectations for Learning

LEARNING PROGRESSION

In previous grades, students should have developed a basic understanding of what goes into an algorithm and know how to write pseudocode. As students create multiple solutions to a problem, they must be able to analyze various solutions and determine the most efficient. At the end of 8th grade, students should be able to identify the parts of a program's pseudocode (input, output, decisions). In future grades, students will learn how to break a problem into steps, and then in addition to just pseudocode be able to represent the algorithm using process and data flow diagrams. Students will be able to explain the steps of the problem and be able to communicate why they broke the problem down the way they did. Finally, they will be able to implement their created algorithm in an appropriate programming language and optimize it using best practices.

IMPORTANT CONCEPTS

- Understand the flow of a program
- Compare different sets of pseudocode and determine the most efficient solution
- Recognize and define iteration

KEY SKILL/PROCEDURES

- Identify the inputs, outputs, processes and decisions
- Use proper symbols to create flow charts to represent pseudocode

Content Elaborations

CLARIFICATIONS

Student should be able to write more than one pseudocode solution for the same problem (e.g., two different sets of directions between two locations). Then, they should be able to analyze each proposed solution and list the pros and cons of each. At this level, the expectation is that looping is done utilizing iteration, not recursion.

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**
	CONTENT FOCUS Student should be able to identify the inputs, outputs, and decision steps within the algorithms they create in pseudocode. Student should be able to create a flow chart representation for each of their pseudocode solutions, using proper symbols for inputs, outputs, and decisions. **COMPUTER SCIENCE PRACTICES** *Practice 3.* Recognizing and Defining Computational Problems 2. Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures. *Practice 4.* Developing and Using Abstractions 4. Model phenomena and processes and simulate systems to understand and evaluate potential outcomes. *Practice 5.* Creating Computational Artifacts 3. Modify an existing artifact to improve or customize it. *Practice 6.* Testing and Refining Computational Artifacts 1. Systematically test computational artifacts by considering all scenarios and using test cases.

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**

ATP.A.9-12.F.a Define and use appropriate problem solving strategies and visual artifacts to create and refine a solution to a real-world problem.

ATP.A.9-12.F.b Define and implement an algorithm by decomposing problem requirements from a problem statement to solve a problem.

Expectations for Learning

LEARNING PROGRESSION

By the end of grade 8:
Students have a basic understanding of what goes into an algorithm and know how to write pseudocode. Students are able to analyze various solutions to determine the most efficient. Students are able to identify the parts of a program's pseudocode (input, output, decisions).

In the 9-12 Foundational Level:
Students will learn how to break a problem into steps, and then, in addition to pseudocode, be able to represent the algorithm using process and data flow diagrams. Students will explain the steps of the problem and be able to communicate why they broke the problem down the way they did. Finally, they will be able to implement their created algorithm in an appropriate programming language and refine it using best practices.

In the 9-12 Advanced Level:
Students will progress to learn about complex list algorithms such as sorting and searching. They will also understand and explain the concept of recursion, where a method calls itself until an exit condition is met.

IMPORTANT CONCEPTS

- Algorithms are the steps to solve a problem
- There are strategies that simplify problem solving
- Problems can be broken down into smaller parts, such as inputs, outputs, decisions, loops and calculations
- The flow of information through a program can be represented visually
- Steps of a computer program can be represented by a diagram

KEY SKILL/PROCEDURES

- Demonstrate how to break a problem into steps
- Create a process flow diagram, data flow diagram and pseudocode for an algorithm
- Explain the steps of a problem
- Implement an algorithm in an appropriate programming language

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**

- Optimize an algorithm by removing redundant or repeated code and using alternative control structures
- Utilize control structures to create an iterative algorithm

Content Elaborations

CLARIFICATIONS

Algorithms are the steps to solve a problem.

A complex problem is one in which the solution requires multiple (i.e., three or more) methods, objects and/or data structures.

Problems can be broken down into smaller parts.

Flow of information through a program can be represented visually.

Steps of a computer program can be represented by a diagram.

Present programming languages to students without extra functionality added by libraries, modules, etc. For example, using JavaScript without additional libraries like jQuery or React.

A problem-solving strategy has the following steps:

- Identify the problem
- Analyze the problem
- Identify decision criteria
- Develop multiple solutions
- Choose the optimal solution

CONTENT FOCUS

The most important concept is the breaking down of problems into smaller pieces. You should also focus on at least one way of visualizing the algorithm.

You should also focus on one programming language to implement the algorithm.

Computer Science Model Curriculum: Public Comment DRAFT

Strand	Algorithmic Thinking and Programming
Topic	**Algorithms**
	COMPUTER SCIENCE PRACTICES *Practice 3.* Recognizing and Defining Computational Problems 1. Identify complex, interdisciplinary, real-world problems that can be solved computationally. 2. Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures. *Practice 5.* Creating Computational Artifacts 2. Create a computational artifact for practical intent, personal expression, or to address a societal issue. 3. Modify an existing artifact to improve or customize it. *Practice 6.* Testing and Refining Computational Artifacts 3. Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability and accessibility.