K-12 Computer Science Standards Writing Steering Committee

Claiborne Building | Thomas Jefferson Room 1-136 | 1201 North Third Street, Baton Rouge, LA 70802

July 30, 2024

Call to Order

Roll Call

Agenda

Agenda

I. Call to Order

II. Roll Call

III. Approval of minutes of the meeting held July 11, 2024

IV. Consideration of a summary report regarding the grade band workgroup recommendations for computer science content standards on Concept 4: Algorithms and Programming

V. Consideration of an update regarding the work of the computer science grade band workgroups

Approval of minutes of the meeting held July 11, 2024

Consideration of a summary report regarding the grade band workgroup recommendations for computer science content standards on Concept 4: Algorithms and Programming

Overarching Themes

Students will be able to

- engage with computational thinking practices to logically manage complex tasks and problem-solve;
- master how algorithms can make problem-solving more accessible and increase task efficiency in real-world applications;
- interact with programming as both individuals and in teams to apply industry best practices; and
- follow the appropriate attribution and licensing laws when programming.

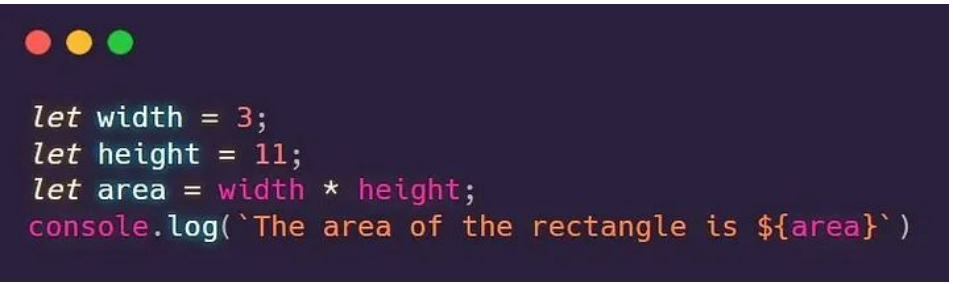
Core Concept 4 Algorithms and Programming

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, how to store the information, practicing the decomposition of more significant problems into simpler ones, recombining existing solutions, and analyzing various solutions to a problem to locate the most appropriate solution.

Subconcepts

1. **Variables and Algorithms**- Students begin working with variables and simple algorithms in K-5. Students continue applying computational thinking with more advanced algorithms and data structures in grades 6-12.

2. **Control Structures**- In grades K-5, students engage with sequential executions and basic control structures. As students progress into grades 6-12, they build on their mastery by exploring more complex arrangements to support complex program structures.



Subconcepts

3. **Modularity**- Students begin working with decomposing and recombining algorithms in K-5. As students progress into grades 6-12, they learn how to recognize increasingly complex patterns, use general, reusable solutions for commonly occurring scenarios, and clearly describe tasks in widely usable ways.

4. **Program Development**- In grades K-5, students engage with simple programs and develop an understanding of how and why people create programs. In 9-12 grades, they increase the complexity of their programming and explore the potential trade-offs made in program design.

9-12 Variables and Algorithms
1A. Explain what computing memory is, where computing data is stored, and how data is retrieved.
1B. Assess variables and classify their scope as global, local, or nested lexical.
1C. Design algorithms that can be adapted to express an idea or solve a problem.
1D. Use and adapt classical algorithms to solve computational problems.

9-12 Control Structures
2A. Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of the choices made.
2B. Design and iteratively develop computational artifacts using events to initiate instructions.
9-12 Modularity
3A. Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
3B. Create artifacts using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

9-12 Program Development

4A. Design and develop computational artifacts by working in team roles using version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation).
4B. Use a standard library and/or application programming interface (API) to create reusable code components to create simple programs and innovate existing programs to increase complexity and refine programs.
4C. Summarize the key phases of the Software Development Life Cycle (SDLC). Utilize the SDLC to create a computational artifact that leads to a minimum viable product.
4D. Modify an existing program to add functionality and discuss intended and unintended implications (e.g., the unintended breaking of other functionalities).

9-12 Program Development

4E. Evaluate and iteratively refine computational artifacts to make them more usable and accessible (e.g., correctness, usability, readability, and program efficiency).
4F. Develop and utilize test cases to verify that a program performs according to its design specifications.
4G. Apply the appropriate documentation techniques to make programs more accessible to debug and to be maintained by others.
4H. Evaluate licenses that limit or restrict the use of computational artifacts when utilizing resources such as libraries.

6-8 Variables and Algorithms
1A. Evaluate and use naming conventions for variables to accurately communicate their meaning to other users and programmers.
1B. Compare and contrast data constants and variables.
1C. Evaluate algorithms in terms of their efficiency, correctness, or clarity.
6-8 Control Structures
2A. Compare and contrast control structure types and explain their functions.
2B. Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

6-8 Modularity
3A. Decompose problems into parts to facilitate program design, implementation, and review.
3B. Create procedures and/or functions with parameters to organize code and make it easier for future reuse.

6-8 Program Development

4A. Seek and incorporate feedback from peers to employ user-centered design solutions.
4B. Incorporate existing resources into original programs and give the proper attributions.
4C. Systematically test document outcomes and refine programs using a range of test cases.
4D. Develop computational artifacts by working as a team, distributing tasks, and maintaining an iterative project timeline.
4E. Test and debug using applicable industry practices to document and peer review code.

K-5 Variables and Algorithms
1A. Create clearly named variables representing different data types and perform operations on their values.
1B. Create, use, and apply an algorithm to complete a task. Compare the results of algorithm usage trials and refine the algorithm.
K-5 Control Structures
2. Define what a control structure is and create programs that include sequences, conditionals, events, and loops.

K-5 Modularity
3A. Define and apply decomposition to create smaller subproblems that can be solved through step-by-step instructions from a complex problem.
3B. Modify, remix, or incorporate parts of an existing problem's solution to develop something new or add more advanced features to a program.

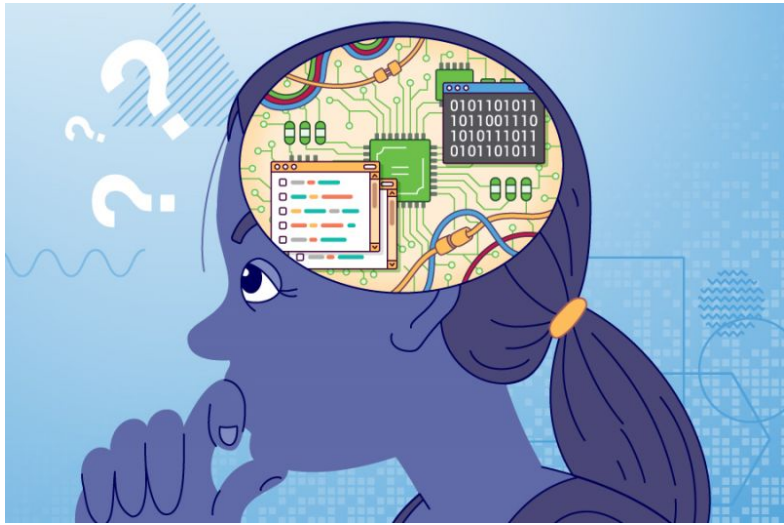
K-5 Program Development

4A. Develop a plan that describes a series of steps to achieve a goal with expected outcomes (Creating a simple program).
4B. Test and debug a program or algorithm to ensure it produces its intended outcome.
4C. Collaborate with a team of peers to design, implement, test, and review the stages of program development.

K-5 Program Development
4D. Describe and justify the steps taken and choices made during a program's development.
4E. Using an iterative process, test a program's execution step by step including and documenting areas of refinement.
4F. Identify intellectual property rights and apply the appropriate attribution when creating or remixing programs.

Progression of Concept 4: Algorithms and Programming

Learning to program is a skill. Mastering the complex thinking, the use of algorithms, the joy of problem solving, and the exhilaration of creating new solutions to problems is a lifelong process. The age appropriate standards created by this Committee lay the foundation for many exciting projects, explorations, concept masteries, and moments of wonder as students progress from K-12. Through the cognitive processes stimulated by programming, students can see our world in new ways.



Recess

Consideration of an update regarding the work of the computer science grade band workgroups

Schedule of K-12 Computer Science Standards Writing Committee Meetings

Date and Time	Meeting and Location
May 7, 2024, 9 a.m. - 4 p.m.	Meeting 1 - Claiborne Building, Baton Rouge
June 7, 2024, 9 a.m. - 4 p.m.	Meeting 2 - Claiborne Building, Baton Rouge
June 20, 2024, 9 a.m. - 4 p.m.	Meeting 3 - Claiborne Building, Baton Rouge
July 11, 2024, 9 a.m. - 4 p.m.	Meeting 4 - Claiborne Building, Baton Rouge
July 30, 2024, 9 a.m. - 4 p.m.	Meeting 5 - Claiborne Building, Baton Rouge
August 13, 2024, 9 a.m. - 4 p.m.	Meeting 6 - Claiborne Building, Baton Rouge
August 27, 2024, 9 a.m. - 4 p.m.	Meeting 7 - Claiborne Building, Baton Rouge

Please contact STEM@la.gov