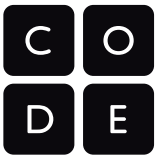
K12 MARCO DE LAS CIENCIAS DE LA COMPUTACIÓN

Este material ha sido traducido al español por Fundación Kodea ®

Comité directivo del marco de las ciencias de la computación K-12

Association for Computing Machinery



CSTEACHERS.ORG
COMPUTER SCIENCE TEACHERS ASSOCIATION

cyber innovation center®
Collaboration • Research • Technology

NATIONAL MATH + SCIENCE INITIATIVE

Transforming Math and Science Education
NMS.org

Cita sugerida: K – 12 Marco de las Ciencias de la Computación. (2016). Recuperado de http://www.k12cs.org.

Atribución sugerida: "El marco de las ciencias de la computación K-12, liderado por la Asociación de Maquinaria de Computación, Code.org, la Asociación de Profesores de las Ciencias de la Computación, el Centro de Innovación Cibernética y la Iniciativa Nacional de Matemáticas y Ciencia en asociación con los estados y distritos, informó el desarrollo este trabajo."

Se proveían ejemplos de programas y recursos para la conveniencia del lector y no representan un endoso.

Reconocimientos

El marco de las ciencias de la computación K-12 fue un esfuerzo de la comunidad. Las siguientes secciones reconocen a las diferentes personas y organizaciones que jugaron un papel importante en el desarrollo del marco.

Comité Directivo

Gracias a Mehran Sahami de la Asociación de Maquinaria de Computación, Cameron Wilson de Code.org, Mark Nelson de la Asociación de Profesores de las Ciencias de la Computación (CSTA), Krystal Corbett del Centro de Innovación Cibernética y Deepa Muralidhar de la Iniciativa Nacional de Matemáticas y Ciencias para guiar el proceso del marco.

Estados y Distritos

Los siguientes estados y distritos participaron en el desarrollo del marco mediante la nominación de escritores y comentarios sobre el marco.

Estados	Distritos
Arkansas	Charles County Public Schools, MD
California	Chicago Public Schools, IL
Georgia	New York City Department of Education, NY
Idaho	San Francisco Unified School District, CA.
Indiana	
Iowa	
Maryland	
Massachusetts	
Nebraska	
Nevada	
New Jersey	
North Carolina	
Utah	
Washington	

Escritores

Las biografías de los escritores se encuentran en el **Apéndice B**.

Julie Alano
Computer Science Teacher, Hamilton Southeastern High School

Derek Babb
Computer Science Teacher, Omaha North Magnet High School

Julia Bell
Associate Professor of Computer Science, Walters State Community College

Tiara Booker-Dwyer
Education Program Specialist, Maryland State Department of Education

Leigh Ann DeLyser
Director of Education and Research, CSNYC

Caitlin McMunn Dooley
Deputy Superintendent for Curriculum and Instruction Georgia Department of Education; Associate Professor, Georgia State University

Diana Franklin
Director of Computer Science Education, UChicago STEM Ed

Dan Frost
Senior Lecturer, University of California, Irvine

Mark A. Gruwell
Co-Facilitator, Iowa STEM Council Computer Science Workgroup

Maya Israel
Assistant Professor, University of Illinois at Urbana Champaign

Vanessa Jones
Instructional Technology Design Coach, Austin Independent School District

Richard Kick
Mathematics and Computer Science Teacher, Newbury Park High School

Heather Lageman
Executive Director of Leadership Development, Baltimore County Public Schools

Todd Lash
Doctoral Student/Contributing Member, University of Illinois, CSTA K–8 Task Force

Irene Lee
Researcher, Massachusetts Institute of Technology

Carl Lyman
Specialist over Information Technology Class Cluster, Utah State Board of Education

Daniel Moix
Computer Science Education Specialist, Arkansas School for Mathematics, Sciences & Arts

Dianne O'Grady-Cunniff
Computer Science Teacher, La Plata High School

Anthony A. Owen
Coordinator of Computer Science, Arkansas Department of Education

Minsoo Park
Director of Teaching and Learning, Countryside School

Shaileen Crawford Pokress
Visiting Scholar, Wyss Institute at Harvard; K–12 Curriculum Designer

George Reese
Director of MSTE, MSTE Office at University of Illinois at Urbana Champaign

Hal Speed
Founder, CS4TX

Alfred Thompson
Computer Science Teacher, Bishop Guertin High School

Bryan Twarek
Computer Science Program Administrator, San Francisco Unified School District

A. Nicki Washington
Associate Professor, Computer Science, Winthrop University

David Weintrop
Postdoctoral Researcher, UChicago STEM Ed

Asesores

Alana Aaron, *New York City Department of Education*
Owen Astrachan, *Duke University*
Karen Brennan, *Harvard University*
Josh Caldwell, *Code.org*
Jill Denner, *Education Training Research*
Brian Dorn, *University of Nebraska (Omaha)*
Phillip Eaglin, *ChangeExpectations.org*
Kathi Fisler, *Worcester Polytechnic Institute*
Jeff Forbes, *Duke University*
Joanna Goode, *University of Oregon*
Shuchi Grover, *SRI International*
Mark Guzdial, *Georgia Tech*
Helen Hu, *Westminster College*
Yasmin Kafai, *University of Pennsylvania*
Fred Martin, *University of Massachusetts (Lowell), CSTA board chair-elect*
Don Miller, *New York City Department of Education*
Tammy Pirmann, *CSTA board member, School District of Springfield Township (PA)*
Meg Ray, *Cornell Tech*
Dave Reed, *Creighton University, CSTA board chair*
Deborah Seehorn, *CSTA board past chair, standards co-chair*
Ben Shapiro, *University of Colorado (Boulder)*
Chinma Uche, *Greater Hartford Academy of Math and Science, CSTA board member*
Sheena Vaidyanathan, *Los Altos School District (CA), CSTA board member*
Uri Wilensky, *Northwestern University*
Aman Yadav, *Michigan State University, CSTA board member*

Revisión

Gracias a los cientos de individuos y organizaciones que brindaron retroalimentación y apoyo durante los tres períodos de revisión pública para el marco. Los grupos que convocaron las revisiones se enumeran en el **Apéndice A**.

Contribuciones Especiales

Gracias a Jennifer Childress de Achieve por su asesoramiento y consulta durante el desarrollo del marco.

Gracias a Heidi Schweingruber de la Junta de Educación en Ciencias de las Academias Nacionales de Ciencias, Ingeniería y Medicina y Thomas Keller de la Alianza de Matemáticas y Ciencias de Maine por compartir su experiencia en el desarrollo del Marco del Consejo Nacional de Investigación para la Educación en Ciencias K–12.

Además de desarrollar los conceptos y las prácticas del marco, los siguientes escritores contribuyeron también los capítulos de orientación: Derek Babb, Leigh Ann DeLyser, Caitlin McMunn Dooley, Maya Israel, Irene Lee y Shaileen Crawford Pokress. Gracias a Courtney K. Blackwell por los capítulos de educación e investigación durante la infancia.

Los siguientes asesores informales proporcionaron retroalimentación crítica durante el proceso de desarrollo del marco: Peter Denning, Escuela Naval de Posgrado; Alan Kay, Instituto de investigación de puntos de vista; Michael Lach, UCHicago STEM Educación en la Universidad de Chicago; y Chris Stephenson, Google.

Tabla de Contenidos

Figuras y Tablas

Figuras

Tablas

Resumen Ejecutivo

La influencia de la computación se siente a diario y se experimenta a nivel personal, social y global. Las ciencias de la computación, la disciplina que hace posible el uso de las computadoras, ha impulsado la innovación en todas las industrias y campos de estudio, desde la antropología hasta la zoología. Las ciencias de la computación también están impulsando enfoques para muchos de los desafíos más difíciles de nuestro mundo; algunos ejemplos incluyen la disminución de muertes en automóviles, la distribución de vacunas médicas y el suministro de plataformas para que los aldeanos rurales puedan participar en economías más grandes, entre otros.

Las ciencias de la computación están impulsando los enfoques de muchos de los desafíos más difíciles de nuestro mundo.

Dado que las ciencias de la computación se han convertido en una parte integral de nuestro mundo, la demanda pública de educación en ciencias de la computación es alta. La mayoría de los padres desean que la escuela de sus hijos ofrezca las ciencias de la computación (Google y Gallup, 2015), y la mayoría de los estadounidenses creen que las ciencias de la computación son tan importantes como la lectura, la escritura y las matemáticas (Horizon Media, 2015). Muchos de los estudiantes de hoy usarán las ciencias de la computación en sus carreras futuras, no solo en campos de ciencias, tecnología, ingeniería y matemáticas (STEM) sino también en campos que no son STEM (Change the Equation, 2015).

Desafortunadamente, la oportunidad de aprender las ciencias de la computación no coincide con la demanda pública. La mayoría de las escuelas de los EE. UU. no ofrecen ni un solo curso de las ciencias de la computación, tampoco de programación (Google y Gallup, 2015), y muchas de las clases existentes no son diversas, ni representan a nuestra población (College Board, 2016). Muchos estudiantes tienen que esperar hasta la escuela secundaria para aprender las ciencias de la computación, a pesar de que nacieron en una sociedad que depende de las ciencias de la computación y nunca han conocido un mundo sin ella. Aunque las computadoras están cada vez más disponibles para los estudiantes en las escuelas de nuestra nación, las oportunidades para aprender las ciencias de la computación no son accesibles para todos. Las agencias de educación estatales y locales han comenzado a adoptar políticas y desarrollar infraestructura clave para apoyar las ciencias de la computación para todos los estudiantes y han expresado un interés mutuo por la orientación en esta nueva frontera.

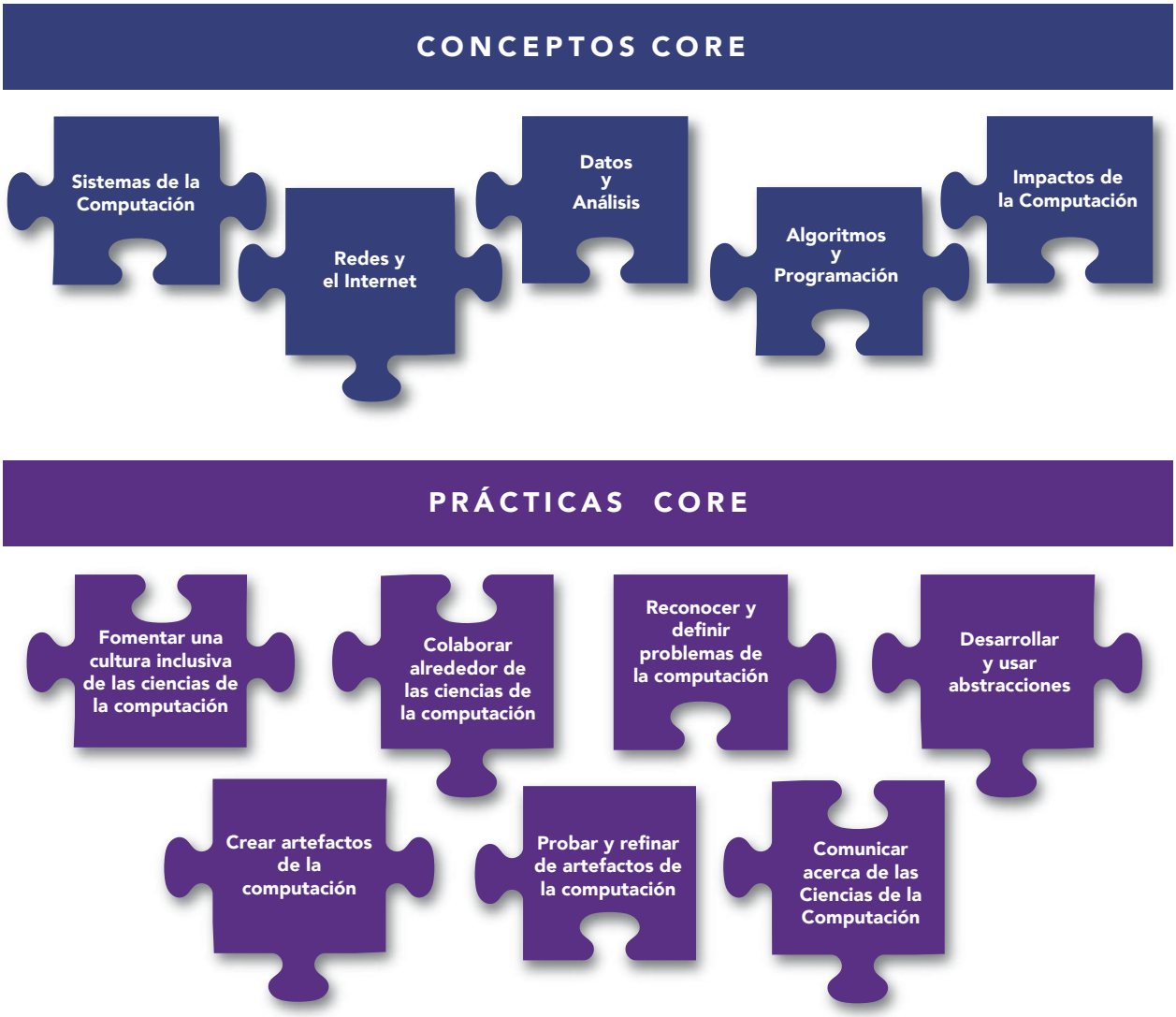
El marco de las ciencias de la computación K–12 informa los estándares al conocimiento de las ciencias de la computación.

La Asociación para la Maquinaria de Computación, Code.org, la Asociación de Profesores de las Ciencias de la Computación, el Centro de Innovación Cibernética y la Iniciativa Nacional de Matemáticas y Ciencia han respondido al llamado organizando estados, distritos y la comunidad de educación en ciencias de la computación para desarrollar pautas conceptuales para la educación de la computación. El marco de las ciencias de la computación K–12 se desarrolló para que los estados, distritos, escuelas y organizaciones informen sobre el desarrollo de estándares y planes de estudio, desarrollen capacidades para la enseñanza de las ciencias de la computación y las implementen por vías al conocimiento de las ciencias de la computación.

El marco promueve una visión en la que todos los estudiantes se involucran de manera crítica en temas de las ciencias de la computación; abordar los problemas de manera innovadora; y cree artefactos de la computación con una intención práctica, personal o social.

El desarrollo del marco fue un esfuerzo comunitario. Veintisiete escritores y veinticinco asesores desarrollaron el marco con comentarios de cientos de revisores, incluidos profesores, investigadores, profesores de educación superior, partes interesadas de la industria y educadores informales. El grupo de escritores y asesores representa a los estados y distritos de todo el país, así como a una variedad de perspectivas académicas y experiencias que trabajan con diversas poblaciones estudiantiles.

Figura 0.1: El marco de las ciencias de la computación K–12



El marco de las ciencias de la computación K–12 ilumina las grandes ideas de las ciencias de la computación a través de una lupa de conceptos (es decir, lo que los estudiantes deben saber) y las prácticas (es decir, lo que deben hacer los estudiantes).

Los conceptos core del marco representan las áreas principales de contenido en el campo de las ciencias de la computación. Las prácticas core representan los comportamientos que los estudiantes con conocimientos de computación utilizan para involucrarse completamente con los conceptos core de las ciencias de la computación.

El marco proporciona una visión unificadora para guiar las ciencias de la computación de un tema para los pocos afortunados a una oportunidad para todos.

Las progresiones de aprendizaje del marco describen cómo la comprensión conceptual y la práctica de las ciencias de la computación de los estudiantes se vuelven más sofisticadas con el tiempo. Los conceptos y las prácticas están diseñados para integrarse y brindar experiencias auténticas y significativas para los estudiantes que participan en ciencias de la computación (ver Figura 0.1)

Una serie de temas importantes se entrelazan en todo el marco. Incluyen:

- **Equidad**. Los temas de equidad, inclusión y diversidad se abordan en los conceptos y prácticas del marco, en recomendaciones para estándares y currículos, y en ejemplos de esfuerzos para ampliar la participación en la educación de las ciencias de la computación.

- **Ideas poderosas**. Los conceptos y prácticas del marco evocan ideas auténticas y poderosas que pueden usarse para resolver problemas del mundo real y conectar la comprensión a través de múltiples disciplinas (Papert, 2000).

- **Pensamiento computacional**. Las prácticas de pensamiento computacional, como la abstracción, el modelado y la descomposición, se entrecruzan con conceptos de las ciencias de la computación como los algoritmos, la automatización y la visualización de datos.

- **Amplitud de aplicación**. Las ciencias de la computación es más que la codificación. Se trata de sistemas físicos y redes; la recopilación, almacenamiento y análisis de datos; y el impacto de las ciencias de la computación en la sociedad. Esta visión amplia de las ciencias de la computación enfatiza la gama de aplicaciones que las ciencias de la computación tienen en otros campos.

Los capítulos del marco brindan orientación crítica a los estados, distritos y organizaciones en áreas clave de interés. Se proporcionan recomendaciones para guiar el desarrollo de estándares rigurosos y accesibles para todos los estudiantes. La guía para diseñar programas de currículo, evaluación, métodos de cursos, certificación y desarrollo docente informará la implementación de la visión del marco. Un capítulo sobre las ciencias de la computación en la educación durante la infancia describe cómo las ciencias de la computación se pueden integrar en el aula de preescolar al preservar, apoyar y mejorar el enfoque durante la infancia en el juego y el aprendizaje socioemocional. La investigación relevante en la que se basa el marco, las brechas en la literatura de investigación de educación en ciencias de la computación K-12 y las oportunidades de estudios adicionales se describen para informar futuras investigaciones y revisiones del marco. Un apéndice incluye un resumen de los comentarios del público enviados durante los períodos de revisión del marco y las revisiones posteriores realizadas por los escritores.

El marco de las ciencias de la computación K-12 se presenta en un momento en que los sistemas educativos de nuestra nación se están adaptando a una visión del siglo 21 de estudiantes que no solo son usuarios de computadoras, sino también creadores de las ciencias de la computación que dominan los conceptos y las prácticas de las ciencias de la computación. A medida que las ciencias de la computación de K–12 continúa ganando impulso, los estados, distritos y organizaciones pueden usar el marco para desarrollar estándares e implementar vías al conocimiento de las ciencias de la computación y estructurar el desarrollo profesional. El marco proporciona una visión unificadora para guiar las ciencias de la computación desde un tema para unos pocos afortunados hasta una oportunidad para todos.

Referencias

Change the Equation. (2015, December 7). The hidden half [Blog post]. Recuperado de http://changetheequation.org/blog/hidden-half

College Board. (2016). AP program participation and performance data 2015 [Data file]. Recuperado de https://research.collegeboard.org/programs/ap/data/participation/ap-2015

Google & Gallup. (2015). Searching for computer science: Access and barriers in U.S. K–12 education. Recuperado de http://g.co/cseduresearch

Horizon Media. (2015, October 5). Horizon Media study reveals Americans prioritize STEM subjects over the arts; science is "cool," coding is new literacy. PR Newswire. Recuperado de http://www.prnewswire.com/news-releases/horizon-medias-tudy-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. IBM Systems Journal, 39 (3/4), 720–729.

Una visión para las ciencias de la computación K-12

1

Una visión para las ciencias de la computación K-12

El marco de las ciencias de la computación K–2 representa una visión en la que todos los estudiantes participan en los conceptos y prácticas de las ciencias de la computación. Comenzando en los primeros grados y continuando hasta el grado 12, los estudiantes desarrollarán una base de conocimiento de las ciencias de la computación y aprenderán nuevos enfoques para la resolución de problemas que aprovechan el poder del pensamiento computacional para convertirse en usuarios y creadores de la tecnología de las ciencias de la computación. Al aplicar las ciencias de la computación como una herramienta para el aprendizaje y la expresión en una variedad de disciplinas e intereses, los estudiantes participarán activamente en un mundo cada vez más influenciado por la tecnología.

El poder de las ciencias de la computación

El poder de las computadoras se deriva de su capacidad para representar nuestra realidad física como un mundo virtual y su capacidad para seguir instrucciones con las que manipular ese mundo. Las ideas, imágenes e información se pueden traducir en bits de datos y procesarlas por las computadoras para crear aplicaciones, animaciones o autos autónomos. La variedad de instrucciones que puede seguir una computadora lo convierte en un motor de innovación que solo está limitado por nuestra imaginación. Cabe destacar que las computadoras pueden incluso seguir instrucciones sobre instrucciones en forma de lenguajes de programación.

Una computadora es un motor de innovación que está limitado solo por nuestra imaginación.

Las computadoras son máquinas rápidas, confiables y poderosas que nos permiten construir, analizar y comunicar digitalmente nuestra experiencia humana. Más que una herramienta, las computadoras son un medio de fácil acceso para la expresión

creativa y personal. En nuestra era digital, las computadoras son tanto la pintura como el pincel. La educación de las ciencias de la computación crea los artistas.

Las escuelas se han adherido a la promesa que ofrecen las computadoras: brindar instrucción, servir como herramienta de productividad y conectarse a una fuente de información cada vez mayor. Esta creencia de que las computadoras pueden mejorar la educación es evidente en la cantidad de iniciativas de dispositivos uno a uno que se ven en los distritos escolares de nuestro país. A pesar de la disponibilidad de computadoras en las escuelas, el aspecto más importante de las ciencias de la computación ha sido retenido por la mayoría de nuestros estudiantes: aprender a crear con computadoras (es decir, las ciencias de la computación).

La alfabetización proporciona un contexto relevante para comprender la necesidad de la educación en ciencias de la computación. Desde una edad temprana, a los estudiantes se les enseña a leer para que puedan influir en lo que se ha escrito, pero también a escribir para que puedan expresar ideas e influir en otros. Aunque la computación es un medio poderoso como la alfabetización, a la mayoría de los estudiantes solo se les enseña cómo usar (es decir, leer) los trabajos de computación que se les proporcionan, en lugar de crear (es decir, escribir) trabajos para ellos mismos. Juntos, los "autores" que han trabajado en el medio de las ciencias de la computación durante las últimas décadas han transformado nuestra sociedad. Aprender las ciencias de la computación permite a los estudiantes convertirse en autores y crear sus propios poemas e historias en forma de programas y software. En lugar de ser consumidores pasivos de las tecnologías informáticas, pueden convertirse en productores y creadores activos. En nuestra era digital, puede "programar o ser programado" (Rushkoff, 2011, p. 1).

En nuestra era digital, las computadoras son tanto la pintura como el pincel. La educación de las ciencias de la computación crea los artistas.

Una visión para las Ciencias de la Computación K-12

Desde el ábaco hasta los teléfonos inteligentes de hoy, desde el primer programa de computadora de Ada Lovelace hasta las ideas poderosas de Seymour Papert, las ciencias de la computación han cambiado nuestro mundo dramáticamente y promete ayudar a mejorar la educación. Las formas de pensar, resolver problemas y crear de las ciencias de la computación se han vuelto invaluables para todos los aspectos de la vida y son importantes más allá de asegurar que tengamos suficientes trabajadores de tecnología calificados. El marco de las ciencias de la computación K–12 prevé un futuro en el que los estudiantes son ciudadanos informados que pueden:

- participar de manera crítica en la discusión pública sobre temas de las ciencias de la computación;
- desarrollar como aprendices, usuarios y creadores de conocimiento y artefactos de las ciencias de la computación;
- comprender mejor el papel de las ciencias de la computación en el mundo que los rodea; y
- aprender, actuar y expresarse en otros temas e intereses.

Esta visión para la educación en ciencias de la computación se comprende mejor imaginando uno de los caminos que María, una estudiante, podría tomar durante su experiencia en K-12:

- En la escuela primaria, María aprende a instruir a las computadoras mediante la secuenciación de acciones como piezas de rompecabezas para crear algoritmos de computadora que dibujen diseños hermosos. Desde la edad joven, entiende que la computación es una experiencia creativa y una herramienta para la expresión personal.
- En la escuela secundaria, María se vuelve más sofisticada en su uso de los conceptos de computación y la comprensión de cómo funciona la computación. Ella usa la computadora, así como las ideas y procesos de la computación, para mejorar las experiencias de aprendizaje en otras disciplinas. La computación sirve como un medio para representar y resolver problemas.
- En la escuela secundaria, María ve oportunidades dentro de su comunidad y sociedad para aplicar las ciencias de la computación de maneras novedosas. Los conceptos y prácticas de las ciencias de la computación le han permitido crear cambios auténticos en pequeña y gran escala y en una amplia variedad de intereses.

Esta visión promete mejorar la experiencia K–12 de todos los estudiantes mientras los prepara para una amplia variedad de experiencias y carreras postsecundarias. Los estudiantes que se gradúan con una fundación de las ciencias de la computación K–12 pasarán a ser miembros de la sociedad con conocimientos de computación que no solo son consumidores de tecnología sino también creadores de la misma.

No solo consumidores de tecnología sino creadores.

Se convertirán en médicos, artistas, empresarios, científicos, periodistas y desarrolladores de software que impulsarán niveles de innovación aún mayores en estos y en una variedad de otros campos, beneficiando a sus comunidades y al mundo. El marco de las ciencias de la computación K–12 está dedicado a hacer que esta visión de la educación en ciencias de la Computación sea accesible para todos.

El caso de las ciencias de la computación

La transversalidad de la computación personal y nuestra creciente dependencia de la tecnología han cambiado el tejido de la sociedad y la vida cotidiana. Independientemente de su futura carrera, muchos estudiantes usarán las ciencias de la computación en el trabajo; según una estimación, más de 7.7 millones de estadounidenses utilizan las computadoras de manera compleja en sus trabajos, casi la mitad de ellas en campos que no están directamente relacionados con las ciencias, la tecnología, la ingeniería y las matemáticas (STEM) (Change the Equation, 2015).

Desafortunadamente, los estudiantes de K–12 de hoy tienen una oportunidad limitada de aprender sobre estos conceptos y prácticas de las ciencias de la computación y de entender cómo las ciencias de la computación influyen en su vida diaria. Cuando menos de la mitad de las escuelas imparten cursos significativos de las ciencias de la computación (Google y Gallup, 2015b), la enorme disparidad en el acceso a menudo margina a los estudiantes tradicionalmente insuficientemente representados, que ya enfrentan desigualdades educativas. Esta brecha de oportunidades se refleja en una alarmante falta de diversidad en la fuerza laboral de tecnología (por ejemplo, Information is Beautiful, 2015; Sullivan, 2014). La mayoría de las clases de las ciencias de la computación se ofrecen solo a estudiantes de secundaria, pero la investigación en otros campos de STEM ha demostrado repeti-

damente que los estereotipos (Scott y Martin, 2014) sobre quién es bueno o quién pertenece a esos campos se establecen desde una edad muy temprana. Dirigir estos mensajes antes y proporcionar un acceso más temprano a las experiencias de las ciencias de la computación puede ayudar a evitar que se formen estos estereotipos (Google, 2014). La participación temprana en ciencias de la computación también les permite a los estudiantes desarrollar fluidez con las ciencias de la computación durante muchos años (Guzdial, Ericson, McKlin y Engelman, 2012) y les brinda la oportunidad de aplicar las ciencias de la computación a otras materias e intereses mientras van a la escuela (Grover, 2014). La falta de oportunidades es particularmente desalentadora, dada la opinión pública y las estadísticas recientes de trabajo en ciencias de la computación:

- Los estadounidenses creen que las ciencias de la computación son tan importantes para aprender como la lectura, la escritura y las matemáticas (Horizon Media, 2015).
- La mayoría de los padres quieren que la escuela de sus hijos ofrezca las ciencias de la computación (Google y Gallup, 2015b).
- Desde 2010, las ciencias de la computación es una de las carreras universitarias de más rápido crecimiento en todos los campos STEM (Fisher, 2015), y las ciencias de la computación avanzada (AP®) es el examen del Emplazamiento Avanzado de mayor crecimiento, a pesar de que se ofrece en solo el 5% de las escuelas. (Code.org, 2015).
- Los trabajos que utilizan las ciencias de la computación son algunos de los empleos mejor pagados y con mayor crecimiento (Bureau of Labor Statistics, 2015), y los trabajos más demandados que sustentan la economía (The Conference Board, 2016).
- Las ciencias de la computación se define como parte de una "educación integral" en la Ley de que todos los estudiantes tienen éxito (2015).

¿Que son las ciencias de la computación?

La educación en ciencias de la computación en las escuelas K–12 incluye alfabetización informática, tecnología educativa, ciudadanía digital, tecnología de la información y las ciencias de la computación. Como la base de toda la informática, las ciencias de la computación son "el estudio de las computadoras y los procesos algorítmicos, incluidos sus principios, sus diseños de hardware y software, sus aplicaciones y su impacto en la sociedad" (Tucker et al., 2006, p. 2). El marco de las ciencias de la computación K–12 organiza este conjunto de conocimientos en cinco conceptos core que representan áreas de contenido clave en ciencias de la computación y siete prácticas que representan acciones que los estudiantes utilizan para interactuar con los conceptos de manera rica y significativa.

Las ciencias de la computación se confunden a menudo con el uso diario de las computadoras, como aprender a usar Internet y crear presentaciones digitales. Los padres, profesores, estudiantes y administradores locales y estatales pueden compartir esta confusión. Una encuesta reciente muestra que la mayoría de los estudiantes creen que crear documentos y presentaciones (78%) y buscar en Internet (57%) son actividades de las ciencias de la computación (Google y Gallup, 2015a). Los padres, los profesores y los directores son casi tan malos como para delinear la diferencia entre las actividades tradicionales de alfabetización informática y las ciencias de la computación, y en realidad, más padres que estudiantes creen que hacer una búsqueda en internet es usar las ciencias de la computación (Google & Gallup, 2015a). Esta confusión se extiende a los departamentos estatales de educación.

> Una encuesta de individuos responsables de las áreas de certificación del estado concluyó que muchos estados no parecían tener una definición o comprensión clara del campo de las "Ciencias de la Computación" y mostraron una tendencia a confundir las ciencias de las computadoras con otras áreas temáticas como: Educación tecnológica/Tecnología educativa (ET/TE), tecnología industrial o de instrucción (IT), sistemas de información de gestión (MIS) o incluso el uso de computadoras para apoyar el aprendizaje en otras áreas temáticas (Khoury, 2007, p. 9).

Estos conceptos erróneos sobre las ciencias de la computación plantean serios desafíos para ofrecer experiencias de las ciencias de la computación de alta calidad para todos los estudiantes. El marco de las ciencias de la computación K–12 aclara no solo que son las ciencias de la computación, sino también lo que los estudiantes deben saber y poder hacer en ciencias de la computación desde el jardín de infantes hasta el grado 12. Las ciencias de la computación se basan en la alfabetización informática, la tecnología educativa, la ciudadanía digital y la tecnología de la información. Sus diferencias y relación con las ciencias de la computación se describen a continuación.

- La alfabetización informática se refiere al uso general de computadoras y programas, como el software de productividad. Los ejemplos mencionados anteriormente incluyen realizar una búsqueda en Internet y crear una presentación digital.

- La tecnología educativa aplica la alfabetización informática a las materias escolares. Por ejemplo, los estudiantes en una clase de inglés pueden usar una aplicación basada en web para crear, editar y almacenar un ensayo en línea en colaboración.

- La ciudadanía digital se refiere al uso apropiado y responsable de la tecnología, como elegir una contraseña adecuada y mantenerla segura.
- La tecnología de la información a menudo se superpone con las ciencias de la computación, pero se enfoca principalmente en aplicaciones industriales de la informática, como instalar software en lugar de crearlo. Los profesionales de la tecnología de la información a menudo tienen experiencia en ciencias de la computación.

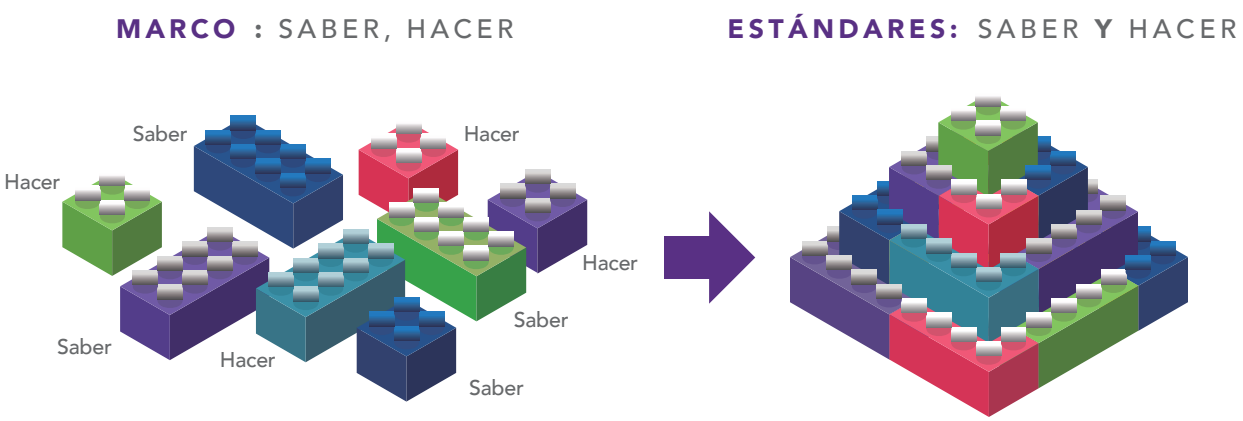
Estos aspectos de la informática se distinguen de las ciencias de la computación porque se centran en el uso de tecnologías informáticas en lugar de comprender por qué funcionan y cómo crear esas tecnologías. Saber por qué y cómo funcionan las computadoras (es decir, las ciencias de la computación), proporciona la base para una comprensión profunda del uso de la computadora y los derechos, responsabilidades y aplicaciones relevantes.

La seguridad de la contraseña es un tema que ilustra la intersección entre la informática y los otros aspectos de las ciencias de la computación. Un estudiante que sabe cómo programar una computadora para iterar sobre todas las palabras en una lista (es decir, una matriz) en una fracción de segundo es un estudiante que probablemente no use una palabra del diccionario para una contraseña. En este caso, entender por qué y cómo funcionan las computadoras, en última instancia, ayuda a los estudiantes a tomar buenas decisiones sobre su uso de las computadoras.

Alcance y Audiencia Deseada

Los conceptos y prácticas del marco de las ciencias de la computación K–12 no son expectativas de desempeño mensurable ni específicas en forma de estándares, ni se detallan planes de lecciones y actividades en forma de currículo. En cambio, el marco de las ciencias de la computación K–12 es una guía de alto nivel que los estados, distritos y organizaciones pueden usar para informar el desarrollo de sus propios estándares y currículos. Como se ilustra en la Figura 1.1, el marco proporciona bloques de construcción de conceptos (que los estudiantes deben conocer) y prácticas (que los estudiantes deben hacer) que se pueden usar para crear estándares (expectativas de rendimiento de lo que los estudiantes deben saber y hacer).

Figura 1.1: Bloques de construcción para estándares



También debe quedar claro que el marco no proporciona el alcance completo del contenido informático para temas avanzados de estudio. El marco describe una alfabetización de referencia para todos los estudiantes, por lo que aquellos que eligen estudiar las ciencias de la computación más a fondo pueden buscar honores, AP (Emplazamiento Avanzado) o cursos especializados en programas de educación técnica y profesional que incluyan contenido más allá del marco.

El marco no prescribe expectativas para cursos específicos. No proporciona resultados específicos de nivel de grado, ni define la estructura del curso (el alcance y la secuencia de temas en un curso en particular) o los métodos de los cursos (el alcance de los temas y la secuencia en múltiples cursos). Los cinco conceptos core del marco no fueron diseñados para servir como unidades independientes en un curso o temas separados que definen cursos completos; en cambio, los conceptos y las prácticas del marco deben integrarse a lo largo de la instrucción.

Los conceptos y prácticas del marco deben integrarse a lo largo de la instrucción.

El marco fue escrito para una audiencia con diversos orígenes, incluidos educadores que están aprendiendo a enseñar las ciencias de la computación. Esta audiencia incluye:

- políticos y administradores responsables del estado/distrito;
- desarrolladores de estándares y planes de estudio (con suficiente experiencia en ciencias de la computación).
- profesores actuales y nuevos de las ciencias de la computación, incluidos profesores de otras áreas temáticas y educadores en entornos informales; y
- organizaciones de apoyo (organizaciones sin fines de lucro, socios de la industria y educación informal).

Principios que Guían el Marco

Los siguientes principios guiaron el desarrollo del marco:

1. Ampliar la participación en ciencias de la computación.
2. Centrarse en lo esencial.
3. No reinventar la rueda.
4. Informar con la investigación actual y orientar la investigación futura.
5. Alinear a marcos reconocidos a nivel nacional.
6. Inspirar la implementación.

Ampliar la Participación en Ciencias de la Computación:

En primer lugar, el marco de las ciencias de la computación K–12 está diseñado para todos los estudiantes, independientemente de su edad, raza, género, discapacidad o estado socioeconómico. La estructura y el contenido del marco reflejan la necesidad de diversidad en computación y atención a cuestiones de equidad, incluida la accesibilidad. La elección de los Impactos de la Computación como uno de los conceptos core y el Fomento de una Cultura de las Ciencias de la Computación Inclusiva como una de las prácticas core hacen que la diversidad, la equidad y la accesibilidad sean temas clave de estudio, además de entrelazarlos a través de otros conceptos y prácticas.

Enfocarse en lo Esencial:

El marco de las ciencias de la computación K–12 describe una alfabetización fundamental en ciencias de la computación, en lugar de una lista exhaustiva de todos los temas de las ciencias de la computación que se pueden aprender dentro de una ruta K–12. Aunque el marco describe que las ciencias de la computación son esencial para todos los estudiantes, se alienta a los educadores y desarrolladores de currículos a crear una experiencia de aprendizaje que se extienda más allá del marco para abarcar los intereses, habilidades y aspiraciones de los alumnos. Además, el marco intenta utilizar un lenguaje sencillo y sin jerga que sea accesible a los instructores y al público en general. Cuando se usan términos técnicos, se consideran necesarios para mantenerse fieles al vocabulario disciplinario e ilustrar completamente los conceptos relevantes.

No Reinventes la Rueda:

El marco de las ciencias de la computación K-12 se basa en una historia de investigación y práctica profesional en educación de las ciencias de la computación. El marco está influenciado por el trabajo de organizaciones profesionales como la Asociación de Profesores de las Ciencias de la Computación (CSTA Standards Task Force, 2011) y marcos de educación en matemáticas, ciencias y tecnología (por ejemplo, ISTE, 2016). Los marcos de cursos reconocidos a nivel nacional como el marco curricular de los Principios de las Ciencias de la Computación de Emplazamiento Avanzada (College Board, 2016) y las pautas curriculares de la Association for Computing Machinery para programas de pregrado de las ciencias de la computación brindaron una visión para los estudiantes que pueden continuar con estudios avanzados de las ciencias de la computación. Marcos de las Ciencias de la Computación de otros países: el Reino Unido (Departamento de Educación de Inglaterra, 2013), Alemania (Hubwieser, 2013), Polonia (Sysło y Kwiatkowska, 2015) y Nueva Zelanda (Bell, Andreae y Robins, 2014) Se utiliza para comparar los conceptos y prácticas del marco.

Informar con la Investigación Actual y la Guía de Investigación Futura:

El marco refleja la investigación actual en educación de las ciencias de la computación, incluidas las progresiones de aprendizaje, las trayectorias y el pensamiento computacional. Donde falta la investigación específica en educación en ciencias de la computación, el marco se basa en la base de conocimientos existente de la comunidad de profesionales y en la investigación de otras áreas de contenido relacionadas para guiar decisiones como la adecuación del desarrollo de conceptos particulares. Las preguntas restantes han guiado una agenda de investigación que informará futuras revisiones del marco.

Alinear a Marcos Reconocidos Nacionalmente:

El desarrollo de un marco para la educación en ciencias de la computación implica tanto definir un tema nuevo para la mayoría de las escuelas como confiar en las estructuras y procesos establecidos que se utilizan en el desarrollo de otras pautas de educación. Debido a que este marco existirá junto con los de otras asignaturas, el marco de las ciencias de la computación K–12 está intencionalmente estructurado de manera similar a otros marcos, como el marco para la educación en ciencias K–12 (NRC, 2012). El uso de una lupa de conceptos y prácticas para ver y describir las ciencias de la computación K-12 proporciona una mayor coherencia en todas las áreas temáticas. El marco de las ciencias de la computación K–12 también reflejó el proceso de desarrollo de otros esfuerzos impulsados por la comunidad. La transparencia y la inclusión se hicieron hincapié en todo el proceso de desarrollo a través de resúmenes públicos, actualizaciones mensuales, foros/webinars, conversaciones con las partes interesadas, talleres de asesores, vistas previas de la comunidad y períodos de revisión pública. En el **Apéndice A** se puede encontrar un resumen de los comentarios del público y las subsiguientes revisiones del marco.

Implementar Inspiración:

Ya sea que un estado o distrito ya esté implementando las ciencias de la computación para todos los estudiantes, o recién haya comenzado, el Marco de Ciencias de la Computación K–12 brinda una visión coherente para inspirar esfuerzos adicionales. El marco contiene capítulos que brindan orientación sobre una variedad de pasos clave de implementación, tales como el desarrollo de estándares, la preparación de profesores y la creación de un currículo que refleje los conceptos y prácticas del marco. La política y la implementación deben ir de la mano para brindar oportunidades de las ciencias de la computación de alta calidad para todos los estudiantes.

Resumen

El objetivo de este proyecto ha sido proporcionar un marco de alto nivel para la educación en ciencias de la computación K-12 mediante la identificación de los conceptos y prácticas core de las ciencias de la computación y la descripción de cómo se ven esos conceptos y prácticas para los estudiantes en varias bandas de grado. El marco proporciona orientación a los estados, distritos y organizaciones que desean diseñar sus propios estándares, currículo, evaluaciones o programas de preparación de profesores. La educación en ciencias de la computación es un campo en evolución con un creciente organismo de investigación a nivel K–12 y muchas lecciones que aprender a medida que los sistemas educativos toman medidas para aumentar las oportunidades en ciencias de la computación. La comunidad que ha desarrollado y apoyado este proyecto cree que el marco de las ciencias de la computación K–12 es un paso inicial para informar, inspirar y dirigir el trabajo de implementación requerido para hacer que la visión del marco sea una realidad: las ciencias de la computación para todos los estudiantes.

Referencias

Bell, T., Andreae, P., & Robins, A. (2014). A case study of the introduction of computer science in NZ schools. *ACM Transactions on Computing Education (TOCE), 14*(10), 10:1–10:31. doi: 10.1145/2602485

Bureau of Labor Statistics. (2015). *Employment projections* [Data file]. Retrieved from http://www.bls.gov/emp/tables.htm

Change the Equation. (2015, December 7). The hidden half [Blog post]. Retrieved from http://changetheequation.org/blog/hidden-half

Code.org. (2015, July 2). Computer science is the fastest growing AP course of the 2010s [Blog post]. Retrieved from http://blog.code.org/post/123032125688/apcs-2015

College Board. (2016). *AP Computer Science Principles course and exam description*. New York, NY: College Board. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

Computer Science Teachers Association Standards Task Force. (2011). *CSTA K–12 computer science standards, revised 2011*. New York, NY: Computer Science Teachers Association and Association for Computing Machinery. Retrieved from http://www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf

The Conference Board. (2016). *National demand rate and OES employment data by occupation* [Data file]. Retrieved from https://www.conference-board.org/

England Department for Education. (2013, September 11). National curriculum in England: Computing programmes of study. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Every Student Succeeds Act of 2015, Pub. L. No. 114-95. 20 U.S.C.A. 6301 (2016).

Fisher, A. (2015, February 10). The fastest-growing STEM major in the U.S. *Fortune*. Retrieved from http://fortune.com/2015/02/10/college-major-statistics-fastest-growing/

Google. (2014). *Women who choose computer science—what really matters: The critical role of encouragement and exposure*. Mountain View: CA. Retrieved from https://www.google.com/edu/resources/computerscience/research/

Google & Gallup. (2015a). *Images of computer science: Perceptions among students, parents, and educators in the U.S.* Retrieved from http://g.co/cseduresearch

Google & Gallup. (2015b). *Searching for computer science: Access and barriers in U.S. K–12 education*. Retrieved from http://g.co/cseduresearch

Grover, S. (2014). *Foundations for advancing computational thinking: Balanced designs for deeper learning in an online computer science course for middle school students* (Doctoral dissertation). Stanford University, CA.

Guzdial, M., Ericson, B. J., McKlin, T., & Engelman, S. (2012). A statewide survey on computing education pathways and influences: Factors in broadening participation in computing. *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (pp. 143–150). New York, NY. doi: 10.1145/2361276.2361304

Horizon Media. (2015, October 5). Horizon Media study reveals Americans prioritize STEM subjects over the arts; science is "cool," coding is new literacy. *PR Newswire*. Retrieved from http://www.prnewswire.com/news-releases/horizon-media-study-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html

Hubwieser, P. (2013). The Darmstadt model: A first step towards a research framework for computer science education in schools. In I. Diethelm & R. T. Mittermeir (Eds.), *Informatics in Schools: Sustainable Informatics Education for Pupils of All Ages. Proceedings of the 6th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 13), Oldenburg, Germany* (pp. 1–14). doi: 10.1007/978-3-642-36617-8_1

Information is Beautiful. (2015). *Diversity in tech: Employee breakdown of key technology companies*. Retrieved from http://www.informationisbeautiful.net/visualizations/diversity-in-tech/

International Society for Technology in Education. (2016). *ISTE standards for students.* Retrieved from https://www.iste.org/resources/product?id=3879&childProduct=3848

Khoury, G. (2007). *Computer science state certification requirements. CSTA Certification Committee report.* Retrieved from the Computer Science Teachers Association website: https://csta.acm.org/ComputerScienceTeacherCertification/sub/TeachCertRept07New.pdf

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

Rushkoff, D. (2010). *Program or be programmed: Ten commands for a digital age.* New York, NY: OR Books.

Scott, A., & Martin, A. (2014). Perceived barriers to higher education in science, technology, engineering, and mathematics. *Journal of Women and Minorities in Science and Engineering, 20*(3), 235–256. doi: 10.1615/JWomenMinorScienEng.2014006999

Sullivan, G. (2014, May 29). Google statistics show Silicon Valley has a diversity problem. *The Washington Post.* Retrieved from https://www.washingtonpost.com/news/morning-mix/wp/2014/05/29/most-google-employees-are-white-men-where-are-allthewomen/

Sysło, M. M., & Kwiatkowska, A. B. (2015). Introducing a new computer science curriculum for all school levels in Poland. In A. Brodnik & J. Vahrenhold (Eds.), *Informatics in Schools: Curricula, Competences, and Competitions. Proceedings of the 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP 2015), Ljubljana, Slovenia (pp. 141–154). doi: 10.1007/978-3-319-25396-1_13

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

Equidad en la Educación de las Ciencias de la Computación

2

Equidad en la Educación de las Ciencias de la Computación

Las ciencias de la computación para todos los estudiantes requieren que la equidad esté a la vanguardia de cualquier esfuerzo de reforma, ya sea en el nivel de política de un marco o en el nivel escolar de instrucción y cultura del aula. Cuando existe equidad, hay apoyos apropiados basados en las necesidades individuales de los estudiantes para que todos tengan la oportunidad de lograr niveles similares de éxito. Inherente a esta meta es una expectativa integral de éxito académico que es accesible y se aplica a cada estudiante. El propósito de la equidad en ciencias de la computación no es preparar a todos los estudiantes para que se especialicen en ciencias de la computación y seguir carreras en ingeniería o tecnología de software. En cambio, se trata de asegurar que todos los estudiantes tengan el conocimiento fundamental que les permita participar de manera productiva en el mundo de hoy y tomar decisiones informadas sobre sus vidas. La equidad no se trata solo de si las clases están disponibles, sino también de cómo se imparten esas clases, cómo se reclutan los estudiantes y cómo la cultura del aula apoya a los diversos alumnos y promueve la permanencia. El resultado de la equidad es un aula diversa de estudiantes, basada en factores como raza, género, discapacidad, estatus socioeconómico y dominio del idioma inglés, todos los cuales tienen altas expectativas y se sienten capacitados para aprender. Las ciencias de la computación es más que una disciplina para unos pocos elegidos; Es una alfabetización esencial del siglo XXI para todos los estudiantes.

La equidad no se trata solo de que hay clases disponibles, pero también acerca de cómo esas clases se enseñan, cómo los alumnos son reclutados y cómo la cultura del aula apoya a diversos estudiantes y promueve la permanencia.

Este capítulo revisa los problemas de equidad relacionados con las ciencias de la computación, describe cómo estos problemas han influido en el desarrollo del marco y ofrece ejemplos breves de los esfuerzos actuales para promover la equidad en la educación en ciencias de la computación. Si bien, la equidad es el enfoque de este capítulo, también sirve como un tema importante que conecta todos los aspectos del marco: los conceptos y prácticas principales, los capítulos de orientación e incluso el proceso de desarrollo. Las recomendaciones adicionales para promover la equidad se pueden encontrar en los siguientes capítulos: **Guía para Desarrolladores de Estándares, Guía de Implementación** y **El Papel de la Investigación en el Desarrollo y Futuro del Marco.**

La necesidad de abordar la equidad en ciencias de la computación

Si bien, las preocupaciones sobre la equidad prevalecen en la educación en ciencias de tecnología, ingeniería y matemáticas (STEM), las ciencias de la computación enfrentan desafíos intensos en términos de acceso, oportunidades y cultura. Un estudio realizado por Google y Gallup (2015b) mostró que menos de la mitad de las escuelas K–12 ofrecen cursos significativos de las ciencias de la computación que incluyen programación. Un análisis de la encuesta de Evaluación Nacional del Progreso Educativo (NAEP) de 2015 mostró que solo el 44% de los alumnos de grado 12 asisten a escuelas secundarias que ofrecen cursos de las ciencias de la computación (Change the Equation, 2016). Los mismos datos de NAEP revelaron que los estudiantes con menos acceso son nativos americanos, negros y latinos; de fondos de ingresos más bajos; y de las zonas rurales.

Estas encuestas representan los mejores escenarios posibles según, Google y Gallup (2015a), "muchos estudiantes, padres, profesores y administradores escolares no distinguen correctamente entre las actividades de las ciencias de la computación y la alfabetización general de computadoras" (p. 3). Es posible que muchos de los encuestados estén confundiendo las ciencias de la computación con la alfabetización informática, disminuyendo el porcentaje informado de cursos reales de las ciencias de la computación. En todo el país, no existe una definición común de las ciencias de la computación, y a menudo, se combina con actividades de alfabetización informática, como buscar

buscar en Internet y crear documentos o presentaciones en la computadora (Google y Gallup, 2015a) referencia del año. Es posible que los estudiantes no sepan nada mejor cuando se les ubica en los llamados cursos de computación que tienen más que ver con aprender a escribir que aprender las ciencias de la computación (Margolis, Estrella, Goode, Holme y Nao, 2010). Según Margolis et al. (2012), "Especialmente en las escuelas con un gran número de estudiantes afroamericanos y latinos, las clases de computación comúnmente solo ofrecen habilidades básicas y rudimentarias para el usuario en lugar de involucrar a los alumnos con las prácticas de resolución de problemas y pensamiento computacional que son la base de las ciencias de la computación" (p. 73).

Incluso cuando los cursos de las ciencias de la computación están disponibles, hay grandes brechas en la participación. Para el examen de Ciencias de la Computación de Emplazamiento Avanzada (AP®) de 2015, solo el 21.9% de los estudiantes eran mujeres, la peor tasa de participación femenina de todos los exámenes AP (College Board, 2016). Solo el 3.9% era negro o afroamericano, el 9% era hispano o latino y el 0.4% era indio americano. Las escuelas a menudo cargan con la culpa de las desigualdades en la educación, pero los estereotipos y las representaciones de los informáticos pueden jugar un papel importante en la forma en que los estudiantes se sienten sobre el tema. Incluso cuando los cursos de las ciencias de la computación están disponibles, hay grandes brechas en la participación. Para el examen de las Ciencias de la Computación de Emplazamiento Avanzado (AP®) de 2015, solo el 21.9% de los estudiantes eran mujeres, la peor tasa de participación femenina de todos los exámenes AP (College Board, 2016). Solo el 3.9% era negro o afroamericano, el 9% era hispano o latino y el 0.4% era indio americano. Las escuelas a menudo cargan con la culpa de las desigualdades en la educación, pero los estereotipos y las representaciones de los medios de comunicación de los científicos de la computación pueden jugar un papel importante en la forma en que los estudiantes se sienten sobre el tema. Los padres y los estudiantes informan que las personas que hacen ciencias de la computación en el cine o la televisión son, en su mayoría, hombres, blancos o asiáticos, y llevan gafas (Google & Gallup, 2015a). Las hembras informan que tienen menos probabilidades de aprender las ciencias de la computación, tienen menos confianza en aprenderlas y tienen menos probabilidades de necesitar saber las ciencias de la computación en su futura carrera (Google y Gallup, 2015a).

Las mujeres que estudian las Ciencias de la Computación AP tienen 10 veces más posibilidades de especializar en ciencias de la computación.

La falta de acceso y participación en el nivel K–12 tiene un efecto claro sobre la representación en ciencias de la computación después de K–12. En 2015, solo el 24.7% de los empleados en ocupaciones de computación y matemáticas eran mujeres, 8.6% negros o afroamericanos, y 6.8% hispanos o latinos (Bureau of Labor Statistics, 2015b). Al observar específicamente a los estudiantes que se gradúan con una licenciatura en ciencias de la computación, solo el 17% de los estudiantes son mujeres, el 8% son negros y el 9% son hispanos (NCES, 2014). La proporción de mujeres con títulos de licenciatura en ciencias de la computación ha disminuido de 1993 a 2012 (NSF, 2015). Afortunadamente, hay signos positivos de que el aprendizaje de las ciencias de la computación en la escuela secundaria se correlaciona con una mayor probabilidad de continuar con las ciencias de la computación en estudios postsecundarios. Los estudiantes que cursan AP de las Ciencias de la Computación en la escuela secundaria tienen seis veces más probabilidades de graduarse en ciencias de la computación que los estudiantes que no tomaron la AP de las Ciencias de la Computación en la escuela secundaria. Las mujeres tienen 10 veces más posibilidades, los estudiantes afroamericanos tienen 7 veces más posibilidades y los estudiantes hispanos tienen 8.5 veces más posibilidades (Morgan y Klaric, 2007).[1]

1 Compare esto con la tasa de participación general de las minorías poco representadas. En 2013, por ejemplo, el 9.2% de los examinados de AP fueron negros o afroamericanos, 18.8% hispanos o latinos y 0.6% indios americanos (College Board, 2014). Advanced Placement® es una marca registrada y/o propiedad de College Board, que no participó en la producción de este producto y no lo respalda.

La opinión pública y el papel de las ciencias de la computación en nuestra economía solo ponen de relieve la disparidad entre el acceso y los beneficios de una educación en ciencias de la computación. La mayoría de los estadounidenses creen que las ciencias de la computación son tan importantes como la lectura, la escritura y las matemáticas (Horizon Media, 2015), y la mayoría de los padres quieren que la escuela de sus hijos ofrezca las ciencias de la computación (Google y Gallup, 2015b).

Las personas que trabajan en ciencias de la computación tienen la oportunidad de trabajar en proyectos divertidos y emocionantes y hacer cosas que ayuden a mejorar la vida de las personas.

Equidad en el Marco

Las ciencias de la computación es una disciplina al servicio de la sociedad, su gente y sus necesidades. Como tal, la equidad, la inclusión y la diversidad son factores críticos en todos los aspectos de las ciencias de la computación. Al configurar un equipo, comprender los beneficios de la inclusión y la diversidad motiva a los estudiantes a buscar activamente a los colaboradores con diferentes perspectivas y antecedentes. Al diseñar una aplicación para teléfonos inteligentes, una preocupación por la equidad motiva a los equipos a pensar en cómo diseñar la interfaz de usuario para aquellos con discapacidades visuales. Estas son solo algunas de las ideas ilustradas en los conceptos y prácticas del marco. El objetivo de promover la diversidad conformó todo el proceso de desarrollo del marco y comenzó con la composición del equipo de personas que desarrollaban el marco.

Los escritores y asesores fueron demográficamente diversos según el género, la raza, el origen étnico, la representación institucional (agencia estatal/distrital, organizaciones sin fines de lucro, investigación, industria, escuelas K–12), así como las poblaciones con las que trabajan o estudian. Los escritores y asesores representaron una gama completa de experiencias educativas, desde la escuela primaria hasta la educación superior; entornos de educación informal y formal; escuelas privadas y públicas; y localidades rurales, urbanas y suburbanas. Por ejemplo, varios escritores y asesores tenían experiencia trabajando con estudiantes nativos americanos en sus territorios soberanos dentro de los EEUU. Si bien la gran mayoría de los escritores enseñaron o estaban enseñando actualmente las ciencias de la computación, también tenían experiencia en una variedad de temas, incluidos otros temas de STEM, así como en humanidades como artes del lenguaje y estudios sociales.

El equipo de redacción incluya profesores cuyas enfoque profesional fue estudiantes con discapacidades cognitivas y físicas y estudiantes en riesgo de fracaso académico.

El equipo de redacción incluyó investigadores de educación especial y profesores cuyo enfoque profesional fueron los estudiantes con discapacidades cognitivas y físicas y los estudiantes en riesgo de fracaso académico. Lea las biografías de los escritores en el **Apéndice B**.

Múltiples elementos relacionados con la visión, la estructura y el contenido del marco enfatizan que las ciencias de la computación es una disciplina en la que todos los estudiantes pueden participar y demostrar competencia. Las siguientes secciones describen las formas en que se consideró la equidad al determinar las ideas esenciales del marco, la amplitud de los conceptos, las progresiones de aprendizaje en los conceptos y las prácticas de las ciencias de la computación.

Ideas Esenciales

El marco describe una base integral y esencial en ciencias de la computación de la que todos los estudiantes pueden beneficiarse, independientemente de si continuarán o no en la educación postsecundaria en ciencias de la computación o en una carrera en ciencias de la computación. Las ciencias de la computación que se describe en el marco no son solo para estudiantes "dotados" o "honrados", sino para todos los estudiantes, incluidos aquellos que pueden y deben superar las expectativas del marco. La significación e importancia de cada concepto y práctica fue evaluada por escritores, asesores y revisores, con una consideración constante de una población estudiantil diversa. Solo las ideas consideradas esenciales se incorporaron a los conceptos y prácticas del marco. Además, se analizaron las descripciones de conceptos y prácticas para asegurarse de que el lenguaje no estuviera sesgado en función del género, la cultura o la discapacidad.

Conceptos Amplios

En lugar de ser ideas prescriptivas y limitadas en ciencias de la computación, las declaraciones del marco son conceptuales y de alto nivel. El marco no especifica la cantidad de tiempo de instrucción para cada concepto o el orden en que deben abordarse los conceptos. En cambio, la naturaleza conceptual del marco permite amplias posibilidades de implementación, incluyendo en algunos casos, integración y aplicación dentro de otros temas. Las escuelas que pueden tener problemas para adaptar una materia adicional a la jornada escolar pueden integrar los conceptos del marco en las ofertas de cursos actuales. Esta integración es especialmente beneficiosa para los estudiantes que merecen experiencias en ciencias de la computación, pero que requieren tiempo de instrucción en las materias básicas tradicionales. Un gráfico que describe la intersección entre las prácticas en matemáticas, las ciencias y las ciencias de la computación está disponible en el capítulo Prácticas del marco (ver Figura 5.2).

Progresiones de Aprendizaje Completas

Los conceptos core en el marco se dividen en subconceptos más específicos, que proporcionan puntos focales para el desarrollo de progresiones de aprendizaje completas desde el jardín de infantes hasta el grado 12. Esta atención a la coherencia y articulación entre las bandas de grado significa que el marco refleja todas las etapas clave en una progresión de aprendizaje. Las progresiones de aprendizaje incompletas requieren oportunidades adicionales fuera de la escuela para llenar los vacíos en el conocimiento, poniendo a los estudiantes sin estas experiencias en desventaja.

Prácticas de las ciencias de la computación

Las prácticas de las ciencias de la computación se centran en hacer las ciencias de la computación, y además de conocerlas, brindar variedades de oportunidades para que diferentes tipos de estudiantes participen en condiciones de igualdad con otros estudiantes. Esto es especialmente cierto para los aprendices del idioma inglés que, de otro modo, podrían tener dificultades para demostrar el conocimiento conceptual de formas tradicionales debido a su capacidad limitada de inglés, ya que pueden demostrar su comprensión de otras maneras. Adicionalmente, el uso de las prácticas también facilita grandes oportunidades de aprendizaje que pueden reforzar la adquisición del lenguaje. Por ejemplo, las prácticas como Comunicar acerca de la computación describen la expectativa de que todos los estudiantes intercambien ideas con diversas audiencias de diversas maneras. En el proceso de colaboración en torno a la computación, los estudiantes solicitan y proporcionan comentarios de los miembros del equipo. Estas interacciones les permiten a los estudiantes del idioma inglés desarrollar su dominio del idioma mientras participan en actividades de las ciencias de la computación auténticas.

Ejemplos de equidad en concepto y declaraciones de práctica

Las siguientes secciones incluyen ejemplos específicos de conceptos y prácticas que abordan la equidad, fomentan la diversidad y fomentan la inclusión. Estos son solo algunos ejemplos de las formas en que se abordan la equidad, la diversidad y la inclusión en las declaraciones de concepto y práctica; hay muchos otros.

Ejemplos: Entrelazar la equidad entre los conceptos

La equidad se entrelaza a lo largo de las declaraciones del concepto. Algunos ejemplos se encuentran a continuación:

La equidad es particularmente evidente en el concepto core de los impactos de la computación, como lo indica la descripción general, "Una persona informada y responsable debe entender las implicancias sociales del mundo digital, incluida la equidad y el acceso a la computación". Las declaraciones de conceptos específicos tratan directamente la equidad. En los grados 9-12, los estudiantes deben entender que “el diseño y uso de tecnologías y artefactos computacionales pueden mejorar, empeorar o mantener un acceso desigual a la información y las oportunidades” (9–12. Impactos de la cultura de la computación). La descripción de esta declaración de concepto elabora sobre cuestiones de equidad y acceso:

> Si bien muchas personas tienen acceso directo a la informática a lo largo del día, muchas otras aún no cuentan con suficiente atención o simplemente no tienen acceso. Algunos de estos desafíos están relacionados con el diseño de tecnologías de computación, como en el caso de tecnologías que son difíciles de usar para personas mayores y personas con discapacidades físicas. Otros déficits de capital, como la exposición mínima a las ciencias de la computación, el acceso a la educación y las oportunidades de capacitación, están relacionados con problemas sistémicos más grandes en la sociedad. (9-12. Impactos de la cultura de la computación).

Los vínculos con la equidad, la inclusión y la diversidad también se encuentran en otras áreas conceptuales. Por ejemplo, en los grados 6–8 bajo el concepto central de Algoritmos y Programación, los estudiantes deben entender que el diseño de soluciones implica “considerar cuidadosamente las diversas necesidades y deseos de la comunidad” (6–8. Algoritmos y Programación/Desarrollo de Programas).

La descripción proporciona un ejemplo de cómo un equipo que emplea un diseño centrado en el usuario puede desarrollar una aplicación que traduce la pronunciación difícil de entender de personas con dificultades del habla a un lenguaje comprensible. En los grados 9-12, los estudiantes aprenden que "los equipos diversos pueden desarrollar programas con un impacto amplio a través de una revisión cuidadosa y aprovechando las fortalezas de los miembros en diferentes roles" (9-12. Algoritmos y Programación/Desarrollo de Programas).

Ejemplos: Entrelazar la Equidad Dentro las Prácticas

Las consideraciones de equidad se incluyen en muchas de las prácticas, pero es el enfoque principal de la primera práctica central.

La práctica de fomentar una cultura de computación inclusiva está dedicada a la equidad, la inclusión y la diversidad. Las declaraciones de práctica asociadas se encuentran a continuación:

Al finalizar el grado 12, los estudiantes deben poder:

1. **Incluir las perspectivas únicas de los demás** y reflexionar sobre las propias perspectivas al diseñar y desarrollar productos de la computación.
2. **Atender las necesidades de los diversos usuarios** finales durante el proceso de diseño para producir artefactos con amplia accesibilidad y facilidad de uso.
3. **Emplear la autogestión y defensa entre pares** para abordar el sesgo en las interacciones, el diseño del producto y los métodos de desarrollo.

Estas declaraciones de práctica enfatizan el papel que la equidad, la inclusión y la diversidad juegan en el diseño de artefactos de la computación. La práctica de colaboración en torno a la computación también describe formas de promover la inclusión en ciencias de la computación. La primera declaración de práctica dice: "Cultive relaciones de trabajo con personas que posean perspectivas, habilidades y personalidades diversas". Para cumplir con esta recomendación, los estudiantes aprenden estrategias para incluir las ideas de todos los miembros del equipo, como tratar de extraer las opiniones de los colaboradores más tranquilos. Otras instancias en la práctica, descripciones generales, declaraciones y progresiones atienden inquietudes sobre equidad, diversidad e inclusión. Estas instancias enfatizan la necesidad de practicar la equidad y la inclusión al hacer las ciencias de la computación, como crear una aplicación o un robot, para beneficiarse de diversos colaboradores y la atención a diversos usuarios.

Esfuerzos para aumentar el acceso y la oportunidad

La investigación de la educación en ciencias de la computación ha descrito cómo la inequidad puede perpetuarse en pequeña escala, como en un aula, así como en gran escala, como en la disponibilidad y programación de cursos. Las diferencias en las oportunidades de aprendizaje a menudo se ven en la línea de género, raza, discapacidad y estatus socioeconómico y se reflejan en un acceso reducido a los recursos, como amplios laboratorios de computación

o profesores adecuadamente capacitados (Ladner e Israel, 2016; Margolis et al., 2010). El papel de las estructuras escolares y los sistemas de creencias de los educadores para limitar el acceso, el reclutamiento y la permanencia de estudiantes afroamericanos y latinos en Los Ángeles están bien documentados por Margolis y asociados (2010) en Stuck in the Shallow End. En un artículo en ACM Inroads (2012), Margolis et al, expliquen , "Las ofertas de cursos (o la falta de ellos), la ubicación fuera del plan de estudios académicos principales, la falta de preparación docente y los recursos de instrucción, así como el clima educativo de la responsabilidad escolar exigidos por la legislación estatal y federal, dan como resultado amplias disparidades y falta de disponibilidad y calidad de las oportunidades de educación en CC para los estudiantes de color (p. 73)." Las desigualdades también se pueden propagar cuando los programas se amplían para satisfacer las necesidades y demandas nacionales de las ciencias de la computación, a menos que el reclutamiento, la expansión y la equidad se supervisen activamente durante el proceso de ampliación (Margolis, Goode, y Chapman, 2015).

Los problemas de equidad también surgen en el nivel de las interacciones de los estudiantes dentro del aula. La investigación confirma que la programación en pares, un tipo de estructura de colaboración cuando los estudiantes programan juntos mientras rotan a través de roles definidos, puede ser beneficiosa para desarrollar el pensamiento computacional y desarrollar el conocimiento de la programación (Denner, Werner, Campe y Ortiz, 2014). Sin embargo, en situaciones de programación de pares, la compatibilidad de pares y el enfoque del trabajo pueden llevar a la inequidad. Los pares de estudiantes con más incompatibilidades resultan en más retiros de cursos y menor permanencia en los cursos de programación iniciales (Watkins y Watkins, 2009). Las parejas de estudiantes cuya programación se centra en la velocidad de finalización, en lugar de la calidad y la colaboración, a menudo muestran interacciones más inequitativas (Lewis y Shah, 2015). Las diferencias en la equidad también pueden aparecer en otras interacciones de los estudiantes fuera de la programación, como aquellas basadas en el acceso de los estudiantes a relaciones productivas entre compañeros (Shah et al., 2013) o grupos de compañeros orientados a la tecnología (Goode, Estrella y Margolis, 2006).

A medida que la instrucción de las ciencias de la computación se traslada al aula principal, los entornos de programación y los planes de estudio deben diferenciarse para una población estudiantil más diversa. Una investigación reciente ha explorado el uso del Diseño Universal para el Aprendizaje (UDL) para desarrollar y perfeccionar las experiencias introductorias de las ciencias de la computación para una amplia gama de estudiantes (Hansen, Hansen, Dwyer, Harlow & Franklin, 2016). Las adaptaciones de aprendizaje y las modificaciones curriculares demuestran que las técnicas establecidas para la instrucción de diferenciación se pueden aplicar fácilmente en ciencias de la computación para involucrar a todos los estudiantes.

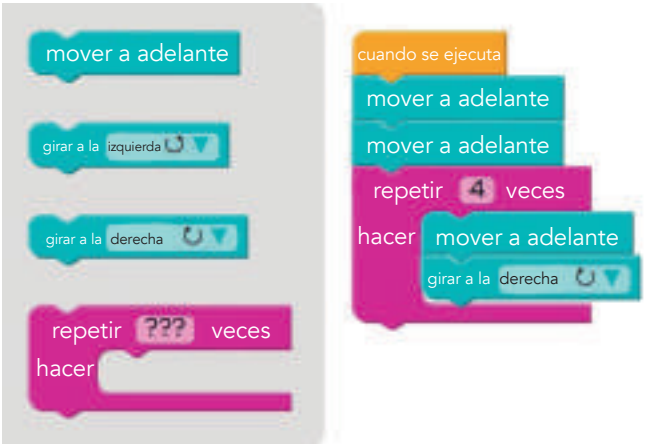
Las técnicas establecidas para la instrucción de diferenciación se pueden aplicar fácilmente en ciencias de computación para involucrar a todos los estudiantes.

Las siguientes secciones describen brevemente los esfuerzos para aumentar el acceso a las ciencias de la computación y cambiar el currículo, la instrucción y la cultura del aula. Aunque no son exhaustivos, los siguientes ejemplos ilustran enfoques prácticos clave que los educadores pueden usar para aumentar la equidad en ciencias de la computación. Cada ejemplo demuestra la importancia de proporcionar acceso y apoyo para presentar oportunidades realistas para que los estudiantes aprendan las ciencias de la computación.

Ejemplos: Llegar a Estudiantes Jóvenes y Principiantes.

Una variedad de enfoques hace que la programación sea más accesible para jóvenes y principiantes. Visual, uno de los lenguajes de programación basados en bloques diseñados para la educación (ver Figura 2.1) permiten a los estudiantes programar sin el obstáculo de los errores de sintaxis (errores en los comandos de escritura) que se encuentran en los métodos tradicionales de lenguajes basados en texto.

Figura 2.1: Ejemplo de lenguaje de programación basado en bloques



Estos lenguajes y entornos han sido diseñados tanto para los programadores jóvenes como para los principiantes, pero también permiten que los estudiantes creen programas complejos, juegos, aplicaciones y animaciones. Las comunidades sociales que han evolucionado a su alrededor les permiten a los estudiantes apoyar a cada uno el desarrollo de otros al compartir, reutilizar y remezclar las creaciones de otros (Brennan y Resnick, 2012; Kafai y Burke, 2014). Programación de entornos en tabletas para niños de hasta 5 años hizo la programación aún más accesible para los niños más pequeños al reducir el número de los comandos y la cantidad de lectura necesaria para navegar por las opciones (Strawhacker & Bers, 2014). UNAEl entorno robótico creado para estudiantes de preescolar utiliza bloques físicos de madera para crear conjuntos de comandos que el robot puede leer y ejecutar (Elkin, Sullivan, & Bers, 2014). Juegos y las aplicaciones que enseñan habilidades de programación están incluso disponibles en los teléfonos inteligentes.

Los distritos escolares de todo el país han transformado sus clases elementales de computación para enfocarse en ciencias de la computación (por ejemplo, las Escuelas Públicas de Lincoln, 2016; Departamento de Educación de la Ciudad de Nueva York, 2016; Distrito Escolar Unificado de San Francisco, 2016). Comenzando en la escuela primaria, los estudiantes usan lenguajes de programación visuales, basados en bloques y aprenden sobre variables, bucles, declaraciones condicionales, funciones, eventos y más en el contexto de hacer proyectos que les resulten divertidos y atractivos. A medida que pasan a los niveles de enseñanza anteriores, aplican sus habilidades para crear proyectos, historias, juegos, aplicaciones, robots de programas y más. Hay planes de estudio a nivel nacional y esfuerzos de desarrollo profesional enfocados en ayudar a los profesores de escuelas primarias a aprender e integrar las ciencias de la computación en la experiencia en el aula de los estudiantes (por ejemplo, Code.org, 2016; Project Lead the Way, 2016).

Si bien la falta de computadoras y el acceso a Internet continúa siendo una desventaja para llegar a todos los estudiantes, muchos temas de las ciencias de la computación, como el pensamiento algorítmico, la búsqueda, la clasificación y la lógica, pueden aprenderse sin computadoras. Las ciencias de la computación "desenchufada" es un método para aprender conceptos de las ciencias de la computación a través de experiencias físicas y kinestésicas y se puede enseñar de forma independiente a la computadora o al acceso en línea (CS Unplugged, 2016). Los estudiantes van en una búsqueda del tesoro para comprender los autómatas de estados finitos, hacen trucos de magia para aprender cómo las computadoras detectan los errores y pasan frutas entre ellos para aprender sobre el enrutamiento de la red y el punto muerto. Los profesores pueden combinar estas experiencias desenchufadas con ejercicios de programación para proporcionar experiencias aún más ricas para los estudiantes jóvenes.

Ejemplos: Llegar a Estudiantes con Discapacidades.

Algunos grupos de investigación e implementación están enfocando su trabajo en asegurar un acceso significativo a las ciencias de la computación para estudiantes con discapacidades. Este trabajo es de importancia crítica ya que el 13% de los estudiantes en las escuelas públicas de los EE.UU. reciben servicios de educación especial conforme a la Ley de Educación para Personas con Discapacidades (NCES, 2016) y otra parte importante de los estudiantes recibe adaptaciones según la Ley de Estadounidenses con Discapacidades. Debido a que las escuelas tienen la obligación legal de satisfacer las necesidades de estos alumnos, es fundamental que la educación en ciencias de la computación se centre también en esta población de alumnos. Se han realizado tres esfuerzos recientes para llegar a los estudiantes con discapacidades. El Quorum lenguaje de programación, desarrollado en la Universidad de Nevada, Las Vegas, puede ser leído por lectores de pantalla de computadoras existentes, haciéndolos accesibles a estudiantes con impedimentos visuales (Quórum,2016). Además de acomodar a estudiantes con discapacidades sensoriales, como discapacidades visuales, los investigadores del Laboratorio de Investigación de Tecnologías Creativas (CTRL-Shift) de la Universidad de Illinois en Urbana/Champaign está desarrollando e investigando estrategias de enseñanza centradas en aumentar el acceso a las ciencias de la computación para estudiantes con discapacidades cognitivas (Creative Technologies Research Lab, 2016). Investigadores del grupo ACCESS CS10K en la Universidad de Washington han desarrollado un plan de estudios de recursos y desarrollo profesional para docentes de alumnos con discapacidad. (AccessCS10K, 2016).

Ejemplos: Llegar a las Mujeres y Minorías Subrepresentadas.

Muchos programas y estrategias están diseñados para involucrar a los estudiantes de grupos que están subrepresentados en ciencias de la computación. Explorar las Ciencias de la Computación (ECS, 2016), en un curso introductorio a nivel de escuela secundaria, y AP Principios de las Ciencias de la Computación, un curso introductorio del nivel universitario, son dos de los más recientes cursos destacados en la escuela. Su contenido y atención a la equidad influyeron en los conceptos y prácticas del marco. Los profesores de estos y otros cursos emplean una serie de estrategias para problemas de combate, como la amenaza de estereotipos, sesgos y mentalidades fijas, que ponen en peligro la equidad en el salón de clases. Juntos, estos programas y estrategias ejemplifican un enfoque para cambiar el currículo, la instrucción y la cultura en el aula para ampliar la participación, especialmente entre las mujeres y minorías subrepresentadas.

ECS está diseñado para involucrar a los estudiantes de secundaria en prácticas computacionales. Proyectos e instrucción se basan en la indagación y la equidad y están diseñados para ser socialmente relevantes y significativos para diversos estudiantes. Margolis et. al (2012) describe el enfoque culturalmente relevante en las lecciones de ECS:

> Estos tipos de lecciones son una forma de ayudar a los estudiantes a construir relaciones personales con las ciencias de la computación (CC) conceptos y aplicaciones: un proceso importante para descubrir la relevancia de la CC para sus propias vidas. Por ejemplo, los estudiantes pueden aprender geometría fractal a través de las Fractals "o" [Cornrow] Braiding ", pero también pueden aprender física y formas de programar pendientes y arcos a través del programa "Skateboarding". Reconocemos que los estudiantes son todos diferentes, la cultura es de múltiples capas, y mientras que algunos estudiantes pueden estar interesados en las culturas de sus antepasados, otros pueden estar interesados en la cultura del hip hop, diseño gráfico, skateboarding,

El programa de desarrollo profesional que lo acompaña se basa en tres pilares principales de las ciencias de la computación: contenidos/conceptos de la ciencia, indagación y equidad. Aunque algunos profesores pueden tener problemas para ver la inequidad en sus aulas (Hu, Heiner y McCarthy, 2016), las sesiones de desarrollo profesional son dirigidos a abordar los sistemas de creencias de los profesores y puede abrir pensamientos sobre el pensamiento deficitario y privilegio preparatorio (Margolis, Goode, Chapman y Ryoo, 2014). Los profesores están capacitados para aprovechar el conocimiento cultural de los estudiantes y llevarlo a la experiencia de aprendizaje (Goode, 2008; Eglash, Gilbert,& Foster, 2013).

El curso de Principios de las Ciencias de la Computación de AP destaca siete grandes ideas y seis prácticas computacionales en un esfuerzo por atraer a una amplia audiencia de estudiantes que pueden estar interesados en ciencias de la computación más allá de la experiencia centrada en la programación tradicional actualmente en las escuelas K–12. Estudiantes aprenden sobre una variedad de temas, desde cómo la computación extiende la expresión humana hasta cómo la tecnología afecta al mundo. El curso de Principios de las Ciencias de la Computación de AP difiere de ECS en algunos aspectos. Primero, a pesar de ser un curso introductorio, los Principios de las Ciencias de la Computación de AP se consideran de nivel de curso universitario y puede no ser apropiado como la única oferta de las ciencias de la computación en una escuela secundaria. Segundo, a diferencia de ECS, no hay un currículo singular; profesores y proveedores de contenido crean planes de lección y recursos basados en el marco del curso proporcionado. Finalmente, los estudiantes pueden calificar para crédito o colocación a nivel universitario al aprobar un examen del fin de año compuesto de opción múltiple preguntas, así como las tareas de rendimiento que se completan como evaluaciones a través del curso durante el

año del marco de las ciencias de la computación K-12, Equidad en la educación de las ciencias de la computación. Estas tareas de desempeño les permiten a los estudiantes alcanzar objetivos de evaluación mensurables al completar algunas tareas que permiten la elección personal en la elección de temas, así como alentar colaboración con compañeros.

Una variedad de acciones para cambiar la cultura del aula requiere que la comunidad las aborde. Bien investigado los obstáculos a la equidad en otras materias incluyen la amenaza de estereotipos (Steele, Spencer y Aronson, 2002),Sesgos implícitos (Greenwald y Krieger, 2006) o inconscientes (Pollard-Sacks, 1999) y mentalidades fijas (Dweck, 2000); su influencia se aplica también a las ciencias de la computación (Cutts, Cutts, Draper, O'Donnell, &Saffrey, 2010; Kumar, 2012; Simon et al., 2008). Por ejemplo, si los tutores enseñan a los estudiantes sobre mentalidades y darles mensajes de mentalidad de crecimiento en su trabajo, los resultados de las pruebas podrían mejorar y la amenaza estereotípica podría ser mitigada (Cutts et al., 2010) La amenaza estereotípica se puede mitigar la alteración de la redacción de las preguntas de la prueba y debe ser neutral en cuanto al género y utilizar ejemplos que son igualmente relevantes para ambos hombres y mujeres (Kumar, 2012). También es importante que los estudiantes tengan diversos modelos a seguir en el campo para que puedan imaginarse a sí mismos como científicos de la computación, así como para disipar los estereotipos de cómo se ven y actúan los científicos de la computación (Goode, 2008).

Otras prácticas que los profesores pueden adoptar y adaptar para cambiar la cultura del aula incluyen:

- practicar una pedagogía culturalmente relevante que une las ciencias de la computación con las experiencias de los estudiantes, cultura e intereses (Margolis et al., 2014);
- desarrollar relaciones con los estudiantes que son respetuosos de las diferencias entre personas, de sus orígenes distintos y de ser empáticos a las necesidades e intereses entre las personas diferentes (Margolis et al., 2014);
- reflexionar sobre las creencias y acciones para abordar los estereotipos entre estudiantes y profesores por igual.(Margolis et al., 2014);
- aplicar estrategias de instrucción que apoyan a los estudiantes con dificultades y aquellos con discapacidades en otras áreas de contenido dentro de la educación de las ciencias de la computación (por ejemplo, si las pautas verbales ayudan en la instrucción de las matemáticas, probablemente también ayudará en la instrucción de las ciencias de la computación) (Snodgrass, Israel, y Reese,2016); y
- conectar las ciencias de la computación con conceptos que motivan a los niños, como la imparcialidad y la justicia social.(Denner, Martínez, Thiry y Adams, 2015).

Resumen

Un enfoque en la equidad, la inclusión y la diversidad en todos los aspectos de la educación de las ciencias de la computación asegurará que la implementación de los esfuerzos se mantiene fieles a la visión del marco de las ciencias de la computación para todos los alumnos. La estructura y el contenido del marco refleja una atención a las cuestiones de equidad y proporciona una fundación sólida para la reforma de la educación de las ciencias de la computación. Existen muchos esfuerzos actuales para promover las ciencias de la computación a diversas poblaciones, incluyendo jóvenes estudiantes, novatos, estudiantes con discapacidades, mujeres y minorías subrepresentadas.

Un enfoque en la equidad asegurará que los esfuerzos del aprendizaje mantengan la fidelidad de la visión de las ciencias de la computación para todos los alumnos.

Referencias

AccessCS10K. (2016). *Including students with disabilities in computing education for the 21st century.* Retrieved from http://www.washington.edu/accesscomputing/accesscs10k

Bootstrap. (2016). *From intro CS & Algebra to high-level courses in computer science, for all students.* Retrieved from http://www.bootstrapworld.org/

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper session presented at the annual meeting of the American Educational Research Association, Vancouver, Canada.

Bureau of Labor Statistics. (2015a). *Employment projections* [Data file]. Retrieved from http://www.bls.gov/emp/tables.htm

Bureau of Labor Statistics. (2015b). *Labor force statistics from the Current Population Survey* [Data file]. Retrieved from http://www.bls.gov/cps

Bureau of Labor Statistics. (2015c). *Occupational employment statistics* [Data file]. Retrieved from http://www.bls.gov/oes

Burning Glass Technologies. (2016). *Beyond point and click: The expanding demand for coding skills.* Retrieved from http://burning-glass.com/research/coding-skills/

Change the Equation. (2016, August 9). New data: Bridging the computer science access gap [Blog post]. Retrieved from http://changetheequation.org/blog/new-data-bridging-computer-science-access-gap-0

Code.org. (2016). *Computer Science Fundamentals* for elementary school. Retrieved from https://code.org/educate/curriculum/elementary-school

College Board. (2014). *The tenth annual AP report to the nation.* Retrieved from http://media.collegeboard.com/digitalServices/pdf/ap/rtn/10th-annual/10th-annual-ap-report-to-the-nation-single-page.pdf

College Board. (2016). *AP program participation and performance data 2015* [Data file]. Retrieved from https://research.collegeboard.org/programs/ap/data/participation/ap-2015

The Conference Board. (2016). *National demand rate and OES employment data by occupation* [Data file]. Retrieved from https://www.conference-board.org/

Creative Technology Research Lab. (2016). *CTRL-Shift: Shifting education.* Retrieved from http://ctrlshift.mste.illinois.edu/home/

CS Unplugged. (2016). Retrieved from http://csunplugged.org/

Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., & Saffrey, P. (2010, March). Manipulating mindset to positively influence introductory programming performance. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 431–435). doi: 10.1145/1734263.1734409

Denner, J., Martinez, J., Thiry, H., & Adams, J. (2015). Computer science and fairness: Integrating a social justice perspective into an after school program. *Science Education and Civic Engagement: An International Journal, 6*(2): 41–54.

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education, 46*(3), 277–296.

Dweck, C. S. (2000). *Self-theories: Their role in motivation, personality, and development.* Philadelphia, PA: Psychology Press.

Eglash, R., Gilbert, J. E., & Foster, E. (2013). Toward culturally responsive computing education. *Communications of the ACM, 56*(7), 33–36.

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice, 13*, 153–169.

Exploring Computer Science. (2016). Retrieved from http://www.exploringcs.org/

Goode, J. (2008, March). Increasing diversity in K–12 computer science: Strategies from the field. *ACM SIGCSE Bulletin, 40*(1), 362–366.

Goode, J., Estrella, R., & Margolis, J. (2006). Lost in translation: Gender and high school computer science. In W. Aspray & J. M. Cohoon (Eds.), *Women and Information Technology: Research on Underrepresentation* (pp. 89–113). Cambridge, MA: MIT Press.

Google & Gallup. (2015a). *Images of computer science: Perceptions among students, parents, and educators in the U.S.* Retrieved from http://g.co/cseduresearch

Google & Gallup. (2015b). *Searching for computer science: Access and barriers in U.S. K–12 education*. Retrieved from http://g.co/cseduresearch

Greenwald, A. G., & Krieger, L. H. (2006). Implicit bias: Scientific foundations. *California Law Review, 94*(4), 945–967.

Hansen, A., Hansen, E., Dwyer, H., Harlow, D., & Franklin, D. (2016). Differentiating for diversity: Using universal design for learning in computer science education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 376–381).

Horizon Media. (2015, October 5). Horizon Media study reveals Americans prioritize STEM subjects over the arts; science is "cool," coding is new literacy. *PR Newswire*. Retrieved from http://www.prnewswire.com/news-releases/horizon-media-study-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html

Hu, H. H., Heiner, C., & McCarthy, J. (2016, February). Deploying Exploring Computer Science statewide. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 72–77).

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: The MIT Press.

Kumar, A. N. (2012, July). A study of stereotype threat in computer science. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 273–278).

Ladner, R., & Israel, M. (2016). "For all" in "computer science for all." *Communications of the ACM, 59*(9), 26–28.

Lewis, C. M., & Shah, N. (2015, July). How equity and inequity can emerge in pair programming. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 41–50).

Lincoln Public Schools. (2016). *Computer science*. Retrieved from http://home.lps.org/computerscience/

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing*. Cambridge, MA: MIT Press.

Margolis, J., Goode, J., & Chapman, G. (2015). An equity lens for scaling: A critical juncture for exploring computer science. *ACM Inroads, 6*(3), 58–66.

Margolis, J., Goode, J., Chapman, G., & Ryoo, J. J. (2014). That classroom 'magic.' *Communications of the ACM,* 57(7), 31–33.

Margolis, J., Ryoo, J., Sandoval, C., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads, 3*(4), 72–78.

Morgan, R., & Klaric, J. (2007). *AP students in college: An analysis of five-year academic careers. Research report no. 2007-4*. Retrieved from the College Board website: http://research.collegeboard.org/sites/default/files/publications/2012/7/researchreport-2007-4-ap-students-college-analysis-five-year-academic-careers.pdf

National Center for Education Statistics. (2014). *Integrated postsecondary education data system* [Data file]. Retrieved from http://nces.ed.gov/ipeds/

National Center for Education Statistics. (2016). *Children and youth with disabilities*. Retrieved from http://nces.ed.gov/programs/coe/indicator_cgg.asp

National Science Foundation. (2015). *Women, minorities, and persons with disabilities in science and engineering*. Arlington, VA: Author.

New York City Department of Education. (2016). *Software engineering program (SEP) Jr.* Retrieved from http://sepnyc.org/sepjr/

Pollard-Sacks, D. (1999). Unconscious bias and self-critical analysis: The case for a qualified evidentiary equal employment opportunity privilege. *Washington Law Review, 74,* 913–1032.

Project GUTS: Growing Up Thinking Scientifically. (2016). Retrieved from http://www.projectguts.org/.

Project Lead The Way. (2016). *PLTW launch (K–5).* Retrieved from https://www.pltw.org/our-programs/pltw-launch.

Quorum. (2016). *About Quorum*. Retrieved from https://www.quorumlanguage.com/about.php

San Francisco Unified School District. (2016). *Computer science for all students in SF.* http://www.csinsf.org/

Shah, N., Lewis, C. M., Caires, R., Khan, N., Qureshi, A., Ehsanipour, D., & Gupta, N. (2013, March). Building equitable computer science classrooms: Elements of a teaching approach. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 263–268).

Simon, B., Hanks, B., Murphy, L., Fitzgerald, S., McCauley, R., Thomas, L., & Zander, C. (2008, September). Saying isn't necessarily believing: Influencing self-theories in computing. In *Proceedings of the Fourth International Workshop on Computing Education Research* (pp. 173–184).

Snodgrass, M. R., Israel, M., & Reese, G. (2016). Instructional supports for students with disabilities in K–5 computing: Findings from a cross-case analysis. *Computers & Education, 100,* 1–17.

Steele, C. M., Spencer, S. J., & Aronson, J. (2002). Contending with group image: The psychology of stereotype and social identity threat. *Advances in Experimental Social Psychology, 34*, 379–440.

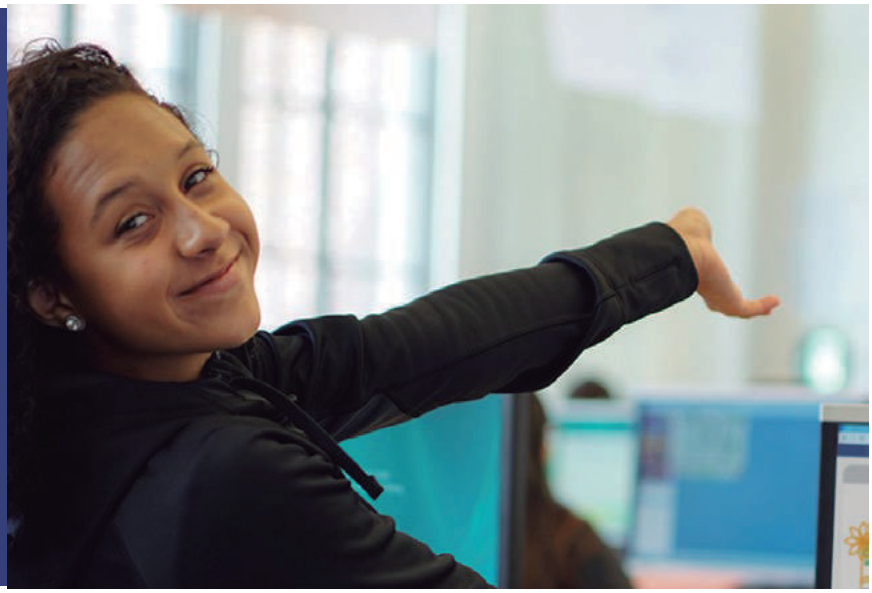
Strawhacker, A. L., & Bers, M. U. (2014, August). *ScratchJr: Computer programming in early childhood education as a pathway to academic readiness and success.* Poster presented at DR K–12 PI Meeting, Washington, DC.

University of Nebraska at Omaha. (2016). *ITEST strategic problem-based approach to rouse computer science (SPARCS).* Retrieved from http://www.unomaha.edu/college-of-information-science-and-technology/itest/index.php

Watkins, K. Z., & Watkins, M. J. (2009). Towards minimizing pair incompatibilities to help retain under-represented groups in beginning programming courses using pair programming. *Journal of Computing Sciences in Colleges, 25*(2), 221–227.

Proceso de Desarrollo

3



Proceso de Desarrollo

El desarrollo del marco de las ciencias de la computación K–12 reunió a educadores profesionales, investigadores, organizaciones y partes interesadas a nivel de estado y distrito para delinear conceptos y prácticas en ciencias de la computación para todos los estudiantes desde kindergarten hasta el grado 12 en los Estados Unidos. Este proyecto fue guiado por un comité de dirección con representación de la Association for Computing Machinery (ACM), Code.org, Computer Science Teachers Association (CSTA), Cyber Innovation Center (CIC), and the National Math and Science Initiative (NMSI).

Una variedad de organizaciones desempeñaron roles especiales durante el proceso de desarrollo, tales como Achieve; the Maine Mathematics and Science Alliance; y los colaboradores de la Board on Science Education at the National Academies of Science, Engineering, and Medicine, quienes aplicaron sus experiencias con las autoridades estatales y nacionales de marcos de educación para informar la estructura del marco de las ciencias de la computación K–12, considerando las expectativas, y el proceso de desarrollo. Research & Evaluation in UChicago STEM Education at the University of Chicago, se documentaron las diversas reuniones, convocatorias y talleres; los resúmenes de las reuniones están disponibles en k12cs.org

Los estados que participaron en el desarrollo del marco de las ciencias de la computación K–12 fueron ya sea que esté actualmente involucrado o planeando a comenzar con la implementación de las ciencias de la computación K–12. Estos estados incluyen Arkansas, California, Georgia, Idaho, Indiana, Iowa, Maryland, Massachusetts, Nebraska, Nevada, New Jersey, North Carolina, Utah y Washington. Los distritos escolares en Chicago (IL), así mismo participaron las ciudades de New York (NY), San Francisco (CA) y el condado de Charles (MD). Cada uno de los estados y los distritos que se comprometieron con el proyecto se les pidió que seleccionaran un escritor para representar al estado en el equipo de redacción. Los equipos a nivel estatal revisaron el marco y participaron en las reuniones durante el Proceso de Desarrollo del marco.

El cronograma de desarrollo consistió en una serie de reuniones con asesores, escritores y partes interesadas desde octubre de 2015 hasta octubre de 2016. Las tres reuniones de asesores se centraron en la articulación de visión y principios de la guía del marco de las ciencias de la computación K–12, identificando los conceptos core y prácticas, y proporcionando retroalimentación durante el proceso de desarrollo. En el transcurso de dos reuniones de partes interesadas, una para lanzar el proyecto del marco y otra en el punto medio, se ofreció a los participantes aportes, actualizaciones de desarrollo recibidas y discusiones en torno a los problemas comunes de la implementación de las ciencias de la computación. Los siete talleres de escritores fueron oportunidades para que los escritores colaboraran en forma presencial y revisar los borradores del marco basado en la opinión del público. Escritores y asesores tuvieron video conferencias programadas entre las reuniones presenciales. Hubo un total de tres períodos de la revisión pública, así como múltiples revisiones internas y grupos de enfoque que cubren temas especiales como currículo, pensamiento computacional y adecuación del desarrollo (incluso los primeros años de la educación durante la infancia).

Este capítulo detalla cómo comenzó el proyecto marco, cómo se involucró a la comunidad y cómo cada parte del marco fue desarrollada.

Comienzos del marco

El impulso para desarrollar el marco provino de un interés mutuo en ciencias de la computación K-12 expresado por múltiples estados. Con la creciente atención de las escuelas, a los padres y a todos los estudiantes de todas las naciones, todos los estados de EE.UU. estaban preguntando cada vez más: "¿Qué deberían saber y hacer los estudiantes en una escuela K–12 en camino a las ciencias de la computación? ¿Cómo se ve las ciencias de la computación en la primaria y la enseñanza media?" El marco de las ciencias de la computación K–12 representa una respuesta a la creciente necesidad de orientación centrada en el estado que refleja el consenso de una amplia y diversa gama de las ciencias de la computación profesionales, investigadores y organizaciones. Muchos de estos estados desconfiaron de participar en otra ola de estándares educativos nacionales similares a los de matemáticas, artes del lenguaje y ciencias, lo que refleja el ritmo y la política de la reforma educativa basada en estándares en los últimos años. Algunos estados expresaron cansancio por los esfuerzos de estos estándares nacionales y el deseo de contar con directrices más flexibles y de alto nivel para la ciencia de la computación K-12 mediante las cuales podrían crear su propia guía desarrollada por el estado.

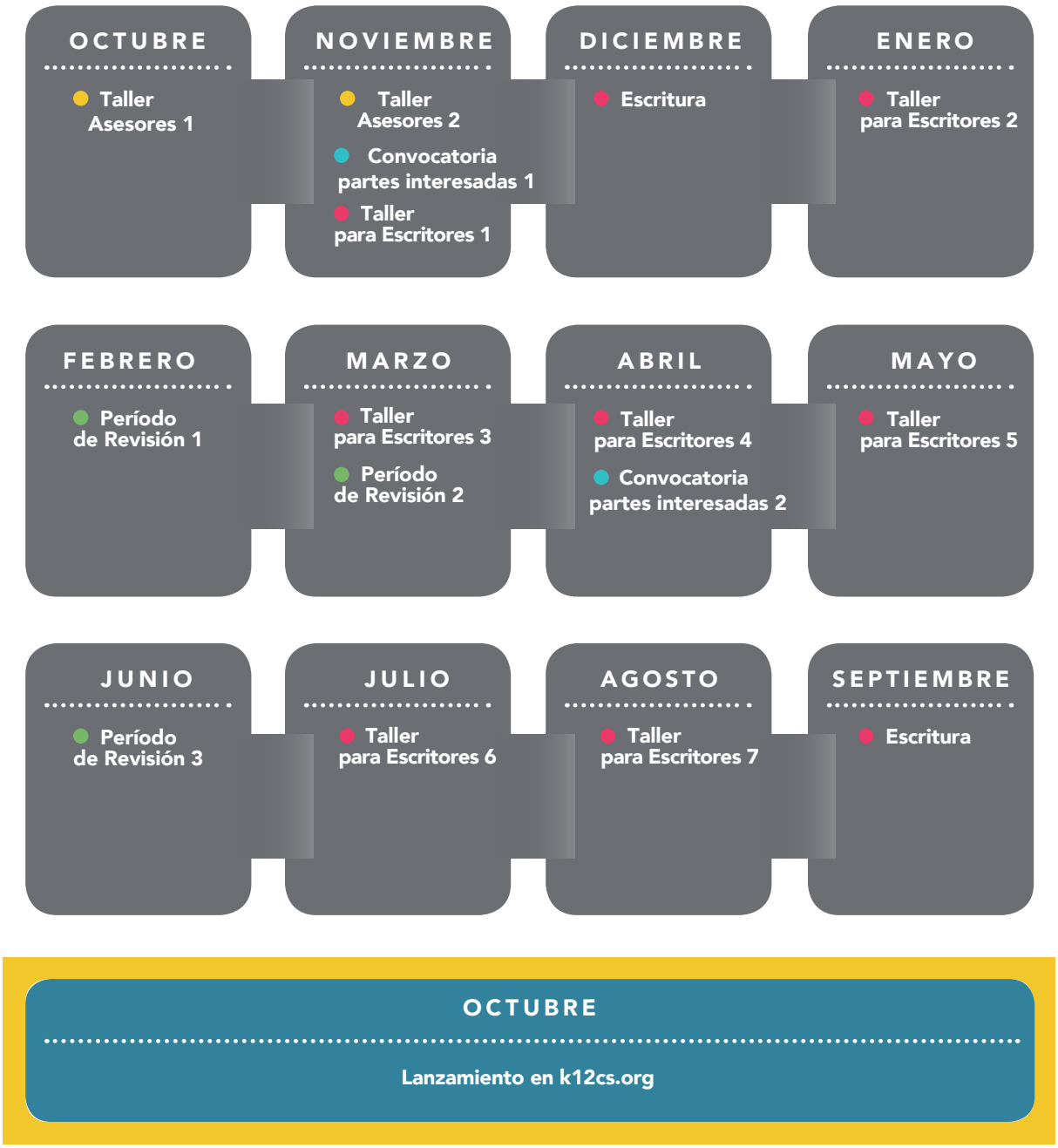
El marco se basa en publicaciones anteriores que describen expectativas detalladas para la educación en ciencias de la computación de K–12 en los EE.UU : un plan de estudios modelo para K–12 de las ciencias de la computación, 2ª edición (Tucker et al., 2006) y los Estándares de las ciencias de la computación de CSTA K – 12 (CSTA Standards Task Force, 2011). Las dos organizaciones detrás de estos documentos, ACM y CSTA, se unieron a Code.org, una organización nacional sin fines de lucro que promueve la educación en informática, para formar el comité directivo inicial para el marco de las ciencias de la computación K–12, que luego incluiría CIC y NMSI. El comité directivo se encargó de guiar el proceso de desarrollo del marco; supervisar el proceso de revisión para asegurar múltiples oportunidades para la participación diversa de la comunidad; aumentar la coherencia entre el marco, los estándares y otros documentos relacionados; y representando el proyecto para aumentar la conciencia pública. Code.org proporcionó personal para dirigir el desarrollo del marco y gestionar el proceso. El proyecto fue financiado por la ACM, Code.org y NMSI. Ver una línea de tiempo del proyecto en la Figura 3.1.

El marco se basa en publicaciones anteriores que describen expectativas detalladas para la educación en ciencias de la computación K-12.

El CSTA comenzó una revisión programada de sus estándares de 2011 al mismo tiempo que comenzó el proceso de desarrollo del marco. Dado que dos procesos relacionados, aunque independientes, se emprendieron en paralelo en la comunidad de educación en ciencias de la computación, fue importante asegurar que la comunidad hablara con una voz coherente y consistente ante los interesados nacionales (Stakeholders). Para mejorar la alineación entre los dos proyectos, los co-presidentes de la revisión de estándares de CSTA sirvieron como asesores del marco de las ciencias de la computación K-12, y se pidió a todos los redactores de estándares de CSTA que fueran escritores del marco, y finalmente aceptaron la mitad este papel. De esta forma, tanto los documentos informados

como los escritores se apoyaban mutuamente. Una versión provisional de los estándares CSTA revisados se publicó en julio de 2016 y fue informada por múltiples aportes, incluidos los borradores del marco de las ciencias de la computación K–12 durante el proceso de desarrollo.

Figura 3.1: Proceso de desarrollo del marco.



Participación de la Comunidad

Los 27 escritores del marco representaron a la comunidad de investigación, educación K–12, asociaciones profesionales, organizaciones sin fines de lucro, departamentos estatales y distritos de educación e industria. El equipo de redacción se basó en la experiencia de nivel de grado y en el contenido, así como en la experiencia de género, raza, etnia, estatal, rural/urbano/suburbano y representación institucional (organizaciones sin fines de lucro, investigación, industria, pública/privada). Los escritores fueron asignados a equipos según su experiencia de nivel de grado y experiencia en contenidos. Los cinco equipos de conceptos core y un equipo de práctica estaban compuestos por escritores, un escritor principal y un facilitador.

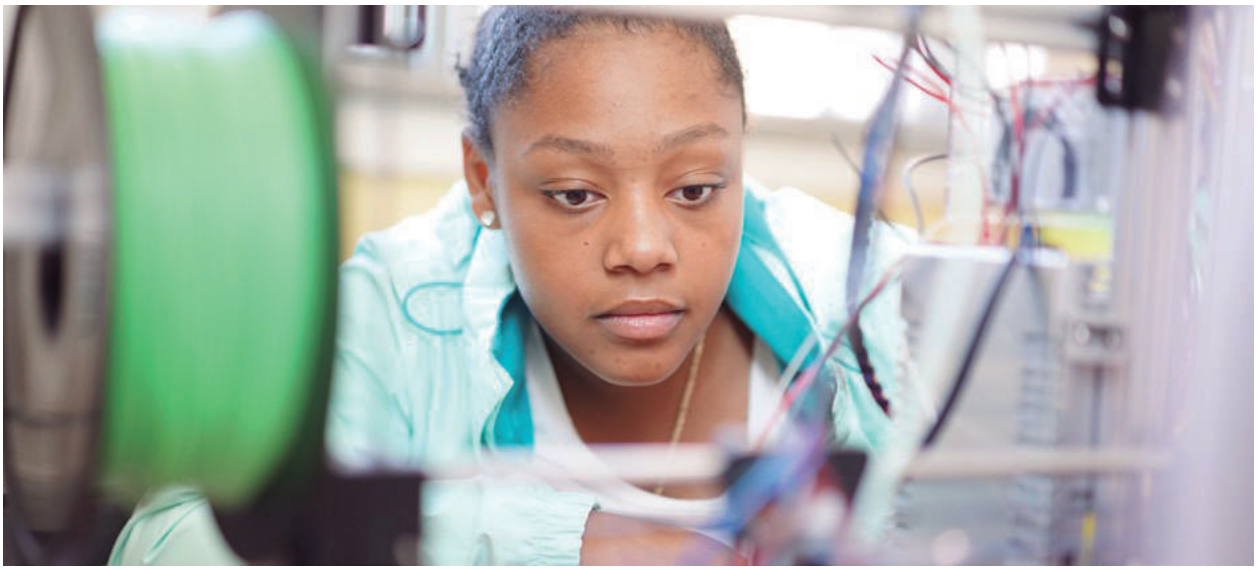
Había 27 escritores y 25 asesores que representaban a estados, distritos, K–12, industria y organizaciones sin fines de lucro.

También hubo 25 asesores, en representación de K–12 y educación superior, que participaron en las reuniones iniciales para identificar los conceptos y prácticas core del marco y proporcionaron retroalimentación durante el proceso de desarrollo. Los asesores eran profesionales e investigadores que representaban cuerpos de trabajo y experiencia que eran valiosos para el desarrollo del marco. Los asesores participaron en revisiones internas y trabajaron en colaboración cercana con el personal de desarrollo y los equipos de redacción para informar las áreas de contenido clave, proporcionar una base de investigación y ayudar a resolver los problemas de contenido durante el desarrollo. Una variedad de partes críticas e interesadas (Stakeholders) participaron en el desarrollo del marco. Los estados participantes fueron representados por personal del departamento de educación del estado y/o miembros de la junta estatal de educación. Los distritos estuvieron representados por coordinadores y especialistas en las ciencias de la computación. Representantes de la industria, organizaciones sin fines de lucro, instituciones de investigación y organizaciones nacionales de educación brindaron información clave durante las reuniones de las partes interesadas.

La revisión pública y la retroalimentación fueron esenciales para el desarrollo del marco. Se compartieron un total de tres borradores del marco para comentarios del público, además de varios borradores internos que se compartieron con los asesores del marco. Las revisiones fueron enviadas por grupos e individuos que representan a la mayoría de los estados de la nación y siete ubicaciones internacionales. Se realizaron grupos de enfoque especiales para recopilar comentarios detallados sobre temas sobre los cuales los escritores querían comentarios y para evaluar las necesidades de audiencias particulares. Los participantes y revisores de los grupos focales pertenecían a la comunidad de educación en ciencias de la computación, así como a la comunidad de educación general y representaban una variedad de roles en la educación, como profesores, administradores, profesores de educación superior, investigadores, especialistas en currículos y asesores tecnológicos. Al final de cada período de revisión, el personal de desarrollo revisó todos los aportes, identificó los temas principales y proporcionó recomendaciones al equipo de redacción. Puede encontrar más información sobre el proceso de revisión, los comentarios recibidos y las revisiones posteriores en **Apéndice A: Comentarios y revisiones**.

Estructura

La decisión de organizar el marco por conceptos y prácticas se basó en el deseo de alinear y complementar la estructura de los marcos educativos ampliamente adoptados para materias como matemáticas, ciencias y artes del lenguaje. En particular, el marco del Consejo Nacional de Investigación (NRC) para la Educación en ciencias K–12 (2012) sirvió de modelo y precedente. El marco de NRC tiene tres dimensiones: Ideas Básicas de Disciplina, Prácticas Científicas y de Ingeniería, y Conceptos Transversales. El marco de ciencias de la computación K–12 comparte dos dimensiones similares, en forma de conceptos y prácticas core. Los conceptos transversales no son una tercera dimensión explícita en el marco de las ciencias de la computación K–12, pero se integran a través de los conceptos core. Los puntos finales de la banda de grado (grados 2, 5, 8 y 12) para los conceptos del marco de las ciencias de la computación K–12 también se modelan a partir de los del marco de NRC y representan etapas clave de la educación, incluido el reconocimiento de la diferencia de desarrollo entre estudiantes menor y mayor de la escuela primaria superior.



Conceptos Core

Los conceptos core del marco de ciencias de la computación K–12 son categorías que representan áreas de contenido principales en el campo de las ciencias de la computación. Representan áreas específicas de importancia disciplinaria más que abstractas, ideas generales. Este enfoque reconoce la importancia de usar contenido y contexto específicos para organizar cuerpos de conocimiento (NRC, 2007), como datos, redes y programación, sobre ideas de dominio general como abstracción y pensamiento computacional. En una discusión sobre las progresiones de aprendizaje en la ciencia, la NRC informó que un "enfoque disciplinario encaja con el creciente reconocimiento de la importancia del contenido y el contexto específicos en el pensamiento y el aprendizaje y el poder de las teorías para definir y organizar entendimientos de dominios particulares, algo que las ideas de dominio general por su naturaleza no tienen el poder de hacerlo "(NRC, 2007, p. 223). El criterio para la selección de un concepto core fue que debería

- tener un amplio significado en el campo de las ciencias de la computación;
- servir como una base útil para aprender o construir otras ideas en ciencias de la computación;
- permitir que los jóvenes estudiantes se involucren con la idea (umbral bajo), pero preservar el potencial de elaboración progresiva y sofisticada (techo alto);
- ser aplicable dentro de otras materias y disciplinas K–12; y
- seguir siendo relevante en ciencias de la computación durante los próximos cinco a diez años, como mínimo.

Los conceptos core del marco:

1. Sistemas de Computación
2. Redes e Internet
3. Datos y Análisis
4. Algoritmos y Programación
5. Impactos de la computación

Los escritores identificaron los subconceptos dentro de cada uno de los conceptos core y los utilizaron para crear progresiones de aprendizaje coherentes y enfocadas que abarcan desde el jardín de infantes hasta el grado 12. La selección de los conceptos core y los subconceptos fue informada por los documentos de las ciencias de la computación relacionados K–12, incluido el Modelo ACM K–12 Currículo (Tucker et al., 2006); los estándares CSTA 2011; el marco curricular de los Principios de Ciencias de la Computación de Advanced Placement® (College Board, 2016); los grandes principios de la computación del Instituto Denning (Denning y Martell, 2015); y marcos internacionales, como el programa nacional de estudios de computación del Reino Unido (Departamento de Educación de Inglaterra, 2013). Los marcos de las disciplinas relacionadas, tales como seguridad cibernética, ciudadanía digital y alfabetización tecnológica, también informaron las progresiones de aprendizaje. Las experiencias de los escritores de marcos con diversas poblaciones de estudiantes demostraron ser invaluables al determinar qué ideas eran esenciales para todos los estudiantes.

Una progresión de aprendizaje describe hitos conceptuales a lo largo de un camino que pasa de la comprensión básica a un conocimiento más sofisticado en un área temática. Los subconceptos del marco proporcionan un punto focal para estas progresiones de aprendizaje para conectar el aprendizaje a través de los grados y articular las ideas esenciales bajo cada concepto central (Hess, 2008).

En lugar de priorizar la cobertura de una amplia gama de contenido, las progresiones de aprendizaje del marco revisan deliberadamente un subconcepto en múltiples bandas de grado con una sofisticación en evolución. Por ejemplo, la progresión de aprendizaje para la Modularidad en el concepto core de Algoritmos y Programación comienza con la simple comprensión de que las tareas se pueden dividir en tareas más pequeñas y que los programas se pueden componer de partes de otros programas. Eventualmente, los estudiantes comprenden que un programa es un sistema de módulos que interactúan, incluyendo otros programas.

Seguridad cibernética, La ciudadanía digital y la alfabetización tecnológica informaron las progresiones de aprendizaje del marco.

El marco de las ciencias de la computación K–12 empleó los principios utilizados en la construcción e investigación de las progresiones de aprendizaje, tal como la identificación de ideas core emergentes que se pueden introducir temprano en la educación de un niño y desarrollarlas a lo largo de varios años (NRC, 2007). Donde faltaba la investigación específica en ciencias de la computación, especialmente en las bandas de grados iniciales, se usó la investigación relacionada con las ciencias y las matemáticas para aproximar las progresiones apropiadas de las ciencias de la computación y guiar la ubicación de los conceptos en bandas de grados particulares. Por ejemplo, la abstracción de procedimientos, en la que los procedimientos usan variables como parámetros para generalizar el comportamiento, se colocó como una expectativa para finales del octavo grado en el concepto core de Algoritmos y Programación basado en la colocación de un concepto relacionado en las progresiones de aprendizaje para las matemáticas — Escribiendo ecuaciones con variables en los grados medios. Otros ejemplos de cómo confiar en las progresiones de aprendizaje de otras disciplinas incluyen el uso de las progresiones de aprendizaje de la ciencia para informar la ubicación de conceptos compartidos con las ciencias de la computación, tales como modelos y simulaciones, y conceptos análogos, como bits (unidades básicas de información digital) y partículas/átomos.

Una progresión de aprendizaje describe hitos conceptuales a lo largo de un camino que pasa de la comprensión básica a un conocimiento más sofisticado en un área temática.

Cabe señalar que la efectividad de las progresiones de aprendizaje del marco se verá significativamente influenciada por las prácticas pedagógicas de los profesores, ya que los métodos tradicionales de enseñanza de las ciencias de la computación pueden no permitir que todos los estudiantes cumplan con las expectativas del marco. Además, las progresiones de aprendizaje del marco guían, pero no prescriben completamente una secuencia de instrucción; hay caminos flexibles para moverse entre las expectativas al final de cada banda de grado.

Conceptos Transversales

Como se mencionó anteriormente, los conceptos core del Marco de Ciencias de la Computación K-12 representan áreas específicas de importancia disciplinaria en lugar de ideas abstractas de dominio general. Estos últimos formaron la base de los conceptos transversales del marco, ideas que tienen aplicación en los diferentes conceptos core y se integran en las declaraciones de conceptos. Estos "conceptos transversales" proporcionan conexiones temáticas a través de los conceptos core. El criterio para la selección de un concepto transversal fue que debía

- aplicar a través de múltiples conceptos core,
- iluminar conexiones entre diferentes conceptos core de las ciencias de la computación,
- Desarrollar la familiaridad con los temas fundamentales de las ciencias de la computación a través de la repetición en diferentes contextos, y
- crear una comprensión más rica de una declaración de concepto en la que se integra.

Los conceptos transversales del marco (enumerados en orden de frecuencia en el marco):

1. Abstracción
2. Relaciones de sistemas
3. Interacción humano-computadora
4. Privacidad y seguridad
5. Comunicación y coordinación

Si bien el marco de las ciencias de la computación K–12 y el Marco de NRC para la Educación en Ciencias K–12 definen conceptos transversales de manera similar, el marco de las ciencias de la computación K–12 los integra en las progresiones de aprendizaje bajo cada concepto central en lugar de una tercera dimensión separada junto con los conceptos y prácticas core. Esta integración se realizó por dos razones principales. Primero, se decidió que integrarlos en la dimensión existente de los conceptos core preservaría su valor e influencia, al tiempo que facilitaría a la audiencia del marco comprender y aplicar el marco. En segundo lugar, la base de investigación y la experiencia del profesional con conceptos transversales en informática no proporcionaron suficiente información para crear una progresión de aprendizaje separada para los conceptos transversales.

El marco integra conceptos transversales bajo cada concepto core en lugar de una tercera dimensión.

Los conceptos transversales sirvieron como una herramienta de escritura interna durante el proceso de desarrollo, y la lista evolucionó basándose en la retroalimentación y a medida que avanzaba la escritura. Los escritores integran intencionalmente conceptos transversales en las declaraciones de concepto durante el proceso de escritura. Hacia el final del proceso de desarrollo, un pequeño equipo de escritores revisó todo el marco y etiquetó las declaraciones de concepto por conceptos transversales particulares y también sugirió revisiones a las declaraciones de concepto para crear una integración más fuerte y explícita con un concepto transversal. Los conceptos transversales enumerados en el material descriptivo para cada declaración de concepto no son las únicas conexiones que se pueden hacer, solo las que son más relevantes. Además, algunos conceptos transversales son más obvios que otros, pero todos ofrecen oportunidades para iluminar temas clave en ciencias de la computación que abarcan los cinco conceptos core del marco. Los usuarios del marco deben decidir la profundidad a la que se enfatiza un concepto transversal cuando se aborda una declaración de concepto.

Los siguientes ejemplos ilustran el concepto transversal, relaciones de sistemas, en declaraciones de diferentes áreas de conceptos core. Una expectativa para el final del grado 12 en el concepto core de Algoritmos y Programación comienza: "Los programas complejos están diseñados como sistemas de módulos interactivos, cada uno con un rol específico, coordinándose para un propósito general común. Estos módulos pueden ser procedimientos dentro de un programa; combinaciones de datos y procedimientos; o programas independientes, pero interrelacionados " (9–12. Algoritmos y programación.Modularidad). Este ejemplo ilustra cómo diferentes partes de un programa crean

un sistema que interactúa para un propósito común. En el concepto básico de Datos y Análisis, al final del grado 12, se espera que los estudiantes entiendan lo siguiente: "Los datos pueden estar compuestos de múltiples elementos de datos que se relacionan entre sí.

Las personas toman decisiones sobre cómo se organizan los elementos de los datos y dónde se almacenan los datos (9–12. Datos y análisis.Almacenamiento). Esta declaración aplica la idea de sistemas a las organizaciones de datos y las relaciones entre diferentes elementos de datos. Lo más obvio es que la idea de sistemas aparece a menudo en el concepto core de las Sistemas de Computación: "Existen niveles de interacción entre el hardware, el software y el usuario de un sistema de computación. Los niveles más comunes de software con los que un usuario interactúa incluyen el software y las aplicaciones del sistema "(9–12. Sistemas de Computación. Hardware y software). Esta declaración describe la función del software del sistema, así como las relaciones entre el hardware, el software y el usuario en un sistema de computación. Estos tres ejemplos muestran el poder de un concepto transversal para iluminar un aspecto clave de las ciencias de la computación en todos los conceptos core del marco.

Las descripciones detalladas de cada concepto transversal se pueden encontrar en el prefacio del capítulo **Conceptos**.

Conexiones Dentro del Marco

Muchas declaraciones de conceptos en el marco se relacionan con otras declaraciones de conceptos, ya sea en la misma banda de grados o en una diferente. Los equipos de redacción asignados a conceptos particulares trabajaron intencionalmente con otros equipos para alinear el contenido en todo el marco. Las frases en negrita en la Figura 3.2 resaltan la relación entre la declaración del concepto en la banda de grado 9-12 del concepto core de Redes e Internet (jerarquía en Internet) y la declaración en la misma banda de grado del concepto central de los sistemas de computación (capas de interacción en hardware y software).

Figura 3.2: Ejemplo de conexión entre dos conceptos en la misma banda de grado.

9–12. Redes e internet.Red de comunicación y organización	9–12. Sistemas de computación.Hardware y Software
La topología de la red está determinada, en parte, por la cantidad de dispositivos que se pueden admitir. A cada dispositivo se le asigna una dirección que lo identifica de forma única en la red. La escalabilidad y confiabilidad de Internet están habilitadas por la **jerarquía** y la redundancia en las redes.	**Existen niveles de interacción entre el hardware, el software y el usuario de un sistema de computación**. Los niveles más comunes de software con los que un usuario interactúa incluyen el software y las aplicaciones del sistema. El software del sistema controla el flujo de información entre los componentes de hardware utilizados para la entrada, salida, almacenamiento y procesamiento.

Las conexiones también existen entre declaraciones en diferentes bandas de grado y describen cómo un concepto puede construirse a otro. Por ejemplo, la declaración de concepto al final de la banda de grado K-2 del concepto core, Algoritmos y Programación, los programas se desarrollan para expresar ideas y abordar problemas y proporcionar una base para comprender una declaración del concepto core de los impactos de la computación al final de la banda de grado (3-5) influenciada por las prácticas culturales. Ver la Figura 3.3 para detalles.

Figura 3.3: Ejemplo de conexión entre dos conceptos en diferentes bandas de grado

K–2.Algoritmos y programación.Desarrollo del programa Las personas desarrollan programas en colaboración y con un propósito, **tales como expresar ideas o abordar problemas**.	3–5. Impactos de la computación.Cultura **El desarrollo y la modificación de la tecnología de las ciencias de la computación están impulsados por las necesidades y deseos de las personas** y pueden afectar a los grupos de manera diferente**. Las tecnologías de las ciencias de la computación influyen, y son influenciadas por las prácticas culturales.**

Las declaraciones de concepto dentro de la misma banda de grado y el concepto core están inherentemente conectadas. Por ejemplo, las declaraciones de los conceptos de Modularidad y desarrollo de programas para la banda de grado 3–5 en Algoritmos y Programación están relacionadas (ver Figura 3.4).

Figura 3.4: Ejemplo de conexión entre dos declaraciones en el mismo concepto central y banda de grado

3–5. Algoritmos y Programación. Modularidad Los programas se pueden dividir en partes más pequeñas para facilitar su diseño, implementación y revisión. **Los programas también se pueden crear incorporando porciones más pequeñas de programas que ya se han creado.**	3–5. Algoritmos y Programación. Desarrollo de Programas. Las personas desarrollan programas utilizando un proceso iterativo que involucra diseño, implementación y revisión. **El diseño a menudo implica reutilizar el código existente o remezclar otros programas dentro de una comunidad**. La gente revisa continuamente si los programas funcionan como se espera, y arreglan, o depuran, las partes que no lo hacen. La repetición de estos pasos permite a las personas perfeccionar y mejorar los programas.

Prácticas Core

Las prácticas del marco de las ciencias de la computación K-12 son los comportamientos que los estudiantes con conocimientos de computación utilizan para involucrarse completamente con los conceptos core de la ciencia de la computación. Los conceptos y las prácticas se integran para proporcionar experiencias completas para los estudiantes que participan en ciencias de la computación. El criterio para la selección de una práctica fue que debía

- capturar comportamientos importantes en los que se involucran los científicos de las ciencias de la computación,
- ser útil para explorar y comprender completamente los conceptos del marco,
- ayudar a los estudiantes a participar con el contenido del curso a través del desarrollo de artefactos, y
- basarse en procesos y competencias con importancia en ciencias de la computación.

Al igual que los conceptos core del marco, las prácticas de los marcos también fueron informadas por las descripciones de las prácticas en los marcos y estándares nacionales en otros temas. Las prácticas se superponen intencionalmente con las de otras disciplinas y utilizan un lenguaje similar para ayudar a los profesores a establecer conexiones entre las ciencias de la computación y las disciplinas con las que están más familiarizados y hacer que el marco sea más accesible para una amplia audiencia.

Las prácticas se superponen intencionalmente con las de otras disciplinas y utilizan un lenguaje similar.

Las prácticas core del marco:

1. Fomentar de una cultura de las ciencias de la computación inclusiva.
2. Colaborar en torno a las ciencias de la computación.
3. Reconocer y definir problemas computacionales.
4. Desarrollar y usar abstracciones.
5. Crear artefactos computacionales.
6. Probar y refinar los artefactos computacionales.
7. Comunicar sobre la computación.

El pensamiento computacional desempeña un papel clave en las prácticas de las ciencias de la computación del marco, ya que abarca las prácticas 3, 4, 5 y 6. Las prácticas 1, 2 y 7 son prácticas generales e independientes en ciencias de la computación que complementan el pensamiento computacional. Múltiples artículos y documentos de investigación informaron sobre la delineación de las prácticas de pensamiento computacional, tal como la definición operacional del pensamiento computacional para la educación K–12 (ISTE y CSTA, 2011) y los patrones de diseño de evaluación para las prácticas de pensamiento computacional en ciencias de la computación secundaria. (Bienkowski, Snow, Rutstein, & Grover, 2015).

Las declaraciones de práctica delinean expectativas específicas para el final del grado 12 y son seguidas por una narrativa que describe la progresión que conduce a esos puntos finales. Esta estructura difiere de la delineación de las bandas de grado de los conceptos core porque la base de investigación actual y la experiencia de los profesionales con las prácticas en ciencias de la computación no brindan suficiente información para crear expectativas claras y con diferentes bandas de grado. Las narraciones describen las progresiones de la práctica de una manera que es menos prescriptiva sobre la adecuación del desarrollo para enfatizar expectativas flexibles.

Referencias

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look* (SRI technical report). Menlo Park, CA: SRI International. Retrieved from http://pact.sri.com/resources.html

College Board. (2016). *AP Computer Science Principles course and exam description.* New York, NY: College Board. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

Computer Science Teachers Association Standards Task Force. (2011). *CSTA K–12 computer science standards, revised 2011.* New York, NY: Computer Science Teachers Association and Association for Computing Machinery. Retrieved from http://www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf

Denning, P. J., & Martell, C. (2015). *Great principles of computing.* Cambridge, MA: MIT Press.

England Department for Education. (2013, September 11). *National curriculum in England: Computing programmes of study.* Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

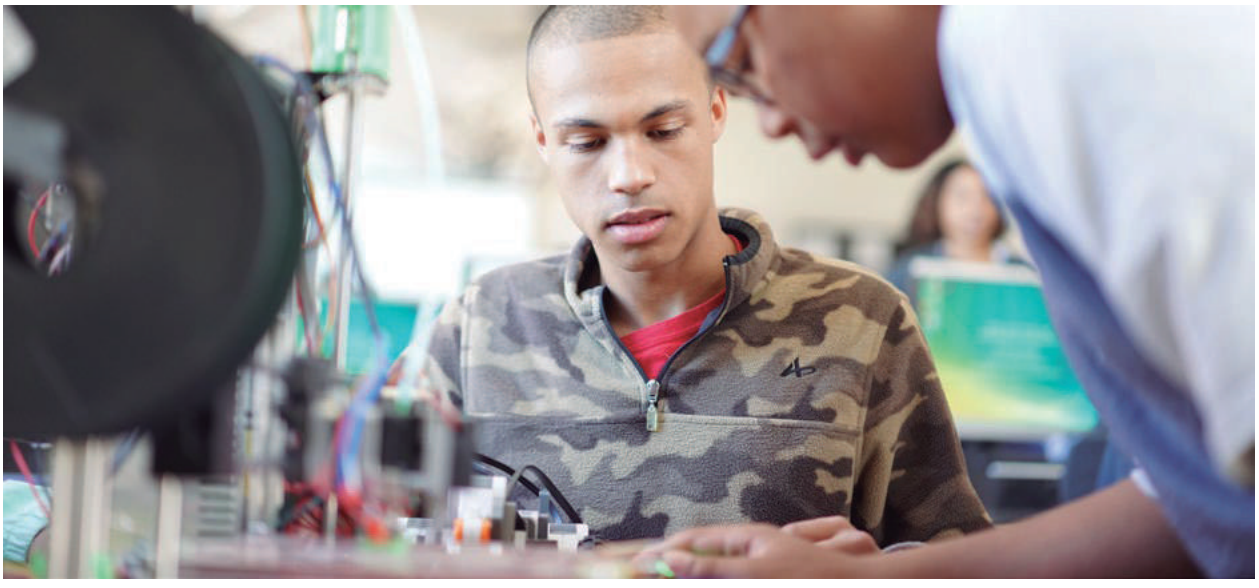
Hess, K. (2008). *Developing and using learning progressions as a schema for measuring progress.* Dover, NH: National Center for the Improvement of Educational Assessment. Retrieved from http://www.nciea.org/publications/CCSSO2_KH08.pdf

International Society for Technology in Education & Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K–12 education.* Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

National Research Council. (2007). *Taking science to school: Learning and teaching science in grades K–8.* Committee on Science Learning-Kindergarten Through Eighth Grade. R. A. Duschl, H. A. Schweingruber, & A. W. Shouse (Eds.). Board on Science Education, Center for Education. Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.



Navegando el Marco

4

Navegando el Marco

El propósito de este capítulo es ayudarlo a navegar las partes del marco. Primero, este capítulo describe el contenido del marco, que se compone principalmente de prácticas, conceptos y capítulos de orientación. Segundo, este capítulo explica las diferentes formas en que puede ver el marco en línea para que pueda organizar la información de acuerdo con su propósito.

Practicas Core

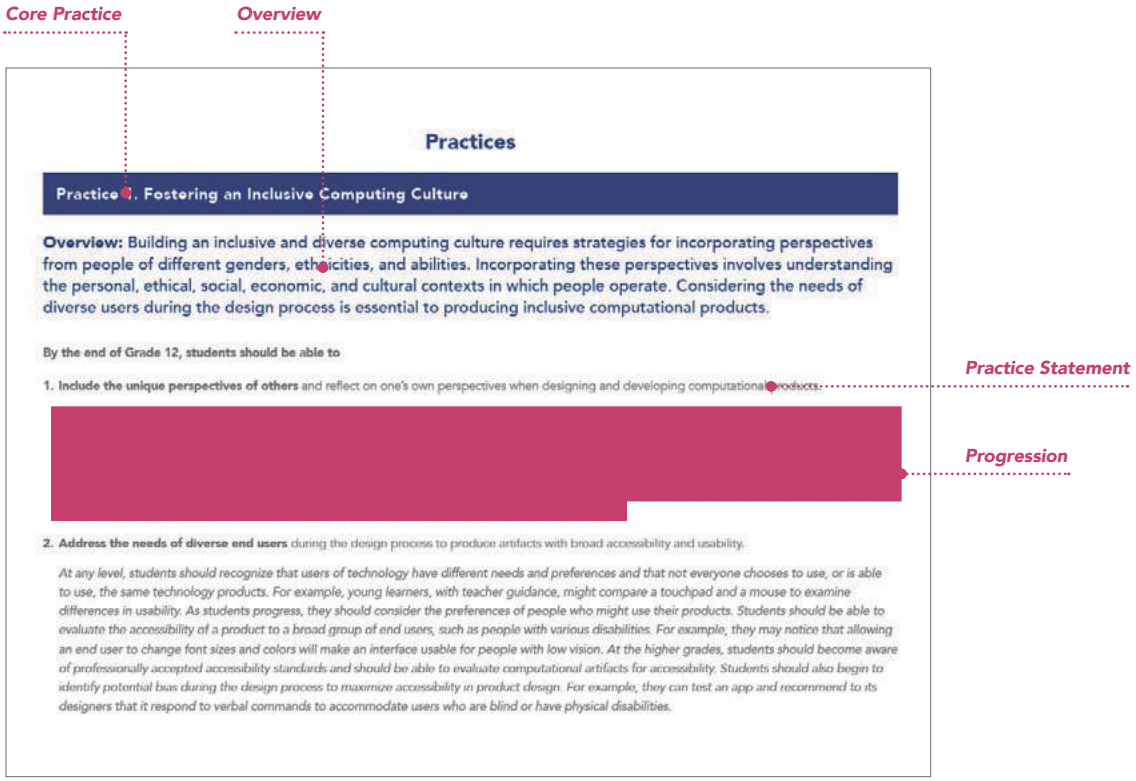
Las prácticas del marco de las ciencias de la computación K-12 son los comportamientos que los estudiantes con conocimientos de computación utilizan para involucrarse completamente con los conceptos core de las ciencias de la computación. Los conceptos y las prácticas se integran para proporcionar experiencias completas para los estudiantes que participan en ciencias de la computación. Si bien las prácticas se integran y se superponen entre sí de manera natural, se muestran en un orden que sugiere un proceso para desarrollar artefactos computacionales. Cuatro de las prácticas también se mencionan como aspectos del pensamiento computacional.

Hay siete **prácticas core**:

1. Fomentar de una cultura de las ciencias de la computación inclusiva.
2. Colaborar en torno a la computación.
3. Reconocer y definir problemas computacionales.
4. Desarrollar y usar de abstracciones.
5. Crear artefactos computacionales.
6. Probar y refinar de artefactos computacionales.
7. Comunicar acerca de la computación.

Cómo leer las prácticas

Figura 4.1: Cómo leer las prácticas



Cada práctica contiene tres partes: (1) **resumen**, (2) **declaración de práctica** y (3) **progresión**.

1. El **resumen** describe la práctica.
2. La **declaración de la práctica** describe lo que los alumnos deberían poder hacer al salir del grado 12.
3. La **progresión** bajo cada meta describe cómo los estudiantes deben exhibir la práctica específica con una sofisticación cada vez mayor desde el jardín de infantes hasta el grado 12. En lugar de bandas de grado, las progresiones usan un formato narrativo para enfatizar los diferentes caminos que pueden tomar los estudiantes en el desarrollo de las prácticas. Los ejemplos en las progresiones describen lo que todos los estudiantes podrían hacer pero no son obligatorios.

Cómo referirse a las prácticas

Cuando se refiera a una declaración de práctica particular, use la siguiente notación:

P[Número de práctica].[Práctica].[Número de declaración de práctica]

Ejemplos:

- P4.Desarrollar y usar de abstracciones.1
- P2.Colaborar en torno a la computación.3

Cuando se hace referencia a una declaración de práctica dentro de la narrativa de otra práctica, se denota sin el nombre de la práctica principal (por ejemplo, P4.1).

Conceptos

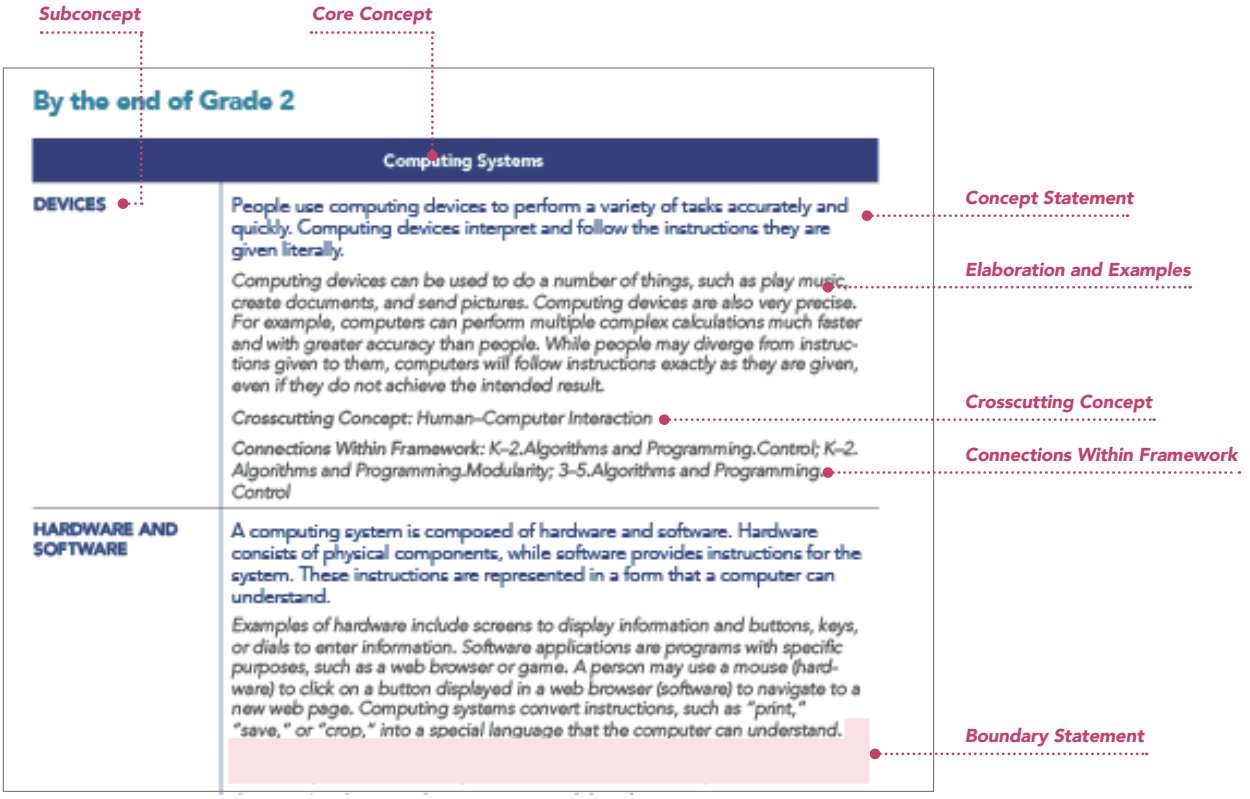
Un concepto core representa un área específica de importancia disciplinaria en ciencias de la computación. Hay **cinco conceptos core**:

1. Sistemas de computación
2. Redes e internet
3. Datos y análisis
4. Algoritmos y programación
5. Impactos de la computación

Cada concepto core se describe en un resumen y está delineado por múltiples s**ubconceptos** que representan las ideas específicas dentro de ese concepto core. Por ejemplo, el concepto core de Datos y Análisis contiene cuatro subconceptos: Colección, Almacenamiento, Visualización y Transformación, e Inferencia y Modelos. Se proporcionan resúmenes de subconjuntos para describir los **subconceptos** y resumir cómo progresa el aprendizaje en múltiples bandas de grado.

Cómo leer los conceptos

Figure 4.2: Cómo leer los conceptos



Las **declaraciones de concepto** en el marco describen hitos conceptuales en diferentes puntos finales de la banda de grado: Grados 2, 5, 8, y 12. Cada declaración de concepto abarca una idea importante y esencial en ciencias de la computación, y todos son considerados igualmente importantes.

Cada declaración de concepto se acompaña de material descriptivo, que incluye la **elaboración y ejemplos**. Algunas declaraciones de conceptos también incluyen **conceptos transversales y conexiones dentro del marco**.

1. La **elaboración y los ejemplos** agregan detalle y profundidad a las declaraciones de concepto. Se incluyen **declaraciones de límites** para aclarar lo que no se espera que se aprenda en ese nivel de grado.
2. **Conceptos transversales** ilumina las conexiones temáticas a través de los diferentes conceptos core y se integran en las declaraciones de conceptos según sea relevante y apropiado. Cuando varios conceptos transversales se enumeran en una declaración, están en orden de importancia en lugar de orden alfabético.
   Hay cinco conceptos transversales:
   1. Abstracción
   2. Relaciones de sistemas
   3. Interacción humano-computadora
   4. Privacidad y seguridad
   5. Comunicación y coordinación
3. **Conexiones dentro del marco** se proporcionan para iluminar las relaciones entre las declaraciones de concepto a través de diferentes conceptos core. Las conexiones en la misma banda de grado ayudan a guiar cuando los conceptos se pueden abordar juntos.

Cómo referirse a los conceptos

Al referirse a una declaración de concepto particular, use la siguiente notación: [Banda de Grado].[Concepto].[Subconcepto]

Ejemplos:

- 3–5. Impactos de la computación.Cultura
- K–2.Algoritmos y Programación.Desarrollo de Programas

Otras partes del marco

El documento del marco completo consta de conceptos, prácticas y capítulos de orientación. El documento del marco también incluye un glosario de términos técnicos clave utilizados en las declaraciones de conceptos y prácticas (**Apéndice C**). Otros recursos están disponibles en el sitio web, como folletos, y se publicarán más a medida que se crean.

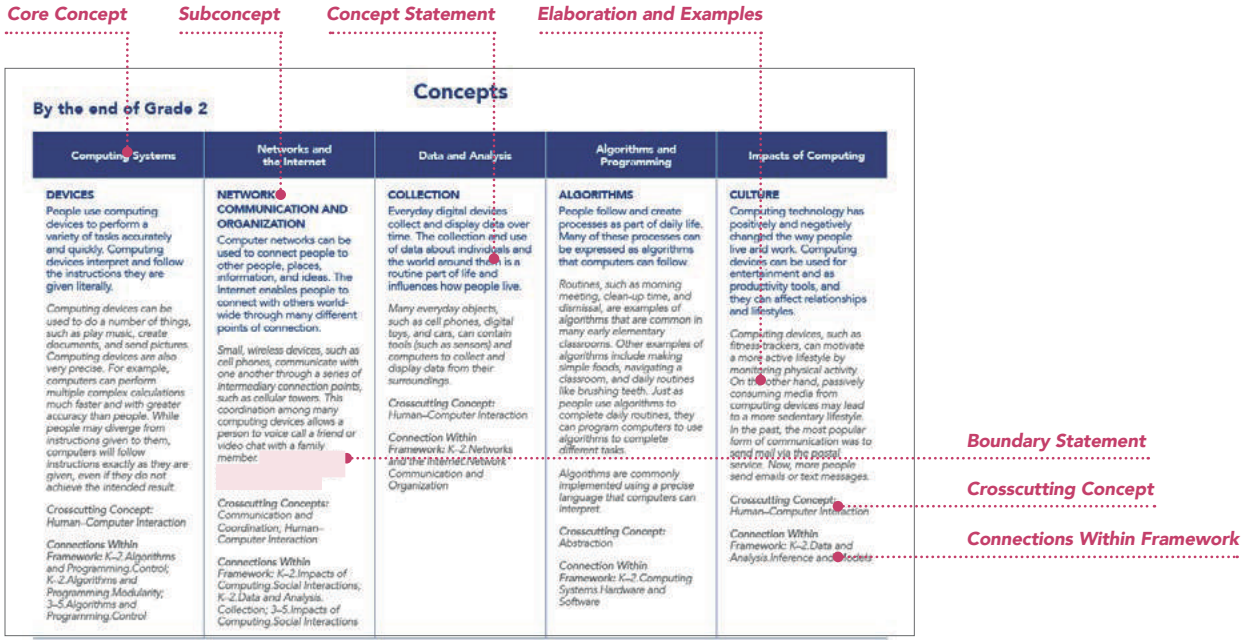
Viendo el Marco

Se espera que las personas que vean el marco tengan diferentes objetivos. Por lo tanto, el marco se puede ver en línea en una variedad de formas para satisfacer sus necesidades.

Los conceptos y prácticas se pueden ver en línea en tres vistas diferentes. Las tres vistas incluyen las prácticas primero y luego muestran los conceptos según la vista seleccionada. Las tres vistas también permiten al usuario filtrar por bandas de grado, conceptos core, prácticas core y conceptos transversales.

Vista de las bandas de grado

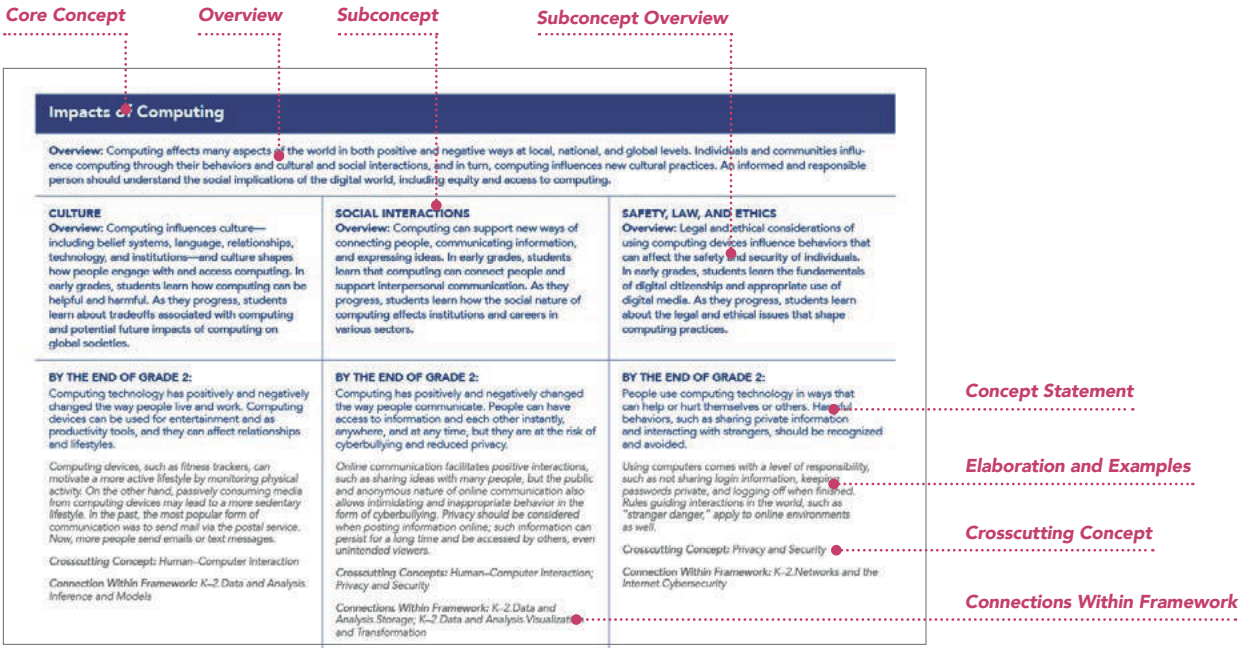
Figura 4.3: Vista de las bandas de grado



Ver por banda de grado le permite ver las declaraciones por banda de grado primero. Bajo cada banda de grado, las declaraciones de concepto se organizan por concepto core y se enumeran bajo el subconcepto asociado. Esta organización es útil para un usuario que desea ver una banda de un solo grado, tal como solo K–2 o solo 6–8, por ejemplo.

Vista de progresión

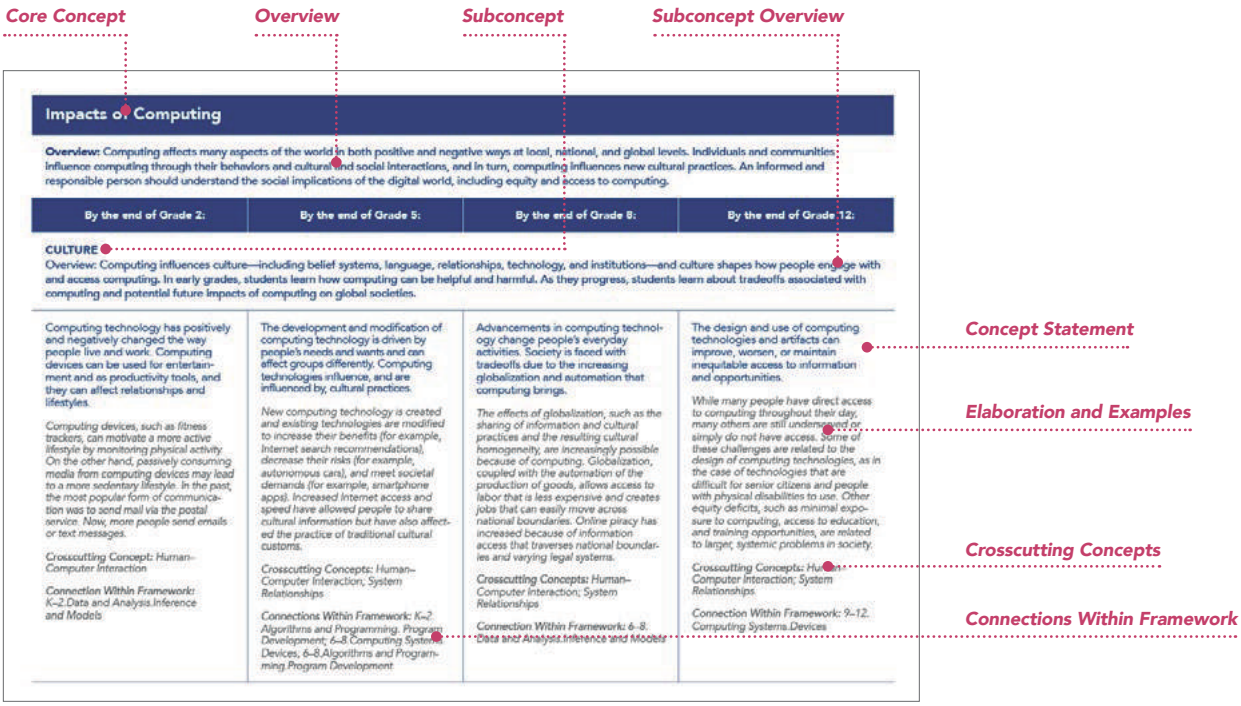
Figure 4.4: Vista de progresión



La visualización por progresión le permite ver las declaraciones de concepto organizadas por concepto core primero y luego por subconcepto. Esta vista también muestra los resúmenes de los conceptos core y los subconceptos. Esta vista es buena para ver el progreso de la comprensión de los estudiantes en las bandas de grado.

Vista Conceptual

Figure 4.5: Vista Conceptual



La visualización por concepto le permite ver las declaraciones organizadas por concepto core primero y luego en la banda de grado. Esta vista es útil para obtener una imagen de todo lo que los estudiantes en un nivel de grado particular necesitarían saber en un concepto core determinado; por ejemplo, lo que un estudiante en los grados 3–5 debería saber en el concepto core de Sistemas de Computación.

Accediendo al marco completo

El documento marco completo, con conceptos, prácticas y capítulos de orientación, está disponible como descarga en k12cs.org. Los capítulos de orientación del marco pueden verse en línea en un formato abreviado.

Prácticas

Incluyendo Pensamiento Computacional

5

Prácticas *Incluyendo Pensamiento Computacional*

Prefacio

Las siete prácticas core de las ciencias de la computación describen los comportamientos y formas de pensar que los estudiantes con conocimientos de computación utilizan para participar plenamente en el mundo interconectado y rico en datos de hoy. Las prácticas se integran naturalmente entre sí y contienen un lenguaje que se superpone intencionalmente para iluminar las conexiones entre ellas. Se muestran en un orden que sugiere un proceso para desarrollar artefactos computacionales. Este proceso es cíclico y puede seguir muchos caminos; en el marco, comienza con el reconocimiento de diversos usuarios y la valoración de las perspectivas de otros y termina con la comunicación de los resultados a un público amplio (ver Figura 5.1).

A diferencia de los conceptos core, las prácticas no están delineadas por bandas de grado. Más bien, las prácticas usan una narrativa para describir cómo los estudiantes deben exhibir cada práctica con una sofisticación cada vez mayor desde el jardín de infantes hasta el grado 12. Además de describir la progresión, estas narrativas también proporcionan algunos ejemplos de la interrelación de las declaraciones de práctica y las formas en que las declaraciones se construyen unas sobre otras.

Pensamiento computacional

El pensamiento computacional está en el corazón de las prácticas de las ciencias de la computación y está delineado por las prácticas 3–6. Las prácticas 1, 2 y 7 son prácticas generales e independientes en ciencias de la computación que complementan el pensamiento computacional.

Figura 5.1: Prácticas core que incluyen el pensamiento computacional

Definir el pensamiento computacional

El pensamiento computacional se refiere a los procesos de pensamiento implicados en la expresión de soluciones como pasos computacionales o algoritmos que pueden realizarse por computadora (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016). Esta definición se basa en la idea de formular problemas y soluciones en una forma que puede ser realizada por un agente de procesamiento de información (Cuny, Snyder y Wing, 2010) y la idea de que las soluciones deben tomar la forma específica de pasos computacionales y los algoritmos serán ejecutados por una computadora (Aho, 2011; Lee, 2016). El pensamiento computacional requiere comprensión de

las capacidades de las computadoras, la formulación de problemas para ser abordados por una computadora y el diseño de algoritmos que una computadora puede ejecutar. El contexto y enfoque más efectivo para desarrollar el pensamiento computacional es aprender las ciencias de la computación; están intrínsecamente conectados.

El pensamiento computacional es esencialmente un proceso de resolución de problemas que implica diseñar soluciones que aprovechen el poder de las computadoras; este proceso comienza antes de que se escriba una sola línea de código. Las computadoras brindan beneficios en términos de memoria, velocidad y precisión de ejecución. Las computadoras también requieren que las personas expresen su pensamiento en una estructura formal, tal como un lenguaje de programación. Al igual que escribir notas en un pedazo de papel para "expresar sus pensamientos," crear un programa permite a las personas externalizar sus pensamientos en una forma que puede ser manipulada y analizada. La programación les permite a los estudiantes pensar sobre su pensamiento; al depurar un programa, los estudiantes depuran su propio pensamiento (Papert, 1980).

El pensamiento computacional se refiere a los procesos de pensamiento involucrados en la expresión de soluciones como pasos computacionales o algoritmos que pueden ser realizados por una computadora.

A pesar de lo que su nombre implica, el pensamiento computacional es fundamentalmente una habilidad humana. En otras palabras, "[h]umanos procesa información; los humanos computan "(Wing, 2008, p. 3718). Este matiz es la base para los enfoques "desconectados" de las ciencias de la computación (es decir, la enseñanza de las ciencias de la computación sin las computadoras) y explica cómo el pensamiento computacional puede aplicarse más allá de los límites de las ciencias de la computación a una variedad de disciplinas, como la ciencia, la tecnología, la ingeniería y Matemáticas (STEM), pero también las artes y las humanidades (Bundy, 2007).

Distinguir el pensamiento computacional

La descripción del pensamiento computacional en el Marco de Ciencias de la Computación K–12 se extiende más allá del uso general de las computadoras o la tecnología en la educación para incluir habilidades específicas, tales como el diseño de algoritmos, problemas de descomposición y fenómenos de modelado. Si el pensamiento computacional puede ocurrir sin una computadora, a la inversa, usar una computadora en clase no necesariamente constituye un pensamiento computacional. Por ejemplo, un estudiante no está necesariamente usando el pensamiento computacional cuando ingresa datos en una hoja de cálculo y crea un gráfico. Sin embargo, esta acción puede incluir pensamiento computacional si el estudiante crea algoritmos para automatizar la transformación de los datos o para alimentar una visualización interactiva de datos.

Un artefacto computacional debe distinguirse mediante la evaluación del proceso utilizado para crearlo (es decir, el pensamiento computacional), además de las características del producto en sí. Por ejemplo, la misma animación digital puede ser el resultado de construir cuidadosamente algoritmos que controlan cuándo se mueven los personajes y cómo interactúan, o simplemente seleccionando personajes y acciones de una plantilla predeterminada. En este ejemplo, es el proceso utilizado para crear la animación lo que define si se puede considerar un artefacto computacional. La evaluación del pensamiento computacional se puede mejorar haciendo que los estudiantes expliquen sus decisiones y el proceso de desarrollo (Brennan y Resnick, 2012).

Prácticas de las ciencias de la computación y otras áreas temáticas

El marco se basa en la creencia de que la ciencia de la computación ofrece oportunidades únicas para desarrollar el pensamiento computacional y que las prácticas del marco se pueden aplicar a otros temas más allá de las ciencias de la computación. Como han señalado Barr y Stephenson (2011), "la comunidad de educación en ciencias de la computación puede desempeñar un papel importante al resaltar las prácticas algorítmicas de resolución de problemas y las aplicaciones de las ciencias de la computación en todas las disciplinas y ayudar a integrar la aplicación de métodos y herramientas computacionales en diversas áreas de aprendizaje". (p. 49).

> ***El pensamiento computacional es una habilidad fundamental para todos, no solo para los científicos de la computación. A la lectura, la escritura y la aritmética, debemos agregar el pensamiento computacional a la capacidad analítica de cada niño (Wing, 2006, p. 33).***

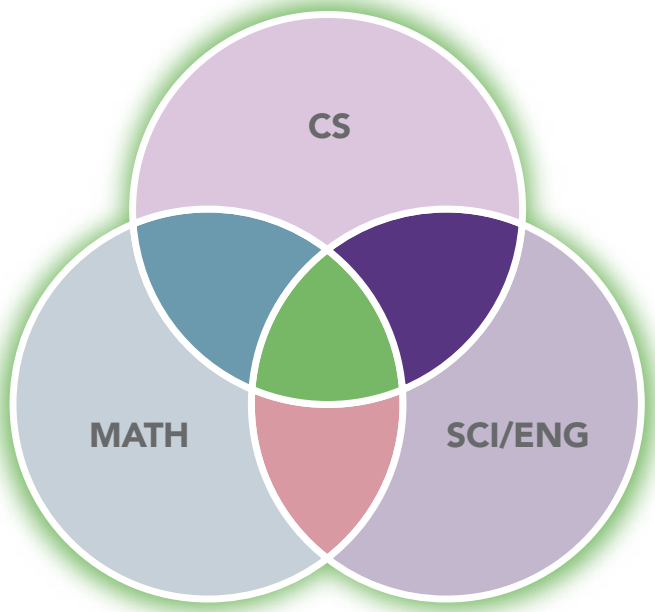
Si bien el pensamiento computacional es un enfoque en ciencias de la computación, también se incluye en los estándares para otras materias. Por ejemplo, se hace referencia explícita al pensamiento computacional en las prácticas de muchos estándares estatales de ciencia[1] e implícitamente en los estándares estatales de matemáticas.[2] Además, la reciente revisión de la Sociedad Internacional de Tecnología en Estándares de Educación para Estudiantes (ISTE, 2016) describe el pensamiento computacional en una manera similar al marco. Todos estos documentos comparten la visión de que el pensamiento computacional es importante para todos los estudiantes. La Figura 5.2 en la página siguiente describe la intersección entre las prácticas en ciencias de la computación, ciencias e ingeniería y matemáticas. Se requieren instrucciones explícitas para crear las conexiones ilustradas en la figura.

Reconocimiento

El equipo de redacción agradece al equipo de trabajo de pensamiento computacional de la Asociación de Profesores de las Ciencias de la Computación por su contribución a esta sección.

[1] Práctica 5: Using Mathematics and Computational Thinking (Next Generation Science Standards Lead States, 2013).
[2] CCSS.Math.Practice.MP2. Reason abstractly and quantitatively (NGA Center for Best Practices & CCSSO, 2010).

Figura 5.2: Relaciones entre las ciencias de la computación, ciencia e ingeniería, y prácticas matemáticas:



CS + Math

- **Desarrollar y utilizar abstracciones**.
  M2. Razonar de manera abstracta y cuantitativa.
  M7. Buscar y hacer uso de la estructura.
  M8. Buscar y expresar regularidad en razonamientos repetidos.
  CS4. Desarrollo y uso de abstracciones.

- **Utilizar herramientas al colaborar**.
  M5. Usar herramientas apropiadas estratégicamente.
  CS2. Colaborando en torno a la computación.

- **Comunicarse con precisión**.
  M6. Atender a la precisión.
  CS7. Comunicar acerca de la computación

CS + Sci/Eng

- **Comunicarse con los datos.**
  S4. Analizar e interpretar datos.
  CS7. Comunicar acerca de la computación

- **Crear artefactos.**
  S3. Planificar y llevar a cabo investigaciones.
  S6. Construir explicaciones y soluciones de diseño.
  CS4. Desarrollo y uso de abstracciones.
  CS5. Creando artefactos computacionales.
  CS6. Pruebas y refinación de artefactos computacionales.

CS + Math + Sci/Eng

- **Modelo**.
  S2. Desarrollar y utilizar modelos.
  M4. Modelar con matemáticas
  CS4. Desarrollo y uso de abstracciones.
  CS6. Pruebas y refinación de artefactos computacionales .

- **Usar el pensamiento computacional.**
  S5. Utilizar las matemáticas y el pensamiento computacional.
  CS3. Reconociendo y definiendo problemas computacionales
  CS4. Desarrollo y uso de abstracciones.
  CS5. Creando artefactos computacionales.

- **Definir problemas.**
  S1. Hacer preguntas y definir problemas.
  M1. Darles sentido a los problemas y perseverar en resolverlos.
  CS3. Reconocer y definir problemas computacionales .

- **Razonar la comunicación.**
  S7. Participar en el argumento de la evidencia
  S8. Obtener, evaluar y comunicar información.
  M3. Construir argumentos viables y critica el razonamiento de los demás.
  CS7. Comunicar acerca de la computación.

* Las prácticas de las ciencias de la computación también se superponen con las prácticas en otros dominios, incluidas las artes del lenguaje en inglés. Por ejemplo, CS1. Fomento de una Cultura Informática Inclusiva y CS2. Colaborando en torno a la computación se superponen con E7. Llegue a comprender otras perspectivas y culturas a través de la lectura, la escucha y las colaboraciones.

Referencias

Aho, A.V. (2011, January) Computation and Computational Thinking. *ACM Ubiquity*, 1, 1-8.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*, 48–54.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Paper presented at the annual meeting of the American Educational Research Association, Vancouver, BC, Canada.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing, 1*, 67–69.

Cuny, J., Snyder, L., & Wing, J.M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress. Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

International Society for Technology in Education. (2016). *ISTE standards for students.* Retrieved from https://www.iste.org/resources/product?id=3879&childProduct=3848

Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12*(1), 3–4. Retrieved from http://www.csteachers.org/resource/resmgr/Voice/csta_voice_03_2016.pdf

National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common core state standards for mathematics.* Washington DC: Author.

Next Generation Science Standards Lead States. (2013). *Next generation science standards: For states, by states.* Washington, DC: The National Academies Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* NY: Basic Books.

Wing, J. M. (2006, March). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366*(1881), 3717–3725.

Prácticas

Práctica 1. Fomentar una cultura computacional inclusiva:

Resumen: La construcción de una cultura de las ciencias de la computación inclusiva y diversa requiere estrategias para incorporar perspectivas de personas de diferentes géneros, etnias y habilidades. Incorporar estas perspectivas implica comprender los contextos personales, éticos, sociales, económicos y culturales en los que operan las personas. Tener en cuenta las necesidades de los diversos usuarios durante el proceso de diseño es esencial para producir productos computacionales inclusivos.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Incluir las perspectivas únicas de los demás** *y reflexione sobre las propias perspectivas al diseñar y desarrollar productos computacionales.*
   En todos los niveles de grado, los estudiantes deben reconocer que las decisiones que toman las personas cuando crean artefactos se basan en intereses, experiencias y necesidades personales. Los jóvenes aprendices deben comenzar a diferenciar sus preferencias tecnológicas de las preferencias tecnológicas de otros. Inicialmente, los estudiantes deben ser presentados con perspectivas de personas con diferentes antecedentes, niveles de habilidad y puntos de vista. A medida que los estudiantes progresan, deben buscar independientemente perspectivas diversas a lo largo del proceso de diseño con el fin de mejorar sus artefactos computacionales. Los estudiantes que están bien versados en el fomento de una cultura de computación inclusiva deben poder diferenciar fondos y habilidades y saber cuándo llamar a otros, cómo buscar conocimiento sobre posibles usuarios finales o buscar intencionalmente información de personas con antecedentes diversos.

2. **Atender las necesidades de los diversos usuarios** *finales durante el proceso de diseño para producir artefactos con amplia accesibilidad y facilidad de uso.*
   En cualquier nivel, los estudiantes deben reconocer que los usuarios de la tecnología tienen diferentes necesidades y preferencias y que no todos eligen o pueden usar los mismos productos tecnológicos. Por ejemplo, los jóvenes estudiantes, con la guía del profesor, pueden comparar un panel táctil y un mouse para examinar las diferencias en la facilidad de uso. A medida que los estudiantes progresan, deben considerar las preferencias de las personas que podrían usar sus productos. Los estudiantes deben poder evaluar la accesibilidad de un producto a un amplio grupo de usuarios finales, como personas con diversas discapacidades. Por ejemplo, pueden notar que permitir que un usuario final cambie el tamaño y el color de las fuentes hará que una interfaz sea utilizable para personas con baja visión. En los grados superiores, los estudiantes deben conocer los estándares de accesibilidad aceptados profesionalmente y deben poder evaluar los artefactos computacionales para la accesibilidad. Los estudiantes también deben comenzar a identificar posibles sesgos durante el proceso de diseño para maximizar la accesibilidad en el diseño del producto. Por ejemplo, pueden probar una aplicación y recomendar a sus diseñadores que respondan a comandos verbales para adaptarse a los usuarios ciegos o con discapacidades físicas.

3. **Emplear la defensa propia y de los compañeros** *para abordar el sesgo en las interacciones, el diseño de productos y los métodos de desarrollo.*
   Después de que los alumnos tengan experiencia en la identificación de diversas perspectivas e incluyan perspectivas únicas (P1.1), deberían comenzar a emplear estrategias de autodefensa, tal como hablar por sí mismos si no se satisfacen sus necesidades. A medida que los estudiantes progresan, deben abogar por sus compañeros cuando se necesitan adaptaciones, tal como un dispositivo periférico de tecnología asistencial, para que alguien use un artefacto computacional. Eventualmente, los estudiantes deben abogar regularmente por ellos mismos y por los demás.

Práctica 2. Colaborar en torno a la computación:

Resumen: La computación colaborativa es el proceso de realizar una tarea computacional trabajando en parejas y en equipos. Debido a que implica solicitar las contribuciones y comentarios de otros, la colaboración efectiva puede conducir a mejores resultados que el trabajo independiente. La colaboración requiere que los individuos naveguen e incorporen perspectivas diversas, ideas en conflicto, habilidades dispares y personalidades distintas. Los estudiantes deben usar herramientas de colaboración para trabajar juntos de manera efectiva y para crear artefactos complejos.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Cultivar relaciones de trabajo con individuos que posean perspectivas, habilidades y personalidades diversas.**
   En cualquier nivel de grado, los estudiantes deben trabajar en colaboración con otros. Al principio, deben aprender estrategias para trabajar con miembros del equipo que poseen varias fortalezas individuales. Por ejemplo, con el apoyo del profesor, los estudiantes deben comenzar a brindar a cada miembro del equipo la oportunidad de contribuir y trabajar juntos como compañeros de aprendizaje. Eventualmente, los estudiantes deben ser más sofisticados al aplicar estrategias para el estímulo y el apoyo mutuos. Deben expresar sus ideas con un razonamiento lógico y encontrar formas de reconciliar las diferencias cooperativamente. Por ejemplo, cuando no están de acuerdo, deben pedir a los demás que expliquen su razonamiento y trabajar juntos para tomar decisiones respetuosas y mutuas. A medida que progresan, los estudiantes deben usar métodos para dar a todos los miembros del grupo la oportunidad de participar. Los estudiantes mayores deben esforzarse por mejorar la eficiencia y la eficacia del equipo mediante la evaluación periódica de la dinámica del grupo. Deben usar múltiples estrategias para hacer que la dinámica de equipo sea más productiva. Por ejemplo, pueden solicitar las opiniones de los miembros del equipo más silencioso, minimizar las interrupciones de los miembros más habladores y dar crédito a los individuos por sus ideas y su trabajo.

2. **Crear estándares del equipo, expectativas y cargas de trabajo equitativas para aumentar la eficiencia y la eficacia.**
   Después de que los alumnos hayan tenido experiencia en cultivar relaciones de trabajo dentro de los equipos (P2.1), deberían adquirir experiencia trabajando en roles de equipo particulares. Al principio, los profesores pueden ayudar a guiar este proceso al proporcionar estructuras de colaboración. Por ejemplo, los estudiantes pueden tomar turnos en diferentes roles en el proyecto, tales como tomador de notas, facilitador o "conductor" de la computadora. A medida que los estudiantes progresan, deberían ser menos dependientes de la asignación de roles del profesores y ser más adeptos a la asignación de roles dentro de sus equipos. Por ejemplo, deben decidir juntos cómo turnarse en diferentes roles. Eventualmente, los estudiantes deben organizar sus propios equipos de forma independiente y crear metas comunes, expectativas y cargas de trabajo equitativas. También deben administrar el flujo de trabajo del proyecto utilizando agendas y cronogramas y deben evaluar el flujo de trabajo para identificar áreas de mejora.

3. **Solicitar e incorporar constructivos a los miembros del equipo y otras partes interesadas (Stakeholders).**
   En cualquier nivel, los estudiantes deben hacer preguntas a otros y escuchar sus opiniones. Al principio, con los andamios de los profesores, los estudiantes deben buscar ayuda y compartir ideas para lograr un propósito en particular. A medida que progresan en la escuela, los estudiantes deben proporcionar y recibir comentarios relacionados con las ciencias de la computación de manera constructiva. Por ejemplo, la programación en pares es un proceso de colaboración que promueve dar y recibir retroalimentación. Los alumnos mayores deben participar en la escucha activa mediante el uso de habilidades de cuestionamiento y deben responder con empatía a los demás. A medida que progresan, los estudiantes deben poder recibir comentarios de múltiples compañeros y deben poder diferenciar opiniones. Con el tiempo, los estudiantes deben buscar colaboradores de diversos entornos. Estos colaboradores pueden incluir usuarios finales, expertos o audiencias generales de foros en línea.

4. **Evaluar y seleccionar herramientas tecnológicas que se pueden utilizar para colaborar en un proyecto.**
   En cualquier nivel, los estudiantes deben poder utilizar herramientas y métodos para colaborar en un proyecto. Por ejemplo, en los primeros grados, los estudiantes podrían intercambiar ideas de forma colaborativa escribiendo en una pizarra. A medida que los alumnos progresan, deben utilizar herramientas de colaboración tecnológica para gestionar el trabajo en equipo, tales como herramientas de intercambio de conocimientos y espacios de proyectos en línea. También deben comenzar a tomar decisiones sobre que herramientas serían las mejores para usar y cuándo usarlas. Eventualmente, los estudiantes deben usar diferentes herramientas y métodos de colaboración para solicitar aportes no solo de los miembros del equipo y compañeros de clase, sino también de otros, tales como los participantes en foros en línea o comunidades locales.

Práctica 3. Reconocer y definir problemas computacionales:

Resumen: La capacidad de reconocer oportunidades apropiadas y valiosas para aplicar la computación es una habilidad que se desarrolla con el tiempo y es fundamental para la computación. Resolver un problema con un enfoque computacional requiere definir el problema, dividirlo en partes y evaluar cada parte para determinar si una solución computacional es apropiada.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Identificar problemas complejos, interdisciplinarios y del mundo real que pueden resolverse computacionalmente.**
   En cualquier nivel, los estudiantes deben ser capaces de identificar problemas que se hayan resuelto computacionalmente. Por ejemplo, los jóvenes estudiantes pueden hablar sobre una tecnología que ha cambiado el mundo, como el correo electrónico o los teléfonos móviles. A medida que avanzan, deben hacer preguntas aclaratorias para comprender si un problema o parte de un problema se puede resolver utilizando un enfoque computacional. Por ejemplo, antes de intentar escribir un algoritmo para ordenar una gran lista de nombres, los estudiantes pueden hacer preguntas sobre cómo se ingresan los nombres y que tipo de clasificación se desea. Los estudiantes mayores deben identificar problemas más complejos que involucran múltiples criterios y restricciones. Eventualmente, los estudiantes deben poder identificar problemas del mundo real que abarcan múltiples disciplinas, como aumentar la seguridad de la bicicleta con la nueva tecnología de casco, y pueden resolverse computacionalmente.

2. **Descomponer problemas complejos del mundo real en subproblemas manejables que podrían integrar soluciones o procedimientos existentes.**
   En cualquier nivel de grado, los estudiantes deben poder dividir los problemas en sus partes componentes. En los niveles iniciales, los estudiantes deben enfocarse en resolver problemas simples. Por ejemplo, en un entorno de programación visual, los estudiantes podrían desglosar (o descomponer) los pasos necesarios para dibujar una forma. A medida que los estudiantes progresan, deben descomponer problemas más grandes en problemas más pequeños manejables. Por ejemplo, los jóvenes estudiantes pueden pensar en una animación como múltiples escenas y, por lo tanto, crear cada escena de forma independiente. Los estudiantes también pueden dividir un programa en subobjetivos: obtener información del usuario, procesar los datos y mostrar el resultado al usuario. Eventualmente, a medida que los estudiantes se enfrentan a problemas complejos del mundo real que abarcan múltiples disciplinas o sistemas sociales, deben descomponer los problemas complejos en subproblemas manejables que podrían resolverse con programas o procedimientos que ya existen. Por ejemplo, los estudiantes podrían crear una aplicación para resolver un problema de la comunidad que se conecta a una base de datos en línea a través de una interfaz de programación de aplicaciones (API).

3. **Evaluar si es apropiado y factible resolver un problema computacionalmente.**
   Después de que los alumnos hayan tenido alguna experiencia en descomponer problemas (P3.2) e identificar subproblemas que puedan resolverse computacionalmente (P3.1), deberían comenzar a evaluar si una solución computacional es la solución más adecuada para un problema en particular. Por ejemplo, los estudiantes podrían preguntarse si usar una computadora para determinar si alguien está diciendo la verdad sería ventajoso. A medida que los estudiantes progresan, deben evaluar sistemáticamente la viabilidad de usar herramientas computacionales para resolver problemas o subproblemas dados, como a través de un análisis de costo-beneficio. Eventualmente, los estudiantes deben incluir más factores en sus evaluaciones, como la forma en que la eficiencia afecta la factibilidad o si el enfoque propuesto plantea inquietudes éticas.

Práctica 4. Desarrollar y usar abstracciones:

Resumen: Las abstracciones se forman identificando patrones y extrayendo características comunes de ejemplos específicos para crear generalizaciones. El uso de soluciones generalizadas y partes de soluciones diseñadas para una reutilización amplia simplifica el proceso de desarrollo al administrar la complejidad.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Extraer características comunes de un conjunto de procesos interrelacionados o fenómenos complejos.**
   Los estudiantes en todos los niveles de grado deben poder reconocer patrones. Los jóvenes aprendices deben poder identificar y describir secuencias repetidas en datos o códigos a través de analogías con patrones visuales o secuencias físicas de objetos. A medida que avanzan, los estudiantes deben identificar los patrones como oportunidades para la abstracción, como reconocer patrones repetidos de código que podrían implementarse de manera más eficiente como un bucle. Eventualmente, los estudiantes deben extraer características comunes de fenómenos o procesos más complejos. Por ejemplo, los estudiantes deben ser capaces de identificar características comunes en múltiples segmentos de código y sustituir un segmento único que utiliza variables para explicar las diferencias. En un procedimiento, las variables tomarían la forma de parámetros. Al trabajar con datos, los estudiantes deben ser capaces de identificar aspectos importantes y encontrar patrones en conjuntos de datos relacionados, como la producción de cultivos, los métodos de fertilización y las condiciones climáticas.

2. **Evaluar las funcionalidades tecnológicas existentes e incorporarlas en nuevos diseños.**
   En todos los niveles, los estudiantes deben poder utilizar abstracciones bien definidas que oculten la complejidad. Al igual que un automóvil oculta los detalles de operación, tal como la mecánica del motor, el comando de "movimiento" de un programa de computadora se basa en detalles ocultos que hacen que un objeto cambie de ubicación en la pantalla. A medida que progresan, los estudiantes deben incorporar funciones predefinidas en sus diseños, entendiendo que no necesitan conocer los detalles de implementación subyacentes de las abstracciones que utilizan.

Eventualmente, los estudiantes deben comprender las ventajas de las funcionalidades existentes (abstracciones) y sentirse cómodos al usarlas, incluidos los recursos tecnológicos creados por otras personas, tales como las bibliotecas y las interfaces de programación de aplicaciones (API). Los estudiantes deben poder evaluar las abstracciones existentes para determinar cuáles deben incorporarse en los diseños y cómo deben incorporarse. Por ejemplo, los estudiantes podrían crear aplicaciones potentes incorporando servicios existentes, como bases de datos en línea que devuelven coordenadas de geolocalización de nombres de calles o información sobre nutrición de alimentos.

3. **Crear módulos y desarrollar puntos de interacción que puedan aplicarse a múltiples situaciones y reducir la complejidad.**
   Después de que los alumnos hayan tenido alguna experiencia en la identificación de patrones (P4.1), problemas de descomposición (P3.2), uso de abstracciones (P4.2) y aprovechamiento de los recursos existentes (P4.2), deberían comenzar a desarrollar sus propias abstracciones. A medida que progresan, los estudiantes deben aprovechar las oportunidades para desarrollar módulos generalizables. Por ejemplo, los estudiantes podrían escribir programas más eficientes diseñando procedimientos que se usan varias veces en el programa. Estos procedimientos se pueden generalizar mediante la definición de parámetros que crean diferentes salidas para una amplia gama de entradas. Más adelante, los estudiantes deben poder diseñar sistemas de módulos interactivos, cada uno con un rol bien definido, que coordinen para lograr un objetivo común. Dentro de un contexto de programación orientado a objetos, el diseño del módulo puede incluir la definición de las interacciones entre objetos. En esta etapa, estos módulos, que combinan datos y procedimientos, pueden diseñarse y documentarse para su reutilización en otros programas. Además, los estudiantes pueden diseñar puntos de interacción, como una interfaz de usuario simple, ya sea de texto o gráfica, que reduce la complejidad de una solución y oculta los detalles de implementación de nivel inferior.

4. **Modelar fenómenos y procesos y simular sistemas para comprender y evaluar resultados potenciales.**
   Los estudiantes en todos los niveles de grado deben poder representar patrones, procesos o fenómenos. Con orientación, los jóvenes estudiantes pueden dibujar para describir un patrón simple, como el amanecer y el atardecer, o mostrar las etapas de un proceso, tal como cepillar los dientes. También pueden crear una animación para modelar un fenómeno, como la evaporación. A medida que progresan, los estudiantes deben entender que las computadoras pueden modelar fenómenos del mundo real, y deben usar las simulaciones de computadoras existentes para aprender sobre sistemas del mundo real. Por ejemplo, pueden usar un modelo preprogramado para explorar cómo los parámetros afectan un sistema, tal como la rapidez con que se propaga una enfermedad. Los estudiantes mayores deben modelar los fenómenos como sistemas, con reglas que gobiernan las interacciones dentro del sistema. Los estudiantes deben analizar y evaluar estos modelos contra las observaciones del mundo real. Por ejemplo, los estudiantes pueden crear un modelo simple de ecosistema productor/consumidor utilizando una herramienta de programación. Con el tiempo, podrían progresar hacia la creación de interacciones más complejas y realistas entre las especies, tales como la depredación, la competencia o la simbiosis, y evaluar el modelo basándose en datos recopilados de la naturaleza.

Práctica 5. Crear artefactos computacionales:

Resumen: El proceso de desarrollo de artefactos computacionales abarca tanto la expresión creativa como la exploración de ideas para crear prototipos y resolver problemas computacionales. Los estudiantes crean artefactos que son personalmente relevantes o beneficiosos para su comunidad y más allá. Los artefactos computacionales se pueden crear combinando y modificando artefactos existentes o desarrollando nuevos artefactos. Ejemplos de artefactos computacionales incluyen programas, simulaciones, visualizaciones, animaciones digitales, sistemas robóticos y aplicaciones.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Planificar el desarrollo de un artefacto computacional** *utilizando un proceso iterativo que incluye la reflexión y la modificación del plan, teniendo en cuenta las características clave, las limitaciones de tiempo y recursos, y las expectativas del usuario.*

   En cualquier nivel de grado, los estudiantes deben participar en la planificación de proyectos y la creación de documentos de una tormenta de ideas. Los alumnos más pequeños pueden hacerlo con la ayuda de los profesores. Con los andamios, los estudiantes deben obtener mayor independencia y sofisticación en la planificación, diseño y evaluación de artefactos. A medida que el aprendizaje progresa, los estudiantes deben planificar sistemáticamente el desarrollo de un programa o artefacto y aplicar intencionalmente técnicas computacionales, como la descomposición y la abstracción, junto con el conocimiento sobre los enfoques existentes para el diseño de artefactos. Los estudiantes deben ser capaces de reflexionar y, si es necesario, modificar el plan para adaptarse a las metas finales.

2. **Crear un artefacto computacional para propósitos prácticos, expresiones personales o para abordar un problema social.**
   Los estudiantes en todos los niveles de grado deben desarrollar artefactos en respuesta a una tarea o un problema computacional. En los primeros niveles de grado, los estudiantes deben poder elegir entre un conjunto de comandos dados para crear historias animadas simples o resolver problemas preexistentes. Los estudiantes más pequeños deben centrarse en los artefactos de importancia personal. A medida que progresan, las expresiones de los estudiantes deben ser más complejas y de importancia cada vez más amplia. Eventualmente, los estudiantes deben involucrarse en el uso independiente y sistemático de los procesos de diseño para crear artefactos que resuelvan problemas con importancia social buscando opiniones de audiencias amplias.

3. **Modificar un artefacto existente para mejorarlo o personalizarlo.**
   En todos los niveles de grado, los estudiantes deben poder examinar los artefactos existentes para entender lo que hacen. A medida que progresan, los estudiantes deben intentar usar las soluciones existentes para lograr una meta. Por ejemplo, los estudiantes podrían conectar un

sensor de luz programable a un artefacto físico que hayan creado para que responda a la luz. Más adelante, deben modificar o remezclar partes de programas existentes para desarrollar algo nuevo o para agregar características y complejidad más avanzadas. Por ejemplo, los estudiantes podrían modificar el código escrito previamente de un juego para un solo jugador para crear un juego para dos jugadores con reglas ligeramente diferentes.

Práctica 6. Probar y refinar los artefactos computacionales:

Resumen: La prueba y el refinamiento es el proceso deliberado e iterativo de mejorar un artefacto computacional. Este proceso incluye la depuración (identificación y corrección de errores) y la comparación de los resultados reales con los resultados previstos. Los estudiantes también responden a las necesidades y expectativas cambiantes de los usuarios finales y mejoran el rendimiento, la confiabilidad, la facilidad de uso y la accesibilidad de los artefactos.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Probar sistemáticamente** *los artefactos computacionales considerando todos los escenarios y utilizando casos de prueba.*
   En cualquier nivel de grado, los estudiantes deben poder comparar los resultados con los resultados esperados. Los estudiantes jóvenes deben verificar si se han cumplido con los criterios y las restricciones dados. A medida que los estudiantes progresan, deben probar los artefactos computacionales considerando errores potenciales, tal como lo que ocurrirá si un usuario ingresa una entrada no válida. Eventualmente, las pruebas deben convertirse en un proceso deliberado que sea más iterativo, sistemático y proactivo. Los alumnos mayores deben poder anticipar errores y usar ese conocimiento para impulsar el desarrollo. Por ejemplo, los estudiantes pueden probar su programa con entradas asociadas con todos los escenarios posibles.

2. **Identificar y corregir** *errores mediante un proceso sistemático.*
   En cualquier nivel de grado, los estudiantes deben poder identificar y corregir errores en los programas (depuración) y utilizar estrategias para resolver problemas con sistemas de computación (solución de problemas). Los jóvenes estudiantes podrían usar prueba y error para corregir errores simples. Por ejemplo, un estudiante puede intentar reordenar la secuencia de comandos en un programa. En un contexto de hardware, los estudiantes podrían intentar arreglar un dispositivo restableciéndolo o verificando si está conectado a una red. A medida que los estudiantes progresan, deberían ser más adeptos a los programas de depuración y comenzar a considerar los errores lógicos: casos en los que funciona un programa, pero no como se desea. De esta manera, los estudiantes examinarán y corregirán su propio pensamiento. Por ejemplo, pueden pasar por su programa, línea por línea, para identificar un bucle que no termina como se espera. Eventualmente, los estudiantes mayores deben avanzar hacia el uso de estrategias más complejas para identificar y corregir errores, como imprimir el valor de una variable de contador mientras se está ejecutando un ciclo para determinar cuántas veces se ejecuta el ciclo.

3. **Evaluar y refinar** *un artefacto computacional varias veces para mejorar su rendimiento, confiabilidad, facilidad de uso y accesibilidad.*
   Una vez que los alumnos hayan adquirido experiencia en pruebas (P6.2), depuración y revisión (P6.1), deben comenzar a evaluar y refinar sus artefactos computacionales. A medida que los estudiantes progresan, el proceso de evaluación y refinamiento debe enfocarse en mejorar el rendimiento y la confiabilidad. Por ejemplo, los estudiantes podrían observar un robot en una variedad de condiciones de iluminación para determinar que un sensor de luz debería ser menos sensible. Más adelante, la evaluación y el refinamiento deben convertirse en un proceso iterativo que también incluya hacer que los artefactos sean más utilizables y accesibles (P1.2). Por ejemplo, los estudiantes pueden incorporar comentarios de una variedad de usuarios finales para ayudar a guiar el tamaño y la ubicación de los menús y botones en una interfaz de usuario.

Práctica 7. Comunicar acerca de la computación:

Resumen: La comunicación implica la expresión personal y el intercambio de ideas con otros. En ciencias de la computación, los estudiantes se comunican con audiencias diversas sobre el uso y los efectos de la computación y lo apropiado de las opciones computacionales. Los estudiantes escriben comentarios claros, documentan su trabajo y comunican sus ideas a través de múltiples formas de medios. La comunicación clara incluye el uso de un lenguaje preciso y considera cuidadosamente las posibles audiencias.

Al final de grado 12, los estudiantes deben ser capaces de:

1. **Seleccionar, organizar e interpretar grandes conjuntos de datos** *de múltiples fuentes para respaldar un reclamo.*
   Seleccionar, organizar e interpretar grandes conjuntos de datos de múltiples fuentes para respaldar una reclamación. En cualquier nivel de grado, los estudiantes deben poder referirse a los datos cuando comunican una idea. Al principio, los estudiantes deben, con orientación, presentar datos básicos mediante el uso de representaciones visuales, tales como guiones gráficos, diagramas de flujo y gráficos. A medida que los estudiantes progresan, deben trabajar con conjuntos de datos más grandes y organizar los datos en esos conjuntos más grandes para facilitar la interpretación y la comunicación a otros, tal como a través de la creación de representaciones de datos básicos. Eventualmente, los estudiantes deben poder seleccionar datos relevantes de conjuntos de datos grandes o complejos para respaldar una reclamación o comunicar la información de una manera más sofisticada.

2. **Describir, justificar y documentar** *los procesos y soluciones computacionales utilizando la terminología apropiada y de acuerdo con la audiencia y el propósito previstos.*
   En cualquier nivel de grado, los estudiantes deben poder hablar sobre las elecciones que hacen al diseñar un artefacto computacional. Al principio, deben usar un lenguaje que articule lo que están haciendo e identifique los dispositivos y conceptos que están usando con la terminología

correcta (por ejemplo, programa, entrada y depurar). Los estudiantes más pequeños deben identificar las metas y los resultados esperados de sus soluciones. En el camino, los estudiantes deben proporcionar documentación para los usuarios finales que explique sus artefactos y cómo funcionan, y deben dar y recibir comentarios. Por ejemplo, los estudiantes pueden proporcionar un resumen del proyecto y solicitar información de los usuarios. A medida que los estudiantes progresan, deben incorporar comentarios claros en su producto y documentar su proceso utilizando texto, gráficos, presentaciones y demostraciones.

3. **Articular las ideas de manera responsable** *mediante la observación de los derechos de propiedad intelectual y la atribución adecuada.*
   Todos los estudiantes deben poder explicar los conceptos de propiedad y compartir. Al principio, los estudiantes deben aplicar estos conceptos a las ideas y creaciones computacionales. Deben identificar instancias de remezclas, cuando las ideas se toman prestadas y se repiten, y deben otorgar la debida atribución. También deben reconocer las aportaciones de los colaboradores. Eventualmente, los estudiantes deben considerar licencias comunes que imponen limitaciones o restricciones al uso de artefactos computacionales. Por ejemplo, una imagen descargada puede tener restricciones que prohíban la modificación de una imagen o su uso con fines comerciales.

Conceptos

Incluyendo Conceptos Transversales

6

Conceptos *Incluyendo Conceptos Transversales*

Prefacio

Los conceptos core del marco de las ciencias de la computación K–12 representan áreas de contenido principales en el campo de la ciencia de la computación. Los conceptos core están delineados por múltiples subconceptos que representan ideas específicas dentro de cada concepto. Las progresiones de aprendizaje para cada subconcepto proporcionan un hilo que conecta el aprendizaje de los estudiantes desde el jardín de infantes hasta el grado 12.

Conceptos core del marco:

1. Sistemas de computación
2. Redes e internet
3. Datos y análisis
4. Algoritmos y programación
5. Impactos de la computación

Los conceptos transversales son temas que ilustran conexiones entre diferentes enunciados de conceptos. Se integran en declaraciones de concepto, en lugar de existir como una dimensión independiente del marco. Los conceptos transversales que se representan en cada declaración de concepto se anotan en el material descriptivo de la declaración.

Conceptos transversales del marco:

- Abstracción
- Relaciones de sistemas
- Interacción humano-computadora
- Privacidad y seguridad
- Comunicación y coordinación

Abstracción: Una abstracción es el resultado de reducir un proceso o conjunto de información a un conjunto de características importantes para uso computacional. Las abstracciones establecen interacciones a un nivel de complejidad reducida al administrar los detalles más complejos por debajo del nivel de interacción. Se puede crear una abstracción para generalizar un rango de situaciones seleccionando propiedades comunes menos detalles de implementación específicos.

Relaciones de sistemas: Las partes de un sistema son interdependientes y están organizadas para un propósito común. Una perspectiva de sistemas brinda la oportunidad de descomponer problemas complejos en partes que son más fáciles de entender, desarrollar, corregir y mantener. Los principios generales de los sistemas incluyen retroalimentación, control, eficiencia, modularidad, síntesis, emergencia y jerarquía.

Interacción humano-computadora: los seres humanos interactúan directamente con computadoras, tales como computadoras portátiles y teléfonos inteligentes, pero también con otros dispositivos, tales como automóviles y electrodomésticos, que tienen computadoras integradas. El desarrollo de interfaces de usuario efectivas y accesibles implica la integración del conocimiento técnico y las ciencias sociales y abarca las perspectivas tanto del diseñador como del usuario.

Privacidad y seguridad: La privacidad es la capacidad de ocultar información y expresarla de forma selectiva. Incluye controles para el almacenamiento, acceso, uso, intercambio y alteración de la información. La seguridad se refiere a las medidas de seguridad que rodean a los sistemas de información e incluye protección contra robo o daño al hardware, software y la información en los sistemas. Seguridad apoya la privacidad.

Comunicación y coordinación: Los procesos en computación se caracterizan por el intercambio confiable de información entre agentes autónomos (comunicación) que cooperan hacia resultados comunes que ningún agente podría producir solo (coordinación). Comunicación y coordinación son procesos distintos pero no independientes. Lo que hace especial a las ciencias de la computación es la escala en la que funciona la comunicación y la coordinación.

Resúmenes de conceptos core y subconceptos:

Sistemas de computación:

Resumen: Las personas interactúan con una amplia variedad de dispositivos computacionales que recopilan, almacenan, analizan y actúan sobre la información de manera que puede afectar las capacidades humanas tanto positiva como negativamente. Los componentes físicos (hardware) y las instrucciones (software) que conforman un sistema de computación se comunican y procesan la información en forma digital. Comprender el hardware y el software es útil para solucionar problemas de un sistema de computación que no funciona como se esperaba.

Dispositivos

Resumen: Muchos objetos cotidianos contienen componentes computacionales que detectan y actúan en el mundo. En los primeros grados, los estudiantes aprenden características y aplicaciones de dispositivos computacionales comunes. A medida que avanzan, los estudiantes aprenden sobre los sistemas conectados y cómo la interacción entre humanos y dispositivos influye en las decisiones de diseño.

Hardware y Software

Resumen: Sistemas de computación utiliza hardware y software para comunicar y procesar información en forma digital. En los primeros grados, los estudiantes aprenden cómo los sistemas utilizan tanto el hardware como el software para representar y procesar la información. A medida que avanzan, los estudiantes obtienen una comprensión más profunda de la interacción entre hardware y software en múltiples niveles dentro de Sistemas de Computación.

Resolución de problemas

Resumen: Cuando los sistemas de computación no funcionan según lo previsto, las estrategias de solución de problemas ayudan a las personas a resolver el problema. En los primeros grados, los estudiantes aprenden que identificar el problema es el primer paso para solucionarlo. A medida que avanzan, los estudiantes aprenden procesos sistemáticos de resolución de problemas y cómo desarrollar sus propias estrategias de solución de problemas basadas en una comprensión más profunda de cómo funcionan los sistemas de computación.

Redes e internet

Resumen: Los dispositivos computacionales típicamente no funcionan de forma aislada. Las redes conectan dispositivos computacionales para compartir información y recursos, y son una parte cada vez más integral de la computación. Las redes y los sistemas de comunicación proporcionan una mayor conectividad en el mundo de la computación al proporcionar una comunicación rápida y segura y facilitar la innovación.

Red de comunicación y organización

Resumen: Dispositivos computacionales se comunican entre sí a través de redes para compartir información. En los primeros grados, los estudiantes aprenden que las computadoras los conectan con otras personas, lugares y cosas de todo el mundo. A medida que avanzan, los estudiantes obtienen una comprensión más profunda de cómo se envía y recibe la información a través de diferentes tipos de redes.

Seguridad cibernética

Resumen: La transmisión segura de información a través de las redes requiere una protección adecuada. En los primeros grados, los estudiantes aprenden cómo proteger su información personal. A medida que avanzan, los estudiantes aprenden formas cada vez más complejas de proteger la información que se envía a través de las redes.

Datos y análisis:

Resumen: Sistemas de computación existen para procesar datos. La cantidad de datos digitales generados en el mundo se está expandiendo rápidamente, por lo que la necesidad de procesar datos de manera efectiva es cada vez más importante. Los datos se recopilan y almacenan para que se puedan analizar para comprender mejor el mundo y realizar predicciones más precisas.

Colección

Resumen: Los datos se recopilan con herramientas y procesos computacionales y no computacionales. En los primeros grados, los estudiantes aprenden cómo se recopilan y utilizan los datos sobre ellos mismos y su mundo. A medida que avanzan, los estudiantes aprenden los efectos de recopilar datos con herramientas computacionales y automatizadas.

Almacenamiento

Resumen: Las funciones básicas de las computadoras son almacenar, representar y recuperar datos. En los primeros grados, los estudiantes aprenden cómo se almacenan los datos en las computadoras. A medida que avanzan, los estudiantes aprenden cómo evaluar diferentes métodos de almacenamiento, incluidas las ventajas y desventajas asociadas con esos métodos.

Visualización y Transformación

Resumen: Los datos se transforman a lo largo del proceso de colección, representación digital y análisis. En los primeros grados, los estudiantes aprenden cómo pueden usarse las transformaciones para simplificar los datos. A medida que avanzan, los estudiantes aprenden sobre operaciones más complejas para descubrir patrones y tendencias y comunicarse con otros.

Inferencia y Modelos

La ciencia de datos es un ejemplo donde las ciencias de la computación sirven a muchos campos. Las ciencias de la computación y la ciencia utilizan datos para hacer inferencias, teorías o predicciones basadas en los datos recopilados de los usuarios o simulaciones. En los primeros grados, los estudiantes aprenden sobre el uso de datos para hacer predicciones simples. A medida que avanzan, los estudiantes aprenden cómo se pueden usar los modelos y las simulaciones para examinar teorías y comprender los sistemas, y cómo las predicciones e inferencias se ven afectadas por conjuntos de datos más complejos y grandes.

Algoritmos y Programación

Resumen: Un algoritmo es una secuencia de pasos diseñados para realizar una tarea específica. Los algoritmos se traducen en programas, o código, para proporcionar instrucciones para dispositivos computacionales. Algoritmos y programación controlan todos los sistemas de computación, permitiendo que las personas se comuniquen con el mundo de nuevas maneras y resuelvan problemas importantes. El proceso de desarrollo para crear programas significativos y eficientes implica elegir qué información usar y cómo procesarla y almacenarla, dividir los problemas grandes en otros más pequeños, combinar las soluciones existentes y analizar diferentes soluciones.

Algoritmos

Resumen: Están diseñados para ser llevados a cabo tanto por humanos como por computadoras. En los primeros grados, los estudiantes aprenden sobre algoritmos apropiados para su edad del mundo real. A medida que avanzan, los estudiantes aprenden sobre el desarrollo, la combinación y la descomposición de los algoritmos, así como la evaluación de los algoritmos en competencia.

Variables

Resumen: Los programas de computación almacenan y manipulan datos utilizando variables. En los primeros grados, los estudiantes aprenden que diferentes tipos de datos, como palabras, números o imágenes, se pueden utilizar de diferentes maneras. A medida que avanzan, los estudiantes aprenden sobre variables y formas de organizar grandes colecciones de datos en estructuras de datos de complejidad creciente.

Control

Resumen: Las estructuras de control especifican el orden en que se ejecutan las instrucciones dentro de un algoritmo o programa. En los primeros grados, los estudiantes aprenden sobre la ejecución secuencial y las estructuras de control simples. A medida que avanzan, los estudiantes amplían su comprensión a combinaciones de estructuras que apoyan la ejecución compleja.

Modularidad

Resumen: La modularidad implica dividir las tareas en tareas más simples y combinar tareas simples para crear algo más complejo. En los primeros grados, los estudiantes aprenden que los algoritmos y los programas pueden diseñarse dividiendo las tareas en partes más pequeñas y recombinando las soluciones existentes. A medida que avanzan, los estudiantes aprenden a reconocer patrones para hacer uso de soluciones generales y reutilizables para escenarios comunes y a describir con claridad tareas de maneras que son ampliamente utilizables.

Desarrollo de Programas

Resumen: Los programas se desarrollan a través de un proceso de diseño que frecuentemente se repite hasta que el programador está satisfecho con la solución. En los primeros grados, los estudiantes aprenden cómo y por qué las personas desarrollan programas. A medida que avanzan, los estudiantes aprenden sobre las ventajas y desventajas en el diseño del programa asociado con decisiones complejas que involucran restricciones, eficiencia, ética y pruebas de los usuarios.

Impactos de la computación

Resumen: La computación afecta muchos aspectos del mundo de manera positiva y negativa a nivel local, nacional y global. Los individuos y las comunidades influyen en la computación a través de sus comportamientos e interacciones sociales y culturales, y a su vez, la computación influye en las nuevas prácticas culturales. Una persona informada y responsable debe comprender las implicaciones sociales del mundo digital, incluida la equidad y el acceso a los sistemas de computación.

Cultura

Resumen: La computación influye en la cultura, incluidos los sistemas de creencias, el lenguaje, las relaciones, la tecnología y las instituciones, y la cultura determina cómo las personas se involucran y acceden a las ciencias de la computación. En los primeros grados, los estudiantes aprenden cómo la computación puede ser útil y perjudicial. A medida que avanzan, los estudiantes aprenden sobre las compensaciones asociadas con la computación y sus futuros impactos potenciales de la computación en las sociedades globales.

Interacciones Sociales

Resumen: La computación puede apoyar nuevas formas de conectar personas, comunicar información y expresar ideas. En los primeros grados, los estudiantes aprenden que las ciencias de la computación pueden conectar a las personas y apoyar la comunicación interpersonal. A medida que avanzan, los estudiantes aprenden cómo la naturaleza social de las ciencias de la computación afecta a las instituciones y las carreras en diversos sectores.

Seguridad, Ley y Ética

Resumen: Las consideraciones legales y éticas del uso de dispositivos computacionales influyen en los comportamientos que pueden afectar el bienestar y la seguridad de las personas. En los primeros grados, los estudiantes aprenden los fundamentos de la ciudadanía digital y el uso apropiado de los medios digitales. A medida que avanzan, los estudiantes aprenden sobre los problemas éticos y legales que dan forma a las prácticas de las ciencias de la computación.

Conceptos

Al final de grado 2:

Sistemas de Computación	
DISPOSITIVOS	Las personas usan dispositivos para realizar una variedad de tareas con precisión y rapidez. Dispositivos de computación interpretan y siguen las instrucciones que se dan literalmente. *Los dispositivos de computación se pueden utilizar para hacer varias cosas, como reproducir música, crear documentos y enviar imágenes. Los dispositivos de computación también son muy precisos. Por ejemplo, las computadoras pueden realizar múltiples cálculos complejos mucho más rápido y con mayor precisión que las personas. Si bien las personas pueden desviarse de las instrucciones que se les dan, las computadoras seguirán las instrucciones exactamente cómo se las dan, incluso si no logran el resultado deseado.* ***Conceptos transversales**: Interacción humano-computadora.* ***Conexiones dentro del marco**: K–2.Algoritmos y programación.Control; K–2. Algoritmos y programación.Modularidad; 3–5. Algoritmos y programación.Control.*
HARDWARE Y SOFTWARE	Un sistema de la computación está compuesto por hardware y software. El hardware consta de componentes físicos, mientras que el software proporciona instrucciones para el sistema. Estas instrucciones están representadas en una forma que una computadora puede entender. *Los ejemplos de hardware incluyen pantallas para mostrar información y botones, teclas o diales para ingresar información. Las aplicaciones de software son programas con fines específicos, como un navegador web o un juego. Una persona puede usar un mouse (hardware) para hacer clic en un botón que se muestra en un navegador web (software) para navegar a una nueva página web. Los sistemas de computación convierten las instrucciones, como "imprimir", "guardar" o "recortar", en un lenguaje especial que la computadora pueda entender. En este nivel, entender que la información de la computadora está codificada es apropiado, pero la comprensión explícita de "bits" se reserva para los niveles de grado posteriores.* ***Conceptos transversales**: Comunicación y coordinación.* ***Conexiones dentro del marco**: K–2.Algoritmos y Programación. Algoritmos; K–2.Algoritmos y Programación.Control.*
RESOLUCIÓN DE PROBLEMAS	Es posible que los sistemas de computación no funcionen como se esperaba debido a problemas de hardware o software. Describir claramente un problema es el primer paso para encontrar una solución. *Los problemas con los sistemas de computación tienen diferentes causas, como configuraciones de hardware, errores de programación o conexiones defectuosas a otros dispositivos. Las formas apropiadas para el desarrollo de resolver problemas incluyen la depuración de programas simples y la búsqueda de ayuda describiendo claramente un problema (por ejemplo, "La computadora no se enciende", "El puntero en la pantalla no se mueve" o "Perdí la página "). No se espera saber cómo diagnosticar o solucionar un problema con un sistema de computación.* ***Conceptos transversales**: Relaciones de sistemas.* ***Conexiones dentro del marco**: 3–5. Algoritmos y programación. Desarrollo de programas.*

Al final de grado 2: continúa de la página anterior

Redes e Internet	
RED DE COMUNICACIÓN Y ORGANIZACIÓN	Las redes de computadoras se pueden usar para conectar personas con otras personas, lugares, información e ideas. Internet permite a las personas conectarse con otras personas en todo el mundo a través de diferentes puntos de conexión. *Los dispositivos inalámbricos pequeños, como los teléfonos celulares, se comunican entre sí a través de una serie de puntos de conexión intermedios, como las torres celulares. Esta coordinación entre muchos dispositivos computacionales permite a una persona llamar por voz a un amigo o chatear por video con un miembro de la familia. No se esperan detalles sobre los puntos de conexión en este nivel.* ***Conceptos transversales****: Comunicación y coordinación; Interacción humano-computadora.* ***Conexiones dentro del marco****: K–2. Impactos de la computación. Interacciones sociales; K–2. Datos y análisis.Colección; 3–5. Impactos de la computación.Interacciones sociales.*
SEGURIDAD CIBERNÉTICA	Conectar dispositivos a una red o al internet proporciona un gran beneficio, se debe tener cuidado al utilizar medidas de autenticación, como contraseñas seguras, para proteger dispositivos e información de accesos no autorizados. *La autenticación es la capacidad de verificar la identidad de una persona o entidad. Los nombres de usuario y las contraseñas, como los de dispositivos computacionales o redes Wi-Fi, proporcionan una forma de autenticar la identidad de un usuario. Debido a que las computadoras hacen que sea fácil adivinar contraseñas débiles, las contraseñas seguras tienen características que las hacen más difíciles de descifrar.* ***Conceptos transversales****: Privacidad y seguridad; Comunicación y coordinación.* ***Conexiones dentro del marco****: K–2. Impactos de la computación.Seguridad, Ley y Ética.*

Al final de grado 2: continúa de la página anterior

Datos y Análisis	
COLECCIÓN	Todos los días los dispositivos digitales recopilan y muestran datos a lo largo del tiempo. La recopilación y el uso de datos sobre individuos y el mundo que los rodea es una parte rutinaria de la vida e influye en cómo vive la gente. *Muchos objetos cotidianos, como teléfonos celulares, juguetes digitales y automóviles, pueden contener herramientas (tales como sensores) y computadoras para recopilar y mostrar datos de su entorno.* ***Conceptos transversales**: Interacción humano-computadora.* ***Conexiones dentro del marco**: K–2. Redes e internet.Red de comunicación y organización.*
ALMACENAMIENTO	Las computadoras almacenan datos que se pueden recuperar más tarde. Las copias idénticas de los datos se pueden hacer y almacenar en múltiples ubicaciones por una variedad de razones, como para proteger contra la pérdida. *Por ejemplo, las imágenes se pueden almacenar en un teléfono celular y verlas más tarde, o el progreso en un juego se puede guardar y continuar más tarde. La ventaja de registrar datos digitalmente, como en imágenes o en una hoja de cálculo, en lugar de en un espacio físico, como una pizarra, es que los datos antiguos (estados de datos recopilados a lo largo del tiempo) se pueden recuperar, copiar y almacenar fácilmente en múltiples lugares. Es por esto por lo que la información personal puesta en línea puede persistir durante mucho tiempo. No se espera la comprensión del almacenamiento local frente al en línea en este nivel.* ***Conceptos transversales**: Relaciones de sistemas; Privacidad y seguridad.* ***Conexiones dentro del marco**: K–2. Impactos de la computación. Interacciones sociales; K–2. Algoritmos y programación.Variables.*
VISUALIZACIÓN Y TRANSFORMACIÓN	Los datos se pueden mostrar para la comunicación de muchas maneras. Las personas usan las computadoras para transformar los datos en nuevos formularios, como gráficos y tablas. *Los ejemplos de pantallas incluyen gráficos de imágenes, gráficos de barras o histogramas. Una tabla de datos que registra una cuenta de los colores favoritos de los estudiantes se puede mostrar como una tabla en una computadora.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: K–2. Impactos de la Computación. Interacciones Sociales.*
INFERENCIA Y MODELOS	Los datos pueden usarse para hacer inferencias o predicciones sobre el mundo. Las inferencias, declaraciones sobre algo que no se pueden observar fácilmente, se basan frecuentemente en datos observados. Las predicciones, declaraciones sobre eventos futuros, se basan en patrones en los datos y se pueden realizar mediante mirar a las visualizaciones de datos, tales como tablas y gráficos. *Las observaciones de la ropa de las personas (chaquetas y abrigos) se pueden usar para hacer una inferencia sobre el clima (afuera hace frío). Los patrones en datos pasados se pueden identificar y extrapolar para hacer predicciones. Por ejemplo, la selección del menú de almuerzo de una persona se puede predecir mediante el uso de datos de selecciones de almuerzos anteriores.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: K–2. Impactos de la computación.Cultura.*

Al final de grado 2: continúa de la página anterior

Algoritmos y Programación	
ALGORITMOS	La gente sigue y crea procesos como parte de la vida cotidiana. Muchos de estos procesos se pueden expresar como algoritmos que las computadoras pueden seguir. *Las rutinas, como la reunión matutina, el tiempo de limpieza y el despido de las clases al fin del día, son ejemplos de algoritmos que son comunes en muchas clases primarias tempranas. Otros ejemplos de algoritmos incluyen preparar comidas simples, navegar por un aula y rutinas diarias como cepillarse los dientes. Así como las personas usan algoritmos para completar rutinas diarias, pueden programar computadoras para usar algoritmos para completar diferentes tareas. Los algoritmos se implementan comúnmente utilizando un lenguaje preciso que las computadoras pueden interpretar.* ***Concepto transversal:*** *Abstracción.* ***Conexiones dentro del marco:*** *K–2. Sistemas de computación.Hardware y Software.*
VARIABLES	La información en el mundo real puede ser representada en programas de computadora. Los programas almacenan y manipulan datos, tales como números, palabras, colores e imágenes. El tipo de datos determina las acciones y atributos asociados con él. *Diferentes acciones están disponibles para diferentes tipos de información. Por ejemplo, los pitufos (imágenes de caracteres) se pueden mover y girar, los números se pueden sumar o restar, y las imágenes se pueden recolorear o recortar.* ***Concepto transversal:*** *Abstracción* ***Conexiones dentro del marco:*** *K–2. Datos y análisis.Almacenamiento.*
CONTROL	Las computadoras siguen secuencias precisas de instrucciones que automatizan las tareas. La ejecución del programa también puede ser no secuencial al repetir patrones de instrucciones y usar eventos para iniciar instrucciones. *Las computadoras siguen las instrucciones literalmente. Los ejemplos de secuencias de instrucciones incluyen pasos para dibujar una forma o mover un personaje a través de la pantalla. Un evento, como presionar un botón, puede desencadenar una acción. Se pueden usar bucles simples para repetir las instrucciones. En este nivel, no se espera distinguir diferentes tipos de bucles.* ***Concepto transversal:*** *Abstracción.* ***Conexiones dentro del marco:*** *K–2. Datos y análisis.Almacenamiento.*

Al final de grado 2: continúa de la página anterior

MODULARIDAD	Las tareas complejas se pueden dividir en instrucciones más simples, algunas de las cuales se pueden desglosar aún más. Del mismo modo, las instrucciones se pueden combinar para realizar tareas complejas. *La descomposición es el acto de dividir las tareas en tareas más simples. Un ejemplo de descomposición es prepararse para una fiesta: implica invitar a los invitados, preparar la comida y poner la mesa. Estas tareas se pueden desglosar aún más. Por ejemplo, poner la mesa implica poner un mantel, doblar servilletas y poner utensilios y platos sobre la mesa. Otro ejemplo es desglosar los pasos para dibujar un polígono. La composición, por otro lado, es la combinación de tareas más pequeñas en tareas más complejas. Para construir una ciudad, la gente construye varias casas, una escuela, una tienda, etc. Para crear un proyecto de arte en grupo, la gente puede pintar o dibujar su animal marino favorito, luego combinarlos para crear un ecosistema.* ***Concepto transversal**: Relaciones de sistemas.* ***Conexiones dentro del marco**: K–2. Sistemas de computación. Dispositivos.*
DESARROLLO DE PROGRAMAS	Las personas desarrollan programas en colaboración y con un propósito, tales como expresar ideas o abordar problemas. *Las personas trabajan juntas para planificar, crear y probar programas dentro de un contexto que es relevante para el programador y los usuarios. La programación se utiliza como una herramienta para crear productos que reflejen una amplia gama de intereses, tales como videojuegos, proyectos de arte interactivo e historias digitales.* ***Concepto transversal**: Interacción humano-computadora.* ***Conexiones dentro del marco**: 3–5. Impactos de la computación.Cultura.*

Al final de grado 2: continúa de la página anterior

Impactos de la Computación	
CULTURA	La tecnología de la computación ha cambiado de manera positiva y negativa la forma en que las personas viven y trabajan. Dispositivos computacionales se pueden usar para entretenimiento y como herramientas de productividad, y pueden afectar las relaciones y los estilos de vida. *Los dispositivos computacionales, como los rastreadores de ejercicios, pueden motivar un estilo de vida más activo al monitorear la actividad física. Por otro lado, el consumo pasivo de medios computacionales puede llevar a un estilo de vida más sedentario. En el pasado, la forma más popular de comunicación era enviar correo a través del servicio postal. Ahora, más personas envían correos electrónicos o mensajes de texto.* ***Concepto transversal**: Interacción humano-computadora.* ***Conexiones dentro del marco**: K–2. Datos y análisis.Inferencia y modelos.*
INTERACCIONES SOCIALES	La computación ha cambiado positiva y negativamente la forma en que las personas se comunican. Las personas pueden tener acceso a la información y entre sí de forma instantánea, en cualquier lugar y en cualquier momento, pero corren el riesgo de acoso cibernético y privacidad reducida. *La comunicación en línea facilita las interacciones positivas, como compartir ideas con muchas personas, pero la naturaleza pública y anónima de la comunicación en línea también permite un comportamiento intimidante e inapropiado en forma de acoso cibernético. La privacidad debe ser considerada al publicar información en línea; dicha información puede persistir por un largo tiempo y ser visitada por otros, incluso por espectadores no intencionados.* ***Conceptos transversales**: Interacción humano-computadora; Privacidad y seguridad.* ***Conexiones dentro del marco**: K–2. Datos y análisis.Almacenamiento; K–2.Datos y análisis.Visualización y transformación.*
SEGURIDAD, LEY Y ÉTICA	Las personas usan la tecnología computacional de manera que puedan ayudarse o lastimarse a sí mismos o a otros. Las conductas dañinas, tales como compartir información privada e interactuar con extraños, deben reconocerse y evitarse. *El uso de computadoras conlleva un nivel de responsabilidad, tales como no compartir información de inicio de sesión, mantener las contraseñas privadas y cerrar la sesión cuando haya terminado. Las reglas que guían las interacciones en el mundo, como "peligro extraño", se aplican también a los entornos en línea.* ***Concepto transversal**: Privacidad y seguridad.* ***Conexiones dentro del marco**: K–2. Redes e internet.Seguridad cibernética.*

Al final de grado 5

Sistemas de Computación	
DISPOSITIVOS	Dispositivos computacionales pueden estar conectados a otros dispositivos o componentes para ampliar sus capacidades, tales como la detección y el envío de información. Las conexiones pueden tomar muchas formas, como físicas o inalámbricas. Juntos, los dispositivos y los componentes forman un sistema de partes interdependientes que interactúan para lograr un propósito común. *Dispositivos computacionales frecuentemente dependen de otros dispositivos o componentes. Por ejemplo, un robot depende de un sensor de luz conectado físicamente para detectar cambios en el brillo, mientras que el sensor de luz depende del robot para obtener energía. Un teléfono inteligente puede usar auriculares conectados de forma inalámbrica para enviar información de audio, y los auriculares son inútiles sin una fuente de música.* ***Conceptos transversales**: Comunicación y coordinación; Relaciones de sistemas.* ***Conexiones dentro del marco**: 3–5. Redes e internet.Red de comunicación y organización.*
HARDWARE Y SOFTWARE	El hardware y el software funcionan juntos como un sistema para realizar tareas, tales como enviar, recibir, procesar y almacenar unidades de información como bits. Los bits sirven como la unidad básica de datos en sistemas de computación y pueden representar una variedad de información. *Por ejemplo, una aplicación de filtro de fotos (software) funciona con una cámara (hardware) para producir una variedad de efectos que cambian el aspecto de una imagen. Esta imagen se transmite y almacena como bits o dígitos binarios, que generalmente se representan como 0's y 1's. Toda la información, incluidas las instrucciones, se codifica como bits. El conocimiento del funcionamiento interno del hardware y software, los sistemas numéricos como el binario o el hexadecimal, y cómo se representan los bits en los medios físicos no son prioridades en este nivel.* ***Conceptos transversales**: Comunicación y coordinación; Abstracción.* ***Conexiones dentro del marco**: 3–5. Datos y análisis.Almacenamiento.*
RESOLUCIÓN DE PROBLEMAS	Los sistemas de computación comparten similitudes, como el uso de energía, datos y memoria. Las estrategias comunes de solución de problemas, tales como verificar que la energía está disponible, verificar que las conexiones físicas e inalámbricas funcionan y borrar la memoria de trabajo al reiniciar programas o dispositivos, son efectivas para muchos sistemas. *Aunque los sistemas de computación pueden variar, se pueden usar estrategias comunes de solución de problemas en ellos, como verificar las conexiones y la alimentación o cambiar una parte de trabajo en lugar de una parte potencialmente defectuosa. El reinicio de una máquina suele ser efectivo porque restablece la computadora. Debido a que los dispositivos computacionales se componen de un sistema interconectado de hardware y software, las estrategias de solución de problemas pueden necesitar abordar ambos.* ***Conceptos transversales**: Relaciones de sistemas; Abstracción.* ***Conexiones dentro del marco**: 3–5. Redes e internet.Red de comunicación y organización.*

Al final de grado 5: continúa de la página anterior

Redes e Internet	
RED DE COMUNICACIÓN Y ORGANIZACIÓN	La información necesita una ruta física o inalámbrica para viajar para ser enviada y recibida, y algunas rutas son mejores que otras. La información se divide en partes más pequeñas, llamadas paquetes, que se envían de forma independiente y se vuelven a ensamblar en el destino. Los enrutadores y los conmutadores se utilizan para enviar correctamente paquetes a través de rutas a sus destinos. *Existen rutas físicas para comunicar información, tales como cables Ethernet y rutas Wi-Fi inalámbricas. Frecuentemente, la información viaja en una combinación de rutas físicas e inalámbricas; por ejemplo, las rutas inalámbricas se originan desde un punto de conexión física. La elección del dispositivo y el tipo de conexión afectará los viajes de información de la ruta y el ancho de banda potencial (la capacidad de transmitir datos o bits en un período de tiempo determinado). Los paquetes y la conmutación de paquetes (el método utilizado para enviar paquetes) son la base para una mejor comprensión de los conceptos de Internet. En este nivel, la prioridad es comprender el flujo de información, en lugar de detalles de cómo funcionan los enrutadores y conmutadores y cómo comparar rutas.* ***Concepto transversal**: Comunicación y coordinación.* ***Conexiones dentro del marco**: 3–5. Sistemas de computación. Dispositivos; 3–5. Dispositivos.Resolución de problemas.*
SEGURIDAD CIBERNÉTICA	La información puede ser protegida usando varias medidas de seguridad. Estas medidas pueden ser físicas y/o digitales. *Una copia de seguridad de datos fuera de línea es útil en caso de una violación de seguridad en línea. Una variedad de aplicaciones de software puede monitorear y tratar virus y malware y alertar a los usuarios de su presencia. Las medidas de seguridad se pueden aplicar a una red o dispositivos individuales en una red. La confidencialidad se refiere a la protección de la información contra la divulgación a personas, sistemas o entidades no autorizados.* ***Concepto transversal**: Privacidad y seguridad.* ***Conexiones dentro del marco**: 3–5. Impactos de la computación.Seguridad, Ley y Ética.*

Al final de grado 5: continúa de la página anterior

Data and Analysis	
COLECCIÓN	Las personas seleccionan herramientas digitales para la colección de datos en función de lo que se está observando y cómo se utilizarán los datos. Por ejemplo, un termómetro digital se usa para medir la temperatura y un sensor GPS se usa para rastrear ubicaciones. *Existe una amplia gama de herramientas de colección de datos digitales; sin embargo, solo algunos son apropiados para ciertos tipos de datos. Las herramientas se eligen en función del tipo de medición que utilizan, así como del tipo de datos que las personas desean observar. Los científicos de datos usan el término observación para describir la colección de datos, ya sea que un humano participe o no en la colección.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco:** 3–5. Algoritmos y programación.Variables; 3–5. Algoritmos y programación.Algoritmos.*
ALMACENAMIENTO	Las diferentes herramientas de software utilizadas para acceder a los datos pueden almacenar los datos de manera distinta. El tipo de datos que se almacenan y el nivel de detalle representado por esos datos afectan los requisitos de almacenamiento. La música, las imágenes, el video y el texto requieren diferentes cantidades de almacenamiento. *El video a menudo requerirá más almacenamiento que la música o las imágenes solo porque el video combina ambos. Por ejemplo, dos imágenes del mismo objeto pueden requerir diferentes cantidades de almacenamiento según su resolución. Las diferentes herramientas de software utilizadas para acceder y almacenar datos pueden agregar datos adicionales sobre los datos (metadatos), lo que da como resultado diferentes requisitos de almacenamiento. Un archivo de imagen es una representación diseñada de una imagen del mundo real y puede abrirse con un editor de imágenes o con un editor de texto, pero el editor de texto no sabe cómo traducir los datos a la imagen. No se espera que se entiendan las representaciones binarias o de 8 bits frente a las de 16 bits en este nivel.* ***Concepto Transversal**: Relaciones de sistemas.* ***Conexiones Dentro Del Marco**: 3–5. Resolución de problemas.Hardware y Software; 3–5. Algoritmos y Programación.Variables.*
VISUALIZACIÓN Y TRANSFORMACIÓN	Las personas seleccionan aspectos y subconjuntos de datos para ser transformados, organizados, agrupados y categorizados para proporcionar vistas distintas y comunicar las perspectivas obtenidas de los datos. *Los datos frecuentemente se clasifican o agrupan para proporcionar claridad adicional. Los puntos de datos se pueden agrupar por una serie de puntos en común sin una etiqueta de categoría. Por ejemplo, una serie de días puede agruparse según la temperatura, la presión del aire y la humedad, y luego se puede clasificar como clima templado, suave o extremo. Los mismos datos podrían manipularse de diferentes maneras para enfatizar aspectos particulares o partes del conjunto de datos. Por ejemplo, cuando se trabaja con un conjunto de datos de canciones populares, los datos pueden mostrarse por género o artista. Las visualizaciones simples de datos incluyen gráficos y tablas, infografías y relaciones que representan características estadísticas de los datos.* ***Conceptos transversales**: Abstracción; Interacción humano-computadora.* ***Conexiones dentro del marco**: 6–8. Impactos de la computación.Interacciones sociales.*

Al final de grado 5: continúa de la página anterior

INFERENCIA Y MODELOS	La precisión de las inferencias y predicciones se relaciona con la forma realista en que se representan los datos. Muchos factores influyen en la precisión de las inferencias y predicciones, como la cantidad y la relevancia de los datos recopilados. *Las personas usan los datos para resaltar o proponer relaciones de causa y efecto y predecir resultados. Basar inferencias o predicciones en datos no garantiza su exactitud; los datos deben ser relevantes y en cantidad suficiente. Un ejemplo de irrelevancia es el uso de datos de color de ojos al inferir la edad de alguien. Un ejemplo de cantidad insuficiente es predecir el resultado de una elección encuestando solo a unas pocas personas.* *Concepto transversal: Relaciones de sistemas.*

Al final de grado 5: continúa de la página anterior

Algoritmos y Programación	
ALGORITMOS	Distintos algoritmos pueden lograr el mismo resultado. Algunos algoritmos son más apropiados para un contexto específico que otros. *Se pueden usar diferentes algoritmos para atar los zapatos o decidir qué trayecto a tomar en el camino a casa desde la escuela. Si bien los resultados finales pueden ser similares, pueden no ser los mismos: en el ejemplo de ir a casa, algunos caminos pueden ser más rápidos, más lentos o directos, dependiendo de diversos factores, tales como el tiempo disponible o la presencia de obstáculos (para ejemplo, un perro que ladra). Algoritmos se puede expresar en lenguajes no computacionales, incluidos lenguaje natural, diagramas de flujo y pseudocódigo.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: 3–5. Datos y análisis.Colección.*
VARIABLES	Los lenguajes de programación proporcionan variables, que se utilizan para almacenar y modificar datos. El tipo de datos determina los valores y las operaciones que se pueden realizar en esos datos. *Las variables son el vehículo a través del cual los programas de computadora almacenan diferentes tipos de datos. En este nivel, la comprensión de cómo usar las variables es suficiente, sin una comprensión más completa de los aspectos técnicos de las variables (como los identificadores y las ubicaciones de la memoria). Los tipos de datos varían según el lenguaje de programación, pero muchos tienen tipos para números y texto. Ejemplos de operaciones asociadas con esos tipos son multiplicar números y combinar texto. Algunos lenguajes visuales basados en bloques no tienen tipos declarados explícitamente, pero aún tienen ciertas operaciones que se aplican solo a tipos particulares de datos en un programa.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: 3–5. Datos y análisis.Almacenamiento.*
CONTROL	Las estructuras de control, incluidos los bucles, los controladores de eventos y los condicionales, se utilizan para especificar el flujo de ejecución. Los condicionales ejecutan de forma selectiva u omiten instrucciones en diferentes condiciones. *Se utilizan diferentes tipos de bucles para repetir las instrucciones de varias maneras según la situación. Los ejemplos de eventos incluyen clics del mouse, escritura en el teclado y colisiones entre objetos. Los controladores de eventos son conjuntos de comandos que están vinculados a eventos específicos. Los condicionales representan decisiones y se componen de una condición booleana que especifica acciones basadas en si la condición se evalúa como verdadera o falsa. La lógica y los operadores booleanos (por ejemplo, AND, OR, NOT) se pueden usar para especificar los grupos de instrucciones apropiados para ejecutar en diversas condiciones.* ***Conceptos transversales**: Abstracción; Comunicación y coordinación.* ***Conexiones dentro del marco**: K–2. Sistemas de computación. Dispositivos.*

Al final de grado 5: continúa de la página anterior

MODULARIDAD	Los programas se pueden dividir en partes más pequeñas para facilitar su diseño, implementación y revisión. Los programas también se pueden crear incorporando porciones más pequeñas de los programas que ya se han creado. *La descomposición facilita aspectos del desarrollo de programas, como las pruebas, al permitir que las personas se centren en una pieza a la vez. La descomposición también permite que diferentes personas trabajen en diferentes partes al mismo tiempo. Un ejemplo de descomposición en este nivel es crear una animación al separar una historia en diferentes escenas. Para cada escena, se debe seleccionar un fondo, colocar los caracteres y programar las acciones. Las instrucciones requeridas para programar cada escena pueden ser similares a las instrucciones en otros programas.* ***Conceptos transversales**: Relaciones de sistemas; Abstracción*
DESARROLLO DE PROGRAMAS	Las personas desarrollan programas utilizando un proceso iterativo que involucra diseño, implementación y revisión. El diseño a menudo implica reutilizar el código existente o remezclar otros programas dentro de una comunidad. Las personas continuamente revisan si los programas funcionan como se espera y arreglan o depuran las partes que no lo hacen. La repetición de estos pasos permite a las personas perfeccionar y mejorar los programas. *El diseño, la implementación y la revisión se pueden dividir en etapas adicionales y pueden tener diferentes etiquetas. La etapa de diseño se produce antes de escribir código. Esta es una etapa de planificación en la que los programadores recopilan información sobre el problema y diseñan una solución. Durante la etapa de implementación, el diseño planificado se expresa en un lenguaje de programación (código) que puede ejecutarse en un dispositivo computacional. Durante la etapa de revisión, se verifica que el diseño y la implementación cumplan con los requisitos, la corrección y la facilidad de uso del programa. Esta revisión podría llevar a cambios en la implementación y posiblemente en el diseño, lo que demuestra la naturaleza iterativa del proceso. Una comunidad es creada por personas que comparten y proporcionan comentarios sobre las creaciones de los demás.* ***Conceptos transversales:** Interacción humano-computadora; Relaciones de sistemas.* ***Conexiones dentro del marco**: K–2. Sistemas de computación. Resolución de problemas.*

Al final de grado 5: continúa de la página anterior

Impacts of Computing	
CULTURA	El desarrollo y la modificación de la tecnología computacional están impulsados por las necesidades y deseos de las personas y pueden afectar a los grupos de maneras diferentes. Las tecnologías computacionales influyen, y son influenciadas por las prácticas culturales. *Se crea nueva tecnología computacionales y las tecnologías existentes se modifican para aumentar sus beneficios (por ejemplo, las recomendaciones de búsqueda en Internet), disminuir sus riesgos (por ejemplo, automóviles autónomos) y satisfacer las demandas de la sociedad (por ejemplo, aplicaciones para teléfonos inteligentes). El mayor acceso a Internet y la velocidad han permitido a las personas compartir información cultural, pero también han afectado la práctica de las costumbres culturales tradicionales.* ***Conceptos transversales**: Interacción humano-computadora; Relaciones de sistemas.* ***Conexiones dentro del marco**: K–2.Algoritmos y programación.Desarrollo de programas; 6–8. Resolución de problemas. Dispositivos; 6–8. Algoritmos y programación.Desarrollo de programas*
INTERACCIONES SOCIALES	La tecnología computacional permite la colaboración local y global. Al facilitar la comunicación y la innovación, la computación influye en muchas instituciones sociales como la familia, la educación, la religión y la economía. *Las personas pueden trabajar en diferentes lugares y en distintos momentos para colaborar y compartir ideas cuando utilizan tecnologías que llegan a todo el mundo. Estas interacciones sociales afectan la forma en que los grupos locales y globales interactúan entre sí y, alternativamente, estas interacciones pueden cambiar la naturaleza de los grupos. Por ejemplo, una clase puede discutir ideas en la misma escuela o en otra nación a través de seminarios web interactivos.* ***Conceptos transversales**: Relaciones de sistemas; Interacción humano-computadora.* ***Conexiones dentro del marco**: K–2. Redes e Internet.Red de Comunicación y Organización.*
SEGURIDAD, LEY Y ÉTICA	Las complicaciones éticas surgen de las oportunidades que brinda la computación. La facilidad de enviar y recibir copias de medios en internet, tales como videos, fotos y música, crea la oportunidad para el uso no autorizado, como la piratería en línea, y el desprecio de derechos de autor, tal como la falta de atribución. *La piratería en línea, la copia ilegal de materiales se ve facilitada por la capacidad de hacer copias de medios digitales de idéntica calidad con poco esfuerzo. Otros temas relacionados con los derechos de autor son el plagio, el uso justo y las fuentes en línea de manera adecuada. El conocimiento de leyes específicas de derechos de autor no es una expectativa a este nivel.* ***Conceptos transversales**: Relaciones de sistemas; Privacidad y seguridad.* ***Conexiones dentro del marco**: 3–5. Redes e Internet.Seguridad cibernética.*

Al final de grado 8

Sistemas de Computación	
DISPOSITIVOS	La interacción entre humanos y dispositivos computacionales presenta ventajas, desventajas y consecuencias no intencionadas. El estudio de interacción humano-computadora puede mejorar el diseño de dispositivos y ampliar las capacidades de los humanos. *La accesibilidad es una consideración importante en el diseño de cualquier sistema computacional. Por ejemplo, los dispositivos de asistencia brindan capacidades tales como escanear información escrita y convertirla a voz. El uso de dispositivos computacionales también tiene consecuencias potenciales, como en las áreas de privacidad y seguridad. Por ejemplo, los teléfonos inteligentes habilitados para GPS pueden proporcionar direcciones a un destino y, sin embargo, permitir que una persona sea rastreada con propósitos maliciosos. Además, la atención requerida para seguir las indicaciones del GPS puede provocar accidentes debido a la conducción distraída.* ***Conceptos transversales**: Interacción humano-computadora; Privacidad y seguridad.* ***Conexiones dentro del marco**: 3–5. Impactos de la computación.Cultura.*
HARDWARE Y SOFTWARE	El hardware y el software determinan la capacidad de un sistema computacional para almacenar y procesar información. El diseño o la selección de un sistema de computación involucra consideraciones múltiples y posibles compromisos, tales como funcionalidad, costo, tamaño, velocidad, accesibilidad y estética. *La capacidad de un sistema computacional está determinada por la velocidad del procesador, la capacidad de almacenamiento y la velocidad de transmisión de datos, así como otros factores. La selección de un sistema de computación sobre otro implica el equilibrio de una serie de compensaciones. Por ejemplo, la selección de una computadora más rápida con más memoria implica los compromisos de velocidad y costo. Elegir un sistema operativo en lugar de otro implica el intercambio de capacidades y compatibilidad, por ejemplo, que aplicaciones pueden instalarse o que dispositivos pueden conectarse. Diseñar un robot requiere elegir hardware y software, y puede implicar un compromiso entre el potencial de personalización y la facilidad de uso. El uso de un dispositivo que se conecta de forma inalámbrica a través de una conexión Bluetooth en comparación con un dispositivo que se conecta físicamente a través de una conexión USB implica un compromiso entre la movilidad y la necesidad de una fuente de alimentación adicional para el dispositivo inalámbrico.* ***Conceptos transversales**: Relaciones de sistemas; Comunicación y coordinación.* ***Conexiones dentro del marco**: 6–8. Datos y análisis.Colección.*

Al final de grado 8: continúa de la página anterior

RESOLUCIÓN DE PROBLEMAS

La solución integral de problemas requiere el conocimiento de cómo funcionan e interactúan los dispositivos computacionales y los componentes. Un proceso sistemático identificará la fuente de un problema, ya sea dentro de un dispositivo o en un sistema más grande de dispositivos conectados.

Al igual que los pilotos usan listas de verificación para solucionar problemas con sistemas de aeronaves, las personas pueden usar un proceso similar y estructurado para solucionar problemas con sistemas de computación y asegurar que no se pasen por alto las soluciones potenciales. Debido a que un dispositivo computacional puede interactuar con dispositivos interconectados dentro de un sistema, es posible que los problemas no se deban al dispositivo computacional específico en sí, sino a los dispositivos conectados a él. Algunos ejemplos de componentes del sistema que pueden necesitar solución de problemas son las conexiones físicas e inalámbricas, los equipos periféricos y el hardware de red. Las estrategias para solucionar problemas de un sistema computacional y depurar un programa incluyen algunos pasos de resolución de problemas que son similares.

***Conceptos transversales**: Relaciones de sistemas; Abstracción*

***Conexiones dentro del marco**: 6–8. Algoritmos y programación. Algoritmos.*

Al final de grado 8: continúa de la página anterior

Redes e Internet	
RED DE COMUNICACIÓN Y ORGANIZACIÓN	Las computadoras envían y reciben información basada en un conjunto de reglas llamadas protocolos. Los protocolos definen cómo se estructuran y envían los mensajes entre las computadoras. Las consideraciones de seguridad, velocidad y confiabilidad se utilizan para determinar la mejor ruta para enviar y recibir datos. Los protocolos permiten que los dispositivos con diferentes hardware y software se comuniquen, de la manera en que las personas con diferentes idiomas nativos pueden usar un idioma común para negocios. *Los protocolos describen los comandos y las respuestas establecidas entre las computadoras en una red, tales como solicitar datos o enviar una imagen. Hay muchos ejemplos de protocolos que incluyen TCP/IP (Protocolo de control de transmisión/Protocolo de Internet) y HTTP (Protocolo de transferencia de hipertexto), que sirven como base para el formateo y la transmisión de mensajes y datos, incluidas las páginas en la World Wide Web. Los enrutadores también implementan protocolos para registrar las rutas más rápidas y confiables al enviar paquetes pequeños como pruebas. La prioridad en este nivel de grado es entender el propósito de los protocolos, mientras que no se espera conocer los detalles de cómo funcionan los protocolos específicos.* ***Conceptos transversales**: Comunicación y coordinación; Abstracción; Privacidad y seguridad.* ***Conexiones dentro del marco**: 6–8. Datos y análisis.Almacenamiento.*
SEGURIDAD CIBERNÉTICA	La información enviada y recibida a través de las redes puede protegerse del acceso y modificación no autorizados de diversas maneras, como la encripción para mantener su confidencialidad y el acceso restringido para mantener su integridad. Las medidas de seguridad para salvaguardar la información en línea abordan de manera proactiva la amenaza de violaciones a los datos personales y privados. *La integridad de la información implica asegurar su consistencia, precisión y confiabilidad. Por ejemplo, HTTPS (Protocolo de transferencia de hipertexto seguro) es un ejemplo de una medida de seguridad para proteger las transmisiones de datos. Proporciona una conexión de navegador más segura que HTTP (Protocolo de transferencia de hipertexto) porque encripta los datos que se envían entre sitios web. En este nivel, es importante comprender la diferencia entre HTTP y HTTPS, pero no cómo funcionan las tecnologías.* ***Concepto transversal**: Privacidad y seguridad.* ***Conexiones dentro del marco**: 6–8. Impactos de la computación.Seguridad, Ley y Ética.*

Al final de grado 8: continúa de la página anterior

Datos y Análisis	
COLECCIÓN	La gente diseña algoritmos y herramientas para automatizar la colección de datos por computadora. Cuando la colección de datos está automatizada, los datos se muestren y se convierten en un formulario que una computadora puede procesar. Por ejemplo, los datos de un sensor analógico se deben convertir a una forma digital. El método utilizado para automatizar la recopilación de datos está influenciado por la disponibilidad de herramientas y el uso previsto de los datos. *Los datos se pueden recopilar de dispositivos o sistemas individuales. El método de recopilación de datos (por ejemplo, encuestas frente a datos de sensores) puede afectar la precisión y la exactitud de los datos. Algunos tipos de datos son más difíciles de recopilar que otros. Por ejemplo, las emociones deben evaluarse subjetivamente en forma individual y, por lo tanto, son difíciles de medir en una población. El acceso a las herramientas puede estar limitado por factores que incluyen el costo, la capacitación y la disponibilidad.* ***Concepto Transversal**: Interacción humano-computadora.* ***Conexiones Dentro Del Marco**: 6–8. Sistemas de computación.Hardware y Software.*
ALMACENAMIENTO	Las aplicaciones almacenan los datos como una representación. Las representaciones se producen en múltiples niveles, desde la disposición de la información en formatos organizados (tales como tablas en software) hasta el almacenamiento físico de bits. Las herramientas de software utilizadas para acceder a la información traducen la representación de bits de bajo nivel en una forma comprensible para las personas. *Las computadoras pueden representar una variedad de datos utilizando valores discretos en muchos niveles diferentes, como caracteres, números y bits. El texto se representa mediante estándares de codificación de caracteres como UNICODE, que representan el texto como números. Todos los números y otros tipos de datos se codifican y almacenan como bits en un medio físico. Los formatos de datos con pérdida y sin pérdida se utilizan para almacenar diferentes niveles de detalle, pero cuando se usan datos digitales para representar mediciones analógicas, como la temperatura o el sonido, la información se pierde. Las representaciones, o formatos de archivo, pueden contener metadatos que no siempre son visibles para el usuario promedio. Existen implicaciones de privacidad cuando los archivos contienen metadatos, tal como la ubicación donde se tomó la fotografía.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: 6–8. Algoritmos y programación.Variables; 6–8. Redes e internet.Red de comunicación y organización.*

Al final de grado 8: continúa de la página anterior

VISUALIZACIÓN Y TRANSFORMACIÓN	Los datos se pueden transformar para eliminar errores, resaltar o exponer relaciones y/o facilitar el procesamiento de las computadoras. *La limpieza de datos es una transformación importante para reducir el ruido y los errores. Un ejemplo de ruido serían los primeros segundos de una muestra en la que un sensor de audio recopila un sonido extraño creado por el usuario que coloca el sensor. Los errores en los datos de la encuesta se limpian para eliminar respuestas erróneas o inapropiadas. Un ejemplo de una transformación que resalta una relación es representar dos grupos (tales como hombres y mujeres) como porcentajes de un todo en lugar de como conteos individuales. Los biólogos computacionales utilizan algoritmos de compresión para hacer que los conjuntos de datos extremadamente grandes de información genética sean más manejables y el análisis más eficiente.* ***Concepto transversal**: Abstracción* ***Conexiones dentro del marco**: 6–8. Algoritmos y programación. Algoritmos.*
INFERENCIA Y MODELOS	Los modelos de computadora se pueden usar para simular eventos, examinar teorías e inferencias, o hacer predicciones con pocos o millones de puntos de datos. Los modelos computacionales son abstracciones que representan fenómenos y utilizan datos y algoritmos para enfatizar características y relaciones clave dentro un sistema. A medida que se recopilan más datos automáticamente, los modelos se pueden refinar. *Los conjuntos de datos muy grandes requieren un modelo para el análisis; son demasiado grandes para ser analizados examinando todos los registros. Mientras que los usuarios individuales están en línea, los sitios web de compras y las publicidades en línea utilizan los datos personales que generan, en comparación con millones de otros usuarios, para predecir lo que les gustaría y hacer recomendaciones. Un sitio web de transmisión de video puede recomendar videos basados en modelos generados por otros usuarios y en base a sus hábitos y preferencias personales. Los datos que se recopilan sobre un individuo y las inferencias potenciales hechas a partir de esos datos pueden tener implicaciones para la privacidad.* ***Conceptos transversales:** Privacidad y seguridad; Abstracción.* ***Conexiones dentro del marco**: 6–8. Algoritmos y programación. Algoritmos; 6–8. Impactos de la computación.Cultura.*

Al final de grado 8: continúa de la página anterior

Algoritmos y Programación	
ALGORITMOS	Los algoritmos afectan la forma en que las personas interactúan con las computadoras y la forma en que las computadoras responden. Las personas diseñan algoritmos que son generalizables a muchas situaciones. Los algoritmos que son legibles son más fáciles de seguir, probar y depurar. *Algoritmos controla qué recomendaciones puede obtener un usuario en un sitio web de transmisión de música, cómo responde un juego a las pulsaciones de los dedos en una pantalla táctil y cómo se envía la información a través de Internet. Un algoritmo que se puede generalizar en muchas situaciones puede producir diferentes salidas, basadas en una amplia gama de entradas. Por ejemplo, un algoritmo para un termostato inteligente puede controlar la temperatura según la hora del día, cuántas personas están en casa y el consumo de electricidad actual. La prueba de un algoritmo requiere el uso de entradas que reflejen todas las condiciones posibles para evaluar su precisión y robustez.* ***Conceptos transversales**: Interacción humano-computadora; Abstracción.* ***Conexiones dentro del marco**: 6–8. Datos y análisis. Inferencia y modelos; 6–8. Sistemas de computación. Resolución de problemas; 6–8. Datos y análisis. Visualización y transformación.*
VARIABLES	Los programadores crean variables para almacenar valores de datos de los tipos seleccionados. Se asigna un identificador significativo a cada variable para acceder y realizar operaciones en el valor por nombre. Las variables permiten la flexibilidad de representar diferentes situaciones, procesar distintos conjuntos de datos y producir resultados variables. *En este nivel, los estudiantes profundizan su comprensión de las variables, incluyendo cuándo y cómo declarar y nombrar nuevas variables. Una variable es como un contenedor con un nombre, en el que los contenidos pueden cambiar, pero el nombre (identificador) no lo hace. El identificador facilita el seguimiento de los datos que se almacenan, especialmente si los datos cambian. Las convenciones de nomenclatura para los identificadores, y la elección cuidadosa de los identificadores, mejoran la legibilidad del programa.* *El término variable se usa de manera distinta en la programación que en la forma en que se usa comúnmente en matemáticas: una variable de programa se refiere a una ubicación en la que se almacena un valor, y el nombre utilizado para acceder al valor se denomina identificador. A una variable de programa se le asigna un valor, y ese valor puede cambiar a lo largo de la ejecución del programa. Los matemáticos normalmente no hacen una distinción entre una variable y el nombre de la variable. Una variable matemática a menudo representa un conjunto de valores para los cuales la declaración que contiene la variable es verdadera.* ***Concepto transversal**: Abstracción.* ***Conexiones dentro del marco**: 6–8. Datos y análisis.Almacenamiento.*

Al final de grado 8: continúa de la página anterior

CONTROL	Los programadores seleccionan y combinan estructuras de control, tales como bucles, controladores de eventos y condicionales, para crear un comportamiento de programa más complejo. *Las declaraciones condicionales pueden tener distintos niveles de complejidad, incluidos condicionales compuestos y anidados. Los condicionales compuestos combinan dos o más condiciones en una relación lógica, y los condicionales anidados entre sí permiten que el resultado de un condicional conduzca a que se evalúe a otro. Un ejemplo de una estructura condicional anidada es decidir qué hacer en función del clima exterior. Si hace sol afuera, decidiré si quiero andar en bicicleta o salir a correr, pero si no hace sol afuera, decidiré si leer un libro o ver la televisión. Se pueden combinar diferentes tipos de estructuras de control entre sí, tales como bucles y condicionales. Diferentes tipos de lenguajes de programación implementan estructuras de control de diferentes maneras. Por ejemplo, los lenguajes de programación funcional implementan la repetición utilizando llamadas a funciones recursivas en lugar de bucles. En este nivel, entender la implementación en múltiples idiomas no es esencial.* ***Concepto transversal**: Abstracción.*
MODULARIDAD	Los programas utilizan procedimientos para organizar el código, ocultar los detalles de la implementación y hacer que el código sea más fácil de reutilizar. Los procedimientos pueden ser reutilizados en nuevos programas. La definición de parámetros para procedimientos puede generalizar el comportamiento y aumentar la reutilización. *Un procedimiento es un módulo (un grupo de instrucciones dentro de un programa) que realiza una tarea particular. En este marco, el procedimiento se utiliza como un término general que puede referirse a un procedimiento real o un método, función o concepto similar en otros lenguajes de programación. Se invocan procedimientos para repetir grupos de instrucciones. Por ejemplo, un procedimiento, como uno para dibujar un círculo, implica muchas instrucciones, pero todas pueden invocarse con una instrucción, como "drawCircle". Los procedimientos que se definen con parámetros se pueden generalizar en muchas situaciones y producirán diferentes salidas basadas en una amplia gama de entradas (argumentos).* ***Conceptos transversales**: Abstracción; Relaciones de sistemas.*
DESARROLLO DE PROGRAMA	Las personas diseñan soluciones significativas para otros definiendo los criterios y limitaciones de un problema, considerando cuidadosamente las diversas necesidades y deseos de la comunidad y evaluando si se cumplieron los criterios y las restricciones. *Los equipos de desarrollo que emplean un diseño centrado en el usuario crean soluciones que pueden tener un gran impacto social, como una aplicación que permite a las personas con dificultades del habla traducir la pronunciación difícil de entender a un lenguaje comprensible. Los casos de uso y los casos de prueba se crean y analizan para satisfacer mejor las necesidades de los usuarios y evaluar si se cumplen los criterios y las restricciones. Un ejemplo de una restricción de diseño es que las aplicaciones móviles deben optimizarse para pantallas pequeñas y una vida útil limitada de la batería.* ***Conceptos transversales**: Interacción humano-computadora; Abstracción.* ***Conexiones dentro del marco**: 3–5. Impactos de la computación.Cultura.*

Al final de grado 8: continúa de la página anterior

Impactos de la Computación	
CULTURAE	Los avances en la tecnología computacional cambian las actividades cotidianas de las personas. La sociedad se enfrenta a compensaciones debido a la creciente globalización y automatización que trae la computación. *Los efectos de la globalización, como el intercambio de información y las prácticas y la homogeneidad culturales resultante, son cada vez más posibles gracias a la computación. La globalización, junto con la automatización de la producción de bienes, permite el acceso a mano de obra que es menos costosa y crea empleos que pueden moverse fácilmente a través de las fronteras nacionales. La piratería en línea ha aumentado debido al acceso a la información que atraviesa las fronteras nacionales y diversos sistemas legales.* ***Conceptos transversales**: Interacción humano-computadora; Relaciones de sistemas.* ***Conexiones dentro del marco**: 6–8. Datos y análisis.Inferencia y modelos.*
INTERACCIONES SOCIALES	Las personas pueden organizarse e involucrarse en torno a temas de interés a través de diversas plataformas de comunicación habilitadas por la computación, tales como las redes sociales y los medios de comunicación. Estas interacciones permiten examinar los problemas utilizando múltiples puntos de vista de una audiencia diversa. *Las redes sociales pueden desempeñar un papel importante en los movimientos sociales y políticos al permitir que las personas compartan ideas y opiniones sobre problemas comunes al tiempo que se relacionan con quienes tienen opiniones diferentes. La computación proporciona un entorno rico para el discurso, pero puede resultar en que las personas consideren puntos de vista muy limitados de una audiencia limitada.* ***Conceptos transversales**: Relaciones de sistemas; Interacción humano-computadora.* ***Conexiones dentro del marco**: 3–5. Datos y análisis.Visualización y transformación; 9–12.Datos y análisis.Visualización y transformación.*
SEGURIDAD, LEY Y ÉTICA	Hay concesiones entre permitir que la información sea pública y mantener la información privada y segura. Se puede engañar a las personas para que revelen información personal cuando haya más información pública disponible sobre ellas en línea. *La ingeniería social se basa en engañar a las personas para que rompan los procedimientos de seguridad y se puede frustrar al ser conscientes de varios tipos de ataques, como correos electrónicos con información falsa y phishing. Los ataques de seguridad frecuentemente comienzan con información personal que está disponible públicamente en línea. Todos los usuarios deben conocer la información personal, especialmente la información financiera, que se almacena en los sitios web que utilizan. La protección de la información personal en línea requiere medidas de autenticación que a menudo pueden dificultar el acceso de la información a los usuarios autorizados.* ***Conceptos transversales**: Privacidad y seguridad; Comunicación y coordinación.* ***Conexiones dentro del marco**: 6–8. Redes e internet.Seguridad cibernética.*

Al final de grado 12

Sistemas de Computación	
DISPOSITIVOS	Los dispositivos computacionales frecuentemente se integran con otros sistemas, incluidos los sistemas biológicos, mecánicos y sociales. Estos dispositivos pueden compartir datos entre sí. La usabilidad, la confiabilidad, la seguridad y la accesibilidad de estos dispositivos, y los sistemas con los que están integrados, son consideraciones importantes en su diseño a medida que evolucionan. *Un dispositivo médico puede integrarse dentro de una persona para monitorear y regular su salud, un audífono (un tipo de dispositivo de asistencia) puede filtrar ciertas frecuencias y magnificar otras, un dispositivo de monitoreo instalado en un vehículo motorizado puede rastrear los patrones y hábitos de conducción de una persona, y un dispositivo de reconocimiento facial puede integrarse en un sistema de seguridad para identificar a una persona. Los dispositivos integrados en los objetos, vehículos y edificios cotidianos les permiten recopilar e intercambiar datos, creando una red (por ejemplo, Internet de las cosas). La creación de sistemas integrados no es una expectativa a este nivel.* ***Conceptos transversales**: Relaciones de sistema; Interacción humano-computadora; Privacidad y seguridad.* ***Conexiones dentro del marco**: 9–12. Redes e Internet. Red de comunicación y organización; 9–12. Datos y análisis.Colección; 9–12. Impactos de computación. Cultura.*
HARDWARE Y SOFTWARE	Existen niveles de interacción entre el hardware, el software y el usuario de un sistema computacional. Los niveles más comunes de software con los que un usuario interactúa incluyen el software y las aplicaciones del sistema. El software del sistema controla el flujo de información entre los componentes de hardware utilizados para la entrada, salida, almacenamiento y procesamiento. *En su nivel más básico, una computadora está compuesta de hardware físico e impulsos eléctricos. Varias capas de software están construidas sobre el hardware e interactúan con las capas encima y debajo de ellas para reducir la complejidad. El software del sistema administra los recursos de un dispositivo computacional para que el software pueda interactuar con el hardware. Por ejemplo, el software de edición de texto interactúa con el sistema operativo para recibir información del teclado, convertir la entrada en bits para su almacenamiento e interpretar los bits como texto legible para mostrar en el monitor. El software del sistema se utiliza en muchos tipos diferentes de dispositivos, tales como televisores inteligentes, dispositivos de asistencia, componentes virtuales, componentes en la nube y aviones no tripulados. En este nivel no se espera el conocimiento de términos avanzados y específicos para arquitectura de computadora, tales como BIOS, kernel o bus.* ***Conceptos transversales**: Abstracción; Comunicación y coordinación; Relaciones de sistema.* ***Conexiones dentro del marco**: 9–12. Redes e Internet. Red de comunicación y organización; 9–12.Algoritmos y programación.Variables; 9–12. Algoritmos y programación. Modulardad.*

Al final de grado 12: continúa de la página anterior

RESOLUCIÓN DE PROBLEMAS	La resolución de problemas complejos implica el uso de múltiples fuentes cuando se investiga, evalúa e implementa posibles soluciones. La resolución de problemas también se basa en la experiencia, tales como cuando las personas reconocen que un problema es similar a uno que han visto antes o adaptan soluciones que han funcionado en el pasado. *La información de resolución de problemas puede provenir de fuentes externas, como manuales de usuario, foros técnicos en línea o sitios web de fabricantes. Distinguir entre fuentes confiables y no confiables es importante. Los ejemplos de estrategias de resolución de problemas complejas incluyen resolver problemas de conectividad, ajustar configuraciones y configuraciones del sistema, asegurar la compatibilidad de hardware y software, y transferir datos de un dispositivo a otro.* ***Conceptos transversales**: Abstracción; Relaciones de sistemas.* ***Conexiones dentro del marco:** 9–12. Algoritmos y programación.Desarrollo de programas.*

Al final de grado 12: continúa de la página anterior

Redes e Internet	
RED DE COMUNICACIÓN Y ORGANIZACIÓN	La topología de la red está determinada, en parte, por la cantidad de dispositivos que se pueden admitir. A cada dispositivo se le asigna una dirección que lo identifica de forma única en la red. *La escalabilidad y confiabilidad de Internet están habilitadas por la jerarquía y la redundancia en las redes. La coordinación a gran escala se produce entre muchas máquinas diferentes en múltiples rutas cada vez que se abre una página web o se visualiza una imagen en línea. A los dispositivos en Internet se les asigna una dirección de Protocolo de Internet (IP) para permitirles comunicarse. Las decisiones de diseño que dirigieron la coordinación entre los sistemas que componen Internet también permitieron la escalabilidad y confiabilidad. La escalabilidad es la capacidad de una red para manejar una cantidad creciente de trabajo o su potencial para ampliarse para adaptarse a ese crecimiento.* ***Conceptos transversales**: Comunicación y coordinación; Abstracción; Relaciones de sistema.* ***Conexiones dentro del marco**: 9–12.Sistemas de computación.Dispositivos; 9–12. Sistemas de computación.Hardware y Software; 9–12. Impactos de computación.Interacciones sociales.*
SEGURIDAD CIBERNÉTICA	La seguridad de la red depende de una combinación de hardware, software y prácticas que controlan el acceso a los datos y sistemas. Las necesidades de los usuarios y la sensibilidad de los datos determinan el nivel de seguridad implementado. *Las medidas de seguridad pueden incluir tokens de seguridad física, autenticación de dos factores y verificación biométrica, pero todas las medidas de seguridad implican compromisos entre la accesibilidad y la seguridad del sistema. Los posibles problemas de seguridad, como los ataques de denegación de servicio, ransomware, virus, gusanos, spyware y phishing, ejemplifican por qué los datos confidenciales deben almacenarse y transmitirse de forma segura. El acceso oportuno y confiable a los servicios de datos e información por parte de usuarios autorizados, conocido como disponibilidad, se garantiza a través de un ancho de banda adecuado, copias de seguridad y otras medidas.* ***Conceptos transversales**: Privacidad y seguridad; Relaciones de sistema; Interacción humano-computadora.* ***Conexiones dentro del marco**: 9–12. Algoritmos y programación Algoritmos y Programación.Algoritmos.*

Al final de grado 12: continúa de la página anterior

Datos y Análisis	
COLECCIÓN	Los datos se recopilan o se generan constantemente a través de procesos automatizados que no siempre son evidentes, lo que genera problemas de privacidad. Los diferentes métodos y herramientas de recopilación que se utilizan influyen en la cantidad y la calidad de los datos que se observan y registran. *Los datos se pueden recopilar y agregar a millones de personas, incluso cuando no están participando activa o físicamente cerca de los dispositivos de recolección de datos. Esta colección automatizada y no evidente puede plantear problemas de privacidad, tales como los sitios de redes sociales que extraen una cuenta incluso cuando el usuario no está en línea. Otros ejemplos incluyen el video de vigilancia utilizado en una tienda para rastrear a los clientes por seguridad o información sobre los hábitos de compra o el monitoreo del tráfico en la carretera para cambiar las señales en tiempo real para mejorar la eficiencia de la carretera sin que los conductores estén al tanto. Los métodos y dispositivos para recopilar datos pueden diferir según la cantidad de almacenamiento requerido, el nivel de detalle recopilado y las tasas de muestreo. Por ejemplo, los buscadores de rango ultrasónicos son buenos para distancias largas y son muy precisos, en comparación con los buscadores de rango infrarrojos, que son mejores para distancias cortas. Los modelos y simulaciones por computadora producen grandes cantidades de datos utilizados en el análisis.* ***Conceptos transversales**: Privacidad y seguridad.* ***Conexiones dentro del marco**: 9–12. Sistemas de computación.Dispositivos; 9–12. Impactos de computación.Seguridad, Ley, y Ética.*
ALMACENAMIENTO	Los datos pueden estar compuestos de múltiples elementos de datos que se relacionan entre sí. Por ejemplo, los datos de población pueden contener información sobre la edad, el género y la altura. Las personas toman decisiones sobre cómo se organizan los elementos de datos y dónde se almacenan los datos. Estas opciones afectan el costo, la velocidad, la confiabilidad, la accesibilidad, la privacidad y la integridad. *Un modelo de datos combina elementos de datos y describe las relaciones entre los elementos. Los modelos de datos representan elecciones tomadas sobre qué elementos de datos están disponibles y son factibles de recopilar. El almacenamiento de datos a nivel local puede aumentar la seguridad pero disminuir la accesibilidad. El almacenamiento de datos en un sistema de almacenamiento redundante basado en la nube puede aumentar la accesibilidad pero reducir la seguridad, ya que puede ser accedido en línea fácilmente, incluso por usuarios no autorizados. Las redundancias de datos y las copias de seguridad son útiles para restaurar datos cuando la integridad se ve comprometida.* ***Conceptos transversales**: Relaciones de sistemas; Privacidad y seguridad.* ***Conexiones dentro del marco**: 9–12. Algoritmos y programación. Algoritmos.*

Al final de grado 12: continúa de la página anterior

VISUALIZACIÓN Y TRANSFORMACIÓN	Las personas transforman, generalizan, simplifican y presentan grandes conjuntos de datos de diferentes maneras para influir en cómo otras personas interpretan y comprenden la información subyacente. Los ejemplos incluyen visualización, agregación, reorganización y aplicación de operaciones matemáticas. *Las visualizaciones, como las infografías, pueden oscurecer los datos e influir positiva o negativamente en las opiniones de las personas sobre los datos. Las personas utilizan herramientas de software o programación para crear visualizaciones de datos potentes e interactivas y realizar una serie de operaciones matemáticas para transformar y analizar datos. Los ejemplos de operaciones matemáticas incluyen aquellas relacionadas con la agregación, como sumar y promediar. El mismo conjunto de datos se puede visualizar o transformar para admitir múltiples lados de un problema.* ***Conceptos transversales**: Abstracción; Interacción humano-computadora.* ***Conexiones dentro del marco**: 6–8. Impactos de la computación. Interacciones sociales.*
INFERENCIA Y MODELOS	La precisión de las predicciones o inferencias depende de las limitaciones del modelo computacional y de los datos sobre los que se basa el modelo. La cantidad, calidad y diversidad de datos y las características elegidas pueden afectar la calidad de un modelo y la capacidad de entender un sistema. Las predicciones o inferencias son probadas para validar modelos. *Los grandes conjuntos de datos se utilizan para hacer modelos utilizados para inferencia o predicciones, como la predicción de terremotos, patrones de tráfico o los resultados de accidentes automovilísticos. Las cantidades más grandes y la mayor calidad de los datos recopilados tenderán a mejorar la precisión de los modelos. Por ejemplo, el uso de datos de 1,000 accidentes automovilísticos generalmente generaría un modelo más preciso que el uso de datos de 100 choques. Además, los accidentes automovilísticos proporcionan una amplia variedad de puntos de datos, como la velocidad de impacto, la marca y modelo del automóvil y el tipo de pasajero, y estos datos son útiles para el desarrollo de medidas de prevención de lesiones.* ***Conceptos transversales**: Abstracción; Privacidad y seguridad.*

Al final de grado 12: continúa de la página anterior

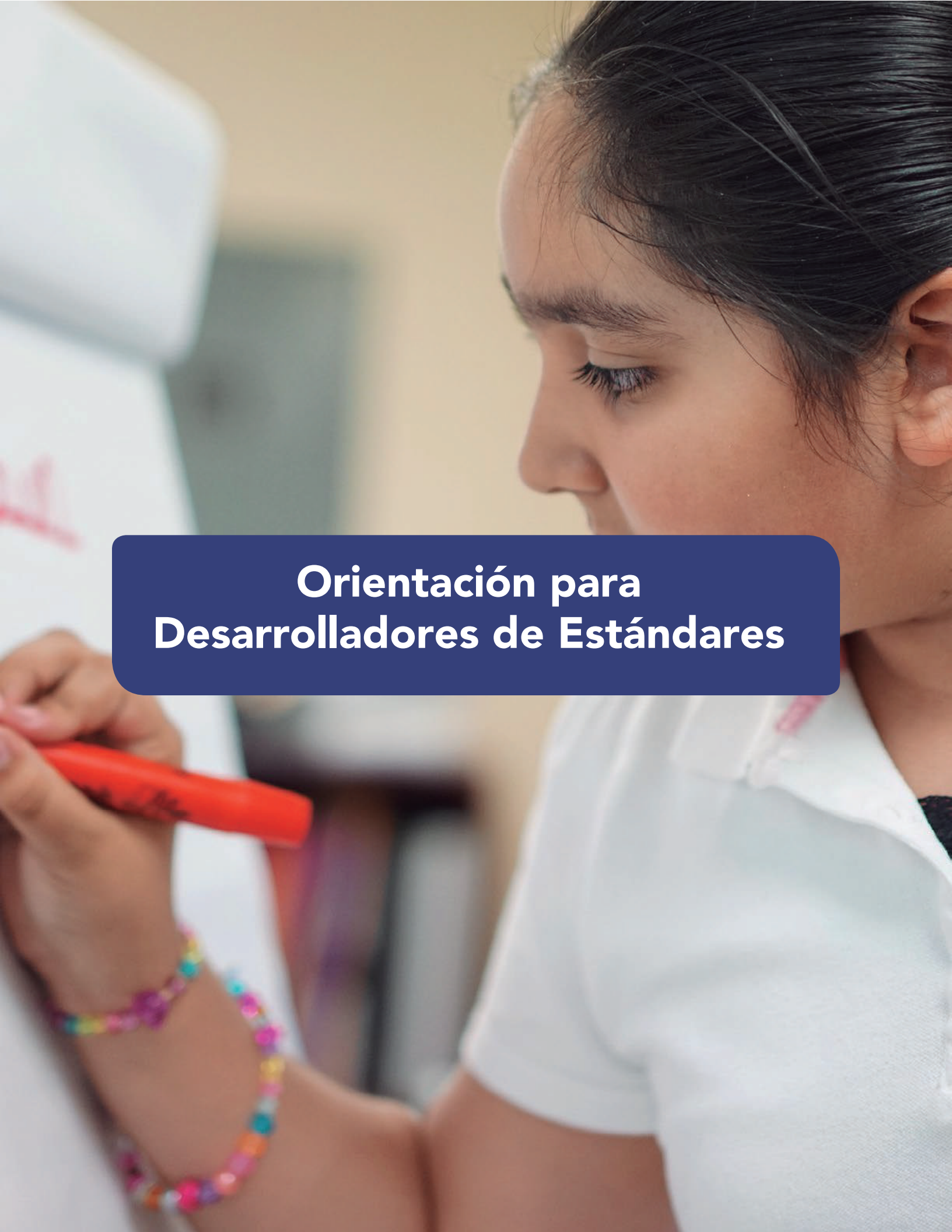
Algoritmos y Programación	
ALGORITMOS	Las personas evalúan y seleccionan algoritmos según el rendimiento, la reutilización y la facilidad de implementación. El conocimiento de algoritmos comunes mejora la forma en que las personas desarrollan software, datos seguros y almacenan información. *Algunos algoritmos pueden ser más fáciles de implementar en un lenguaje de programación particular, trabajar más rápido, requieren menos memoria para almacenar datos y pueden aplicarse en una variedad más amplia de situaciones que otros algoritmos. Los algoritmos utilizados para buscar y ordenar datos son comunes en una variedad de aplicaciones de software. Los algoritmos de cifrado se utilizan para proteger los datos, y los algoritmos de compresión hacen que el almacenamiento de datos sea más eficiente. En este nivel, el análisis puede involucrar cálculos simples de pasos. No se espera un análisis que use una notación matemática sofisticada para clasificar el rendimiento del algoritmo, tal como la notación Big-O.* ***Conceptos transversales**: Abstracción; Privacidad y seguridad.* ***Conexiones dentro del marco**: 9–12. Datos y análisis.Almacenamiento; 9–12. Redes e internet.Seguridad cibernética.*
VARIABLES	Las estructuras de datos se utilizan para gestionar la complejidad del programa. Los programadores eligen estructuras de datos en función de la funcionalidad, el almacenamiento y las compensaciones de rendimiento. *Una lista es un tipo común de estructura de datos que se utiliza para facilitar el almacenamiento eficiente, el pedido y la recuperación de valores y varias otras operaciones en su contenido. Las compensaciones están asociadas con la elección de diferentes tipos de listas. No se espera el conocimiento de estructuras de datos avanzadas, como pilas, colas, árboles y tablas hash. Los tipos definidos por el usuario y la programación orientada a objetos son conceptos opcionales en este nivel.* ***Conceptos transversales**: Abstracción; Relaciones de sistema.* ***Conexiones dentro del marco**: 6–8. Sistemas de computación.Hardware y Software.*
CONTROL	Los programadores consideran compensaciones relacionadas con la implementación, la legibilidad y el rendimiento del programa al seleccionar y combinar estructuras de control. *La implementación incluye la elección del lenguaje de programación, que afecta el tiempo y el esfuerzo necesarios para crear un programa. La legibilidad se refiere a que tan claro está el programa para otros programadores y se puede mejorar a través de la documentación. La discusión del rendimiento se limita a una comprensión teórica del tiempo de ejecución y los requisitos de almacenamiento; No se espera un análisis cuantitativo. Las estructuras de control en este nivel pueden incluir sentencias condicionales, bucles, controladores de eventos y recursión. La recursión es una técnica de control en la que un procedimiento se llama a sí mismo y es apropiado cuando los problemas pueden expresarse en términos de versiones más pequeñas de sí mismos. La recursión es un concepto opcional en este nivel.* ***Conceptos transversales**: Abstracción; Relaciones de sistemas.*

Al final de grado 12: continúa de la página anterior

MODULARIDAD	Los programas complejos se diseñan como sistemas de módulos interactivos, cada uno con una función específica, coordinados para un propósito general común. Estos módulos pueden ser procedimientos dentro de un programa; combinaciones de datos y procedimientos; o programas independientes, pero interrelacionados. Los módulos permiten una mejor gestión de tareas complejas. *Las aplicaciones de software requieren un enfoque sofisticado para el diseño que utiliza una perspectiva de sistemas. Por ejemplo, la programación orientada a objetos descompone los programas en módulos llamados objetos que emparejan datos con métodos (variables con procedimientos). El enfoque en este nivel es entender un programa como un sistema con relaciones entre módulos. La elección de la implementación, como el lenguaje de programación o el paradigma, puede variar.* ***Conceptos transversales**: Relaciones de sistema; Abstracción.* ***Conexiones dentro del marco**: 9–12. Sistemas de computación.Hardware y Software.*
DESARROLLO DE PROGRAMAS	Los equipos diversos pueden desarrollar programas con un amplio impacto a través de una revisión cuidadosa y aprovechando las fortalezas de los miembros en diferentes roles. Las decisiones de diseño a menudo implican compensaciones. El desarrollo de programas complejos es asistido por recursos como bibliotecas y herramientas para editar y administrar partes del programa. El análisis sistemático es crítico para identificar los efectos de los errores persistentes. *A medida que los programas se vuelven más complejos, la elección de los recursos que ayudan al desarrollo del programa se vuelve cada vez más importante. Estos recursos incluyen bibliotecas, entornos de desarrollo integrados y herramientas de depuración. El análisis sistemático incluye la prueba del rendimiento y la funcionalidad del programa, seguido de la prueba del usuario final. A veces, se hace referencia a una compensación común en el desarrollo de programas como "Rápido/Bueno/Barato: Elija dos": se puede desarrollar software rápidamente, con alta calidad o con poco uso de recursos (por ejemplo, dinero o cantidad de personas), pero el gerente del proyecto puede elegir solo dos de los tres criterios.* ***Conceptos transversales**: Interacción humano-computadora; Relaciones de sistema; Abstracción.* ***Conexiones dentro del marco**: 9–12. Sistemas de computación.Resolución de problemas.*

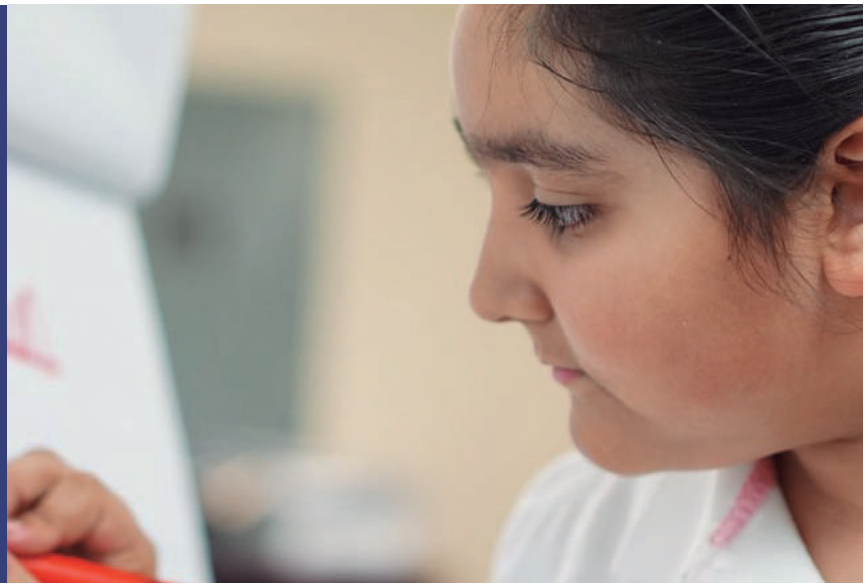
Al final de grado 12: continúa de la página anterior

Impactos de la Computación	
CULTURA	El diseño y el uso de tecnologías y artefactos computacionales pueden mejorar, empeorar o mantener un acceso desigual a la información y las oportunidades. *Si bien muchas personas tienen acceso directo a la informática a lo largo del día, muchas otras aún no cuentan con suficiente atención o simplemente no tienen acceso. Algunos de estos desafíos están relacionados con el diseño de tecnologías de computación, como en el caso de tecnologías que son difíciles de usar para personas mayores y personas con discapacidades físicas. Otros déficits de equidad, tales como la exposición mínima a la computación, el acceso a la educación y las oportunidades de capacitación, están relacionados con problemas sistémicos más grandes en la sociedad.* ***Conceptos transversales**: Interacción humano-computadora; Relaciones de sistema.* ***Conexiones dentro del marco**: 9–12. Sistemas de computación. Dispositivos.*
INTERACCIÓN SOCIAL	Muchos aspectos de la sociedad, especialmente las carreras, se han visto afectados por el grado de comunicación que ofrece las ciencias de la computación. El aumento de la conectividad entre personas en distintas culturas y en diferentes campos profesionales ha cambiado la naturaleza y el contenido de muchas carreras. *Las carreras han evolucionado, y han surgido nuevas carreras. Por ejemplo, los administradores de redes sociales aprovechan las plataformas de redes sociales para guiar la presencia de un producto o compañía y aumentar la interacción con su audiencia. La conectividad global también ha cambiado la forma en que los equipos en diferentes campos, como la computación y la biología, trabajan juntos. Por ejemplo, la base de datos genéticos en línea disponible por el Proyecto del Genoma Humano, los algoritmos necesarios para analizar los datos y la capacidad de los científicos de todo el mundo para compartir información han acelerado el ritmo de los descubrimientos médicos y llevado al nuevo campo de la biología computacional.* ***Conceptos transversales**: Relaciones de sistema; Interacción humano-computadora.* ***Conexiones dentro del marco**: 9–12. Redes e Internet.Red de comunicación y organización.*
SEGURIDA., LEY Y ÉTICA	Las leyes rigen muchos aspectos de las ciencias de la computación, tales como la privacidad, los datos, las propiedades, la información y la identidad. Estas leyes pueden tener efectos beneficiosos y perjudiciales, tales como acelerar o retrasar los avances en ciencias de la computación y proteger o infringir los derechos de las personas. Las diferencias internacionales en leyes y ética tienen implicaciones para la computación. *Las cuestiones legales en las ciencias de la computación, tales como las relacionadas con el uso de Internet, cubren muchas áreas del derecho, reflejan un campo tecnológico en evolución y pueden implicar concesiones. Por ejemplo, las leyes que obligan al bloqueo de algunos sitios web para compartir archivos pueden reducir la piratería en línea, pero pueden restringir el derecho a la libertad de información. Los cortafuegos se pueden usar para bloquear virus y malware dañinos, pero también se pueden usar para censurar los medios. El acceso a ciertos sitios web, como los sitios de redes sociales, puede variar según las leyes de una nación y puede estar bloqueado con fines políticos.* ***Conceptos transversales**: Relaciones de sistema; Privacidad y seguridad; Abstracción.* ***Conexiones dentro del marco**: 9–12. Datos y análisis.Colección.*



Orientación para Desarrolladores de Estándares

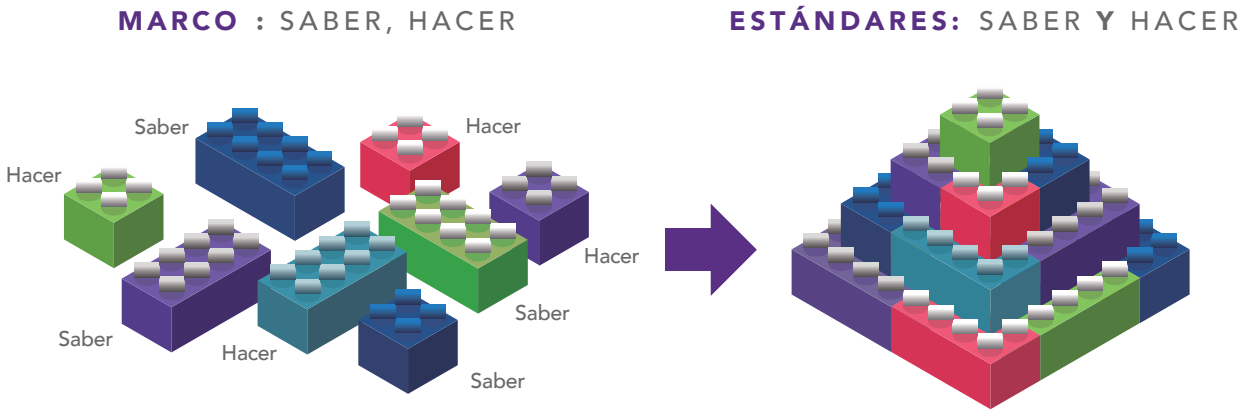
7



Orientación para Desarrolladores de Estándares

El marco de las ciencias de la computación K–12 está diseñado para servir como una base, desde la cual todos los estados, distritos y organizaciones pueden desarrollar estándares de educación en ciencias de la computación para los estudiantes de K-12. Los estándares desempeñan un papel vital en el logro de la visión de las ciencias de la computación para todos los estudiantes. Se democratizan las ciencias de la computación al establecer objetivos de aprendizaje para todos los estudiantes y la expectativa de que todas las escuelas brinden oportunidades para alcanzar esos objetivos, de manera que todos los niños, independientemente de su edad, raza, género, discapacidad, nivel socioeconómico o a qué escuela asisten, sean capaz de tener experiencias atractivas y rigurosas de las ciencias de la computación. Como se ilustra en la Figura 7.1, el marco proporciona los componentes básicos mediante los cuales los estados pueden desarrollar sus propios estándares.

Figura 7.1: Bloques de construcción para los estándares



Los estándares son un componente esencial de un plan de educación más amplio y pueden proporcionar una base con la cual alinear los otros componentes, como el currículo, la instrucción, el desarrollo profesional y la evaluación, para preparar mejor a los estudiantes para el éxito en la universidad y en el lugar de trabajo. También comunican los objetivos principales de aprendizaje a los responsables políticos, administradores, profesores, padres y estudiantes. Los estándares computacionales brindan información sobre una disciplina que será nueva para muchos profesores y ofrecerá puntos de partida inspiradores para crear proyectos, lecciones y actividades. Los estándares facilitan el intercambio de contenido, como las lecciones, entre los profesores y son una forma útil de categorizar ese contenido para facilitar la búsqueda y recuperación. Los estándares consistentes promueven la alineación y las conexiones entre los diferentes distritos dentro de un estado, de modo que si un estudiante se muda a una escuela diferente, no terminará con expectativas diferentes.

El propósito de esta guía es proporcionar información y recomendaciones para el desarrollo de los estándares de educación en ciencias de la computación K–12

- al principio para establecer los criterios y preparar escritores de estándares,
- durante el proceso de escritura con ejemplos y ejercicios, y
- después para ayudar a evaluar el resultado.

Esta guía fue desarrollada en colaboración con la organización educativa sin fines de lucro Achieve, basada en recomendaciones para desarrolladores de estándares del Consejo Nacional de Investigación (NRC, 2012). También utiliza criterios y procedimientos que Achieve ha establecido y refinado basándose en aspectos de estándares de calidad de contenido académico.

Estas categorías se describen en Tabla 7.1:

Tabla 7.1: Resumen - Orientación para desarrolladores de estándares

CRITERIO	RESUMEN
Rigor: ¿Cuál es la exigencia intelectual de los estándares?	El rigor es el sello distintivo por excelencia de estándares ejemplares. Es la medida de cuán cerca un conjunto de estándares representa el contenido y la demanda cognitiva necesaria para que los estudiantes tengan éxito en cursos universitarios con crédito sin remediación y en trabajos de nivel de entrada, de alta calidad y de alto crecimiento. Recomendamos que los escritores de estándares establezcan y articulen el nivel apropiado de rigor en ciencias de la computación para preparar a todos los estudiantes para el éxito en la universidad y las carreras.
Enfoque/Manejabilidad: ¿Se han tomado decisiones sobre qué es lo más importante para que los estudiantes aprendan y que hay una cantidad manejable de contenido?	Los estándares de alta calidad establecen prioridades sobre los conceptos y habilidades que deben adquirirse al graduarse de la escuela secundaria. Las elecciones deben basarse en el conocimiento y las habilidades esenciales para que los estudiantes tengan éxito en la educación postsecundaria y en el mundo del trabajo. Un enfoque agudo también ayuda a asegurar que el conocimiento y las habilidades acumulativas que se espera que los estudiantes aprendan sean manejables. Recomendamos estándares de nivel de grado que comuniquen claramente las expectativas de los estudiantes en cada etapa. En el caso de los estándares de nivel de grado, recomendamos que se brinde orientación a los usuarios para crear sus propios estándares de nivel de grado o estándares de mapeo para cursos específicos.

La tabla continúa en la página siguiente

Continuación de la tabla de la página anterior

Especificidad: ¿Son los estándares lo suficientemente específicos para transmitir el nivel de desempeño esperado de los estudiantes?	Los estándares de alta calidad son precisos y proporcionan detalles suficientes para transmitir el nivel de rendimiento esperado sin ser demasiado prescriptivo. Los estándares que mantienen un nivel de precisión relativamente consistente ("tamaño de grano") son más fáciles de entender y usar. Aquellos que son demasiado amplios o vagos dejan demasiado abierto a la interpretación, lo que aumenta la probabilidad de que los estudiantes tengan diferentes niveles de desempeño, mientras que los estándares que son demasiado prescriptivos fomentan un enfoque de lista de verificación para la enseñanza y el aprendizaje que socava las oportunidades de los estudiantes para demostrar su entendiendo de manera equitativa. Recomendamos que los desarrolladores de estándares escriban estándares que no sean ni demasiado amplios ni demasiado específicos y que el tamaño del grano sea consistente en todos los estándares.
Equidad/Diversidad: ¿Fueron los estándares escritos para todos los estudiantes por un conjunto diverso de escritores y revisores? ¿Pueden los estudiantes demostrar un desempeño de múltiples maneras?	Los estándares, al igual que otros aspectos de la infraestructura educativa, desempeñan un papel en la creación de un entorno equitativo para todos los estudiantes. Recomendamos que se preste atención a la diversidad y la equidad no solo en la composición de los grupos que escriben, asesoran y revisan los estándares, sino también en el contenido de los estándares mediante el diseño de estándares en los que TODOS los estudiantes pueden participar y que sean lo suficientemente flexibles para permitirles demostrar dominio de múltiples maneras.
Claridad/Accesibilidad: ¿Están los estándares claramente escritos y presentados en un formato legible y fácil de usar y accesible al público en general?	La claridad requiere algo más que una prosa simple y sin jerga que esté libre de errores. Los estándares también deben comunicarse en un lenguaje que pueda obtener una aceptación generalizada no solo por parte del personal docente de educación superior, sino también por parte de empleadores, profesores, padres, juntas escolares, legisladores y otros que tienen interés en la escolarización. Un formato sencillo y funcional facilita el acceso del usuario. Recomendamos que los redactores de estándares consideren el nivel de conocimiento de los usuarios de los estándares al aclarar los términos y proporcionar ejemplos.
Coherencia/Progresión: ¿Los estándares transmiten una visión unificada de la disciplina, establecen conexiones entre las principales áreas de estudio y muestran una progresión significativa del contenido en todos los grados?	La forma en que se clasifican y desglosan los estándares en hilos de apoyo debe reflejar una estructura coherente de la disciplina y/o revelar relaciones significativas entre los hilos y cómo el estudio de una complementa el estudio de otra. Si los estándares sugieren una progresión, esa progresión debe ser significativa y apropiada en todos los grados o tramos. Recomendamos que los redactores de estándares comuniquen claramente las progresiones de contenido y prácticas en los estándares.
Medición: ¿Es cada estándar mensurable, observable o verificable de alguna manera?	En general, los estándares deben centrarse en los resultados, en lugar de los procesos de enseñanza y aprendizaje. Los estándares deben usar verbos de desempeño que requieran que los estudiantes demuestren conocimientos y habilidades y deben evitar usar aquellos que se refieren a actividades de aprendizaje, tales como examinar, investigar y explorar, o a procesos cognitivos que son difíciles de verificar, como apreciar. Recomendamos asegurar que cada estándar sea mensurable.
Integración de prácticas y conceptos: ¿Cada estándar refleja al menos una práctica y un concepto?	Para asegurar que la instrucción refleje saber y hacer ciencia de la computación, los conceptos core de las ciencias de la computación deben enseñarse junto con las prácticas integrándolas completamente al nivel de los estándares. Recomendamos que los estándares integren las prácticas de las ciencias de la computación con las declaraciones conceptuales.
Conexiones a otras disciplinas: ¿Existen formas explícitas en las que se demuestre que las ciencias de la computación son relevantes en otras materias?	Hay muchas áreas posibles de superposición entre las ciencias de la computación y áreas de materias tales como matemáticas, ciencias e ingeniería, así como humanidades, incluidos idiomas, estudios sociales, arte y música. Hacer conexiones intencionales entre los estándares de ciencias de la computación y los estándares académicos en otras disciplinas promoverá una experiencia de educación más coherente. Recomendamos que los estándares de ciencias de la computación se escriban para alinearse con los estándares estatales de matemáticas y ciencias, así como los estándares de otras disciplinas.

Recomendaciones

Recomendación 1: Rigor. Recomendamos que los estándares establezcan y articulen el nivel apropiado de rigor en ciencias de la computación para preparar a todos los estudiantes para el éxito en la universidad y las carreras.

Los estándares de alta calidad crean expectativas fundamentales para todos los estudiantes, en lugar de solo aquellos interesados en estudios avanzados, y los preparan para una variedad de experiencias postsecundarias.

Los estándares apuntan a preparar a los estudiantes para las demandas del mundo que encontrarán después de la graduación. Esa preparación es aún más difícil cuando el mercado laboral cambia rápidamente a medida que la influencia de la tecnología en la fuerza laboral crece constantemente. Por lo tanto, es fundamental que los estándares describan expectativas rigurosas en ciencias de la computación para todos los estudiantes. Además, algunos estudiantes querrán especializarse en los campos de las ciencias de la computación y requerirán un nivel de demanda intelectual aún mayor que el necesario para todos los estudiantes.

Para facilitar el uso apropiado de los estándares, diferenciando entre estándares de carrera técnica para cursos avanzados y estándares académicos básicos para todos los estudiantes es crucial. El primero puede ser equivalente a las expectativas de cursos especializados en informática, en particular, carreras y vías de educación técnica. En contraste, los estándares para todos los estudiantes describen expectativas que serán importante que cada estudiante se reúna para ayudar a asegurar su éxito futuro en cualquier campo elegido.

Por ejemplo, los diferentes estándares en la Figura 7.2 se basan en la misma práctica y concepto en la banda de grados 9–12 del Marco de Ciencias de la Computación K–12, y comparan un estándar para un curso avanzado con un estándar para todos los estudiantes.

Figura 7.2: Diferenciar rigor para todos los alumnos

Práctica: Pruebas y refinación de artefactos computacionales.	Concepto: El análisis sistemático es crítico para identificar los efectos de los errores persistentes. (9-12.Algoritmos y programación.Desarrollo del programa)

Ejemplo 1: Pruebe y refine los componentes del software utilizando pruebas unitarias para identificar errores persistentes durante un ciclo de desarrollo de programación ágil.

Ejemplo 2: Pruebe y refine un programa utilizando un proceso de depuración sistemático como parte de un proceso de desarrollo iterativo más amplio.

El primer estándar sería más apropiado para estudiantes de secundaria en un curso especializado de educación técnica y de carrera, ya que requiere un producto (componentes de software) y metodologías (pruebas unitarias y desarrollo ágil) que sean específicos para la industria del software. El segundo estándar establece una meta para todos los estudiantes que refleja un producto más general (cualquier programa de computadora) y aun así mantiene el rigor a través de la expectativa de un proceso sistemático e iterativo.

Los estándares para todos los estudiantes deben tener el rigor suficiente para ayudarlos a prepararse para ingresar y tener éxito en los cursos postsecundarios básicos que requieren habilidades como el pensamiento crítico, la resolución de problemas y la alfabetización computacional. El rigor se aplica igualmente a las prácticas y conceptos.

El marco de las ciencias de la computación K–12 se escribió para describir una visión de la educación de las ciencias de la computación para todos los estudiantes, por lo que la mayoría de los estándares basados en el marco se podrían escribir en un nivel de rigor destinado a todos los estudiantes, en lugar de hacerlo para los estudiantes en cursos avanzados. Se debe tener cuidado de alinear los estándares con las habilidades de los estudiantes apropiadas para su grado. Es posible que la retroalimentación o las restricciones actuales del sistema puedan influenciar a los escritores de estándares para tratar de limitar el rigor de los estándares, particularmente en los niveles de primaria. Sin embargo, la investigación sobre el uso de los estudiantes de la secuencia y la iteración y la práctica de otros aspectos del pensamiento computacional indica que los estudiantes pueden aprender las ciencias de la computación a edades tempranas (Flannery et al., 2013) cuando tengan el apoyo y las oportunidades para hacerlo. Los redactores de estándares deben tener cuidado de mantener el rigor a un nivel lo suficientemente alto como para que los estudiantes más jóvenes puedan asegurar que todos los estudiantes tengan acceso a una educación de las ciencias de la computación de alta calidad. Las declaraciones conceptuales en las bandas de grados K–2 y 3–5 del marco han sido revisadas por expertos en educación en computación durante la infancia y proporcionan un plan para las expectativas apropiadas para los estudiantes de edad primaria. Consulte la Figura 7.3 para conocer los criterios para determinar si un estándar tiene la cantidad correcta de rigor.

Las ciencias de la computación se aplican a muchas carreras y diplomas universitarias:

Negocio:
Los profesionales de negocios pueden aplicar los procesos aprendidos en ciencias de la computación para expandir un negocio y optimizar su eficiencia.

Música:
Los músicos pueden diseñar sonidos, efectos y filtros. Pueden crear un sistema para controlar la música usando gestos para manipular sonidos y elementos visuales para un show en vivo.

Biología:
los investigadores pueden analizar una base de datos de secuencias genéticas para genes similares a un gen de cáncer conocido.

Deportes:
los entrenadores pueden crear algoritmos para analizar el rendimiento de los atletas como una herramienta de entrenamiento o desarrollar estrategias utilizando datos en tiempo real en el campo.

Figura 7.3: Determinar la cantidad correcta de rigor para un estándar.

Un estándar debe cumplir con estos tres criterios:

- ¿El estándar requiere un nivel apropiado de comprensión conceptual?
- ¿Requiere la aplicación de ese concepto?
- ¿Requiere compromiso con una práctica?

Para ayudar a asegurar que los estándares establezcan expectativas que preparen a los estudiantes para el éxito en los cursos postsecundarios de nivel de ingreso y las carreras profesionales, los comentarios de los empleadores y miembros de la facultad, incluso de instituciones de dos años, son cruciales. La participación de los revisores con una perspectiva sobre la preparación de los estudiantes para cursos postsecundarios y carreras proporcionará información valiosa sobre el rigor necesario en los estándares.

Recomendación 2: Enfoque/Manejabilidad. Recomendamos que los estándares sean limitadas en número, se centren en el contenido y las prácticas descritas en el marco y se escriban para niveles de grado individuales o cursos.

Los estándares de alta calidad dan prioridad a los conceptos y habilidades que deben ser adquiridos por los estudiantes. Un enfoque agudo ayuda a asegurar que los conocimientos y las habilidades que se espera que los estudiantes aprendan sean importantes y manejables en cualquier grado o curso.

Un enfoque claro dentro de los estándares ayuda a los profesores a ver y priorizar las experiencias de aprendizaje para los estudiantes. Por lo tanto, el marco se desarrolló para describir un conjunto básico de conceptos y prácticas, que se seleccionaron según los criterios desarrollados por el equipo de redacción y examinados por la comunidad de las ciencias de computación durante los períodos de revisión. Los estándares basados en el marco deben centrarse en el conjunto de conceptos y prácticas que se describen aquí, en lugar de incorporar temas adicionales que podrían incluirse en cursos avanzados de las ciencias de la computación.[1] Consulte la Tabla 7.2 para ver ejemplos de temas importantes que son esenciales o no esenciales para todos los estudiantes aprender.

1 Los temas adicionales serían apropiados para los estándares de los cursos avanzados, si están claramente designados como tales y no como estándares para todos los estudiantes (consulte la Recomendación 1).

Tabla 7.2: Ejemplos de temas esenciales y no esenciales

IMPORTANTE Y ESENCIAL PARA TODOS LOS ESTUDIANTES	IMPORTANTE PERO NO ESENCIAL PARA TODOS LOS ESTUDIANTES:
• Estrategias de resolución de problemas	• Sistemas operativos
• Búsqueda y clasificación	• Eficiencia algorítmica
• Representaciones de datos digitales	• Bases de datos relacionales
• Medidas básicas de seguridad en línea	• Métodos de criptografía

Este enfoque ayudará a asegurar que el tiempo limitado disponible para la educación en ciencias de la computación en K–12 se concentre en aquellas áreas que son prioritarias para todos los estudiantes. Se podrían agregar estándares adicionales para los cursos electivos de las ciencias de la computación, pero esos deben ser anotados como electivos y no para todos los estudiantes. La figura 7.4 proporciona un ejemplo de un estándar enfocado apropiadamente en el concepto.

Figura 7.4: Enfocarse en el concepto

Práctica: Reconocimiento y definición de problemas computacionales.
Concepto: las diferentes herramientas de software utilizadas para acceder a los datos pueden almacenar los datos de manera diferente. El tipo de datos que se almacenan y el nivel de detalle representado por esos datos afectan los requisitos de almacenamiento. (3–5.Datos y análisis.Almacenamiento).

Estándar que se enfoca en el concepto: evaluar la conveniencia de las diferentes maneras de almacenar datos según el tipo de datos y el nivel de detalle.

Estándar que incluye conceptos extraños: evalúe la adecuación de las representaciones binarias, octales y hexadecimales de los datos y convierta entre bits y bytes.

Otro aspecto del enfoque apropiado es que los estándares se desarrollan para niveles de grado o cursos específicos. Aunque las declaraciones del marco se escriben para las bandas de grado (es decir, K–2, 3–5, 6–8, 9–12) y, más precisamente, los puntos finales de la banda de grado, los estándares desarrollados a partir del marco deben escribirse para los niveles de grado individuales. Por ejemplo, las expectativas del marco para el final del quinto grado (Grados 3–5) pueden informar los estándares en los tres niveles de grado (grados 3, 4 y 5) o solo en el grado 5. Si los estándares de nivel de grado no son posibles, se debe proporcionar orientación sobre cómo los usuarios de los estándares pueden crear su propio nivel de grado o las expectativas de los estudiantes específicas del curso. Reducir el enfoque de las metas de los estudiantes en cada nivel de grado o curso, ya sea por redactores de estándares o por administradores del distrito y del estado, permitirá la alineación en todo el sistema educativo y asegurará que los profesores tengan el apoyo que necesitan para centrarse en estándares particulares durante un curso o nivel de grado.

Recomendación 3: Especificidad. Recomendamos que los redactores de estándares atiendan la especificidad de los estándares para asegurar que no sean ni demasiado amplios ni demasiado específicos y que el tamaño del grano, cuando sea posible, sea consistente en todos los estándares.

Los estándares de alta calidad son precisos y proporcionan detalles suficientes para transmitir el nivel de rendimiento esperado sin ser demasiado prescriptivo. Aquellos que mantienen un nivel de precisión relativamente consistente tienden a tener una interpretación y un uso consistentes. Por el contrario, los que son demasiado amplios o vagos dejan demasiado abierto a la interpretación y se implementan de manera inconsistente, y los estándares demasiado específicos reducen las oportunidades de los estudiantes para demostrar su comprensión de manera flexible.

Escribir los estándares a un nivel útil de especificidad requiere un equilibrio entre ser demasiado vago y específico (ver Figura 7.5). Un nivel de especificidad consistente y apropiado ayudará a asegurar que los profesores tengan la comprensión y el apoyo que necesitan para ayudar a los estudiantes a alcanzar los estándares. Cuando los estándares son demasiado amplios, un profesor debe interpretar la intención de los estándares: decidir qué tipos de conexiones deben entenderse y qué complejidad de problemas deben resolverse. La especificidad útil a menudo se puede agregar con declaraciones de límites, que especifican que contenido no se espera, aclarando el alcance del material que se debe enseñar. Por ejemplo, los estudiantes al final del octavo grado deben saber que existen protocolos de red para permitir que diferentes computadoras se comuniquen entre sí, pero no la estructura de los mensajes enviados mediante un protocolo específico, como HTTP (Hypertext Transfer Protocol).

Figura 7.5: Un espectro de especificidad en los estándares

	Estandard	Comentarios
Demasiado vago	Usa condicionales en un programa.	Este estándar carece de contexto y es demasiado vago para ser evaluado.
Equilibrado	Diseñe un algoritmo que utilice de manera eficiente sentencias condicionales para representar múltiples ramas de ejecución.	Este estándar especifica el tipo de producto y un nivel de rigor, pero permite múltiples contextos en los que demostrar el rendimiento.
Demasiado específico	Crea una aplicación para ayudar a los amigos a decidir entre ver una película de comedia, acción o ciencia ficción mediante el uso de tres declaraciones de tipo "if/sí.	El contexto para este estándar es demasiado específico y no permite un rango de demostraciones de desempeño.

La consistencia en el nivel de especificidad en todos los estándares también es importante (ver Figura 7.6). En la práctica, se puede interpretar que los estándares dentro del mismo documento tienen niveles de especificidad iguales y, por lo tanto, pueden asignarse cantidades iguales de tiempo de instrucción. Es más difícil para los educadores, diseñadores de currículos y desarrolladores de evaluaciones usar estándares que varían en el alcance a través de los niveles de grado.

Figura 7.6: Calibrando la especificidad entre escritores de estándares

Cree un conjunto de tres a cinco estándares que varían en especificidad y tienen diferentes escritores de estándares (como grupos pequeños o individuos) que los ponen en orden y los comparan. Discuta las diferencias y las características de cada estándar, luego seleccione uno o dos ejemplos de especificidad que el grupo debe apuntar al escribir estándares.

Recomendación 4: Equidad/Diversidad. Recomendamos que la diversidad y la equidad sean atendidas mediante el desarrollo de estándares que permitan la participación de TODOS los estudiantes y permitan flexibilidad en cómo los estudiantes pueden demostrar competencia. La composición de los grupos de partes interesadas (Stakeholders) que escriben y revisan los estándares debe ser diversa.

El marco se basa en la creencia de que todos los estudiantes, independientemente de su raza, género, clase socioeconómica o discapacidad, cuando reciben el apoyo adecuado, pueden aprender todos los conceptos y prácticas descritos en el marco.

Los estándares equitativos crean expectativas para los estudiantes con una variedad de intereses universitarios y profesionales, permiten demostraciones flexibles de rendimiento, no asumen la preparación fuera de la escuela y están escritos por partes interesadas con diversas perspectivas. Los estándares que se crean para todos los estudiantes se centran en los aspectos centrales de las ciencias de la computación que se aplican a una amplia gama de opciones universitarias y profesionales, en lugar de contenido extraño con una aplicación limitada.

Los conceptos y prácticas del marco de las ciencias de la computación K–12 representan la alfabetización en ciencias de la computación para todos los estudiantes, no solo para los estudiantes interesados en especializarse en el campo o seguir carreras técnicas. Si se espera que todos los estudiantes reciban educación en ciencias de la computación, también debe ser equitativa y permitir que los estudiantes demuestren sus conocimientos y habilidades de múltiples maneras.

Cuando un estándar es particularmente prescriptivo, como cuando se asemeja al alcance ("tamaño de grano") de un elemento de evaluación, prescribe una manera particular en que los estudiantes deben demostrar su comprensión, creando el potencial para un ambiente de salón de clases injusto. Los estándares equitativos no están sesgados a favor o en contra de los estudiantes de un contexto particular. Esto incluye hacer que los estándares sean accesibles para estudiantes con necesidades especiales o estudiantes de inglés.

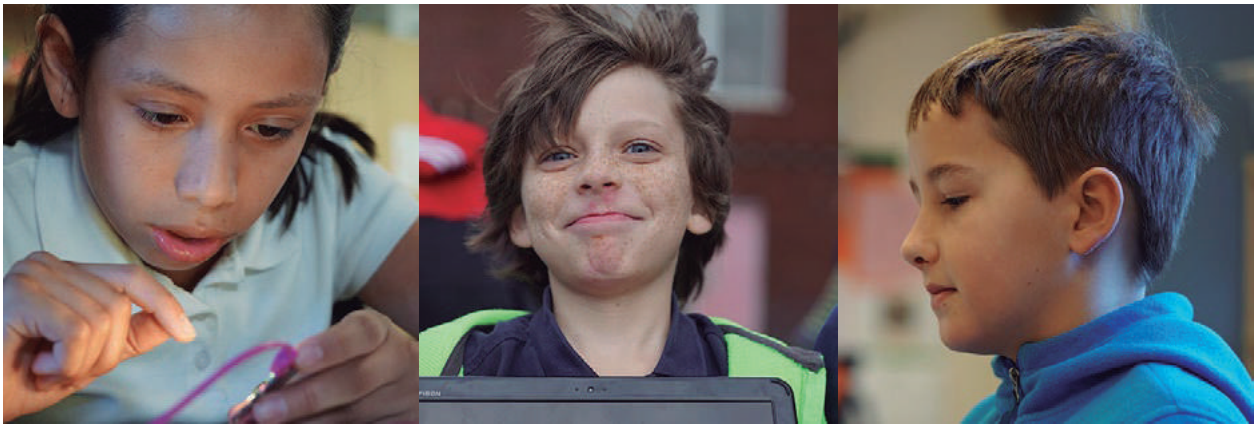
Los estándares equitativos no presuponen el conocimiento del contenido, y por lo tanto un nivel de preparación, en ciencias de la computación, sino que incluyen todas las etapas clave en una progresión de aprendizaje. Las progresiones de aprendizaje incompletas requieren oportunidades fuera de la escuela para llenar los vacíos en el conocimiento, poniendo a los estudiantes sin estas experiencias en desventaja.

El desarrollo de estándares equitativos requiere diversas partes interesadas (Stakeholders). Los escritores y revisores que participan en el desarrollo de los estándares deben incluir una representación diversa de instituciones de dos y cuatro años; la comunidad investigadora industria; y lo más importante, la educación K–12, incluida la experiencia durante la infancia, los estudiantes del idioma inglés y los estudiantes con discapacidades. Esta diversidad ayudará a asegurar que las diferentes perspectivas y áreas de experiencia estén involucradas en la decisión de desarrollo de cada estándar y que los escritores y revisores puedan revisar cada declaración y ejemplo para detectar posibles sesgos. Por ejemplo, la creación de estándares que requieren equipos o software específicos que no son fácilmente accesibles perjudicará a ciertos grupos, como las comunidades rurales o pobres.

Recomendación 5: Claridad/Accesibilidad. Recomendamos que los redactores de estándares aclaren los estándares para el usuario promedio de los estándares, incluyendo la definición de términos y el suministro de ejemplos.

Los estándares de alta calidad están claramente escritos y presentados en un formato legible, fácil de usar y sin errores, al que pueden acceder tanto los instructores seleccionados como el público en general.

Escribir estándares claros y accesibles es un reto. Como expertos en contenido, los escritores tienden a desviarse hacia el lenguaje técnico. Además, los científicos de las ciencias de la computación pueden utilizar términos de muy diferentes maneras que muchos de los usuarios de los estándares. La precisión en el significado es importante, pero también es la conciencia de la audiencia que leerá e implementará los estándares. En todos los casos, los redactores de estándares deben prestar atención a la comprensión técnica del usuario, así como al contenido real del estándar.



Los redactores de estándares de las ciencias de la computación deben considerar la posible comprensión técnica del usuario promedio, dados los escenarios actuales en los que se enseña las ciencias de la computación. En lugar de disminuir el rigor, los escritores deberían considerar cómo enmarcar el lenguaje de los estándares para que sea accesible a los educadores que están enseñando ciencias de la computación fuera de su área principal de certificación y pueden no tener una formación en ciencias de la computación. En la mayoría de las escuelas primarias, los profesores son generalistas, sin capacitación especial en ciencias de la computación. Los formuladores de políticas y los miembros de la comunidad también deben entender las prioridades educativas comunicadas por los estándares. Por lo tanto, es fundamental que los estándares de la informática sean accesibles para muchas audiencias diferentes.

El uso preciso del lenguaje es muy útil para crear una comprensión común de los resultados de los estudiantes entre varios usuarios, como educadores, desarrolladores de currículos y diseñadores de evaluaciones. Las aclaraciones podrían llegar a través de declaraciones de límites que describan los límites de los estándares; notas entre paréntesis dentro de las propias normas; o declaraciones separadas, no evaluables, que acompañan a los estándares. Esto es particularmente cierto cuando palabras como abstracción, paralelización e incluso algoritmos pueden usarse de manera diferente en diferentes disciplinas. Los términos técnicos deben definirse y, tan a menudo como sea posible, agregar expresiones de lenguaje sencillo para que los lectores, especialmente los profesores, puedan entender y aplicar los estándares de manera consistente tanto para el currículo como para la evaluación. Las explicaciones, un lenguaje más simple y/o descripciones detalladas serían útiles para asegurar una aplicación consistente de los estándares (ver Figura 7.7).

Figura 7.7: Ejemplo de términos técnicos versus lenguaje simple en los estándares

Estándar 1: utilice una API llamando a un procedimiento y suministrando argumentos con los tipos de datos adecuados para emplear eficientemente una funcionalidad de alto nivel.

Estándar 2: seleccione y use un procedimiento de una biblioteca de procedimientos (API) y proporcione información apropiada como argumentos para reemplazar el código repetitivo.

El segundo estándar conserva "IPA" (Interfaz de Programación de Aplicaciones), agrega una redacción más accesible como "biblioteca de procedimientos", y presenta el término de programación específico "argumento" con la "entrada apropiada" más general. El segundo estándar continúa usando los términos "procedimiento" y "argumentos", ya que son términos necesarios que proporcionan claridad. Los estándares accesibles utilizan terminología clave para proporcionar claridad y evitar términos extraños y jerga técnica.

Los ejemplos son muy útiles para comunicar la intención de los estándares a los usuarios. Sin embargo, cuando se utilizan ejemplos, recomendamos que siempre haya varios ejemplos presentes. El uso de ejemplos individuales a menudo puede parecer un factor limitante o una prescripción inadvertida del currículo (Achieve, 2010).

Recomendación 6: Coherencia/Progresión. Recomendamos que los estándares se organicen como progresiones que apoyen el aprendizaje de contenidos y prácticas de los alumnos entre varios grados.

La coherencia se refiere a qué tan bien un conjunto de estándares transmite una visión unificada de la disciplina, estableciendo conexiones entre las principales áreas de estudio y mostrando una progresión significativa del contenido a través de los niveles de grado y la duración de los grados.

La investigación sobre el aprendizaje de los estudiantes indica que los estudiantes necesitan ayuda explícita para conectar nuevas ideas con ideas que se han aprendido anteriormente (Marzano, 2004). Para apoyar a los profesores mientras ayudan a los estudiantes a hacer estas conexiones, los estándares deben describir los niveles de desarrollo de una progresión de aprendizaje apropiados para el desarrollo, y las progresiones de aprendizaje incorporadas en los estándares deben ser evidentes para los usuarios. Esto es cierto tanto para el contenido como para las prácticas, ya que las instalaciones de los estudiantes con cada una de las prácticas cambian y se profundizan con el tiempo cuando se les brinda oportunidades adecuadas de instrucción. Las pantallas separadas que muestran la progresión de cada dimensión a través de K-12 han sido muy útiles para los educadores en la implementación de estándares.

Los redactores del marco tuvieron el cuidado de describir progresiones coherentes de contenido y habilidades a lo largo de las bandas de grado. Sin embargo, los estándares basados en el marco pueden escribirse para niveles de grado individuales. En ese caso, se debe tener cuidado para asegurar que la progresión de nivel de grado a nivel de grado sea coherente y basada en la investigación tanto como sea posible y que el conocimiento y la práctica de los estudiantes se basen en el contenido y las habilidades aprendidas anteriormente. Las progresiones en el marco giran en torno a un subconcepto central en cada área de concepto central y reflejan hitos de desarrollo apropiados que crecen en sofisticación desde el jardín de infantes hasta el grado 12 (consulte la Figura 7.8).

Figura 7.8: Ejemplo de progresión de aprendizaje

Sistemas de computacion. Hardware y Software:

Al final de grado 2: Un sistema de computación está compuesto por hardware y software. El hardware consta de componentes físicos, mientras que el software proporciona instrucciones para el sistema. Estas instrucciones están representadas en una forma que una computadora puede entender.

Al final de grado 5: El hardware y el software funcionan juntos como un sistema para realizar tareas, como enviar, recibir, procesar y almacenar unidades de información como bits. Los bits sirven como la unidad básica de datos en los sistemas informáticos y pueden representar una variedad de información.

Al final de grado 8: El hardware y el software determinan la capacidad de un sistema de computación para almacenar y procesar información. El diseño o la selección de un sistema de computación involucra múltiples consideraciones y posibles compromisos, tales como funcionalidad, costo, tamaño, velocidad, accesibilidad y estética.

Al final de grado 12: Existen niveles de interacción entre el hardware, el software y el usuario de un sistema de computación. Los niveles más comunes de software con los que un usuario interactúa incluyen el software y las aplicaciones del sistema. El software del sistema controla el flujo de información entre los componentes de hardware utilizados para la entrada, salida, almacenamiento y procesamiento.

Recomendación 7: Mensurabilidad. Recomendamos asegurar de que cada estándar sea objetivo y mensurable.

Los estándares deben centrarse en los resultados, en lugar de los procesos de enseñanza y aprendizaje. Deben hacer uso de verbos de desempeño que requieran que los estudiantes demuestren conocimientos y habilidades, y que cada estándar sea mensurable, observable o verificable de alguna manera.

Para ser efectivos para la enseñanza y el aprendizaje, los estándares deben ser observables y mensurable. Lo que el estándar pretende que un estudiante entienda o pueda hacer debe ser claro. En consecuencia, los profesores deben poder determinar claramente si se ha cumplido con la expectativa de saber si los estudiantes necesitan más ayuda con estos conceptos.

Sin embargo, no es necesario que los estándares se escriban de manera tal que puedan probarse en una evaluación sumativa a gran escala. Simplemente deben ser observables en cierta medida, incluso por un profesor de aula. La selección cuidadosa de los verbos utilizados en cada estándar, junto con la especificidad del contenido, ayudará a asegurar que el estándar sea observable y mensurable (ver la Tabla 7.3).

Tabla 7.3: Ejemplos de verbos que ayudan a medir

VERBOS QUE SE REFIEREN A DESEMPEÑO OBSERVABLE O RESULTADOS	VERBOS QUE SE REFIEREN A ACTIVIDADES DE APRENDIZAJE	VERBOS QUE SE REFIEREN A PROCESOS COGNITIVOS
Crear	Examinar	Saber
Desarrollar	Explorar	Entender
Probar	Observar	Apreciar
Refinar	Descrubir	
Comunicar		

Recomendación 8: Integración de Practicas y Conceptos. Recomendamos que los estándares integren las prácticas de las ciencias de la computación con las declaraciones de concepto.

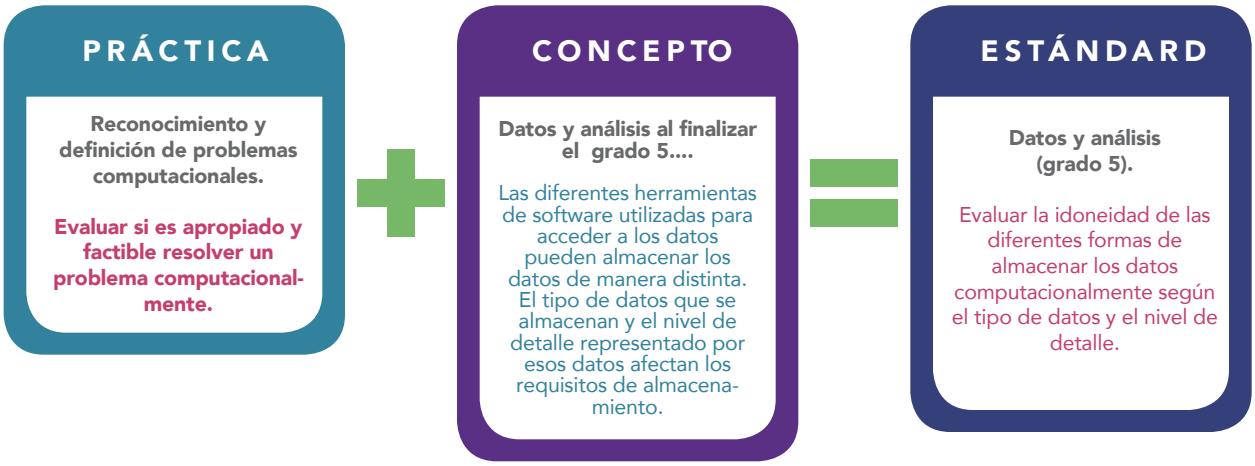
Para realizar la visión descrita en este marco y asegurar que todos los estudiantes puedan convertirse en usuarios competentes del conocimiento y la práctica de las ciencias de la computación; las prácticas y los conceptos deben integrarse en los estándares, así como en el plan de estudios y la instrucción.

Los conjuntos anteriores de estándares de educación en muchas disciplinas diferentes incluían estándares de práctica y contenido separados. Sin embargo, debido a que los profesores y los diseñadores de currículos estaban más familiarizados y cómodos con los estándares de contenido, los estándares de práctica rara vez se implementaban. Estaban separados, por lo que por lo general quedaron excluidos o "cubiertos" en la primera semana de clases y luego se olvidaron, o se usaron de manera irregular. Una forma eficiente de ayudar a asegurar que las prácticas se incluyan a lo largo de la instrucción es integrarlos completamente con los estándares de contenido.

Más importante aún, parte de la visión de la educación en ciencias de la computación es que los estudiantes serán competentes en el uso y la aplicación del conocimiento, no solo en la memorización. Si la aplicación y la comprensión profunda son las metas, los estándares de educación deben escribirse para reflejar esa meta. Al combinar una práctica con cada declaración de concepto para crear un estándar, los estándares resultantes describen más cercana el comportamiento, las habilidades y el conocimiento profundo que queremos que los estudiantes tengan.

La Figura 7.9 a continuación muestra un ejemplo de cómo integrar una práctica de ciencias de la computación con una declaración de concepto del marco.

Figura 7.9: Ejemplo de integración de una práctica y concepto para crear un estándar.



Los siguientes pasos fueron tomados para crear este ejemplo.

1. El escritor eligió una declaración de práctica específica dentro de la Práctica 3: Reconocimiento y definición de problemas computacionales.
2. El escritor seleccionó una parte de la declaración del concepto de Datos y Análisis como contexto para aplicar la práctica.
3. La práctica y el concepto se combinaron para crear una expectativa de rendimiento observable que exige la aplicación de la práctica en el contexto del concepto. La fuente del verbo en negrita en la declaración de práctica ayudó a enfocar la acción en el estándar.

La Figura 7.10 proporciona otro ejemplo de integración de una práctica y un concepto para crear un estándar. Al utilizar la lista de verificación proporcionada en la Recomendación 1: Rigor, vemos que esta estándar requiere un nivel apropiado de comprensión del contenido, como se refleja en la parte del concepto [resaltada en azul] y el compromiso con una práctica [resaltada en magenta], que facilita la aplicación del contenido [el estándar en su conjunto].

Figura 7.10: Segundo ejemplo de integración de una práctica y un concepto para crear un estándar.

No se espera, ni se recomienda, que cada declaración de concepto se combine con declaraciones de todas las prácticas para formar múltiples estándares. Por ejemplo, aunque hay un total de 68 declaraciones de conceptos y siete prácticas (cada una de las cuales tiene múltiples declaraciones), un conjunto estándar K–12 no debe esperar tener 476 estándares (es decir, 68 multiplicados por 7). Solo las declaraciones de práctica que son más relevante a una declaración de concepto deben ser considerado. Además, recuerda que las prácticas integradoras con declaraciones de conceptos a menudo introducen más rigor a la expectativa de desempeño del estudiante que se vería en la declaración de concepto por sí sola porque los estudiantes ahora tendrán que hacer algo con ese conocimiento conceptual. Se debe tener cuidado para garantizar que la combinación particular de las prácticas y los conceptos no introducen un nivel de rigor más alto que el apropiado para la banda de grado.
La Figura 7.11 proporciona un ejercicio para desarrolladores de estándares que usan estas consideraciones.

Figura 7.11: Ejercicio en la creación de estándares.

1. Como grupo, elija el mismo concepto y práctica y cree un estándar a partir del emparejamiento.
2. Comparar el estándar propuesto de cada uno.
3. Preguntar:
   a. ¿Es el rigor apropiado para la banda de grado?
   b. ¿Está clara la expectativa de rendimiento?
   c. ¿Refleja con precisión los componentes del concepto y la práctica?
   d. ¿Es este un estándar apropiado para todos los estudiantes o solo para aquellos que realizaran estudios extendidos?

Recomendación 9: Conexiones a otras disciplinas. Recomendamos que los estándares de las ciencias de la computación se escriban para alinearse con otros estándares académicos, como matemáticas y ciencias.

Hay muchas formas posibles de que las ciencias de la computación se conecten con otras asignaturas, como matemáticas, ciencias e ingeniería, así como humanidades, así como idiomas, estudios sociales, arte y música. Hacer conexiones intencionales entre los estándares de las ciencias de la computación y los estándares académicos en otras disciplinas ayudará a los profesores a entender cómo las ciencias de la computación pueden conectarse con su implementación de estándares en otras materias y promover experiencias de educación más coherentes para los estudiantes. Aunque relacionados, la tecnología/alfabetización digital y las ciencias de la computación son temas distintos.

Con un tiempo limitado en el aula, la educación de los estudiantes debe ser lo más coherente posible. Cuando el contenido en diferentes disciplinas está relacionado o conectado, es importante señalar esas conexiones con los educadores y facilitarlos a través de estándares. Cuando las alineaciones potenciales no se reconocen en los estándares, puede requerirse tiempo de instrucción adicional para cubrir todo. Por ejemplo, si se requiere un concepto matemático básico para los estándares de ciencias de la computación de tercer grado, pero no se incluye en los estándares de matemáticas hasta el quinto grado, los profesores de tercer grado deberán agregar ese concepto a su plan de estudios de las ciencias de la computación, o podrían terminar ignorando el contenido de las ciencias de la computación debido a la impresión de que es demasiada abrumadora.

Además de alinear las expectativas de nivel de grado, también puede ser útil incluir ejemplos clarificadores que se alineen y se conecten con los estándares de matemáticas, ciencias e ingeniería (consulte la Figura 7.12).

Figura 7.12: Ejemplo de un estándar de las ciencias de la computación que se conecta con un estándar de ciencia

Estándar: Pruebe y refine un programa utilizando una amplia gama de entradas hasta que se cumplan los criterios y las restricciones.

Este estándar se conecta con el diseño de ingeniería MS-ETS1-2 de Next Generation Science Standard: evaluar las soluciones de diseño de la competencia mediante un proceso sistemático para determinar qué tan bien cumplen los criterios y las limitaciones del problema.

También se debe prestar atención a las conexiones con materias fuera de la ciencia, tecnología, ingeniería y matemáticas, como los estándares de alfabetización en artes del lenguaje para materias técnicas. Dado que las ciencias de la computación no son actualmente requerida o evaluada en la mayoría de los estados, será muy útil ilustrar cómo los estándares se conectan y ayudan a cumplir los estándares existentes en otras materias. Estas conexiones se pueden hacer a través de materiales auxiliares, como cruces y ejemplos, y se pueden utilizar como una herramienta para integrar el contenido de otras materias en ciencias de la computación o incrustar el contenido de las ciencias de la computación en otras materias. Esto es particularmente cierto para los grados K-8, ya que las restricciones presupuestarias pueden no permitir profesores de las ciencias de la computación separados en las escuelas primarias y secundarias.

En 2010, Running on Empty, de la Association for Computing Machinery, informó que "existe una confusión profunda y generalizada en los estados sobre qué debe constituir y cómo diferenciar la educación, la alfabetización y la fluidez de la tecnología; educación en tecnología de la información; y las ciencias de la computación como materia académica (p. 9)." Si bien es plausible combinar los estándares de alfabetización digital/tecnológica con los estándares académicos de las ciencias de la computación, se debe tener cuidado para no confundir el tratamiento de uno con el de otro. Por ejemplo, si bien una presentación digital se puede utilizar para comunicar el proceso de desarrollo de software de un equipo, la creación de la presentación digital o el uso general del software de productividad de oficina no es una actividad de las ciencias de la computación. Nuevamente, Running on Empty informó que "en consonancia con los esfuerzos para mejorar el 'conocimiento de la tecnología', los estados se centran casi exclusivamente en aspectos de computación basados en habilidades (como el uso de una computadora en otras actividades de aprendizaje) y tienen pocos estándares sobre los aspectos conceptuales de las ciencias de la computación que proporcionan las bases para la innovación y el estudio más profundo en el campo (por ejemplo, desarrollar un entendimiento de un algoritmo) "(p. 7). Si combina la alfabetización digital y las ciencias de la computación en un conjunto de estándares, es importante que la distinción se mantenga clara a través de hilos identificables por separado.

Referencias

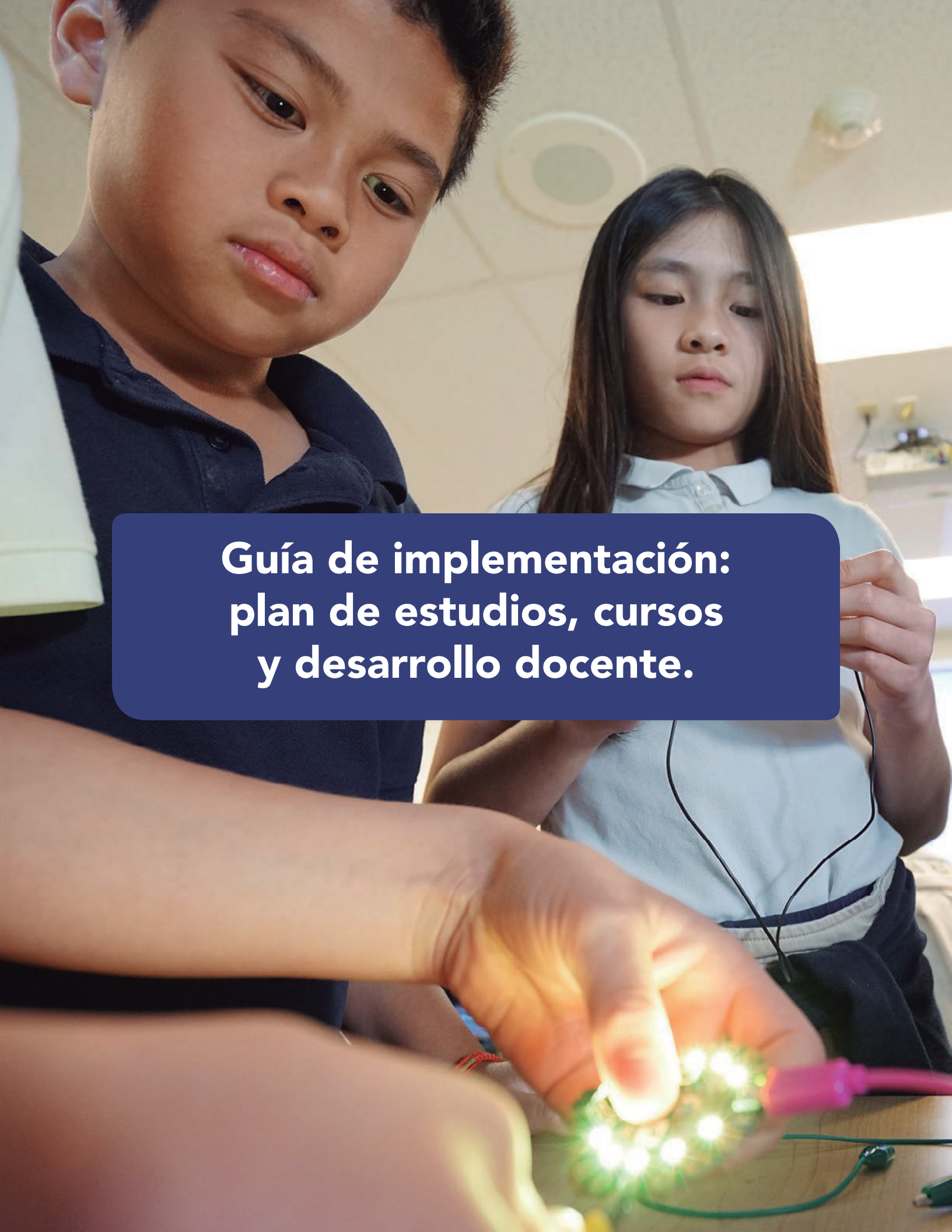
Achieve. (2010). *International science benchmarking report.* http://www.achieve.org/files/InternationalScienceBenchmarkingReport.pdf

Association for Computing Machinery & Computer Science Teachers Association. (2010). *Running on empty: The failure to teach K–12 computer science in the digital age*. Retrieved from http://runningonempty.acm.org/fullreport2.pdf

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 1–10).

Marzano, R. J. (2004). *Building background knowledge for academic achievement: Research on what works in schools.* Alexandria, VA: Association for Supervision and Curriculum Development.

National Research Council. (2012). *A Framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.



Guía de implementación: plan de estudios, cursos y desarrollo docente.

8

Guía de Implementación: Plan de Estudios, Cursos y Desarrollo Docente.

Un aumento en el interés en todo el país ha llevado a muchas escuelas, distritos y estados a comenzar a descubrir cómo aumentar las oportunidades de los estudiantes para aprender las ciencias de la computación. Están abordando preguntas difíciles, como: ¿Cómo se acumula el aprendizaje con el tiempo? ¿Qué infraestructura técnica se requiere para ofrecer las ciencias de la computación? ¿Cómo encaja las ciencias de la computación en el horario escolar? ¿Cómo se prepararán suficientes profesores? Aunque los esfuerzos de base para integrar las ciencias de la computación a nivel de aula han existido durante décadas,[1] los esfuerzos actuales buscan también abordar las ciencias de la computación a nivel estatal y distrital, a gran escala, y para todos los estudiantes. En cualquier nivel de grado, la implementación de las ciencias de la computación está acompañada por las oportunidades únicas y los desafíos de agregar una nueva disciplina al alcance de la educación K–12.

La reforma educativa en computación a gran escala sigue siendo una nueva frontera, y los estados y distritos están probando una variedad de enfoques. A partir de septiembre de 2016, los gobernadores de seis estados se comprometieron a trabajar para alcanzar tres objetivos de política: ofrecer al menos un curso de las ciencias de la computación en todas sus escuelas secundarias, financiar oportunidades de desarrollo profesional para crear capacidad docente y crear estándares integrales de las ciencias de la computación K-12 (Gobernadores de las ciencias de la computación, 2016). Los distritos escolares grandes en ciudades como la ciudad de Nueva York, San Francisco y Oakland han lanzado iniciativas de varios años que llevarán a todas las escuelas a ofrecer las ciencias de la computación, y algunas también tienen como objetivo brindar instrucción a cada estudiante. Las Escuelas Públicas de Chicago han ido un paso más allá al convertirse en el distrito más grande de la nación en crear un requisito de graduación de las ciencias de la computación para todos los estudiantes (Escuelas Públicas de Chicago, 2016).

1 El uso del lenguaje de programación del Logo en las escuelas K–12 alcanzó su punto máximo a principios y mediados de los años ochenta. Los objetivos del Logo fueron más allá de la introducción de los fundamentos de computación. El Logo incluyó la oportunidad de crear con tecnología y, en el proceso, desarrollar habilidades de pensamientos computacionales.

Este capítulo sugiere temas para que los comités estatales y los departamentos de educación consideren a medida que desarrollan y adoptan políticas para apoyar la expansión de las ciencias de la computación K-12, y proporciona orientación a los administradores de distrito, escuela y profesores, y educadores informales que están planeando implementar las ciencias de la computación. La educación K–12 es un sistema complejo compuesto por muchas partes interactivas que deben coordinarse para lograr un propósito común: las ciencias de la computación para todos los estudiantes. Este capítulo aborda específicamente el plan de estudios, la evaluación, los métodos de los cursos, la infraestructura técnica, la participación de las partes interesadas (Stakeholders), los programas en curso, la certificación y el desarrollo profesional. La visión, los conceptos y las prácticas del marco desempeñan un papel clave en cada una de estas piezas. El desarrollo de estándares también desempeña un papel vital en la implementación de la educación en ciencias de la computación y se analiza en el capítulo "**Orientación para desarrolladores de estándares**".

Los estados, distritos y escuelas deben iniciar un proceso intencional y reflexivo que mantenga la revisión y reevaluación en etapas clave con el objetivo de mejorar la ejecución y los resultados.

Los temas que se discuten en este capítulo tienen la intención de generar más conversaciones que conduzcan a una infraestructura sostenible para la educación en las ciencias de la computación (consulte la Figura 8.1 para conocer las políticas estatales recomendadas). Debido a la creciente demanda de las ciencias de la computación y la preocupación genuina de los padres, los educadores y la industria de que los estudiantes se están quedando atrás, los esfuerzos de reforma pueden centrarse fácilmente en el acceso y la escala a costa de la calidad y la sostenibilidad. Estos objetivos no son mutuamente excluyentes. Los estados, distritos y escuelas deben iniciar un proceso intencional y reflexivo que mantenga la revisión y reevaluación en etapas clave con el objetivo de mejorar la ejecución y los resultados.

En un momento en que solo unos pocos estados han comenzado a iniciar esfuerzos a nivel estatal para implementar la educación en ciencias de la computación, no hay modelos claros que demuestren el éxito a largo plazo.

Figura 8.1: Políticas recomendadas que promueven y apoyan la educación de las ciencias de la computación

Las siguientes recomendaciones se han extraído de Making Computer Science Fundamental to K–12 Education: Eight Policy Ideas (Code.org, 2015).

Definir las ciencias de la computación y establecer K–12 estándares de las ciencias de la computación: Los estándares proporcionan una base para alinear todas las demás políticas bajo una visión coherente de las ciencias de la computación para todos los estudiantes.

Asignar fondos para el desarrollo profesional riguroso y el apoyo del curso: Los estados y distritos pueden dedicar fondos para desarrollar la capacidad de enseñar las ciencias de la computación, incluidos los materiales del curso y la infraestructura técnica.

Implementar vías claras de certificación: Además del desarrollo de vías de certificación tradicionales, incentivos y vías aceleradas, las vías alternativas ayudarán a abordar la necesidad a corto y largo plazo de los profesores de las ciencias de la computación.

Crear incentivos en las instituciones de educación superior para ofrecer las ciencias de la computación a profesores en pre-servicio: Los estados pueden crear subvenciones competitivas para que las escuelas de educación aumenten la cantidad de profesores en formación que pueden enseñar las ciencias de la computación e incentivar las oportunidades de asociación entre los distritos escolares locales y las escuelas de educación para crear vías directas para que los profesores ingresen a los distritos escolares con mayor necesidades.

Establecer posiciones dedicadas a las ciencias de la computación en las autoridades de educación estatales y locales: Cualquier esfuerzo sostenido para escalar la educación en ciencias de la computación requerirá posiciones de liderazgo a nivel estatal y distrital.

Requerir que todas las escuelas secundarias ofrezcan las ciencias de la computación: Si bien la educación en ciencias de la computación desde el jardín de infantes hasta el grado 12 es la visión a largo plazo, los estados y los distritos pueden actuar a corto plazo al exigir que todas las escuelas secundarias ofrezcan al menos un curso de las ciencias de la computación.

Permita que las ciencias de la computación cuenten como un requisito core de graduación: los estados que permiten que las ciencias de la computación cuenten como un requisito de graduación, en lugar de una clase optativa, ven aumentos en la cantidad de estudiantes que toman cursos avanzados de informática y aumentos en la participación de minorías poco representadas. *

Permita que las ciencias de la computación cuenten como un requisito de admisión para las instituciones de educación superior: las políticas que no permiten que las ciencias de la computación satisfagan un requisito de admisión pueden reducir el incentivo para que los estudiantes tomen las ciencias de la computación, incluso cuando cumple con el requisito de graduación de la escuela secundaria.

* Revisión de los datos de Advanced Placement® (AP) de 2012 por estado para AP Computer Science y AP Calculus proporcionados por el College Board.

Las conversaciones iniciadas en este capítulo deben estar revisadas con frecuencia e informadas por las lecciones aprendidas en cada estado y en todo el país. En lugar de pensar en la sostenibilidad como el establecimiento de reformas que "duran y permanecen iguales", los formuladores de políticas y los responsables de tomar decisiones deben pensar en la sostenibilidad como el establecimiento de reformas que "duran y cambian" (Century, 2009, párr. 7). Debido a la naturaleza evolutiva de la educación en ciencias de la computación, los planes de implementación deben ser flexibles y adaptables, mientras que se atiende a la demanda del público de experiencias de las ciencias de la computación de alta calidad para todos los estudiantes.

Incorporación de las ciencias de la computación en los sistemas K–12

Agregar ciencias de la computación a la educación K–12 requiere más que simplemente agregar o revisar contenido. Un enfoque reflexivo para incluir las ciencias de la computación en la instrucción K–12 también abarcará cursos y modelos de instrucción; infraestructura técnica; y participación de importantes grupos de partes interesadas (Stakeholders), como administradores, padres, profesores y personal de apoyo. Es importante considerar la implementación para todos los estudiantes, independientemente de su raza, género, discapacidad, estado socioeconómico o dominio del idioma inglés, ya que las desigualdades pueden propagarse al implementar reformas a gran escala. El reclutamiento, la expansión y la equidad deben monitorearse activamente durante todo el proceso de implementación (Margolis, Goode y Chapman, 2015).

Barra lateral 23: El reclutamiento, la expansión y la equidad deben ser monitoreados activamente durante todo el proceso de implementación (Margolis, Goode, & Chapman, 2015).

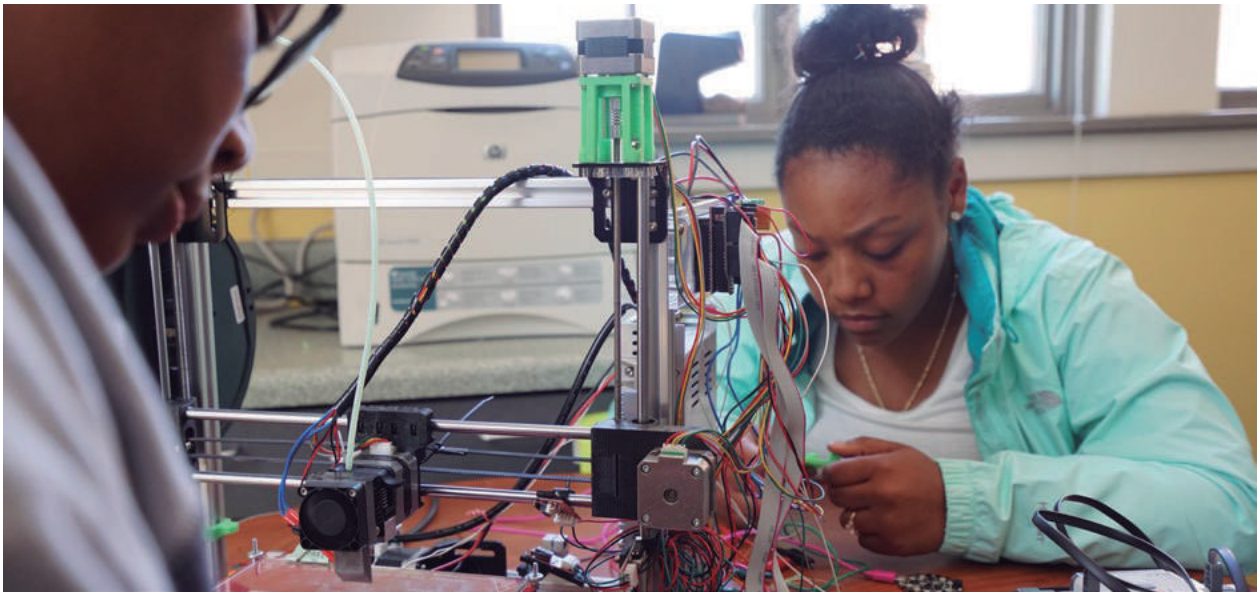
Currículo

A medida que aumentan las oportunidades en ciencias de la computación en todo el país, tanto dentro como fuera de la escuela, los estudiantes ingresarán a las aulas con una amplia gama de experiencias en ciencias de la computación. Algunos estudiantes habrán tenido una gran exposición temprana, mientras que otros estudiantes no habrán tenido ninguna exposición temprana. Para satisfacer las necesidades de todos los estudiantes, los profesores deben estar preparados para enseñar a esta amplia gama de estudiantes de manera equitativa. Esta sección describe algunas de las muchas consideraciones que los educadores deberán tener en cuenta al seleccionar y/o desarrollar experiencias curriculares significativas para todos los estudiantes. Aunque el marco puede guiar las consideraciones del currículo a un alto nivel, las expectativas de rendimiento (es decir, los estándares) que integran los conceptos y las prácticas del marco deberían guiar en última instancia lo que sucede en el aula. Los desarrolladores formales de currículo y los proveedores de contenido pueden apoyar a los profesores del aula, especialmente a aquellos que son nuevos en ciencias de la computación, al desarrollar materiales curriculares de alta calidad alineados con una visión coherente de K–12.

Usos y limitaciones para el desarrollo curricular

Cuando se elabora un currículo completo de las ciencias de la computación K–12 basado en el marco, es importante comprender la amplitud de los enfoques que promueve el marco y sus limitaciones.

El marco proporciona amplias expectativas para las ciencias de la computación K–12 que están diseñadas para incluir diversos enfoques curriculares. Las escuelas, los distritos y los estados deben considerar el marco como un documento que proporciona claridad de contenido (el "qué") pero no dicta la implementación (el "cómo"). Por ejemplo, los diferentes planes de estudio de la escuela primaria que enseñan a los jóvenes a programar, uno con robots físicos y otro en un entorno virtual en línea, pueden inculcar un conocimiento de algoritmos (K–2.Algorithms y Programming.Algorithms) y ayudar a los estudiantes comprender cómo responde la computadora a los comandos (K–2.Algorithms y Programming.Control).



El concepto y las declaraciones de práctica son grandes ideas que pueden usarse para informar lecciones, una serie de lecciones o unidades de currículo. Las organizaciones que usan el modelo de diseño curricular Entendimiento por diseño pueden usar el marco para informar los "entendimientos duraderos" y las "preguntas esenciales" para lecciones y unidades (Wiggins y McTighe, 2005). Los entendimientos duraderos, similares a los enunciados conceptuales del marco, iluminan las ideas principales y recurrentes que dan importancia y significado a los elementos del currículo individual, tales como hechos y habilidades, y las preguntas esenciales reflejan las preguntas clave que centran el currículo en torno a una comprensión más profunda.

Además, el marco presenta las grandes ideas que los estudiantes deben entender al final de una banda de grados, pero no abarca todo el aprendizaje que puede tener lugar dentro de esa banda de grado. Aunque el marco se centra en una base esencial en ciencias de la computación, se alienta a las escuelas y los profesores a crear material de apoyo adicional y extender las experiencias de instrucción más allá de estas competencias básicas, al tiempo que se presta especial atención a la adecuación del desarrollo. Se debe alentar a las escuelas y aulas que ya superan las expectativas en el marco a seguir haciéndolo, mientras que otras pueden usar el marco como una aspiración y un punto de partida.

Figura 8.2: Conceptos y prácticas del marco de las ciencias de la computación K–12

Conceptos Core	Prácticas Core
1. Sistemas de computación	1. Fomentar una cultura computacional inclusiva
2. Las redes e internet	2. Colaboración en torno a la computación
3. Datos y análisis	3. Reconociendo y definiendo problemas computacionales
4. Algoritmos y Programación	4. Desarrollar y usar abstracciones
5. Impactos de la computación	5. Creando artefactos computacionales
	6. Probar y refinar artefactos computacionales
	7. Comunicar sobre de la computación

Consideraciones para la alineación curricular

Al seleccionar o desarrollar un currículo, materiales de instrucción y herramientas computacionales, tales como entornos de programación, que se alineen con el marco, es importante reflexionar sobre las siguientes consideraciones pedagógicas.

Primero, los planes de estudio de las ciencias de la computación deben proporcionar una visión integral de la ciencia informada por los cinco conceptos básicos y las siete prácticas centrales de las ciencias de la computación, tal como se describe en el marco (ver Figura 8.2). Desafortunadamente, las ciencias de la computación y la programación (o codificación) a menudo se consideran sinónimos en la educación K–12. Esta creencia conduce a cursos que se centran solo en la programación y dejan de lado otras áreas de las ciencias de la computación que influyen en nuestro mundo, como Internet, datos y perspectivas culturales y sociales sobre las ciencias de la computación.

Segundo, los currículos alineados con el marco deben ser apropiados para el desarrollo según las progresiones de la banda de grado en el marco. Es importante determinar hasta qué punto el contenido se alinea con los conceptos y prácticas en el marco, incluida la ubicación en la banda de grado. Además, debido a que los conceptos y las prácticas están enmarcados en las progresiones de aprendizaje de K-12, es importante que los planes de estudios de nivel de grado se ajusten a una experiencia coherente de K-12. El marco es compatible con las vías de los cursos que se ofrecen en cada grado o solo en grados particulares dentro de las bandas de grado; Los estados y distritos decidirán en función de su contexto local.

El plan de estudios debe integrar los conceptos y las prácticas en experiencias significativas para los estudiantes, en lugar de centrarse únicamente en los conceptos

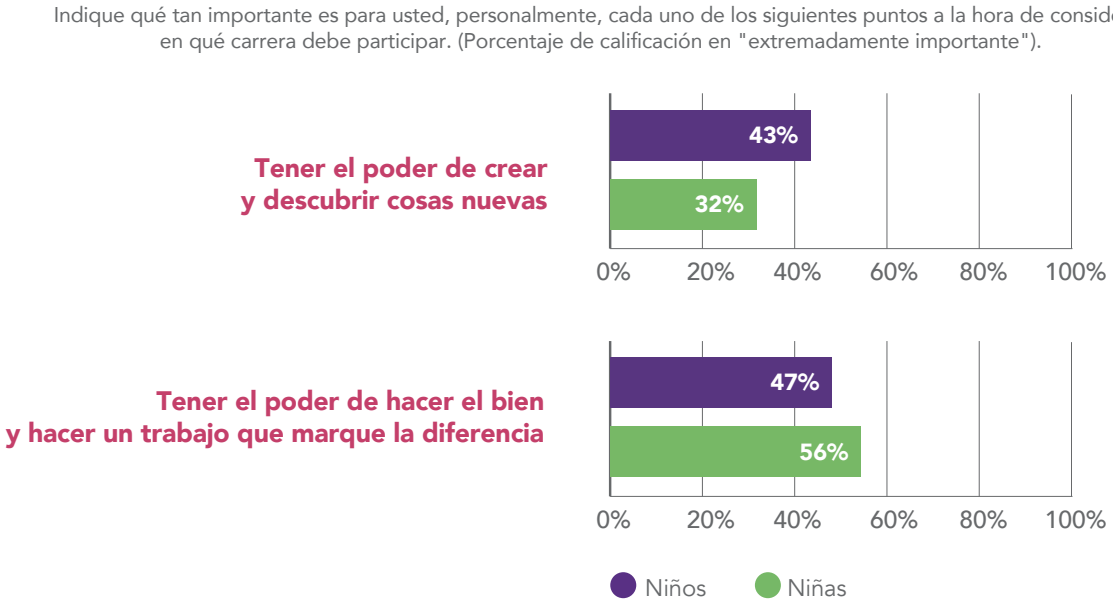
Tercero, los currículos deben integrar los conceptos y las prácticas en experiencias significativas para los estudiantes, en lugar de centrarse únicamente en los conceptos. Las lecciones y actividades deben proporcionar no solo oportunidades de contenido rico para que los estudiantes aprendan sobre conceptos de las ciencias de la computación, sino también oportunidades significativas para que los estudiantes se involucren en prácticas de las ciencias de la computación. La instrucción diaria y las actividades de los estudiantes deben integrar las prácticas de las ciencias de la computación entre sí y con los conceptos. Además, algunos conceptos pueden abordarse fácilmente en la misma lección o actividad; el material descriptivo de cada declaración de concepto proporciona una guía sobre qué ideas se conectan entre sí.

Finalmente, es importante considerar si el aprendizaje de un entorno de programación es el enfoque principal de un currículo o si el uso de las herramientas de programación se basa en los conceptos y las prácticas. Un enfoque de "herramienta primero" puede no proporcionar todas las experiencias necesarias para que un estudiante se involucre completamente en los conceptos y prácticas del marco. En su lugar, el currículo debe diseñarse desde una perspectiva de "primero el contenido", en el que las herramientas de programación, el equipo (por ejemplo, los robots) e incluso los idiomas, son un vehículo para aprender los conceptos y las prácticas, en lugar de convertirse en el foco en sí mismos.

Socialmente relevante y culturalmente situado

El currículo debe incluir proyectos que ofrezcan oportunidades para crear tecnologías innovadoras en contextos socialmente relevantes y culturalmente situados. En una encuesta sobre las preferencias de carrera de los estudiantes de 13 a 17 años con destino a la universidad (consulte la Figura 8.3), los estudiantes que valoran tener el poder de crear y descubrir cosas nuevas y trabajar en un campo de vanguardia también muestran un gran interés en las ciencias de la computación. (Fundación Educativa WGBH y ACM, 2009). Los conceptos y las prácticas del marco brindan oportunidades para que los estudiantes se involucren en este tipo de actividades, como crear una variedad de artefactos computacionales en múltiples áreas de las ciencias de la computación.

Figura 8.3: Características de las carreras que los estudiantes consideran importantes.



Fuente: WGBH Educational Foundation y la Association for Computing Machinery, 2009.

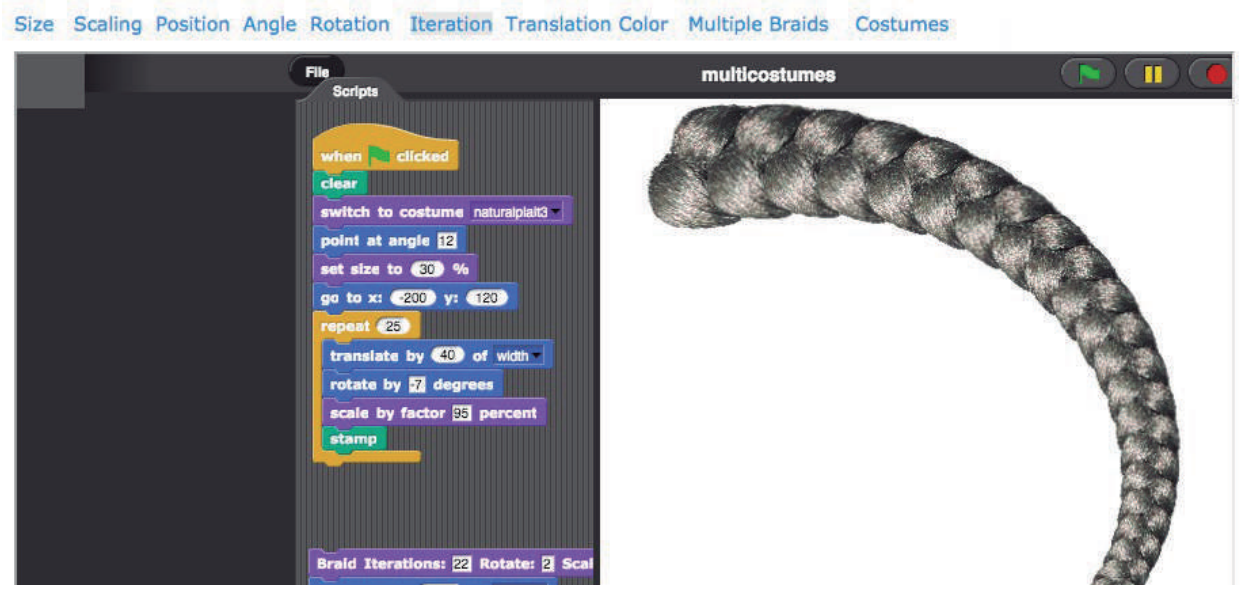
La encuesta también encontró que los estudiantes que tienen menos probabilidades de mostrar interés en las ciencias de la computación son aquellos a quienes les gusta trabajar con personas de una manera interconectada, social e innovadora, y aquellos que consideran importante marcar la diferencia en la vida de otras personas para una carrera (Fundación Educativa WGBH y ACM, 2009). Además, las mujeres estudiantes tenían más probabilidades que los hombres de calificar una diferencia como "extremadamente importante" en una carrera. Estos resultados pueden referirse más a los estereotipos que los estudiantes tienen de las ciencias de la computación como incompatibles con estos deseos, en lugar de la realidad de cómo se practica las ciencias de la computación en las carreras y la influencia que puede tener en otros. Cualquiera que sea la razón de estas diferencias, el currículo juega un papel importante para abordar las percepciones erróneas acerca de las carreras en ciencias de la computación, y los conceptos y prácticas del marco pueden guiar experiencias que llegan a todos los estudiantes. Por ejemplo, el concepto core Impactos de la Computación resalta la influencia que tiene la computación en las interacciones sociales y culturales de las personas a nivel comunitario y social, y aborda los deseos de los estudiantes de participar en campos que marcan una diferencia en la vida de los demás. La práctica del marco de crear artefactos computacionales enfatiza la creación de un "artefacto computacional. . . para abordar un problema de la sociedad "(P5. Crear artefactos computacionales.2). Prácticas adicionales, como el fomento de una cultura de computación inclusiva, la colaboración en torno a la computación y la comunicación sobre la computación,

Los estudiantes que valoran tener el poder de crear y descubrir cosas nuevas y trabajar en un campo de vanguardia también muestran un gran interés en las ciencias de la computación.

enfatizar la naturaleza social de las ciencias de la computación. Las ciencias de la computación es una disciplina de transformación basada en proyectos, y los estudiantes estarán más comprometidos con proyectos que se centran en problemas del mundo real y de la comunidad para el bien social (Goldweber et al., 2013).

Los proyectos deben reconocer y desarrollar los ricos antecedentes culturales, o "fondos de conocimiento", que los estudiantes traen al aula. Los fondos para el conocimiento se refieren a los "cuerpos de conocimiento y habilidades acumulados históricamente y desarrollados culturalmente, esenciales para el funcionamiento y bienestar del hogar o del individuo" (Moll, Amanti, Neff y González, 1992, p. 133). Al aprender sobre los estudiantes y sus familias, los profesores pueden desarrollar la perspectiva de que "los hogares de sus estudiantes contienen recursos culturales y cognitivos ricos y que estos recursos pueden y deben usarse en su sala de clases para brindar lecciones culturalmente receptivas y significativas que atraigan a los estudiantes 'conocimiento previo "(López, sf, párr. 3). El currículo Explorando las Ciencias de la Computación, también analizado en el capítulo Equidad en Educación en Ciencias de la Computación, utiliza herramientas de diseño situadas culturalmente (Eglash, 2003) para "alentar a los estudiantes a expresar artísticamente conceptos de diseño computacional de la historia latina/ afroamericana, afroamericana y nativa americana como así como actividades culturales en danza, patinaje, grafiti, y más "(Margolis et al., 2012, p. 76). A través de lecciones situadas culturalmente (ver la Figura 8.4 para un ejemplo), los estudiantes construyen relaciones personales con conceptos y prácticas de las ciencias de la computación y, en última instancia, sienten que las ciencias de la computación son más relevantes para sus vidas. El uso de contextos computacionales situados culturalmente tiene beneficios adicionales, como la posibilidad de contrarrestar las barreras para aumentar la participación de las minorías en las ciencias de la computación. Ver cómo los aspectos de la cultura de un estudiante se basan en conceptos y prácticas computacionales puede reducir el conflicto de identidad, en el que los estudiantes sienten que su identidad personal o cultural es incompatible con la participación o el éxito académico en una materia, por ejemplo, "mi gente no hace la computación." (Eglash, Bennett, O'Donnell, Jennings, & Cintorino, 2006).

Figura 8.4: Ejemplo de una actividad computacional situada culturalmente.



Los contextos socialmente relevantes y culturalmente situados ofrecen oportunidades para integrar las ciencias de la computación con otras materias. Los profesores deben buscar en las comunidades de sus alumnos ejemplos de proyectos y aplicaciones de educación en las ciencias de la computación que puedan alinearse con el marco. Sin embargo, estos proyectos deben ser cuidadosamente elaborados y organizados para principiantes. La investigación en cursos introductorios de pregrado sugiere que desarrollar proyectos auténticos enfocados en la ayuda humanitaria que motiven a los principiantes es difícil, ya que los problemas humanitarios son complejos por su propia naturaleza (Rader, Hakkarinen, Moskal, & Hellman, 2011). Por otro lado, algunos investigadores han notado que los novatos tienden a favorecer las tareas que perciben como fáciles y divertidas y que evitan los problemas que son demasiado abiertos (Cliburn y Miller, 2008). Por lo tanto, teniendo en cuenta las experiencias de los estudiantes con las ciencias de la computación y sus niveles de habilidad, los currículos socialmente relevantes y culturalmente situados son prometedores para involucrar a todos los estudiantes.

Evaluación

Las evaluaciones se utilizan de múltiples maneras en la educación K–12. Esta sección se centra en la evaluación a nivel del aula, en lugar de las pruebas de alto nivel que pueden dominar la conversación sobre la reforma en otras áreas temáticas. Generalmente, la evaluación en el aula puede ser formativa o sumativa. La evaluación formativa se utiliza durante las actividades en el aula para modificar la instrucción o proporcionar a los estudiantes información inmediata sobre su aprendizaje o progreso, mientras que la evaluación sumativa se utiliza para evaluar o medir el aprendizaje de los estudiantes al final de un período de instrucción (Black, Harrison, Lee, Marshall, & William, 2003). Actualmente, las evaluaciones sumativas y los exámenes programáticos de final de curso son poco frecuentes en ciencias de la computación fuera de Advanced Placement® (AP), Bachillerato Internacional o exámenes de certificación reconocidos por la industria en programas de Carreras y Educación Técnica (CET), en inglés Career and Technical Education (CTE). Hay varias facetas importantes que deben tenerse en cuenta en las evaluaciones de las ciencias de la computación en el aula: el uso de tareas auténticas, la amplitud de conceptos que se están evaluando y el papel especial que pueden desempeñar las computadoras en la entrega de la instrucción y la medición del rendimiento. Los métodos de evaluación basados en proyectos y en el portafolio son fundamentales para medir de manera auténtica el rendimiento en el aula de las ciencias de la computación. Las tareas de rendimiento suelen ser más flexibles que las evaluaciones tradicionales que buscan una solución o respuesta a una pregunta de evaluación. Estas tareas les permiten a los estudiantes demostrar su comprensión de múltiples maneras que resaltan su creatividad, intereses y comprensión. En consecuencia, estas evaluaciones pueden proporcionar a un educador una mejor comprensión del conocimiento y razonamiento de los estudiantes. Estas evaluaciones no tradicionales pueden ser útiles para medir el uso de algoritmos, el pensamiento computacional y la resolución de problemas por parte de los estudiantes, que generalmente son difíciles de medir con preguntas de opción múltiple. Las prácticas identificadas en el marco, como Comunicar acerca de la computación, Colaborar en torno a la computación y Crear artefactos computacionales, son claves para enfatizar en cualquier tarea de rendimiento de las ciencias de la computación. El curso Principios de ciencias de la computación de AP emplea tareas de desempeño y rúbricas de acompañamiento a las que se puede acceder libremente, que los profesores pueden encontrar útiles como punto de partida para diseñar sus propias evaluaciones.

La evaluación debe reflejar múltiples aspectos de las ciencias de la computación según lo definido por los cinco conceptos core y las siete prácticas del marco. La mayoría de las evaluaciones de las ciencias de la computación se centran principalmente

en programación (Yadav et al., 2015) e ignoran otros aspectos de las ciencias de la computación, como el análisis de datos o el impacto de las ciencias de la computación en la sociedad. Múltiples conceptos pueden ser abordados simultáneamente. Por ejemplo, los profesores pueden evaluar la capacidad de los alumnos para analizar las ventajas y desventajas de los diferentes algoritmos de criptografía, que abordan la idea del rendimiento algorítmico (algoritmos y programación), así como la ciberseguridad (Redes e Internet). Incluso cuando el enfoque es la programación, los estudiantes deben ser evaluados no solo por su capacidad para escribir el programa sino también por su capacidad para comunicar la importancia y el proceso de desarrollo del producto (Comunicar acerca de la computación), incluida la colaboración entre los miembros (Colaborar en torno a la computación). Por ejemplo, los estudiantes pueden enviar documentos de planificación utilizados para producir el programa, hacer una presentación sobre el impacto que su programa tendrá en un público objetivo y escribir una reflexión sobre cómo trabajó el equipo para armar el programa.

En comparación con otras asignaturas, las ciencias de la computación brindan oportunidades únicas para aprovechar el aprendizaje en línea y la evaluación computarizada, a la vez que mantiene una experiencia auténtica para demostrar el rendimiento. Las plataformas dedicadas a las ciencias de la computación les permiten a los estudiantes crear programas como juegos, aplicaciones y simulaciones dentro de un entorno que también recopila datos, analiza los logros y comunica el progreso tanto a los estudiantes como a los profesores. Estas plataformas tienen la capacidad de integrar naturalmente la instrucción, la práctica y la evaluación. Las plataformas de aprendizaje y evaluación en línea también tienen el potencial de llegar a los estudiantes en los distritos escolares rurales, que a menudo se encuentran en una situación de desventaja para encontrar profesores en áreas de gran necesidad, tales como las ciencias de la computación.

Cursos y métodos de instrucción

El marco describe una experiencia K–12 en ciencias de la computación que desarrolla sofisticación a lo largo del tiempo. Los cursos y las vías de instrucción que tratan la ciencia de la computación como una disciplina, en lugar de una electiva individual, se requieren para implementar esta visión. Los cursos y el currículo no existen en un vacío de instrucción y deben tener en cuenta los conceptos y las prácticas según lo establecido en las cuatro bandas de grado en el marco. Esta sección explora las opciones para construir un camino de instrucción K–12 en ciencias de la computación.

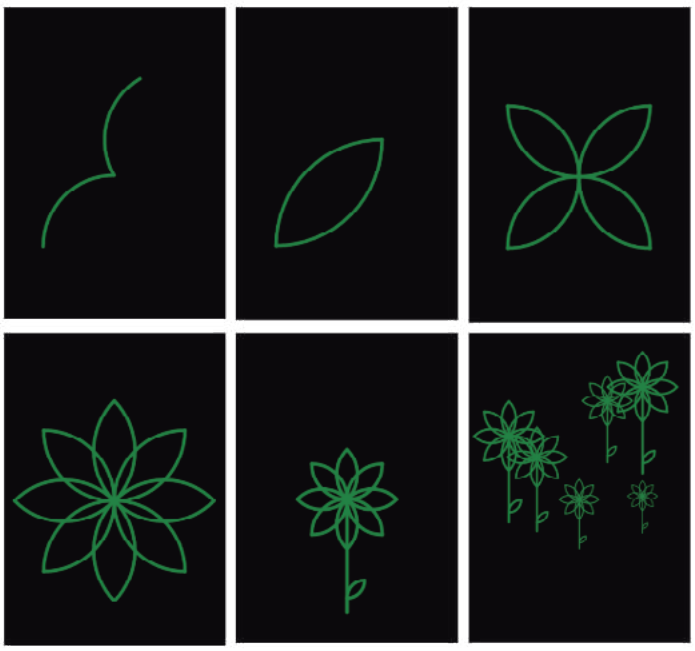
Cursos integrados de las ciencias de la computación

En lugar de agregarse aún más trabajo a los educadores, las ciencias de la computación pueden ayudar al movimiento actual hacia la educación interdisciplinaria. Las aulas pueden infundir las ciencias de la computación en prácticamente cualquier otra área temática, incluyendo matemáticas, ciencias, artes del lenguaje en inglés, idiomas mundiales, ciencias sociales, bellas artes, aprendizaje en servicio, educación física y de la salud, y programas CET. En todo el marco, se proporcionan oportunidades para la integración y la aplicación dentro de otras áreas de contenido. Por ejemplo, los conceptos dentro de los sistemas de computación, cuando se combinan con las prácticas del marco, podrían incorporarse a las bellas artes a medida que los estudiantes producen obras de arte que incluyen música digital, animación y sistemas de iluminación. La organización de redes en Redes e Internet se puede utilizar para reforzar ideas de redes en otras áreas de contenido, tales como las conexiones entre los personajes de una historia en artes del lenguaje en inglés o las formas en que se propagan las enfermedades contagiosas que se propagan a través de

poblaciones en clases de salud y educación física. La programación puede ayudar en la recopilación o representación de datos en aulas de matemáticas o ciencias, vinculando los conceptos core de Datos y Análisis y Algoritmos y Programación. Y las declaraciones del concepto Impactos de la Computación se pueden reforzar a medida que los estudiantes consideran los efectos de la computación en los idiomas mundiales, tal como el software de traducción, el aprendizaje a través del servicio del aprendizaje, tales como la tecnología puede permitir a las personas en diferentes lugares conectarse, comunicarse y colaborar, o ciencias sociales tales como la interacción de los medios sociales y los movimientos políticos o el uso de la tecnología para monitorear las comunicaciones. Los conceptos y las prácticas del marco de ciencias de la computación K–12 se pueden integrar en otras áreas de contenido de muchas otras formas también.

Existe una historia de abundantes ejemplos de experiencias computacionales integradas y apropiadas para la edad en el nivel K–8 (por ejemplo, Papert, 1980). En su libro clásico sobre niños y computación, Mindstorms, Seymour Papert describe una situación que refleja interacciones comunes entre dos niños que trabajan y juegan con el lenguaje de programación Logo. La experiencia comienza con un estudiante que desea dibujar una flor en la pantalla de la computadora. Una estudiante le pregunta a la otra si tiene algún programa preexistente que pueda usar para dibujar los pétalos. Modifican un programa que dibuja un arco varias veces hasta que se dan cuenta de cómo usar su comprensión de los ángulos para crear un pétalo completo y luego varios pétalos. Luego, los estudiantes proceden a crear un jardín de flores repitiendo el procedimiento para dibujar una sola flor y usando variables para aleatorizar el tamaño y la ubicación. Este ejemplo, que se muestra en la Figura 8.5, ilustra la integración de las matemáticas, la programación y el juego.

Figura 8.5: Un ejemplo del proceso iterativo que los estudiantes pueden usar para crear un jardín de flores



Algunos esfuerzos recientes han incorporado las ciencias de la computación en las aulas de matemáticas y ciencias. Bootstrap y Project GUTS (Growing Up Thinking Scientifically), dos programas patrocinados por la National Science Foundation, demuestran que los estudiantes pueden aprender conceptos y prácticas de ciencias de la computación dentro de los objetivos generales de sus clases de álgebra y ciencias. Bootstrap (2016) utiliza la programación de videojuegos como un enfoque para enseñar temas tales como el teorema de Pitágoras, la fórmula de distancia y las ecuaciones lineales. El proyecto GUTS (2016) incorpora proyectos de simulación y modelado en la Tierra, la vida y las ciencias físicas para explorar temas tales como la contaminación del agua, los ecosistemas y las reacciones químicas. Las experiencias dentro de estos programas son modulares y pueden abarcar desde un par de lecciones hasta varias semanas de contenido. Si bien los efectos en el enfoque de instrucción y el desempeño posterior en las evaluaciones siguen siendo explorados, existen muchos programas similares en todo el país, por ejemplo, Geometría Algorítmica, 2016; Centro C-STEM de UC Davis, 2016, lo que refuerza este enfoque como un modelo que se puede usar para exponer a los estudiantes a las ciencias de la computación utilizando recursos preexistentes.

Se debe tener en cuenta que la integración es una cuestión de implementación a nivel de escuela o distrito, no dentro de los estándares a nivel estatal. Las ciencias de la computación pueden integrarse en materias preexistentes, pero los estándares estatales para las ciencias de la computación, que aclaran las expectativas disciplinarias específicas, deben seguir siendo un conjunto discreto de estándares, en lugar de mezclarse con las expectativas de desempeño de otras asignaturas. Unos pocos estados han creado con éxito estándares computacionales como hilos discretos dentro de un conjunto más amplio de estándares. Por ejemplo, Indiana ha creado un capítulo de las ciencias de la computación dentro de sus estándares generales de ciencias (Departamento de Educación de Indiana, 2016), y Massachusetts ha combinado los Estándares de Alfabetización Digital y la Ciencias de la Computación con hilos discretos para cada uno de ellos, el Departamento de Primaria y Secundaria Educación de Massachusetts 2016). Los estándares de Massachusetts delinean las diferencias entre la alfabetización informática, la ciudadanía digital y las ciencias de la computación. Estos enfoques permiten que las escuelas y los distritos decidan cómo, currículo; cuándo el nivel de grado y/o dónde la materia se integra en ciencias de la computación. Además, esta integración a nivel de estándares no significa necesariamente que las ciencias de la computación estén integradas en otras asignaturas, ya que es igualmente viable que exista instrucción en ciencias de la computación tales como cursos independientes que integran contenido de otras disciplinas.

Cursos de las ciencias de la computación independientes:

Ofrecer ciencias de la computación como cursos únicos e independientes tiene el beneficio de no afectar el tiempo de instrucción en otras materias. En la escuela primaria, las ciencias de la computación pueden existir como una clase especial, similar a la música, el arte y la educación física, a través de la cual los estudiantes rotan durante su horario semanal. En la escuela intermedia, las ciencias de la computación puede ser una experiencia dedicada de un semestre o año en un nivel de grado particular o disponible en todos los niveles de grado. En la escuela secundaria, las ciencias de la computación pueden enseñarse en cursos introductorios; cursos de AP y cursos especializados como ciberseguridad, diseño de juegos o robótica. Desafortunadamente, cuando las ciencias de la computación se ofrecen como un curso independiente, a menudo es como un curso optativo. Las ciencias de la computación como optativa presentan una desventaja en comparación con la integración de las ciencias de la computación en una materia que todos los estudiantes toman, tales como las matemáticas o la ciencia, porque

menos estudiantes serán expuestos a las ciencias de la computación. Además, los estudiantes que tienen percepciones erróneas preconcebidas acerca de las ciencias de la computación pueden auto seleccionarse fuera de las ciencias de la computación antes de ni siquiera intentar un curso. Se debe tener en cuenta que los cursos de las ciencias de la computación independientes y la integración interdisciplinaria no se excluyen mutuamente; un curso de las ciencias de la computación puede presentar proyectos que se integran en el contexto de otras disciplinas, tales como matemáticas, ciencias y arte.

Revisar cursos de tecnología y las ciencias de la computación

Uno de los lugares más naturales para aumentar la participación de las ciencias de la computación es a través de créditos y cursos de educación tecnológica preexistentes. Muchos estados y distritos tienen algún tipo de crédito de graduación en tecnología o alfabetización informática, que se puede modificar para permitir que las ciencias de la computación cuenten. Por ejemplo, en 2015, Maryland revisó su requisito de graduación de tecnología de un crédito y permitió que los cursos de las ciencias de la computación cumplieran con este requisito. Anteriormente, solo los cursos que se ajustaban a una perspectiva sobre tecnología que incluía ingeniería, manufactura, transporte, agricultura o medicina se permitían contar (Departamento de Educación del Estado de Maryland, 2005). Antes de 2015, algunos distritos escolares de Maryland ya habían tomado la iniciativa de integrar las ciencias de la computación en sus cursos de educación de tecnología general, lo que llevó a un aumento en el porcentaje de minorías y mujeres poco representadas que tomaban las ciencias de la computación, en comparación con los distritos que aún consideraban que las ciencias de la computación era una optativa (Wilson & Yongpradit, 2015). Desde el cambio en el requisito de graduación de tecnología, Maryland ha informado un aumento dramático en la inscripción de ciencias de la computación (CTE Maryland, 2016). Si los cursos de educación tecnológica cuentan o no como un requisito de graduación, la revisión de cursos obsoletos para enfocarse en ciencias de la computación es práctica apropiada.

Construyendo el camino
Con la introducción de un nuevo tema en el espacio K–12, las escuelas necesitarán desarrollar planes para una implementación gradual. Estos planes deben tener en cuenta los años tempranos de implementación en que los estudiantes de la parte superior de las calificaciones no habrán tenido los fundamentos básicos y pueden no haber aprendido las progresiones completas de los conceptos y prácticas antes de la graduación.

Las escuelas pueden optar por convocar grupos de trabajo con disciplinas individuales (por ejemplo, matemáticas, ciencias, humanidades) para determinar si alguno de los conceptos o prácticas ha sido ya cubiertos o fuertemente alineados con el plan de los estudios actual. El marco ofrece oportunidades para las escuelas a pensar críticamente sobre cómo implementar las ciencias de la computación, tales como a través de lecciones colaborativas entre disciplinas o integración en otras disciplinas.

Consideraciones para personal de las aulas

Los distritos y las escuelas deben considerar cómo desarrollar y dotar de personal nuevos para los cursos de las ciencias de la computación con recursos de personal limitados; qué buscar al contratar profesores de las ciencias de la computación; y cómo aumentar el número de profesores de las ciencias de la computación en un momento en el que hay escasez de temas de ciencias, tecnología, ingeniería y matemáticas (Barth, Dillon, Hull, & Higgins, 2016).

De la misma forma en que reúnen a los equipos verticales para las materias tradicionales, como matemáticas y ciencias, las escuelas o los distritos pueden querer reunir equipos verticales multigrado para las ciencias de la computación. Estos equipos pueden comunicar enfoques positivos, lecciones o ejemplos que son particularmente atractivos para los estudiantes y áreas de fortaleza o debilidad en el aprendizaje de los estudiantes. Como diseñadores de un área temática nueva, estos equipos verticales pueden convertirse en comunidades de aprendizaje profesional, explorando el contenido o los recursos computacionales de los grados superiores o inferiores para desarrollar una mejor comprensión de los resultados esperados de los estudiantes.

Los sistemas escolares que han estado trabajando hacia las ciencias de la computación del distrito brindan un ejemplo del nivel de colaboración requerido para hacer posible una reforma de este tipo. Desde el comienzo de su esfuerzo, las Escuelas Públicas del Condado de Broward en Florida, el séptimo distrito escolar más grande de la nación, formaron un equipo de implementación de ciencias de la computación compuesto por representantes responsables de la certificación, desarrollo profesional, comunicaciones de medios, liderazgo escolar, educación técnica profesional y ciencia, tecnología, ingeniería y matemáticas (STEM). Este equipo estableció una estructura para la colaboración interdepartamental y el desarrollo de una estrategia integral del distrito, lo que resultó en un fuerte aumento en la participación de los estudiantes. Antes de la implementación, en el año escolar 2013-2014, nueve escuelas secundarias y ninguna escuela intermedia o primaria ofrecían ciencias de la computación y atendían a un total de 240 estudiantes de ciencias de la computación. Después de dos años, 33 escuelas secundarias, 34 escuelas intermedias y 113 escuelas primarias ofrecían las ciencias de la computación. Más de 38,000 estudiantes tomaron las ciencias de la computación en 2015–16 (Escuelas Públicas del Condado de Broward, 2016). Este cambio dramático requirió la capacitación de cerca de 1,000 profesores a través de una asociación con proveedores de desarrollo profesional, instituciones de educación superior y partes interesadas (Stakeholders) de la comunidad local (Casa Blanca, 2016).

A pesar de un mayor acceso, los distritos deben moderar continuamente el crecimiento con una implementación de calidad basada en la evidencia. Por ejemplo, las Escuelas Públicas del Condado de Broward están actualmente investigando y evaluando un modelo para integrar las ciencias de la computación y STEM en el programa de primaria. En general, el desarrollo de una secuencia K–12 será un proceso continuo e iterativo basado en una pedagogía computacional emergente que incluye a todos los estudiantes (Snodgrass, Israel y Reese, 2016). Las escuelas y los distritos deberán ser flexibles en su implementación y, posiblemente, explorar múltiples opciones antes de decidir cuál es el ajuste adecuado para sus estudiantes y la comunidad..

Las siguientes secciones resaltan diferentes modelos en los niveles K–8 y secundaria (consulte la Figura 8.6). Estos diferentes modelos se pueden mezclar y combinar para crear vías K–12 personalizadas (consulte la Figura 8.7).

Modelos de primaria y secundaria

La implementación de la instrucción de ciencias de la computación en el nivel K–8 suele ser más flexible que en la escuela secundaria. Los modelos potenciales incluyen unidades de instrucción dedicadas a las ciencias de la computación en las clases de tecnología general y artes de los medios de comunicación, clases semanales dedicadas de computación que se ofrecen como optativas e integración de la instrucción de las ciencias de la computación en otras áreas de contenido. Estos cursos pueden ser impartidos por una variedad de profesores, como profesores de aula de primaria, profesores de área de asignaturas en la escuela, por ejemplo, matemáticas, ciencias, tecnología, música, arte, biblioteca/artes de medios o profesores dedicados a las ciencias de la computación.

Independientemente del modelo que entrega la instrucción, se debe prestar atención a alinear esas experiencias de instrucción con las progresiones del marco.

Modelos de secundaria

El contenido del marco está destinado a todos los estudiantes, y aunque algunos distritos han sido motivados para crear un requisito de graduación en informática, todas las escuelas secundarias deben ofrecer al menos un curso riguroso de informática. Más de la mitad de todos los estudiantes del último año de secundaria no asisten a una escuela secundaria que ofrezca cursos de informática (Change the Equation, 2016). Según los análisis de los datos del Centro Nacional de Estadísticas de Educación, el porcentaje de estudiantes de último año de secundaria que tomaron ciencias de la computación en 2015 (22%) (Change the Equation, 2016) fue menor que en 1990 (25%), con un mínimo de 19 % en 2009 (Nord et al., 2011). Los estados y distritos que buscan dar un primer paso hacia la informática para todos los estudiantes deben considerar comenzar ofreciendo ciencias de la computación en todas las escuelas secundarias.

Todas las escuelas secundarias deben ofrecer al menos un curso riguroso en las ciencias de la computación.

En el transcurso de la implementación de un camino K–12 en ciencias de la computación para todos los estudiantes, los programas de la escuela secundaria tendrán que adaptarse a medida que más estudiantes ingresen a la escuela secundaria con experiencias previas en la escuela primaria. Los cursos de escuela secundaria diseñados para estudiantes que no han tenido experiencia previa en ciencias de la computación continuarán desempeñando un papel importante para ponerse al día con los estudiantes que, por ejemplo, optaron por no participar en cursos electivos en grados anteriores o fueron transferidos entre distritos escolares.

A medida que los estudiantes que ganaron experiencias introductorias deciden continuar su estudio de las ciencias de la computación, las escuelas y los distritos pueden optar por incluir cursos básicos y cursos más avanzados, como los cursos que pueden proporcionar créditos de ingreso a la universidad, preparación técnica y profesional o cursos AP. El contenido de estos cursos será más avanzado que el del marco, pero la organización de conceptos y prácticas centrales del marco puede proporcionar una estructura de guía. Por ejemplo, AP (Emplazamiento Avanzado), por definición, se considera un trabajo de nivel universitario, y el marco proporciona los entendimientos fundamentales que preceden a estas experiencias. A la inversa, algunos cursos existentes, como AP Computer Science, se enfocan principalmente en algoritmos y programación, y las escuelas necesitarán implementar cursos complementarios o currículos para abordar el conjunto completo de conceptos en la banda de grado del 9-12 del marco.

Los estados y los distritos deben considerar si la ciencia de la computación vive dentro de una vía académica, una vía Carreras y Educación Técnica (CET)* o ambas, ya que la elección afectará el acceso, la financiación y el contenido del curso. En 2010, el informe de la Asociación para Maquinaria de Computación, Running on Empty observe

> Un factor importante que contribuye a la confusión sobre la educación en ciencias de la computación es que los cursos de las ciencias de la computación o "computacionales" se organizan en varios departamentos dentro de las escuelas. Por ejemplo, algunos se ubican en los departamentos de matemáticas o ciencias y otros en los departamentos de educación vocacional. Cuando los cursos de las ciencias de la computación se ubican dentro de la educación vocacional, rara vez forman parte del currículo "core" que un estudiante debe tomar. Además, el currículo de estos cursos tiende a centrarse en habilidades de tecnología o de TI más amplias en lugar de conceptos más profundos de las ciencias de la computación. (Wilson, Sudol, Stephenson y Stehlik, 2010, p. 15)

Los cursos de Carreras y Educación Técnica (CET), con su enfoque profesional, brindan grandes oportunidades para los estudiantes que desean explorar especializaciones en ciencias de la computación, como ciberseguridad, administración de bases de datos y software ingeniería. Hay espacio tanto para la clasificación académica y Carreras y Educación Técnica (CET) de las ciencias de la computación. Por ejemplo, los primeros cursos fundacionales en un currículo CET se puede codificar dualmente como parte de la ruta CET, así como una crédito de matemáticas, ciencias o tecnología. Los sistemas CET difieren de estado a estado, y se recomienda que los estados proporcionen orientación a los distritos para que la mayor cantidad posible de estudiantes puedan tener experiencias de las ciencias de la computación fundamentales y participar en las vías de CET.

Los cursos fundamentales en un currículo Carreras y Educación Técnica (CET)* se pueden codificar dos veces como parte de la ruta de CET, así como un crédito de matemáticas, ciencias o tecnología.
****En inglés: Career and Technical Education (CTE).***

Ejemplo vías a K–12

Últimamente, las vías del curso serán impulsadas por el marco y/o estándares que se desarrollan en base al marco. La cantidad de estándares, ya sean específicos para cada grado o agrupados y voluntarios u obligatorios, afectará la elección del camino. En los ejemplos de las vías en la Figura 8.7, una experiencia computacional puede variar desde unas pocas horas a la semana hasta un curso de un semestre o año. Para el propósito de las vías de ejemplo, se supone que la escuela primaria incluye los grados K–5, la escuela intermedia cubre los grados 6-8 y la secundaria cubre los grados 9–12. Los diferentes modelos en cada grupo de grado están organizados por una estimación de la cantidad total de tiempo de instrucción enfocado en ciencias de la computación que el modelo puede permitir, de menor a mayor. Los ejemplos dentro de cada banda de grado no son mutuamente excluyentes; Se pueden combinar muchas opciones para crear vías adicionales para la instrucción en ciencias de la computación.

Figura 8.6: Opciones para implementar las ciencias de la computación

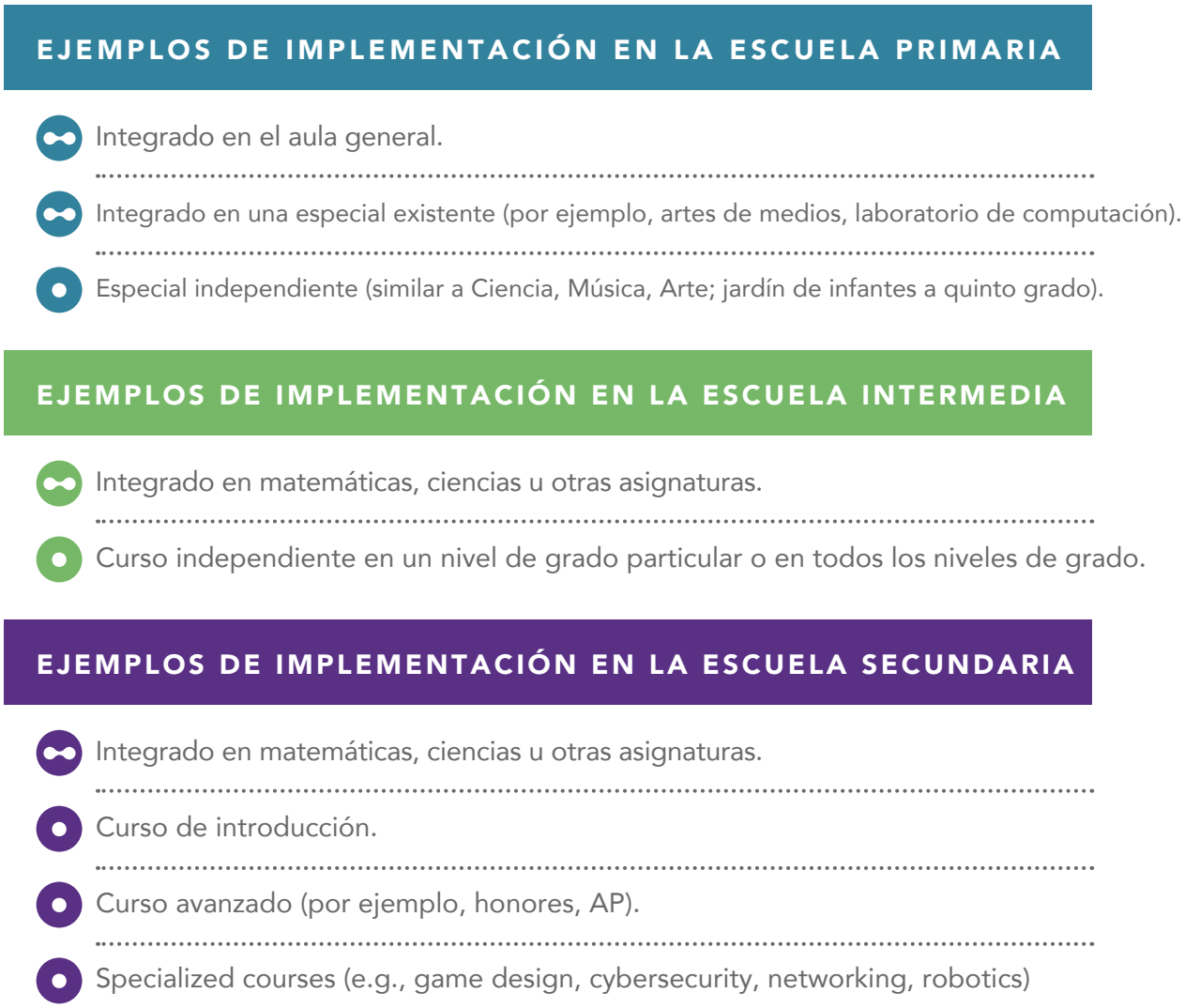


Figura 8.7: Múltiples vías para implementar las ciencias de la computación K–12

Infraestructura Técnica

La educación moderna en las ciencias de la computación no suele requerir configuraciones complejas de hardware o configuraciones de software, aunque deben cumplirse algunas condiciones técnicas básicas. La disponibilidad de computadoras es la mayor preocupación y debe abordarse en el plan de tecnología del distrito o del estado al involucrar a la oficina de tecnología del sistema escolar como un actor clave en las iniciativas de educación en ciencias de la computación. Algunas estrategias comunes a nivel de aula para lidiar con la falta de computadoras son hacer que los estudiantes trabajen de manera colaborativa en los dispositivos, tener un centro computacional (particularmente en el nivel de primaria) o hacer que los estudiantes roten a través de una experiencia de computación dentro de un programa semanal.

Si bien algunos programas y currículos de educación en ciencias de la computación requieren recursos instalados localmente, muchos programas y currículos en ciencias de la computación ahora existen en línea. El uso de programas y currículos que están en línea permite a las escuelas cubrir gran parte del programa con dispositivos compatibles con el navegador, como netbooks y, en algunos casos, tabletas. El uso de una solución basada en la web para respaldar los currículos de ciencias de la computación requiere una conexión de red robusta, pero desafortunadamente, el 23% de los distritos escolares aún carecen de conectividad a una internet rápida y estable (Education Superhighway, 2015). Las escuelas querrán considerar sus opciones basadas en sus propios recursos tecnológicos actuales.

Partes interesadas (Stakeholders)

Aunque la infraestructura física es importante, igualmente importante es el cultivo de partes interesadas (Stakeholders) de la comunidad que pueden apoyar la implementación de las ciencias de la computación. Las ciencias de la computación son disciplinas relativamente nuevas en la educación K–12, por lo que los formuladores de políticas y los administradores de las instituciones educativas pueden no tener claro el contenido adecuado para las clases independientes de las ciencias de la computación o las experiencias integradas dentro de otras áreas de contenido. También pueden tener ideas erróneas sobre el público objetivo de la educación en ciencias de la computación. Por ejemplo, algunas partes interesadas pueden creer que las ciencias de la computación deberían estar disponible solo como oportunidades de enriquecimiento para los estudiantes que normalmente tienen un alto rendimiento, en lugar de para toda la población estudiantil. Sin embargo, al igual que otras áreas temáticas, todos los estudiantes son capaces de aprender los conceptos básicos de las ciencias de la computación, y en la comunidad de educación en ciencias de la computación se acepta ampliamente que los fundamentos de las ciencias de la computación son esenciales para desarrollar habilidades de pensamiento crítico y comprender la tecnología que las personas utilizan e interactuar con diariamente.

"Existe una confusión profunda y generalizada dentro de los estados sobre qué debe constituir y cómo diferenciar la educación, la alfabetización y la fluidez de la tecnología; educación en tecnología computacional; y las ciencias de la computación como asignatura académica." (Wilson, Sudol, Stephenson, & Stehlik, 2010, p. 9).

Socios Comunitarios y de Negocios

La comunidad escolar debe estar educada tanto para informar como para generar apoyo para la implementación de las ciencias de la computación. Los eventos escolares como las noches de regreso a la escuela, las conferencias de padres y profesores, las reuniones de la junta escolar o las exhibiciones académicas, como las ferias de ciencias, pueden ser útiles para comunicar la naturaleza específica de la educación en informática en una escuela o distrito en particular y para involucrar a los funcionarios electos locales. Resaltar y presentar el trabajo de los alumnos es una excelente oportunidad para iniciar una discusión y abordar ideas erróneas sobre qué son las ciencias de la computación, para quién son y cuándo los alumnos deben aprenderla. Muchas organizaciones de educación en ciencias de la computación tienen recursos e ideas para eventos comunitarios y de divulgación.[2]

2 Por ejemplo, el Centro Nacional para la Mujer y la Tecnología de la Información (NCWIT) tiene kits de Outreach-in-a-Box, y Code.org tiene ideas para eventos de Hora de Código.

Un líder escolar entrevistado en LeadCS.org (2015) explica la importancia de involucrar a los socios comerciales al implementar nuevos programas de las ciencias de la computación:

> La idea general es el vecindario: para obtener la colaboración total del vecindario en el que se encuentra la escuela. En algún momento necesitará ayuda para financiar, ya sea a través de subvenciones o negocios locales aquí o allá. [Obtenga] esa colaboración por adelantado para que, si tiene que ir a solicitar financiamiento, ellos estarán en la mesa y sepan con anticipación que la solicitud no sale de la nada. (p. 2)

Además de la financiación, la comunidad y los socios comerciales pueden proporcionar oradores-invitados, asistentes de enseñanza, oportunidades de excursiones y campamentos, e incluso pasantías.

Educación informal

Las organizaciones de educación informal son esenciales para el ecosistema de la educación en ciencias de la computación y deben incluirse como partes interesadas críticas en los esfuerzos de implementación del estado y el distrito. Las organizaciones que participan en redes de educación informal, como las redes estatales después de la escuela, pueden desempeñar un papel importante en el apoyo a las asociaciones con la industria y las instituciones de educación superior, incluida la organización de oportunidades de desarrollo profesional a gran escala.

La educación informal es esencial para el ecosistema de la educación en ciencias de la computación.

La educación informal ofrece a los jóvenes oportunidades para aumentar su interés en ciencias de la computación, desarrollar relaciones con modelos de conducta y desarrollar capacidades para participar en una variedad de actividades computacionales. La educación informal se usa aquí para referirse a organizaciones que brindan servicios extracurriculares, extraescolares, después de la escuela, campamentos u otros entornos de aprendizaje más allá del alcance del día escolar, incluidos los desarrolladores de juegos y aplicaciones en Internet y dispositivos móviles. La educación informal puede proporcionar mayores oportunidades para el aprendizaje basado en proyectos, como lo demuestra el movimiento de creadores en los programas después de la escuela. La educación informal proporciona un entorno natural para actividades socialmente relevantes y situadas culturalmente, como proyectos de impacto en la comunidad. La educación informal también permite a los estudiantes ampliar la base enseñada en la escuela y explorar una amplia gama de temas especializados novedosos más allá del alcance de la educación formal. Estas actividades pueden guiarse por los conceptos y prácticas del marco para proporcionar un enlace al aprendizaje en la escuela.

Los programas de educación informal también tienen el potencial de aumentar significativamente la cantidad de estudiantes y la diversidad de estudiantes que están expuestos a las ciencias de la computación. Según una estimación, 7 millones de estudiantes tienen acceso a oportunidades de aprendizaje STEM después de la escuela, según una encuesta de padres (Afterschool Alliance, 2015, p. 7). Además, los padres de poblaciones afroamericanas, hispanas, asiáticas y caucásicas informan que los niños participan en actividades STEM después de la escuela a tasas similares (aproximadamente el 80%), y los padres de hombres y mujeres reportan niveles similares de acceso (Afterschool Alliance, 2015). Algunas organizaciones se centran en servir a mujeres (e.g., Girls Who Code, 2016; National

Center for Women & Information Technology, 2016; National Girls Collaborative Project, 2016) and minority (e.g., Black Girls Code, 2016; Level Playing Field Institute, 2016) los estudiantes y un gran porcentaje de los participantes que asisten a estos programas expresan interés o planes para estudiar las ciencias de la computación en el nivel postsecundario (Girls Who Code, 2016). El marco tiene potencial para aumentar la coherencia entre la educación formal e informal en las ciencias de la computación, y unir ambos bajo una visión común.

Desarrollo docente

El desarrollo docente es una parte crítica de la infraestructura de educación en ciencias de la computación. El desarrollo docente se utiliza aquí como un término amplio que incluye la preparación, la certificación, la obtención de licencias y el desarrollo profesional continuo de los docentes. Se trata de partes interesadas en la educación superior, agencias estatales, distritos escolares y organizaciones que proporcionan desarrollo profesional.

El marco de las ciencias de la computación K–12 puede informar el diseño de los programas de desarrollo docente. Los conceptos y las prácticas ayudan a los diseñadores de programas y profesores novicios a organizar y comprender la amplitud del conocimiento en ciencias de la computación. El marco fue escrito intencionalmente con el entendimiento de que lo utilizarían profesores que están familiarizados con las ciencias de la computación y profesores que son nuevos en las ciencias de la computación. Esta sección detalla el papel que desempeña el marco en el desarrollo docente y describe las políticas que apoyan y promueven el desarrollo de los docentes en ciencias de la computación.

Preparación de profesores en formación

Existe una carencia a nivel nacional de programas de preparación para profesores en ciencias de la computación. La mayoría de los estados no tienen ni un solo programa de preparación de profesores universitarios en ciencias de la computación. Por ejemplo, en 2014–15, solo 51 profesores de las ciencias de la computación, en los 50 estados, se graduaron de profesores de

los programas de preparación con certificación explícita en ciencias de la computación (Título II, 2016). UTeach, un programa de preparación de profesores STEM que opera en 44 universidades, ha observado que de todas sus asignaturas, el aumento del número de profesores de las ciencias de la computación ha demostrado ser el más difícil (Heitin, 2016). Los colegios y universidades que buscan desarrollar la capacidad de ofrecer cursos especiales para la educación en ciencias de la computación y crear nuevos programas enfrentan una tarea difícil, con desafíos como financiamiento, tiempo, personal, inscripción y experiencia. El panorama actual de la preparación de los profesores de las ciencias de la computación ha reflejado un desafío paradójico: las escuelas secundarias no pueden ofrecer clases de las ciencias de la computación porque no pueden encontrar profesores preparados, y los programas de preparación no preparan a los profesores porque no hay suficientes clases de las ciencias de la computación para que los graduados potenciales puedan enseñar.

Usar el marco para guiar el contenido de preparación docente

El marco se puede utilizar de varias maneras para guiar los programas de preparación de profesores, ya que ayudan a los profesores a desarrollar el contenido y el conocimiento pedagógico necesario para satisfacer las necesidades de una población estudiantil diversa. Esta sección describe cómo el marco de referencia informa a los programas de preparación de profesores, incluida la organización de cursos, el conocimiento del contenido y la práctica pedagógica.

Los conceptos y las prácticas proporcionan una estructura organizativa para enmarcar el conocimiento del contenido de los profesores, pero la profundidad de los cursos no debe estar limitada por las expectativas del estudiante en el marco. Los cinco conceptos core se pueden usar para determinar qué cursos se requieren, lo que podría requerir cursos fuera de la vía tradicional de las ciencias de la computación, que a menudo se centra en la programación. Por ejemplo, los cursos de las ciencias de la computación pueden necesitar integrar el contenido de los cursos de ciencia de datos o de ética. Estos cursos híbridos brindan una oportunidad para la colaboración entre diferentes departamentos en instituciones de educación superior para cumplir con los requisitos en múltiples carreras. Este enfoque puede disminuir la necesidad de profesores de las ciencias de la computación específicos y aumentar la exposición a las ciencias de la computación para estudiantes de otras carreras.

Además de adquirir conocimientos de la materia y habilidades pedagógicas generales, los profesores de las ciencias de la computación deben desarrollar un conocimiento de contenido pedagógico específico para la enseñanza de las ciencias de la computación (Tucker et al., 2006). En la intersección entre el conocimiento de la materia y la pedagogía, el conocimiento del contenido pedagógico incluye una comprensión de lo que hace que el aprendizaje de temas específicos sea fácil o difícil: las concepciones y preconcepciones que estudiantes de diferentes edades y orígenes traen consigo al aprendizaje de los temas y lecciones que se enseñan con mayor frecuencia "(Shulman, 1986, pág. 9). Los cursos de métodos de enseñanza ofrecen el entorno en el que se puede desarrollar el conocimiento pedagógico y de contenido pedagógico. El modelado explícito de la pedagogía de las ciencias de la computación y las oportunidades para ejercer el conocimiento del contenido pedagógico proporcionarán a los docentes estrategias para trabajar con todos los estudiantes, como aquellos que pueden tener problemas con la programación. Las progresiones de aprendizaje del marco pueden usarse para apoyar el desarrollo de los docentes en el conocimiento del contenido pedagógico porque las progresiones describen hitos conceptuales clave y muestran cómo la comprensión de los estudiantes puede desarrollarse con el tiempo. Enfocar las actividades del curso de aprendizaje alrededor del marco las progresiones

de aprendizaje brindan un contexto en el que los profesores en formación pueden identificar posibles ideas erróneas e ideas que los estudiantes pueden encontrar difíciles de comprender.

Los programas de preparación para el preservicio pueden inspirarse en la visión del marco para desarrollar estudiantes que entiendan el mundo a través de la lupa de las ciencias de la computación y puedan aplicar las ciencias de la computación a una variedad de intereses y disciplinas. Para alinearse con esta visión, los programas de preparación de profesores deben alentar a los profesores en formación a conectar las ciencias de la computación con una variedad de contextos personales, prácticos y sociales. Por ejemplo, las actividades del curso podrían incluir una discusión crítica de los eventos actuales que demuestran la omnipresencia de las ciencias de la computación y las formas en que estos eventos se conectan a los conceptos y prácticas relevantes en el marco. Los docentes en preservicio también podrían adquirir experiencia en la producción de una amplia gama de artefactos computacionales del mundo real que son personalmente relevantes y significativos, oportunidades que eventualmente brindarán a los estudiantes.

La realización de esta visión del aula de las ciencias de la computación también requiere que los profesores sean capaces de integrar los conceptos y prácticas del marco en experiencias de aprendizaje significativas. Los docentes en formación deben desarrollar instalaciones para combinar prácticas, como promover las necesidades de diversos usuarios finales, solicitar e incorporar comentarios en el proceso de diseño y defender las decisiones de diseño, con conceptos de los cinco conceptos core del marco. La integración de los conceptos y prácticas también puede servir como puntos focales para proyectos, lecciones y actividades, y puede proporcionar contextos para examinar diferentes enfoques pedagógicos.

Estructuración de programas de preparación docente

Cada vez más, los programas de preparación de docentes en las universidades están creando programas innovadores para exponer a los docentes en formación a aspectos de las ciencias de la computación para que puedan integrarlos en su instrucción o agregar las ciencias de la computación como otra certificación (por ejemplo, DeLyser, 2016). Esta sección describe cómo se pueden estructurar los programas de preparación de profesores para preparar a los profesores para enseñar múltiples áreas de contenido o para integrar las ciencias de la computación en otras áreas de contenido. Las asociaciones entre estos programas, los distritos escolares y los departamentos de educación estatales son fundamentales para aumentar el número de profesores de las ciencias de la computación. Una opción para preparar a los profesores en formación en ciencias de la computación sin crear vías de preparación completas es agregar las ciencias de la computación a los programas de preparación de profesores para otras materias. Por ejemplo, la Illinois State University (2016) tiene un programa de educación en ciencias de la computación que se puede agregar a una especialización en educación matemática, lo que resulta en una doble certificación. Los programas de UTeach en la Universidad de Texas en Austin (UTeach College of Natural Sciences, 2016) y sus 43 universidades asociadas están diseñados para las ciencias de la computación y otras carreras STEM para obtener la licencia de enseñanza sin agregar más tiempo a sus planes de licenciatura. Estos tipos de programas preparan a sus graduados para enseñar dos asignaturas, lo que puede hacer que esos graduados sean atractivos para los distritos escolares que necesitan profesores que puedan enseñar uno o dos cursos de las ciencias de la computación. Dependiendo del programa y con la planificación adecuada, los estudiantes pueden graduarse con una certificación de las ciencias de la computación con pocos o ningún crédito adicional para

para la incorporación del contenido de las ciencias de la computación en un curso obligatorio para todas las carreras de educación, como la teoría del aprendizaje o un curso de tecnología educativa, puede exponer a todos los profesores en formación a las ciencias de la computación. Por ejemplo, el departamento de educación de la Universidad de Purdue incorporó un módulo de una semana sobre pensamiento computacional en un curso obligatorio para estudiantes de primaria y secundaria (Yadav, Zhou, Mayfield, Hambrusch y Korb, 2011). El contenido del pensamiento computacional reemplazó un módulo sobre resolución de problemas y pensamiento crítico al abordar objetivos similares dentro del contexto de las ciencias de la computación. Al aprender conceptos de pensamiento computacional, los profesores pueden estar mejor preparados para integrarlo en su enseñanza y ser más capaces de articular usos más amplios como una herramienta de resolución de problemas en otras disciplinas (Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014).

La incorporación de las ciencias de la computación en un curso obligatorio para todas las carreras de educación puede exponer a todos los profesores en formación a las ciencias de la computación.

Los departamentos estatales y distritales responsables de la contratación de profesores son influyentes para aumentar el número de profesores calificados en ciencias de la computación. Como se sugirió anteriormente, los estados y los distritos deben comunicar su plan de implementación de las ciencias de la computación para involucrar a una variedad de partes interesadas (Stakeholders) en la educación. Esta colaboración es particularmente relevante cuando se contratan más profesores de las ciencias de la computación, ya que las asociaciones con instituciones de educación superior pueden llevar a estrategias conjuntas dirigidas a las ciencias de la computación, oportunidades de desarrollo profesional continuo, colocaciones para pasantías profesionales y un grupo de candidatos para nuevas posiciones de enseñanza en las ciencias de la computación (Barth et al., 2016). Las asociaciones entre estados o distritos y la educación superior pueden facilitar la comunicación sobre las vacantes proyectadas en ciencias de la computación y la demanda de graduados con experiencia en educación en ciencias de la computación. Por ejemplo, los distritos que están integrando ciencias de la computación en cursos preexistentes deben hacer conscientes de los programas que los candidatos a profesores que han tenido experiencia en la integración de las ciencias de la computación se verán favorecidos en el proceso de contratación (consulte la Figura 8.8 para una muestra de actividad de entrevista). Estas asociaciones también pueden beneficiar a los distritos a través de las subvenciones de asociaciones de calidad docente proporcionadas a través del Título II de la Ley de educación superior (2008) que se otorgan a los colegios de educación que trabajan con distritos escolares de alta necesidad para mejorar la preparación de los profesores. En relación cercana con la preparación de los profesores en proceso, la siguiente sección trata el panorama de la certificación de los profesores de ciencias de la computación y ofrece recomendaciones para desarrollar vías de certificación.

Figura 8.8: Ejemplo de actividad de entrevista basada en el marco

El marco puede usarse para iniciar conversaciones durante la contratación o la reasignación para evaluar la preparación de los profesores.

Elija uno de los enunciados conceptuales del marco y pregunte a los profesores candidatos cómo lo integrarán con una de las siete prácticas en el aula. Pídales que describan un proyecto que hayan facilitado en un aula que demuestre el dominio de una de las declaraciones de concepto.

Certificación

Las vías de certificación en prácticas de las ciencias de la computación deben ser directa y claramente comunicadas para que también sean apoyadas por los programas de preparación de profesores, las cuales son contribuciones clave para la sostenibilidad de la implementación de las ciencias de la computación. Desafortunadamente, basado en un análisis de los datos estatales reportados por Code.org (2016a), solo 27 estados ofrecen una opción de certificación en las ciencias de la computación. La Asociación de Profesores de Ciencias de la Computación ha informado que de los estados que ofrecieron la certificación de ciencias de la computación en 2013, 12 no la requerían para enseñar (Comité de Certificación CSTA, 2013). La falta de vías de certificación significa que muchos profesores que actualmente enseñan ciencias de la computación están certificados en otra materia. Similar a la paradoja del programa de aprendizaje, la paradoja de la certificación es que "los estados dudan en exigir la certificación cuando no tienen programas para capacitar a los profesores, y los programas de capacitación de profesores dudan en crear programas para los cuales no existe una vía clara de certificación" (Stephenson, 2015, párr. 2).

El informe de 2013 Bugs in the System resume un panorama de certificación en el que los futuros profesores de las ciencias de la computación se ven frustrados por procesos poco claros, los programas de preparación son pocos y los administradores están confundidos sobre lo que es incluso las ciencias de la computación (CSTA Certification Committee, 2013). El informe dice:

> Este informe sobre la certificación de profesores de ciencias de la computación en los 50 estados y el Distrito de Columbia deja claro que los procesos de certificación/otorgamiento de licencias para las ciencias de la computación tienen fallas profundas. En Florida, por ejemplo, los futuros profesores de las ciencias de la computación tienen que tomar un curso de métodos de ciencias de computación K–8 que no se ofrece en ningún programa de preparación de profesores en el estado. Los futuros profesores de las ciencias de la computación a menudo tienen dificultades para determinar cuáles son los requisitos de certificación/licenciatura en sus propios estados porque nadie parece saberlo. Agregue a esa frustración la confusión que persiste en torno a qué son las ciencias de la computación y dónde encaja en los académicos de K-12, y es sorprendente que los profesionales con una experiencia tan valiosa perseveren para convertirse en profesores de las ciencias de la computación. Pero lo hacen. (Resumen

Las reformas recientes a nivel nacional, junto con la demanda de educación en ciencias de la computación de los estudiantes, padres y líderes empresariales, brindan oportunidades y motivación para que los estados instituyan vías de certificación. La Ley de Educación de STEM de 2015 amplió la definición de STEM para incluir las ciencias de la computación. A esto le siguió el Acta de que todos los estudiantes tuvieron éxito (2015), incluida las ciencias de la computación, como parte de una "educación integral", junto con temas como artes del lenguaje en inglés, matemáticas y ciencias. Como resultado, los programas de aprobación de STEM en el estado de Maryland han utilizado la aclaración como una oportunidad para renovar sus programas para incluir contenido computacional. Este tipo de modificación al respaldo de STEM podría ser considerado por otros estados que apoyan y promueven la educación STEM a través de la certificación o el desarrollo docente.

The Every Student Succeeds Act 2015 (La Ley de Todos los Estudiantes tiene Éxito 2015) incluye las ciencias de la computación como parte de una "educación integral" junto con artes del lenguaje en inglés, matemáticas y ciencias.

Los estados implementan la certificación docente en ciencias de la computación de diferentes maneras. Algunos estados tienen una certificación completa de profesores de las ciencias de la computación, y otros estados tienen un respaldo de las ciencias de la computación que los profesores certificados pueden obtener además de su certificación primaria. Para los estados que buscan desarrollar o expandir sus vías de certificación de profesores de las ciencias de la computación, existen múltiples opciones disponibles. Las ideas potenciales en desarrollo por Code.org (2016b) incluyen sugerencias sobre lo que puede suceder de inmediato, a corto y largo plazo. Si bien la creación de una ruta de certificación completa proporciona sostenibilidad a largo plazo, también requiere tiempo, recursos y colaboración con las instituciones de preparación de profesores. Las soluciones inmediatas y de corto plazo incluyen

- utilizar vías de certificación CET alternativas existentes,
- permitir que los profesores enseñen las ciencias de la computación bajo una licencia temporal mientras obtienen educación profesional en ciencias de la computación o mientras buscan la certificación completa,
- exigir las ciencias de la computación en las vías existentes para la educación tecnológica,
- crear endosos adicionales para profesores que ya están certificados en otras áreas de contenido, y
- desarrollar o adoptar un examen de licencia de profesor de las ciencias de la computación para aprobación.

Estos tipos de soluciones se pueden instituir en paralelo con el desarrollo de una ruta de certificación completa (Code.org, 2016b). En cada caso, los conceptos y prácticas del marco deben utilizarse para informar la selección o el desarrollo de los cursos o exámenes necesarios para la certificación, y los recursos y requisitos para la certificación deben publicarse y ser fácilmente accesibles.

Desarrollo profesional en servicio

Desarrollo profesional en servicio del aprendizaje profesional en ciencias de la computación se basa en experiencias en programas de asistencia técnica para proporcionar una experiencia coherente de desarrollo docente basada en los conceptos y prácticas del marco. El desarrollo profesional se está utilizando actualmente como una forma de preparar a los profesores existentes para

satisfacer la demanda de cursos de las ciencias de la computación. Los distritos escolares deben considerar la posibilidad de colaborar estrechamente con universidades, programas de educación informal y organizaciones para ofrecer desarrollo profesional a gran escala y asegurar una experiencia de desarrollo docente constante.

La falta de profesores de las ciencias de la computación actuales ha llevado a los sistemas escolares a satisfacer las necesidades inmediatas mediante la creación de capacidad entre los profesores existentes.

Los esfuerzos para desarrollar la capacidad docente en ciencias de la computación enfrentan un desafío especial porque los profesores que asisten a oportunidades de desarrollo profesional representan una amplia gama de experiencia. En 2013, un estudio panorámico del desarrollo profesional de las ciencias de la computación, Construyendo un sistema operativo para las ciencias de la computación, encontró que más de la mitad de todos los profesores que asisten al desarrollo profesional en educación de las ciencias de la computación son principiantes en ciencias de la computación, en lugar de los actuales profesores de las ciencias de la computación (Century et al., 2013).

Además, las tres cuartas partes de los proveedores de desarrollo profesional de las ciencias de la computación encuestados informaron que trabajan con las ciencias de la computación participantes que son nuevos en ciencias de la computación. Estos hallazgos son esperados, ya que la falta de profesores de ciencias de la computación actuales ha impulsado a los sistemas escolares a satisfacer la necesidad inmediata de cursos de ciencias de la computación mediante el desarrollo de capacidades entre profesores existentes certificados en otras áreas, como matemáticas, ciencias y tecnología de educación general (Century et al., 2013).

La gran variedad de experiencias de las ciencias de la computación en la formación de docentes requiere experiencias de desarrollo profesional que se diferencien para satisfacer las necesidades de múltiples poblaciones: docentes que ya tienen experiencia y están certificados en ciencias informáticas pero que necesitan educación continua, docentes de otras disciplinas nuevo en la enseñanza de ciencias de la computación, y Carreras y Educación Técnica que se preparan para integrar contenido de ciencias de la computación en otras áreas de contenido. La diferenciación de los niveles de comodidad de los profesores con las ciencias de la computación puede afectar si ese profesor continúa enseñando ciencias de la computación. En un estudio de talleres diseñados para profesores con experiencia previa en ciencias de la computación, los profesores que no tenían formación en ciencias de la computación experimentaron frustración y finalmente dejaron de enseñar las ciencias de la computación (Ericson, Guzdial y Biggers, 2007). Además, las necesidades de los educadores de escuelas secundarias que imparten cursos independientes de las ciencias de la computación pueden diferir de las necesidades de los docentes de escuela primaria que desean incorporar las ciencias de la computación en su enseñanza.

Aunque existen prácticas efectivas que se aplican a todas las experiencias de desarrollo profesional, las siguientes recomendaciones abordan temas específicos de las ciencias de la computación.

Personalizar el desarrollo profesional para conocer los antecedentes variados de los profesores en ciencias de la computación

La experiencia de los docentes con las ciencias de la computación varía, al igual que su área principal de certificación. Cuando sea posible, la audiencia para un taller debe ser homogénea en base a la experiencia en ciencias de la computación y al área de certificación primaria. Cuando esto no sea posible, un taller podría incluir sesiones para que los profesores se dividan en grupos según la experiencia o la certificación.

El desarrollo profesional debe prestar atención a las inquietudes de los profesores principiantes por su falta de conocimiento del contenido

Dada la introducción a las ciencias de la computación en muchos sistemas educativos, es natural que muchos profesores que asisten al desarrollo profesional no tengan experiencia en ciencias de la computación. Si bien no disminuye la importancia del conocimiento del contenido o la práctica pedagógicos general para la enseñanza de las ciencias de la computación, los proveedores de desarrollo profesional deben prestar atención a las ansiedades de los profesores sobre el conocimiento del contenido, ayudándoles a ver que muchos profesores se encuentran en la misma situación. El desarrollo profesional puede inculcar una mentalidad de crecimiento en los participantes, en la que el aprendizaje se desarrolla con el tiempo, durante un taller y durante el año escolar, mientras que los profesores imparten instrucción. El desarrollo profesional debe verse como un espacio seguro para probar cosas nuevas o difíciles.

Los proveedores deben conectar experiencias de desarrollo profesional a un contexto curricular

El contenido disciplinario y pedagógico debe aprenderse en el contexto de las metas educativas, los marcos curriculares y / o los cursos de un profesor. El desarrollo profesional que se enfoca principalmente en un lenguaje de programación o en cómo usar una herramienta, sin darles tiempo a los participantes para que hagan planes para usar esas herramientas o idiomas en sus cursos, es menos práctico y práctico, ya que no prepara a los profesores para entregar un currículo significativo. Alternativamente, el desarrollo profesional relacionado con los conceptos y las prácticas en el marco puede brindar oportunidades para practicar el contenido de la enseñanza (es decir, enseñanza micro) y se puede contextualizar a un currículo particular que los profesores usarán en sus clases.

El desarrollo profesional debe incluir un enfoque en aumentar el acceso y la equidad

Los cursos de las ciencias de la computación a menudo carecen de diversidad y pueden intimidar a muchos estudiantes. Los profesores deben tener experiencia en participar y reflexionar sobre las mismas prácticas en el marco que se espera de los estudiantes, particularmente en términos de acceso y equidad, como incorporar diversas perspectivas en un diseño, satisfacer las necesidades de diversos usuarios finales y crear cargas de trabajo equitativas para equipos. Las ciencias de la computación traen temas únicos que requieren el énfasis de prácticas pedagógicas particulares, como las prácticas equitativas que abordan la exposición variada que los estudiantes tienen en las ciencias de la computación y los estereotipos que existen sobre el campo (Ryoo, Goode y Margolis, 2016). Las oportunidades de desarrollo profesional centradas en estrategias de enseñanza equitativas han demostrado tener éxito en el reclutamiento y la retención de mujeres y minorías poco representadas (Cohoon, Cohoon, & Soffa, 2011). El tema de la equidad en ciencias de la computación se aborda más detalladamente en el capítulo Equidad en Educación en Ciencias de la Computación.

El desarrollo profesional debe abordar la gestión de un entorno productivo de laboratorio de computación.

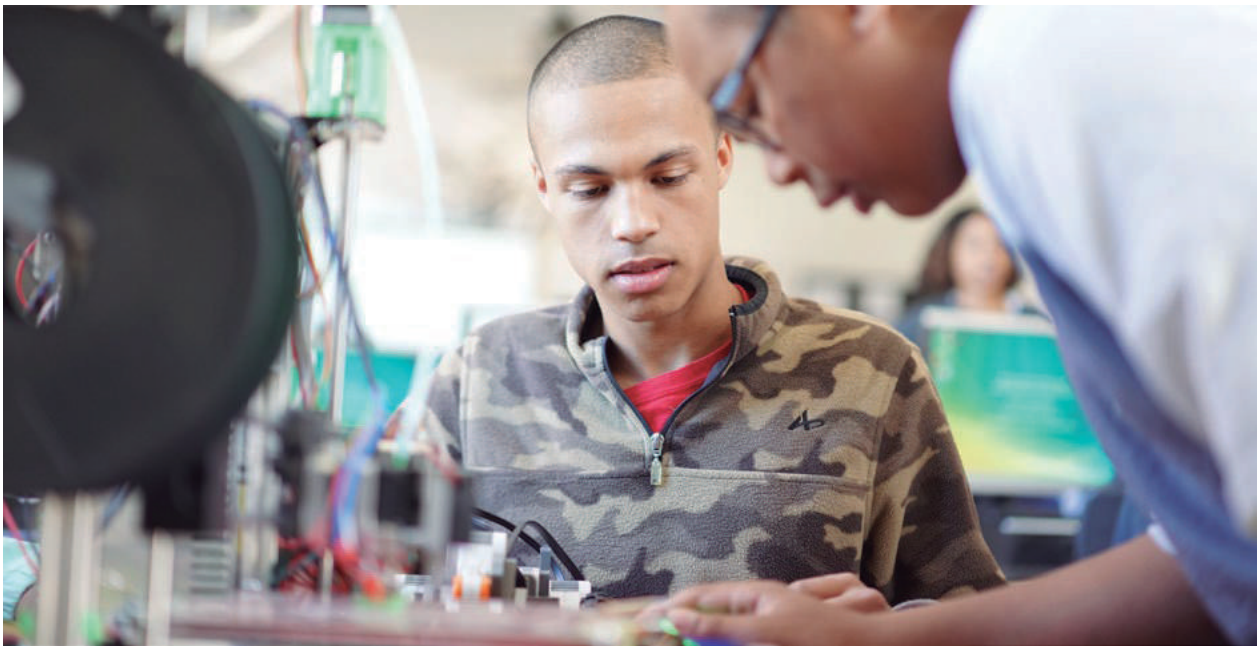
La computadora juega un papel mucho más importante en el aula de las ciencias de la computación que otras. Los estudiantes a menudo se mueven entre el trabajo en computadora y la instrucción en el aula, a veces dentro del mismo período, y otras veces trabajan en la computadora durante días.

El desarrollo profesional debe incluir un enfoque en aumentar el acceso y la equidad.

A veces, un entorno en línea puede proporcionar instrucción a través de videos y tutoriales. Los profesores deben aprender a administrar un aula en la que la computadora sea el medio principal para demostrar el desempeño, así como un auxiliar de enseñanza ocasional. Varias de estas recomendaciones están inspiradas en las sugerencias en Creación de un sistema operativo para informática (Century et al., 2013), que ofrece orientación adicional relacionada con las necesidades de los docentes en ciencias de la computación, incluido un retrato de la población docente en ciencias de la computación y los contextos en los que enseñan. El estudio es accesible en http://outlier.uchicago.edu/computerscience/OS4CS/.

Resumen

Los escritores, asesores y organizaciones que han desarrollado este marco reconocen que los esfuerzos para implementar las ciencias de la computación K-12 existen en un entorno educativo con múltiples prioridades, incluida la participación de los estudiantes, las tasas de graduación de la escuela secundaria, las pruebas de alto nivel, la responsabilidad de los profesores y el déficit presupuestario . De manera similar a un principio que ayudó a guiar los debates y discusiones durante el desarrollo del marco, los formuladores de políticas y educadores deben tomar decisiones constantemente basadas en lo que es mejor para los estudiantes.



Referencias

Afterschool Alliance. (2015). *America after 3 PM: Full STEM ahead.* Washington, DC. Retrieved from http://afterschoolalliance.org/AA3PM/STEM.pdf

Algorithmic Geometry. (2016). Retrieved from http://www.algogeom.org/

Barth, P., Dillon, N., Hull, J., & Higgins, B. H. (2016). *Fixing the holes in the teacher pipeline: An overview of teacher shortages.* Center for Public Education. Retrieved from http://www.centerforpubliceducation.org/Main-Menu/Staffingstudents/An-Overview-of-Teacher-Shortages-At-a-Glance/Overview-of-Teacher-Shortages-Full-Report-PDF.pdf

Black Girls Code. (2016). Retrieved from http://www.blackgirlscode.com

Black, P., Harrison, C., Lee, C., Marshall, B., & Wiliam, D. (2003). *Assessment for learning: Putting it into practice.* Maidenhead, Berkshire, UK: Open University Press.

Bootstrap. (2016). *From intro CS & Algebra to high-level courses in computer science, for all students.* Retrieved from http://www.bootstrapworld.org

Broward County Public Schools. (2016). *Broward codes.* Retrieved from http://browardschools.com/browardcodes

Century, J. (2009, September 25). The vanishing innovation: Why "sustaining change" must be as important as "scaling up." *Education Week.* Retrieved from http://www.edweek.org/ew/articles/2009/09/30/05century.h29.html

Century, J., Lach, M., King, H., Rand, S., Heppner, C., Franke, B., & Westrick, J. (2013). *Building an operating system for computer science.* Chicago, IL: CEMSE, University of Chicago with UEI, University of Chicago. Retrieved from http://outlier.uchicago.edu/computerscience/OS4CS/

Change the Equation (2016, August 9). New data: Bridging the computer science access gap [Blog post]. Retrieved from http://changetheequation.org/blog/new-data-bridging-computer-science-access-gap-0

Chicago Public Schools. (2016, February 24). New CPS computer science graduation requirement to prepare students for jobs of the future [Press release]. Retrieved from http://cps.edu/News/Press_releases/Pages/PR2_02_24_2016.aspx

Cliburn, D. C., & Miller, S. (2008). Games, stories, or something more traditional: The types of assignments college students prefer. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (pp. 138–142), Portland, OR.

Code.org. (2015). *Making computer science fundamental to K–12 education: Eight policy ideas.* Retrieved from https://code.org/files/Making_CS_Fundamental.pdf

Code.org. (2016a). *Promote computer science.* Retrieved from https://code.org/promote

Code.org. (2016b). *Teacher certification recommendations.* Unpublished paper.

Cohoon, J., Cohoon, J. M., & Soffa, M. (2011). Focusing high school teachers on attracting diverse students to computer science and engineering. In *Proceedings of the 41st ASEE/IEEE Frontiers in Education Conference* (pp. F2H-1–F2H-5). doi: 10.1109/FIE.2011.6143054

Computer Science Teachers Association Certification Committee. (2013). *Bugs in the system: Computer science teacher certification in the U.S.* Retrieved from the Computer Science Teachers Association and the Association for Computing Machinery website: https://csta.acm.org/ComputerScienceTeacherCertification/sub/CSTA_BugsInTheSystem.pdf

CTE Maryland. (2016). *Maryland public schools CTE enrollment.* Retrieved from https://www.mdctedata.org/dashboards/summary.php?c=IT&y=2016&l=25

DeLyser, L. A. (2016). *Building a computer science teacher pipeline for New York City.* Retrieved from the NYC Foundation for Computer Science Education website: http://pipeline.csnyc.org/#

Education Superhighway. (2015). *2015 state of the states: A report on the state of broadband connectivity in America's public schools.* Retrieved from http://stateofthestates.educationsuperhighway.org/assets/sos/full_report-55ba0a64dcae0611b15ba9960429d323e2eadbac5a67a0b369bedbb8cf15ddbb.pdf

Eglash, R. (2003). *Culturally situated design tools.* Retrieved from http://csdt.rpi.edu/

Eglash, R., Bennett, A., O'Donnell, C., Jennings, S., & Cintorino, M. (2006). Culturally situated design tools: Ethnocomputing from field site to classroom. *American Anthropologist (108)*2, 347–362.

Ericson, B., Guzdial, M., & Biggers, M. (2007). Improving secondary CS education: Progress and problems. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 298–301).

Every Student Succeeds Act of 2015, Pub. L. No. 114-95. 20 U.S.C.A. 6301 (2016).

Girls Who Code. (2016). *About us.* Retrieved from https://girlswhocode.com/about-us/

Goldweber, M., Barr, J., Clear, T., Davoli, R., Mann, S., Patitsas, E., & Portnoff, S. (2013). A framework for enhancing the social good in computing education: A values approach. *ACM Inroads, 4*(1), 58–79.Go

Governors for Computer Science. (2016). Retrieved from http://www.governorsforcs.org/about

Heitin, L. (2016, August). Physics not offered at 2 in 5 high schools, analysis finds. *Education Week*, *36*(1), 6.

Higher Education Act of 2008, title II U.S.C. § 205 *et seq.*

Illinois State University. (2016). *Teacher education in computer science (TECS).* Retrieved from http://tecs.illinoisstate.edu/resources/undergraduates

Indiana Department of Education. (2016). *Science & computer science.* Retrieved from http://www.doe.in.gov/standards/science-computer-science

LeadCS.org. (2015). *Advice from school leaders: Preparing for computer science in your school.* Chicago, IL: CEMSE, Outlier Research & Evaluation, University of Chicago. Retrieved from http://www.leadCS.org

Level Playing Field Institute. (2016). Retrieved from http://www.lpfi.org

Lopez, J. K. (n.d.). Funds of knowledge. In *Bridging Spanish language barriers in Southern schools.* Retrieved from http://www.learnnc.org/lp/editions/brdglangbarriers/939

Margolis, J., Ryoo, J., Sandoval, C., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads*, *3*(4), 72–78.

Margolis, J., Goode, J., & Chapman, G. (2015). An equity lens for scaling: A critical juncture for Exploring Computer Science. *ACM Inroads, 6*(3), 58–66.

Maryland State Department of Education. (2005). *Maryland technology education state curriculum.* Retrieved from http://mdk12.msde.maryland.gov/instruction/curriculum/technology_education/vsc_technologyeducation_standards.pdf

Massachusetts Department of Elementary and Secondary Education. (2016, June). *2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework.* Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

Moll, L., Amanti, C., Neff, D., & Gonzalez, N. (1992). Funds of knowledge for teaching: Using a qualitative approach to connect homes and classrooms. *Theory into Practice 31*(2), 132–141.

National Center for Women & Information Technology. (2016). Retrieved from https://www.ncwit.org

National Girls Collaborative Project. (2016). Retrieved from https://ngcproject.org

Nord, C., Roey, S., Perkins, R., Lyons, M., Lemanski, N., Brown, J., and Schuknecht, J. (2011). *The nation's report card: America's high school graduates (NCES 2011-462).* U.S. Department of Education, National Center for Education Statistics. Washington, DC: U.S. Government Printing Office.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York, NY: Basic Books.

Project GUTS: Growing Up Thinking Scientifically. (2016). Retrieved from http://www.projectguts.org

Rader, C., Hakkarinen, D., Moskal, B. M., & Hellman, K. (2011) Exploring the appeal of socially relevant computing: Are students interested in socially relevant problems? In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 423–428). doi: 10.1145/1953163.1953288

Ryoo, J., Goode, J., & Margolis, J. (2016): It takes a village: Supporting inquiry- and equity-oriented computer science pedagogy through a professional learning community. *Computer Science Education*. doi: 10.1080/08993408.2015.1130952

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*(2), 4–14. doi: 10.3102/0013189X015002004

Snodgrass, M. R., Israel, M., & Reese, G. (2016). Instructional supports for students with disabilities in K–5 computing: Findings from a cross-case analysis. *Computers & Education, 100,* 1–17.

STEM Education Act of 2015, Pub. L. No. 114-59. 129 Stat. 540 (2015).

Stephenson, C. (2015, July 16). The thorny issue of CS teacher certification [Blog post]. Retrieved from https://research.googleblog.com/2015/07/the-thorny-issue-of-cs-teacher.html

Title II. (2016). *2015 Title II Reports: National teacher preparation data* [Data file]. Retrieved from https://title2.ed.gov/Public/Home.aspx

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

UC Davis C-STEM Center. (2016). *Transforming math education through computing*. Retrieved from http://c-stem.ucdavis.edu/

UTeach College of Natural Sciences. (2016). *Computer science.* Retrieved from https://austin.uteach.utexas.edu/certifications-and-degrees/certifications/high-school-certifications/computer-science

White House. (2016, September 13). *FACT SHEET: New progress and momentum in support of President Obama's Computer Science for All initiative*. Retrieved from https://www.whitehouse.gov/sites/default/files/microsites/ostp/csforall-fact-sheet-9-13-16-long.pdf

Wiggins, G., & McTighe, J. (2005). *Understanding by design* (Expanded 2nd ed.). Alexandria, VA: Association for Supervision and Curriculum Development.

Wilson, C., & Yongpradit, P. (2015, June 9). Maryland moves to increase diversity in computer science [Blog post]. Retrieved from http://blog.code.org/post/121123281798/md

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). *Running on empty: The failure to teach K–12 computer science in the digital age*. Retrieved from the Association for Computing Machinery and the Computer Science Teachers Association website: http://runningonempty.acm.org/

WGBH Educational Foundation & the Association for Computing Machinery. (2009). *New image for computing: Report on market research (April 2009).* Retrieved from http://www.acm.org/membership/NIC.pdf

Yadav, A., Burkhart, D., Moix, D., Snow, E., Bandaru, P., & Clayborn, L. (2015). *Sowing the seeds: A landscape study on assessment in secondary computer science education*. Retrieved from the Computer Science Teachers Association website: https://csta.acm.org/Research/sub/Projects/ResearchFiles/AssessmentStudy2015.pdf

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), Article 5, 1–16.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., and Korb, J. (2011) Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 465–470), Dallas, TX.

Las Ciencias de la Computación en la Educación Infantil

9

Las Ciencias de la Computación en la Educación Infantil

En medio de las preocupaciones sobre una fuerza laboral no preparada del siglo XXI, los responsables de la formulación de políticas de los Estados Unidos han puesto mayor énfasis en los temas de ciencia, tecnología, ingeniería y matemáticas (STEM) para asegurar que los jóvenes estén lo suficientemente equipados para competir en la economía cada vez más global (por ejemplo, la National Science Board , 2012; Consejo de Asesores sobre Ciencia y Tecnología del Presidente de los Estados Unidos, 2010; Comité Económico Conjunto del Congreso de los Estados Unidos, 2012). Las ciencias de la computación, en particular, se ha centrado recientemente en el ámbito nacional, con iniciativas a gran escala a nivel local, estatal y federal con el objetivo de asegurar que los estudiantes adquieran habilidades de alfabetización computacional consideradas como "una habilidad" básica "necesaria para las oportunidades económicas y la movilidad social. "(Smith, 2016, párr. 1).

Muchas de estas iniciativas en las ciencias de la computación se centran en los entornos de educación K–12 y postsecundaria. Por ejemplo, Maryland ha establecido estándares para las ciencias de la computación preescolar (pre-K) (Departamento de Educación del Estado de Maryland, 2015), mientras que varios distritos escolares, incluido el Distrito Escolar Unificado de San Francisco (Twarek, 2015) y las Escuelas Públicas de Boston (2016), se encargaron de iniciar la educación en informática en el nivel preescolar.

Con el mismo espíritu de proporcionar oportunidades de aprendizaje de las ciencias de la computación para todos los estudiantes de K-12 como un medio para asegurar una fuerza laboral preparada y productiva para el siglo XXI, se ha demostrado que invertir en la educación durante la infancia es uno de los mejores medios para cerrar los logros tempranos y las brechas de desarrollo, que posteriormente ayudan al bienestar económico y social de la comunidad en general (Heckman, 2006; Heckman y Masterov, 2007; Magnuson, Meyers, Ruhm, & Waldfogel, 2004).

Desde estudios históricos sobre el Programa preescolar High/Scope Perry (Schweinhart et al., 2011) y el Proyecto de antecedentes de Carolina (Campbell, Ramey, Pungello, Sparling y Miller-Johnson, 2002) hasta una reciente

revisión de 84 intervenciones de educación durante la infancia (Camilli, Vargas, Ryan y Barnett, 2010), la investigación muestra constantemente que las experiencias de aprendizaje temprano de alta calidad tienen efectos positivos a corto y largo plazo en el aprendizaje y el desarrollo de los niños. Si bien las "experiencias de aprendizaje de calidad" específicas pueden parecer diferentes de un aula a otra, varios elementos comunes incluyen interacciones de apoyo emocional y emocional entre profesores y alumnos; recursos y materiales curriculares apropiados para el desarrollo; actividades de aprendizaje estructurado individualizadas para satisfacer las diferentes necesidades de los estudiantes; y oportunidades para la exploración y el juego (por ejemplo, Pianta, Barnett, Burchinal y Thornberg, 2011; Yoshikawa et al., 2013).

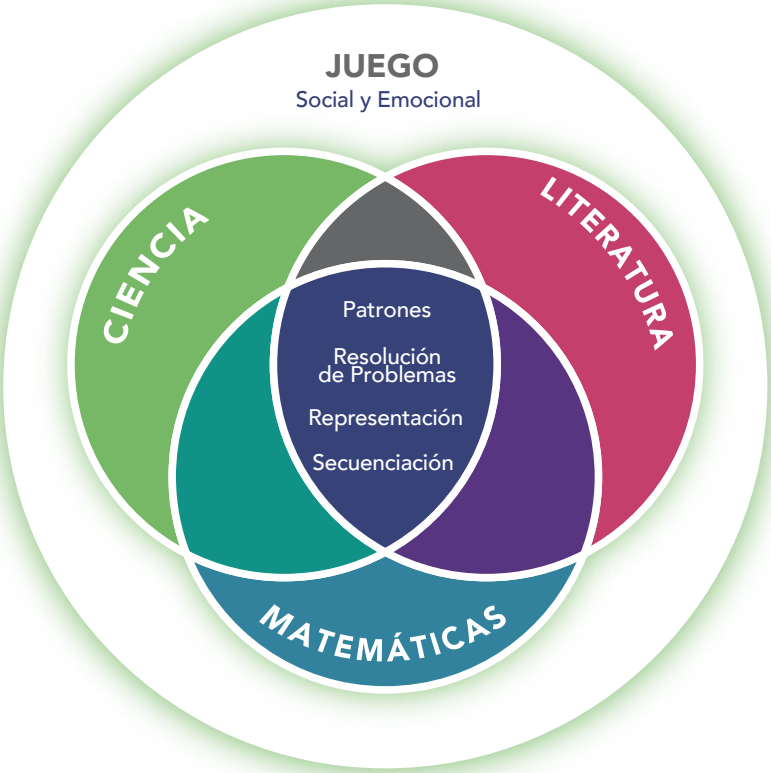
Dado el gran impacto que la educación de la primera infancia puede tener en el aprendizaje y el desarrollo de los niños pequeños, y los impactos posteriores que este aprendizaje fundamental tiene en el bienestar económico y social más amplio, ¿cuál es, entonces, el lugar de la informática en la educación de la primera infancia? En respuesta a esta falta de claridad con respecto a cómo se ve la informática en el aula de la primera infancia, este capítulo describe las aplicaciones prácticas de los conceptos y prácticas del marco de las ciencias de la computación K–12 apropiados para el entorno de pre-K.

Ideas poderosas en las ciencias de la computación de Pre-K

Investigadores científicos, educadores y formuladores de políticas han sugerido una gran cantidad de ideas sobre los principios de la educación en ciencias de la computación. El propuesto del Marco de Ciencias de la Computación K-12 es en sí mismo una amalgama de tales principios y se basa en las "ideas poderosas" de Papert (1980) para articular conceptos y prácticas de las ciencias de la computación específicos para el ambiente de aprendizaje K-12. Sin embargo, en lugar de simplemente aplicar estos conceptos y prácticas core al entorno de preescolar y, por lo tanto, asumir que son apropiados para el desarrollo y necesarios para que todos los niños pequeños aprendan, este capítulo describe un conjunto de "ideas poderosas" específicas para la educación durante la infancia. Estos conceptos y prácticas de las ciencias de la computación preescolar construyen conocimiento y comprensión fundamentales para su posterior participación en ciencias de la computación a nivel de escuela primaria. Se basan en la literatura de investigación sobre educación en ciencias de la computación en entornos de educación durante la infancia y en el marco construccionista de Papert (1980), que enfatiza la participación de los niños en la construcción de conocimiento a través de la construcción de objetos físicos, donde las tecnologías computacionales son herramientas con las que los niños pueden construir y diseñar para desarrollar tal conocimiento. Los profesores apoyan estas experiencias de aprendizaje activo proporcionando un apoyo estructurado que ayuda a guiar a los estudiantes a una participación más profunda y un pensamiento de nivel superior. El construccionismo proporciona la base para gran parte de la investigación y la práctica de la educación en ciencias de la computación y se alinea con las concepciones tradicionales de la educación durante la infancia como un entorno de aprendizaje práctico, interactivo y basado en el juego (Bers, Ponte, Juelich, Viera y Schenker, 2002).).

Como se describe en la Figura 9.1, cuatro ideas poderosas están integradas en las áreas de contenido core de matemáticas, alfabetización y ciencias, y la quinta, el aprendizaje social y emocional, se entiende como un marco holístico para todas las prácticas educativas de la primera infancia. Además, estas ideas poderosas están abarcadas por la base pedagógica de los entornos de aprendizaje temprano: JUGAR.

Figura 9.1: Integración de ideas poderosas en ciencias de la computación y educación durante la infancia.



En las siguientes secciones, estas cinco ideas poderosas se describen en contextos actuales y cotidianos de pre-kínder y luego se extendió a un contexto de las ciencias de la computación. De esta manera, las ciencias de la computación se convierten en una extensión natural del compromiso diario de los niños con su entorno y se basa en lo que los educadores ya lo hacemos en su práctica diaria. Además, cada idea poderosa está conectada a una o más de las prácticas del marco para proporcionar información sobre la progresión desde la pre-K a las ciencias de la computación primaria.

1. **Aprendizaje social y emocional:** las sólidas competencias afectivas, conductuales y cognitivas proporcionan la base para un aprendizaje y desarrollo exitosos.

Tal como lo define la Colaboración para el aprendizaje académico, social y emocional (CASEL, 2012), el aprendizaje social y emocional (SEL) implica los procesos mediante los cuales los niños y adultos adquieren y aplican de manera efectiva los conocimientos, las actitudes y las habilidades necesarias para comprender a manejar emociones, establecer y alcanzar metas positivas, sentir y mostrar empatía por los demás, establecer y mantener relaciones positivas y tomar decisiones responsables (p. 4). "Las cinco competencias principales de SEL son autoconciencia, autogestión, conciencia social, relación Habilidades y toma de decisiones responsables.

Los niños desarrollan habilidades sociales y emocionales a través de interacciones lúdicas con compañeros y adultos, y la investigación muestra continuamente que estas interacciones pueden tener impactos significativos en el aprendizaje y desarrollo de los niños (por ejemplo, Mashburn et al., 2008; Pianta et al., 2002). Más de 500 estudios demuestran los beneficios positivos de SEL para las relaciones interpersonales, la cognición y el aprendizaje académico de los niños en todas las áreas de contenido (p. Ej., Klem & Connell, 2004; Weissberg & Cascarino, 2013; Weissberg, Durkak, Doitrovich, & Gullota, 2015) y una sólida base SEL desarrollada durante la infancia puede tener un impacto duradero en el éxito académico y profesional del futuro de los niños (por ejemplo, Camilli et al., 2010; Chetty et al., 2011). Es importante destacar que cuando los profesores se involucran emocionalmente con los estudiantes, demuestran que les importan, escuchan las necesidades y deseos de los estudiantes y se toman el tiempo para tener en cuenta los tonos momentáneos del clima del aula, ambos pueden modelar y obtener SEL entre los niños.

Conexiones dentro del marco: P1. Fomentar una cultura de computación inclusiva, P2. Colaborar en torno a la computación y P7. Comunicar sobre la computación

Dentro el contexto del marco, las prácticas 1, 2 y 7 abarcan poder trabajar y comunicarse con equipos con muchas perspectivas diferentes. Los profesores pueden fomentar un entorno de las ciencias de la computación inclusivo al presentar oportunidades para que los estudiantes compartan, colaboren y se apoyen entre sí. También pueden animar a los niños a ser conscientes de su propio compromiso. Estas habilidades pueden abordarse de manera proactiva a través de conversaciones sobre las diferencias en los comportamientos, opiniones y perspectivas; abogando por uno mismo y amigos; y luchas a través de compromisos lúdicos que se resolvieron (o podrían resolverse) a través de la mediación y la resolución empática de problemas.

En ciencias de la computación, los mejores productos son creados por equipos formados por miembros con antecedentes variados que escuchan y respetan las ideas de los demás. Además, las ciencias de la computación son más que solo crear productos e implica la comunicación eficaz (verbal y visual) de procesos y soluciones para un público más amplio. Estos principios pueden desarrollarse en el aula de preescolar al fomentar el desarrollo social y emocional de los niños a través del juego.

En ciencias de la computación, los mejores productos están creados por equipos formados por miembros con antecedentes variados que escuchan y respetan las ideas de los demás.

Ejemplo cotidiano

Aprender a jugar con un nuevo compañero de juegos es una ocurrencia regular en un ambiente de aprendizaje temprano y, a menudo, implica un proceso de negociación que requiere el andamiaje del profesor. Los educadores pueden establecer estructuras para ayudar a facilitar este proceso en tres fases. Primero, el niño A elige qué juego jugar durante cinco minutos. Luego, el niño B elige qué juego jugar durante cinco minutos. En la tercera iteración, los estudiantes practican compromisos para encontrar un juego que ambos disfruten. Los profesores pueden ayudar a facilitar esta tercera fase al alentar a la niña que establezca un atributo de un juego que quiere jugar sin nombrar el juego real. Por ejemplo, "construir algo" en lugar de "jugar con bloques." Entonces el niño B refina esta

sugerencia con algo que él quiere. Si los niños tienen más habilidades verbales y de razonamiento, cada uno podría describir varios atributos y luego resolver problemas juntos para crear un juego que satisfaga todos los atributos. Esto permite que cada niño tome el turno de ser el líder y el turno de ser un oyente y seguidor respetuoso, así como también les proporciona un plan para tener una conversación que tenga en cuenta múltiples opiniones.

Ejemplo computacional

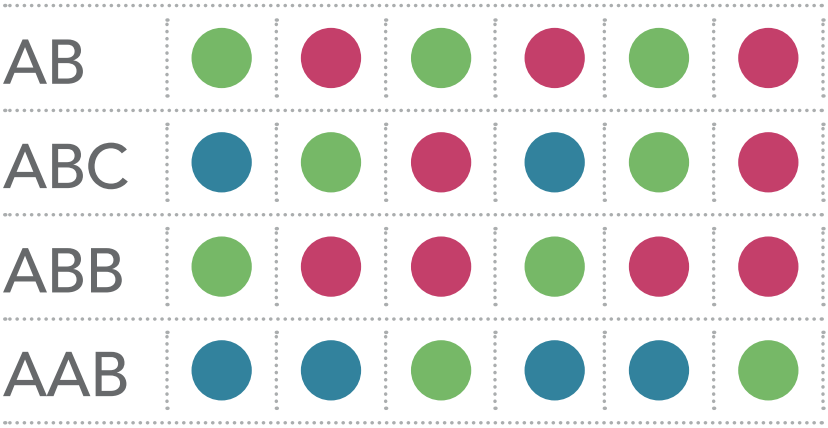
A menudo, los entornos de educación durante la infancia no están equipados con dispositivos computacionales individuales, de manera que los niños trabajan en parejas o en pequeños grupos de forma predeterminada. Los educadores pueden aprovechar esta configuración natural al facilitar experiencias de programación en parejas para niños. En el nivel más simple, esta facilitación podría ayudar a los estudiantes a aprender a compartir el dispositivo mediante el uso de las tarjetas "Mi turno/Tu turno" y los niños pasan las tarjetas de un lado a otro para designar a quién le toca usar la computadora. Yendo un paso más allá, los educadores pueden brindar oportunidades para el aprendizaje colaborativo asistido por computadora (Dillenbourg, 1999; Goodyear, Jones y Thompson, 2014) en el que los niños trabajan juntos en la computadora para resolver una tarea o problema compartido, en lugar de simplemente tomar turnos separados utilizando el dispositivo. La programación de pares implica que una persona asuma el rol de "conductor" mientras que la otra es el "navegador." El conductor es la persona que controla las acciones de la computadora y se enfoca en los detalles, mientras que el navegador tiene una vista más amplia de la imagen del problema y ayuda al responder preguntas y buscar posibles problemas o errores. En el nivel preescolar, los profesores pueden ayudar a facilitar la programación de parejas entre dos niños con las mismas tarjetas de "Mi turno/Tu turno" para designar los roles de conductor/navegador, así como alentar a los niños a participar en habilidades de colaboración y comunicación para fomentar el andamiaje entre pares. Los educadores pueden brindar más apoyo y andamiaje al participar en la programación de pares de niños/profesores.

La programación en pares implica que una persona asuma el rol de "conductor" mientras que la otra es el "navegador."

2. Patrones: los patrones nos ayudan a comprender el mundo mediante la organización de objetos e información usando características comunes (por ejemplo, color, forma, tamaño).

En ciencias de la computación, los patrones permiten a las personas reducir la complejidad al generalizar y aplicar soluciones a múltiples situaciones. Aprender sobre patrones en los primeros años puede construir una base para desarrollar y usar abstracciones (por ejemplo, definir y llamar a procedimientos), resolver problemas computacionales de manera más efectiva (por ejemplo, usar bucles en lugar de repetir comandos) y hacer inferencias (por ejemplo, usar modelos y simulaciones para sacar conclusiones). Un ejemplo se muestra en la Figura 9.2.

Figura 9.2: Identificar patrones



Conexiones al marco: P4. Desarrollo y uso de abstracciones

Un aspecto del desarrollo y uso de abstracciones es la capacidad de categorizar elementos/objetos/-códigos e identificar atributos generales basados en esas categorizaciones (o "abstraer" patrones más generales para describir las categorizaciones). La abstracción es un paso más allá del reconocimiento de patrones, donde el enfoque está en identificar y describir características repetidas pero aún no categorizar elementos/objetos/código en función de esas características o abstraer atributos más generales para definir esas categorizaciones.

Ejemplo cotidiano

Los niños aprenden a reconocer patrones a través de rutinas que brindan experiencias para notar y nombrar las características de las cosas en sus mundos. Por ejemplo, un niño puede notar y nombrar los colores en un salón de clases o notificar y nombrar diferentes características de Legos®. Con el tiempo, pueden comenzar a identificar características repetidas y, a su vez, crear imágenes o mover objetos para mostrar características repetidas. Por ejemplo, un niño puede colocar objetos de colores en un orden: verde, rojo, verde, rojo, verde, etc. Esos patrones pueden volverse aún más elaborados a medida que se toman en cuenta colores o características adicionales: amarillo, verde, rojo, amarillo verde rojo. Además, el ritmo es un gran ejemplo de un patrón porque presenta un patrón repetido de sonido y movimiento. Los niños pueden mostrar patrones en movimiento a través de la danza, repitiendo un movimiento físico.

Al extender este paso un paso más allá en las habilidades de abstracción temprana, los estudiantes pueden recibir una categoría específica (por ejemplo, perro o gato) e identificar lo que saben (es decir, los atributos) en función de esa categorización. Por ejemplo, para la categoría "gato", algunos atributos que los niños sabrán son el tamaño aproximado (significativamente más pequeño que un caballo), el color (no púrpura, verde o azul), el número de ojos (dos) y las piernas (cuatro), y la existencia de una cola. Los niños también saben que a un gato le gusta saltar y que maúlla, pero un gato no puede ladrar ni puede volar.

Ejemplo computacional
En el mundo digital, las computadoras usan patrones para organizar la información. Un profesor puede mostrar cómo los patrones repetidos están en todas partes del mundo. Por ejemplo, los patrones de códigos de barras que están en todas partes en una tienda de comestibles están destinados a proporcionar información sobre el artículo a la caja registradora. El código de barras utiliza funciones repetidas (líneas finas, líneas gruesas y espacio) para dar instrucciones a los dispositivos computacionales. El "pitido, pitido, pitido" del escáner también es un patrón en la tienda de comestibles porque es una función repetida. Los patrones se pueden agrupar. Entonces, por ejemplo, podríamos agrupar el patrón de pitido de código de barras, pitido de código de barras, pitido de código de barras.

Los profesores pueden usar un entorno de programación visual basado en bloques para presentar un patrón de comandos y guiar a los estudiantes en la identificación del patrón. Muchos lenguajes de programación basados en bloques utilizan diferentes colores para los tipos de bloques. Los profesores pueden demostrar cómo se puede usar una serie de comandos para dibujar una forma simple, como un cuadrado o un triángulo, y pedir a los estudiantes que identifiquen patrones utilizando diferentes colores como señales visuales. Esta actividad puede servir como un precursor para que los estudiantes creen sus propios programas de forma independiente.

3. Resolución de problemas: los niños construyen conocimiento a través de la resolución de problemas.

Los niños pequeños se involucran naturalmente en los procesos de resolución de problemas en sus vidas diarias mientras exploran e interactúan con el mundo que los rodea. Los profesores pueden ayudar a hacer que la resolución de problemas sea "visible" al hacer preguntas para descubrir el razonamiento y los procesos de pensamiento de los niños (por ejemplo, ¿Cómo lo supo? ¿Qué lo hizo pensar eso?), Además de ofrecer métodos estructurados para organizar los problemas de los niños. Uno de estos métodos que se usa a menudo en la informática es un proceso de desarrollo iterativo. Este proceso implica identificar un problema; idear y probar soluciones; evaluando los resultados; y revisando y rehaciendo para encontrar la mejor solución. Para este proceso es fundamental cometer errores y aprender de ellos para resolver nuevos problemas de manera efectiva en diferentes situaciones.

Conexiones del marco: P3. Reconocimiento y definición de problemas computacionales, P5.Creación de artefactos computacionales y P6. Probar y refinar artefactos computacionales.
En un sentido, las ciencias de la computación son estudios de problemas, procesos de resolución de problemas y las soluciones que resultan de tales procesos. Al involucrarse en las actividades de resolución de problemas desde el principio puede establecer la base para reconocer y definir problemas computacionales, participar en estrategias de prueba y refinamiento, y desarrollar y evaluar soluciones computacionales para problemas del mundo real.

Ejemplo cotidiano

Los niños suelen resolver problemas cuando construyen con bloques. Por ejemplo, al hacer un puente de bloques para que los autos de juguete se crucen, un niño puede mover dos bloques de base uno cerca del otro y agregar un bloque en la parte superior. Si el bloque superior no llega a través de los bloques de la base, entonces el niño podría acercar los bloques de la base y tratar de equilibrar el bloque de nuevo. Un profesor puede hacer que este comportamiento de resolución de problemas sea visible para los niños al explicar cómo la acción del niño demuestra la revisión y el rehacer. La profesora puede hablar sobre el proceso de pensamiento del estudiante que involucró este tipo de resolución de problemas, haciendo comentarios como, "Wow, me di cuenta de que descubrió que el puente no funcionaría". ¿Cómo supiste qué hacer a continuación para resolver tu problema?

Además, los profesores pueden involucrar a los niños en un proceso de desarrollo iterativo mediante la creación de un "Estudio de Inventores" en su salón de clases donde los niños utilizan las habilidades de resolución de problemas para crear algo nuevo con la ayuda de los andamios de los profesores y los recursos tecnológicos (consulte la Figura 9.3). Por ejemplo, si un niño quiere hacer un títere de calcetín, primero puede dibujar imágenes de su títere de calcetín en papel y anotar todos los materiales que necesitaría para hacerlo. Luego, con la ayuda de una profesora, ella podría ver imágenes de títeres de calcetines en línea y buscar instrucciones sobre cómo hacer uno. Luego, el niño puede comparar y revisar su dibujo original y la lista de materiales en función de lo que ella y su profesora encuentran en línea antes de intentar construir el títere de calcetín real. A lo largo del proceso de construcción, la profesora también puede usar la tecnología para ayudar a documentar este proceso de diseño de resolución de problemas tomando fotos en el camino para que la niña pueda mirar atrás y reflexionar sobre los diferentes pasos que se tomaron para crear su títere de calcetín.

Un profesor también puede presentar problemas explícitamente y pedir soluciones creativas para los niños. Por ejemplo, puede presentar un escenario: un ratón quiere saltar sobre una cama, pero la caja de zapatos cercana es demasiado corta para ayudar al ratón a llegar a su destino. ¿Cómo podría el ratón resolver este problema? ¿Qué otra cosa podría usar un ratón para llegar a la parte superior de la cama? En este ejemplo, el profesor puede solicitar muchas ideas diferentes y enfatizar que no hay una respuesta correcta para resolver este problema, pero que algunas soluciones pueden ser más efectivas y eficientes que otras.

Ejemplo computacional

Cuando los desarrolladores crean una nueva tecnología, a menudo utilizan un protocolo de diseño iterativo que implica crear una versión anterior de la tecnología, probarla, evaluar los resultados, realizar revisiones y luego probarla de nuevo. En el aula, a veces la tecnología no funciona, y los profesores pueden involucrar a los niños en un proceso de resolución de problemas iterativo similar para descubrir por qué. Pueden verificar si el dispositivo está encendido, si está apagado o si está físicamente roto. Estas diferentes comprobaciones son importantes para averiguar por qué la tecnología no funciona y cómo hacer que funcione nuevamente.

Figura 9.3: Estudiante utilizando recursos tecnológicos durante "Inventors Studio".

Inventores y iPads durante la infancia

erikson TEC Center

BY AMANDA ARMSTRONG – JANUARY 22, 2015
POSTED IN: SHOW ME VIDEOS, TEC

Ravenswood Prekindergarten Teacher Yeliz Zurawic tells us how she uses iPads during "Inventors Studio". This one-on-one activity allows students to create their own inventions and present them to the class at the end of the week. During this interview, she shares how the iPad is uses iPads to enhance student learning through investigation, problem solving, and creativity.

4. **Representación**: la gente puede representar conceptos usando símbolos

Cualquier idioma que tenga una versión impresa es un ejemplo de cómo se puede representar el idioma. En el caso del inglés, el idioma está representado por palabras o partes de palabras, que denotan sonidos y significados. De manera similar, los lenguajes computacionales están representados por números, texto y símbolos.

Conexiones al marco: P4. Desarrollo y uso de abstracciones, P5. Creación de artefactos computacionales y P7. Comunicación sobre computación.

Comprender la representación en los primeros años puede construir una base para entender cómo las computadoras representan información y simulan el comportamiento de los sistemas, los cuales son importantes para desarrollar y usar abstracciones. Además, la creación de artefactos computacionales implica el desarrollo de simulaciones y visualizaciones que requieren una comprensión de cómo las computadoras representan los datos, y la comunicación efectiva sobre la computación implica presentar información a través de representaciones visuales (por ejemplo, guiones gráficos, gráficos).

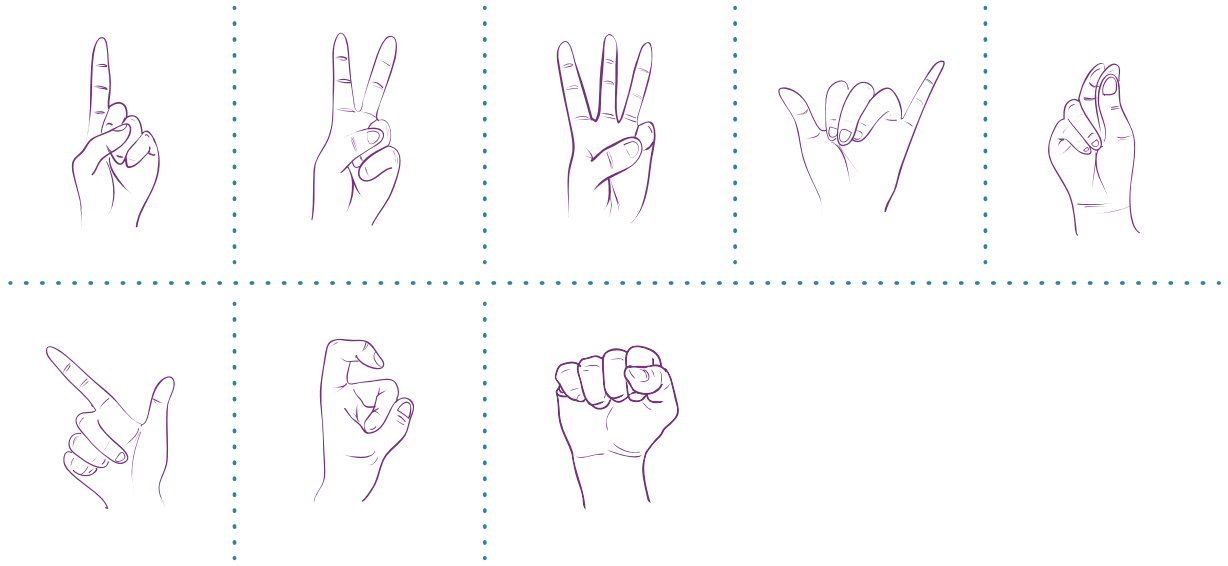
Ejemplo cotidiano

Los niños hacen dibujos de su familia que a menudo se parecen a "personas de papa": círculos para la cabeza y el cuerpo y líneas que sobresalen como brazos y piernas. A veces, estas representaciones de "gente de papa" tienen significados más profundos e historias detrás de ellos. Los profesores pueden ampliar la idea de representación pidiéndole a un niño que comparta lo que significa su imagen y escribiendo, en un texto impreso, una oración para describir la representación pictórica del niño.

Alternativamente, los profesores pueden introducir la idea de representación presentando una imagen o símbolo que representa una idea, por ejemplo, un profesor puede representar direcciones con iconos como una flecha para avanzar, una espiral para dar la vuelta, o un octágono para detenerse. Estas representaciones icónicas pueden ser utilizado en tarjetas para jugar juegos tales como "Luz roja, Luz verde", en el que los niños miran representaciones pictóricas de las direcciones y actuar en consecuencia.

De manera similar, los profesores pueden explorar la representación con los niños al comparar diferentes sistemas de números culturales. En Estados Unidos, los niños pueden contar de uno a cinco en una sola mano, donde cada dedo representa un valor de uno. En chino, sin embargo, pueden contar de uno a diez en una mano. Los profesores pueden enseñar a los niños el sistema chino y luego discutir las diferencias entre los dos. El sistema estadounidense se basa únicamente en la cantidad de dedos que se sostienen, pero el sistema chino también utiliza qué dedos se usan, la forma en que se sostiene (curva) y el ángulo (hacia abajo versus hacia arriba). La figura 9.4 muestra ejemplos de representaciones de números.

Figura 9.4: Ejemplo de representar números con los dedos



Ejemplo computacional

En el mundo digital, los científicos de computación usan representaciones para “comunicarse” con las computadoras. Un niño puede ver una representación demostrada como íconos de aplicaciones en un teléfono inteligente, donde cada ícono representa una aplicación diferente. Otros "botones" son visibles en el mundo de la computación. Por ejemplo, los interruptores de encendido/apagado y los tableros de instrumentos digitales en automóviles son ejemplos de representaciones de información en el mundo de la computación. Los profesores pueden involucrar a los niños en la exploración de los diferentes tipos de representaciones en los dispositivos computacionales en su salón de clases. Además, las computadoras usan representaciones para funcionar más eficientemente. Por ejemplo, las computadoras representan colores con valores numéricos. Usando una aplicación simple de procesamiento de texto, dibujo o edición de fotos en un dispositivo computacional, los niños pueden jugar con los valores numéricos RGB (rojo, verde, azul)

de un color. Los profesores pueden aprender sobre los andamios que señalan cómo las computadoras representan los colores con números y cómo cambian los valores de los números, el color que aparece en la pantalla cambia (Ver la Figura 9.5).

Figura 9.5: Valores numéricos que representan colores

5. **Secuenciación**: la secuenciación es el proceso de organizar eventos, ideas y objetos en un orden específico

Los niños a menudo aprenden acerca de la secuencia a través de la alfabetización durante la infancia y las matemáticas. Por ejemplo, los niños aprenden que las historias siguen una secuencia (principio, medio, final). Del mismo modo, la secuenciación se explora a través de números ordinales (primero, segundo, tercero), así como el tamaño y la magnitud (de menor a mayor). En ciencias de la computación, la secuenciación es una base importante para los algoritmos, que son conjuntos precisos de instrucciones que las computadoras siguen para realizar una tarea específica. Es fundamental que las personas den instrucciones en la secuencia correcta porque las computadoras hacen exactamente lo que están programadas para hacer; Si las instrucciones no se secuencian correctamente, el algoritmo no logrará el resultado deseado.

Conexiones al marco: P3. Reconocimiento y definición de problemas computacionales, P4. Desarrollo y uso de abstracciones y P5.Creación de artefactos computacionales

El aprendizaje sobre la secuenciación en los primeros años puede construir una base para el aprendizaje de uno de los cinco conceptos core del marco, algoritmos y programación: ideas clave en la resolución de problemas computacionales, la abstracción y la creación de artefactos. Por ejemplo, entender que las instrucciones siguen una secuencia específica establece la base para que los niños puedan descomponer (o descomponer) problemas complejos en pasos más pequeños que, si se siguen secuencialmente, resolverán el problema.

Ejemplo cotidiano

A los niños les encanta contar historias; hablan sobre lo que sucedió durante el fin de semana, sobre eventos familiares y sobre diferentes acontecimientos en el aula. Cada una de estas historias se puede dividir en una secuencia de actividades. Los profesores pueden hacer preguntas específicas para ayudar a los niños a ampliar estas ideas. Por ejemplo, una profesora podría preguntar: "¿Qué sucedió por última vez?", "¿Qué pasó primero?" Y "¿Qué sucedió en el medio?"

Otra forma de extender la idea de secuenciar es que los profesores pidan a los niños que den instrucciones para una tarea diaria, como vestirse. Los profesores pueden pedir a los niños que ordenen las actividades en lo que hacen primero, segundo y tercero mientras se visten para el día. También pueden hacer que los niños dibujen el resultado de secuencias específicas para mostrar cómo el orden de los eventos en una secuencia puede hacer una gran diferencia (por ejemplo, ponerse los calcetines después de los zapatos, tomar una ducha después de vestirse). Alternativamente, los profesores pueden hacer que los estudiantes participen en una actividad de secuenciación de historias en la que los niños deben colocar una serie de imágenes en el orden correcto para que la historia tenga sentido. La figura 9.6 muestra una secuencia para hacer una hamburguesa con queso.

Figura 9.6: Secuencia de pasos para hacer una hamburguesa con queso

Ejemplo computacional

Una secuencia de tareas puede explicarse en el contexto del uso de una herramienta digital. Por ejemplo, un profesor puede explicar que los alimentos se escanean en el supermercado y así se indica el precio. La secuencia es (1) se escanea un alimento y (2) el precio se indica en la caja registradora. Esta entrada/salida de información digital desde el escaneo del artículo hasta la visualización del precio en el registro puede ser ejecutada por los niños a través del juego de simulación.

Los entornos de programación basados en bloques emplean interfaces táctiles y comandos reducidos para hacer que la programación sea accesible.

Los estudiantes que están preparados para el desarrollo pueden usar entornos de programación simples basados en bloques para crear algoritmos y programas simples compuestos de secuencias de comandos. Estos entornos permiten a los estudiantes crear programas sin el obstáculo de la escritura que se encuentra en los idiomas tradicionales basados en texto. A menudo emplean interfaces táctiles y conjuntos de comandos reducidos para hacer que la programación sea accesible para los jóvenes estudiantes. Los bloques visuales son representaciones de los comandos que una computadora sigue para ejecutar programas como animaciones (Strawhacker & Bers, 2014). También hay entornos robóticos creados para estudiantes de preescolar que usan bloques de madera tangibles para crear conjuntos de comandos que el robot puede leer para moverse, hacer sonidos y parpadear luces (Elkin, Sullivan, & Bers, 2014).

De las ideas poderosas de Pre-K al marco K–12

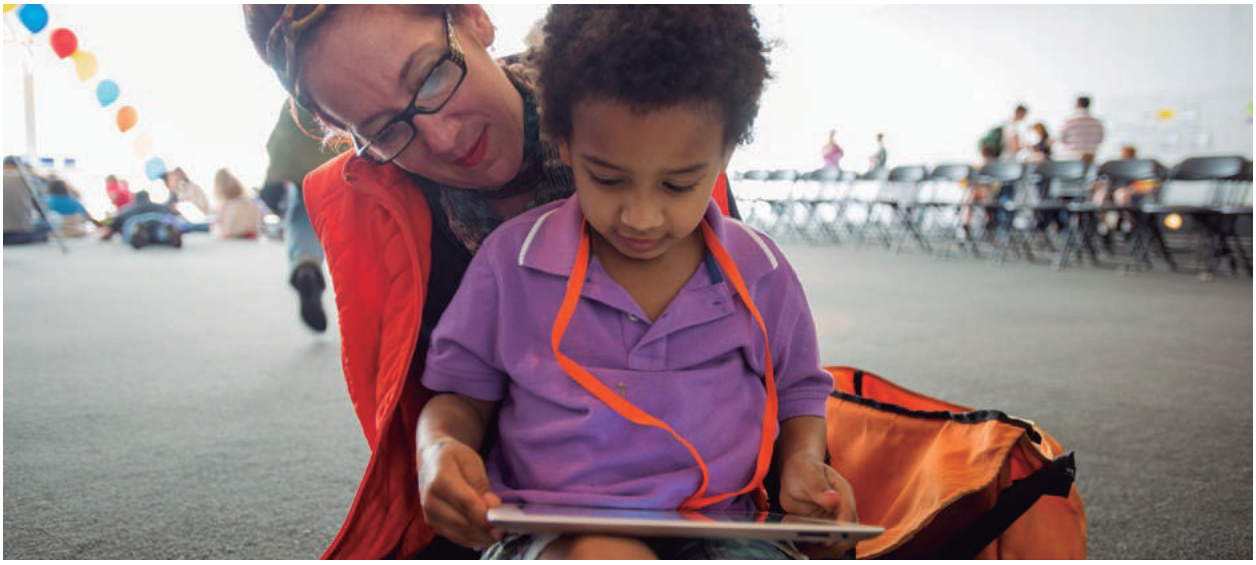
La importancia de la educación de alta calidad durante la infancia no puede ser exagerada, y al igual que los niños participan en actividades de alfabetización temprana, matemáticas y ciencias, también pueden participar en el aprendizaje de las ciencias de las computación fundamentales. A pesar del reciente impulso para hacer que los entornos de aprendizaje temprano sean más académicos (NRC, 2009), las ciencias de la computación proporcionan herramientas para desarrollar más que habilidades técnicas y conocimiento del contenido, y puede integrarse en prácticas de aprendizaje temprano basadas en el juego y apropiadas para el desarrollo (Copple & Bredekamp, 2009; Bers et al., 2002). Como explicó Resnick (2003), las ciencias de la computación son muy adecuadas para la educación durante la infancia, ya que ofrece un entorno de aprendizaje donde los niños pequeños pueden "jugar para aprender mientras aprenden a jugar" (Bers, Flannery, Kazakoff y Sullivan, 2014, pág. 146). Además, el entorno de aprendizaje temprano a menudo se caracteriza por actividades abiertas y basadas en proyectos que atraviesan áreas de contenido, y el aprendizaje suele ser un proceso continuo durante el día, en lugar de agruparse en segmentos de 30 o 60 minutos (Copple y Bredekamp, 2009). Este contexto interdisciplinario brinda la oportunidad única de integrar la informática en otros dominios de las asignaturas, así como el uso de las ciencias de la computación como vehículos para el aprendizaje interdisciplinario (Bers & Horn, 2010; Morgado, Cruz, & Kahn, 2010).

Las ciencias de la computación están bien adaptadas para la educación durante la infancia, ya que ofrece un entorno de aprendizaje donde los niños pequeños pueden "jugar para aprender mientras aprenden a jugar" (Bers et al., 2014, p. 146).

Si bien se ha realizado algún trabajo con niños en edad preescolar, la mayoría de los estudios aún se centran en estudiantes de edad preescolar o mayores, y pocos estudios brindan información sobre cómo los educadores pueden implementar las ciencias de la computación en sus prácticas de aula durante la infancia (consulte DevTech Research Group, 2016 y Morgado et al., 2010 para excepciones). Como explican Fessakis, Gouli y Mayroudi (2013), "[No] no es la disponibilidad de entornos de programación computacional apropiados para el desarrollo, sino más bien el desarrollo de actividades de aprendizaje diseñadas de manera apropiada y material de apoyo que se habría aplicado y verificado y podría ser fácilmente integrado en la práctica escolar diaria por profesores bien informados y preparados "(p. 90). Como tal, este capítulo comenzó la tarea de articular formas específicas, basadas en la investigación, en las cuales las ciencias de la computación pueden integrarse en el entorno de aprendizaje durante la infancia. En el **Apéndice D** puede encontrarse una revisión de la investigación en educación durante la infancia relacionada con las ciencias de la computación.

Como lo sugieren los ejemplos presentados aquí, la integración de las prácticas relacionadas con las ciencias de la computación en la educación durante la infancia no es una desviación de las nociones tradicionales de práctica apropiada para el desarrollo; más bien, las prácticas computacionales tempranas apoyan la pedagogía basada en el juego y amplían lo que los educadores ya están haciendo en sus aulas, y pueden guiar a los jóvenes a darse cuenta, nombrar y reconocer cómo las ciencias de la computación dan forma al mundo moderno. De esta manera, las ciencias de la computación en pre-K dan vida a la disciplina de las ciencias de la computación, que las ciencias de la computación K–12 se expande en el marco más amplio. De hecho, las conexiones del marco descritas en cada poderosa idea en este capítulo explicitan cómo estas ideas relacionadas con las ciencias de la computación exploradas durante la infancia construyen la base para participar en conceptos y prácticas más específicas que se describen en el marco.

Al comprender las ciencias de la computación como una disciplina, los profesores pueden dividir el campo en lecciones manejables y hacer que el mundo computacional sea "visible" para los estudiantes (Welch & Dooley, 2013; Dooley & Welch, 2015). Como tales, los educadores proporcionan una manera para que los niños se vuelvan participantes activos en sociedades digitales, que en la última instancia los posicionarán mejor para convertirse en pensadores, creadores y líderes en nuestro mundo cada vez más digital.



Referencias

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. In I. R. Berson & M. J. Berson (Eds.), *High-tech tots: Childhood in a digital world* (pp. 49–70). Greenwich, CT: Information Age Publishing.

Bers, M.U., Flannery, L.P., Kazakoff, E.R, & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157.

Bers, M. U., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics into early childhood education. *Information Technology in Childhood Education Annual, 2002*(1), 123–145.

Boston Public Schools. (2016). *Computer science in BPS*. Retrieved from http://www.bostonpublicschools.org/domain/2054

Camilli, G., Vargas, S., Ryan, S., & Barnett, W. S. (2010). Meta-analysis of the effects of early education interventions on cognitive and social development. *The Teachers College Record*, *112*, 579–620.

Campbell, F. A., Ramey, C. T., Pungello, E., Sparling, J., & Miller-Johnson, S. (2002). Early childhood education: Young adult outcomes from the Abecedarian Project. *Applied Developmental Science*, *6*, 42–57.

Chetty, R., Friedman, J. N., Hilger, N., Saez, E., Schanzenbach, D., & Yagan, D. (2011). How does your kindergarten classroom affect your earnings? Evidence from Project STAR. *The Quarterly Journal of Economics*, *126*(4), 1593–1660. doi: 10.1093/qje/qjr041

Collaborative for Academic, Social, and Emotional Learning. (2012). *Effective social and emotional learning programs—preschool and elementary school education.* Chicago, IL: Author. Retrieved from http://static1.squarespace.com/static/513f79f9e4b05ce7b70e9673/t/526a220de4b00a92c90436ba/1382687245993/2013-casel-guide.pdf

Copple, C., & Bredekamp, S. (2009). *Developmentally appropriate practice in early childhood programs serving children from birth through age 8*. Washington, DC: National Association for the Education of Young Children.

DevTech Research Group. (2016). *Early childhood robotics curriculum*. Medford, MA: Tufts University DevTech Research Group. Retrieved from http://tkroboticsnetwork.ning.com/page/robotics-curriculum

Dillenbourg, P. (1999). *Collaborative learning: Cognitive and computational approaches. Advances in learning and instruction series*. New York, NY: Elsevier Science, Inc.

Dooley, C. M., & Welch, M. M. (2015). Emergent comprehension in a digital world. In A. DeBruin-Parecki & S. Gear (Eds.), *Developing early comprehension: Laying the foundation for reading success*. Baltimore, MD: Brookes.

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice, 13*, 153–169.

Fessakis, G., Gouli, E., & Mayroudi, E. (2013). Problem solving by 5-6 year old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Goodyear, P., Jones, C., & Thompson, K. (2014). Computer-supported collaborative learning: Instructional approaches, group processes, and educational designs. In J. M. Spector, M. D. Merrill, J. Elen, & M. J. Bishop (Eds.), *Handbook of research on educational communications and technology* (pp. 439–451). New York, NY: Springer Science and Business Media.

Heckman, J. (2006). Skill formation and the economics of investing in disadvantaged children. *Science, 312*, 1900–1902.

Heckman, J., & Masterov, D. (2007). The productivity argument for investing in young children. *Review of Agricultural Economics, 29*, 446–493.

Klem, A. M., & Connell, J. P. (2004). Relationships matter: Linking teacher support to student engagement and achievement. *Journal of School Health, 74*(7), 262–273.

Magnuson, K., Meyers, M. K., Ruhm, C. J., & Waldfogel, J. (2004). Inequality in preschool education and school readiness. *American Educational Research Journal, 41*, 115–157.

Maryland State Department of Education. (2015). *Computer science*. Retrieved from http://archives.marylandpublicschools.org/MSDE/divisions/dccr/cs.html

Mashburn, A. J., Pianta, R. C., Hamre, B. K., Downer, J. T., Barbarin, O. A., Bryant, M., . . . & Howes, C. (2008). Measures of classroom quality in prekindergarten and children's development of academic, language, and social skills. *Child Development, 79*(3), 832–749. doi: 10.1111/j.1467-8624.2008.01154.x

Morgado, L., Cruz, M., & Kahn K. (2010). Preschool cookbook of computer programming topics. *Australasian Journal of Educational Technology, 26*(3), 309–326.

National Research Council. (2009). *Mathematics learning in early childhood: Paths toward excellence and equity.* Committee on Early Childhood Mathematics. C. T. Cross, T. A. Woods, & H. Schweingruber (Eds.). Center for Education. Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

National Science Board. (2012). *Science and engineering indicators 2012* (NSB 12-01). Arlington, VA: National Science Foundation. Retrieved from https://www.nsf.gov/statistics/seind12/pdf/seind12.pdf

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NY: Basic Books.

Pianta, R. C., Barnett, W. S., Burchinal, M., & Thornberg, K. R. (2011). The effect of preschool education: What we know, how public policy is or is not aligned with the evidence base, and what we need to know. *Psychological Sciences in the Public Interest, 10*(2), 49–88. doi: 10.1177/1529100610381908

Pianta, R. C., La Paro, K. M., Payne, C., Cox, M. J., & Bradley, R. (2002). The relation of kindergarten classroom environment to teacher, family, and school characteristics and child outcomes. *The Elementary School Journal, 102*(3), 225–238.

Resnick, M. (2003), Playful learning and creative societies. *Education Update, 8*(6). Retrieved May 1, 2009 from http://web.media.mit.edu/~mres/papers/education-update.pdf

Schweinhart, L. J., Montie, J., Xiang, Z., Barnett, W. S., Belfield, C. R., & Nores, M. (2011). *Lifetime effects: The High/Scope Perry preschool study through age 40: Summary, conclusions, and frequently asked questions.* Ypsilanti, MI: High/Scope Press. Retrieved from http://www.highscope.org/file/Research/PerryProject/specialsummary_rev2011_02_2.pdf

Smith, M. (2016). *Computer science for all*. Washington, DC: Office of Science and Technology Policy, Executive Office of the President. Retrieved from https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

Strawhacker, A. L., & Bers, M. U. (2014, August). *ScratchJr: Computer programming in early childhood education as a pathway to academic readiness and success.* Poster presented at DR K–12 PI Meeting, Washington, DC.

Twarek, B. (2015). *Pre-K to 12 computer science scope and sequence.* Retrieved from http://www.csinsf.org/curriculum.html

U.S. Congress Joint Economic Committee. (2012). *STEM education: Preparing for the jobs of the future*. Washington, DC: Author. Retrieved from http://www.jec.senate.gov/public/_cache/files/6aaa7e1f-9586-47be-82e7-326f47658320/stem-education---preparing-for-the-jobs-of-the-future-.pdf

U.S. President's Council of Advisors on Science and Technology. (2010). *Prepare and inspire: K–12 education in science, technology, engineering, and math (STEM) for America's future*. Washington, DC: Executive Office of the President. Retrieved from https://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-stemed-report.pdf

Weissberg, R. P., & Cascarino, J. (2013). Academic + social-emotional learning = national priority. *Phi Delta Kappan, 95*(2), 8–13.

Weissberg, R. P., Durkak, J. A., Doitrovich, C. E., & Gullota, T. P. (2015). Social emotional learning: Past, present, and future. In J. A. Durlak, C. E. Domitrovich, R. P. Weissberg, & T. P. Gullota (Eds.), *Handbook of social and emotional learning: Research and practice* (pp. 3–19). New York, NY: Guilford.

Welch, M. M., & Dooley, C. M. (2013, May). Digital equity for young children: A question of participation. *Learning and Leading with Technology,* 28–29.

Yoshikawa, H., Weiland, C., Brooks-Gunn, J., Burchinal, M. R., Espinosa, L. M., Gormley, W. T., . . . & Zazlow, M. J. (2013). *Investing in our future: The evidence base on preschool education*. Ann Arbor, MI: Society for Research in Child Development; New York, NY: Foundation for Child Development.

El Rol de la Investigación en el Desarrollo y Futuro del Marco

10

El Rol de la Investigación en el Desarrollo y Futuro del Marco

El Marco de Ciencias de la Computación K–12 fue informado por un cuerpo creciente de investigación sobre educación en ciencias de la computación, así como por una literatura más amplia de los campos de la educación en ciencias, tecnología, ingeniería y matemáticas (STEM). En particular, los conceptos, prácticas y progresiones de aprendizaje en el corazón del marco fueron influenciados por la investigación sobre temas tales como la forma en que los estudiantes aprenden las ciencias de la computación, cómo interactúan entre ellos en entornos computacionales y a qué edad demuestran dominio en un tema de conceptos específicos.

El marco fue informado por un cuerpo creciente de investigación sobre educación en informática.

Como la educación en ciencias de la computación es un campo joven en relación con otras áreas temáticas de K-12, la investigación educativa que la acompaña tiene una serie de áreas en las cuales mejorar, y quedan preguntas importantes por abordar (Lishinski, Good, Sands y Yadav, 2016). La gran demanda pública de oportunidades de aprendizaje en ciencias de la computación y la escasez de investigadores capacitados en educación de las ciencias de la computación han llevado a que los profesionales realicen muchas investigaciones (es decir, profesores de ciencias de la computación), si es que la investigación se realiza (Franklin, 2015). Además, los rápidos avances tecnológicos de las últimas décadas plantean otro desafío: la investigación es incapaz de mantenerse al día con las nuevas tecnologías y las formas cada vez más diversas en las que podemos interactuar con ellas. De hecho, la literatura más amplia sobre tecnología en educación ha pasado de centrarse en el medio tecnológico a una visión sistémica del contenido de los medios,

contexto y usuarios en un esfuerzo por reconocer y comprender las complejas interacciones que ofrece nuevas tecnologías (por ejemplo, Guernsey, 2012; Valkenburg y Peter, 2013).

Aunque la tecnología continúa cambiando, los fundamentos conceptuales de la educación en ciencias de la computación y los desafíos que enfrentan los educadores y los estudiantes en el campo siguen siendo los mismos. Por ejemplo, el establecimiento de herramientas de programación basadas en bloques no ha eliminado las dificultades conceptuales de aprender y enseñar conceptos de programación. Del mismo modo, conceptos erróneos sobre conceptos específicos, tales como variables, bucles y lógica booleana, que se articularon en una investigación realizada en la década de 1980 (por ejemplo, De Boulay, 1989; Pea y Kurland, 1984; Pea, Soloway y Spohrer, 1987; Soloway, 1986) no disipado en entornos informáticos desarrollados más recientemente (por ejemplo, Cooper, Grover, Guzdial y Simon, 2014; Grover, Pea y Cooper, 2016; Guzdial, 2016).

Los escritores consultaron literatura relevante para informar los primeros borradores de las declaraciones marco. Más tarde, utilizaron un enfoque más estructurado para recopilar e incorporar la investigación en el concepto específico y las declaraciones de práctica, así como para determinar la estructura y la ubicación de tales declaraciones dentro de las progresiones de aprendizaje K-12. Este enfoque implicaba buscar la investigación existente en el campo, entrevistar a expertos en áreas clave de investigación, identificar la investigación relacionada con los conceptos y prácticas centrales en el marco, mapear la investigación a conceptos específicos y declaraciones de práctica, y realizar modificaciones basadas en los resultados de la investigación. Teniendo en cuenta la madurez de las ciencias de la computación como una disciplina de K-12 y la carga de delinear los conceptos y prácticas core en ciencias de la computación K-12, la investigación a menudo era insuficiente o todavía estaba en desarrollo. En esos casos, el equipo de redacción se basó en su propia experiencia y años de experiencia en educación en ciencias de la computación K-12, así como en la experiencia colectiva de las comunidades de educación e investigación en ciencias de la computación.

Los asesores desempeñaron un papel importante en la construcción de la base de investigación utilizada para informar el marco, así como en proporcionar orientación más allá de la investigación disponible. Como líderes en los campos de las ciencias de la computación, la educación y la investigación en educación en ciencias de la computación, los asesores aportaron diversos conocimientos y perspectivas desde una variedad de áreas de investigación, como la diversidad y la equidad, la educación primaria, el aprendizaje colaborativo apoyado por computadora, la interacción humano-computadora, computación social, preparación docente, pensamiento computacional e integración interdisciplinaria. Su trabajo, y el trabajo de sus socios de investigación, se refleja en la base de investigación que informa la visión, la estructura y el contenido del marco.

Este capítulo no pretende ser una revisión completa de toda la investigación relacionada con la educación en ciencias de la computación K–12. Varias revisiones recientes sobre la investigación del pensamiento computacional K-12 (Grover & Pea, 2013) y la educación en computación (Guzdial, 2016) proporcionan información más amplia que está más allá del alcance de este capítulo, pero que se hizo referencia durante el proceso de desarrollo del marco. Más bien, la intención de este capítulo es describir ejemplos de las formas en que se utilizó la investigación para informar las declaraciones del marco y reconocer y ofrecer consideraciones para abordar las brechas en la literatura actual de investigación en educación en ciencias de la computación que surgió a través del proceso de desarrollo del marco. El final de este capítulo incluye consideraciones para examinar las políticas y los problemas de implementación.

Investigación que informa el marco

Las siguientes secciones proporcionan ejemplos de la investigación que conducía el desarrollo de los conceptos y prácticas core del marco, así como las declaraciones de conceptos y prácticas específicas. El Apéndice E contiene una lista completa de referencias a las que los escritores del marco se refirieron en su desarrollo del contenido del marco.

Investigación que informa los conceptos

Las declaraciones del concepto core se organizaron de acuerdo con las progresiones del aprendizaje, hitos conceptuales en un camino que dirige al alumno desde ideas básicas a un conocimiento más sofisticado. Fundamentalmente, las progresiones del aprendizaje se basan en la investigación de la psicología del desarrollo relacionada con los procesos cognitivos, sociales y de comportamiento más amplios que ocurren a lo largo del desarrollo de un niño, así como la estructura lógica de contenido específico dentro de un dominio de materia. De hecho, las progresiones de aprendizaje específicas del contenido basadas en dicha investigación sirven como base para varios conjuntos de estándares estatales de matemáticas y ciencias (NGA Center for Best Practices & CCSSO, 2010; NGSS Lead States, 2013).

Para desarrollar las progresiones en el aprendizaje de las ciencias de la computación, los redactores y asesores del marco identificaron ideas core emergentes que pueden introducirse temprano en la educación de un estudiante y desarrollarse en años posteriores. Si bien es limitado en comparación con otros dominios de materias, se consultó la investigación de la progresión del aprendizaje de las ciencias de la computación cuando estaba disponible.

Incluso cuando existe evidencia para desarrollar progresiones de aprendizaje, los desafíos permanecen con respecto a capturar la compleja relación entre el conocimiento previo de los estudiantes, sus habilidades cognitivas y cambios en el desarrollo a lo largo del tiempo y luego generalizar la misma progresión de aprendizaje a una población diversa de estudiantes (Duschl, Schweingruber y Shouse, 2007). Las progresiones de aprendizaje son especialmente pertinente para la educación en ciencias de la computación, donde los esfuerzos para construir vías coherentes de enseñanza K-12 para todos los estudiantes están a la vanguardia de muchas iniciativas locales y estatales (Cernavskis, 2015). Sugerencias para una mayor investigación para abordar los desafíos en el desarrollo de progresiones de aprendizaje se discuten más adelante en este capítulo.

Para las áreas donde las progresiones de aprendizaje específicas de las ciencias de la computación no existían, los escritores recurrieron a las progresiones de aprendizaje en ciencias y matemáticas para guiar la ubicación de los conceptos de las ciencias de la computación relacionados en una banda de grado en particular. Como se describió anteriormente en el capítulo **Proceso de desarrollo**, la ubicación de la abstracción de procedimientos, en la cual los procedimientos usan variables como parámetros para generalizar el comportamiento, ya que una expectativa para el final del octavo grado fue informada por la colocación de un concepto relacionado en las progresiones del aprendizaje de las matemáticas: la escritura de ecuaciones con variables. La comprensión de los bits (unidades básicas de información digital) se colocó como una expectativa en las bandas de grado de primaria y secundaria basadas en la ubicación de los conceptos análogos de partículas y átomos en las progresiones de aprendizaje de ciencias. En algunos casos, se utilizaron progresiones de aprendizaje de ciencias para informar la ubicación de conceptos muy similares, como modelos y simulaciones. Al igual que en ciencias y matemáticas, los escritores y asesores reconocen la necesidad de que las progresiones del aprendizaje de las ciencias de la computación continúen evolucionando a medida que surgen nuevas investigaciones.

Los escritores recurrieron a las progresiones de aprendizaje en ciencias y matemáticas para guiar la colocación de conceptos relacionados con las ciencias de la computación.

Para aumentar el apoyo a la colocación adecuada de las declaraciones conceptuales, las progresiones del aprendizaje también fueron informadas por las experiencias de los escritores y asesores del marco con estudiantes de K–12 de diversas poblaciones, así como las experiencias del grupo más amplio de educadores e investigadores de K–12 quien revisó el marco. A continuación hay varios ejemplos más de cómo la investigación informó el desarrollo de declaraciones conceptuales.

Ejemplo 1: Algoritmos y programación

La investigación actual sobre pensamiento computacional, desarrollo de algoritmos y programación temprana fue informativa para los enunciados de algoritmos y programación. La investigación sobre los intentos de los alumnos de cuarto grado de desarrollar instrucciones y algoritmos paso a paso sugiere lo que los estudiantes saben antes de la instrucción formal y las oportunidades potenciales para currículos apropiados para el desarrollo (Dwyer, Hill, Carpenter, Harlow y Franklin, 2014). Esta investigación sobre el desarrollo algorítmico temprano es un aspecto importante de esta expectativa al final del Grado 5 en el concepto central de Algoritmos y Programación: "Las personas desarrollan programas utilizando un proceso iterativo que involucra diseño, implementación y revisión. El diseño a menudo implica reutilizar el código existente o remezclar otros programas dentro de una comunidad. La gente revisa continuamente si los programas funcionan como se espera, y reparan o depuran partes que no lo hacen. La repetición de estos pasos permite a las personas refinar y mejorar los programas "(3–5. Algoritmos y programación.Desarrollo de programas).

Investigaciones adicionales informaron la ubicación de la banda de grado de conceptos de programación específicos en el marco. Seiter y Foreman (2013) recopilaron proyectos creados en un entorno de programación basado en bloques de estudiantes de Grados 1 a 6 y evaluaron la demostración de cada proyecto de conceptos de pensamiento computacional. Descubrieron que las acciones de programación asociadas con la representación de datos, como la referencia y asignación de variables, no están significativamente presentes hasta grados posteriores (es decir, grados 5 y 6). Este hallazgo guio la colocación de variables en la banda de 6–8 grados del marco. Seiter y Forma también descubrieron que los usos de la lógica condicional comienzan a aparecer en el Grado 3 y aumentan hasta el Grado 6, lo que guio la ubicación de las declaraciones condicionales en la banda de grados 3–5 del marco.

La metodología utilizada por Seiter y Foreman para analizar los proyectos de los estudiantes evoca preguntas sobre las condiciones bajo las cuales funciona y no funciona. Se requiere investigación futura para validar si al analizar el código del programa de los estudiantes se puede inferir con precisión lo que comprenden. Por ejemplo, Aivaloglou y Hermans (2016) analizaron 250,000 proyectos Scratch y descubrieron que los procedimientos y los bucles condicionales no se usaban comúnmente. Estos resultados no significan que los estudiantes no puedan usar procedimientos o bucles condicionales o que estos conceptos se deban enseñar a una edad posterior; la ausencia de estas construcciones puede deberse al entorno de programación específico utilizado o simplemente al deseo de los estudiantes de usar la herramienta de una manera particular. Una vía prometedora de investigación es usar reflexiones escritas y entrevistas con estudiantes para evaluar artefactos computacionales. Por ejemplo, Brennan y Resnick (2012) han explorado el uso de entrevistas basadas en artefactos para evaluar lo que los estudiantes comprenden y sus razones para las decisiones de programación.

Ejemplo 2: Impactos de las ciencias de la computación

También hay investigaciones sobre la influencia de la cultura en las interacciones de los individuos con la tecnología (por ejemplo, Evers y Day, 1997; Leidner y Kayworth, 2006). Esta investigación informa principalmente los Impactos de las declaraciones de computación en el subconcepto Cultura. Por ejemplo, la siguiente declaración se basa en los hallazgos de Evers y Day (1997) de que la cultura influye en las preferencias de diseño y la aceptación de una interfaz: "El desarrollo y la modificación de la tecnología computacional se basa en las necesidades y deseos de las personas y puede afectar a los grupos de manera diferente". Las tecnologías computacionales influyen, y están influenciadas por, las prácticas culturales "(3-5. Impactos de computación.Cultura). La siguiente declaración en la progresión Cultura se basa en el trabajo de Leidner y Kayworth (2006), cuya revisión de la literatura sobre cultura y tecnología de las ciencias de la computación concluyó que la cultura influye en la información y la tecnología y viceversa: "Los avances en la tecnología computacional cambian las actividades cotidianas de las personas. La sociedad se enfrenta a compensaciones debido a la creciente globalización y automatización que brinda las ciencias de la computación"(6–8. Impactos de las ciencias de la computación. Cultura).

En cada uno de estos ejemplos, se debe tener en cuenta que esta investigación se relaciona e informa los conceptos en el marco, pero que se necesita más investigación para enfocarse específicamente en la adecuación del desarrollo de los conceptos dentro del contexto de una progresión de aprendizaje.

Investigación que informa las prácticas

Los estudios de investigación clave proporcionaron una base para la identificación de las prácticas core en el marco, así como las declaraciones de práctica en cada práctica core. Weintrop y col. (2015) definen una taxonomía de pensamiento computacional para la ciencia y las matemáticas que consta de prácticas en cuatro categorías principales: datos, modelado y simulación, resolución de problemas computacionales y pensamiento sistémico. Esta investigación fue útil para abordar el objetivo del marco de empoderar a los estudiantes para que aprendan, se desempeñen y se expresen en otros campos e intereses al ayudar a identificar prácticas computacionales que tuvieron aplicación dentro y más allá de las ciencias de la computación. Otro informe que influyó en el conjunto de prácticas básicas en el marco fue Patrones de diseño de evaluación para prácticas de pensamiento computacional en

las ciencias de la computación secundaria (Bienkowski, Snow, Rutstein y Grover, 2015). En este informe, Bienkowski et al., describe su enfoque para diseñar evaluaciones para medir las habilidades de pensamiento computacional de los estudiantes. Este trabajo incluyó la creación de un dominio de prueba que consiste en construcciones clave, como "Diseñar y aplicar abstracciones y modelos" y "Colaborar con compañeros en actividades computacionales" (p. 10), muchas de las cuales inspiraron y se superponen con las prácticas del marco.

Los siguientes párrafos proporcionan algunos ejemplos de investigación que informaron prácticas core específicas y declaraciones prácticas. Cabe señalar que aunque estos estudios informaron las prácticas en el marco, algunos requieren investigación adicional para proporcionar apoyo directo dentro de un contexto K-12.

La investigación centrada en estudiantes tradicionalmente subrepresentados en ciencias de la computación, como estudiantes con discapacidades, mujeres y estudiantes de algunos grupos minoritarios, influyó en la identificación de la primera práctica, Fomentar una cultura de computación inclusiva. Ladner e Israel (2016) abogan por consideraciones para incluir a todos los estudiantes en ciencias de la computación y describen los desafíos que incluyen la necesidad de una pedagogía culturalmente relevante y una mayor relevancia para los estudiantes de las ciencias de la computación no tradicionales. Otro trabajo identifica estrategias y recursos que los profesores pueden integrar en las lecciones para alentar a un conjunto diverso de estudiantes a participar y aprender (Israel, Wherfel, Pearson, Shehab y Tapia, 2015). Aunque no está en el contexto del aula, la investigación en el lugar de trabajo sobre la mayor efectividad de los grupos culturalmente diversos en comparación con los grupos culturalmente homogéneos (por ejemplo, Watson, Kumar y Michaelsen, 1993) informaron esta declaración práctica: "Incluya las perspectivas únicas de los demás y reflexione sobre las propias perspectivas al diseñar y desarrollar productos computacionales ”(P1.Fomentar una cultura de computación inclusiva.1).

La investigación también sugiere que la autodeterminación (o autodefensa) es de importancia crítica para los estudiantes con discapacidades (Wehmeyer, 2015; Wehmeyer et al., 2012) y se refleja en la tercera declaración práctica, "Emplear la autodefensa y la defensa de los pares para abordar el sesgo en las interacciones, el diseño del producto y los métodos de desarrollo ”(P1.Fomentar una cultura de computación inclusiva.3). Investigaciones específicas relacionadas con la colaboración informaron las declaraciones en Collaborating Around Computing. Por ejemplo, la investigación mencionada anteriormente encontró que los equipos culturalmente diversos en el lugar de trabajo son más efectivos para identificar perspectivas de problemas y generar soluciones alternativas que los equipos homogéneos (Watson, Kumar y Michaelsen, 1993). Esta investigación influyó en la inclusión de esta declaración: "Cultive relaciones de trabajo con personas que posean diversas perspectivas, habilidades y personalidades" (P2. Colaboración en torno a la computación.1). Aunque la investigación apunta a que los estudiantes de diferentes habilidades aprenden más en niveles similares o de habilidades mixtas (es decir, los estudiantes de menor capacidad aprenden más en grupos de habilidades mixtas) (Lou, Abrami y d'Apollonia, 2001), Collaborating Around Computing trata sobre grupos con diversas habilidades, en lugar de habilidades. La investigación a pequeña escala sobre la programación de parejas, una técnica pedagógica para grupos que trabajan en la misma computadora, sugiere que "las parejas menos equitativas buscaron completar las tareas rápidamente y esto puede haber conducido a patrones de marginación y dominación" y que "estos hallazgos son importantes para comprender los mecanismos de inequidad y diseñar prácticas de colaboración equitativas en las ciencias de la computación”(Lewis & Shah, 2015, p. 41). Esta investigación informó la siguiente declaración práctica: "Crear estándares del equipo, expectativas y cargas de trabajo equitativas para aumentar

eficiencia y efectividad "(P2. Colaboración en torno a la computación.2). Se requiere más investigación para estudiar el efecto de esta práctica en el aprendizaje de la informática, además de la colaboración.

Existe una gran cantidad de literatura sobre la creación de artefactos computacionales por parte de los estudiantes, como juegos, aplicaciones y animaciones. Esta investigación proporciona algunos de los fundamentos de las declaraciones en Creación de artefactos computacionales. Por ejemplo, un estudio de los proyectos de programación de los estudiantes en el entorno Scratch sugiere que el aprendizaje y el pensamiento computacional se apoyan mediante la remezcla de partes de artefactos existentes (Dasgupta, Hale, Monroy-Hernández y Hill, 2016). Otro trabajo sugiere que una progresión de tres etapas que ayuda a los estudiantes a avanzar desde el rol de usuario a modificador a creador de artefactos computacionales también puede construir un pensamiento computacional (Lee et al., 2011). Estos ejemplos de investigación informan el acto de remezclar e informan la progresión del aprendizaje para esta declaración práctica: "Modificar un artefacto existente para mejorarlo o personalizarlo" (P5. Creación de artefactos computacionales.3).

Literatura adicional en el campo proporciona orientación general para otras prácticas core. Por ejemplo, los argumentos sobre los beneficios y la importancia de la abstracción de la enseñanza y el aprendizaje (por ejemplo, Sprague y Schahczenski, 2002) generalmente informan el desarrollo y el uso de las abstracciones como una práctica core. La segunda declaración práctica en Reconocimiento y definición de problemas computacionales, "Descomponer problemas complejos del mundo real en subproblemas manejables que podrían integrar soluciones o procedimientos existentes", se basa en los beneficios de desglosar un programa y etiquetar secciones con submetas (Morrison, Margulieux, & Guzdial, 2015). La investigación sobre estudiantes universitarios en clases de programación introductoria, la mayoría de los cuales eran estudiantes de ciencias de la computación e ingeniería, muestra resultados positivos asociados con los programas de prueba al inicio del proceso de desarrollo (Buffardi y Edwards, 2013) e informa la práctica de Pruebas y refinación de artefactos computacionales. Se requiere más investigación para validar este resultado con estudiantes de K–12 en una población general.

Equity and Ac

Agenda de Investigación

Una visión de la educación en ciencias de la computación, a la escala, rigor y nivel de coherencia que promueve el marco, es una nueva frontera para el sistema educativo estadounidense. Se requerirá investigación para evaluar la implementación de las ciencias de la computación K-12 y desarrollar los objetivos y el alcance de la educación en ciencias de la computación para reflejar las lecciones aprendidas de los primeros años de implementación. La investigación adicional iluminará formas más efectivas para que los estudiantes de todas las edades aprendan las ciencias de la computación y para que los profesores brinden instrucción y brinden experiencias significativas para todos los estudiantes.

La necesidad de investigación sobre educación en ciencias de la computación K–12 ofrece una oportunidad única para los investigadores en educación de las ciencias de la computación y ciencias del aprendizaje. Las siguientes secciones describen una agenda de investigación sugerida que se guía por áreas identificadas en el proceso de desarrollo del marco que carecen de una base sólida de investigación o que requieren más estudio: equidad y acceso, progresiones de aprendizaje, conocimiento del contenido pedagógico y facilitación del aprendizaje en otras disciplinas. Estas categorías no están solas; La investigación en estas áreas puede y debe superponerse y debe explorarse en conjunto para avanzar en la educación en ciencias de la computación en su conjunto.

El diseño de la educación en computación centrado en el alumno (Guzdial, 2016) proporciona una revisión de la investigación de temas tal como cómo se enseña actualmente las ciencias de la computación, por qué debe enseñarse y los desafíos para enseñarla a todos los estudiantes. El capítulo siete incluye preguntas adicionales de investigación en ciencias de la computación K-12 relacionadas con la escuela primaria y secundaria.

El objetivo principal de las siguientes consideraciones de investigación es informar futuras iteraciones del marco actual, que, a su vez, puede informar el campo más amplio de la educación en ciencias de la computación K–12. Las sugerencias se guían por la visión del marco de las ciencias de la computación K-12 y adoptan intencionalmente una lente de política y práctica en un esfuerzo por garantizar que la investigación futura tenga implicaciones prácticas en el mundo real que impulsen la enseñanza y el aprendizaje de la educación en ciencias de la computación en el aula.

Equidad y acceso

La intención del marco es proporcionar una base de las ciencias de la computación para todos los estudiantes, independientemente de sus antecedentes, desafíos físicos, diferencias de aprendizaje o aspiraciones profesionales futuras. A pesar de la creciente demanda de personas con habilidades computacionales (BLS, 2015) y un mayor enfoque en lograr una cohorte más diversa de las ciencias de la computación (Sullivan, 2014), el campo sigue siendo predominantemente masculino y desproporcionadamente blanco y asiático (Marcus, 2015). De hecho, de todos los campos de STEM las ciencias de la computación consistentemente tiene las mayores disparidades de género, y las mujeres representan solo el 23% de la industria computacional, una cifra que no ha cambiado durante más de una década (BLS, 2015; NSF, 2014). Del mismo modo, las personas negras e hispanas representan solo el 14% del campo de la computación, y es menos probable que tengan acceso a cursos y tecnología de computación. (Google & Gallup, 2015b; Margolis, Estrella, Goode, Holme, & Nao, 2010).

El trabajo previo sobre equidad y acceso en ciencias de la computación K–12 ha abordado el tema desde una variedad de ángulos y proporciona una visión inicial para construir en futuros esfuerzos de investigación. Los proyectos han explorado la comunicación interpersonal (Lewis y Shah, 2015), los problemas sistémicos y el privilegio preparatorio dentro de las escuelas (Margolis et al., 2010), y el desarrollo de un currículo culturalmente relevante y la evaluación de su efectividad para mitigar la inequidad (Margolis et al., 2012). Sobre la base de una revisión de la literatura sobre las disparidades de género en STEM, Blickenstaff (2005) sugirió siete formas prácticas para que los educadores, administradores y desarrolladores de planes de estudio creen entornos de aprendizaje más positivos para todos los estudiantes. Más allá de proporcionar un acceso más equitativo a los cursos y recursos, Blickenstaff (2005) sugiere crear materiales curriculares que enfaticen el impacto en el mundo real de STEM, incluyendo cómo los campos pueden ayudar a mejorar la calidad de vida y ayudar a resolver problemas sociales. Otros han señalado el papel fundamental que juegan los estereotipos sociales para disuadir a las niñas y estudiantes de color de ingresar al campo de las ciencias de la computación (por ejemplo, Goode, Estrella y Margolis, 2006; Google y Gallup, 2015a).

A menos que la educación en ciencias de la computación K–12 crezca de manera equitativa, hay pocas razones para creer que se responderá al llamado a una mayor diversidad en el campo. Para asegurar la equidad en la educación en ciencias de la computación y las industrias relacionadas con la tecnología, el campo de investigación puede trabajar hacia una comprensión basada en la evidencia de los factores que respaldan entornos computacionales inclusivos, así como soluciones prácticas para aumentar el acceso, el interés y la eficacia en poblaciones de estudiantes tradicionalmente subrepresentadas.

Los investigadores sugieren crear un currículo que enfatice el impacto del STEM en el mundo real, incluyendo cómo los campos pueden ayudar a mejorar la calidad de vida y ayudar a resolver problemas sociales.

En el nivel más básico, los investigadores pueden rastrear no solo los números generales de inscripción en cursos de las ciencias de la computación K-12, sino también la diversidad de participación, incluidas las diferencias en el acceso y la participación de estudiantes de grupos tradicionalmente subrepresentados. Este trabajo se volverá aún más crítico a medida que los esfuerzos de implementación estatales y distritales se amplíen, abriendo la posibilidad muy real de intenciones no coincidentes (es decir, las ciencias de la computación para todos) y la realidad (es decir, las ciencias de la computación para unos pocos seleccionados). Los investigadores pueden profundizar en la comprensión del papel de los profesores y los administradores escolares, así como las características de los contextos escolares, distritales y estatales, para comprender mejor cómo las características y disposiciones individuales (por ejemplo, el nivel socioeconómico, la raza/etnia, género, lenguaje hablado, autoeficacia, experiencia previa, etc.) se cruzan con estos contextos relacionales e institucionales para dar forma a la experiencia, el éxito y la participación sostenida de los estudiantes en las ciencias de la computación. Este trabajo incluye el estudio de las experiencias en el aula de estudiantes de grupos subrepresentados; posibles diferencias en los niveles de interés, autoeficacia y logros; y la efectividad de los enfoques destinados a abordar de manera proactiva tales brechas.

Al monitorear las tendencias de educación en ciencias de la computación de K-12 ahora, los investigadores preparan el escenario para estudios longitudinales que pueden proporcionar una idea de las implicaciones a largo plazo de las experiencias de las ciencias de la computación de los estudiantes, así como una mejor comprensión de las diferencias entre aquellos que eligen estudiar las ciencias de la computación, ciencia y aquellos que toman otros caminos académicos y profesionales. Dichos datos a largo plazo serán críticos para comprender los factores más influyentes, como el acceso a las oportunidades computacionales, las experiencias de los estudiantes y el interés y la autoeficacia en las ciencias de la computación, para involucrar y mantener una industria computacional diversa. Se puede encontrar más información sobre equidad y acceso en el capítulo **Equidad en la educación en ciencias de la computación**.

Progresiones de aprendizaje

Las progresiones de aprendizaje proporcionan una estructura organizativa al marco y son fundamentales para asegurar la coherencia a lo largo de una vía de educación en ciencias de la computación. Las progresiones del aprendizaje se han desarrollado en la investigación en educación en ciencias y matemáticas (por ejemplo, Achieve, 2015), pero tienen espacio para crecer en la educación en ciencias de la computación (Seiter y Foreman, 2013). La investigación de la progresión del aprendizaje es crucial para informar versiones futuras del marco e influir en el aprendizaje del alumno y las prácticas docentes. Los estudiantes se ven directamente afectados por progresiones de aprendizaje incompletas e inapropiadas para el desarrollo, ya que podrían conducir a una transferencia de conocimiento deficiente (Perkins y Salomon, 1988) o al desarrollo de conceptos erróneos. Como se sugiere en llevar la ciencia a la escuela, "las ideas bien probadas sobre las progresiones del aprendizaje podrían proporcionar una guía muy necesaria tanto para el diseño de secuencias de instrucción como para evaluaciones a gran escala y en el aula" (NRC, 2007, p. 220).

Es importante destacar que el desarrollo de las progresiones de aprendizaje no implica simplemente disminuir el rigor del contenido y los procesos que se consideran apropiados para los estudiantes mayores y llevarlo a niveles de grado más jóvenes. Además, no se puede suponer que lo que funciona en la escuela secundaria o en contextos postsecundarios funcionará, y es apropiado para el desarrollo, para el contexto de la escuela primaria. De hecho, los escritores del marco dedicaron una atención crítica para asegurar que las progresiones tuvieran tanto sentido lógico en términos de contenido de las ciencias de la computación como sentido de desarrollo para que coincida con los diferentes niveles de estudiantes en el rango de grado K-12. Por lo tanto, si bien las investigaciones previas sobre el entorno de educación superior en ciencias de la computación pueden informar los estudios actuales y futuros en el contexto K – 12, simplemente replicar este trabajo es insuficiente para el desarrollo de progresiones sólidas de aprendizaje de las ciencias de la computación en el nivel K – 12.

En el entorno de K–12, la investigación sobre las progresiones del aprendizaje de las ciencias de la computación acaba de comenzar, y gran parte del trabajo actual se centra en prácticas de pensamiento computacional amplias o conceptos computacionales muy específicos. Por ejemplo, el trabajo de Dwyer y colegas (2014) investigó el conocimiento previo de los alumnos de cuarto grado antes de participar en un plan de estudios de pensamiento computacional para construir puntos de anclaje iniciales para progresiones de aprendizaje elemental. Los estudios de seguimiento examinaron el uso de estructuras de control interactivas de los estudiantes en un lenguaje de programación basado en bloques y sugirieron que la instrucción explícita de los elementos de programación relacionados con el diseño centrado en el usuario (por ejemplo, el manejo de eventos como clics del mouse) debe esperar hasta el quinto grado (Hansen et al., 2015; Hansen, Iveland, Carlin, Harlow y Franklin, 2016). Cabe señalar que estos resultados están altamente contextualizados al entorno de programación, ya que los resultados pueden variar con los diferentes entornos. Este hallazgo también contrasta con la investigación sobre la creación de juegos y programas digitales para niños, que ilustra que los niños pueden participar en procesos de diseño centrados en el usuario junto con el aprendizaje de contenidos y prácticas computacionales novedosas (por ejemplo, Brennan y Resnick, 2012; Kafai y Burke, 2015). Además, si bien algunos trabajos han investigado las trayectorias de aprendizaje de las ciencias de la computación en varios niveles de grado (por ejemplo, Brennan y Resnick, 2012; Seiter y Foreman, 2013), no existe una base consistente para las progresiones de aprendizaje en todo el rango de K-12.

El marco propone una progresión de aprendizaje K-12 que puede evaluarse y desarrollarse a través de investigaciones futuras.

Estos hallazgos mixtos y limitados apuntan directamente a la necesidad de más investigación para comprender no solo lo que los niños en ciencias de la computación son cognitivamente capaces de hacer y cuándo, sino también las formas más efectivas para que ese aprendizaje ocurra para una población estudiantil más amplia y diversa. El marco en sí mismo propone una progresión de aprendizaje de K–12 que se puede evaluar y construir a través de investigaciones futuras. Estos estudios pueden ayudar a identificar y validar los puntos de referencia de aprendizaje dentro y entre las bandas de grado en función de las progresiones del marco, así como utilizar el marco como punto de partida para proponer progresiones de aprendizaje alternativas para evaluar. Esta línea de trabajo también puede investigar los conceptos específicos de que los estudiantes tienen problemas de aprendizaje, posibles conceptos erróneos y la edad a la que estas dificultades comienzan a ayudar a informar revisiones futuras de las progresiones del marco.

Los estudios relacionados pueden evaluar cómo las prácticas de enseñanza específicas y los enfoques pedagógicos influyen en los diferentes tipos de resultados de aprendizaje en ciencias de la computación. La investigación podría abordar los efectos de entornos de programación altamente andamiada en comparación con contextos más abiertos y de "juego libre"; la influencia de los contextos de computación social (por ejemplo, programación de pares); y el impacto de los currículos "enchufados" versus "desconectados" en el aprendizaje y las disposiciones de las ciencias de la computación de los estudiantes. Es importante destacar que este trabajo no puede clasificarse para una edad o nivel de grado, sino que debe estudiar con un objetivo más amplio de comprender si ciertas prácticas pedagógicas y de enseñanza influyen de manera diferencial en los resultados de los estudiantes en una variedad de niveles de desarrollo y conocimiento.

Del mismo modo, los estudios de niños más pequeños pueden investigar cómo la adquisición temprana de ciertos conceptos y prácticas computacionales influye en las experiencias de aprendizaje posteriores. Además de investigar si y cuánto conocimiento de las ciencias de la computación retienen los niños a lo largo de la experiencia K-12, incluso cuando hay brechas en el aprendizaje (Guzdial, 2016, p. 100), los estudios longitudinales también pueden enfatizar el papel que juegan las primeras experiencias en ciencias de la computación en el desarrollo de futuras actividades de las ciencias de la computación y la autoeficacia.

Además, la investigación futura puede explorar la influencia de diferentes progresiones de aprendizaje en las actividades y autoeficacia de los estudiantes hacia las ciencias de la computación, además de los resultados de logros para comprender mejor si las diferentes vías influyen en el compromiso futuro y el éxito en el campo de la computación. Esta línea de trabajo puede extenderse para comprender los efectos de los cuentos para diferentes poblaciones de estudiantes. Luego puede informar el desarrollo de progresiones de aprendizaje adaptables y flexibles para garantizar que todos los estudiantes tengan oportunidades sólidas de aprendizaje de las ciencias de la computación en su educación primaria y secundaria.

Conocimiento del contenido pedagógico

El Conocimiento del Contenido Pedagógico (CCP) es el conocimiento que los profesores tienen sobre la enseñanza como práctica combinada con su experiencia en la materia (Shulman, 1986). Además, la importancia de los factores contextuales, incluidas las actitudes de los docentes, la autoeficacia y los juicios de valor, así como la especificidad de dominio impregnan las conceptualizaciones más recientes de CCP (Bender et al., 2015). En ciencias de la computación, CCP podría incluir las mejores prácticas para explicar los conceptos de ciencias de la computación, cómo abordar los conceptos erróneos comunes que los profesores y los estudiantes podrían tener sobre la comprensión de cómo crear un entorno computacional inclusivo con estudiantes de diversos orígenes y experiencia previa.

Explorar CCP es particularmente crítico para la ampliación de la educación en ciencias de la computación, especialmente dados los variados (y más a menudo falta) requisitos de certificación de los profesores de las ciencias de la computación (CSTA Teacher Certification Task Force, 2008). La falta de requisitos consistentes deja a las escuelas y distritos con el desafío de tratar de ofrecer cursos de las ciencias de la computación sin profesores suficientemente capacitados. De hecho, una de las barreras principales para ofrecer las ciencias de la computación en las aulas K-12, según los líderes escolares, es la falta de profesores con las habilidades necesarias para enseñarla (Gallup, 2015b). Lograr la visión del marco de la educación en ciencias de la computación para todos los estudiantes requiere que las

comunidades de investigación y desarrollo profesional construyen el CCP de los profesores relacionado con cada área central en el marco. Este requisito se aplica a los docentes de las ciencias de la computación en formación, así como a los docentes en formación que integran las ciencias de la computación en otras materias.

Lograr la visión del marco requiere que las comunidades de investigación y desarrollo profesional desarrollen el CCP de los profesores.

La investigación previa sobre las ciencias de la computación CCP (Conocimiento del Contenido Pedagógico) de los docentes existe principalmente en el nivel postsecundario y se centra en la enseñanza de cursos introductorios. Por ejemplo, la investigación ha analizado la efectividad de compartir CCP a través de ejemplos que presentan un ejercicio y describen respuestas correctas e incorrectas, ideas erróneas esperadas, comentarios apropiados para dar a los estudiantes y métodos para provocar discusión (Koppelman, 2007; Koppelman, 2008). Otra investigación ha analizado técnicas específicas para disminuir la carga cognitiva y aumentar el rendimiento al enseñar programación a través de ejemplos trabajados, como secciones de código pre-etiquetadas o pedirles a los estudiantes que generen sus propias etiquetas (Morrison, Margulieux y Guzdial, 2015).

Recientemente se han tomado medidas fundamentales para crear un modelo teórico para enmarcar el CCP de las ciencias de la computación de los profesores de primaria y secundaria (Bender et al., 2015; Hubwieser, Magenheim, Mühling y Ruf, 2013; Saeli, Perrenet, Jochems y Zwaneveld, 2011) Saeli y sus colegas (2011) revelaron que el coaching es un componente crítico del CCP de los profesores de secundaria específicos para la programación de la enseñanza, señalando que el coaching puede ayudar a impulsar los procesos de resolución de problemas, reflexión y pensamiento algorítmico de los estudiantes. Los autores también delinearon dificultades comunes para los estudiantes, incluida la traducción del lenguaje humano a un lenguaje que es entendible por una computadora y poder cambiar entre diferentes lenguajes de programación (Saeli et al., 2011). En un estudio de seguimiento, Saeli, Perrenet, Jochems y Zwaneveld (2012) conceptualizaron aún más CCP para la programación, sugiriendo que los profesores de secundaria utilicen lenguajes de programación simples para ayudar a compensar los desafíos sintácticos y proporcionar varios problemas significativos para que los estudiantes resuelvan en lugar de muchos problemas que carecen de significado relevante en el mundo real y personal.

El trabajo de Hubwieser y colegas (2013) y Bender y colegas (2015) para desarrollar un modelo de competencia de enseñanza de las ciencias de la computación es quizás la conceptualización más rigurosa de la CCP de las ciencias de la computación K-12 hasta la fecha. Los autores proponen cinco dimensiones core de contenido y cinco categorías de creencias/orientaciones motivacionales requeridas para una enseñanza efectiva de las ciencias de la computación, y proporcionan descripciones específicas de cómo se ven esas competencias en la práctica. Por ejemplo, la dimensión de contenido 3: Cuestiones relacionadas con el alumno describe la necesidad de que los profesores comprendan cómo "adaptar sus métodos de enseñanza, contextos, representaciones de contenido y material a los diferentes requisitos que se producen debido a la diversidad de los estudiantes de las ciencias de la computación" (Bender et al., 2015, p. 528). Del mismo modo, Creencias/Orientación motivacional Categoría 2: Creencias sobre la enseñanza y el aprendizaje

en las ciencias de la computación sugiere que los profesores deberían estar "convencidos de que los estudiantes están aprendiendo de manera autónoma y al acercarse críticamente a los contenidos de ciencias de la computación" para ser profesores de ciencias de la computación eficaces (Bender et al., 2015, p. 528).

Proyectos recientes, como CSTeachingTips.org, han adoptado un enfoque de investigación aplicada para documentar CCP mediante la recopilación y el informe de las mejores prácticas del campo (Lewis, 2016). Además, la comunidad de educación en ciencias de la computación está explorando las mejores prácticas, como la programación de parejas (por ejemplo, Denner, Werner, Campe y Ortiz, 2014; Hanks, 2008), la instrucción de pares (Kothiyal, Majumdar, Murthy e Iyer, 2013), e identificar y abordar conceptos erróneos
(Ohrndorf, 2015).

A pesar del trabajo que ya se está haciendo en torno al CCP de ciencias de la computación, este cuerpo de investigación sigue siendo incipiente, y las preguntas sobre CCP de ciencias de la computación en diferentes niveles de grado, la influencia de factores contextuales en las prácticas docentes y el aprendizaje de los estudiantes, y el papel del desarrollo profesional previo y en servicio permanecen sin respuesta. Aunque la investigación ha conceptualizado modelos generales de CCP de ciencias de la computación K – 12 o 9–12 (por ejemplo, Bender et al., 2015; Hubwieser et al., 2013; Saeli et al., 2011), las competencias específicas de los profesores dentro de cada banda de grado son notablemente ausente. Por lo tanto, los futuros investigadores pueden evaluar los modelos actuales de CCP en diferentes niveles de grado, así como proponer y probar nuevos modelos que brinden una comprensión más granular de CCP específica para profesores de primaria, secundaria y preparatoria.

Comprender los modelos CCP se vuelve aún más relevante para los niveles de grado más jóvenes, donde si las ciencias de la computación se enseñan actualmente, se integra en otras áreas temáticas. Yadav, Mayfield, Zhou, Hambrusch y Korb (2014) sugieren que los profesores deberían aprender los conceptos de pensamiento computacional para estar mejor preparados para integrarlo en su enseñanza y poder articular usos más amplios como una herramienta para resolver problemas en otras disciplinas. Los estudios futuros pueden extender este trabajo explorando cómo los profesores de una variedad de disciplinas incorporan e implementan prácticas de pensamiento computacional en el aula y la capacitación y preparación específicas que necesitan para hacerlo. Además, la investigación puede investigar las prácticas de integración más efectivas para impulsar el interés y el aprendizaje de los estudiantes en ciencias de la computación y también en otras áreas temáticas.

A medida que el aprendizaje de las ciencias de la computación se vuelve cada vez más popular y pasa de una oferta básica optativa a una principal, los profesores de las ciencias de la computación necesitarán comprender no solo qué aspectos de los estudiantes de las ciencias de la computación pueden tener dificultades para aprender, sino también cómo aplicar técnicas establecidas para diferenciar la instrucción para cumplir con las necesidades de una población más diversa. Investigaciones recientes han explorado el uso de Universal Design for Learning para desarrollar y refinar experiencias introductorias de las ciencias de la computación para una amplia gama de estudiantes (Hansen, Hansen, Dwyer, Harlow y Franklin, 2016), pero se necesita más trabajo para comprender completamente la clave componentes de CCP necesarios para garantizar que todos los estudiantes no solo tengan acceso a las ciencias de la computación sino que también tengan oportunidades diferenciadas para participar en el campo que satisfaga sus necesidades únicas de aprendizaje. Como parte de esta línea de investigación, la investigación también puede explorar el apoyo instructivo único y los recursos necesarios para estudiantes con y sin experiencia previa en computación. Este soporte podría incluir el desarrollo y prueba de materiales suplementarios para ambos extremos del espectro (por ejemplo, materiales menos complejos para aquellos sin experiencia y oportunidades más avanzadas para aquellos con experiencia más extensa), así como explorar el papel que extendió las oportunidades de aprendizaje, como clubes después de la escuela, campamentos, pasantías y cursos de nivel universitario pueden tener resultados estudiantiles.

Además, dadas las percepciones erróneas continuas de lo que son las ciencias de la computación, combinada con estereotipos desenfrenados y una clara falta de diversidad en el campo de la computación, es tan importante tener en cuenta cómo los profesores ven y piensan en las ciencias de la computación como los estudiantes. Si los profesores son los que crean el entorno computacional, también son los que pueden estar restableciendo los estándares culturales y los estereotipos en sus alumnos. El sentido de pertenencia en las aulas STEM es un fuerte predictor del interés y la motivación de las mujeres en los campos (Good, Rattan y Dweck, 2012; Smith, Lewis, Hawthorne y Hodges, 2013), de modo que asegurar que los profesores tengan un CCP sólido de impactos directos en la forma en que abordan las ciencias de la computación y el entorno de aula que crean. La investigación futura puede explorar cómo las actitudes y expectativas de los profesores de las ciencias de la computación afectan el aprendizaje de los alumnos y las actitudes hacia las ciencias de la computación. Parte de este trabajo podría incluir investigación sobre programas de desarrollo profesional en formación y en servicio para comprender cómo pueden ayudar a los profesores a desarrollar una fuerte autoeficacia y actitudes positivas hacia las ciencias de la computación, además de ayudar a desarrollar su conocimiento de contenido y prácticas de implementación.

Facilitar el aprendizaje en otras disciplinas

A medida que las escuelas trabajan para implementar las vías de educación en ciencias de la computación, se les puede pedir a muchos profesores, especialmente a los que enseñan en las aulas de K-8, que integren el contenido de ciencias de la computación en su plan de estudios actual. Además, se les puede pedir a los profesores de secundaria o preparatoria de otras disciplinas que enseñen un curso dedicado a las ciencias de la computación, por lo que es imperativo que estos profesores sepan cómo aplicar los conocimientos y habilidades de sus disciplinas principales para enseñar las ciencias de la computación y viceversa. Significativamente, las ciencias de la computación son medios que puede facilitar y apoyar el aprendizaje en todas las disciplinas y en todo el plan de estudios tradicional de primaria y secundaria. Más que solo crear conexiones interdisciplinarias, las ciencias de la computación son la base de una forma de expresión, al igual que la escritura y el arte, que potencia la voz de los estudiantes más allá de los estrictos dominios temáticos de "alfabetización" o "programación."

Las ciencias de la computación son la base de una forma de expresión, al igual que la escritura y el arte.

Ya sea que integren las ciencias de la computación en el currículo o que la enseñen por su cuenta, los profesores requerirán capacitación y apoyo más allá del conocimiento del contenido: tendrán que lidiar no solo con el aprendizaje de una nueva área temática sino también con el cambio potencial de las prácticas pedagógicas para reflejar una educación más interdisciplinaria y estudiantil en un entorno de aprendizaje centrado. Al igual que con cualquier innovación educativa nueva, será necesario fortalecer la confianza de los profesores en su capacidad para enseñar ciencias de la computación de manera efectiva, así como comunicar claramente el valor de las ciencias de la computación para el aprendizaje de los estudiantes para garantizar una enseñanza de alta calidad con impactos significativos.

Además, la experiencia externa de educadores y académicos de otras disciplinas es crítica a medida que crece la base de conocimiento de ciencias de la computación K–12, y las colaboraciones con otros campos pueden conducir a hallazgos interesantes e imprevistos. Las colaboraciones entre investigadores de los campos de la educación y las ciencias de la computación son importantes para construir una comunidad de investigación rigurosa y significativa que tenga un impacto más amplio dentro y más allá de estas disciplinas.

En el nivel postsecundario, los estudios han investigado la viabilidad de integrar las ciencias de la computación con otras áreas temáticas, incluida la bioinformática (LeBlanc y Dyer, 2004), y cómo el uso de múltiples formas de medios, como imágenes y sonidos, puede mejorar la introducción en cursos de las ciencias de la computación de nivel universitario (Guzdial, 2013). Schulz y Pinkwart (2015) también exploraron cómo la computación física podría integrarse en los cursos de educación STEM de nivel secundario de formación, y descubrieron que los profesores estudiantes creían que las competencias esenciales para la computación, la física y la biología estaban suficientemente cubiertas en el currículo. En el nivel K-12, los investigadores también han investigado la viabilidad de los planes de estudios integrados de las ciencias de la computación. Goldschmidt, MacDonald, O'Rourke y Milonovich (2011) discutieron formas simples en que los conceptos de ciencias de la computación pueden integrarse en todos los dominios de materias en el nivel K-12, incluido el gimnasio, el arte y la música. Goldberg, Grunwald, Lewis, Feld y Hug (2012) adoptaron un enfoque similar, quienes llevaron

las ciencias de la computación a clases que los estudiantes de secundaria y preparatoria ya toman, incluyendo arte, salud y estudios sociales. Aunque los autores no midieron explícitamente los resultados a nivel de los alumnos, los profesores informaron que, junto con sus alumnos, aumentaron su comprensión y mejoraron sus percepciones de las ciencias de la computación (Goldberg et al., 2012).

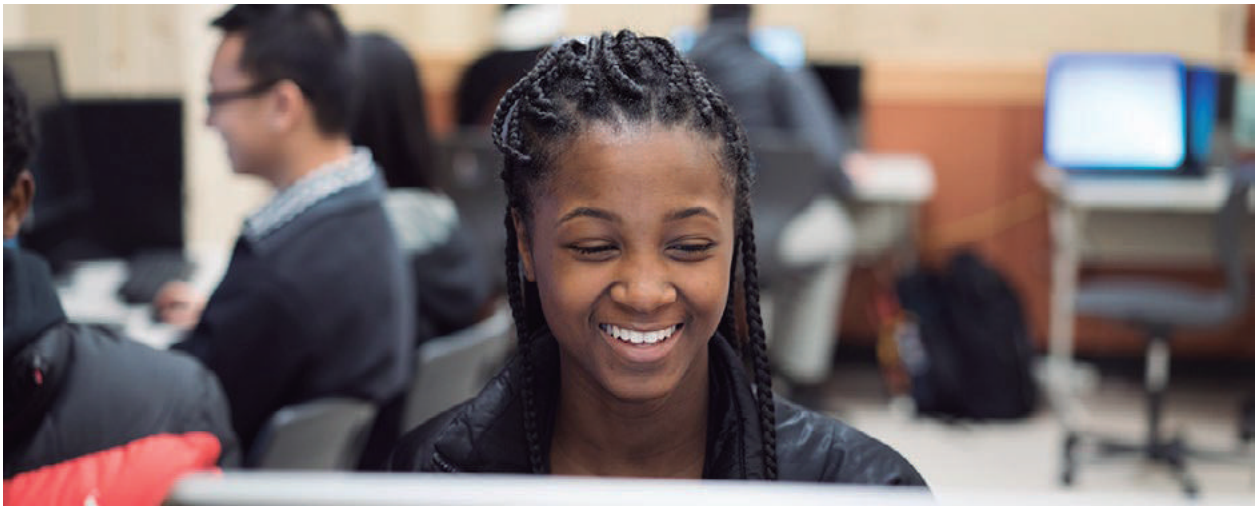
Los investigadores también han buscado explícitamente comprender la transferencia del aprendizaje de las ciencias de la computación a otras materias, especialmente las matemáticas, con resultados mixtos (para revisiones, ver Palumbo, 1990; Simon et al., 2006). Otros se han centrado en utilizar las ciencias de la computación como medio para aprender en otras áreas temáticas y las condiciones que deben cumplirse para facilitar dicho aprendizaje. Por ejemplo, Kafai, Franke, Ching y Shih (1998) mostraron que la programación podría usarse como un medio para que los estudiantes de primaria expresen fracciones, mientras que Schanzer, Fisler, Krishnamurthi y Felleisen (2015) encontraron resultados positivos en el logro de álgebra para los estudiantes que participaban en un currículo que enseñaba álgebra a través de la programación de computadoras. Además, Grover, Pea y Cooper (2016) mostraron que entre los estudiantes de secundaria que participaban en un plan de estudios introductorio de ciencias de la computación centrado en la programación y el pensamiento algebraico, la experiencia previa en matemáticas y computación, así como el dominio del idioma inglés, eran predictores críticos del pensamiento algebraico. Resultados y concepciones de las ciencias de la computación, el estudio de Lewis y Shah (2012) de estudiantes de primaria, mostró resultados similares; los estudiantes con puntajes matemáticos más altos obtuvieron mejores resultados en las evaluaciones de programación. En un estudio posterior, Lewis (2014) sugiere que la comprensión de los estudiantes de secundaria de la sustitución algebraica puede transferirse a las ciencias de la computación, señalando las dificultades que los estudiantes tienen con dicha transferencia de conocimiento.

Dados los hallazgos mixtos sobre la transferencia del aprendizaje, así como la falta de investigación sobre la transferencia a dominios que no son STEM, se necesita más investigación para proporcionar una comprensión sólida de si, cómo y en qué contexto el aprendizaje de las ciencias de la computación puede transferirse a diferentes materias y viceversa la investigación fundamental puede ayudar a identificar contenido complementario en asignaturas de ciencias no computacionales y desarrollar y probar diferentes modelos de integración de contenido. Esta línea de trabajo puede incluir la identificación de mejores prácticas y estrategias de enseñanza efectivas que conducen al aprendizaje de los estudiantes en todas las materias, incluido el CCP específico relevante para la integración de múltiples materias. Como muchos profesores de ciencias no computacionales pueden estar enseñando conceptos de las ciencias de la computación en sus aulas, los investigadores también pueden explorar qué capacitación y apoyo son necesarios para asegurar que estos profesores desarrollen no solo estrategias de enseñanza efectivas para apoyar el aprendizaje de los estudiantes, sino también actitudes positivas y autoeficacia en la enseñanza Conceptos de las ciencias de la computación.

Además de explorar las conexiones más obvias entre el aprendizaje de las ciencias de la computación y otras áreas de contenido de STEM, la investigación futura puede investigar el logro de los estudiantes y los resultados de actitud en áreas que no son de STEM, como la alfabetización, el arte y los estudios sociales, después de recibir la instrucción de las ciencias de la computación incorporada en el área de contenido dada. Además, dicha investigación puede abordar estos temas con diversos grados de granularidad. Por ejemplo, a un nivel más granular, los estudios podrían investigar si la depuración del aprendizaje en un contexto computacional se transfiere a la resolución de problemas en matemáticas, mientras que a un nivel más amplio, los estudios podrían explorar los efectos de todas las ciencias de la computación con currículos

integrados sobre el aprendizaje del alumno en todas las materias. Los investigadores pueden extender este trabajo para desarrollar una comprensión más matizada de las condiciones que ayudan a apoyar dicha transferencia de conocimiento, incluidas las prácticas docentes, la experiencia previa y las diferencias de transferencia para diferentes dominios de asignaturas.

Además, dicha investigación se puede mejorar a través de colaboraciones entre investigadores de las ciencias de la computación y aquellos de otras áreas de contenido para explorar similitudes y diferencias entre CCP y progresiones de aprendizaje en diferentes dominios. Incluso la National Science Foundation (NSF) ha reconocido la importancia de tales colaboraciones en su solicitud de Asociaciones de Computación STEM +, que requiere propuestos de proyectos para integrar las ciencias de la computación en la educación STEM. Los proyectos financiados actualmente son prometedores para avanzar en el campo a medida que investigan temas que van desde la integración de las ciencias de la computación en las aulas de ciencias y matemáticas de primaria y secundaria hasta el desarrollo de nuevos currículos interdisciplinarios de las ciencias de la computación para centrarse en el desarrollo profesional para profesores y administradores escolares para ampliar tales iniciativas. Los estudios futuros pueden basarse en los hallazgos de estos proyectos y continuar los esfuerzos de colaboración en curso de un esfuerzo por avanzar no solo en la educación en ciencias de la computación sino también en la enseñanza y el aprendizaje en todo el sector educativo K-12 en general.



Limitaciones

A pesar de los continuos avances en la investigación de educación en ciencias de la computación K-12, el campo sigue necesitando estudios rigurosos para proporcionar evidencia empírica para abordar muchas de las preguntas sin respuesta descritas en la agenda de investigación anterior. La base de evidencia sobre la que descansa el marco es incompleta, y las limitaciones en la investigación actual se analizan a continuación. Estas limitaciones no lo abarcan todo, sin embargo, se pueden ver como cuestiones procesables que ofrecen información adicional para futuros estudios.

Primero, y quizás lo más apremiante, gran parte de la investigación publicada sobre educación en las ciencias de la computación de EE. UU., se realiza con estudiantes universitarios, muchos de los cuales han optado por los programas

de la licenciatura en ciencias de la computación, con menos estudios dirigidos a estudiantes de K–12. Mientras que la comunidad internacional de investigación proporciona una idea del contexto educativo K-12, los sistemas escolares internacionales varían dramáticamente del sistema estadounidense. Dado que ninguno de estos dos contextos (postsecundario nacional o K-12 internacional) se transfiere fácilmente al entorno K-12 de EE. UU., sigue habiendo una base de investigación limitada relacionada con las aulas de las ciencias de la computación K-12 en los Estados Unidos (Hubwieser et al., 2011). El Marco de Ciencias de la Computación K–12 promueve un enfoque renovado en la investigación de educación en ciencias de la computación y exige específicamente más estudios para validar las progresiones de aprendizaje propuestas e informar futuras revisiones al marco.

En segundo lugar, la investigación que existe en el entorno de K–12 a menudo es limitada en muestra y alcance, lo que dificulta la generalización de los resultados en diferentes niveles de grado y poblaciones de estudiantes, así como los conceptos y planes de estudio de ciencias de la computación. Iniciativas más recientes han comenzado a abordar esta limitación mediante la adaptación de materiales curriculares convencionales para estudiantes con diferencias de aprendizaje (por ejemplo, Israel et al., 2015; Wille, Pike y Century, 2015); integrar la informática en la alfabetización primaria en un distrito escolar grande y diverso (Milenkovic, Acquavita y Kim, 2015); evaluar las actividades destinadas a ayudar a los alumnos diversos con diferentes niveles de preparación para las matemáticas y el idioma inglés en grandes distritos escolares urbanos (Grover, Jackie y Lundh, 2015); desarrollar progresiones de aprendizaje para el nivel K–8 (Isaacs et al., 2016); y llevar a cabo evaluaciones de implementación a gran escala en todo el entorno de aprendizaje K–12 (Mark y DeLyser, 2016). Si bien ciertamente se necesita más investigación, las ideas y recomendaciones que se producen a partir de estas iniciativas iniciales pueden informar futuras revisiones del marco y sus subproductos, como los estándares, los currículos y el desarrollo profesional.

Finalmente, como con cualquier entorno de investigación aplicada, las limitaciones metodológicas plantean amenazas a la validez de los resultados de la investigación; Este problema puede ser aún más pertinente en la educación en ciencias de la computación K–12 debido a la novedad de realizar investigaciones a este nivel y, por lo tanto, a la falta de medidas preestablecidas válidas y confiables para hacerlo. Aunque hay un trabajo en curso para desarrollar instrumentos sólidos (por ejemplo, Wille y Kim, 2015), la variedad de materiales curriculares y enfoques de implementación en los esfuerzos de educación en ciencias de la computación requiere más estudios de investigación metodológica para comprender las mejores prácticas para medir lo que la ciencia de la computación K–12 la educación se ve a nivel de aula, los factores que afectan la implementación y, en última instancia, los resultados de actitud y aprendizaje de los alumnos y profesores.

Consideraciones de política e implementación

La reciente afluencia de atención por parte de las agencias educativas federales, estatales y locales ha llevado la educación en ciencias de la computación K–12 a la atención nacional, y en todos los niveles del gobierno, las políticas nuevas y recomendadas continuarán avanzando. Estas políticas incluyen contar las ciencias de la computación para un requisito de graduación y esperar que todas las escuelas dentro de un distrito involucren a todos los estudiantes en al menos una experiencia en ciencias de la computación en todos los niveles de aprendizaje (es decir, primaria, secundaria y preparatoria). Al mismo tiempo, cómo las entidades locales promulgan dichas políticas y los cuáles son políticas

que en el aula parece aún no se han entendido claramente. La implementación de cualquier innovación educativa seguramente será compleja, con muchas partes móviles y factores que afectan tanto la fidelidad de la implementación como los resultados posteriores de docentes y estudiantes.

Una extensa revisión de la investigación de implementación de innovación educativa realizada por Century y Cassata (en prensa) proporciona un marco contextual para comprender las complejidades de la implementación de las ciencias de la computación en el entorno educativo K-12. Los autores sugieren cinco facetas de la investigación de implementación:

1. informar el diseño y desarrollo de innovaciones;
2. evaluar si y en qué medida una innovación logra los resultados deseados;
3. comprender por qué funciona una innovación, para quién y en qué contextos;
4. mejorar el diseño y el uso de la innovación en entornos aplicados; y
5. desarrollo de la teoría informativa (Century y Cassata, en prensa).

Estas cinco categorías van más allá de la fidelidad de implementación para comprender qué, por qué, cómo y para quién de las innovaciones educativas en un esfuerzo por capturar todas las facetas que influyen en los resultados de tales iniciativas. A un nivel más granular, la investigación de implementación puede trabajar para medir factores estructurales (por ejemplo, frecuencia y duración; contenido cubierto y omitido; modificación y suplementación) y factores de interacción (por ejemplo, facilitación de prácticas pedagógicas por parte de los profesores, actitudes hacia la innovación, maestría e interés de los estudiantes y autoeficacia en el dominio del contenido) (Century, Cassata, Rudnick y Freeman, 2012).

Con base en estas concepciones más amplias de la investigación de implementación de innovación educativa, existen varias consideraciones con respecto a las políticas y las innovaciones curriculares en la educación en ciencias de la computación K-12. Estas consideraciones incluyen la comprensión y medición del currículo y la evaluación; curso y vías de instrucción; desarrollo profesional docente; y participación de las partes interesadas de la comunidad y la industria. Por ejemplo, además de utilizar evaluaciones de logros tradicionales para medir el aprendizaje de ciencias de la computación (por ejemplo, pruebas de opción múltiple y de respuesta corta), se podrían desarrollar más medidas "no tradicionales", como tareas de rendimiento y carteras de trabajo de los estudiantes, que capturan la riqueza y la creatividad del entorno computacional que de otro modo se perdería en la mayoría de las evaluaciones estándar. Del mismo modo, seguramente se explorarán varios modelos de cursos (por ejemplo, cursos integrados, cursos independientes, itinerarios de cursos y educación profesional y técnica), de modo que un componente importante para comprender la implementación será explorar los modelos más efectivos para diferentes niveles de grado, poblaciones estudiantiles y objetivos de aprendizaje. Parte de este trabajo necesariamente incluye comprender las mejores prácticas para asegurar que los profesores tengan suficiente capacitación y apoyo antes y durante su tiempo en el aula, así como establecer oportunidades y requisitos de certificación más consistentes. Además, dado el enfoque en alentar a los estudiantes a persistir en el campo de la computación a largo plazo, es probable que las asociaciones comunitarias y de la industria también desempeñen un papel en cómo las escuelas y los distritos involucran a sus estudiantes en ciencias de la computación.

Para abordar estos componentes de implementación, los líderes de la escuela y del distrito también requerirán conocimientos, recursos y apoyo adicionales en su esfuerzo por proporcionar instrucción las ciencias de la computación de alta calidad y desarrollo profesional. De hecho, un factor importante en la implementación de la innovación educativa es el apoyo al liderazgo y el valor otorgado a la implementación de la innovación (Century et al., 2012; Century y Cassata, en prensa). Una iniciativa, LeadCS.org, ofrece herramientas desarrolladas por investigadores y anécdotas de profesores, escuelas, distritos y líderes asociados para ayudar a los líderes educativos a llevar o mejorar las iniciativas computacionales existentes en sus propios contextos. El sitio web también proporciona enlaces a recursos adicionales relacionados con infraestructura, instrucción e implementación.

Mirando hacia el futuro

El Marco de Ciencias de la Computación K–12 se basa en investigaciones disponibles actualmente, pero aún queda un largo camino por recorrer. Una revisión de la investigación actual encontró áreas de alineación con los conceptos, prácticas y declaraciones core del marco, lo que resultó en un marco más convincente y robusto. Sin embargo, esta revisión de investigación también identificó lagunas en el campo, que se reunieron para crear una agenda de investigación en el futuro. Esta agenda crea una oportunidad para que los investigadores actuales y futuros de educación en ciencias de la computación exploren preguntas de investigación significativas para avanzar en el campo y apoyar el marco.

A medida que se difunde la educación en ciencias de la computación, particularmente con el uso de este marco, será necesario hacer más preguntas y completar más estudios para sugerir respuestas. La comunidad de partes interesadas (Stakeholders) en la educación en ciencias de la computación, incluidos investigadores, profesores, administradores y formuladores de políticas, deben unirse para avanzar en la investigación que apuntalará el futuro de la educación en ciencias de la computación K–12.

LeadCS.org ofrece herramientas desarrolladas por investigadores para ayudar a los líderes educativos con iniciativas de las ciencias de la computación.

Referencias

Achieve. (2015). *The role of learning progressions in competency-based pathways.* Retrieved from http://www.achieve.org/files/Achieve-LearningProgressionsinCBP.pdf

Aivaloglou, E., & Hermans, F. (2016, September). How kids code and how we know: An exploratory study on the Scratch repository. In *Proceedings of the Twelfth Annual International Conference on International Computing Education Research* (pp. 53–61).

Bender, E., Hubwieser, P., Schaper, N., Margaritis, M., Berges, M., Ohrndorf, L., . . . Schubert, S. (2015). Towards a competency model for teaching computer science. *Peabody Journal of Education, 90*(4), 519–532. doi: 10.1080/0161956X.2015.1068082

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look* (SRI technical report). Menlo Park, CA: SRI International. Retrieved from http://pact.sri.com/resources.html

Blickenstaff, J. C. (2005). Women and science careers: Leaky pipeline or gender filter? *Gender and Education, 17*(4), 369–386. doi: 10.1080/09540250500145072

Brennan, K. & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Paper presented at the annual meeting of the American Educational Research Association, Vancouver, BC, Canada.

Buffardi, K., & Edwards, S. H. (2013, August). Effective and ineffective software testing behaviors by novice programmers. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 83–90).

Bureau of Labor Statistics. (2015). *Occupational employment statistics* [Data file]. Retrieved from http://www.bls.gov/oes

Century, J., & Cassata, A. (in press). Measuring implementation and implementation research: Finding common ground on what, how and why. *Review of Research in Education, Centennial Edition.* American Educational Research Association.

Century, J., Cassata, A., Rudnick, M., & Freeman, C. (2012). Measuring enactment of innovations and the factors that affect implementation and sustainability: Moving toward common language and shared conceptual understanding. *Journal of Behavioral Health Services & Research, 39*(4), 343–361.

Cernavskis, A. (2015, June 25). In San Francisco, computer science for all . . . soon. *The Hechinger Report.* Retrieved from http://hechingerreport.org/san-francisco-plans-to-be-first-large-district-to-bring-computer-science-to-all-grades/

Computer Science Teachers Association Teacher Certification Task Force. (2008). *Ensuring exemplary teaching in an essential discipline: Addressing the crisis in computer science teacher certification.* New York, NY: Computer Science Teachers Association and the Association for Computing Machinery.

Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). A future for computing education. *Communications of the ACM, 57*(11), 34–46.

Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016, February). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (pp. 1438–1449).

De Boulay, B. (1989). Some difficulties of learning to program. In E. Soloway and J. Spohrer (Eds.), *Studying the novice programmer.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education, 46*(3), 277–296.

Dwyer, H., Hill, C., Carpenter, S., Harlow, D., & Franklin, D. (2014, March). Identifying elementary students' pre-instructional ability to develop algorithms and step-by-step instructions. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 511–516).

Evers, V., & Day, D. (1997). The role of culture in interface acceptance. In S. Howard, J. Hammond, & G. Lindgaard (Eds.), *Proceedings of Human-Computer Interaction INTERACT'97* (pp. 260–267). Sydney, Australia: Chapman & Hall.

Franklin, D. F. (2015, February). Putting the computer science in computing education research. *Communications of the ACM*, *58*(2), 34–36.

Goldberg, D. S., Grunwald, D., Lewis, C., Feld, J. A., & Hug, S. (2012). Engaging computer science in traditional education: The ECSITE project. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 351–356).

Goldschmidt, D., MacDonald, I., O'Rourke, J., & Milonovich, B. (2011). An interdisciplinary approach to injecting computer science into the K–12 classroom. *Journal of Computing Science in College*, *(26)*6, 78–85.

Good, C., Rattan, A., & Dweck, C. S. (2012). Why do women opt out? Sense of belonging and women's representation in mathematics. *Journal of Personality and Social Psychology*, *102*(4), 700–717.

Goode, J., Estrella, R., & Margolis, J. (2006). Lost in translation: Gender and high school computer science. In W. Aspray & J. M. Cohoon (Eds.), *Women and information technology: Research on underrepresentation* (pp. 89–113). Cambridge, MA: MIT Press.

Google & Gallup. (2015a). *Images of computer science: Perceptions among students, parents, and educators in the U.S.* Retrieved from http://g.co/cseduresearch

Google & Gallup. (2015b). *Searching for computer science: Access and barriers in U.S. K–12 education*. Retrieved from http://g.co/cseduresearch

Grover, S., Jackiw, N., & Lundh, P. (2015). Thinking outside the box: Integrating dynamic mathematics to advance computational thinking for diverse student populations. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1543062

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 41*(1), 38–43.

Grover, S., Pea, R., & Cooper, S. (2016, March). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 552–557), Memphis, TN.

Guernsey, L. (2012). *Screen time: How electronic media from baby videos to educational software affects your young child.* Philadelphia, PA: Basic Books.

Guzdial, M. (2013, August). Exploring hypotheses about media computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 19–26).

Guzdial, M. (2016). *Learner-centered design of computing education: Research on computing for everyone*. Synthesis lectures on human-centered informatics. Morgan and Claypool.

Hanks, B. (2008). Problems encountered by novice pair programmers. *Journal on Educational Resources in Computing (JERIC)*, *7*(4), Article 2.

Hansen, A., Dwyer, H., Hill, C., Iveland, A., Martinez, T., Harlow, D., & Franklin, D. (2015). Interactive design by children: A construct map for programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 267–270).

Hansen, A., Hansen, E., Dwyer, H., Harlow, D., & Franklin, D. (2016). Differentiating for diversity: Using universal design for learning in computer science education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 376–381).

Hansen, A., Iveland, A., Carlin, C., Harlow, D., & Franklin, D. (2016, June). User-centered design in block-based programming: Developmental and pedagogical considerations for children. In *Proceedings of the 15th International Conference on Interaction Design and Children* (pp. 147–156).

Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M. N., . . . Schubert, S. (2011, June). Computer science/informatics in secondary education. In *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education—Working Group Reports* (pp. 19–38).

Hubwieser, P., Magenheim, J., Mühling, A., & Ruf, A. (2013, August). Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 1–8).

Isaacs, A., Binkowski, T. A., Franklin, D., Rich, K., Strickland, C., Moran, C., . . . Maa, W. (2016). Learning trajectories for integrating K–5 computer science and mathematics. In *2016 CISE/EHR Principal Investigator & Community Meeting for CS in STEM Project Description Booklet* (p. 79). Retrieved from https://www.ncwit.org/sites/default/files/file_type/pi_book_compressed_2016.pdf

Israel, M., Wherfel, Q., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 students with disabilities to learn computational thinking and computer programming. *TEACHING Exceptional Children, 48*(1), 45–53.

Kafai, Y. B., Franke, M. L., Ching, C. C., & Shih, J. C. (1998). Game design as an interactive learning environment for fostering students' and teachers' mathematical inquiry. *International Journal of Computers for Mathematical Learning, 3*(2), 149–184. doi: 10.1023/A:1009777905226

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313–334.

Koppelman, H. (2007). Exercises as a tool for sharing pedagogical knowledge. In *Proceedings of the 12th Annual SIGCSE Conference in Innovation and Technology in Computer Science Education* (p. 361). doi: 10.1145/1268784.1268933

Koppelman, H. (2008, June). Pedagogical content knowledge and educational cases in computer science: An exploration. In *Proceedings of the Informing Science and IT Education Conference (InSITE)* (pp. 125–133), Varna, Bulgaria.

Kothiyal, A., Majumdar, R., Murthy, S., & Iyer, S. (2013, August). Effect of think-pair-share in a large CS1 class: 83% sustained engagement. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 137–144).

Ladner, R., & Israel, M. (2016). "For all" in "computer science for all." *Communications of the ACM, 59*(9), 26–28.

LeBlanc, M. D., & Dyer, B. D. (2004). Bioinformatics and computing curricula 2001: Why computer science is well positioned in a post-genomic world. *ACM SIGCSE Bulletin*, *36*(4), 64–68.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37.

Leidner, D. E., & Kayworth, T. (2006). A review of culture in information systems research: Toward a theory of information technology culture conflict. *MIS Quarterly*, *30*(2), 357–399.

Lewis, C. M. (2014, July). Exploring variation in students' correct traces of linear recursion. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (pp. 67–74).

Lewis, C. M. (2016). You wouldn't know it from SIGCSE proceedings, but we don't only teach CS1 (Abstract Only). In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (p. 494).

Lewis, C. M., & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 57–62).

Lewis, C. M., & Shah, N. (2015, July). How equity and inequity can emerge in pair programming. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 41–50).

Lishinski, A., Good, J., Sands, P., & Yadav, A. (2016, September). Methodological rigor and theoretical foundations of CS education research. In *Proceedings of the Twelfth Annual International Conference on International Computing Education Research* (pp. 161–169).

Lou, Y., Abrami, P. C., & d'Apollonia, S. (2001). Small group and individual learning with technology: A meta-analysis. *Review of Educational Research, 71*, 449–521.

Marcus, B. (2015, August 12). The lack of diversity in tech is a cultural issue. *Forbes.* Retrieved from http://www.forbes.com/sites/bonniemarcus/2015/08/12/the-lack-of-diversity-in-tech-is-a-cultural-issue/#622c464a3577

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing.* Cambridge, MA: MIT Press.

Margolis, J., Ryoo, J., Sandoval, C., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads*, *3*(4), 72–78.

Mark, J., & DeLyser, L. (2016). CSNYC knowledge forum: Launching research and evaluation for computer science education, for every school and every student in New York City. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1637654

Milenkovic, L., Acquavita, T., & Kim, D. (2015). Investigating conceptual foundations for a transdisciplinary model integrating computer science into the elementary STEM curriculum. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1542842

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015, July). Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 21–29).

National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common core state standards*. Washington DC: Author.

National Research Council. (2007). *Taking science to school: Learning and teaching science in grades K–8*. Committee on Science Learning-Kindergarten Through Eighth Grade. R. A. Duschl, H. A. Schweingruber, & A. W. Shouse (Eds.). Board on Science Education, Center for Education. Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

National Science Foundation. (2014). *Women, minorities, and persons with disabilities in science and engineering, Table 5–1: Bachelor's degrees awarded, by sex and field: 2002–2012* [Data file]. Retrieved from https://www.nsf.gov/statistics/wmpd/2013/sex.cfm

Next Generation Science Standards Lead States. (2013). *Next generation science standards: For states, by state*s. Washington, DC: The National Academies Press.

Ohrndorf, L. (2015, July). Measuring knowledge of misconceptions in computer science education. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 269–270).

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65–89.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*, 137–168.

Pea, R. D., Soloway, E., & Spohrer, J. C. (1987). The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics, 9*, 5–30.

Perkins, D. N., & Salomon, G. (1988). Teaching for transfer. *Educational Leadership*, *46*(1), 22–32.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics in Education*, *10*(1), 73–88.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2012). Programming: Teachers and pedagogical content knowledge in the Netherlands. *Informatics in Education*, *11*(1), 81–114.

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015, February). Transferring skills at solving word problems from computing to algebra through Bootstrap. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 616–621).

Schulz, S., & Pinkwart, N. (2015, November). Physical computing in STEM education. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 134–135).

Seiter, L., & Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 59–66).

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher 15*(2), 4–14. doi: 10.3102/0013189X015002004

Simon, Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., . . . Tutty, J. (2006, January). Predictors of success in a first programming course. In *Proceedings of the Eighth Australasian Computing Education Conference* (pp. 189–196), Hobart, Tasmania, Australia.

Smith, J. L., Lewis, K. L., Hawthorne, L., & Hodges, S. D. (2013). When trying hard isn't natural: Women's belonging with and motivation for male-dominated STEM fields as a function of effort expenditure concerns. *Personality and Social Psychology Bulletin*, *39*(2), 131–143.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850–858.

Sprague, P., & Schahczenski, C. (2002). Abstraction the key to CS1. *Journal of Computing Sciences in Colleges, 17*(3), 211–218.

Sullivan, G. (2014, May 29). Google statistics show Silicon Valley has a diversity problem. *The Washington Post.* Retrieved from https://www.washingtonpost.com/news/morning-mix/wp/2014/05/29/most-google-employees-are-white-men-where-are-allthewomen/

Valkenburg, P. M., & Peter, J. (2013). The differential susceptibility to media effects model. *Journal of Communication*, *63*, 221–243. doi: 10.1111/jcom.12024

Watson, W. E., Kumar, K., & Michaelsen, L. K. (1993). Cultural diversity's impact on interaction process and performance: Comparing homogeneous and diverse task groups. *Academy of Management Journal, 36*(3), 590–602.

Wehmeyer, M. L. (2015). Framing the future self-determination. *Remedial and Special Education*, *36*(1), 20–23.

Wehmeyer, M. L., Shogren, K. A., Palmer, S. B., Williams-Diehm, K. L., Little, T. D., & Boulton, A. (2012). The impact of the self-determined learning model of instruction on student self-determination. *Exceptional Children, 78*(2), 135–153.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wille, S. J., & Kim, D. (2015). Factors affecting high school student engagement in introductory computer science classes. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 675–675), New York, NY. doi: 10.1145/2676723.2691891

Wille, S. J., Pike, M., & Century, J. (2015). Bringing AP Computer Science Principles to students with learning disabilities and/or an ADHD: The hidden underrepresented group. Abstract. http://www.nsf.gov/awardsearch/showAward?AWD_ID=1542963

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), Article 5, 1–16.

Apéndices

Apéndice A: Comentarios y Revisiones

La revisión pública y la retroalimentación fueron esenciales para el desarrollo del marco. El marco pasó por tres períodos de revisión pública. El primer período de revisión incluyó un borrador de las declaraciones del concepto de la banda de grados 9-12 y las descripciones generales de las prácticas. El segundo y el tercero incluyeron borradores completos de declaraciones conceptuales, descripciones generales y prácticas e incluyeron un glosario y un prefacio que describieron el marco y proporcionaron una breve descripción de los capítulos importantes en el marco. Los otros capítulos destinados a ser incluidos en este documento no se incluyeron en este proceso de revisión pública.

Cada período de revisión pública duró dos o tres semanas. Los períodos de revisión se anunciaron ampliamente: cada organización en el comité directivo envió un correo electrónico o publicó un anuncio en su blog; se pidió a cada estado que participaba en el desarrollo que mantuviera un grupo de enfoque; se pidió a cada asesor que revisara el marco; se enviaron correos electrónicos a docenas de organizaciones en ciencias de la computación y educación; y el marco se envió por correo electrónico a los cientos de miembros de la lista de correo electrónico del Marco de Ciencias de la Computación K–12.

Las preguntas en el formulario de revisión en línea diferían ligeramente para cada revisión, ya que el tipo de comentarios que los escritores necesitaban cambiaba a medida que se desarrollaban las declaraciones. Los tres formularios de revisión pidieron a los revisores que calificaran su impresión general del borrador (eligiendo entre excelente, muy bueno, bueno, justo y pobre). En cada formulario de revisión, se pidió a los revisores que calificaran cada declaración conceptual y práctica en dos criterios: si era claro y entendible para un novato en ciencias de la computación y si era importante que todos los estudiantes lo supieran que la formulación de las preguntas varió ligeramente entre los períodos de revisión. También se proporcionaron cuadros de comentarios abiertos en todo el formulario para que los revisores resaltaran las fortalezas y dieran sugerencias para mejorar. Además, el segundo formulario de revisión solicitó a los revisores que calificaran la progresión de cada práctica y la declaración de concepto sobre si era apropiado para el desarrollo de la banda de grado. El tercer formulario de revisión solicitó a los revisores que calificaran cada resumen del concepto, resumen del subconcepto, resumen de la práctica y progresión de la práctica sobre si el nivel de detalle era apropiado para comprender el resumen o la progresión.

A lo largo del proceso de redacción, los equipos de redacción revisaron las declaraciones conceptuales utilizando una lista de criterios. La segunda revisión pública solicitó a los revisores que calificaran el marco utilizando la misma lista:

- **Esencial**: ¿Es esta una idea core de las ciencias de la computación? ¿Es importante y esencial para todos los estudiantes? ¿Cómo se convierte en una persona alfabetizada computacionalmente? ¿Qué beneficio tiene para la persona y la sociedad?

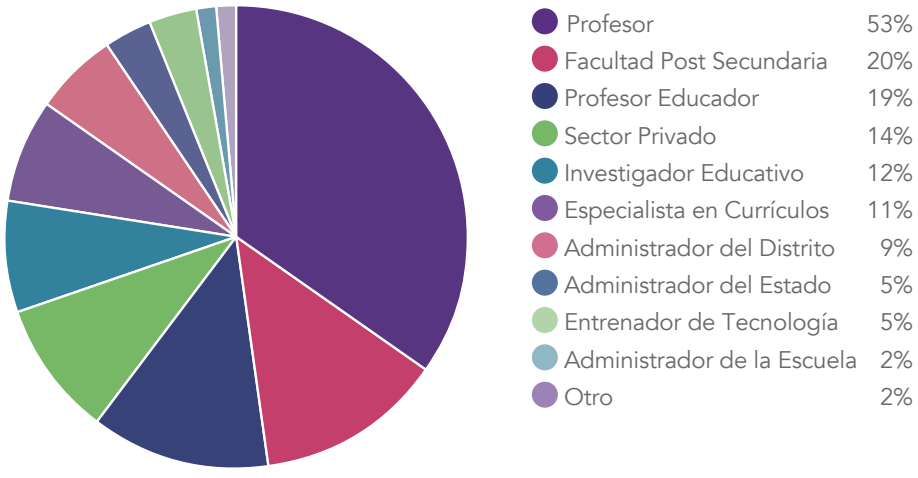
- **Potente en la aplicación**: ¿Es útil conocer el concepto o realizar la práctica? ¿Es útil para resolver problemas, útil para iluminar otras ideas en adelante y útil para comprender un conjunto de conocimientos más amplio? ¿Provoca extensiones, fomenta conexiones interdisciplinarias y muestra potencial para una amplia gama de aplicaciones?

- **Relevante y claro**: ¿La declaración es comprensible para los profesores y relevante para los estudiantes? ¿Los principiantes en ciencias de la computación sentirán que la declaración del concepto es accesible/atractiva?
- **Diversos**: ¿Qué tan bien describen las declaraciones del marco de una visión diversa, equitativa y accesible de las ciencias de la computación?
- **Informado por la investigación**: ¿La declaración es informada por la investigación? ¿Cómo se puede revisar la declaración para reflejar la investigación de la educación en las ciencias de la computación? ¿La declaración apunta a posibles áreas de investigación?
- **Adecuado para el desarrollo**: ¿es apropiado para el desarrollo y adecuado para la escuela secundaria?
- **Interdisciplinario**: ¿La declaración del marco es útil y aplicable fuera del dominio de las ciencias de la computación? ¿Hay oportunidades para hacer conexiones interdisciplinarias? ¿Complementa conceptos y prácticas en matemáticas, ciencias, etc.?
- **Promover la preparación universitaria y profesional**: ¿Qué tan bien contribuyen los conceptos y prácticas a la preparación profesional y universitaria?

En general, 152 personas presentaron revisiones (40 de las cuales también participaron en una revisión grupal), y más de 306 individuos participaron en una revisión grupal. Se realizaron más de 50 revisiones grupales durante los tres períodos de revisión. Estas revisiones incluyeron representación de 38 estados de EE. UU., 1 territorio de EE. UU. Y 7 ubicaciones internacionales.

La Figura A.1 muestra el desglose general de las ocupaciones informadas por los revisores. Al final de cada período de revisión, el equipo de desarrollo revisó todos los aportes, identificó los temas principales, analizó las calificaciones y proporcionó recomendaciones al equipo de redacción en función de los datos. El equipo de redacción tuvo acceso a los temas, las recomendaciones y todos los c

Figura A.1: Ocupaciones de los revisores



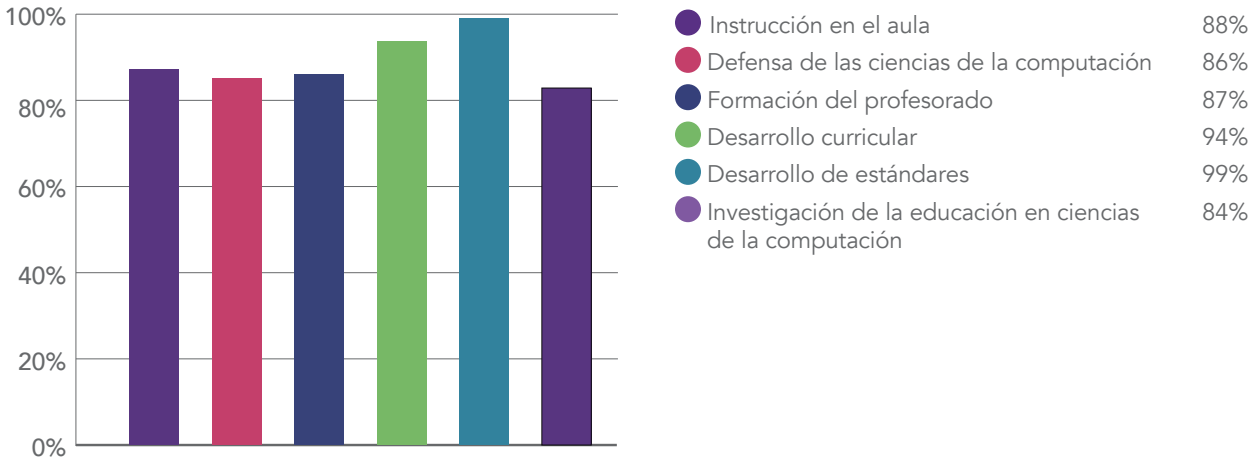
Las siguientes tres secciones describen los temas generales de retroalimentación que se aplicaron al marco en general, los principales temas que se aplicaron a los conceptos core y los principales temas para las prácticas. Aunque los escritores examinaron todos los comentarios, se les dio licencia para decidir cómo se deben abordar los comentarios. Aplicaron los datos que se presentaron de las diversas revisiones, junto con su juicio profesional, para determinar sus revisiones. En algunos casos, los escritores decidieron no abordar, o solo abordar de manera menor, los comentarios sugeridos. Esas decisiones y razones se incluyen a continuación.

Temas generales de retroalimentación

Retroalimentación positiva

Además de proporcionar comentarios constructivos, se pidió a los revisores que comentaran sobre las fortalezas del marco. Los escritores utilizaron comentarios positivos como guía para ayudarlos a comprender mejor lo que era ideal en términos de estructura, tamaño de grano, voz y estilo de escritura general. Los temas surgieron de los comentarios que apuntaban a que el marco estaba bien desarrollado, era integral y una buena progresión para el aprendizaje K-12. En los períodos de revisión dos y tres, las personas escribieron acerca de cuánto mejoró el marco desde el primer período de revisión, que agregaron ejemplos que hicieron que el contenido fuera más fácil de entender y que los cambios en el lenguaje hicieron que el documento fuera más fácil de entender. Los revisores también consideraron que el marco abarca los conceptos y prácticas correctos y apreciaron que incluye mucho más que simplemente "codificación". Otros revisores escribieron acerca de cuánto se necesita un documento como este en la educación en ciencias de la computación y estaban entusiasmados con el esfuerzo general. Los altos porcentajes de revisores también indicaron que creían que el marco sería útil de varias maneras (ver Figura A.2).

Figura A.2: Respuestas de la encuesta sobre la importancia del marco



Durante el período de la tercera revisión, también se pidió a los revisores que informaran sobre la utilidad general del marco en términos de lo que podría informar. Los revisores fueron abrumadoramente positivos sobre los diferentes usos potenciales del marco. Además, cuando se les preguntó sobre su impresión general del marco, el 92% de los revisores lo calificaron como excelente, muy bueno o bueno, lo que demuestra que era un documento muy favorecido. Si bien hubo comentarios positivos sobre el documento durante todo el proceso, los revisores también ofrecieron comentarios constructivos extensos con el interés de mejorar continuamente el marco con cada ciclo de revisión.

Descripción general de la retroalimentación constructiva

Los revisores también indicaron varias áreas de preocupación que se aplicaban tanto a los conceptos como a las prácticas. Estas áreas de preocupación caen en tres categorías: mecánica e idioma, contenido y uso:

- Mecánica y lenguaje
  - Lenguaje demasiado técnico.
  - Voz inconsistente.
  - Contenido poco claro.
- Contenido
  - Contenido demasiado amplio.
  - Superposiciones de contenido - Falta contenido.
  - El contenido no es esencial para todos los alumnos.
  - Renombrar los conceptos y prácticas core.
- Uso
  - Audiencia poco clara.
  - La integración y la alineación con otras disciplinas no son suficientes.

Las siguientes secciones describen cada una de estas áreas de preocupación y la respuesta del equipo de redacción.

Lenguaje demasiado técnico

En los tres períodos de revisión, los comentaristas criticaron varios términos utilizados en el marco. Muchos revisores escribieron que el vocabulario era demasiado técnico y confuso, particularmente para alguien que es un novato en ciencias de la computación. En ocasiones se utilizaron siglas que algunos revisores desconocían. También se expresó cierta confusión sobre los términos que tienen diferentes significados en las ciencias de la computación que en otros campos. Debido a que el marco estaba destinado a principiantes en ciencias de la computación, muchos revisores cuestionaron si la terminología técnica era apropiada. Algunos revisores consideraron que el lenguaje no era atractivo, y los revisores frecuentemente pidieron que se definieran los términos.

Por otro lado, algunos revisores querían ver términos de las ciencias de la computación en el marco. Algunos de los comentaristas con conocimientos de las ciencias de la computación cuestionaron por qué los escritores parecían estar usando otras palabras para evitar usar un término, en lugar de simplemente usar el término en sí.

Respuesta

Los escritores intentaron encontrar un equilibrio entre el uso de términos que son esenciales para comprender las ciencias de la computación y eliminar toda jerga, acrónimos o términos demasiado técnicos o avanzados. El equipo de redacción revisó cada término técnico y eliminó los que no eran necesarios para comprender el marco. Los términos que se consideraron esenciales para las ciencias de la computación se usaron en el marco y se definieron en un **Glosario** (ver **Apéndice C**). El borrador del glosario se proporcionó durante los períodos de revisión segundo y tercero. Los escritores también consideraron el nivel de lectura, y el equipo de desarrollo probó el nivel de lectura de las declaraciones después de la segunda y tercera revisión. Los escritores y un editor eliminaron lenguaje potencialmente confuso y usaron palabras simples siempre que fue posible. Además, parte de la terminología de las ciencias de la computación más compleja se describió o explicó en una declaración de concepto o práctica.

Voz inconsistente

En los tres períodos de revisión, los revisores comentaron la falta de una voz consistente para las declaraciones del marco. La retroalimentación varió de muy general (por ejemplo, las descripciones generales se escriben de manera diferente, algunas declaraciones conceptuales se leen de manera muy diferente de otras declaraciones conceptuales) a muy específicas sobre la elección del idioma (por ejemplo, el tema de las declaraciones como "grupos", "estudiantes" o "personas"; usando "computadoras" versus "dispositivos computacionales"). Un área de preocupación acerca de las declaraciones de práctica fue que algunas declaraciones sonaban como objetivos de final de grado 12, pero otras parecían declaraciones de objetivos generales que podrían aplicarse en K-12.

Respuesta

A lo largo del proceso, el equipo de desarrollo creó y refinó las pautas en torno al estilo y la estructura de la escritura. Se incorporó un editor al equipo de redacción para crear una guía de estilo para la elección coherente del idioma y la estructura de las oraciones. El editor también editó todos los conceptos y prácticas antes de cada uno de los siguientes períodos de revisión y antes del lanzamiento final.

Pequeños equipos de redacción abordaron temas de voz consistente en conceptos, prácticas y descripciones generales. Para cada uno (conceptos, prácticas y descripciones generales), un equipo de tres escritores y un miembro del equipo de desarrollo hicieron que el estilo y el formato del lenguaje fueran consistentes.

El equipo de redacción de prácticas aclaró que todas las declaraciones de práctica deberían reflejar los objetivos "para el final del grado 12", y el equipo revisó todas las declaraciones para reflejar este punto final. El equipo de prácticas también revisó las descripciones generales para alinear el contenido y el uso de los verbos con las declaraciones y progresiones.

Contenido poco claro

Muchos comentarios de las revisiones abordaron la claridad en las declaraciones y prácticas conceptuales. Los revisores querían más ejemplos o explicaciones para aclarar las declaraciones. Algunos comentarios transmitían que los revisores no estaban seguros sobre la intención de una declaración, o que malinterpretaron la intención. La mayoría de estos comentarios se hicieron sobre las declaraciones en las bandas de grados K–2 y 3–5. Los revisores de las prácticas apreciaron los ejemplos, diciendo que los ejemplos dan vida a las prácticas. Otras prácticas fueron menos claras y los revisores pidieron más ejemplos.

Respuesta

El equipo de redacción agregó profundidad a las declaraciones y prácticas. Algunos borradores del marco tenían declaraciones que eran tan concisas que la intención ya no era clara. Los escritores agregaron más descripción para que el tema de la declaración fuera claro. Las declaraciones conceptuales de K-5 en particular, aunque originalmente se escribieron para ser accesibles, recibieron comentarios que los estudiantes ya conocen o ya están aprendiendo estas ideas. El equipo de redacción utilizó esta retroalimentación, junto con los puntajes del nivel de lectura expresado por Lexile para estas declaraciones, para ampliar las ideas de las ciencias de la computación. El equipo de redacción de prácticas realizó muchas ediciones para aclaración. Por ejemplo, en respuesta a los comentarios sobre la complejidad y la falta de claridad en torno a la abstracción, los escritores intentaron encontrar un equilibrio entre las necesidades de un lector novato y las necesidades de un lector experto. Los escritores aclararon la definición de abstracción y ampliaron los ejemplos para contextualizar la práctica de la abstracción. Los equipos de redacción de conceptos y prácticas agregaron ejemplos y aclaraciones en el material descriptivo de cada concepto y en las progresiones de las declaraciones de práctica. También eligieron incluir solo los ejemplos más destacados y específicos y describir cómo cada ejemplo reflejaba el concepto o la declaración práctica.

Contenido demasiado amplio

Los revisores señalaron que varios subconceptos y declaraciones conceptuales eran muy amplios e incluían muchas ideas diferentes. Los revisores que tenían experiencia en la redacción de estándares estaban preocupados porque el concepto y las declaraciones de práctica no se traducirían bien a las normas. Otros comentarios decían que la redacción era confusa y que se necesitaba más claridad para comprender la intención de cada declaración.

Respuesta

El equipo de redacción redujo el enfoque de cada concepto y declaración práctica y priorizó los temas e ideas clave. Cada declaración fue revisada para enfocarse en ideas más simples y menos. Por ejemplo, en el

concepto core de Algoritmos y Programación, el desarrollo del programa originalmente contenía muchos temas diferentes, pero finalmente se centró en el proceso iterativo de diseño, implementación y revisión de programas, incluida la consideración de diversos usuarios y equipos diversos. Otro ejemplo está en el concepto core de Sistemas de Computación, en el que un subconcepto centrado en el software del sistema o sistemas operativos se combinó con el subconcepto de hardware y software.

Superposiciones de contenido

Algunos revisores identificaron áreas de superposición entre subconceptos o entre declaraciones de práctica. Por ejemplo, muchos revisores identificaron áreas de superposición entre las prácticas de Fomentar una cultura de computación inclusiva y la colaboración en torno a la computación o entre la práctica de creación de artefactos computacionales y muchas otras prácticas.

Respuesta

El equipo de redacción revisó cuidadosamente los comentarios específicos sobre las áreas de superposición e intentó minimizar esta superposición.

Algunas ideas se eliminaron por completo de un concepto o práctica core y se dejaron en otro. Cuando se dejó un tema en varias secciones, se hizo un esfuerzo para asegurar que solo se incorporara el aspecto específico del tema abordado en el concepto o práctica. Por ejemplo, en respuesta a la retroalimentación sobre la representación de datos que aparece tanto en los conceptos core de Datos y Análisis como en Algoritmos y Programación, las ideas superpuestas se conservaron en las declaraciones de concepto de Datos y Análisis, y las declaraciones de Algoritmos y Programación se reformularon para enfocarse en variables y tipos de datos.

El equipo de prácticas utilizó la retroalimentación específica del revisor para distinguir más claramente entre las prácticas. Por ejemplo, la redacción que se centró en la colaboración en equipos se eliminó de Fomentar una cultura de computación inclusiva para delinear más claramente su límite con Colaborando en torno a la computación.

Por otro lado, el equipo de redacción consideró necesario parte de la superposición. Por ejemplo, algunos temas que se consideraron esenciales para múltiples conceptos, como la interacción humano-computadora y la privacidad y seguridad, se convirtieron en conceptos transversales. Para las prácticas, algunos términos e ideas de Fomentar una cultura informática inclusiva se dejaron intencionalmente en otras prácticas; Esta superposición fue deliberada para enfatizar la importancia de la diversidad en todas las prácticas. Del mismo modo, el equipo de redacción recibió comentarios contradictorios sobre la Creación de artefactos computacionales y otras prácticas: los revisores pensaron que esta práctica era esencial, pero estaban preocupados por la superposición con las otras prácticas. Los escritores intentaron minimizar la superposición enfocando esta práctica en el propósito de modificar un artefacto.

Contenido perdido

Algunos revisores sugirieron contenido que aún no estaba incluido en los conceptos y prácticas del marco.

Respuesta

Los escritores del marco respondieron a estas sugerencias de varias maneras distintas.

En algunos casos, cuando el contenido aún no aparecía en las declaraciones, los escritores estuvieron de acuerdo con los revisores. Por ejemplo, basándose en los comentarios para incluir la ciberseguridad, el equipo de redacción contó con el asesoramiento de la Iniciativa Nacional para la Educación en Ciberseguridad y el Centro de Innovación Cibernética sobre las ideas centrales de la educación en ciberseguridad y creó un subconcepto de ciberseguridad. Otros ejemplos incluyen el acoso cibernético, los componentes computacionales de la ciudadanía digital, las carreras en ciencias de la computación y los bits. En otros casos, los revisores querían ver un mayor enfoque en temas específicos. Por ejemplo, en base a la retroalimentación, los escritores ampliaron el enfoque de los sistemas en el concepto core de Sistemas de Computación y decidieron incluir el modelado en la práctica Desarrollando y Usando Abstracciones para transmitir cómo interactuar con modelos y simulaciones y cómo poder contribuir a una parte del proceso de hacer uno.

Otra sugerencia de los revisores fue asegurar que la declaración no restringiera o limitara contenido que se podría enseñar. Por ejemplo, algunos revisores creían que las referencias a la programación orientada a objetos y el orden de presentación de ellas en las declaraciones Algoritmos y Programación eliminó la programación funcional como posible paradigma en los niveles de secundaria y preparatoria. Los escritores revisaron las declaraciones para permitir el uso de la programación funcional. En el material descriptivo, los escritores proporcionaron ejemplos que fueron descritos como formas "posibles" de hacer las cosas que las formas esperadas de hacer algo.

Se consideraron otras sugerencias, pero finalmente los escritores decidieron que el contenido sugerido iba más allá de las ciencias de la computación que todos los alumnos deberían saber. Por ejemplo, los revisores sugirieron incluir en las declaraciones de práctica ideas como máquinas inteligentes y gráficos por computadora, analizar artefactos hechos por otros y contribuir a las comunidades de código. El equipo de redacción determinó que no todos los estudiantes deberían estar obligados a hacer esto o que algunas de las ideas estaban fuera del alcance de las prácticas y necesitarían un lenguaje excesivamente técnico.

Contenido no esencial para todos los alumnos

Algunas revisiones señalaron declaraciones de conceptos o prácticas particulares como no esenciales para que todos los estudiantes aprendan o no sean core y centrales para la informática. Otras revisiones señalaron declaraciones particulares como demasiado avanzadas para una clase de secundaria.

Respuesta

En el transcurso de los tres borradores, el número de declaraciones conceptuales se redujo drásticamente. Los escritores con experiencia en cada banda de grado revisaron todas las declaraciones conceptuales para esa banda de grado para determinar si cada declaración era (1) esencial para todos los estudiantes y (2) apropiada para el desarrollo de la banda de grado dada. El equipo de redacción reconsideró todas las declaraciones, centrándose en las marcadas por los revisores.

Debido a que el marco describe conceptos y prácticas para todos los estudiantes, los escritores eliminaron el contenido que se enseñaría en un curso especializado de secundaria. Los escritores también cuestionaron si cada declaración era de igual importancia y eliminaron partes de declaraciones que se consideraron no esenciales para todos los estudiantes. Por ejemplo, los escritores eliminaron ideas en el concepto core de Algoritmos y Programación que los revisores dijeron que no eran necesarios para que todos los estudiantes aprendan en la escuela secundaria, como la recursividad y la programación orientada a objetos, e hicieron estas ideas opcionales. En las prácticas, los escritores consideraron cuidadosamente hasta qué punto se incluyó el modelado.

Renombrar los conceptos y prácticas core

Algunos revisores estaban preocupados por el mensaje enviado por los nombres de los conceptos core, subconceptos y prácticas. Estuvieron de acuerdo con la división de los cinco conceptos core, por ejemplo, pero criticaron la denominación por ser demasiado técnica o no representar el poder de las ciencias de la computación.

Respuesta

Los escritores consultaron con los asesores y se refirieron a comentarios específicos para modificar los nombres de los conceptos, prácticas y subconceptos básicos. Los escritores hicieron ediciones reflexivas y precisas de estos nombres para mayor claridad y para enfatizar las poderosas ideas de las ciencias de la computación que cada una abarcaba. Por ejemplo, las Redes y Comunicación se convirtieron en Redes e Internet, los datos y la información se convirtieron en Datos y Análisis, y las Pruebas y Refinación iterativa se convirtieron en Pruebas y Refinación de Artefactos Computacionales. Para los subconceptos, un ejemplo ilustrativo se encuentra en el concepto core Impactos de la computación. Después de los primeros comentarios sobre los subconceptos, el equipo de redacción reestructuró los subconceptos en agrupaciones más naturales y cambió los nombres para que fuera más fácil de leer y claro

de escribir, con el entendimiento de que algunos lectores pueden no tener experiencia en ciencias de la computación, lo que influyó en las revisiones del lenguaje. El equipo de redacción de prácticas aclaró la audiencia y la intención de las prácticas en el capítulo Prefacio de las **Prácticas**. Aunque, las prácticas están escritas para un público adulto profesional, las prácticas en sí están dirigidas a todos los estudiantes. Por lo tanto, aunque un informático no pueda participar en la práctica, todos los estudiantes, independientemente de su futura profesión, deberían poder participar en las prácticas.

La integración y la alineación con otras disciplinas no son suficientes

En los tres períodos de revisión, los revisores querían ver conexiones explícitas con otras disciplinas. Los comentarios sobre las prácticas indicaron que los revisores querían ver más conexiones con las prácticas en ciencias, ingeniería, alfabetización y matemáticas. Por otro lado, los revisores también mencionaron que parte del contenido del marco ya se enseñó en otras disciplinas.

Respuesta

Los escritores intentaron incluir conexiones interdisciplinarias para cada declaración de concepto o área de concepto, pero se dieron cuenta de que esto estaba más allá del alcance del marco y la experiencia de los escritores. Las conexiones interdisciplinarias y la enseñanza de las ciencias de la computación integradas en otras áreas de contenido se abordan ampliamente en el capítulo de la Guía de implementación. Los materiales complementarios al marco relacionados con las conexiones interdisciplinarias podrían publicarse en el futuro o podrían crearse a nivel de estándares o plan de estudios.

Los escritores también revisaron las declaraciones que fueron etiquetadas por los comentarios como similares al contenido enseñado en otras materias. Aunque los escritores asumieron que la superposición con otro contenido era deseable (y podría ayudar en la integración), los escritores decidieron enfocar cada declaración en el contenido de las ciencias de la computación. Los escritores revisaron las declaraciones para enfatizar el aspecto computacional de la declaración. Por ejemplo, el equipo de redacción modificó las declaraciones K–2 en el concepto core de Datos y Análisis para centrarse en la herramienta digital y la recopilación automatizada, diferenciando el contenido de los datos en matemáticas.

Las prácticas se revisaron para centrarse más específicamente en el contexto informático de cada práctica, en lugar de las habilidades más generales del siglo XXI. Sin embargo, existe una superposición natural en la forma en que las prácticas de pensamiento computacional se alinean con las prácticas de diseño de ingeniería o prácticas matemáticas. El capítulo **Prácticas** incluye un diagrama (Figura 5.2) que muestra algunas de estas conexiones potenciales.

Temas principales solo para conceptos

Algunos de los comentarios de los revisores se aplicaron solo a los conceptos.
Los temas principales de estos comentarios fueron:

- los conceptos transversales necesitaban ser revisados, y
- las progresiones del subconcepto no fueron consistentes.

Conceptos transversales necesitan ser revisados

Los comentarios del primer período de revisión solicitaron la inclusión de conceptos transversales. Después de presentar esta lista en la próxima revisión, los comentarios sugirieron que se revisara la lista.

Respuesta
Aunque los conceptos transversales fueron una herramienta de escritura desde el inicio del proyecto, la lista no se presentó durante la primera revisión pública. Después de recibir comentarios que solicitaban conceptos transversales, el borrador de la lista se compartió durante la próxima revisión pública. Sin embargo, de esta lista preliminar, algunos de los conceptos transversales originales finalmente parecían encajar mejor en un concepto central único, mientras que algunas ideas originalmente ubicadas en un concepto central único se determinaron que eran aplicables a conceptos centrales múltiples, lo que sugiere que eran transversales. Por ejemplo, la interacción hombre-computadora fue originalmente un subconcepto en Impactos de la informática, pero se convirtió en un concepto transversal debido a la superposición con otras áreas conceptuales. Ética y Seguridad fueron originalmente un concepto transversal, pero finalmente la ética se incorporó a Impactos de la Computación, y el concepto transversal se convirtió en Privacidad y Seguridad.
Después del segundo período de revisión, un pequeño equipo de asesores y escritores leyó cuidadosamente el borrador de las declaraciones, los comentarios de la revisión y la literatura relevante para identificar la lista final de conceptos transversales. La lista completa se incluye en el prefacio del capítulo **Conceptos**.

Las progresiones del subconcepto no fueron consistentes

Durante el segundo y tercer período de revisión, los revisores comentaron los "saltos", o aumentos en la sofisticación y el contenido, de un punto final de la banda de grado al siguiente dentro de una progresión. Algunos de los comentarios fueron que los saltos eran demasiado grandes o pequeños para el subconcepto especificado o que los saltos tenían un tamaño inconsistente en todo el marco. Algunos revisores escribieron que el contenido en una banda de grado en particular no es esencial como parte de la progresión y que esto combinado con un pequeño salto no asegurara la inclusión de la declaración del concepto.

Respuesta
Después del segundo y tercer período de revisión, los escritores examinaron de cerca las progresiones en las bandas de calificación y escribieron descripciones generales de los subconceptos para describir la progresión. Cuando fue necesario, los escritores insertaron un lenguaje para asegurarse de que había una transición fluida entre las diferentes bandas de grados y que no se agregaba demasiado en cada salto. Por ejemplo, los revisores querían ver

programación como una expectativa en la banda de grado K–2 y la eliminación de la programación funcional de la banda de grado 3–5, cambiando estas progresiones de K–5 en los subconceptos de Algoritmos y Programación. En algunos casos, los escritores consideraron comenzar algunas progresiones de subconcepto en la banda de grados 3–5 o terminar una progresión en la banda de grados 6–8, pero finalmente todas las progresiones abarcaron K–12 en el borrador final.

Algunos subconceptos con saltos más pequeños se condensaron o combinaron para producir menos conceptos en general. Por ejemplo, tres subconceptos dentro de Redes e Internet se combinaron en un solo subconcepto con mayores saltos en la progresión.

Temas principales solo para prácticas

Algunos de los comentarios de los revisores se aplicaron solo a las prácticas. Temas principales de estos comentarios fueron:

- el pensamiento computacional no se enfatizó lo suficiente,
- hubo confusión sobre las prácticas escritas en progresiones en lugar de bandas de calificación, y
- las prácticas se centraron demasiado estrechamente en la programación.

El pensamiento computacional no se enfatizó lo suficiente

En la primera revisión, muchos revisores preguntaron por qué no se incluyó el pensamiento computacional. El pensamiento computacional no se mencionaba explícitamente en las prácticas, y muchos revisores creían que era importante incluir las palabras.

Respuesta
El equipo de redacción de prácticas escribió una sección del capítulo **Prácticas** que explicaba por qué el pensamiento computacional no era una práctica. En cambio, incluyeron cuatro prácticas que llamaron específicamente como prácticas de pensamiento computacional. Dentro de estas prácticas, se incluyeron ideas core para el pensamiento computacional, como la descomposición y la abstracción. El equipo de redacción y el equipo de desarrollo revisaron con frecuencia la forma en que el pensamiento computacional se incluyó en las prácticas.

Hubo confusión sobre las prácticas escritas en progresiones en lugar de bandas de grado

Durante la segunda y tercera revisión, los revisores expresaron confusión sobre la forma en que las prácticas se escribieron de manera diferente a los conceptos. Durante el segundo período de revisión, el lenguaje de las progresiones confundió a los revisores porque parecían estar escritos para las bandas de grado. Los revisores también estaban preocupados de que las prácticas pudieran ignorarse si el lector no podía aplicarlas fácilmente. Algunos revisores querían coherencia entre los conceptos y las prácticas y expresaron una preferencia por las bandas de grados. Por otro lado, muchos otros revisores, incluidos algunos que tenían experiencia en la redacción de estándares, se opusieron a las bandas de grado para las prácticas y alentaron el uso de progresiones.

Respuesta

Al intentar hacer que las prácticas sean lo más utilizables posible, el equipo de redacción escribió las progresiones en el formato que podría ser más útil para los escritores y profesionales de estándares. El equipo de redacción tomó la posición de que la estructura narrativa proporciona flexibilidad e intentar escribir las prácticas en bandas de grado crearía puntos de referencia artificiales. En respuesta a la retroalimentación crítica, las progresiones se volvieron a trabajar para que sean independientes del nivel de grado y para aliviar la confusión sobre la redacción y el lenguaje. Las progresiones incluyeron un punto de partida y un punto final, pero no incluyeron bandas de grado. Los escritores también alinearon cada progresión con la declaración práctica correspondiente y eligieron verbos específicos para reflejar mejor lo que realmente se hace en la práctica.

Las prácticas se centraron demasiado en la programación

Durante el primer período de revisión, los revisores dijeron que las prácticas se centraron demasiado en la programación. Muchos de los ejemplos incluidos en las prácticas fueron específicos de la programación y no ayudaron al lector a establecer conexiones entre las prácticas y otros conceptos core.

Respuesta

La intención de las prácticas es que cada práctica se pueda combinar con declaraciones conceptuales en los cinco conceptos core, por lo que el equipo de redacción intentó aclarar esto con el lenguaje y los ejemplos en las prácticas. El equipo de redacción reformuló ejemplos en múltiples prácticas para asegurarse de que los ejemplos ayudaran a la conexión a conceptos y elementos core de las ciencias de la computación además de la programación. En particular, la Comunicación sobre Computación y la Creación de Artefactos Computacionales se revisaron en función de los comentarios para centrarse en múltiples aspectos de las ciencias de la computación.

Organizaciones que convocaron revisiones

- AccessCS10K
- Achieve
- Association of State Supervisors of Mathematics
- California Department of Education
- Center for Applied Special Technology (CAST)
- Center for Elementary Math and Science Education, University of Chicago
- Change the Equation
- Chicago Public Schools, IL
- Code.org
- CodeCombat
- Codesters
- CodeVA
- Computing At School, United Kingdom
- Capital Region Academies for the Next Economy
- CS Teachers Ann Arbor Public Schools
- Computer Science Teachers Association, New Hampshire
- Computer Science Teachers Association, Delaware
- Computer Science Teachers Association, Minnesota
- CSNYC
- Cyber Innovation Center
- Deer Valley Unified School District, Computer Science Education Committee
- Expanding Computing Education Pathways, Maryland
- Gesellschaft für Informatik e.V., Germany
- Google Inc.
- Green Bay Area Public Schools, WI
- International Association of Privacy Professionals
- Indiana Department of Education
- Johnston County Schools, North Carolina
- Maine Mathematics and Science Alliance
- Maryland State Department of Education
- Massachusetts Computing Attainment Network
- Massachusetts Department of Education
- Microsoft Corporation
- Minnetonka Public Schools, MN
- Mississippi Department of Education
- Museum of Science, Boston
- New Jersey Department of Education
- National Initiative for Cybersecurity Education
- North Carolina Business and Industry—Programming and Engineers
- North Carolina Career and Technical Education Directors
- North Carolina Department of Public Instruction
- Nevada Computer Science Team
- New York City Department of Education, NY
- Ohio Review Team
- Oracle Academy
- Pacific Northwest National Laboratory (PNNL)
- Project Lead The Way
- San Francisco Unified School District (SFUSD), CA
- SRI, Center for Technology in Learning
- Washington Computer Science Learning Standards Advisory Committee
- Washington State Leadership and Assistance for Science Education Reform
- Washington State
- Washington STEM

Apéndice B: Biografías de Escritores y Personal de Desarrollo

Escritores

Julie Alano

Computer Science Teacher, Hamilton Southeastern High School Fishers, Indiana.

Julie Alano enseña ciencias de la computación en Hamilton Southeastern High School. Ella ha expandido el programa de ciencias de la computación desde que comenzó allí como profesora de matemáticas en agosto de 1998. La escuela ahora ofrece cuatro niveles de ciencias de la computación, y está trabajando para incluir las ciencias de la computación en el currículo K-8. Con una maestría en tecnología educativa, Julie también comenzó un equipo de tecnología dirigido por estudiantes en la escuela. En mayo de 2016, Julie fue nombrada Profesora del Año del Distrito de Escuelas del Sureste de Hamilton. Julie es presidenta del Capítulo Hoosier Heartland de la Computer Science Teachers Association (CSTA) después de ayudar a iniciar el grupo. También es miembro del Equipo de Liderazgo de Defensa de las Ciencias de la Computación de CSTA y una Facilitadora de Principios de Ciencias de la Computación de Code.org.

Derek Babb

Computer Science Teacher, Omaha North Magnet High School Omaha, Nebraska.

Derek Babb es profesor de las ciencias de la computación en la escuela secundaria de Omaha North Magnet. Ha enseñado ciencias de la computación durante 11 años en entornos de escuelas secundarias urbanas y suburbanas. Además de escribir para el Marco de Ciencias de la Computación K–12, ha participado en la redacción de estándares de ciencias de la computación para el estado de Nebraska y los distritos escolares locales. Ha estado involucrado en la defensa de las ciencias de la computación a nivel local, sirviendo como miembro fundador y presidente del capítulo de la Asociación de Profesores de las Ciencias de la Computación de Omaha. Derek está comprometido a expandir la educación en ciencias de la computación en su escuela y distrito y espera servir como entrenador y asesor para los nuevos profesores de ciencias de la computación a medida que comienzan.

Julia Bell

Associate Professor of Computer Science, Walters State Community College Morristown, Tennessee.

Julia Bell es profesora asociada de las ciencias de la computación en Walters State Community College. Anteriormente trabajó con Northwest Arkansas Community College en Bentonville, AR. Ha trabajado como directora de programas de redes y analista de sistemas para el Departamento de Policía de Fayetteville, instructora de programación de TN Code Academy, tutora de TN Achieves, diseñadora certificada de Quality Matters y revisora de cursos, y examinadora forense certificada por A.C.E. Recibió el premio de la Facultad del Año 2012 y varios premios de la Facultad Good as Gold de Phi Theta Kappa. Durante dos años, ha trabajado con el campamento de verano de codificación de la Fundación Nicewonger enseñando codificación y trabajo en red a estudiantes de secundaria y preparatoria, como escritora de los Estándares Interinos de Ciencias de Computación CSTA K–12, revisada en 2016, y como presentadora móvil de la Fundación Nicewonger. Los intereses de investigación de Julia incluyen ciberseguridad, impactos de ciberseguridad en niños y redes NSX del futuro.

Tiara Booker-Dwyer

Education Program Specialist, Maryland State Department of Education Baltimore, Maryland.

Tiara Booker-Dwyer es especialista en programas de educación para el Departamento de Educación del Estado de Maryland (MSDE). En este puesto, proporciona liderazgo a los sistemas escolares locales y a las instituciones postsecundarias para planificar, desarrollar e implementar programas educativos de las ciencias de la computación, ingeniería y tecnología. Desarrolla, coordina e instala experiencias de aprendizaje profesional y asiste en iniciativas departamentales relacionadas con la reforma escolar y la educación en ciencias, tecnología, ingeniería y matemáticas. Antes de unirse a MSDE, Tiara fue directora de programa para la Mesa Redonda de Negocios de Maryland, donde desarrolló alianzas estratégicas y dirigió grupos de partes interesadas (Stakeholders) en la implementación de programas diseñados para preparar a los estudiantes para futuros mercados laborales. Tiara comenzó su carrera realizando investigaciones en neurociencia en Johns Hopkins antes de hacer la transición a la educación, donde lidera esfuerzos colaborativos para implementar experiencias de aprendizaje de las ciencias de la computación de alta calidad en todo el estado.

Leigh Ann DeLyser

Director of Education and Research, CSNYC New York, New York.

Leigh Ann DeLyser es la directora de educación e investigación de la Fundación de Nueva York para la Educación en Ciencias de la Computación (CSNYC). En este rol, Leigh Ann está trabajando para expandir las ciencias de la computación a todas las escuelas del sistema de escuelas públicas de la ciudad de Nueva York. CSNYC es el socio privado en la iniciativa de $80 millones que requiere que cada escuela ofrezca una unidad de las ciencias de la computación a cada estudiante en las escuelas públicas. Es coautora del informe Running on Empty, un análisis de estándares de las ciencias de la computación en 50 estados. Antes de obtener su doctorado en ciencias de la computación y psicología cognitiva de la Universidad Carnegie Mellon, Leigh Ann era profesora de ciencias de la computación y matemáticas en la escuela secundaria y miembro de la junta directiva de la Asociación de Profesores de Ciencias de la Computación durante dos períodos. También ayudó a comenzar las Academias de Ingeniería de Software en la ciudad de Nueva York como prueba de concepto de que todos los estudiantes podían aprender las ciencias de la computación.

Caitlin McMunn Dooley

Deputy Superintendent for Curriculum and Instruction, Georgia. Department of Education Associate Professor, Georgia State University Atlanta, Georgia.

Caitlin McMunn Dooley es la superintendente adjunta de currículo e instrucción para las escuelas públicas de Georgia y profesora asociada en la Universidad Estatal de Georgia. Su investigación sobre el aprendizaje de los niños y los profesores, el desarrollo de la alfabetización digital y el pensamiento computacional se ha publicado en más de 50 artículos, capítulos y editoriales. El último proyecto financiado por la Fundación Nacional de Ciencias de Caitlin estudia cómo integrar la informática en el currículo en los grados 3–5. Caitlin promueve la integración de la informática como una parte esencial del aprendizaje académico K-12 y del desarrollo de la alfabetización digital. Caitlin enseñó en la primera infancia y los grados de primaria en Virginia antes de convertirse en maestra educadora, profesora, madre, investigadora y líder escolar.

Diana Franklin

Director of Computer Science Education, UChicago STEM Ed Chicago, Illinois.

Diana Franklin es la directora de educación en ciencias de la computación en UChicago STEM Ed. Ha enseñado computación a nivel universitario durante 14 años como profesora titular en la Universidad de California, Santa Bárbara y como profesora asociada en la Universidad Politécnica del Estado de California, San Luis Obispo. Su investigación se centra en comprender cómo los niños aprenden conceptos de computación en la escuela primaria para diseñar entornos de aprendizaje y planes de estudio. Recibió el premio CAREER de la National Science Foundation, el premio de tutoría del centro nacional para mujeres y tecnología de la información y tres premios de enseñanza. Es autora de una guía práctica para la diversidad de género para la facultad de CS, de Morgan Claypool.

Dan Frost

Senior Lecturer, University of California, Irvine, Irvine, California.

Dan Frost ha mantenido un gran interés en la educación en ciencias de la computación K-12 durante las dos décadas que ha enseñado ciencias de la computación a nivel universitario. Su artículo de SIGCSE (Grupo de interés especial sobre educación en ciencias de la computación) de 1997, sobre las ciencias de la computación de cuarto grado, basado en muchos años de enseñanza e investigación en el aula, contribuyó a la reciente mejora de la educación en ciencias de la computación en los niveles primario y secundario. De 1997 a 1999, Dan presidió el comité de la Asociación de Docentes de las Ciencias de la Computación que escribió un plan de estudios modelo para ciencias de la computación K–12: objetivos y esquemas de nivel 1. Fue el investigador principal de una subvención de la National Science Foundation que entrelazó las ciencias de la computación, el diseño de juegos y la educación cultural para los estudiantes de secundaria de los indios americanos, quienes crearon juegos que relataban historias tradicionales y prácticas culturales.

Mark A. Gruwell

Co-Facilitator, Iowa STEM Council Computer Science Workgroup Estherville, Iowa.

Mark A. Gruwell co-facilita el Grupo de Trabajo de Ciencias de la Computación del Consejo STEM de Iowa, que aboga y promueve iniciativas de educación en ciencias de la computación K-12 en Iowa. Mark comenzó la programación de computadoras en la escuela secundaria y continuó la programación en la universidad, donde logró el reconocimiento de la Asociación de Bandmasters de Florida por crear BandBase, una aplicación que automatiza la programación y otros procesos para festivales de música del distrito y del estado. Mientras se desempeñaba como director académico de Iowa Lakes Community College, Mark dirigió esfuerzos para crear e implementar

el programa de dos años de diseño y desarrollo de juegos de computadora de la universidad. Además de enseñar campamentos de computación de verano, Mark es un emprendedor que diseña aplicaciones computacionales que ayudan a las universidades a asesorar a los estudiantes, programar y acreditar profesores adjuntos y acreditarlos.

Maya Israel

Assistant Professor, University of Illinois at Urbana Champaign, Champaign, Illinois.

Maya Israel es profesora asistente en la Facultad de Educación de la Universidad de Illinois en Urbana Champaign. Sus principales áreas de especialización incluyen el apoyo a estudiantes con discapacidades y la participación significativa de otros estudiantes con dificultades en ciencias, tecnología, ingeniería y matemáticas (STEM), con énfasis en el pensamiento computacional y la programación de computadoras. Investiga modelos y tecnologías de instrucción accesibles que promueven la participación de los estudiantes, la resolución colaborativa de problemas y la persistencia. Actualmente, Maya es investigadora co-principal de una beca STEM + C de la National Science Foundation para crear trayectorias de aprendizaje que alinean el pensamiento computacional con la instrucción matemática. Ha publicado en revistas de alto rango como Niños Excepcionales, Revista de Investigación sobre Tecnología en Educación, Revista de Investigación en Enseñanza de las Ciencias y Computadoras y Educación.

Vanessa Jones

Instructional Technology Design Coach, Austin Independent School District Austin, Texas.

Vanessa Jones es una entrenadora de diseño de tecnología educativa para el Distrito Escolar Independiente de Austin. Es facilitadora de Code.org Texas y ha capacitado a cientos de educadores en conceptos básicos de las ciencias de la computación. Fue nombrada una de los 20 educadores más inspiradores de Intel para la educación y ha presentado en numerosas conferencias nacionales y estatales, como la Sociedad Internacional de Tecnología en Educación, mostrando iniciativas de las ciencias de la computación. Vanessa ha escrito varias subvenciones para enriquecer la infusión de las ciencias de la computación en el aula de primaria y secundaria. Es miembro de la organización CS4TX (Computer Science for Texas), y su pasión es continuar desarrollando una comunidad de estudiantes de las ciencias de la computación para aprender algo nuevo todos los días. Ella cree que todos los estudiantes deberían tener acceso para comprender los conceptos básicos de las ciencias de la computación y que las ciencias de la computación son el gran ecualizador de equidad.

Richard Kick

Mathematics and Computer Science Teacher, Newbury Park High School Newbury Park, California.

Richard Kick enseña matemáticas y las ciencias de la computación en la Newbury Park High School. Rich obtuvo un título en educación matemática de la Universidad de Illinois en Urbana Champaign y una maestría en matemáticas de la Universidad Estatal de Chicago. Enseñó ciencias de la computación Advanced Placement® (AP) utilizando Pascal a partir del primer año de ciencias de la computación AP, seguido de C ++ y luego Java. Después de trabajar como programador de C ++ en Fermi National Accelerator Laboratory, Rich se desempeñó como lector de exámenes de College Board, líder de mesa, líder de preguntas y miembro del Comité de Desarrollo de Pruebas de Ciencias de la Computación. Es cinco veces instructor piloto de Principios de Ciencias de la Computación y actualmente es copresidente del Comité de Desarrollo de Principios de Ciencias de la Computación.

Heather Lageman

Executive Director of Leadership Development, Baltimore County Public Schools Towson, Maryland.

Heather Lageman sirve como directora ejecutiva de desarrollo de liderazgo para las Escuelas Públicas del Condado de Baltimore en la Oficina de Desarrollo Organizacional. Es presidenta de la filial Learning Forward Maryland, presidenta electa de la Learning Forward Foundation y vicepresidenta de la filial de Maryland de la Asociación para la Supervisión y el Desarrollo del Currículo. Heather se ha desempeñado como directora del plan de estudios del Departamento de Educación del Estado de Maryland (MSDE) y administró la implementación a nivel estatal del Programa de Inducción de Profesores de Maryland. Durante Race to the Top, se desempeñó como directora de la agencia de educación local Race to the Top para Maryland y administró aspectos programáticos y fiscales de los proyectos del distrito. Heather anteriormente se desempeñó como especialista en MSDE administrando el Título IIA No Child Left Behind y proporcionando liderazgo para los programas y políticas de desarrollo profesional de profesores estatales, así como los coordinadores de desarrollo profesional. Antes de eso, se desempeñó como especialista en la Oficina Secundaria de Artes del Lenguaje Inglés de MSDE, donde sus responsabilidades incluían el desarrollo e implementación del plan de estudios del condado, evaluaciones y desarrollo profesional. Heather se dedica a apoyar el aprendizaje profesional y el desarrollo de educadores inspirados e innovadores.

Todd Lash

Doctoral Student/Contributing Member, University of Illinois Doctoral Student/ CSTA K–8 Task Force Champaign, Illinois.

Todd Lash es un educador de primaria de 17 años. Habiendo disfrutado el tiempo como profesor de aula y especialista en medios de la biblioteca escolar durante los últimos tres años, Todd trabajó como entrenador de instrucción para las ciencias de la computación. Actualmente, estudiante de doctorado de primer año en la Universidad de Illinois, Todd formó parte del equipo de los Estándares Interinos de Ciencias de la Computación CSTA K–12, Revisado 2016 y está activo en la Fuerza de Tareas K–8 de la Asociación de Profesores de Ciencias de Computación (CSTA). Como parte de una beca STEM + C de la National Science Foundation, Todd es parte de un equipo que trabaja para desarrollar trayectorias de aprendizaje de las ciencias de la computación a través de un plan de estudios de matemáticas integrado.

Irene Lee

Researcher, Massachusetts Institute of Technology Cambridge, Massachusetts.

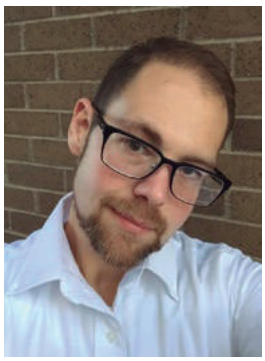
Irene Lee es investigadora en el Programa de Educación para Profesores Scheller y Arcade de Educación del Instituto Tecnológico de Massachusetts. Es la fundadora y directora del Proyecto GUTS: Growing Up Thinking Scientifically y anteriormente fue investigadora principal de New Mexico Computer Science for All, Young Women Growing Up Thinking Computationally y GUTS y Girls. Irene es la presidenta de la Fuerza de Tarea de Pensamiento Computacional de la Asociación de Profesores de las Ciencias de la Computación (CSTA) y sirvió como miembro del equipo de los Estándares Interinos de las Ciencias de Computación CSTA K–12, revisados en 2016. Anteriormente, diseñó y desarrolló videojuegos educativos y para Electronic Arts y Theatrix Interactive y trabajó en educación informal como especialista en ciencias. Irene es la ex presidenta del Supercomputing Challenge y el Swarm Development Group y la ex directora del Learning Lab del Instituto Santa Fe.

Carl Lyman

Specialist in the Information Technology Class Group, Utah State Board of Education, Salt Lake City, Utah.

Carl Lyman comenzó a enseñar programación a sus alumnos de tercer grado en 1982. Llevaba su computadora Apple II + de casa a la escuela todos los días. Enseñó resolución de problemas y programación a sus alumnos usando Turtle Graphics y Terrapin Logo. Sus alumnos aprendieron a resolver problemas, bucles, afirmaciones si-entonces y procedimientos. Hoy se llama "codificación". Carl pasó más de 30 años como profesor: seis años como profesor de primaria y más de 27 años impartiendo clases de las ciencias de la computación,

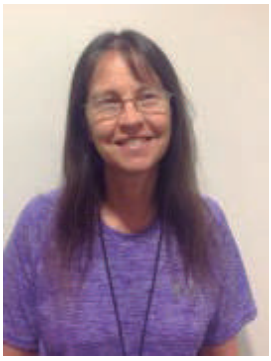
aplicaciones computacionales, programación, medios digitales y clases de soporte de tecnología de la información. Durante 10 años, trabajó en la Junta de Educación del Estado de Utah en educación vocacional y técnica, supervisando tecnología de la información (que incluye las ciencias de la computación), medios digitales, desarrollo web y cursos de programación de computadoras. Trabajó duro en Utah para capacitar a los profesores para enseñar ciencias de la computación y poner a disposición de los estudiantes más oportunidades en ciencias de la computación. Carl se ha jubilado recientemente.



Daniel Moix

Computer Science Education Specialist, Arkansas School for Mathematics, Sciences & Arts, Hot Springs, Arkansas.

Daniel Moix ha enseñado ciencias de la computación desde 2003 en la Arkansas School for Mathematics, Sciences & Arts; Colegio de los Ouachitas; y Bryant High School. Es el vicepresidente de la Asociación de Profesores de Ciencias de la Computación (CSTA) de Arkansas, miembro del Equipo de Liderazgo de Defensa de las Ciencias de la Computación de CSTA y el primer especialista en educación en ciencias de la computación de K-12 de Arkansas. Daniel fue el líder de nivel de grado 9–12 para los Estándares Interinos de Ciencias de Computación CSTA K–12, Revisado 2016 y recibió el Premio Presidencial 2015 a la Excelencia en Enseñanza de Matemáticas y Ciencias.



Dianne O'Grady-Cunniff

Computer Science Teacher, La Plata High School La Plata, Maryland.

Dianne O'Grady-Cunniff está trabajando en proyectos para llevar las ciencias de la computación a todos los estudiantes en K-12, después de enseñar las ciencias de la computación en la universidad y la escuela secundaria. Centrándose en el desarrollo del currículo y la capacitación y el apoyo de los profesores durante los últimos años, es una profesora/líder del equipo de CS Matters en Maryland y una facilitadora de Code.org. Trabajó con las Escuelas Públicas del Condado de Charles para capacitar a cientos de profesores para enseñar las ciencias de la computación y llevar la informática a todas las escuelas del distrito durante los últimos dos años. La educación en informática es la pasión de Dianne, y ella cree que cada niño debería tener la oportunidad de crear con tecnología de las ciencias de la computación.

Anthony A. Owen

Coordinator of Computer Science, Arkansas Department of Education
Little Rock, Arkansas.

Anthony A. Owen se desempeña como coordinador de las ciencias de la computación de Arkansas en el Departamento de Educación de Arkansas (ADE). Comenzó su carrera en educación como profesor de matemáticas y ciencias y luego se desempeñó como especialista en matemática y las ciencias de la computación de ADE K-12. Anthony actualmente sirve como líder estatal para el desarrollo y la implementación de la iniciativa de las ciencias de la computación del gobernador Asa Hutchinson. En este cargo, asesora y coordina con múltiples entidades nacionales y estatales, incluido el servicio como miembro de la Comisión de la Junta de Educación Regional del Sur sobre Ciencias de la Computación, Tecnología de la Información y Campos de Carrera Relacionados, así como con la Fuerza de Tarea de las Ciencias de la Computación del Gobernador Hutchinson, que identifica las necesidades de ciencia y tecnología computacional del estado. Anthony fue elegido recientemente como el representante del departamento de estado de la Computer Science Teachers Association. Anthony recibió una licenciatura en matemática con menores en educación y ciencias de la computación y una maestría en liderazgo educativo de la Universidad Estatal de Henderson. Recibió un doctorado en derecho de la Facultad de Derecho de Bowen en la Universidad de Arkansas en Little Rock en 2013 y fue admitido en el Colegio de Abogados de Arkansas en 2014.

Minsoo Park

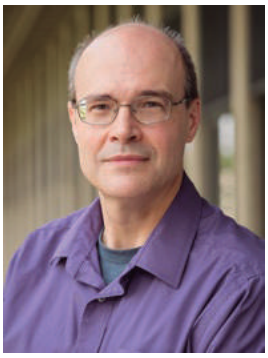
Director of Teaching and Learning, Countryside School Champaign, Illinois.

Minsoo Park es el director de enseñanza y aprendizaje en Countryside School. En 10 años de enseñanza, se ha desempeñado principalmente como profesor de ciencias de la computación/álgebra de la escuela intermedia, coordinador de tecnología y coordinador del Bachillerato Internacional del Programa de los Años Intermedios en las Escuelas Públicas de Chicago. Durante los últimos tres años, se desempeñó como especialista en enriquecimiento y tecnología en el Distrito Escolar Champaign de la Unidad 4, implementó proyectos dirigidos por los estudiantes y desarrolló unidades de integración de las ciencias de la computación y matemática en toda la escuela que enfatizan el proceso de metacognición y aprendizaje a través de conceptos computacionales y prácticas de pensamiento computacional. Ha sido reconocido con el Premio "Those Who Excel" por la Junta de Educación del Estado de Illinois por la innovación tecnológica. Está certificado en ciencias de la computación, matemáticas, ciencias sociales, ciencias físicas y educación tecnológica.

Shaileen Crawford Pokress

Visiting Scholar, Wyss Institute at Harvard; K–12 Curriculum Designer Cambridge, Massachusetts.

Shay Pokress es un desarrollador de currículum especializado en las ciencias de la computación K-12. Actualmente, Shay es investigadora visitante en el Instituto Wyss de Ingeniería Bioinspirada de Harvard, donde está desarrollando currículos para las ciencias de la computación basados en estándares en torno a las capacidades únicas de Root, un robot diseñado específicamente para aprender el pensamiento computacional. Antes de unirse al equipo de Root, Shay se desempeñó como directora de instrucción en Project Lead The Way, donde desarrolló cursos de las ciencias de la computación Advanced Placement® y fue la escritora principal de Launch Computer Science, un plan de estudios ampliamente adoptado que utiliza un enfoque de infusión para conectar el problema basada en la informática a los estándares de contenido K-5. En el Laboratorio de Medios del Instituto de Tecnología de Massachusetts, desarrolló y dirigió el programa educativo para App Inventor, una plataforma para crear aplicaciones móviles que tiene como objetivo democratizar la computación móvil. Como investigadora asociada senior en TERC, Shay se enfocó en el desarrollo profesional de profesores en matemáticas y ciencias. Shay obtuvo su licenciatura en ciencias de la computación de la Universidad de Cornell y su maestría de Harvard Graduate School of Education. Ella cree que el acceso a una educación de las ciencias de la computación de calidad es un problema de justicia social.



George Reese

Director of MSTE, MSTE Office at University of Illinois at Urbana Champaign, Champaign, Illinois.

George Reese es el director de la Oficina de Educación en Matemáticas, Ciencia y Tecnología (MSTE) de la Facultad de Educación de la Universidad de Illinois en Urbana Champaign. La oficina de MSTE trabaja para mejorar la enseñanza y el aprendizaje con apoyo tecnológico en matemáticas y ciencias a través del diseño curricular y las asociaciones de desarrollo profesional de profesores con escuelas y distritos. Antes de trabajar en MSTE, George era profesor de matemáticas en la escuela secundaria en la escuela de Americanos Nativos de Santa Fe en Santa Fe, NM. Actualmente es el presidente de la junta del Consejo de Profesores de Matemáticas de Illinois.

Hal Speed

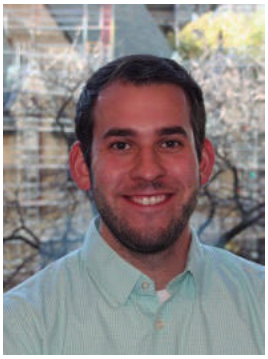
Founder, CS4TX Austin, Texas.

Hal Speed es un defensor de la educación en ciencias de la computación para todos los estudiantes en los grados K-12 y cree que estas habilidades son necesarias para la movilidad socioeconómica y la prosperidad futura de las naciones en la era digital. Fundó CS4TX (Computer Science for Texas) para coordinar actividades en todo Texas y representar al estado en la iniciativa nacional CSforAll y la Alianza de Rutas de Educación en Computación en Expansión. Hal es ingeniero de experiencia en productos en Dell y se desempeña como facilitador de Code.org, presidente del comité de la Asociación de Profesores de las Ciencias de la Computación y vicepresidente del Grupo de Interés Especial de la Asociación de Educación de Computación de Texas. Tiene una licenciatura en ingeniería eléctrica y una maestría en administración de empresas de Virginia Tech.

Alfred Thompson

Computer Science Teacher, Bishop Guertin High School Nashua, New Hampshire.

Alfred Thompson es profesor de las ciencias de la computación en la escuela secundaria Bishop Guertin y es miembro de la junta de la Asociación de Docentes de las ciencias de la computación. Ha sido desarrollador profesional de software, autor de libros de texto, evangelista desarrollador, coordinador de tecnología escolar, miembro de la junta escolar y más. Alfred se ve a sí mismo como un activista de educación en ciencias de la computación que trabaja para ayudar a llegar a más jóvenes con el conocimiento de que pueden hacer del mundo un lugar mejor a través del software. Es autor del popular blog "Computer Teacher."



Bryan Twarek

Computer Science Program Administrator, San Francisco Unified School District San Francisco, California.

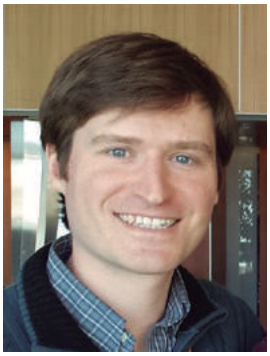
Bryan Twarek (BT) es el administrador del programa de ciencias de la computación del Distrito Escolar Unificado de San Francisco (SFUSD), donde está trabajando para expandir la enseñanza de ciencias de la computación a todos los estudiantes y todas las escuelas dentro de las escuelas públicas de San Francisco. Su objetivo es asegurar que todos los estudiantes de SFUSD tengan acceso equitativo a una enseñanza rigurosa y atractiva de las ciencias de la computación, desde pre-kindergarten hasta 12 ° grado. Con este fin, supervisa las políticas, el desarrollo de los currículos y el desarrollo profesional. También es escritor de los Estándares Interinos de Ciencias de la Computación CSTA K–12, Revisado 2016 y miembro de la junta de los Computer Using Educators de San Francisco (afiliado de la Sociedad Internacional de Tecnología en Educación). Anteriormente, trabajó como decano, profesor, entrenador de instrucción y especialista integrador de tecnología.

BT se graduó de la Universidad de Yale con un título en psicología y neurociencia humana. Obtuvo su maestría en política y administración de educación urbana de la Universidad Loyola Marymount.

A. Nicki Washington

Associate Professor, Computer Science, Winthrop University Rock Hill, South Carolina.

Nicki Washington es profesora asociada de las ciencias de la computación en la Universidad Winthrop. Antes de esto, ella era profesora asociada en la Universidad de Howard. Sus intereses de investigación se centran en la educación en ciencias de la computación, aumentando específicamente la participación de minorías subrepresentadas. Sus proyectos de investigación han incluido asociaciones con las Escuelas Públicas del Distrito de Columbia, Explorando CS y Google. Su investigación más reciente incluye el desarrollo de la Encuesta de Actitud e Identidad de Ciencias de la Computación, una herramienta para medir el impacto de la identidad étnica en las actitudes de los estudiantes hacia la búsqueda y la persistencia en las ciencias de la computación. Se graduó en 2000 de la Universidad Johnson C. Smith.



David Weintrop

Postdoctoral Researcher, UChicago STEM Ed Chicago, Illinois.

David Weintrop es investigador postdoctoral en UChicago STEM Ed en la Universidad de Chicago. Tiene un doctorado en las ciencias del aprendizaje de la Northwestern University y una licenciatura en ciencias de la computación de la Universidad de Michigan. Antes de comenzar su carrera académica, pasó cinco años trabajando como desarrollador en un par de startups de software en Chicago. La investigación de David se enfoca en el diseño, implementación y evaluación de entornos de programación introductorios accesibles y atractivos. También está interesado en el uso de herramientas tecnológicas para apoyar la exploración y la expresión a través de diversos contextos, incluyendo aulas de ciencia, tecnología, ingeniería y matemáticas, y espacios informales. Su trabajo se encuentra en la intersección de la interacción humano-computadora, el diseño y las ciencias del aprendizaje. David ganó la medalla de oro en la Competencia de Investigación Estudiantil en la conferencia ACM Computer Science Education 2015 por su trabajo de disertación y ha presentado su investigación en Google, el Instituto de Tecnología de Massachusetts y conferencias en todo el mundo.

Personal de Desarrollo

Pat Yongpradit

Chief Academic Officer, Code.org.

Pat Yongpradit es el director académico de Code.org, una organización sin fines de lucro dedicada a promover la educación en ciencias de la computación. Como una voz nacional en la educación en ciencias de la computación K-12, su pasión es brindar oportunidades de ciencias de la computación a cada escuela y estudiante en los Estados Unidos. A lo largo de su carrera como profesor de las ciencias de la computación en la escuela secundaria, inspiró a los estudiantes a crear juegos y aplicaciones móviles para causas sociales e implementó iniciativas para ampliar la participación en ciencias de la computación entre los grupos subrepresentados. Ha aparecido en el libro American Teacher: Heroes in the Classroom y en 2010 fue reconocido como Microsoft Worldwide Innovative Educator. Tiene una licenciatura en neurobiología, una maestría en educación secundaria y está certificado en biología, física, matemáticas, salud y tecnología. Mientras Pat actualmente pasa más tiempo enfocado en ciencias de la computación desde una perspectiva nacional, su corazón todavía está en el aula.

Katie Hendrickson

Advocacy and Policy Manager, Code.org.

Katie Hendrickson es gerente de defensa y políticas en Code.org. Trabaja en temas de política y defensa estatal, incluida la implementación estatal de iniciativas de educación en las ciencias de la computación. Antes de unirse a Code.org, fue becaria distinguida de educadora Albert Einstein 2014–15, ubicada en la Fundación Nacional de Ciencias en la Dirección de Ingeniería y las Ciencias de la Computación y Ciencias de la Información. Enseñó matemáticas secundarias durante seis años en Alexander Middle School y Athens Middle School, donde recibieron el Premio Estatal Secundario Buck Martin a la excelencia en la enseñanza de las matemáticas del Consejo de Profesores de Matemáticas de Ohio. Cofundó el Círculo de Profesores de Matemáticas del Sureste de Ohio, y su investigación de tesis investigó la identidad de los profesores y el desarrollo profesional. Tiene un doctorado en currículo e instrucción y una maestría en estudios culturales en educación de la Universidad de Ohio.

Rachel Phillips

Director of Research and Evaluation, Code.org.

Rachel Phillips es la directora de investigación y evaluación en Code.org. Antes de unirse a Code.org, realizaron evaluaciones y tuvieron en cuenta un currículo para el Big History Project. Además, fue la directora del programa de un proyecto de investigación financiado por la Fundación Nacional de Ciencias que estudiaba los impactos de las modificaciones y los logros en los jóvenes de bajos ingresos y cómo esas actividades pueden aumentar la participación en los campos de ciencia, tecnología, ingeniería y matemáticas (STEM). Sus intereses de investigación incluyen el aprendizaje de estudiantes y profesores en espacios de educación formal, con un enfoque particular en jóvenes tradicionalmente marginados y desatendidos. La mayor parte de su investigación ha sido en el contexto de los campos STEM, y sus publicaciones más recientes están relacionadas con el aprendizaje en entornos de juego en línea y la metodología controlada para estudiar el aprendizaje en este tipo de entornos. Obtuvo su doctorado en ciencias del aprendizaje de la Universidad de Washington en 2011 y una maestría en artes en la enseñanza de la Universidad Americana en 2006.

Debbie Carter

Editor, Educational Consultant.

Miranda Parker

Intern, Georgia Tech.

Lian Halbert

Operations, Code.org

Consultores/Asesores de Procesos

Courtney K. Blackwell

Outlier Research & Evaluation, UChicago STEM Education, University of Chicago.

Jeanne Century

Outlier Research & Evaluation, UChicago STEM Education, University of Chicago.

Jennifer Childress

Achieve.

Thomas Keller

Maine Mathematics and Science Alliance.

Heather King

Outlier Research & Evaluation, UChicago STEM Education, University of Chicago.

Apéndice C: Glosario

El glosario incluye definiciones de términos utilizados en las declaraciones del marco. Estos términos son definidos para los lectores del marco y no necesariamente están destinados a ser las definiciones o términos que serían vistos por los estudiantes.

Tabla C.1: Términos del glosario

TÉRMINO	DEFINICIÓN
abstracción	(proceso): El proceso de reducir la complejidad al centrarse en la idea principal. Al ocultar detalles irrelevantes para la pregunta en cuestión y reunir detalles relacionados y útiles, la abstracción reduce la complejidad y permite centrarse en el problema. (producto): una nueva representación de una cosa, un sistema o un problema que replantea un problema de forma útil al ocultar detalles irrelevantes para la pregunta en cuestión [MDESE, 2016].
accesibilidad	El diseño de productos, dispositivos, servicios o entornos para personas con discapacidades. Los estándares de accesibilidad que generalmente son aceptados por grupos profesionales incluyen las Pautas de Accesibilidad al Contenido en la Web (WCAG) 2.0 y los estándares de Aplicaciones de Internet Ricas Accesibles (ARIA). [Wikipedia].
algoritmo	Un proceso paso a paso para completar una tarea.
almacenamiento	(lugar) Un lugar, generalmente un dispositivo, en el que se pueden ingresar datos, en el que se pueden almacenar los datos, y desde el cual se pueden recuperar los datos en un momento posterior [FOLDOC].
análogo	La característica definitoria de los datos que se representa de forma continua y física. Mientras que los datos digitales son un conjunto de símbolos individuales, los datos analógicos se almacenan en medios físicos, como las ranuras de la superficie en un disco de vinilo, la cinta magnética de un videocasete u otros medios no digitales [Techopedia].
app	Un tipo de software de aplicación diseñado para ejecutarse en un dispositivo móvil, como un teléfono inteligente o una tableta. También conocida como aplicación móvil [Techopedia].
artefacto computacional	Cualquier cosa creada por un humano usando un proceso de pensamiento computacional y un dispositivo computacional. Un artefacto computacional puede ser, entre otros, un programa, imagen, audio, video, presentación o archivo de página web [College Board, 2016].
artefacto	Cualquier cosa creada por un humano. Ver artefacto computacional para la definición utilizada en ciencias de la computación.
audiencia	Usuarios finales esperados de un artefacto o sistema computacional.

autenticación	La verificación de la identidad de una persona o proceso. [FOLDOC].
automatizar; automatización	automatizar: para vincular sistemas y software dispares para que se vuelvan autoactivos o autorreguladores. [Ross, 2016] automatización: el proceso de automatizar.
Boolean	Un tipo de datos o expresión con dos valores posibles: verdadero y falso [FOLDOC].
bucle	Una estructura de programación que repite una secuencia de instrucciones siempre que una condición específica sea verdadera [Tech Terms].
caso de prueba	Un conjunto de condiciones o variables bajo las cuales un probador determinará si el sistema que se está probando cumple los requisitos o funciona correctamente [STF].
ciencia de la computación	El estudio de las computadoras y los procesos algorítmicos, incluidos sus principios, sus diseños de hardware y software, su implementación y su impacto en la sociedad [ACM, 2006].
cifrado	La conversión de datos electrónicos a otra forma, llamada texto cifrado, que nadie puede entender fácilmente, excepto las partes autorizadas [TechTarget].
ciudadanía digital	Los estándares de comportamiento apropiado y responsable con respecto al uso de la tecnología [MDESE, 2016].
código	Cualquier conjunto de instrucciones expresadas en un lenguaje de programación [MDESE, 2016].
comentario	Se agregó una anotación legible por el programador en el código de un programa de computadora para que el código sea más fácil de entender. Los comentarios son generalmente ignorados por las máquinas [Wikipedia].
complejidad	La cantidad mínima de recursos, como memoria, tiempo o mensajes, necesarios para resolver un problema o ejecutar un algoritmo [NIST/DADS].
componente	Un elemento de un grupo más grande. Por lo general, un componente proporciona un servicio particular o un grupo de servicios relacionados [Tech Terms, TechTarget].
computación	Cualquier actividad orientada a objetivos que requiera, se beneficie o cree procesos algorítmicos [MDESE, 2016].
computacional	Relacionado con computadoras o métodos de computación.
computadora	Una máquina o dispositivo que realiza procesos, cálculos y operaciones según las instrucciones proporcionadas por un programa de software o hardware [Techopedia].

condicional	Una característica de un lenguaje de programación que realiza diferentes cálculos o acciones dependiendo de si una condición booleana especificada por el programador se evalúa como verdadera o falsa [MDESE, 2016]. Un condicional podría referirse a una declaración condicional, expresión o construcción condicional.
conectividad	La capacidad de un programa o dispositivo para vincularse con otros programas y dispositivos [Webopedia].
conexión	Una vinculación física o inalámbrica entre múltiples sistemas computacionales, computadoras o dispositivos computacionales.
confiabilidad	Un atributo de cualquier sistema que constantemente produce los mismos resultados, preferiblemente cumpliendo o excediendo sus requisitos [FOLDOC].
configuración	Proceso: definir las opciones que se pueden configurar al instalar o modificar hardware y software o el proceso de creación de la configuración [TechTarget]. Producto: los detalles específicos de hardware y software que indican exactamente como está compuesto el sistema, especialmente en términos de dispositivos conectados, capacidad o capacidad [TechTarget].
conmutador (switch)	Un dispositivo de alta velocidad que recibe paquetes de datos entrantes y los redirige a su destino en una red de área local (LAN) [Techopedia].
control; estructura de control	control: en general, el poder de dirigir el curso de las acciones. En programación, el uso de elementos de código de programación para dirigir qué acciones tienen lugar y el orden en que tienen lugar. estructura de control: una estructura de programación (código) que implementa el control. Los condicionales y los bucles son ejemplos de estructuras de control.
controlador de eventos	Un procedimiento que especifica qué debe suceder cuando ocurre un evento específico.
cultura; prácticas culturales	cultura: una institución humana manifestada en el comportamiento aprendido de las personas, incluidos sus sistemas de creencias específicos, lenguaje (s), relaciones sociales, tecnologías, instituciones, organizaciones y sistemas para usar y desarrollar recursos [NCSS, 2013]. prácticas culturales: las exhibiciones y comportamientos de una cultura.
datos	Información recopilada y utilizada para referencia o análisis. Los datos pueden ser digitales o no digitales y pueden tener muchas formas, incluidos números, texto, muestra de manos, imágenes, sonidos o videos [CAS, 2013; Tech Terms].
depuración	El proceso de encontrar y corregir errores (debugging) en los programas [MDESE, 2016].
descomponer; descomposición	descomponer: Para descomponerse en componentes. descomposición: descomponer un problema o sistema en componentes [MDESE, 2016].

digital	Una característica de la tecnología electrónica que utiliza valores discretos, generalmente 0 y 1, para generar, almacenar y procesar datos [Techopedia].
dispositivo computacional	Un dispositivo físico que utiliza hardware y software para recibir, procesar y generar información. Las computadoras, los teléfonos móviles y los chips de computadora dentro de los electrodomésticos son ejemplos de dispositivos computacionales.
dispositivo	Una unidad de hardware físico que proporciona una o más funciones computacionales dentro de un sistema de computación. Puede proporcionar entrada a la computadora, aceptar salida o ambas [Techopedia].
eficiencia	Una medida de la cantidad de recursos que utiliza un algoritmo para encontrar una respuesta. Generalmente se expresa en términos de los cálculos teóricos, la memoria utilizada, el número de mensajes pasados, el número de accesos al disco, etc., [NIST/DADS].
ejecutar; ejecución	ejecutar: para llevar a cabo una instrucción o conjunto de instrucciones programas, aplicaciones, etc. ejecución: el proceso de ejecución de una instrucción o conjunto de instrucciones [FOLDOC].
encapsulamiento	La técnica de combinar datos y los procedimientos que actúan sobre ellos para crear un tipo [FOLDOC].
enrutador	Un dispositivo o software que determina la ruta por la que viajan los paquetes de datos desde el origen hasta el destino [TechTarget].
entrada	Las señales o instrucciones enviadas a una computadora [Techopedia].
error (bug)	Un error en un programa de software. Puede hacer que un programa se cierre inesperadamente o se comporte de manera no intencionada [Tech Terms]. El proceso de búsqueda y corrección de errores (errores) se denomina depuración. [Wikipedia].
escalabilidad	La capacidad de una red para manejar una cantidad creciente de trabajo o su potencial para ampliarse para acomodar ese crecimiento [Wikipedia].
estructura de los datos	Una forma particular de almacenar y organizar datos dentro de un programa de computadora para adaptarse a un propósito específico para que se pueda acceder y trabajar de manera adecuada [TechTarget].
estructura	Un término general utilizado en el marco para discutir el concepto de encapsulación sin especificar una metodología de programación particular.
evento	Cualquier suceso identificable que tenga importancia para el hardware o software del sistema. Los eventos generados por el usuario incluyen pulsaciones de teclas y clics del mouse; Los eventos generados por el sistema incluyen la carga de programas y errores [TechTarget].
hardware	Los componentes físicos que componen un sistema informático, computadora o dispositivo computacional [MDESE, 2016].
hilo	Una secuencia de letras, números y/u otros símbolos. Una cadena puede representar, por ejemplo, un nombre, dirección o título de la canción. Algunas funciones comúnmente asociadas con cadenas son longitud, concatenación y subcadena [TechTarget].
identificador	El nombre único y definido por el usuario de un elemento del programa en el código, tales como una variable o un procedimiento. Un nombre de identificador debe indicar el significado y el uso del elemento que se indique [Techopedia].

implementación	El proceso de expresar el diseño de una solución en un lenguaje de programación, código, que puede ejecutarse en un dispositivo computacional.
inferencia	Se llegó a una conclusión sobre la base de pruebas y razonamientos [Oxford].
integridad	La integridad general, precisión y consistencia de los datos [Techopedia].
interacción humano-computadora (IHC)	El estudio de cómo las personas interactúan con las computadoras y en qué medida los sistemas computacionales se desarrollan o no para una interacción exitosa con los seres humanos [TechTarget].
Internet	La colección global de redes de computadoras y sus conexiones, todas utilizando protocolos compartidos para comunicarse [CAS, 2013].
iterativo	Implica la repetición de un proceso con el objetivo de acercarse a una meta, objetivo o resultado deseado [MDESE, 2016].
jerarquía	Una estructura organizativa en la que los elementos se clasifican según los niveles de importancia [TechTarget].
memoria	Almacenamiento temporal utilizado por dispositivos computacionales [MDESE, 2016].
modelo	Una representación de alguna parte de un problema o un sistema [MDESE, 2016]. Nota: Esta definición difiere de la utilizada en la ciencia.
modularidad	La característica de un software/aplicación web que se ha dividido (descompuesto) en módulos más pequeños. Una aplicación puede tener varios procedimientos que se llaman desde su procedimiento principal. Los procedimientos existentes podrían reutilizarse recombinándolos en una nueva aplicación [Techopedia].
modulo	Un componente de software o parte de un programa que contiene uno o más procedimientos. Uno o más módulos desarrollados independientemente forman un programa [Techopedia].
operación	Una acción, resultante de una sola instrucción, que cambia el estado de los datos [Free Dictionary].
paquete	La unidad de datos enviada a través de una red [Tech Terms].
parámetro	Un tipo de variable especial utilizada en un procedimiento para referirse a uno de los datos recibidos como entrada por el procedimiento. [MDESE, 2016].
pensamiento computacional	La capacidad humana de formular problemas para que sus soluciones puedan representarse como pasos computacionales o algoritmos para ser ejecutados por una computadora [Lee, 2016].
piratería	La copia, distribución o uso de software ilegal [TechTarget].

procedimiento	Un módulo de código independiente que cumple una tarea concreta y está referenciado dentro de un cuerpo más grande de código de programa. El papel fundamental de un procedimiento es ofrecer un único punto de referencia para alguna meta o tarea pequeña que el desarrollador o programador pueda activar invocando el procedimiento en sí [Techopedia]. En este marco, procedimiento se usa como un término general que puede referirse a un procedimiento real o un método, función o módulo de cualquier otro nombre por el cual los módulos son conocidos en otros lenguajes de programación.
proceso	Una serie de acciones o pasos tomados para lograr un resultado particular [Oxford].
programa; programación	programa (s): conjunto de instrucciones que la computadora ejecuta para lograr un objetivo particular [MDESE, 2016]. programar (v): Producir un programa por programación. programación: El arte de analizar problemas y diseñar, escribir, probar y mantener programas para resolverlos [MDESE, 2016].
protocolo	El conjunto de reglas especiales utilizadas por los puntos finales en una conexión de telecomunicaciones cuando se comunican. Los protocolos especifican interacciones entre las entidades comunicantes [TechTarget].
prototipo	Una aproximación temprana de un producto final o sistema de información, a menudo construido con fines de demostración [TechTarget, Techopedia].
red	Un grupo de dispositivos computacionales (computadoras personales, teléfonos, servidores, conmutadores, enrutadores, etc.) conectados por cables o medios inalámbricos para el intercambio de información y recursos.
redundancia	Un diseño de sistema en el que un componente está duplicado, por lo que si falla, habrá una copia resguardada [TechTarget].
remezcla	El proceso de crear algo nuevo a partir de algo viejo. Originalmente un proceso que involucraba música, la remezcla implica crear una nueva versión de un programa mediante la recombinación y modificación de partes de programas existentes, y a menudo agregando nuevas piezas, para formar nuevas soluciones [Kafai & Burke, 2014].
resolución de problemas	Un enfoque sistemático para la resolución de problemas que a menudo se usa para encontrar y resolver un problema, error o falla dentro del software o un sistema computacional [Techopedia, TechTarget].
seguridad cibernética	La protección contra el acceso o la alteración de los recursos computacionales mediante el uso de tecnología, procesos y capacitación [TechTarget].
seguridad	Ver la definición de ciberseguridad.
simular; simulación	similar: imitar la operación de un proceso o sistema del mundo real. simulación: imitación de la operación de un proceso o sistema del mundo real [MDESE, 2016].

sistema de computación	Una colección de una o más computadoras o dispositivos computacionales, junto con su hardware y software, integrados con el fin de realizar tareas compartidas. Aunque un sistema de computación puede limitarse a una sola computadora o dispositivo computacional, más comúnmente se refiere a una colección de múltiples computadoras, dispositivos computacionales y hardware conectados.
sistema	Una colección de elementos o componentes que trabajan juntos para un propósito común [TechTarget]. Ver también la definición de sistema de computación.
software	Programas que se ejecutan en un sistema de computación, computadora u otro dispositivo computacional.
tipo de datos	Una clasificación de datos que se distingue por sus atributos y los tipos de operaciones que se pueden realizar en ellos. Algunos tipos de datos comunes son entero, cadena, booleano (verdadero o falso) y coma flotante.
topología	La configuración física y lógica de una red; la disposición de una red, incluidos sus nodos y enlaces de conexión. Una topología lógica es la forma en que los dispositivos aparecen conectados al usuario. Una topología física es la forma en que realmente están interconectados con alambres y cables [PCMag].
usuario final (o usuario)	Una persona para quien está diseñado un producto de hardware o software, a diferencia de los desarrolladores [TechTarget].
variable	Un nombre simbólico que se utiliza para realizar un seguimiento de un valor que puede cambiar mientras se ejecuta un programa. Las variables no solo se usan para números; También pueden contener texto, incluyendo oraciones completas (cadenas) o valores lógicos (verdadero o falso). Una variable tiene un tipo de datos y está asociada con una ubicación de almacenamiento de datos; su valor normalmente cambia durante el curso de la ejecución del programa. [CAS, 2013; Techopedia] Nota: esta definición difiere de la utilizada en matemáticas.

Referencias

Algunas definiciones vinieron directamente de las fuentes enumeradas en la Tabla C.2, mientras que otras fueron extraídas o adaptadas para incluir contenido relevante para este marco.

Tabla C.2: Referencias Glosarías

ACM, 2006	**A Model Curriculum for K–12 Computer Science** Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.
CAS, 2013	**Computing At School's Computing in the National Curriculum: A Guide for Primary Teachers** Computing At School. (2013). *Computing in the national curriculum: A guide for primary teachers.* Belford, UK: Newnorth Print. Retrieved from http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf
College Board, 2016	**College Board Advanced Placement® Computer Science Principles** College Board. (2016). *AP Computer Science Principles course and exam description*. New York, NY: College Board. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf
FOLDOC	**Free On-Line Dictionary of Computing** Free on-line dictionary of computing. (n.d.). Retrieved from http://foldoc.org
Free Dictionary	**The Free Dictionary** The free dictionary. (n.d.). Retrieved from http://www.thefreedictionary.com
Kafai & Burke, 2014	**Connected Code: Why Children Need to Learn Programming** Kafai, Y., & Burke, Q. (2014). *Connected code: Why children need to learn programming.* Cambridge, MA: MIT Press.
Lee, 2016	**Reclaiming the Roots of CT** Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12*(1), 3–4. Retrieved from http://www.csteachers.org/resource/resmgr/Voice/csta_voice_03_2016.pdf
MDESE, 2016	**Massachusetts Digital Literacy and Computer Science (DL&CS) Standards** Massachusetts Department of Elementary and Secondary Education. (2016, June). 2016 *Massachusetts digital literacy and computer science (DLCS) curriculum framework*. Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

La tabla continúa en la página siguiente

La tabla continúa de la página anterior

NCSS, 2013	**College, Career & Civic Life (C3) Framework for Social Studies State Standards** National Council for the Social Studies. (2013). *The college, career, and civic life (C3) framework for social studies state standards: Guidance for enhancing the rigor of K–12 civics, economics, geography, and history.* Silver Spring, MD: Author. Retrieved from http://www.socialstudies.org/system/files/c3/C3-Framework-for-Social-Studies.pdf
NIST/DADS	**National Institute of Science and Technology Dictionary of Algorithms and Data Structures** Pieterse, V., & Black, P. E. (Eds.). (n.d). *Dictionary of algorithms and data structures.* Retrieved from https://xlinux.nist.gov/dads//
Oxford	**Oxford Dictionaries** Oxford dictionaries. (n.d.). Retrieved from http://www.oxforddictionaries.com/us
PCmag	**PCmag.com Encyclopedia** PCmag.com encyclopedia. (n.d.). Retrieved from http://www.pcmag.com/encyclopedia/term/46301/logical-vs-physical-topology
Ross, 2016	**What Is Automation** Ross, B. (2016, May 10). What is automation and how can it improve customer service? *Information Age*. Retrieved from http://www.information-age.com/industry/software/123461408/what-automation-and-how-can-it-improve-customer-service
STF	**Software Testing Fundamentals** Software testing fundamentals. (n.d). Retrieved from http://softwaretestingfundamentals.com
Tech Terms	**Tech Terms** Tech terms computer dictionary. (n.d.). Retrieved from http://www.techterms.com
Techopedia	**Techopedia** Techopedia technology dictionary. (n.d.). Retrieved from https://www.techopedia.com/dictionary
TechTarget	**TechTarget Network** TechTarget network. (n.d.). Retrieved from http://www.techtarget.com/network
Webopedia	**Webopedia** Webopedia. (n.d.). Retrieved from http://www.webopedia.com
Wikipedia	**Wikipedia** Wikipedia: The free encyclopedia. (n.d.). Retrieved from https://www.wikipedia.org/

Apéndice D: Revisión de Investigación de la Primera Infancia

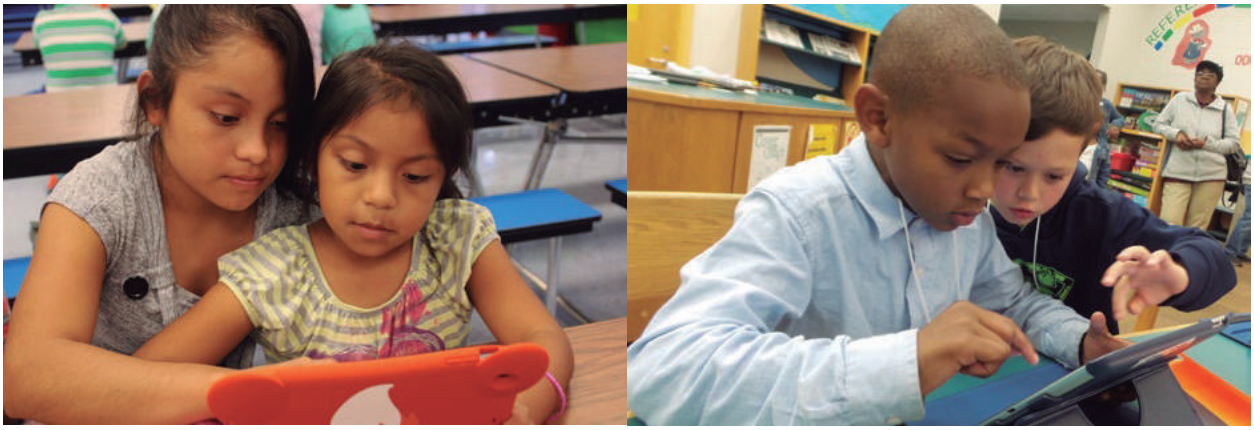
Definido como "el estudio de las computadoras y los procesos algorítmicos, incluidos sus principios, sus diseños de hardware y software, sus aplicaciones y su impacto en la sociedad" (Tucker et al., 2006, p. 2), las ciencias de la computación se basan en el uso de las tecnologías computacionales, un tema históricamente debatido en el sector de la educación durante la primera infancia. Es importante destacar que comprometerse con la tecnología no es lo mismo que hacer las ciencias de la computación, pero el acceso y uso de la tecnología a menudo es un precursor y una ventaja para participar en ciencias de la computación. Por lo tanto, aunque se debe tener precaución al interpretar la investigación sobre los efectos de la tecnología en el aprendizaje y el desarrollo de los niños, esta investigación proporciona un trasfondo para comprender un importante factor contextual en la educación en ciencias de la computación. A pesar de la naturaleza ubicua de la tecnología en el mundo actual, e incluso la participación de los niños muy pequeños en experiencias mediadas digitalmente (por ejemplo, Rideout, 2013; Blackwell, Wartella, Lauricella y Robb, 2015), la integración de la tecnología en la educación durante la primera infancia a menudo se mantiene oposición a las nociones tradicionales de entornos de aprendizaje de pre-kinder. Aunque existe una gran cantidad de investigaciones sobre los impactos positivos de los medios educativos de alta calidad para el aprendizaje y el desarrollo de los niños pequeños (por ejemplo, Fisch y Truglio, 2001; Huston, Anderson, Wright, Linebarger y Schmitt, 2001; Pasnik y Llorente, 2013 ; Penuel et al., 2012), persisten las preocupaciones sobre las posibles consecuencias negativas derivadas de demasiado tiempo frente a la pantalla o la exposición a contenido violento (ver Anderson y Bushman, 2001, para su revisión). Además, a menudo se considera que la tecnología interrumpe y desplaza las interacciones sociales, el juego imaginativo y el aprendizaje activo de los niños (Donohue, 2015). De hecho, una razón principal por la cual los educadores de la primera infancia no usan la tecnología con más frecuencia, incluso si tienen acceso a ella, sigue siendo que estas actitudes fundamentales hacia la tecnología son la antítesis de lo que deberían ser las experiencias de educación temprana (e.g., Cordes & Miller, 2000; Lindahl & Folkesson, 2012).

Además, en 2015, la Academia Estadounidense de Pediatría (AAP, 2015) realizó una revisión histórica a su postura sin tiempo de pantalla al reconocer que los medios educativos de calidad, especialmente cuando se usan con la participación activa de cuidadores, pueden beneficiar el aprendizaje y el desarrollo de los niños pequeños.

Si bien persisten las preocupaciones, varias organizaciones profesionales importantes han revisado su postura sobre el papel de los medios digitales y la tecnología, ya que reconocen que las tecnologías digitales son herramientas esenciales para el aprendizaje y la creación (Dooley, Flint, Holbrook, May y Albers, 2011). En 2012, la Asociación Nacional para la Educación de Niños Pequeños (NAEYC) y el Centro Fred Rogers (FRC) emitieron una declaración de posición conjunta que apoya la tecnología de uso apropiado e intencional para el desarrollo a principios de

educación infantil (NAEYC y FRC, 2012). Además, en 2015, la Academia Estadounidense de Pediatría (AAP, 2015) realizó una revisión histórica de su postura sin tiempo de pantalla al reconocer que los medios educativos de calidad, especialmente cuando se utilizan con la participación activa de cuidadores, pueden beneficiar el aprendizaje y el desarrollo de los niños pequeños.

Como sugieren las declaraciones de NAEYC / FRC (2012) y AAP (2015), la tecnología puede tener un lugar en el aprendizaje de los niños pequeños pero no sustituye las actividades educativas tradicionales (por ejemplo, interacciones sociales entre pares y adultos-niños, juegos imaginativos, aprendizaje práctico) que son tan fundamentales para la educación de la primera infancia. Cuando se trata de las ciencias de la computación, se toma un marco similar, donde las tecnologías computacionales complementan las actividades de aprendizaje práctico. Es decir, si bien existen currículos de la computación digital de alta calidad y apropiados para el desarrollo, pueden y deben utilizarse para soportar entornos computacionales físicos (es decir, sin tecnología digital) en la educación durante la primera infancia. Como señaló Haugland (1992), la combinación de actividades basadas en computadora con actividades desconectadas puede mejorar la resolución de problemas, la abstracción y las habilidades verbales de los niños pequeños.



Basándose en nociones del construccionismo (Papert, 1980) y alineado con prácticas apropiadas para el desarrollo (Copple y Bredekamp, 2009), Bers, Ponte, Juelick, Viera y Schenker (2002) articularon cuatro principios de integración de la tecnología en el contexto de las ciencias de la computación en educación de la primera infancia:

1. Los entornos tecnológicos pueden ayudar al aprendizaje de los estudiantes al participar en actividades prácticas, de investigación activa y basadas en juegos;
2. Los objetos físicos ofrecen un apoyo crítico para desarrollar habilidades de pensamiento concretas y comprender fenómenos abstractos;
3. Las "ideas poderosas" transcurriculares son necesarias para conectar todas las áreas de aprendizaje; y
4. La autorreflexión es crítica para involucrar a los estudiantes en procesos de pensamiento metacognitivo.

Si bien gran parte de la retórica de las ciencias de la computación se ha centrado en preparar a los estudiantes para la fuerza laboral del siglo XXI, estos cuatro principios, y el marco constructivista en general, aclaran las oportunidades para que la educación en ciencias de la computación se expanda más allá del desarrollo de habilidades técnicas y el contenido

la adquisición de conocimientos. Vista como tal, las ciencias de la computación proporciona una plataforma para que los estudiantes desarrollen y participen en procesos de pensamiento de orden superior, resolución de problemas y pensamiento metacognitivo que son transferibles fuera del entorno de programación computacional (diSessa, 2000; Papert, 1980; Clements y Natasi, 1999) Como Papert (1980) articuló, el valor de las ciencias de la computación proviene de "ver las ideas de las ciencias de la computación no solo como instrumentos de explicación de cómo funcionan realmente el aprendizaje y el pensamiento, sino también como instrumentos de cambio que podrían alterar y posiblemente mejorar, la forma en que las personas aprenden y piensan "(págs. 208–209).

Aunque no hay conceptos o prácticas de ciencias de la computación especificados en los principios de Bers y colegas (2002), brindan orientación para los enfoques de enseñanza de las ciencias de la computación en entornos de aprendizaje temprano. De hecho, el desarrollo de interfaces de usuario tangibles es un ejemplo, en el que los manipuladores concretos tradicionales (p. Ej., Bloques de construcción) y las pantallas táctiles digitales ofrecen una experiencia de aprendizaje combinada para aprender conocimientos y habilidades computacionales fundamentales (p. Ej., Horn, AlSulaiman y Koh, 2013; Horn, Crouser y Bers, 2012). Por ejemplo, Horn y colegas (2013) diseñaron una experiencia combinada en la que los niños participan en actividades de programación de computadoras colocando pegatinas en el libro de cuentos en papel, que luego controlan las acciones de los personajes digitales en un teléfono inteligente o tableta. Por lo tanto, incrustado en la experiencia de lectura tradicional hubo oportunidades para que los niños pequeños se involucraran en conceptos clave de las ciencias de la computación, incluyendo secuenciación y bucles.

La investigación muestra que la participación en un entorno estructurado de programación de computadoras ayuda al sentido numérico, la memoria visual y las habilidades del lenguaje de los niños pequeños.

Otros han desarrollado interfaces de programación tangibles para llevar los lenguajes tradicionales de programación de computadoras en línea al mundo real (por ejemplo, Bers y Horn, 2010; Horn y Jacob, 2007; Wyeth, 2008). Originalmente conceptualizado por Perlman (1976) en la década de 1970, las interfaces tangibles ofrecen una manera de eliminar las barreras de texto y habilidades motoras que limitan la capacidad de los niños pequeños para participar en la programación de computadoras. Por ejemplo, Bers y Horn (2010) desarrollaron un lenguaje de programación tangible mediante el uso de bloques entrelazados que permiten a los niños en edad preescolar construir físicamente un programa de computadora en lugar de escribir uno con un teclado y un mouse en la computadora. Por lo tanto, al proporcionar múltiples puntos de entrada, conectados y desconectados, para desarrollar habilidades y conocimientos computacionales, los educadores pueden ofrecer oportunidades de desarrollo apropiadas y atractivas para los niños pequeños que pueden despertar un interés temprano en las ciencias de la computación y el aprendizaje en general. Un creciente cuerpo de investigación sobre experiencias de aprendizaje temprano de las ciencias de la computación conectadas, desconectadas y combinadas sugiere que los niños de 3 y 4 años pueden participar en actividades de las ciencias de la computación, como crear, ejecutar y depurar un programa de computación, y pueden aprender y aplicar conceptos clave de ciencias de la computación, que incluyen secuenciación, bucles, parámetros y condicionales(e.g., Bers, 2008a, 2008b;

Elkin, Sullivan y Bers, 2014; Flannery & Bers, 2013; Kazakoff, Sullivan y Bers, 2013; Martínez, Gómez y Benotti, 2015; Morgado, Cruz y Kahn, 2010; Sullivan, Elkin y Bers, 2015). Por ejemplo, Martínez y sus colegas (2015) descubrieron que los niños de 3 a 5 años podían aprender y aplicar conceptos básicos de las ciencias de la computación de secuenciación, condicionales y bucles al participar en una combinación de actividades desconectadas y actividades de robótica conectada.

Del mismo modo, Gordon, Ackerman y Breazeal (2015) utilizaron robots sociales para ayudar a los estudiantes de preescolar a explorar varios conceptos de ciencias de la computación, incluida la lógica basada en eventos, la secuencia y el no determinismo. A partir de cuatro años de trabajo con más de 150 niños de 3 a 5 años, Morgado y sus colegas (2010) desarrollaron un "libro de cocina" de temas computacionales preescolares que van desde la sintaxis de programación simple hasta temas más complejos de entrada/salida y relaciones cliente/servidor. Otros han investigado si los niños pequeños pueden transferir el conocimiento de las actividades relacionadas con las ciencias de la computación a otras áreas de contenido. Por ejemplo, Kazakoff y sus colegas (2013) mostraron que los niños de preescolar y jardín de infantes que participaron en un taller de robótica y programación de una semana aumentaron significativamente sus habilidades de secuencia de historias desde antes hasta después del taller, lo que sugiere una transferencia de conocimiento de las ciencias de la computación en el contexto de la alfabetización. La investigación sobre robots sociales también apoya la transferencia del aprendizaje de las ciencias de la computación a la alfabetización (por ejemplo, Fridin, 2014; Gordon y Breazeal, 2015; Kory Westlund y Breazeal, 2015; Movellan, Eckhardt, Virnes y Rodríguez, 2009). Un estudio de Kory Westlund y Breazeal (2015) encontró que los niños en edad preescolar aprendieron nuevas palabras y crearon historias al participar en un juego de contar historias con un robot social, mientras que Movellan y sus colegas (2009) mostraron hallazgos similares incluso con niños más pequeños de entre 18 y 24 meses, donde el compromiso con un robot social aumentó el conocimiento de las palabras de vocabulario objetivo en un 27% durante un período de dos semanas. Además, una revisión de Clements (1999) mostró que la participación en un entorno estructurado de programación de computadoras ayudó al sentido numérico, la memoria visual y las habilidades del lenguaje de los niños pequeños. Es importante destacar que la programación computacional no estructurada

tuvo poco efecto en la capacidad de los niños para aprender conceptos de las ciencias de la computación o transferir el aprendizaje, destacando la necesidad de un andamiaje intencional del profesor que ayude a los estudiantes a hacer conexiones entre la programación de la computadora y otras experiencias académicas y cotidianas (Clements, 1999).

Finalmente, la participación temprana en la programación de computadoras se ha observado como una forma de aumentar el interés de los estudiantes en seguir carreras relacionadas con la informática y la ciencia, la tecnología, la ingeniería y las matemáticas (STEM), así como disminuir los estereotipos basados en el género de los niños antes de que se formalicen por completo en la escuela primaria posterior (Madill et al., 2007; Metz, 2007; Steele, 1997). El trabajo anterior sugiere que el interés y la autoeficacia de STEM y las ciencias de la computación disminuyen en la escuela primaria y media tardía y persisten durante la educación secundaria y postsecundaria (por ejemplo, Google y Gallup, 2015; Pajares, 2006; Unfried, Faber y Wiebe, 2014) . Esta tendencia es aún más severa para las mujeres y las minorías subrepresentadas (por ejemplo, Corbett, Hill y St. Rose, 2008; Google, 2014; Master, Cheryan y Meltzoff, 2016; Weinburgh, 1995), lo que sugiere la necesidad de proporcionar experiencias tempranas con computadoras relacionadas con la ciencia que se basan y continúan a lo largo de la experiencia educativa de los niños de K-12. Este apéndice y el Marco de Ciencias de la Computación K – 12 en sí ofrecen una base para abordar esta necesidad de hacer que las oportunidades de aprendizaje de las ciencias de la computación estén disponibles para todos los estudiantes a medida que progresan desde pre-kindergarten hasta la escuela secundaria y más allá.

Referencias

American Academy of Pediatrics. (2015). *Growing up digital: Media research symposium*. Washington, DC: Author. Retrieved from https://aap.org/en-us/Documents/digital_media_symposium_proceedings.pdf

Anderson, C. A., & Bushman, B. J. (2001). Effects of violent video games on aggressive behavior, aggressive cognition, aggressive affect, physiological arousal, and prosocial behavior: A meta-analytic review of the scientific literature. *Psychological Science, 12*, 353–359.

Bers, M. U. (2008a). *Blocks to robots: Learning with technology in the early childhood classroom*. New York, NY: Teachers College Press.

Bers, M. U. (2008b). *Blocks, robots and computers: Learning about technology in early childhood*. New York, NY: Teachers College Press.

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood: Revisiting developmental assumptions through new technologies. In I. R. Berson & M. J. Berson (Eds.), *High-tech tots: Childhood in a digital world* (p. 49–70). Greenwich, CT: Information Age Publishing.

Bers, M. U., Ponte, I., Juelich, K., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics into early childhood education. *Information Technology in Childhood Education Annual, 2002*(1), 123–145.

Blackwell, C. K., Wartella, E., Lauricella, A. R., & Robb, M. (2015). *Technology in the lives of educators and early childhood programs: 2014 survey of early childhood educators*. Report for the Center on Media and Human Development, Northwestern University; the Fred Rogers Center, Latrobe, PA; and the National Association for the Education of Young Children, Washington, DC.

Clements, D. (1999). The future of educational computing research: The case of computer programming. *Information Technology in Childhood Education Annual*, 147–179.

Clements, D., & Nastasi, B. (1999). Metacognition, learning, and educational computer environments. *Information Technology in Childhood Education Annual, 1*, 5–38.

Copple, C., & Bredekamp, S. (2009). *Developmentally appropriate practice in early childhood programs serving children from birth through age 8*. Washington, DC: National Association for the Education of Young Children.

Corbett, C., Hill, S. C., & St. Rose, A. (2008). *Where the girls are: The facts about gender equity in education*. Washington, DC: American Association of University Women. Retrieved from http://www.aauw.org/files/2013/02/Where-the-Girls-Are-The-Facts-About-Gender-Equity-in-Education.pdf

Cordes, C., & Miller, E. (2000). *Fool's gold: A critical look at computers in childhood*. New York, NY: Alliance for Childhood. Retrieved from http://www.allianceforchildhood.net/projects/computers/computers_reports.htm

diSessa, A. A. (2000). Changing minds: *Computers, learning and literacy*. Cambridge, MA: MIT Press.

Donohue, C. (2015). Technology and digital media as tools for teaching and learning in the digital age. In C. Donohue (Ed.), *Technology and digital media in the early years: Tools for teaching and learning* (pp. 21–35). New York, NY: Routledge. Washington, DC: National Association for the Education of Young Children.

Dooley, C. M., Flint, A. S., Holbrook, T., May, L., & Albers, P. (2011). The digital frontier in early childhood education. *Language Arts, 89*(2), 83–85.

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice, 13*, 153–169.

Fisch, S. M., & Truglio, R. T. (Eds.). (2001). *"G" is for growing: Thirty years of research on children and Sesame Street*. Mahwah, NJ: Lawrence Erlbaum Associates.

Flannery, L. P., & Bers, M. U. (2013). Let's dance the "robot hokey-pokey!" Children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education, 46*(1), 81–101.

Fridin, M. (2014). Storytelling by a kindergarten social assistive robot: A tool for constructive learning in preschool education. *Computers & Education, 70*, 53–64.

Google. (2014). *Women who choose computer science—what really matters: The critical role of exposure and encouragement.* Mountain View, CA: Author. Retrieved from https://docs.google.com/file/d/0B-E2rcvhnlQ_a1Q4VUxWQ2dtTHM/edit

Google & Gallup. (2015). *Images of computer science: Perceptions among students, parents, and educators in the U.S.* Retrieved from http://g.co/cseduresearch

Gordon, M., Ackermann, E., & Breazeal, C. (2015, March). Social robot toolkit: Tangible programming for young children. In *Proceedings of the 10th ACM/IEEE International Conference on Human-Robot Interaction: Extended Abstracts,* Portland, OR.

Gordon, G., & Breazeal, C. (2015, January). Bayesian active learning-based robot tutor for children's word-reading skills. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence,* Austin, TX.

Haugland, S. W. (1992). The effect of computer software on preschool children's developmental gains. *Journal of Computing in Childhood Education, 3*(1), 15–30.

Horn, M. S., AlSulaiman, S., & Koh, J. (2013, June). Translating Roberto to Omar. In *Proceedings of the 12th ACM International Conference on Interaction Design and Children,* New York, NY.

Horn, M. S., Crouser, R. J., & Bers, M. U. (2012). Tangible interaction and learning: The case for a hybrid approach. *Personal and Ubiquitous Computing, 16*(4), 379–389.

Horn, M. S., & Jacob, R. J. K. (2007). Designing tangible programming languages for classroom use. In *Proceedings of TEI'07 First International Conference on Tangible and Embedded Interaction*, Baton Rouge, LA.

Huston, A. C., Anderson, D. R., Wright, J. C., Linebarger, D. L., & Schmitt, K. L. (2001). Sesame Street viewers as adolescents: The recontact study. In S. M. Fisch & R. T. Truglio (Eds.), *"G" is for growing: Thirty years of research on children and Sesame Street* (pp.131–144). Mahwah, NJ: Lawrence Erlbaum Associates.

Kazakoff, E. R., Sullivan, A., & Bers, M. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*, 245–255.

Kory Westlund, J. M., & Breazeal, C. (2015, March). The interplay of robot language level with children's language learning during storytelling. In *Proceedings of the 10th Annual ACM/IEEE International Conference on Human-Robot Interaction* (pp. 65–66).

Lindahl, M., & Folkesson, A. (2012). Can we let computers change practice? Educators' interpretations of preschool tradition. *Computers in Human Behavior, 28*(5), 1728–1737. doi: 10.1016/j.chb.2012.04.012

Madill, H., Campbell, R. G., Cullen, D. M., Armour, M. A., Einsiedel, A. A., Ciccocioppo, A. L., . . . & Coffin, W. L. (2007). Developing career commitment in STEM-related fields: Myth versus reality. In R. J. Burke & M. C. Mattis (Eds.), *Women and minorities in science, technology, engineering and mathematics: Upping the numbers* (pp. 210–244). Northampton MA: Edward Elgar Publishing.

Martinez, C., Gomez, M. J., & Benotti, L. (2015, July). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 159–164), Vilanius, Lithuania.

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology, 108*(3), 1–14. doi: 10.1037/edu0000061

Metz, S. S. (2007). Attracting the engineering of 2020 today. In R. J. Burke & M. C. Mattis (Eds.), *Women and minorities in science, technology, engineering and mathematics: Upping the numbers* (pp. 184–209). Northampton, MA: Edward Elgar Publishing.

Morgado, L., Cruz, M., & Kahn K. (2010). Preschool cookbook of computer programming topics. *Australasian Journal of Educational Technology, 26*(3), 309–326.

Movellan, J., Eckhardt, M., Virnes, M., & Rodriguez, A. (2009, March). Sociable robot improves toddler vocabulary skills. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, San Diego, CA.

National Association for the Education of Young Children & Fred Rogers Center. (2012). *Position statement: Technology and young children*. Washington, DC: Author. Retrieved from http://www.naeyc.org/content/technology-and-young-children

Pajares, F. (2006). Self-efficacy during childhood and adolescence: Implications for teachers and parents. In F. Pajares & T. C. Urdan (Eds.), *Self-efficacy beliefs of adolescents* (pp. 339–367). Greenwich, CA: Information Age Publishing.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Pasnik, S., & Llorente, C. (2013). *Preschool teachers can use a PBS KIDS transmedia curriculum supplement to support young children's mathematics learning: Results of a randomized controlled trial. A report to the CPB-PBS Ready To Learn Initiative*. Waltham, MA, and Menlo Park, CA.

Penuel, W. R., Bates, L., Gallagher, L. P., Pasnik, S., Llorente, C., Townsend, E., . . . & VanderBorght, M. (2012). Supplementing literacy instruction with a media-rich intervention: Results of a randomized controlled trial. *Early Childhood Research Quarterly, 27*(2), 115–127. doi: 10.1016/j.ecresq

Perlman, R. (1976). *Using computer technology to provide a creative learning environment for preschool children. Logo memo no 24*. Cambridge, MA: MIT Artificial Intelligence Laboratory.

Rideout, V. J. (2013). *Zero to eight: Children's media use in America 2013*. San Francisco, CA: Common Sense Media.

Steele, C. M. (1997). A threat in the air: How stereotypes shape intellectual identity and performance. *American Psychologist, 52*, 613–629.

Sullivan, A., Elkin, M., & Bers, M. (2015, June). KIBO robot demo: Engaging young children in programming and engineering. In *Proceedings of the 14th ACM International Conference on Interaction Design and Children,* Medford, MA.

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

Unfried, A., Faber, M., & Wiebe, E. N. (2014, April). *Gender and student attitudes toward science, technology, engineering, and mathematics.* Paper presented at the annual meeting of the American Educational Research Association, Philadelphia, PA.

Weinburgh, M. (1995). Gender differences in student attitudes toward science: A meta analysis of the literature from 1970 to 1991. *Journal of Research in Science Teaching, 32*(4), 387–398.

Wyeth, P. (2008). How young children learn to program with sensor, action, and logic blocks. *International Journal of the Learning Sciences, 17*(4), 517–550.

Apéndice E: Bibliografía de la Investigación del Marco

Abelson, H., & diSessa. A. (1986). *Turtle geometry*. Cambridge, MA: MIT Press.

Achieve. (2015). *The role of learning progressions in competency-based pathways*. Retrieved from http://www.achieve.org/files/Achieve-LearningProgressionsinCBP.pdf

Aho, A. (2011, January). Computation and computational thinking. *ACM Ubiquity.* Retrieved from http://ubiquity.acm.org/article.cfm?id=1922682

Aivaloglou. E., & Hermans, F. (2016, September). How kids code and how we know: An exploratory study on the Scratch repository. In *Proceedings of the Twelfth Annual International Conference on International Computing Education Research* (pp. 53–61).

Akgün, L., & Özdemir, M. E. (2006). Students' understanding of the variable as general number and unknown: A case study. *The Teaching of Mathematics*, *9*(1), 45–51.

Anderson, N., Lankshear, C., Timms, C., & Courtney, L. (2008). "Because it's boring, irrelevant and I don't like computers": Why high school girls avoid professionally-oriented ICT subjects. *Computers & Education, 50*(4), 1304–1318.

Arkansas Department of Education. (2016). *Computer science curriculum framework documents*. Retrieved from http://www.arkansased.gov/divisions/learning-services/curriculum-and-instruction/curriculum-framework-documents/computer-science

Baker, T. R., & White, S. H. (2003). The effects of GIS on students' attitudes, self-efficacy, and achievement in middle school science classrooms. *Journal of Geography*, *102*(6), 243–254.

Barr, D., Harrison, J., & Conery, L. (2011, March/April). Computational thinking: A digital age skill for everyone. *Learning and Leading with Technology,* 20–23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54.

Battista, M. T., & Clements, D. H. (1986). The effects of Logo and CAI problem-solving environments on problem-solving abilities and mathematics achievement. *Computers in Human Behavior, 2*(3), 183–193.

Baxter, G. P., & Glaser, R. (1997). *An approach to analyzing the cognitive complexity of science performance assessments* (CSE Technical Report 452). Los Angeles, CA: University of California, Center for the Study of Evaluation, National Center for Research on Evaluation, Standards, and Student Testing.

Beaumont-Walters, Y., & Soyibo, K. (2001). An analysis of high school students' performance on five integrated science process skills. *Research in Science & Technological Education*, *19*(2), 133–145.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20–29.

Bender, E., Hubwieser, P., Schaper, N., Margaritis, M., Berges, M., Ohrndorf, L., . . . Schubert, S. (2015). Towards a competency model for teaching computer science. *Peabody Journal of Education, 90*(4), 519–532. doi: 10.1080/0161956X.2015.1068082

Ben-Zvi, D., & Arcavi, A. (2001). Junior high school students' construction of global views of data and data representations. *Educational Studies in Mathematics*, *45*, 35–65.

Berger, B., Daniels, N. M., & Yu, Y. W. (2016). Computational biology in the 21st century: Scaling with compressive algorithms. *Communications of the ACM, 59*(8), 72–80. doi: 10.1145/2957324

Bers, M. (2008). *Blocks to robots: Learning with technology in the early childhood classroom*. New York City, NY: Teachers College Press.

Bers, M. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2). Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look* (SRI technical report). Menlo Park, CA: SRI International. Retrieved from http://pact.sri.com/resources.html

Biggers, M., Brauer, A., & Yilmaz, T. (2008). Student perceptions of computer science: A retention study comparing graduating seniors vs. CS leavers. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (pp. 402–406), Portland, OR.

Blickenstaff, J.C. (2005). Women and science careers: Leaky pipeline or gender filter? *Gender and Education, 17*(4), 369–386. doi: 10.1080/09540250500145072

Boston Public Schools. (2016). *Computer science in BPS.* Retrieved from http://www.bostonpublicschools.org/domain/2054

Brennan, K., Balch, C., & Chung, M. (2014). *Creative computing.* Retrieved from http://scratched.gse.harvard.edu/guide/files/CreativeComputing20141015.pdf

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design.* Paper presented at the annual meeting of the American Educational Research Association, Vancouver, BC, Canada.

Brown, N. C. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), Article 9. doi: 10.1145/2602484

Buchanan, G., Farrant, S., Jones, M., Thimbleby, H., Marsden, G., & Pazzani, M. (2001). Improving mobile Internet usability. In *Proceedings of the 10th International World Wide Web Conference* (pp. 673–680), Hong Kong.

Buffardi, K., & Edwards, S. H. (2013, August). Effective and ineffective software testing behaviors by novice programmers. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 83–90).

Buffum, P. S., Martinez-Arocho, A. G., Frankosky, M. H., Rodriguez, F. J., Wiebe, E. N., & Boyer, K. E. (2014, March). CS Principles goes to middle school: Learning how to teach "big data." In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 151–156), Atlanta, GA. doi: 10.1145/2538862.2538949

Burns, K., & Polman, J. (2006). The impact of ubiquitous computing in the Internet age: How middle school teachers integrated wireless laptops in the initial stages of implementation. *Journal of Technology and Teacher Education, 14*(2), 363–385.

Buzzetto-More, N., Ukoha, O., & Rustagi, N. (2010). Unlocking the barriers to women and minorities in computer science and information systems studies: Results from a multi-methodolical study conducted at two minority serving institutions. *Journal of Information Technology Education, 9,* 115–131.

Campbell, P. F., & McCabe, G. P. (1984). Predicting the success of freshmen in a computer science major. *Communications of the ACM, 27*(11), 1108–1113.

Carter, L. (2006). Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp. 27–31), Houston, TX.

Century, J., & Cassata, A. (in press). Measuring implementation and implementation research: Finding common ground on what, how and why. *Review of Research in Education, Centennial Edition.* American Educational Research Association.

Century, J., Cassata, A., Rudnick, M., & Freeman, C. (2012). Measuring enactment of innovations and the factors that affect implementation and sustainability: Moving toward common language and shared conceptual understanding. *Journal of Behavioral Health Services & Research, 39*(4), 343–361.

Cernavskis, A. (2015, June 25). In San Francisco, computer science for all . . . soon. *The Hechinger Report.* Retrieved from http://hechingerreport.org/san-francisco-plans-to-be-first-large-district-to-bring-computer-science-to-all-grades/

Chou, J.-R., & Hsiao, S.-W. (2007). A usability study on human-computer interface for middle-aged learners. *Computers in Human Behavior, 23*, 2040–2063. doi: 10.1016/j.chb.2006.02.011

Clarke, V. A., & Teague, G. J. (1996). Characterizations of computing careers: Students and professionals disagree. *Computers in Education, 26*(4), 241–246.

Clements, D. H., & Gullo, D. F. (1974). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*(6), 1051–1058.

Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. In *Proceedings of XP2000*, Sardinia, Italy.

College Board. (2016). *AP Computer Science Principles course and exam description.* New York, NY: Author. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

Common Sense Media. (2012). *Digital literacy and citizenship in a connected culture.*

Computational thinking in STEM: Skills taxonomy. (n.d.). Retrieved from http://gk12northwestern.wikispaces.com/file/view/CTSTEM_Skills_Taxonomy_4teachers.pdf

Computer Science Teachers Association & International Society for Technology in Education. (2011). Computational thinking teacher resources (2nd ed.). Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf

Computer Science Teachers Association Standards Task Force. (2011). *CSTA K–12 computer science standards, revised 2011.* New York, NY: Computer Science Teachers Association and Association for Computing Machinery. Retrieved from http://www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf

Computer Science Teachers Association Standards Task Force. (2016). *[Interim] CSTA K–12 computer science standards, revised 2016.* Springfield, OR: Computer Science Teachers Association and Association for Computing Machinery. Retrieved from http://www.csteachers.org/?page=CSTA_Standards

Computer Science Teachers Association Teacher Certification Task Force. (2008). *Ensuring exemplary teaching in an essential discipline: Addressing the crisis in computer science teacher certification.* New York, NY: Computer Science Teachers Association and the Association for Computing Machinery.

Cooper, S., Grover, S., Guzdial, M., & Simon, B. (2014). A future for computing education. *Communications of the ACM, 57*(11), 34–46.

CPALMS. (2016). *Science standards and access points.* Retrieved from http://www.cpalms.org/Downloads.aspx

Cotten, S. R., Shank, D. B., & Anderson, W. A. (2014). Gender, technology use and ownership, and media-based multitasking among middle school students. *Computers in Human Behavior, 35,* 99–106.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking: A guide for teachers.* Retrieved from the Computing at School website: http://computingatschool.org.uk/computationalthinking

Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016, February). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (pp. 1438–1449).

De Boulay, B. (1989). Some difficulties of learning to program. In E. Soloway and J. Spohrer (Eds.), *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Denner, J. (2011). What predicts middle school girls' interest in computing? *International Journal of Gender, Science and Technology 3*(1), 53–69.

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education, 46*(3), 277–296.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58,* 240–249.

Denning, P. J. (2013). The science in computer science. *Communications of the ACM, 56*(5), 35–38. doi: 10.1145/2447976.2447988

Denning, P. J., & Martell, C. (2015). *Great principles of computing*. Cambridge, MA: MIT Press.

DevTech Research Group at Tufts University. (2016). *The early childhood robotics network: Early childhood robotics curriculum*. Retrieved from http://tkroboticsnetwork.ning.com/page/robotics-curriculum

Di Vano, D., & Mirolo, C. (2011). "Computer science and nursery rhymes": A learning path for the middle school. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp.238–242), New York, NY

Dodds, Z., & Erlinger, M. (2013). MyCS: Building a middle-years CS curriculum. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (p. 330), New York, NY.

Dorling, M., & Browning, P. (2015). *Computing progression pathways KS1 (Y1) to KS3 (Y9) by topic*. Computing at School. Retrieved from http://community.computingatschool.org.uk/resources/1692

Dwyer, H., Hill, C., Carpenter, S., Harlow, D., & Franklin, D. (2014, March). Identifying elementary students' pre-instructional ability to develop algorithms and step-by-step instructions. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 511–516).

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice, 13*, 153–169. Retrieved from http://www.jite.org/documents/Vol13/JITEv13IIPvp153-169Elkin882.pdf

England Department for Education. (2013, September 11). *National curriculum in England: Computing programmes of study*. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Evers, V., & Day, D. (1997). The role of culture in interface acceptance. In S. Howard, J. Hammond, & G. Lindgaard (Eds.), *Proceedings of Human-Computer Interaction INTERACT'97* (pp. 260–267). Sydney, Australia: Chapman & Hall.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97.

Fields, D. A., Giang, M., and Kafai, Y. (2014). Programming in the wild: Trends in youth computational participation in the online Scratch community. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 2–11), New York, NY.

Fincher, S. (2015). What are we doing when we teach computing in schools? *Communications of the ACM, 58*(5), 24–26. doi: 10.1145/2742693

Fish, M. C., Gross, A. L., & Sanders, J. S. (1986). The effect of equity strategies on girls' computer usage in school. *Computers in Human Behavior, 2*, 127–134.

Forte, A., & Guzdial, M. (2004). Computers for communication, not calculation: Media as a motivation and context for learning. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Track 4, Volume 4* (p. 40096.1). Washington, DC: IEEE Computer Society. doi: 10.1109/HICSS.2004.1265259

Franklin, D., Conrad, P., Aldana, G., & Hough, S. (2011, March). Animal Tlatoque: Attracting middle school students to computing through culturally-relevant themes. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 453–458), Dallas, TX.

Franklin, D. F. (2015, February). Putting the computer science in computing education research. *Communications of the ACM, 58*(2), 34–36.

Franklin, D., Hill, C., Dwyer, H., Iveland, A., Hansen, A., & Harlow, D. (2016). Initialization in Scratch: Seeking knowledge transfer. In *Proceedings of the 47th ACM Symposium on Computing Science Education* (pp. 217–222), Memphis, TN.

Franklin, D., Hill, C., Dwyer, H., Iveland, A., Killina, A., & Harlow, D. (2015). Getting started in teaching and researching computer science in the elementary classroom. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 552–557), Kansas, City, MO.

Freischlad, S. (2007, November). Exploration module for understanding the functionality of the Internet in secondary education. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research, Volume 88* (pp. 183–186). Australian Computer Society, Inc.

Friedman, B. (1996). Value-sensitive design. *Interactions, 3*(6), 16–23.

Friend, M., & Robert, C. (2013). Efficient egg drop contests: How middle school girls think about algorithmic efficiency. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 99–106).

Garfield, J. (1995). How students learn statistics. *International Statistical Review/Revue Internationale de Statistique, 63*(1), 25–34.

Georgia Department of Education. (2016). *Information technology career cluster.* Retrieved from http://www.gadoe.org/Curriculum-Instruction-and-Assessment/CTAE/Pages/cluster-IT.aspx

Goldberg, D., Grunwald, D., Lewis, C., Feld, J., Donley, K., & Edbrooke, O. (2013). Addressing 21st century skills by embedding computer science in K–12 classes. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 637–638).

Goldberg, D. S., Grunwald, D., Lewis, C., Feld, J. A., & Hug, S. (2012). Engaging computer science in traditional education: The ECSITE project. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 351–356).

Goldschmidt, D., MacDonald, I., O'Rourke, J., & Milonovich, B. (2011). An interdisciplinary approach to injecting computer science into the K–12 classroom. *Journal of Computing Science in College, (26)*6, 78–85.

Good, C., Rattan, A., & Dweck, C. S. (2012). Why do women opt out? Sense of belonging and women's representation in mathematics. *Journal of Personality and Social Psychology, 102*(4), 700–717.

Goode, J., & Chapman, G. (2013). *Exploring computer science: A high school curriculum exploring what computer science is and what it can do* (Version 5.0).

Grosslight, L., Unger, C., Jay, E., & Smith, C. L. (1991). Understanding models and their use in science: Conceptions of middle and high school students and experts. *Journal of Research in Science Teaching, 28*(9), 799–822.

Grover, S. (2015). *"Systems of assessments" for deeper learning of computational thinking in K–12.* Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL.

Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K–12. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 57–62).

Grover, S., Jackiw, N., & Lundh, P. (2015). Thinking outside the box: Integrating dynamic mathematics to advance computational thinking for diverse student populations. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1543062

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 41*(1), 38–43.

Grover, S., & Pea, R. (2013b). Using a discourse-intensive pedagogy and Android's App Inventor for introducing computational concepts to middle school students. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 723–728), Denver, CO.

Grover, S., Pea, R., & Cooper, S. (2014). Remedying misperceptions of computer science among middle school students. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 343–348), Atlanta, GA.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education, 25*(2), 199–237.

Grover, S., Pea, R., & Cooper, S. (2016, March). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 552–557), Memphis, TN.

Grover, S., Rutstein, D., & Snow, E. (2016, March). "What is a computer?": What do secondary school students think? In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 564–569), Memphis, TN.

Guernsey, L. (2012). *Screen time: How electronic media from baby videos to educational software affects your young child.* Philadelphia, PA: Basic Books.

Guzdial, M. (2013a). Human-centered computing: A new degree for Licklider's world. *Communications of the ACM, 56*(5), 32–34. doi: 10.1145/2447976.2447987

Guzdial, M. (2013b). Exploring hypotheses about media computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 19–26).

Guzdial, M. (2016). *Learner-centered design of computing education: Research on computing for everyone.* Synthesis lectures on human-centered informatics. Morgan and Claypool.

Hanks, B. (2008). Problems encountered by novice pair programmers. *Journal on Educational Resources in Computing (JERIC), 7*(4), Article 2.

Hansen, A., Dwyer, H., Hill, C., Iveland, A., Martinez, T., Harlow, D., & Franklin, D. (2015). Interactive design by children: A construct map for programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 267–270).

Hansen, A., Iveland, A., Carlin, C., Harlow, D., & Franklin, D. (2016, June). User-centered design in block-based programming: Developmental and pedagogical considerations for children. In *Proceedings of the 15th International Conference on Interaction Design and Children* (pp. 147–156).

Hansen, A. K., Hansen, E. R., Dwyer, H. A., Harlow, D. B., & Franklin, D. (2016). Differentiating for diversity: Using universal design for learning in computer science education. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education* (pp. 376–381), Memphis, TN.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive Learning Environments, 1*(1), 1–32.

Haspékian, M. (2003). Between arithmetic and algebra: A space for the spreadsheet? Contribution to an instrumental approach. In *Proceedings of the 3rd Conference of the European Society for Research in Mathematics Education,* Bellaria, Italy.

Herl, H. E., O'Neil, Jr., H. F., Chung, G. K. W. K., & Schacter, J. (1999). Reliability and validity of a computer-based knowledge mapping system to measure content understanding. *Computers in Human Behavior, 15*, 315–333.

Hoadley, C., Xu, H., Lee, J. J., & Rosson, M. B. (2010). Privacy as information access and illusory control: The case of the Facebook News Feed privacy outcry. *Electronic Commerce Research and Applications, 9*(1), 50–60.

Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M. N., . . . Schubert, S. (2011, June). Computer science/informatics in secondary education. In *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education—Working Group Reports* (pp. 19–38).

Hubwieser, P., Magenheim, J., Mühling, A., & Ruf, A. (2013, August). Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 1–8).

Idaho State Department of Education. (2016). *Idaho K–12 computer science standards.* Draft document.

Indiana Department of Education. (2016). *Science & computer science.* Retrieved from http://www.doe.in.gov/standards/science-computer-science

International Society for Technology in Education. (2012). *Computational thinking toolkit.* Retrieved from http://www.iste.org/learn/computational-thinking/ct-toolkit

International Society for Technology in Education. (2016). *ISTE standards for students.* Retrieved from https://www.iste.org/resources/product?id=3879&childProduct=3848

International Society for Technology in Education & Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K–12 education.* Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Isaacs, A., Binkowski, T. A., Franklin, D., Rich, K., Strickland, C., Moran, C., . . . Maa, W. (2016). Learning trajectories for integrating K–5 computer science and mathematics. In *2016 CISE/EHR principal investigator & community meeting for CS in STEM project description booklet* (p. 79). Retrieved from https://www.ncwit.org/sites/default/files/file_type/pi_book_compressed_2016.pdf

Israel, M., Pearson, J., Tapia, T., Wherfel, Q., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross case analysis. *Electronic Commerce Research and Applications, 82,* 263–279. doi: 10.1016/j.compedu.2014.11.022

Israel, M., Wherfel, Q., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 students with disabilities to learn computational thinking and computer programming. *TEACHING Exceptional Children, 48*(1), 45–53.

Jarman, S., & Bell, T. (2014, November). A game to teach network communication reliability problems and solutions. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 43–49).

Joint Task Force on Computing Curricula, Association for Computing Machinery, & IEEE Computer Society. (2013, December 20). *Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science*. Association for Computing Machinery and IEEE.Retrieved from https://www.acm.org/education/CS2013-final-report.pdf

Kafai, Y. (2016). From computational thinking to computational participation in K–12 education. *Communications of the ACM, 59*(8), 26–27. doi: 10.1145/2955114

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational Psychologist, 50*(4), 313–334.

Kafai, Y. B., Franke, M. L., Ching, C. C., & Shih, J. C. (1998). Game design as an interactive learning environment for fostering students' and teachers' mathematical inquiry. *International Journal of Computers for Mathematical Learning, 3*(2), 149–184. doi: 10.1023/A:1009777905226

Kay, A. (n.d.). Squeak Etoys, children, and learning. *Viewpoints Research Institute (VPRI Research Note RN-2005-001).* Retrieved from http://www.squeakland.org/resources/articles/article.jsp?id=1009

Kay, A. (2003). Afterward: Our human condition "from space." In B. J. Allen-Conn & K. Rose (Eds.), *Powerful ideas in the classroom: Using Squeak to enhance math and science learning* (pp. 73–79). Glendale, CA: Viewpoints Research Institute.

Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia, 21*(4), 371–391.

Kazakoff, E. R. (2015). Technology-based literacies for young children: Digital literacy through learning to code. In K. L. Heider & M. Renck Jalongo (Eds.), *Young children and families in the information age* (pp. 43–60). New York, NY: Springer.

Koppelman, H. (2007). Exercises as a tool for sharing pedagogical knowledge. In *Proceedings of the 12th Annual SIGCSE Conference in Innovation and Technology in Computer Science Education* (p. 361). doi: 10.1145/1268784.1268933

Koppelman, H. (2008, June). Pedagogical content knowledge and educational cases in computer science: An exploration. In *Proceeding of the Informing Science and IT Education Conference (InSITE)* (pp. 125–133). Varna, Bulgaria.

Kothiyal, A., Majumdar, R., Murthy, S., & Iyer, S. (2013, August). Effect of think-pair-share in a large CS1 class: 83% sustained engagement. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 137–144).

Kreitmayer, S., Rogers, Y., Laney, R., & Peake, S. (2012, May). From participatory to contributory simulations: Changing the game in the classroom. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 49–58).

Kuhn, A., McNally, B., Schmoll, S., Cahill, C., Lo, W. T., Quintana, C., & Delen, I. (2012, May). How students find, evaluate and utilize peer-collected annotated multimedia data in science inquiry with zydeco. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 3061–3070).

Kurland, D. M., & Pea, R. D. (1985). Children's mental models of recursive LOGO programs. *Journal of Educational Computing Research, 1*(2), 235–243.

Ladner, R., & Israel, M. (2016). "For all" in "computer science for all." *Communications of the ACM, 59*(9), 26–28.

LeBlanc, M. D., & Dyer, B. D. (2004). Bioinformatics and computing curricula 2001: Why computer science is well positioned in a post-genomic world. *ACM SIGCSE Bulletin*, *36*(4), 64–68.

Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12*(1), 3–4. Retrieved from http://www.csteachers.org/resource/resmgr/Voice/csta_voice_03_2016.pdf

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K–8 curriculum. *ACM Inroads, 5*(4), 64–71.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37.

Lee, M. J., & Ko, A. J. (2011). Personifying programming tool feedback improves novice programmers' learning. In *Proceedings of the Seventh International Workshop on Computing Education Research* (pp. 109–116), Providence, RI. doi: 10.1145/2016911.2016934

Leidner, D. E., & Kayworth, T. (2006). A review of culture in information systems research: Toward a theory of information technology culture conflict. *MIS Quarterly*, *30*(2), 357–399.

Leutenegger, S., & Edgington, J. (2007, March). A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin 39*(1), 115–118.

Lewis, C. M. (2012). *Applications of out-of-domain knowledge in students' reasoning about computer program state* (Doctoral dissertation).

Lewis, C. M. (2014, July). Exploring variation in students' correct traces of linear recursion. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (pp. 67–74).

Lewis, C. M. (2016). You wouldn't know it from SIGCSE proceedings, but we don't only teach CS1 (Abstract Only). In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (p. 494).

Lewis, C. M., & Shah, N. (2012). Building upon and enriching grade four mathematics standards with programming curriculum. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 57–62).

Lewis, C. M., & Shah, N. (2015, July). How equity and inequity can emerge in pair programming. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 41–50).

Li, C., Dong, Z., Untch, R. H., & Chasteen, M. (2013). Engaging computer science students through gamification in an online social network based collaborative learning environment. *International Journal of Information and Education Technology, 3*(1), 72–77. doi: 10.7763/IJIET.2013.V3.237

Lishinski, A., Good, J., Sands, P., & Yadav, A. (2016, September). Methodological rigor and theoretical foundations of CS education research. In *Proceedings of the Twelfth Annual International Conference on International Computing Education Research* (pp. 161–169).

Lou, Y., Abrami, P.C., & d'Apollonia, S. (2001). Small group and individual learning with technology: A meta-analysis. *Review of Educational Research, 71*, 449–521.

Lowther, D. L., Bassoppo-Moyo, T., & Morrison, G. R. (1998). Moving from computer literate to technologically competent: The next educational reform. *Computers in Human Behavior, 14*(1), 93–109.

Lu, J. J., & Fletcher, G. H. L. (2009, March). Thinking about computational thinking. In *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education* (pp. 260–264), Chattanooga, TN.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K–12? *Computers in Human Behavior, 41*, 51–61.

Malik, S., & Agarwal, A. (2012). Use of multimedia as a new educational technology tool—A study. *International Journal of Information and Education Technology, 2*(5), 468–471. doi: 10.7763.IJIET.2012.V2.181

Manduca, C., & Mogk, D., (2002, April). *Using data in undergraduate science classrooms: Final report on an interdisciplinary workshop at Carleton College*. Northfield, MN: Science Education Resource Center, Carleton College. Retrieved from http://serc.carleton.edu/usingdata/report.html

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Ronaldsson, L., & Settle, A. (2014, June). Computational thinking in K–9 education. In *Proceedings of the Working Group Reports of the 2014 Innovation & Technology in Computer Science Education Conference*, Uppsala, Sweden. doi: 10.1145/2713609.2713610

Marcus, B. (2015, August 12). The lack of diversity in tech is a cultural issue. *Forbes.* Retrieved from http://www.forbes.com/sites/bonniemarcus/2015/08/12/the-lack-of-diversity-in-tech-is-a-cultural-issue/#622c464a3577

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2010). *Stuck in the shallow end: Education, race, and computing.* Cambridge, MA: MIT Press.

Margolis, J., Ryoo, J., Sandoval, C., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads, 3*(4), 72–78.

Mark, J., & DeLyser, L. (2016). CSNYC knowledge forum: Launching research and evaluation for computer science education, for every school and every student in New York City. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1637654

Maryland State Department of Education. (2005). *Maryland technology education state curriculum.* Retrieved from http://mdk12.msde.maryland.gov/instruction/curriculum/technology_education/vsc_technologyeducation_standards.pdf

Maryland State Department of Education. (2015). *Computer science.* Retrieved from http://archives.marylandpublicschools.org/MSDE/divisions/dccr/cs.html

Massachusetts Department of Elementary and Secondary Education. (2016, June). *2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework.* Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

Matias, J. N., Dasgupta, S., & Hill, B. M. (2016, May). *Skill progression in Scratch revisited.* Paper presented at the Conference on Human Factors in Computing Systems, San Jose, CA.

Matuk, C., & King Chen, J. (2011, March). *WISE Ideas: A technology-enhanced curriculum to scaffold students' generating data, managing evidence, and reasoning about the seasons.* Teacher design focus group presented at the Cyberlearning Tools for STEM Education Conference, Berkeley, CA.

McCrickard, D. S., & Lewis, C. (2012). *Workshop on designing for cognitive limitations.* Paper presented at the Designing Interactive Systems Conference, Newcastle, UK.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010, August). Learning computer science concepts with Scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research* (pp. 69–76).

Mesaroş, A. M., & Diethelm, I. (2012, November). Ways of planning lessons on the topic of networks and the Internet. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 70–73).

Milenkovic, L., Acquavita, T., & Kim, D. (2015). Investigating conceptual foundations for a transdisciplinary model integrating computer science into the elementary STEM curriculum. Abstract. Retrieved from http://www.nsf.gov/awardsearch/showAward?AWD_ID=1542842

Milner, S. (1973). *The effects of computer programming on performance in mathematics.* Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Moore, T., Wick, M., & Peden, B. (1994). Assessing student's critical thinking skills and attitudes toward computer science. *ACM SIGSCE Bulletin, 26*(1), 263–267.

Morgan, C., Mariotti, M. A., & Maffei, L. (2009). Representation in computational environments: Epistemological and social distance. *International Journal of Computers for Mathematical Learning, 14*(3), 241–263.

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015, July). Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 21–29).

Moursund, D. (1983). *Introduction to computers in education for elementary and middle school teachers.* Eugene, OR: International Council for Computers in Education.

Moursund, D., & Ricketts, D. (2016). Computational thinking. *IAE-pedia.* Retrieved from http://iae-pedia.org/Computational_Thinking

National Association of State Directors of Career Technical Education Consortium & National Career Technical Education Foundation. (2012). *Common career technical core.* Silver Spring, MD: Authors.

National Governors Association Center for Best Practices & Council of Chief State School Officers. (2010). *Common core state standards.* Washington DC: Author.

National Research Council. (2007). *Taking science to school: Learning and teaching science in grades K–8.* Committee on Science Learning-Kindergarten Through Eighth Grade. R. A. Duschl, H. A. Schweingruber, & A. W. Shouse (Eds.). Board on Science Education, Center for Education. Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking.* Washington, DC: The National Academies Press. Retrieved from http://www.nap.edu/catalog/12840.html

National Research Council. (2012). *A framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

The New Zealand Curriculum Online. (2014). *Technology.* Retrieved from http://nzcurriculum.tki.org.nz/The-New-Zealand-Curriculum/Technology

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NH: Prentice Hall.

Next Generation Science Standards Lead States. (2013). *Next generation science standards: For states, by state*s. Washington, DC: The National Academies Press.

North, A. S., & Noyes, J. M. (2002). Gender influences on children's computer attitudes and cognitions. *Computers in Human Behavior, 18*, 135–150.

Ohrndorf, L. (2015, July). Measuring knowledge of misconceptions in computer science education. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 269–270).

Ouyang, Y., Wolz, U., & Rodger, S. H. (2010, March). Effective delivery of computing curriculum in middle school: Challenges and solutions. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 327–328). Milwaukee, WI.

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of Educational Research, 60*(1), 65–89.

Papert, S. (1971). *A computer laboratory for elementary schools.* Cambridge, MA: Massachusetts Institute of Technology.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York, NY: Basic Books.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*, 137–168.

Pea, R. D., Soloway, E., & Spohrer, J. C. (1987). The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics, 9*, 5–30.

Pellegrino, J. W., & Hilton, M. L. (Eds.). (2012). *Education for life and work: Developing transferable knowledge and skills in the 21st century.* Washington, DC: The National Academies Press.

Perkins, D. N., & Salomon, G. (1988). Teaching for transfer. *Educational Leadership, 46*(1), 22–32.

Perkovic, L., & Settle, A. (2009). *Computational thinking across the curriculum: A conceptual framework.* Retrieved from http://compthink.cs.depaul.edu/FinalFramework.pdf

Perkovic, L., Settle, A., Hwang, S., & Jones, J. (2010, June). A framework for computational thinking across the curriculum. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education* (pp. 123–127).

Pillars of cyber security. (n.d.). Retrieved from https://www.usna.edu/CyberCenter/si110/lec/pillarsCybSec/lec.html

Plant, E. A., Baylor, A. L., Doerr, C. E., & Rosenberg-Kima, R. B. (2009). Changing middle-school students' attitudes and performance regarding engineering with computer-based social models. *Computers & Education, 53,* 209–215.

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: What works? *Communications of the ACM, 56*(8), 34–36. doi: 10.1145/24920072492020

Proulx, V. K. (1993, January). Computer science in elementary and secondary schools. In *Proceedings of the IFIP TC3/*WG3.1/WG*3.5 Open Converence on Informatics and Changes in Learning.* Retrieved from http://www.ccs.neu.edu/home/vkp/Papers/Gmunden93.pdf

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 265–269).

Resnick, M., Ocko, S., & Papert, S. (1988). LEGO, Logo, and design. *Children's Environments Quarterly, 5*(4), 14–18.

Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers and Education, 50*, 559–578. doi: 10.1016/j.compedu.2007.09.020

Rodger, S. H., Brown, D., Hoyle, M., MacDonald, D., Marion, M., Onstwedder, E., . . . Ward, E. (2014, June). Weaving computing into all middle school disciplines. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 207–212), Uppsala, Sweden. doi: 10.1145/2591708.2591754

Rodger, S. H., Hayes, J., Lezin, G., Qin, H., Nelson, D., Tucker, R., . . . Slater, D. (2009). Engaging middle school teachers and students with Alice in a diverse set of subjects. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (pp. 271–275), Chattanooga, TN.

Rose, D., & Meyer, A. (2000). Universal design for learning. *Journal of Special Education Technology, 15*(1), 67–70.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics in Education*, *10*(1), 73–88.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2012). Programming: Teachers and pedagogical content knowledge in the Netherlands. *Informatics in Education*, *11*(1), 81–114.

Schacter, J., Herl, H. E., Chung, G. K. W. K., Dennis, R. A., & O'Neil, Jr., H. F. (1999). Computer-based performance assessments: A solution to the narrow measurement and reporting of problem solving. *Computers in Human Behavior, 15*, 403–418.

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015, February). Transferring skills at solving word problems from computing to algebra through Bootstrap. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 616–621).

Schanzer, E. T. (2015). *Algebraic functions, computer programming, and the challenge of transfer* (Doctoral dissertation).

Schofield, E., Erlinger, M., & Dodds, Z. (2014, March). MyCS: CS for middle-years students and their teachers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 337–342), Atlanta, GA. doi: 10.1145/2538862.2538901

Schulte, C., & Knobelsdorf, M. (2007). Attitudes towards computer science—computing experiences as a starting point and barrier to computer science. In *Proceedings of the Third International Workshop on Computing Education Research* (pp. 27–38). doi: 10.1145/1288580.1288585

Schulz, S., & Pinkwart, N. (2015, November). Physical computing in STEM education. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 134–135).

Seiter, L., & Foreman, B. (2013, August). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (pp. 59–66).

Sengupta, P., Kinnebrew, J. S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K–12 science education: A theoretical framework. *Education and Information Technologies, 18*, 351–380.

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012, July). Infusing computational thinking into the middle- and high-school curriculum. In *Proceedings of the 17th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 22–27), Haifa, Israel.

Sherin, B. L. (2001). A comparison of programming languages and algebraic notation as expressive languages for physics. *International Journal of Computers for Mathematical Learning, 6*(1), 1–61. doi: 10.1023/A:1011434026437

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher 15*(2), 4–14. doi: 10.3102/0013189X015002004

Simon, Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., . . . Tutty, J. (2006, January). Predictors of success in a first programming course. In *Proceedings of the Eighth Australasian Computing Education Conference* (pp. 189–196), Hobart, Tasmania, Australia.

Smith, J. L., Lewis, K. L., Hawthorne, L., & Hodges, S. D. (2013). When trying hard isn't natural: Women's belonging with and motivation for male-dominated STEM fields as a function of effort expenditure concerns. *Personality and Social Psychology Bulletin*, *39*(2), 131–143.

Snyder, L. (2010). *Six computational thinking practices.* Retrieved from https://csprinciples.cs.washington.edu/sixpractices.html

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM, 29*(9), 850–858.

Sprague, P., & Schahczenski, C. (2002). Abstraction the key to CS1. *Journal of Computing Sciences in Colleges, 17*(3), 211–218.

Sudol, L. A., Stehlik, M., & Carver, S. (2009). *Mental models of data: A pilot study.* Paper presented at Ninth Baltic Sea Conference on Computing Education Research (Koli Calling 2009), Koli National Park, Finland.

Sullivan, G. (2014, May 29). Google statistics show Silicon Valley has a diversity problem. *The Washington Post.* Retrieved from https://www.washingtonpost.com/news/morning-mix/wp/2014/05/29/most-google-employees-are-white-men-where-are-allthewomen/

Syslo, M. M. (2015). From algorithmic to computational thinking: On the way for computing for all students. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education.* Vilnius, Lithuania. doi: 10.1145/2729094.2742582

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education, 12*(2), Article 8.

Texas Education Agency. (2011). *Technology applications TEKS.* Retrieved from http://tea.texas.gov/Curriculum_and_Instructional_Programs/Curriculum_Standards/TEKS_Texas_Essential_Knowledge_and_Skills_(TEKS)_Review/Technology_Applications_TEKS/

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

Twarek, B. (2015). *Pre-K to 12 computer science scope and sequence*. Retrieved from http://www.csinsf.org/curriculum.html

Valkenburg, P. M., & Peter, J. (2013). The differential susceptibility to media effects model. *Journal of Communication*, *63*, 221–243. doi: 10.1111/jcom.12024

Vekiri, I., & Chronaki, A. (2008). Gender issues in technology use: Perceived social support, computer self-efficacy and value beliefs, and computer use beyond school. *Computers & Education, 51*, 1392–1404.

Watson, W. E., Kumar, K., & Michaelsen, L. K. (1993). Cultural diversity's impact on interaction process and performance: Comparing homogeneous and diverse task groups. *Academy of Management Journal, 36*(3), 590–602.

Wehmeyer, M. L. (2015). Framing the future self-determination. *Remedial and Special Education*, *36*(1), 20–23.

Wehmeyer, M. L., Shogren, K. A., Palmer, S. B., Williams-Diehm, K. L., Little, T. D., & Boulton, A. (2012). The impact of the self-determined learning model of instruction on student self-determination. *Exceptional Children, 78*(2), 135–153.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Weintrop, D., & Wilensky, U. (2015). Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 101–110), Omaha, NE.

Werner, L., Campe, S., & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. In *Proceedings of the Special Interest Group in Computer Science Education* (pp. 215–220), Raleigh, NC.

Werner, L., & Jenner, D. (2009). Pair programming in middle school: What does it look like? *Journal of Research on Technology in Education, 42*(1), 29–49.

Werner, L., Denner, J., & Campe, S. (2012, February–March). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 215–220). doi: 10.1145/2157136.2157200

Werner, L., Denner, J., & Campe, S. (2014). Using computer game programming to teach computational thinking skills. In K. Schrier (Ed.), *Learning, education, and games* (Vol. 1, pp. 37–53). Pittsburgh, PA: ETC Press.

Werner, L., Denner, J., Campe, S., Ortiz, E., DeLay, D., Hartl, A. C., & Laursen, B. (2013). Pair programming for middle school students: Does friendship influence academic outcomes? In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 421–426). doi: 10.1145/2445196.2445322

Whitley, Jr., B. E. (1997). Gender differences in computer-related attitudes and behavior: A meta-analysis. *Computers in Human Behavior, 13*(1), 1–22.

Wille, S. J., & Kim, D. (2015). Factors affecting high school student engagement in introductory computer science classes. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 675–675), New York, NY. doi: 10.1145/2676723.2691891

Wille, S. J., Pike, M., & Century, J. (2015). Bringing AP Computer Science Principles to students with learning disabilities and/or an ADHD: The hidden underrepresented group. Abstract. http://www.nsf.gov/awardsearch/showAward?AWD_ID=1542963.

Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM, 43*(5), 108–114.

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software, 17*(4), 19–25.

Wing, J. M. (2006, March). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366*(1881), 3717–3725.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), Article 5, 1–16.

Yardi, S., & Bruckman, A. (2007). What is computing?: Bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the Third International Workshop on Computing Education Research* (pp. 39–50). doi: 10.1145/1288580.1288586

Zare-ee, A., & Shekarey, A. (2010). The effects of social, familial, and personal factors on students' course selection in Iranian technical schools. *Procedia: Social and Behavioral Sciences, 9*, 295–298.

Apéndice F: Preguntas Frecuentes

P: ¿Cuál es el marco??
R: El marco es una línea base, un conjunto esencial de conceptos y prácticas de las ciencias de la computación. El enfoque del marco es iluminar ideas poderosas en ciencias de la computación K-12. Cada uno de los conceptos core están delineado con expectativas en cuatro puntos finales diferentes de la banda de grado: Grados 2, 5, 8 y 12. Las prácticas no están delineadas por bandas de grado explícitas, sino que proporcionan una narrativa que describe la progresión de cada práctica desde el jardín de niños hasta el grado 12.

P: ¿Por qué necesitamos un marco y por qué es importante?
R: La computación está a nuestro alrededor en el mundo moderno, sin embargo, muchos estudiantes no entienden cómo funcionan estas tecnologías. El interés en ciencias de la computación siga aumentando, y la educación K–12 está ansiosa por satisfacer la demanda. Sin embargo, las ciencias de la computación son bastante nuevas en la educación K–12, y los estados, distritos, escuelas y profesores necesitan orientación para un camino apropiado de K–12 en ciencias de la computación.

P: ¿Cuál es la visión del marco?
R: El propósito es crear un marco de alto nivel de conceptos y prácticas de las ciencias de la computación que capacite a los estudiantes para
• ser ciudadanos informados que puedan participar críticamente en debates públicos sobre temas relacionados con las ciencias de la computación;
• desarrollarse como aprendices, usuarios y creadores de conocimientos y artefactos computacionales;
• comprender mejor el papel de las ciencias de la computación en el mundo que los rodea; y
• aprender, actuar y expresarse en otras materias e intereses.

P: ¿Qué son las ciencias de la computación?
R: Las ciencias de la computación son estudios de las computadoras y los procesos algorítmicos, incluidos sus principios, diseño, implementación e impacto en la sociedad. (Tucker, 2006, p. 2).

P: ¿Para quién es el marco?
R: El Marco de Ciencias de la Computación K–12 es un paso inicial en un proceso que informará las decisiones a nivel estatal y distrital para introducir y mejorar la educación en ciencias de la computación. El marco fue escrito para una variedad de audiencias con una amplia gama de antecedentes en ciencias de la computación. Los públicos principales del marco son los encargados de la formulación de políticas y los administradores estatales, los encargados de la formulación de políticas y los administradores del distrito, los desarrolladores de normas, los desarrolladores de currículos, los proveedores de desarrollo profesional, los actuales y nuevos investigadores y educadores computacionales en entornos formales e informales.

P: ¿Por qué son importantes las ciencias de la computación?
R: Las ciencias de la computación respaldan muchos aspectos del mundo moderno. La omnipresencia de la computación personal en nuestras vidas y nuestra dependencia exponencialmente creciente de todas las cosas relacionadas con la tecnología han cambiado la estructura de la sociedad y la vida cotidiana. Desafortunadamente, los estudiantes de K–12 hoy tienen una oportunidad limitada de aprender sobre estas ideas y prácticas de las ciencias de la computación y analizar cómo las ciencias de la computación influyen en su vida diaria.

P: ¿Cuáles son los conceptos y prácticas core del marco?
R: Los conceptos Core son categorías que representan las principales áreas de contenido en el campo de las ciencias de la computación. Representan áreas específicas de importancia disciplinaria en lugar de ideas abstractas y generales. Las prácticas centrales son los comportamientos que los estudiantes alfabetizados en ciencias de la computación usan para involucrarse plenamente con los conceptos core de las ciencias de la computación.

Conceptos core:

1. Sistema de computación.
2. Redes y Internet.
3. Datos y análisis.
4. Algoritmos y Programación.
5. Impactos de la Computación.

Prácticas core:

1. Fomentar una cultura de las ciencias de la computación inclusiva.
2. Colaborar en torno a las ciencias de la computación.
3. Reconocer y definir problemas computacionales.
4. Desarrollar y usar abstracciones.
5. Crear artefactos computacionales.
6. Probar y refinar los artefactos computacionales
7. Comunicar acerca de la computación.

P: ¿Quién creó el marco?
R: El comité directivo para el marco está compuesto por representantes de las siguientes organizaciones: Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center y National Math + Science Initiative. El equipo de redacción estaba compuesto por representantes de los estados participantes, distritos escolares, educadores de educación primaria y secundaria, docentes de educación superior y organizaciones de investigación y sin fines de lucro. Puede ver sus biografías en el Apéndice B: Biografías de escritores y personal de desarrollo). Los principales investigadores y representantes de organizaciones en apoyo de la educación en ciencias de la computación se desempeñaron como asesores de los escritores y el personal de desarrollo. Más de 100 profesionales de la educación en ciencias de la computación y organizaciones de partes interesadas (Stakeholders) sirvieron como revisores, con más de 530 revisiones enviadas.

P: ¿Participaron los profesores?
R: Muchos de los escritores eran profesores actuales o habían sido profesores en el pasado. Su experiencia abarcó desde el jardín de infantes hasta el grado 12 e incluyó una variedad de materias fuera de las ciencias de la computación. Algunos de los asesores eran profesores, muchos miembros del personal de desarrollo eran antiguos profesores y los profesores fueron incluidos en los períodos de revisión y en los grupos focales durante el proceso de desarrollo.

P: ¿Cómo participaron los estados en el desarrollo del marco?
R: Para cada uno de los estados involucrados, representantes del departamento de educación del estado y la junta de educación asistieron a las reuniones de las partes interesadas (Stakeholders) para proporcionar comentarios sobre el desarrollo del marco. A los diez estados que participaron en el lanzamiento del desarrollo del marco se les pidió que nominaran a alguien de su estado para formar parte del equipo de redacción y que convocaran a un grupo para revisar los borradores del marco en su estado.

P: ¿Cómo se involucró el público en la creación del marco?
R: Se llevaron a cabo tres períodos de revisión pública durante el desarrollo del marco en 2016. Cada uno duró de dos a tres semanas, y cada uno fue publicitado ampliamente para alentar al público a leer los borradores de las versiones y enviar comentarios a través de un formulario en línea. La respuesta recibida fue leída por los escritores y el equipo de desarrollo, y los escritores abordaron temas comunes. Puede leer más en el capítulo **Proceso de desarrollo** y ver un resumen de las revisiones públicas en el **Apéndice A: Comentarios y revisiones**.

P: ¿Cómo se utilizó la investigación para informar el desarrollo del marco?
R: El equipo de redacción consideró la literatura actual sobre educación en informática desde el comienzo del proceso de redacción. Una vez completado el segundo borrador del marco, se completó una revisión sistemática de la literatura relacionada con los conceptos y prácticas. A partir de esta revisión, se ajustaron las declaraciones de concepto y práctica para alinearlas con la investigación actual en el campo. Puede leer más en **El papel de la investigación en el desarrollo y el futuro del marco**.

P: ¿Cuál es la relación entre el marco y los estándares?
R: El marco delinea ampliamente los conceptos que los estudiantes deben conocer y las prácticas que los estudiantes deben exhibir, pero no proporciona el nivel de detalle de los estándares de grado por grado, objetivos o descripciones de los cursos o planes de lecciones. En cambio, sirve como una guía integral para el desarrollo de estándares, currículo, evaluaciones, educación docente y programas extracurriculares. El marco no es un conjunto de estándares. Los estados pueden usar el marco para desarrollar estándares que combinen los conceptos y prácticas en expectativas de desempeño que sean claras, específicas y mensurables.

P: ¿Cómo es el marco distinto de los estándares nacionales?
R: Las declaraciones del marco **no** son estándares. A propósito, no son tan prescriptivos o mensurables como los estándares de desempeño. No abordan la granularidad individual de nivel de grado; en su lugar, abordan los bandas de grados y describen cómo progresa el aprendizaje de una banda a la otra. Hay muchas menos declaraciones en el marco que en un documento de estándares. El objetivo del marco es proporcionar un conjunto mínimo de conceptos y prácticas que describan la alfabetización básica en ciencias de la computación que todos los estudiantes deberían tener. Un objetivo explícito del marco es mostrar significación/aplicación más allá de las ciencias de la computación y la significación para cada ciudadano, no solo para los estudiantes de las ciencias de la computación. El marco está diseñado intencionalmente para la personalización y estará disponible gratuitamente. Describe lo que los estudiantes deben aprender usando una prosa no técnica que es fácil de entender por un público amplio. Los estados y los distritos deben tomar la decisión final sobre los documentos que usan cuando desarrollan sus propios estándares de las ciencias de la computación. El marco distingue entre conceptos y prácticas. Un documento de estándares debería integrar estas dos dimensiones en cada estándar. Puede leer más sobre cómo el marco puede informar el desarrollo de estándares en el capítulo **Guía para desarrolladores de estándares**.

P: ¿Cuál es la relación con los estándares K–12 de la Asociación de Profesores de Ciencias de la Computación (CSTA)?
R: El Marco de Ciencias de la Computación K–12 sirvió como una de las mayores aportaciones a la revisión provisional de 2016 de los estándares de las ciencias de la computación de CSTA K – 12 para asegurar la alineación y permitir que la comunidad de educación en ciencias de la computación hable con una voz coherente sobre lo que K– 12 estudiantes deben saber y poder hacer. Los copresidentes del grupo de trabajo de revisión de estándares de CSTA sirvieron como asesores del marco, y la mitad de los redactores de estándares de CSTA (incluidos todos los redactores principales de CSTA) sirvieron como redactores del marco.

P: ¿Se planean futuras revisiones del marco?
R: Se anticipa que el marco será revisado en el futuro.

P: ¿Cómo aborda el marco a la alfabetización informática y la ciudadanía digital?
R: El marco define las ciencias de la computación K-12, que son diferentes de la ciudadanía digital y la alfabetización informática. La ciudadanía digital se define como las normas de comportamiento apropiado y responsable con respecto al uso de la tecnología (Departamento de Educación Primaria y Secundaria de Massachusetts, 2016). La alfabetización informática se centra en el uso de tecnologías y programas informáticos existentes, como el procesamiento de textos y las hojas de cálculo (Centro Nacional para Mujeres y Tecnología de la Información).

Las ciencias de la computación, por otro lado, se trata de "la capacidad de crear y adaptar nuevas tecnologías" e implica analizar cómo funcionan las computadoras y cómo nos afectan (Centro Nacional de Mujeres y Tecnología de la Información, sin fecha, párr. 3). El Marco de Ciencias de la Computación K–12 está destinado a complementar la instrucción de alfabetización computacional y ciudadanía digital. La instrucción en las tres áreas es importante para todos los estudiantes.

P: ¿Cómo aborda el marco al pensamiento computacional?
R: El pensamiento computacional se refiere a los procesos de pensamiento involucrados en la expresión de soluciones como pasos computacionales o algoritmos que puede llevar a cabo una computadora (Cuny, Snyder y Wing, 2010; Aho, 2011; Lee, 2016). Está delineado en cuatro de las siete prácticas de las ciencias de la computación en el marco. Las ciencias de la computación ofrecen una oportunidad única para que los estudiantes desarrollen el pensamiento computacional, pero la oportunidad de aplicar el pensamiento computacional se extiende más allá del contexto de las ciencias de la computación. Las revisiones recientes a los Estándares de la Sociedad Internacional de Tecnología en Educación para Estudiantes 2016 están alineados con esta definición de pensamiento computacional. Estos documentos respaldan la visión compartida de que el pensamiento computacional es importante para todos los estudiantes en todas las clases.

P: ¿Cómo se implementa el marco en las escuelas?
R: El marco se puede utilizar de varias maneras. Puede informar el desarrollo del currículo, el desarrollo de estándares, las rutas K-12, la preparación de profesores y el desarrollo profesional, y la evaluación en el aula, entre otros. Puede implementarse como cursos independientes o integrarse en otras áreas temáticas, particularmente en los niveles de primaria y secundaria. Puede leer más en el capítulo **Guía de implementación**.

P: ¿Dónde puedo encontrar el marco?
R: Puede descargar una copia del marco completo (todos los capítulos, incluido el material de orientación) como un archivo PDF en k12cs.org. ¿Solo quieres los conceptos y prácticas? Haga clic en "Declaraciones de marco" en el menú principal para acceder al menú desplegable y seleccione una de las tres vistas. Desde cada página, puede descargar un archivo PDF de los conceptos y prácticas.

P: Si necesito ayuda para usar el marco, ¿a quién debo contactar?
R: Puede comunicarse con el personal de desarrollo con sus preguntas utilizando el formulario que se encuentra en k12cs.org.

Referencias

Aho, A. V. (2011, January). Computation and computational thinking. *ACM Ubiquity, 1*, 1–8. .

Cuny, J., Snyder, L., & Wing, J. M. (2010). *Computational thinking: A definition.* Unpublished manuscript.

Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12*(1), 3–4. Retrieved from http://www.csteachers.org/resource/resmgr/Voice/csta_voice_03_2016.pdf

Massachusetts Department of Elementary and Secondary Education. (2016, June). *2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework.* Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

National Center for Women & Information Technology. (n.d.). *Moving beyond computer literacy: Why schools should teach computer science.* Retrieved from https://www.ncwit.org/resources/moving-beyond-computer-literacy-why-schools-should-teach-computer-science/moving-beyond

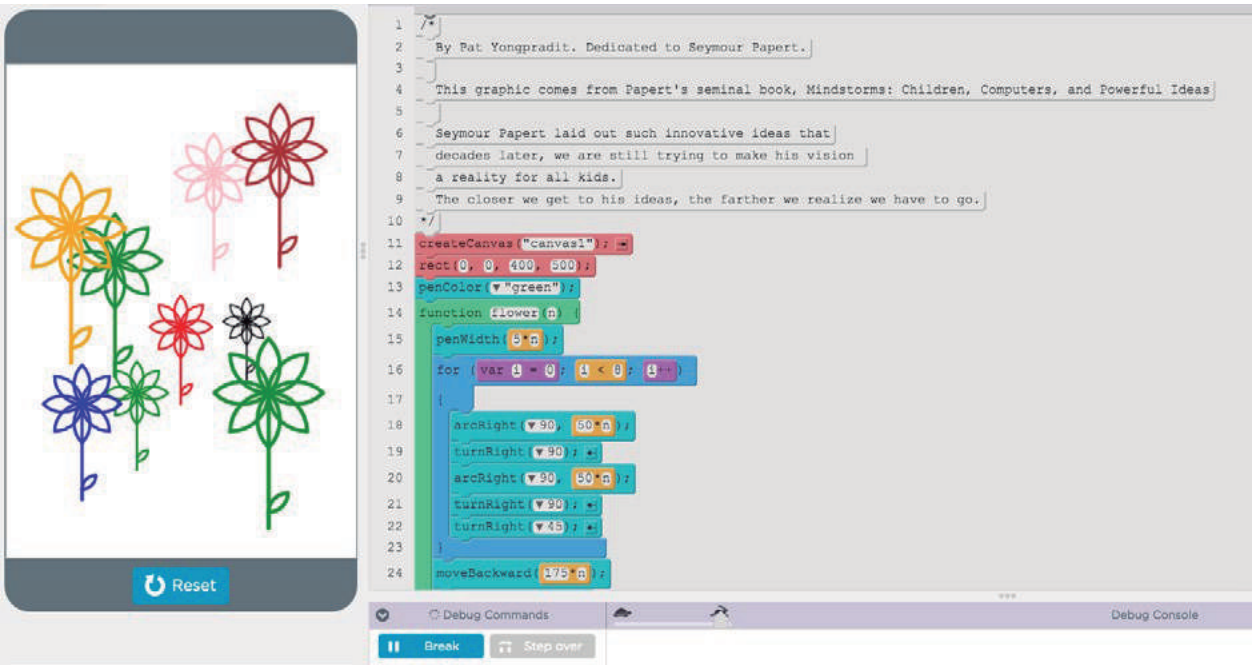
Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

Créditos Fotográficos

Gracias a los estudiantes y profesores de los distritos escolares que invitaron a los fotógrafos a sus aulas y permitieron que sus fotografías fueran incluidas en este documento..
Lincoln High School, Tacoma, Washington: portada, páginas 4, 21, 23, 39, 41, 42, 46, 54, 85, 87, 143, 151, 168, 176, 202, and 219.
John Muir Elementary School, Seattle, Washington: páginas 12, 24, 31, 55, 57, 65, 67, 69, 70, 134, 181, 183, 199, 201, 208, 211, and 215.
Mount View Elementary School, White Center, Washington: páginas 7, 9, 16, 53, 123, 125, 134, 136, 145, 147, 205, and 229.
También agradecemos a las organizaciones que compartieron gentilmente fotografías de estudiantes que participan en ciencias de la computación, así como a los estudiantes, padres y profesores en las imágenes.
DevTech Research Group: páginas 194, 196, 272, and 273.
AccessCS10K from the University of Washington: páginas 33.
WCTE Cookeville, TN: páginas 187 y 270.
WHRO Norfolk, VA: páginas 187 y 270.

En la portada

El programa que se muestra en el monitor se inspiró en una anécdota en un libro que ha influido en muchos educadores de las ciencias de la computación, Seymour Papert's Mindstorms (1980). En esta anécdota, Papert describe una situación hipotética en la que los estudiantes escriben un programa para dibujar un jardín de flores. La anécdota ilustra cómo jugar con algunas ideas poderosas puede conducir a resultados hermosos. Las poderosas ideas de Papert inspiraron a una generación de educadores que continúan trabajando para hacer realidad a su visión. Estas flores han crecido en la tierra fértil que preparó Papert. Seymour Papert falleció el 31 de julio de 2016. Cuanto más nos acercamos a sus ideas, más nos damos cuenta de que tenemos que ir.



Comentarios de la traducción al español del Marco de las Ciencias de la Computación K-12

Información sobre algunos términos únicos del marco de las ciencias de la computación.

Core: La palabra corazón se deriva de una raíz del idioma Latino Cor(e) por eso en términos de este documento la palabra "core" debe ser considerada un término transversal entrelazado dentro este documento y significa igual a estos otros términos similares, tales como conceptos clave, central, básico, fundamentales etc. En este documento, la palabra "core" se ocupa mayormente para representar los significados de las otras palabras indicadas.
La palabra "core" puede también conectar las ciencias de la computación a un mundo más amplio en términos de una relación común entre otras naciones y nacionalidades.

Banda de grado: es un concepto entrelazado en toda parte del documento. De hecho, es el "core" o corazón de los conceptos y prácticas de la metodología del marco. Por lo tanto, a continuación una explicación breve.
Bandas de grado son tramos compuestos de distintos niveles de enseñanza y se refieren a la medición del desarrollo de habilidades de los alumnos dentro cada nivel desde kínder hasta el final de la etapa escolar.

Depende del contexto, la palabra inglés "grade" -> grado en español puede significar varias cosas, pero en el caso de educación la palabra se presenta dos significados distintos, 1) puede significar calificación; y 2) puede significar nivel de enseñanza. En este documento "grado" significa solamente nivel de enseñanza y no tiene nada que ver con el concepto de calificaciones respecto de este documento. Los niveles referenciados por bandas de grado dentro el marco CCK-12, se pueden comprender así:

- K-2 significa un tramo de tres niveles distintos compuestos de las edades 5 a 7, o sea, grados (niveles) K, 1, 2.
- 3-5 significa un tramo de tres niveles distintos compuestos de las edades 8 a 10, o sea grados (niveles) 3, 4, 5.
- 6-8 significa un tramo de tres niveles distintos compuestos de las edades 11 a 14 o sea grados (niveles) 6, 7, 8.
- 9-12 significa un tramo de cuatro niveles distintos compuestos de las edades 15 a 18 o sea grados (niveles) 9, 10, 11, 12.

De esta manera, podemos entrelazar el concepto de las bandas de grados para lograr una congruencia entre los distintos sistemas educativos, chileno/estadounidense, para que no se afecte el significado de los "conceptos core." Intentamos con esta pequeña ayuda iluminar a los educadores chilenos, para que puedan comprender la idea y adaptarse sin dificultades.

El rigor diferenciador es un término para medir rigurosamente los avances del desarrollo de las habilidades de los alumnos de las ciencias de la computación dentro y entre las distintas bandas de grados (tramos de niveles de la enseñanza) y está vinculado íntimamente con los conceptos core y prácticas core.

Conceptos Core:

1. Sistemas de computación.
2. Las redes e internet.
3. Datos y análisis.
4. Algoritmos y Programación.
5. Impactos de la computación.

Prácticas Core

1. Fomentar una cultura computacional inclusiva.
2. Colaboración en torno a la computación.
3. Reconociendo y definiendo problemas computacionales.
4. Desarrollar y usar abstracciones.
5. Creando artefactos computacionales.
6. Probar y refinar artefactos computacionales.
7. Comunicar sobre de la computación.

STEM: (Science, Technology, Engineering and Mathematics). El Marco de Ciencias de la Computación K–12 fue informado por un creciente cuerpo de investigación en educación en ciencias de la computación, así como también por una literatura más amplia de los campos de la educación en ciencia, tecnología, ingeniería y matemáticas (STEM). En particular, los conceptos, prácticas y progresiones de aprendizaje en el corazón del marco fueron influenciados por la investigación sobre temas tales como la forma en que los estudiantes aprenden ciencias de la computación, cómo interactúan entre ellos en entornos computacionales y a qué edad demuestran dominio en un tema de conceptos específicos.

PCK: El Conocimiento del Contenido Pedagógico (CCP) es el conocimiento que los profesores tienen sobre la enseñanza como práctica combinada con su experiencia en la materia (Shulman, 1986). Además, la importancia de los factores contextuales, incluidas las actitudes de los docentes, la autoeficacia y los juicios de valor, así como la especificidad de dominio impregnan las conceptualizaciones más recientes de CCP.

Creative Commons: Otro punto a cerca de este documento. Dado que fue elaborado por la entidad "Creative Commons" y es un documento de fuente abierta. Se debe considerar el marco CCK-12 como un documento vivo, el cual los usuarios finales puedan adaptar, alterar, extender y compartir, según sus experiencias y descubrimientos del marco para continuar elaborando la metodología del marco CCK-12.