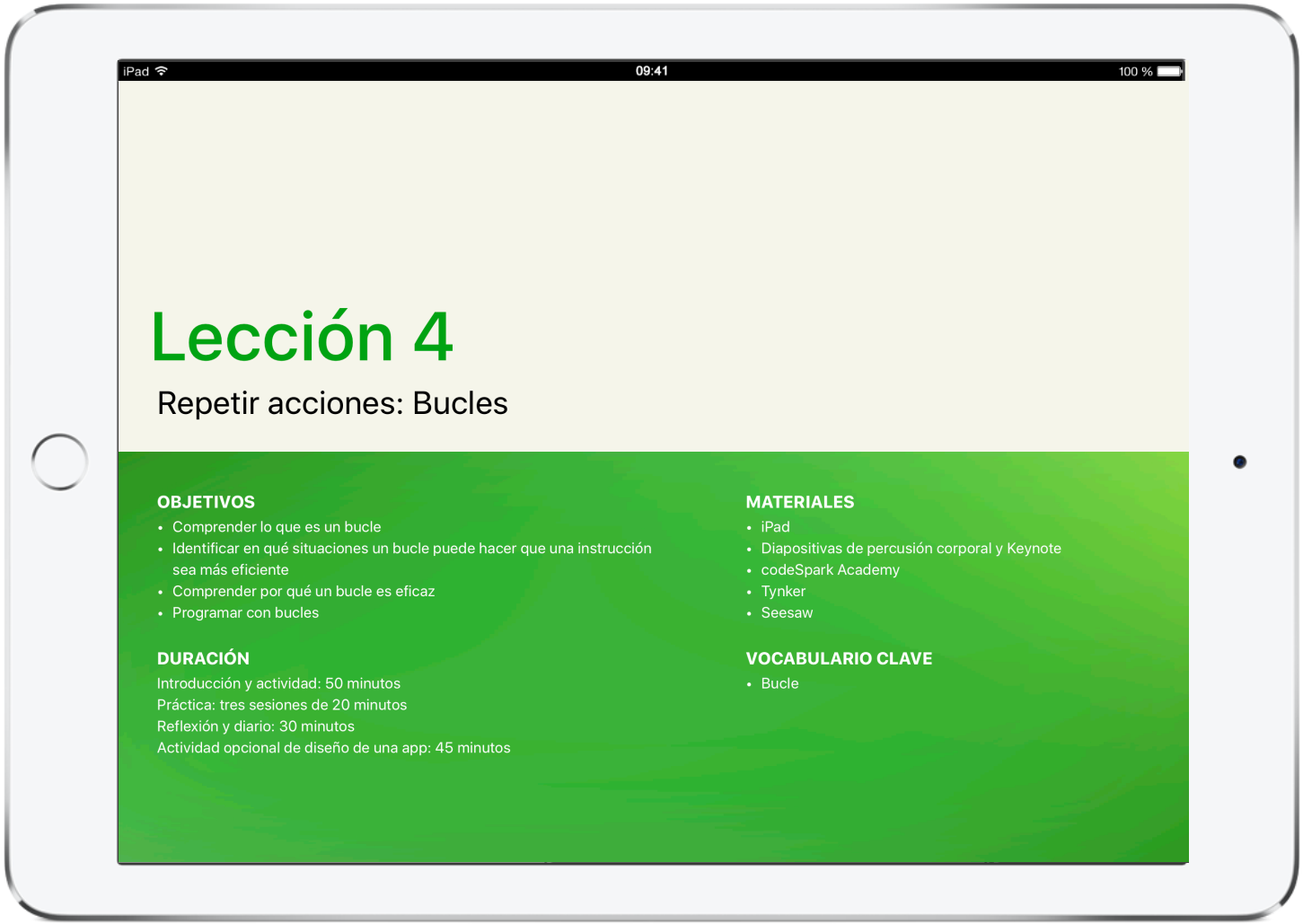
Guía del plan de estudios de Empezar a programar

Septiembre de 2017

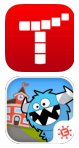
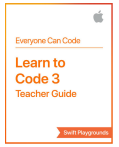


Programación para todos

La tecnología tiene un lenguaje, que se llama código. El código se utiliza para programar, y nosotros creemos que la programación es una habilidad esencial. Aprender a programar te enseña a resolver problemas y trabajar en conjunto de manera creativa. Además, te permite crear apps que hacen realidad tus ideas. Creemos que todas las personas deberían tener la oportunidad de crear algo que pueda cambiar el mundo. Por eso, hemos diseñado un nuevo programa con recursos y herramientas que permiten a cualquier persona aprenderlo, escribirlo y enseñarlo.

Plan de estudios de Programación para todos

Programación para todos incluye una variedad de recursos para que los estudiantes sin experiencia en programación aprendan a crear sus primeras apps. En la tabla que aparece a continuación, se proporciona una descripción general de todos los recursos gratuitos de enseñanza y aprendizaje disponibles.

Plan de estudios	Dispositivo	Audiencia	App	Requisitos previos	Descripción general	Materiales de aprendizaje	Recursos de apoyo	Número de horas de lección incluidas
		Desde la educación inicial hasta el segundo grado		Ninguno	Comienza a pensar como un programador explorando, en la práctica, conceptos de programación mediante apps de base visual.	• Lecciones de la app codeSpark Academy • Curso de Cadete Espacial de Tynker	• Empezar a programar 1: Guía para profesores	30 horas, en las que se incluyen las lecciones de las apps y la Guía para profesores
		Desde el tercer grado hasta el quinto grado		Ninguno	Explora los conceptos fundamentales de programación y practica cómo es pensar como un programador mediante apps de base visual.	• Curso Hechizos de dragón de Tynker	• Empezar a programar 2: Guía para profesores	36 horas, en las que se incluyen las lecciones de las apps y la Guía para profesores
		Escuela secundaria en adelante		Ninguno	Aprende conceptos fundamentales de programación utilizando código Swift real.	• App Swift Playgrounds • Lecciones de Aprender a programar 1 y 2 • Curso de iTunes U	• Aprender a programar 1 y 2: Guía para profesores • Distintivos de Swift Playgrounds de Apple Teacher Learning Center	Hasta 85 horas, en las que se incluyen las lecciones de Aprender a programar 1 y 2, y la Guía para profesores
		Escuela secundaria en adelante		Aprender a programar 1 y 2	Amplía tus habilidades de programación y comienza a pensar como un desarrollador profesional de apps.	• App Swift Playgrounds • Lecciones de Aprender a programar 3	• Aprender a programar 3: Guía para profesores	Hasta 45 horas, en las que se incluyen las lecciones de Aprender a programar 3 y la Guía para profesores
		Escuela media y superior en adelante		Ninguno	Adquiere experiencia práctica con las herramientas, las técnicas y los conceptos que se necesitan para crear una app de iOS básica desde cero.	Libro y archivos de proyectos de Introducción al desarrollo de apps con Swift	• Introducción al desarrollo de apps con Swift: Guía para profesores	90 horas
		Escuela media y superior en adelante		Ninguno	Construye una base sólida en el uso de Swift, de UIKit y de redes mediante actividades prácticas y proyectos guiados. Los estudiantes podrán crear una app con diseño propio al final de este curso.	Libro y archivos de proyectos de Desarrollo de apps con Swift	• Desarrollo de apps con Swift: Guía para profesores	180 horas

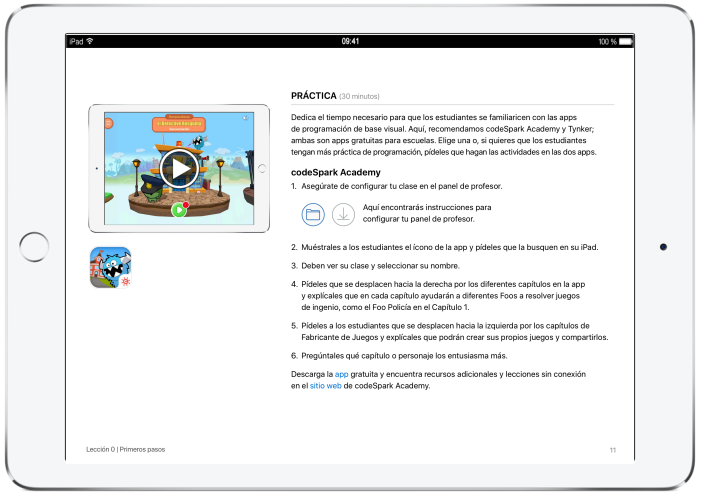
Descripción general

Los primeros años de educación son un excelente momento para presentar conceptos de programación como una manera de pensar en el mundo cotidiano y el digital, y para desarrollar las habilidades fundamentales de pensamiento computacional. Las apps que están especialmente diseñadas para los estudiantes más jóvenes, como codeSpark Academy y Tynker, presentan una base visual de programación con el objetivo de desarrollar habilidades de resolución de problemas, fomentar la perseverancia y promover la creatividad. La app codeSpark Academy está diseñada para estudiantes de entre cinco y siete años. La interfaz sin texto del juego permite que puedan jugar todos los estudiantes, incluidos los que todavía no saben leer, los que están aprendiendo español y los que tienen dificultades de lectura. En Tynker, los estudiantes de entre cinco y once años comienzan experimentando con bloques visuales y, luego, avanzan a la programación basada en texto mientras resuelven juegos de ingenio y crean proyectos.

En el aula

Tynker y codeSpark Academy, junto con las lecciones de la Guía para profesores de Empezar a programar, están diseñadas para ayudarte a introducir la programación en los primeros años de la escuela primaria. En las lecciones, se destacan conceptos clave de programación a medida que se demuestra que la programación es una forma de pensar que puede aplicarse a otras áreas del aprendizaje y a la vida cotidiana.

La Guía para profesores proporciona el apoyo que necesitas para ayudar a tus estudiantes a resolver los juegos de ingenio de programación, sin importar el nivel de experiencia que tengas en esta área. Se incluyen actividades de extensión, actividades de diseño de apps, preguntas para reflexionar, ideas para escribir entradas de diario, una planilla de evaluación y otros recursos para ayudarte a profundizar la comprensión del material por parte de los estudiantes. Puedes enseñar las lecciones en un solo bloque o dividirlas en secciones. En los Apéndices, se incluyen mapas de equivalencias en los que se muestra una correlación preliminar de las lecciones con los estándares provisionales para las ciencias de la computación K-12 de nivel 1 de la Asociación de Profesores de Ciencias de la Computación.



Características principales

codeSpark Academy

Aprender. Los estudiantes resuelven juegos de ingenio en este juego sin texto para aprender conceptos básicos de ciencias de la computación, como secuencias, bucles, instrucciones condicionales y mucho más.

Crear. Luego, los estudiantes aplican el conocimiento adquirido programando sus propios proyectos en Fabricante de Juegos.

Aprendizaje autodirigido. No se necesita tener experiencia en programación para enseñar, aprender o jugar. El plan de estudios y el panel de profesor para informes de progreso están disponibles para los profesores en diez idiomas distintos.

Tynker

Entorno de programación. Los estudiantes avanzan a su propio ritmo a través de juegos de ingenio de programación organizados en niveles, para aprender conceptos y aplicarlos de forma creativa.

Evaluación automática. En el panel de profesor, se evalúa el dominio que tienen los estudiantes de las habilidades con juegos de ingenio, cuestionarios y análisis de código.

Función de Swift. A medida que los estudiantes resuelven los juegos de ingenio, pueden pasar de los bloques visuales a los bloques de Swift, lo que les permite familiarizarse con Swift y prepararse para programar en un futuro.

Guía para profesores de Empezar a programar

Archivos descargables. Los archivos de plantillas para las actividades de los estudiantes y las presentaciones de Keynote sirven de apoyo para el aprendizaje de los estudiantes durante la clase.

Respuestas. Las soluciones de los juegos de ingenio de Tynker y codeSpark Academy te facilitan la tarea de ayudar a los estudiantes que no puedan avanzar.

Ejemplos de trabajos de estudiantes. Mira cómo pueden quedar las actividades.

Reflexiones. Estas preguntas e indicaciones para debatir en clase te ayudan a revisar y reforzar la conexión que hay entre la aplicación de los conceptos dentro y fuera del entorno de programación.

Consejos y ejemplos. Se incluyen ideas para ampliar o simplificar todas las lecciones.

Actividades de diseño de apps. Estas lecciones guían a los estudiantes a través de un proceso de diseño para conceptualizar y preparar el prototipo de una idea de app que resuelve un problema en el aula o la escuela.

Resumen del curso

Empezar a programar 1

Al participar en exploraciones interactivas y prácticas de conceptos de programación en el contexto de situaciones cotidianas, los estudiantes comienzan a pensar como programadores. También aprenden sobre comandos, secuencias, bucles, eventos y algoritmos. Al trabajar de manera colaborativa, los estudiantes practicarán cómo predecir el resultado de su código y cómo depurar su código y el de los demás. También pondrán en práctica sus habilidades en apps de programación de base visual, resolviendo juegos de ingenio y diseñando sus propias creaciones. Las actividades de diseño opcionales guían a los estudiantes a través de un proceso de diseño para pensar el concepto y preparar el prototipo de una idea de app que resuelve un problema en el aula o la escuela.

Lección 0 | Primeros pasos. Averigua qué saben los estudiantes sobre apps y programación, configura el muro de trabajo y muéstrales las apps clave que usarán en las lecciones. Los estudiantes aprenden sobre los diferentes roles que hay en un equipo de diseño de apps.

Lección 1 | Poner en orden: Introducción a la secuenciación. Los estudiantes exploran secuencias cotidianas, construyen una secuencia a partir de una historia conocida y resuelven problemas en apps de programación de base visual a través de secuencias simples. Los estudiantes exploran los diferentes propósitos que cumplen las apps.

Lección 2 | Identificar los pasos: Creación de secuencias. Al examinar la importancia del orden cuando se crea una secuencia de instrucciones, los estudiantes aprenden que las mismas acciones se pueden reordenar para crear una secuencia distinta y diseñar su propio baile de locos. Descubren más comandos y resuelven problemas más complejos en las apps de programación. Los estudiantes comparan diferentes apps que ayudan a los usuarios a aprender ideas nuevas.

Lección 3 | Elegir un orden: Secuenciación flexible. Los estudiantes aprenden que pueden ordenar de manera flexible algunos de los pasos de una secuencia y, de esa forma, crear sus propias secuencias flexibles. Exploran las distintas formas de resolver un juego de ingenio y comparten las soluciones de su código con otros estudiantes. Los estudiantes exploran apps que ayudan a los usuarios a interactuar con otras personas de su comunidad.

Lección 4 | Repetir acciones: Bucles. Los estudiantes identifican bucles en contextos cotidianos y usan la creatividad para crear sus propios bucles de percusión corporal. Observan cómo se representan los bucles en programación y los usan para optimizar y simplificar su código. Los estudiantes aprenden sobre el diseño de la interfaz de usuario y cómo puede lograr que una app resulte divertida y fácil de usar.

Lección 5 | Reparar código: Depuración. Una vez que comprenden la importancia de la perseverancia y la depuración en una variedad de contextos, los estudiantes revisan y aplican sus nuevas habilidades de programación para resolver un desafío y depurar las soluciones de otros estudiantes. Practican predecir el resultado de sus códigos e identificar los errores cuando el resultado que obtienen no es el esperado. Los estudiantes aprenden más en detalle sobre el diseño de apps para ayudar a resolver un problema.

Lección 6 | Generar sucesos: Eventos y Acciones. Los estudiantes exploran cómo los eventos pueden hacer que la jugabilidad y el código de su app sean más atractivos y tengan más capacidad de respuesta, y aprenden a programar con eventos. Consideran la forma en que generamos eventos en la vida real y crean un control remoto de un robot para practicar cómo provocar eventos. Los estudiantes comienzan a diseñar sus propias apps.

Lección 7 | Seguir reglas para lograr resultados: Instrucciones "if". Los estudiantes aprenden sobre las instrucciones condicionales y sobre cómo reconocer instrucciones "if" en la vida real. Piensan en las instrucciones "if" que funcionan como las reglas en los juegos de mesa que conocen. Los estudiantes programan con instrucciones "if", lo que hace que su código tenga más capacidad de respuesta a las condiciones del entorno. Los estudiantes usan diagramas de flujo para mostrar cómo funcionan sus apps.

Lección 8 | Resolver problemas: Algoritmos Con todo lo que han aprendido hasta el momento, los estudiantes aprenden a diseñar algoritmos que implican una serie de pasos para resolver un problema. Comienzan con una receta simple y avanzan al diseño y la programación de su propio juego de laberinto. Los estudiantes preparan prototipos de sus apps y completan un proyecto principal para documentar el proceso de diseño de su app.

Resumen del curso (continuación)

Empezar a programar 2

En Empezar a programar 2, los estudiantes explorarán conceptos fundamentales de programación y practicarán cómo es pensar como un programador. Además de aprender sobre algoritmos, funciones, bucles, instrucciones condicionales y variables, descubrirán los conceptos básicos del diseño de la interfaz de usuario. Los estudiantes trabajarán tanto en forma colaborativa como individual a medida que fortalecen sus habilidades de programación resolviendo problemas de programación reales, probando el código de sus compañeros y diseñando programas para una variedad de bots. Además, pondrán en práctica estas habilidades en Tynker al resolver una serie de problemas y aplicar los conceptos que aprendieron con las actividades realizadas en clase. Las actividades de diseño opcionales guían a los estudiantes a través de un proceso de diseño para pensar el concepto y preparar el prototipo de una idea de app que resuelve un problema en el aula o la escuela.

Lección 0 | Primeros pasos. Descubre qué saben los estudiantes sobre apps y programación, enséñales a usar una app de programación de base visual, como Tynker, y configura los diarios digitales de los estudiantes con una app, como Seesaw. Se presenta a los estudiantes un desafío de diseño de apps.

Lección 1 | Pensar en pasos: Resolver problemas mediante algoritmos. Los estudiantes aprenden que los algoritmos son una serie de instrucciones creadas para resolver un problema o realizar una tarea. En la app Tynker, los estudiantes eligen un dragón y, luego, crean algoritmos para resolver juegos de ingenio mientras desarrollan habilidades para formar secuencias. Los estudiantes diseñan y prueban algoritmos en actividades de la clase. Los estudiantes comienzan a proponer ideas de apps que pueden ayudar a resolver un problema.

Lección 2 | Pensar en soluciones: Depuración. Los estudiantes exploran cómo detectar y solucionar errores en sus algoritmos y su programación. En Tynker, modifican algoritmos que tienen errores con el objetivo de crear un programa apropiado para resolver los juegos de ingenio. Los estudiantes aprenden la función que cumplen los teclados en las apps y cómo pueden aplicarla a sus propias ideas de apps.

Lección 3 | Pensar en círculos: Búsqueda de bucles. Habiendo aprendido que los bucles son patrones repetitivos, los estudiantes diseñan un algoritmo para crear una serpiente con bucles. En Tynker, usan los bucles "for" para resolver juegos de ingenio detectando patrones. Los estudiantes proponen ideas sobre cómo su app puede aprovechar la cámara y el micrófono integrados.

Lección 4 | Pensar en partes: Composición y descomposición. A fin de detectar un algoritmo para realizar la canción del vaso, los estudiantes dividen la rutina en distintos componentes de movimiento. En Tynker, resuelven problemas dividiéndolos en subproblemas más pequeños. Los estudiantes piensan en cómo la pantalla táctil puede hacer que su app sea más interactiva.

Lección 5 | Pensar en conjuntos: Abstracción. Los estudiantes exploran las similitudes y la generalización a medida que categorizan objetos en conjuntos y, luego, explican su razonamiento. En Tynker, usan la abstracción para detectar similitudes entre problemas y resuelven juegos de ingenio, cuya complejidad va aumentando, con sus nuevas herramientas de programación. Los estudiantes piensan en cómo usar herramientas, como Bluetooth, para conectarse a dispositivos cercanos.

Lección 6 | Pensar en patrones: Formar funciones. Mientras crean una rutina de interpretación para un bot de comandos, los estudiantes la dividen en funciones y, luego, intercambian los algoritmos para poner a prueba los resultados reales y los esperados. En Tynker, los estudiantes usan un nombre y llaman a funciones a medida que reutilizan conjuntos de instrucciones para programar de forma más eficiente. Los estudiantes aprenden los tipos de datos a los que podría conectarse su app a través de GPS.

Lección 7 | Pensar en detalles: Instrucciones condicionales. Los estudiantes emprenden una aventura de viaje virtual, en la que definen su destino con un conjunto de condiciones que este debe cumplir mediante instrucciones "if". En Tynker, usan instrucciones "if" para tomar decisiones y elegir alternativas dentro de los juegos de ingenio. Los estudiantes proponen ideas sobre maneras innovadoras de lograr que sus apps sean únicas.

Resumen del curso (continuación)

Lección 8 | Pensar en ciclos: Bucles "while" y bucles anidados. Mientras administran una tienda de rosquillas virtual, los estudiantes usan bucles "while" y bucles anidados para crear algoritmos para que el bot de rosquillas glasee suficientes rosquillas para cada cliente. En Tynker, los estudiantes usan bucles para acortar el código. Los estudiantes forman grupos de diseño de apps y comienzan a preparar el prototipo de una app diseñada por ellos.

Lección 9 | Pensar de manera convencional y no convencional: Variables, entrada de datos y salida de datos. Mientras usan variables para diseñar un algoritmo para una competencia de poesía improvisada, los estudiantes interpretan una canción o un rap basado en el aporte del público. En Tynker, usan variables a medida que se enfrentan a juegos de ingenio más complejos, utilizando todas las habilidades de programación que aprendieron hasta el momento. Los estudiantes realizan entrevistas con los usuarios para que les ayuden a identificar el público de su app.

Lección 10 | Pensar en práctica: Diseño de IU. Los estudiantes analizan los aspectos que constituyen un diseño acertado y diseñan un cartel para su escuela. En la actividad de Tynker, usan todas las habilidades que adquirieron para completar las lecciones de Empezar a programar 2. Los estudiantes aprenden sobre la interfaz de usuario y la experiencia del usuario, y crean un panel de ideas para el diseño de su app. Elaboran una promoción de la app en el proyecto principal.

Información adicional

Descargar los recursos de Empezar a programar

- Tynker
- codeSpark Academy
- Empezar a programar 1
- Empezar a programar 2

Descargar los recursos de Swift Playgrounds

- Aprender a programar 1 y 2: Curso de iTunes U
- Aprender a programar 1 y 2: Guía para profesores
- Aprender a programar 3: Guía para profesores
- App Swift Playgrounds

Descargar las guías de Desarrollo de apps con Swift

- Introducción al desarrollo de apps con Swift
- Introducción al desarrollo de apps con Swift: Guía para profesores
- Desarrollo de apps con Swift
- Desarrollo de apps con Swift: Guía para profesores

Recursos adicionales

- Obtén más información sobre el programa Programación para todos.
- Obtén más información sobre Swift.
- Obtén más información sobre Xcode.
- Conéctate con otros docentes en los foros de Apple Developer.
- Obtén más información sobre codeSpark Academy.
- Obtén más información sobre Tynker.

Correlación curricular: Empezar a programar 1

Las lecciones de Empezar a programar 1 tienen una correlación con el concepto de algoritmos y programas de los estándares provisionales de 2016 para las ciencias de la computación K–12 de nivel 1 de la Asociación de Profesores de Ciencias de la Computación (CSTA) (preescolar a segundo grado: K-2).

Correlación de Empezar a programar 1 y los estándares para ciencias de la computación K–12 de la CSTA, nivel 1 (preescolar a segundo grado)

Capítulos de Empezar a programar 1	**Estándar de la CSTA**	**1A-A-7-1** Contenido de acreditación	**1A-A-5-2** Construir programas	**1A-A-5-3** Diseñar documentos	**1A-A-4-4** Representación de datos	**1A-A-3-5** Descomposición	**1A-A-3-6** Clasificación de elementos	**1A-A-3-7** Algoritmos	**1A-A-6-8** Análisis y depuración
	Correlación general		●	●	●	●	●	●	●
	Poner en orden: Introducción a la secuenciación		●	●	●	●	●	●	●
	Identificar los pasos: Creación de secuencias		●	●	●	●	●	●	●
	Elegir un orden: Secuenciación flexible		●	●	●	●	●	●	●
	Repetir acciones: Bucles		●	●	●	●	●	●	●
	Reparar código: Depuración		●	●	●	●		●	●
	Generar sucesos: Eventos y acciones		●	●	●	●		●	●
	Seguir reglas para lograr resultados: Instrucciones "if"		●	●	●	●		●	●
	Resolver problemas: Algoritmos		●	●	●	●		●	●

Clave: ● Correlación general ● Se correlaciona con el estándar

Correlación curricular: Empezar a programar 2

Las lecciones de Empezar a programar 2 tienen una correlación con el concepto de algoritmos y programas de los estándares provisionales de 2016 para las ciencias de la computación K–12 de nivel 1 de la Asociación de Profesores de Ciencias de la Computación (CSTA) (grados 3 a 5).

Correlación de Empezar a programar 2 y los estándares para las ciencias de la computación K–12 de la CSTA, nivel 1 (grado 3 a 5)

Capítulos de Empezar a programar 2	**1B-A-2-1** Estrategias de colaboración	**1B-A-7-2** Citación de fuentes y documentación	**1B-A-5-3** Planificación	**1B-A-5-4** Construcción de programas	**1B-A-5-5** Operaciones matemáticas	**1B-A-3-6** Descomposición	**1B-A-3-7** Algoritmos	**1B-A-6-8** Análisis y depuración
Correlación general	●	●	●	●	●	●	●	●
Pensar en pasos: Resolver problemas mediante algoritmos	●		●	●		●	●	●
Pensar en soluciones: Depuración	●	●	●	●		●	●	●
Pensar en círculos: Búsqueda de bucles	●		●	●		●	●	●
Pensar en partes: Composición y descomposición	●		●	●		●	●	●
Pensar en conjuntos: Abstracción	●		●	●			●	●
Pensar en patrones: Formar funciones	●	●	●	●		●	●	●
Pensar en detalles: Instrucciones condicionales	●	●	●	●		●	●	●
Pensar en ciclos: Bucles "while" y bucles anidados	●		●	●	●	●	●	●
Pensar de manera convencional y no convencional: Variables, entrada de datos y salida de datos	●		●	●		●	●	●
Pensar en práctica: Diseño de IU	●	●	●	●		●		●

Clave: ● Correlación general ● Se correlaciona con el estándar

Las funcionalidades están sujetas a cambios. Es posible que algunas funcionalidades no estén disponibles en todas las regiones o en todos los idiomas.