Pensamiento computacional en el aula: recursos y estrategias

Autoría: Nivard Arboix González
El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por la profesora: Teresa Romeu Fontanillas
PID_00301455
Primera edición: septiembre 2024

Introducción

El pensamiento computacional es una competencia que ayuda al alumnado a desarrollar habilidades de pensamiento tanto en el marco de la computación como en distintos ámbitos de la vida. Es una competencia básica y fundamental para los ciudadanos de la sociedad digital que debe formar parte de la enseñanza curricular. Por este motivo, es necesario que los docentes aprendan a integrar significativamente esta competencia en su didáctica.

Este recurso está orientado a futuras y futuros maestros con el objetivo de facilitar la adquisición de un conocimiento básico y fundamental sobre el pensamiento computacional y su aplicación en contextos educativos.

1. Definición y justificación

1.1. Definición

El término *pensamiento computacional* no es nuevo. Ya entre 1970 y 1980 Seymur Paper introdujo este término en algunas de sus publicaciones cuando inició sus trabajos con el lenguaje de programación Logo en el MIT. Pero no fue hasta el 2006 cuando Wing publicó un artículo en el que establecía los elementos clave que definían una primera aproximación al concepto de *pensamiento computacional*:

> «Implica la resolución de problemas, el diseño de sistemas y la comprensión de la conducta humana, haciendo uso de los conceptos fundamentales de la informática. [...] El pensamiento computacional es una habilidad fundamental para todos, no solo para los informáticos. En la lectura, escritura y aritmética, deberíamos añadir el pensamiento computacional a la capacidad analítica de cada niño».
>
> Wing (2006, p. 33)

Posteriormente, otros autores y autoras han enriquecido este término y, si bien no se ha llegado a una única definición consensuada, la comunidad científica da por buena la propuesta elaborada en 2015 por el CSTA (Computer Science Teachers Association) y la ISTA (Society for Technology in Education) que define el pensamiento computacional como un proceso de solución de problemas que incluye, entre otras, las siguientes características:

- Formular problemas de forma que se pueda utilizar un ordenador o máquinas para resolverlos.
- Organizar y analizar datos lógicamente.
- Representar datos a través de abstracciones tales como modelos y simulaciones.
- Automatizar soluciones a través del pensamiento algorítmico (una serie de pasos discretos y ordenados).
- Identificar, analizar e implementar posibles soluciones con el fin de conseguir la combinación más efectiva y eficiente de pasos y recursos.
- Generalizar y transferir este proceso de solución de problemas a una amplia variedad de situaciones.

En los últimos años, el concepto de pensamiento computacional ha superado la barrera de los ordenadores para redefinirse como un proceso de pensamiento que no depende necesariamente de las ciencias de la computación (Bocconi *et al.*, 2016). Es decir, la competencia en pensamiento computacional va más allá de la tecnología, y las habilidades asociadas a ella son aplicables en diversos contextos del día a día.

1. Definición y justificación

1.2. Justificación

Actualmente, la mayoría de los países de la Unión Europea incluyen el pensamiento computacional en los currículos de educación primaria y secundaria obligatoria, tal y como evidencia el informe *Reviewing Computational Thinking in Compulsory Education* (Bocconi *et al.*, 2022). En el informe del Join Research Center elaborado por Bocconi *et al.* (2016) ya se identificaban dos razones por las que los gobiernos tendían a incluir cada vez más el pensamiento computacional en los currículos. En primer lugar, para que los niños y jóvenes puedan «pensar de forma diferente, expresarse a través de una variedad de medios, resolver problemas del mundo real y analizar temas cotidianos desde una perspectiva diferente». Y, en segundo lugar, «para impulsar el crecimiento económico, cubrir los puestos de trabajo TIC y prepararse para futuros puestos de trabajo».

La relación entre programación y pensamiento computacional es muy estrecha. Sin embargo, la finalidad del pensamiento computacional no es aprender a programar, sino desarrollar habilidades que permitan un pensamiento reflexivo y resolutivo de problemas, atendiendo a necesidades sociales de distinto tipo, con la ayuda de las herramientas informáticas que tenemos en nuestro entorno cotidiano (Ballarades; Avilés; Pérez, 2016). Por este motivo, la tendencia de los currículos actuales es situar el pensamiento computacional en el ámbito de la competencia digital. Este hecho se pone de manifiesto en los diferentes marcos de referencia de la competencia digital que han publicado UNESCO (2018), JRC European Union (2017), ISTE (2021, 2017) e INTEF (2017), en los que el pensamiento computacional toma protagonismo en algunas de las áreas que se describen.

La literatura sobre el pensamiento computacional en edades escolares cuenta con un considerable número de publicaciones, pero hasta la fecha no existe un amplio consenso sobre los contextos y las didácticas más apropiadas para su enseñanza. Sin embargo, los autores y autoras coinciden en su importancia en la formación de niños y jóvenes. Las distintas miradas convergen en la idea de que en la sociedad digital el pensamiento computacional es de gran valor desde el punto de vista educativo, social, cultural y económico. Permite resolver problemas científicos y tecnológicos, pero también de otros muchos ámbitos. Fomenta la práctica del trabajo colaborativo, la creatividad y el razonamiento lógico. Y consolida la fluidez para comunicar ideas de forma efectiva y creativa (Resnick y Rusk, p. 122), entre otros. Por eso, autores como Bers *et al.* (2019, p. 131) definen el pensamiento computacional como una competencia fundamental del siglo XXI.

2. Las cinco habilidades fundamentales del pensamiento computacional

2.1. Introducción

Buena parte de los puestos de trabajo que existirán en los próximos años todavía no han sido inventados. Las personas tendrán que ser capaces de adaptarse a trabajos emergentes y de aplicar un conjunto de habilidades propias del siglo XXI, entre las que desempeñan un papel importante las habilidades tecnológicas vinculadas al pensamiento computacional (Coronel y Lima, 2020).

En este sentido, existe una extensa literatura sobre las habilidades asociadas al pensamiento computacional. Bocconi *et al.* (2016) identificaron las habilidades más presentes en la literatura y confeccionaron la siguiente lista:

- Abstracción
- Información
- Análisis
- Pensamiento algorítmico
- Automatización
- Pensamiento recursivo
- Representación de datos
- Simulación
- Formulación de problemas para poder resolverlos con ordenadores
- Creación de programas de computadora
- Generalización
- Evaluación y depuración

En el ámbito educativo, esta lista se sintetiza en cinco habilidades objetivo que todo estudiante debería adquirir de manera gradual durante su escolarización obligatoria (CAS, 2015; Dorling, 2015):

- Algorística
- Descomposición
- Abstracción
- Generalización y patrones
- Evaluación

Estas habilidades se convierten en habilidades de gran valor, además, porque son aplicables en muchos contextos más allá de lo tecnológico. Por este motivo, deberían formar parte de la formación integral de cualquier persona, para que pueda vivir, trabajar, aprender, comunicarse y participar en la sociedad de la información (Joint Research Center, 2016).

2. Las cinco habilidades fundamentales del pensamiento computacional

2.2. Algorística

Un algoritmo es una secuencia de instrucciones o un conjunto de normas para llegar a una solución. En la elaboración de un algoritmo, es necesario saber definir cada una de las acciones a realizar y el orden en que deben realizarse.

En el día a día, aplicamos algoritmos constantemente, por ejemplo en la realización de una receta de cocina, al dar instrucciones a una persona para ir de un punto a otro, o cuando hacemos la colada.

En programación, el algoritmo es el conjunto de códigos que permite llegar a una solución. Es el texto que describe cómo hacer algo con un determinado lenguaje de programación. De la misma forma que en cualquier otro ámbito, el algoritmo es reutilizable y aplicable en problemas de las mismas características. Una vez hemos creado el algoritmo para responder si un número es par o impar, lo podemos aplicar a cualquier número, de la misma manera que podemos hacer la receta de un bizcocho aunque esté en una cocina diferente. Las instrucciones son las mismas.

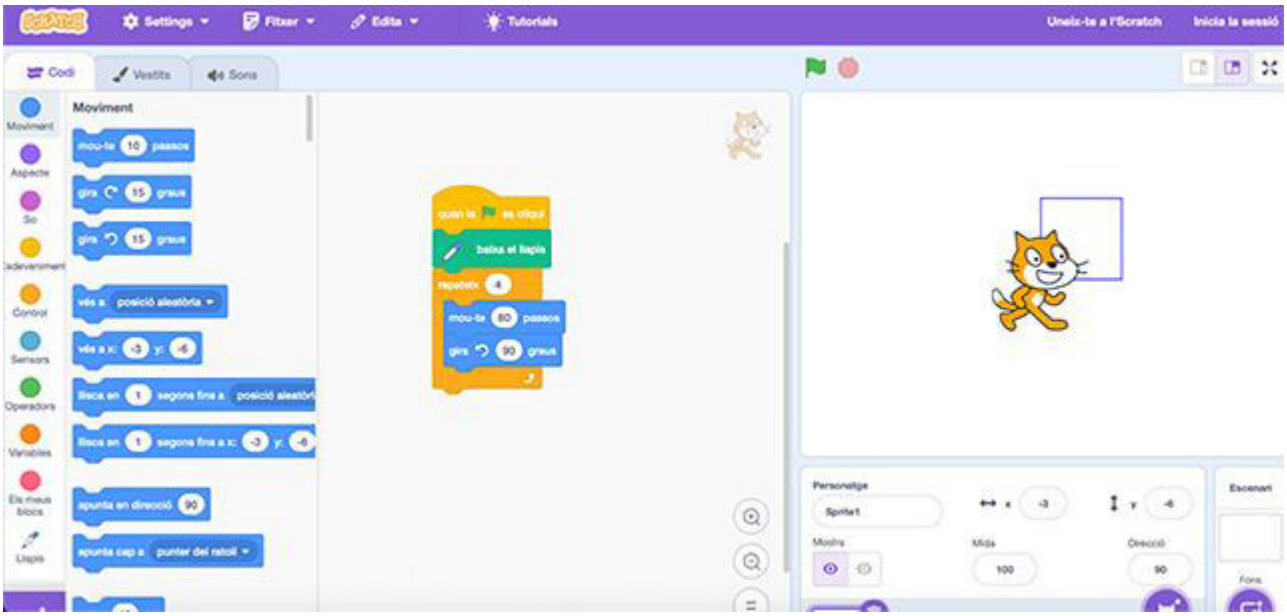

Figura 1. Algoritmo
Fuente: Scratch.mit.edu

En la figura 1 se puede ver un algoritmo programado con el lenguaje Scratch del MIT.

Scratch es un entorno de programación visual desarrollado para el MIT Media Lab que permite a los estudiantes crear historias interactivas, juegos y animaciones mediante el uso de bloques de programación en lugar de código de programación tradicional. Esto hace que aprendan los conceptos básicos de la programación, tales como secuencias de instrucciones, bucles y condicionales, y apliquen las habilidades básicas del pensamiento computacional de forma visual, intuitiva y divertida.

En este caso concreto de la figura 1, el programa indica bloque a bloque los pasos necesarios para dibujar un cuadrado en pantalla. El primer bloque ejecuta el programa. El segundo baja el lápiz –indispensable para poder dibujar. El tercer bloque corresponde a una repetición: un cuadrado tiene cuatro lados y cuatro ángulos iguales, y, por tanto, dibujaremos cuatro veces lo mismo. Los bloques cuarto y quinto responden al dibujo de un lado y a un giro de noventa grados respectivamente. Este algoritmo con el lenguaje de programación Scratch sirve para dibujar cualquier cuadrado. Podemos cambiar su tamaño pero, mientras sea un cuadrado, siempre será un algoritmo válido.

2. Las cinco habilidades fundamentales del pensamiento computacional

2.3. Descomposición

En la resolución de problemas complejos, una habilidad fundamental es saber dividir este problema en problemas más sencillos, cuya resolución conduce a la solución.

La descomposición es una habilidad aplicada en muchos ámbitos de la vida, como en la organización de un viaje familiar (definición del presupuesto, compra de los billetes de avión, reserva de las habitaciones de hotel, programación de las jornadas, etc.). A menudo, en las clases de matemáticas se aplica de forma natural cuando un problema requiere varias operaciones para encontrar datos que permiten realizar otras operaciones que darán lugar a la respuesta final.

La descomposición es una habilidad clave para hacer frente a la programación de cualquier proyecto. Por ejemplo, un equipo de programadores no puede abordar la programación de un videojuego como una sola labor. Requiere una planificación del trabajo que divida el videojuego en tareas más pequeñas y asumibles que permitan ir completando el proyecto de forma razonable. En este mismo sentido, el alumnado de primaria necesita aplicar esta habilidad para descomponer las tareas complejas planteadas en una serie de pasos e irse acercando a la solución final de una forma más sencilla y alentadora.

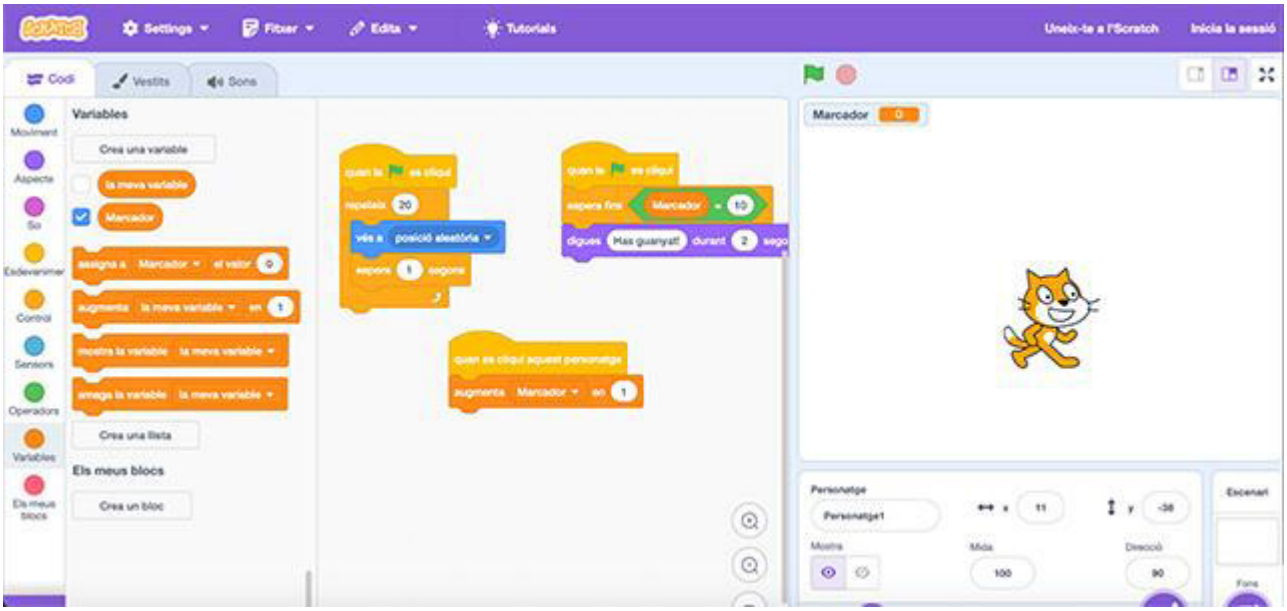

Figura 2. Descomposición
Fuente: Scratch.mit.edu

En la figura 2 se presenta un juego elaborado con Scratch. El objetivo del juego consiste en llegar a los diez puntos haciendo clic en el personaje que se va moviendo aleatoriamente por la pantalla a una velocidad de una posición por segundo. Esta tarea la podemos fragmentar en tres retos más sencillos:

1. Elaborar un código para que el personaje se mueva aleatoriamente por la pantalla cada segundo.
2. Crear un marcador para que aumente su valor cuando se clique sobre el personaje.
3. Establecer una condición para que el programa esté pendiente de cuando el marcador llegue a diez puntos. En ese momento, el personaje debe decir «¡Has ganado!».

De esta forma, la realización del juego se simplifica resolviendo tres retos que por separado son más asumibles.

2. Las cinco habilidades fundamentales del pensamiento computacional

2.4. Abstracción

La abstracción es una habilidad clave del pensamiento computacional. Consiste en identificar lo que es realmente relevante y descartar todo lo que no es estrictamente necesario. De esta manera, la información importante resulta más comprensible y, al mismo tiempo, resulta más simple centrarse en lo esencial.

Por ejemplo, en las estaciones de metro a menudo podemos encontrar una representación de las estaciones sobre una línea recta. En esta representación aparecen de forma ordenada las estaciones de la línea, se señala la estación donde nos encontramos y se indican las correspondencias con otras líneas. Se ha obviado el recorrido real que hace el metro: todas las estaciones están dispuestas en línea recta. También se han obviado las distancias entre estaciones, puesto que en la realidad no son todas equidistantes. Pero esta abstracción permite a las personas identificar dónde se encuentran, cuántas paradas deben hacer para llegar a su destino o en qué estación deben bajar para cambiar de línea.

Otro ejemplo lo encontramos en el menú diario de los restaurantes. En una sola hoja de papel se presentan unas opciones de primer plato, unas opciones de segundo plato y el postre, con los correspondientes precios. No se especifican todos los ingredientes que se utilizan en cada plato ni el proceso que se sigue para cocinarlos, pero esta simplificación se centra en lo esencial, y así, de forma rápida y sencilla, los clientes pueden elegir el almuerzo por un precio cerrado.

A la hora de programar, la abstracción permite centrarse en el núcleo del programa. Permite identificar los bloques de código a partir de los cuales se creará el programa entero. En la figura 3 se muestra un programa para dibujar un pentágono regular de un color y tamaño determinados. A partir de este ejemplo, se pide al alumnado que cree un programa para dibujar un hexágono.

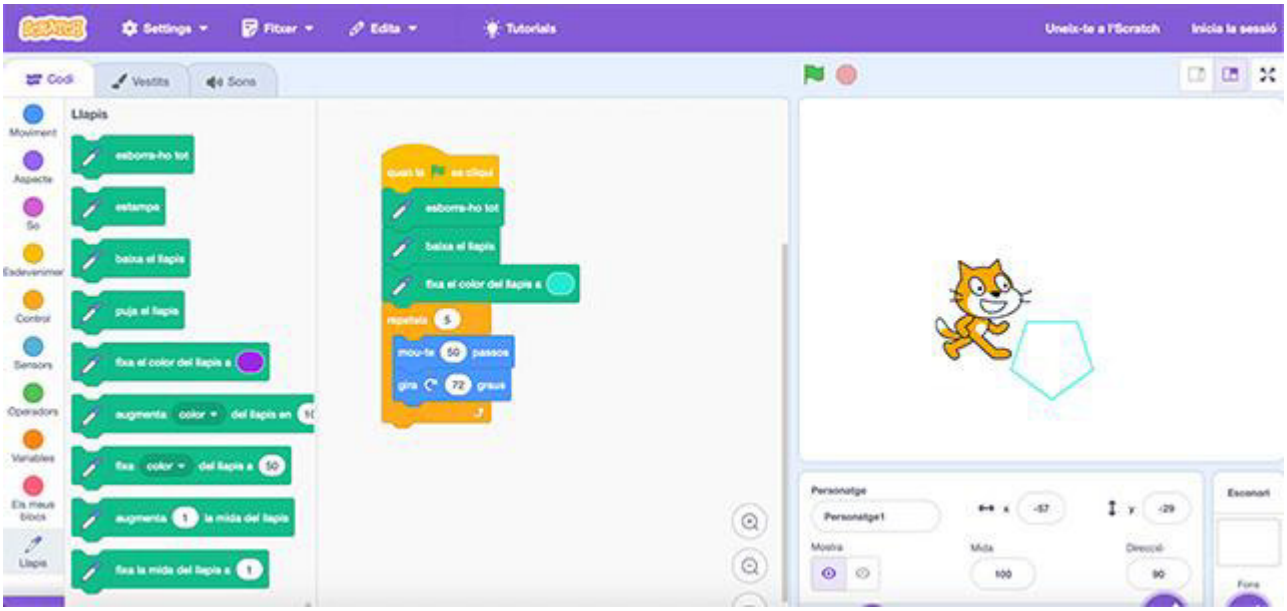

Figura 3. Abstracción
Fuente: Scratch.mit.edu

Analizando el programa, podemos observar que cualquier polígono se podrá dibujar haciendo una repetición de tantas veces como lados tenga el polígono y dando un giro en grados equivalente a dividir una vuelta entera, 360 grados, por el número de lados, ya que el dibujo comienza y termina en el mismo punto. En la siguiente tabla se muestran tres ejemplos:

Taula 1. Aplicación de la abstracción

	Número de lados	Repeticiones	Giro en grados (360/lados)
Pentágono	5	5	360/5 = 72

Hexágono	6	6	360/6 = 60
Pentágono	7	7	360/7 = 51,43

Fuente: elaboración propia.

En este caso, pues, la elaboración del programa se reduce a identificar el número de lados del polígono y a dividir 360 grados por ese número de lados. Este es el corazón de esta programación, y el código se programará a partir de este concepto central. Habrá que aplicar otros conceptos computacionales y habilidades para completarlo, pero el núcleo es esta abstracción.

2. Las cinco habilidades fundamentales del pensamiento computacional

2.5. Generalización y patrones

La identificación de patrones permite realizar predicciones, establecer normas y resolver problemas que comparten una misma analogía.

Encontramos patrones en muchas situaciones del día a día. En la carretera, los servicios de tráfico tienen muy claro un patrón que se repite casi cada semana: los viernes muchos vehículos salen de las grandes ciudades, y los domingos vuelven. Este patrón permite anticipar medidas para mejorar el tráfico: abrir carriles alternativos, avisar a los usuarios para que salgan o regresen de forma escalonada, activar más servicios de emergencia por si hay algún accidente, etc.

En la calle encontramos otro ejemplo de patrones, en este caso para realizar un programa eficiente: los semáforos siguen una sucesión de iluminación (verde, naranja, rojo) en unos intervalos de tiempo determinados. La programación se basa en una repetición de esa serie de colores a lo largo del día. Basta programarla una vez y hacer que el programa se repita permanentemente. Es decir, la identificación del patrón reduce la escritura de la programación a una sola vez.

En programación, los patrones son básicos y fundamentales para elaborar programas más eficientes y reutilizar programas para contextos de las mismas características. Cuando identificamos patrones y estos patrones nos permiten establecer una respuesta común a problemas de la misma analogía, decimos que hemos llegado a una generalización.

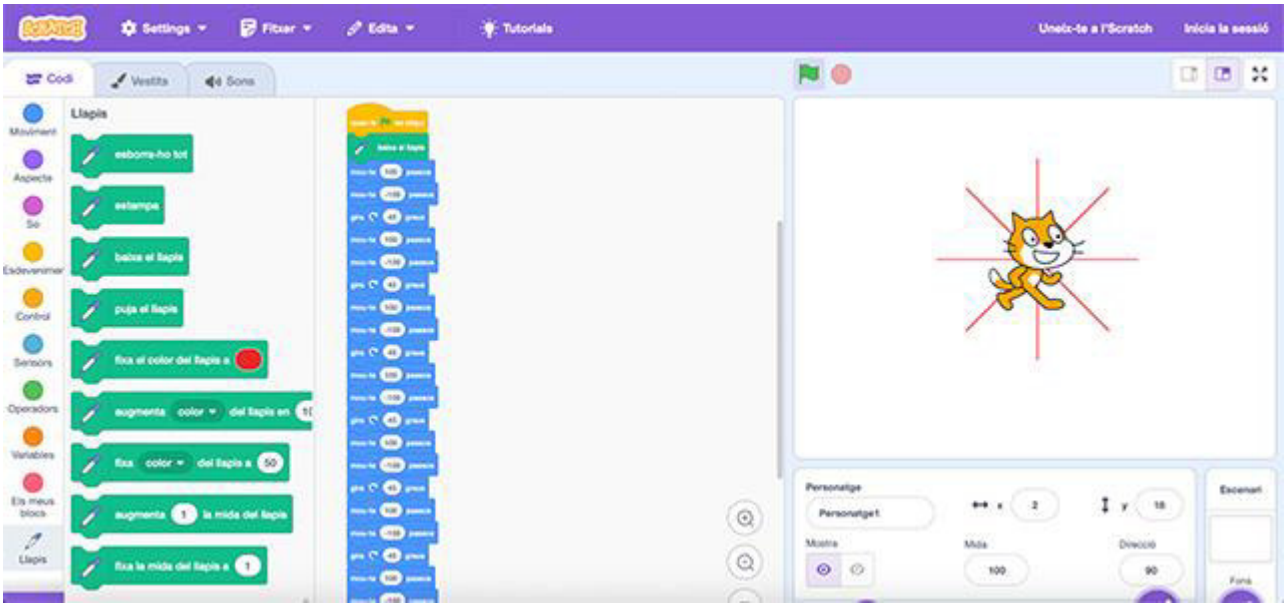

Figura 4. Búsqueda de patrones
Fuente: Scratch.mit.edu

En este programa se dibuja en la pantalla una figura en forma de estrella de ocho puntas. Una lectura atenta del programa permite observar que hay tres bloques de código que se repiten sucesivamente: muévete hacia delante (100), muévete hacia delante (–100) y gira (45º). Este conjunto de bloques forma un patrón y se repite ocho veces. Esto significa que este programa se puede escribir de forma más eficiente, introduciendo un bloque de repetición (bucle) de la siguiente forma:

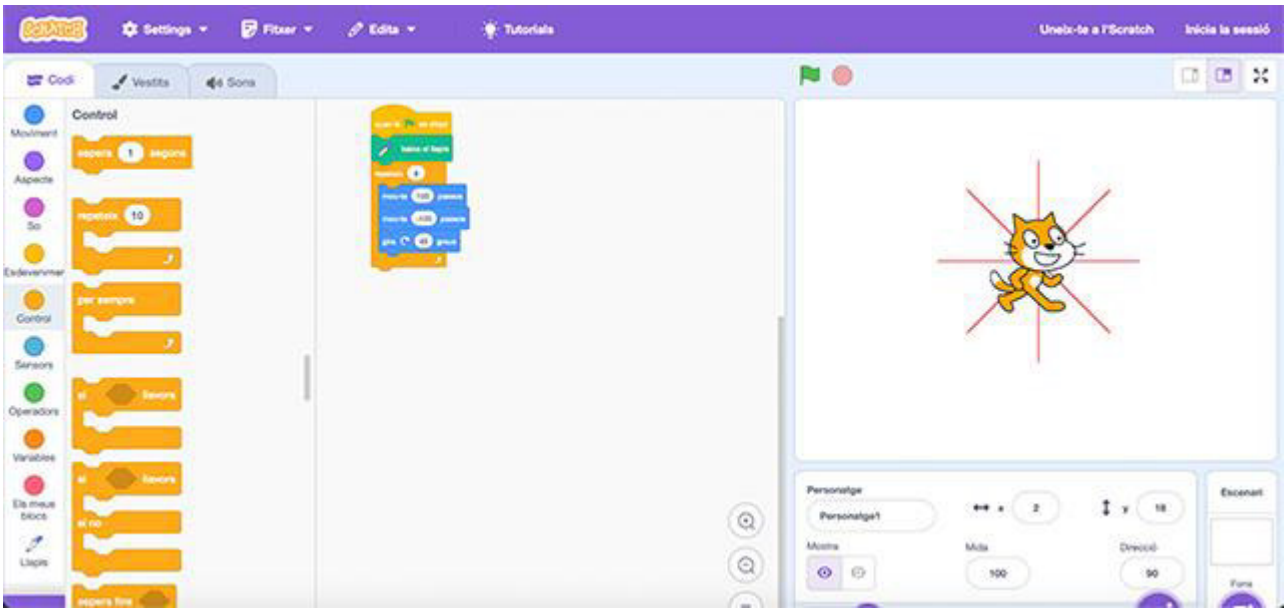

Figura 5. Aplicación de patrones
Fuente: Scratch.mit.edu

2. Las cinco habilidades fundamentales del pensamiento computacional

2.6. Evaluación

Aplicamos la evaluación constantemente en muchas actividades de nuestra vida. De la misma forma que es importante una evaluación externa que valide nuestro trabajo, también lo es la capacidad como personas de valorar nosotros mismos y objetivamente lo que hemos hecho. No se trata de una evaluación cuantitativa, sino cualitativa. Es recordar el objetivo con el que iniciamos una tarea y valorar si el resultado ha sido bueno, si requiere mejoras o si no ha sido correcto.

Por ejemplo, si hemos pintado una habitación de nuestra casa, valoramos el resultado final. ¿Están todas las paredes bien pintadas? ¿El color es uniforme? ¿Las esquinas y los rincones han quedado bien? Este análisis debe permitirnos dar el trabajo por bueno y, por tanto, por terminado, o bien llevar a cabo acciones complementarias para obtener el resultado esperado.

En otras situaciones puede pasar que, aunque el resultado sea correcto, no sea la mejor opción.

Pongamos por caso que una escuela decide abrir una nueva puerta de acceso más segura para su alumnado. El resultado final es muy bueno en muchos sentidos: la puerta es mayor, la acera de enfrente es amplia y separada de los vehículos por una barandilla, y para las familias es más fácil localizar a sus hijos e hijas. Pero la nueva puerta tiene un pequeño desnivel respecto a la calle que se ha salvado con dos escalones. La evaluación de esta solución permite observar que el alumnado en silla de ruedas y los familiares con cochecito de bebé tendrán un impedimento importante para acceder a la escuela.

En programación, la evaluación es una habilidad imprescindible para alcanzar los mejores resultados y la máxima eficiencia. Si concebimos la programación como un diagrama de flujo, la evaluación sería siempre el paso previo a su finalización. Si la evaluación es buena, hemos terminado. Si la evaluación no es suficientemente buena, volvemos al objetivo inicial y revisamos el trabajo para mejorar la solución.

En la figura 6 se muestra un reto de Scratch en el que se pide dibujar una circunferencia de color azul. Tal y como puede verse en el área derecha, el resultado es correcto. Sin embargo, aplicando la habilidad de evaluación, en el código se identifica un error.

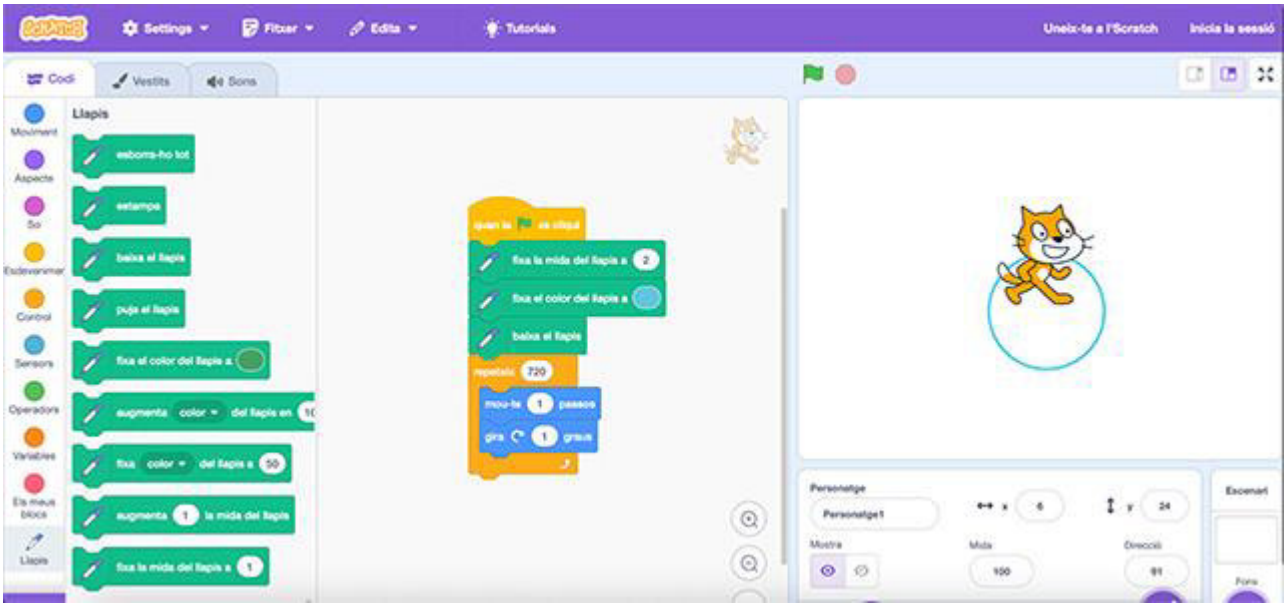

Figura 6. Evaluación
Fuente: Scratch.mit.edu

El dibujo de la circunferencia se puede trazar con un movimiento repetitivo de un píxel y un grado 360 veces, que corresponden a los 360 grados de una vuelta entera. Sin embargo, en este caso la repetición es de 720 veces, lo que implica que se trazan dos

vueltas dibujando dos circunferencias idénticas una sobre la otra. Si trasladamos esta programación al mundo industrial, podría tratarse de un programa que cose una circunferencia en una camiseta. Esto significa que en la producción de cada camiseta se estaría utilizando el doble de hilo por unidad, y esto, multiplicado por miles de camisetas, sería un gasto considerable. Por ese motivo, la evaluación es una habilidad muy importante en el pensamiento computacional. Hay que saber valorar el trabajo realizado y determinar objetivamente si se ha encontrado la mejor solución para el objetivo planteado.

2. Las cinco habilidades fundamentales del pensamiento computacional

2.7. Resumen de ideas clave

Las habilidades de pensamiento computacional deben concebirse también como un todo en el que las habilidades están interrelacionadas y se aplican a menudo de forma integrada. En estas relaciones, la evaluación es un punto de convergencia con todas las demás habilidades para establecer con frecuencia la necesidad de aplicar o repetir alguna habilidad para lograr un mejor resultado.

En resumen, las ideas clave que definen las habilidades de pensamiento computacional son las siguientes:

Habilidad	Descripción
Algorística	Desarrollar un conjunto de instrucciones paso a paso para resolver un problema o realizar una tarea de manera eficiente y efectiva
Descomposición	Dividir un problema complejo en partes más pequeñas y manejables para comprender mejor el problema y abordarlo de forma más efectiva.
Abstracción	Identificar aspectos esenciales de un problema e ignorar los detalles no relevantes. Utilizar modelos y representaciones para simplificar y comprender el problema.
Generalización y patrones	Identificar tendencias o secuencias regulares en datos o problemas y utilizarlas para predecir o resolver situaciones nuevas.
Evaluación	Revisar el resultado final de una tarea y valorar objetivamente si se ha cumplido con su objetivo inicial. En caso negativo, identificar qué mejoras deberían aplicarse para lograr un resultado satisfactorio.

3. Conexiones entre el pensamiento computacional y ámbitos curriculares de primaria

Los ejemplos explicados junto con las habilidades del pensamiento computacional nos han permitido captar que estas habilidades se pueden aplicar en muchos ámbitos de la vida, por lo que su valor traspasa la frontera de los entornos de programación para convertirse en un conjunto de habilidades clave para la ciudadanía. Si bien existen varios entornos de tecnología digital (como Scratch o Code) que permiten realizar un trabajo sistemático del pensamiento computacional, existen otros ámbitos curriculares que permiten trabajar el pensamiento computacional a través de actividades concretas.

En este sentido, a continuación se presentan ejemplos de actividades basadas en las habilidades de pensamiento computacional enmarcadas en distintos ámbitos curriculares.

Algorística

- Escribir los pasos para realizar una multiplicación italiana.
- Elaborar un diagrama para explicar la fotosíntesis.
- Explicar en vídeo los pasos para realizar un modelo de papiroflexia.
- Elaborar un tutorial para editar una imagen.
- Redactar las instrucciones de un juego.
- Crear un guion ilustrado de un corto.
- Escribir una receta de cocina.
- Redactar los pasos para resolver un tipo de problema matemático.
- Dibujar un recorrido en un mapa.
- Inventar los movimientos para una coreografía.
- Elaborar una línea del tiempo.
- Redactar los pasos para la realización de un experimento.
- Secuenciar en fotografías la construcción de una maqueta.
- Hacer un árbol genealógico.
- Crear una serie de ejercicios de calentamiento para una clase de educación física

Descomposición

- Planificar un texto con las ideas clave de acuerdo con una estructura del tipo inicio, nudo y desenlace.
- Dividir un problema matemático complejo en partes más sencillas.
- Identificar las comarcas de Cataluña: primero las de costa, después las de montaña y finalmente las de interior.
- Organizar un proyecto de trabajo en una sucesión de tareas.
- Escribir lo necesario para organizar una fiesta.
- Identificar los instrumentos presentes en una canción.
- Identificar los datos necesarios para calcular un área.
- Fraccionar el recorrido de un robot en tramos.
- Enumerar los ingredientes para elaborar una receta de cocina.
- Elaborar un *checklist* para completar una tarea compleja.
- Organizar la visita a una ciudad a partir de sus puntos de interés.
- Escribir el contenido para una mochila de colonias.
- Dividir una canción en partes para aprenderla.
- Explicar las posiciones de los jugadores de un equipo de baloncesto.
- Explicar las etapas que sigue un producto cuando lo compramos por internet.

Abstracción

- Resumir el ciclo del agua en cinco ideas.
- Enumerar los elementos clave de un tema trabajado.
- Elaborar un mapa conceptual sobre los tipos de cuadriláteros.
- Describir una imagen.
- Explicar lo más destacado del fin de semana.
- Presentar una idea para un proyecto en menos de un minuto.
- Contar quién fue Thomas Alba Edison.
- Identificar los datos útiles para la resolución de un problema matemático.
- Realizar un croquis sobre el mapa de un país.
- Elaborar un decálogo sobre internet segura.
- Hacer un esbozo sobre una maqueta.
- Explicar el argumento de una película.
- Convertir un texto informativo en una infografía.
- Crear un cómic para explicar la leyenda de san Jorge.
- Hacer un resumen sobre un partido de voleibol escolar.

Generalización y patrones

- Deducir una norma ortográfica a partir de textos en los que se aplica.
- Identificar patrones en series alfanuméricas o de imágenes.
- Identificar patrones musicales dentro de una canción.
- Realizar texturas en dibujos a partir de patrones.
- Descubrir el elemento que no cumple el patrón dentro de una serie.
- Jugar al SET.
- Clasificar objetos de acuerdo a cualidades específicas: color, forma, tamaño, etc.
- Aplicar una estrategia conocida en un problema matemático.
- Detectar la frecuencia de un tren a partir de sus horarios.
- Aplicar un bucle para bloques de programación iguales.
- Autoevaluarse a partir de un trabajo de referencia.
- Clasificar animales según su grupo de vertebrados.
- Prever hechos de acuerdo con patrones observados.
- Completar figuras de tangram.
- Trabajar rutinas para consolidar hábitos.

Evaluación

- Utilizar una lista de control para revisar el trabajo una vez finalizado.
- Compartir los objetivos al inicio de la actividad y revisarlos al terminar.
- Definir unos criterios que permitirán valorar si una tarea se ha cumplido de acuerdo con su objetivo inicial.
- Evaluar el trabajo de otra persona a partir de una rúbrica.
- Elaborar una crítica sobre un libro.
- Escoger la mejor opción de entre varias argumentando la respuesta.
- Elaborar un portafolio durante la realización de un tema o proyecto.
- Optimizar el código de una programación para que sea más eficiente.
- Analizar datos para tomar una decisión.
- Enumerar los puntos fuertes y débiles sobre un tema de debate.
- Identificar errores en la resolución de unos problemas matemáticos.
- Describir la metacognición asociada a un proceso de aprendizaje.
- Autoevaluar el trabajo individual en el marco de un trabajo en equipo.
- Grabarse en vídeo y valorar los aspectos destacados y los aspectos a mejorar.
- Innovar un aspecto de la escuela para mejorarlo.

n los enlaces s gu en es desarrollamos los ejemplos e las in o habil a es fun amentales del pensam ento computa ional:

- **Algorística**
- **Descomposición**
- **Abstracción**
- **Generalización y patrones**
- **Evaluación**

Conclusión

A lo largo de este material se ha puesto de manifiesto que el pensamiento computacional es una competencia esencial en el siglo XXI. Incluye un conjunto de habilidades clave para la resolución de problemas, para el uso eficiente de lenguajes de programación y también para otros muchos ámbitos de la vida. Por este motivo, es primordial que todos los docentes integren su práctica en sus programaciones didácticas. En este sentido, Bocconi *et al.* (2016) sugieren que el pensamiento computacional se puede integrar de diversas formas en el currículo: como tema transversal, como asignatura separada, dentro de otras asignaturas como las matemáticas o la tecnología, o, incluso, de forma separada del ámbito digital.

Lo importante es definir como centro educativo una secuenciación del trabajo del pensamiento computacional a lo largo de los cursos, estableciendo una coherencia vertical y asegurando una buena competencia del alumnado al terminar la etapa de educación primaria. Es por ello que varios autores subrayan la necesidad de formar a los equipos de maestros para que adquieran una buena experiencia didáctica en esta materia, no solo para convertirse en buenos guías en el correspondiente aprendizaje de los alumnos, sino también para integrar y trabajar el pensamiento computacional en el día a día y desde diferentes perspectivas y situaciones.

Bibliografía

Bocconi, S., Chioccariello, A., Dettori *et al.* (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*, EUR 28295 EN. https://data.europa.eu/doi/10.2791/792158

CAS, Computing at School (2015). *CAS Barefoot* [Página web]. http://barefootcas.org.uk/

Dorling, M. (2015). *CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic* [Documento en línea]. http://community.computingatschool.org.uk/resources/1692

García-Peñalvo, F. y Rojas-López, A. (2020). Evaluación de habilidades del pensamiento computacional para predecir el aprendizaje y retención de estudiantes en la asignatura de programación de computadoras en educación superior. *RED. Revista de Educación a Distancia*, *63*(20), art. 4. DOI: http://dx.doi.org/10.6018/red.409991

Loureiro, A.C. *et al.* (2022). Computational thinking in teacher digital competence frameworks. *Revista Prisma Social*, (38), pp. 77-93. https://revistaprismasocial.es/article/view/4783

Polanco, N., Ferrer, S. y Fernández, M. (2021). Aproximación a una definición de pensamiento computacional. *RIED. Revista Iberoamericana de Educación a Distancia, 24*(1), pp. 55-76. DOI: http://dx.doi.org/10.5944/ried.24.1.27419

Soria, E. y Rivero, C. (2019). Pensamiento computacional: una nueva exigencia para la educación del siglo XXI. *Revista Espaço Pedagógico*, *[S. l.]*, *2*(26) pp. 323-337. https://seer.upf.br/index.php/rep/article/view/8702