1. Introducción al Pensamiento Computacional

- 1.1 Definiciones previas: conceptos básicos asociados
- 1.2 Pensamiento computacional: evolución conceptual
- 1.3 Componentes del Pensamiento Computacional
- 1.4 Saberes básicos, competencias específicas y criterios de evaluación del bloque de pensamiento computacional
- Situación de aprendizaje 1. ¿Cómo pensamos de forma computacional?

Copyright 2026

1.1 Definiciones previas: conceptos básicos asociados

En un momento donde vivimos rodeados de lo digital, surgen nuevas formas de inteligencias (inteligencia digital) y la aparición de nuevas habilidades como es la **codigoalfabetización** (*codeliteracy*) (Zapata-Ros, 2015). Según el autor, podemos definir la codigoalfabetización como el proceso de enseñanza-aprendizaje de la lectoescritura con los diferentes lenguajes de programación, donde lo importante no será el lenguaje de programación en sí mismo, sino la capacidad de realizar estas creaciones independientemente del propio lenguaje de programación. Una persona está códigoalfabetizada cuando es capaz de entender y crear con un lenguaje que los dispositivos programables entiendan. Según el autor, las personas que desarrollan y evolucionan esta capacidad se dice que piensan computacionalmente, que al fin y al cabo no es más que un proceso cognitivo que nos permite resolver problemas, y que finalmente será expresado de una forma códigoalfabetizada.

González ahonda en la concreción y diferenciación de conceptos importantes en la codigoalfabetización como son algoritmo y programa. Un **algoritmo** es una secuencia ordenada de instrucciones u operaciones cuya ejecución en ese correcto orden nos va a dar lugar la solución deseada para un problema. La construcción de estos algoritmos se produce en nuestra mente tras un espacio de tiempo variable de reflexión personal (Moschovakis, 2001). Para facilitar la creación de estos algoritmos y estandarizar propuestas universales independientemente de los lenguajes usados por los humanos, surgieron herramientas que nos van a ayudar en la construcción visual de estos algoritmos como pueden ser los **diagramas de flujo**: unos símbolos o dibujos que van a representar las operaciones básicas de cualquier algoritmo: secuencia, condición, repetición e iteración (Barrera, 2013).

El paso para convertir un algoritmo en un programa es el arte de codificar. La **codificación** tiene que ver con crear un código fuente en un determinado lenguaje de programación partiendo de un algoritmo previamente creado (González-González, 2019). La codificación nos permite la comunicación entre los humanos que crean esos algoritmos y el lenguaje que entiendan las máquinas. Para que los dispositivos puedan ser programados y realizar las funciones que deseamos que hagan, necesitamos convertir ese algoritmo que resuelve el problema en un “ **programa**” con un lenguaje que sí pueda ser entendido por esta máquina, para que pueda ser procesado y finalmente ejecutado (Zapata-Ros, 2015). A esta variedad de lenguajes se les denomina l**enguajes de programación**, de los cuales podemos encontrar cientos de ellos con diferentes propósitos (Chatley, Donaldson y Mycroft 2019). Chatley argumenta que cada lenguaje

Copyright 2026

de programación, al tener sus propias reglas de sintaxis y sus propios conjuntos de instrucciones, evolucionarán en un futuro a otros miles de lenguajes de programación que surgirán a partir de estos.

Bers (2017) argumenta que mientras el pensamiento computacional tiene que ver con habilidades del pensamiento para resolver problemas, la “codificación” se puede ver como una herramienta para enseñar el pensamiento computacional. La codificación es considerada según la Agenda Digital europea como una habilidad clave ya que ayuda a poner en práctica habilidades del siglo XXI tales como la resolución de problemas, el pensamiento analítico y el trabajo en equipo (Bocconi, Chioccariello, Dettori, Ferrari y Engelhardt, 2016).

Así pues, podemos concluir que cualquier programa estará asociado a un algoritmo que habrá resuelto el problema subyacente, de la misma forma que no todos los algoritmos podrán ser expresados como un programa. Además, es importante resaltar que un mismo programa podrá ser escrito por diferentes personas, con diferentes lenguajes de programación utilizando diferente código. Todas estas definiciones y precisiones terminológicas se espera que contribuyan a una lectura más ágil y precisa de las páginas que siguen a continuación, donde nos adentramos de lleno en las diferentes formas que existen para definir el pensamiento computacional.

Todos los conceptos y literatura comentadas en este capítulo están extraídos del trabajo de revisión sistemática sobre métodos de evaluación del pensamiento computacional (Ruiz y Bustamente, 2021).

Bibliografía

Barrera, Lizardo (2013). *Algoritmos y programación para la enseñanza y aprendizaje de la matemática escolar.* En SEMUR, Sociedad de Educación Matemática Uruguaya (Ed.), VII Congreso Iberoamericano de Educación Matemática (pp. 6680-6687). Montevideo, Uruguay: SEMUR.

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom.* Routledge.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site).

Chatley, R., Donaldson, A., & Mycroft, A. (2019). The next 7000 programming languages. *In Computing and Software Science,* 250-282

González-González, C. S. (2019). Estado del arte en la enseñanza del pensamiento computacional y la programación en la etapa infantil. *Education in the Knowledge Society, 2019, Vol. 20, n. 1*, 35

Copyright 2026

Moschovakis, Y. N. (2001). What is an algorithm?. *In Mathematics unlimited—2001 and beyond* (pp. 919-936). Springer, Berlin, Heidelberg.

Ruiz Reinales, C., & Bustamante, J. C. Pensamiento computacional en educación infantil y primaria: una revisión sistemática.

Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *Revista de Educación a Distancia (RED),* (46).

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU



Copyright 2026

1.2 Pensamiento computacional: evolución conceptual

La primera referencia formal de pensamiento computacional la podemos encontrar en el artículo de **Wing** (2006), donde define el pensamiento computacional como la **habilidad que implica resolver problemas, diseñar sistemas, y entender el comportamiento humano** a partir de los conceptos fundamentales de la informática, y que incluye una gama de herramientas mentales que reflejan la amplitud del campo de la informática. La autora argumenta no sólo lo que es el pensamiento computacional, sino también lo que no es:

1. *"Conceptualizing, not programming" (conceptualizar, no programar').* Pensar como un programador de dispositivos va mucho más allá de estar capacitado para programar un ordenador, ya que ello requiere tener la capacidad de pensar en múltiples capas de abstracción.
2. *"A way that humans, not computers, think" ('una manera en que los humanos piensan, no las computadoras').* El pensamiento computacional es una habilidad que las personas usamos para resolver problemas, no para simular el pensamiento de un ordenador. Las máquinas están a nuestra disposición y su forma de realizar las cosas es predecible. Sin embargo, las personas, tenemos la capacidad de ser creativos, inteligentes y espontáneos. Por lo tanto, somos los humanos los que creamos estas máquinas para que nos ayuden, utilizando para ello nuestra inteligencia para acometer y resolver problemas que seguramente seríamos incapaces de poder realizar antes de inventar las máquinas.
3. *"Fundamental, not rote skill" (habilidad básica, no puramente mecánica).* Se considera una habilidad básica aquella que cualquier ser humano tiene que poseer para poder desenvolverse en esta sociedad actual.
4. *"Complements and combines mathematical and engineering thinking" (se complementa y se combina con el pensamiento matemático e ingeniero).* El pensamiento computacional tiene una relación en su origen con el pensamiento matemático, como el resto de ciencias. De la misma forma, tiene una relación con el pensamiento desarrollado en estudios de ingeniería puesto que lo que se crean son construcciones de sistemas informáticos para interactuar con nuestro mundo físico.

Copyright 2026

5. *"Ideas, not artifacts" (ideas, no artefactos).* El pensamiento computacional no sólo está relacionado con las creaciones hardware o software que el ser humano sea capaz de diseñar, sino que también es una habilidad que podemos usar siempre para resolver problemas tan cotidianos como preparar un plato en la cocina o para gestionar mejor nuestra agenda personal.
6. *"For everyone, everywhere" (para cualquiera, en cualquier parte).* El pensamiento computacional será una realidad cuando lo tengamos tan integrado en nuestras formas de abordar tareas, que lo más lógico sea que desaparezca como término y filosofía explícitos.

La Doctora Wing actualizará su propia definición argumentando que el pensamiento computacional incluye los procesos de pensamiento implicados en la formulación de problemas y de sus soluciones, de tal modo que éstos estén representados de una manera que pueda ser abordada efectivamente por un agente-procesador de información (Wing, 2008).

A partir de este momento, se suceden en el tiempo diferentes aportaciones, todas ellas enfocadas a enriquecer los currículos educativos. Fruto del trabajo colaborativo de la "*Computer Science Teachers Association"* (CSTA, 2011) y la "*International Society for Technology in Education*" (ISTE) de los Estados Unidos surge su propia aportación: un enfoque para resolver un problema concreto que ayuda a la inclusión de tecnologías digitales con ideas humanas. Todo ello no reemplaza el énfasis en creatividad, razonamiento o pensamiento crítico pero refuerza esas habilidades al tiempo que realza formas de organizar el problema de manera que el ordenador pueda ayudar (CSTA & ISTE, 2011).

En 2012 la Royal Society (Reino Unido) crea su primera definición al respecto, donde se argumenta que el pensamiento computacional es el proceso de reconocimiento de los aspectos computables en el mundo que nos rodea, y de aplicar las herramientas y técnicas de las Ciencias de la Computación para comprender y razonar sobre sistemas y procesos, tanto naturales como artificiales (Royal Society, 2012).

Otra aportación interesante ha sido la que realizaron Grover y Pea (2013), quienes proponen los principales conceptos que ellos piensan que han generado el mayor consenso, y que por lo tanto, deberían estar presentes en cualquier currículo educativo:

1. Abstracción y generalización de patrones (incluyendo modelos y simulaciones)
2. Procesamiento sistemático de la información
3. Sistemas de símbolos y representación
4. Noción algorítmica de control de flujo
5. Descomposición estructurada de problemas
6. Pensamiento iterativo, recursivo y paralelo
7. Lógica condicional
8. Limitadores de eficiencia y rendimiento

Copyright 2026

9. Depuración y detección sistemática de errores

Kafai y Burke (2014) amplían las definiciones anteriores con un concepto innovador, definiéndolo como un tipo de pensamiento basado en procesos ejecutados por una persona o una máquina utilizando métodos y modelos que permiten resolver problemas así como diseñar sistemas que por sí solos no podrían hacerlo.

El equipo de desarrollo de Scratch (Lamb y Johnson, 2011) el software educativo más utilizado en el mundo (Zhang y Nouri, 2019), aportó su visión definiendo el pensamiento computacional como un conjunto de conceptos, prácticas y perspectivas que está fundamentado en el ámbito de la informática. Para ellos, aprender a programar y compartir sus propias creaciones provoca en los estudiantes que se desarrollen como pensadores computacionales, aprendiendo conceptos básicos a la vez que son capaces de desarrollar estrategias de resolución de problemas, diseño y formas de colaboración (ScratchEd Team, 2015). En la misma línea, es visto como una metodología que implementa conceptos básicos de la computación que ayudan a resolver cualquier clase de problemas, forjar estrategias y ejecutar tareas de tal forma que nos permita afrontar los problemas con eficacia y posibilidades de éxito. (Olabe, Basogain y Basogain, 2015).

Para finalizar este apartado de definiciones, hilaremos la primera de las definiciones realizada por Wing (2006, 2008) con la postura de Bers (2017), la cual destaca que aunque la resolución de problemas tiene su importancia dentro de la definición más operacional del pensamiento computacional, le otorga especial relevancia al hecho de que el principal potencial es la posibilidad de expresar y crear ideas mientras programamos, argumentando que la programación, al igual que la escritura, es una forma de expresarse. Así, si con el lenguaje somos capaces de concretar múltiples y variadas representaciones, con los lenguajes de programación somos capaces también de expresarnos y crear productos (del Mar Sánchez-Vera, 2019).

Todos los conceptos y literatura comentadas en este capítulo están extraídos del trabajo de revisión sistemática sobre métodos de evaluación del pensamiento computacional (Ruiz y Bustamente, 2021).

Bibliografía

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom.* Routledge.

CSTA (2011). K–12 Computer Science Standards (Level 2) [Documento en línea]. Recuperado de http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K12_CSS.pdf

Copyright 2026

CSTA & ISTE (2011). Operational Definition of Computational Thinking for K–12 Education [Documento en línea]. Recuperado

dehttp://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

del Mar Sánchez-Vera, M. (2019). El pensamiento computacional en contextos educativos: una aproximacion desde la Tecnologia Educativa/Computational Thinking in Educational Environments: An Approach from Educational Technology/El pensament computacional en contextos educatius: una aproximado des de la Tecnologia Educativa. *Research in Education and Learning Innovation Archives (REALIA), (23),* 24-40.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42(1),* 38-43.

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

Olabe, X. B., Basogain, M. Á. O., & Basogain, J. C. O. (2015). Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje. *Revista de Educación a Distancia (RED),* (46)

Royal Society (Great Britain). (2012*). Shut down or restart?: The way forward for computing in UK schools.* Royal Society.

Ruiz Reinales, C., & Bustamante, J. C. Pensamiento computacional en educación infantil y primaria: una revisión sistemática.

ScratchEd Team [Portal Web] (2015). Computational Thinking webinars. Recuperado 2 de Junio de 2015, de http://scratched.gse.harvard.edu/content/1488

Wing, J. M. (2006). 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 A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7–14.

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141,* 103607.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Copyright 2026

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
GOBIERNO DE ARAGON

Copyright 2026

1.3 Componentes del Pensamiento Computacional

El pensamiento computacional es un término que sirve para aglutinar una serie de habilidades del pensamiento, imprescindibles para cualquier estudiante del siglo XXI (Kong, Lai y Sun, 2020). El Reino Unido, a través de su currículum en Ciencias de la Computación (DFE.U, 2013), y en el que nos vamos a apoyar principalmente como base para la realización de los ejercicios de este capítulo, detalla que el pensamiento computacional lo forman 6 conceptos (lógica, algoritmos, descomposición, patrones, abstracción, y evaluación sistemática) y 5 aproximaciones (experimentación, creación, depuración, perseverancia y colaboración). Los conceptos explicados serían los siguientes:

1. **Lógica**: el razonamiento lógico nos ayuda a explicar por qué algo sucede. Por esta razón, podemos utilizar el razonamiento lógico para determinar lo que queremos que haga un algoritmo de forma exacta.
2. **Algoritmos**: un algoritmo está escrito para ser entendido por humanos y es una secuencia de instrucciones o conjunto de reglas para solucionar un problema.
3. **Descomposición**: la capacidad que tenemos para poder fraccionar un problema en partes más pequeñas a través de las cuales podamos resolver problemas complejos y gestionar proyectos grandes.
4. **Patrones**: estamos rodeados de patrones. Ser capaces de identificar patrones nos permitirá hacer predicciones, crear reglas y resolver problemas más generales
5. **Abstracción**: la abstracción tiene que ver con simplificar las cosas; identificando qué es importante sin preocuparnos demasiado por lo anecdótico o irrelevante. La abstracción nos permite manejar la complejidad.
6. **Evaluación sistemática**: la evaluación es sistemática y rigurosa; tiene que ver con juzgar la calidad, la efectividad y la eficiencia de las soluciones, sistemas, productos y procesos. La evaluación comprueba que las soluciones aportadas resuelven el problema que nos planteábamos inicialmente no solo de una forma exitosa, sino además la más optimizada. Esto tiene mucho que ver con enseñar a nuestros alumnos que el error no tiene que generar frustración, sino que la programación nos da la maravillosa oportunidad de solucionar un error al estar en nuestras manos, en nuestra comprensión.

Y a continuación las aproximaciones:

Copyright 2026

1. **Experimentación** (‘Tinkering’): significa probar ... haciendo. En los más pequeños es la forma más natural de probar los juegos, de una forma espontánea mediante la exploración y el descubrimiento. Para los más mayores tiene que ver más con el concepto de ensayo-error-mejora.
2. **Creación** (‘Creating’): Programar es un proceso creativo que implica tanto originalidad como la generación de un producto final valioso.
3. **Depuración** (‘Debugging’): los errores en un algoritmo, programa o código y el proceso de encontrarlos y arreglarlos se denomina ‘debugging’. Algunos ‘bugs’ son errores lógicos, otros son errores sintácticos.
4. **Perseverancia** (‘Persevering’): programar puede ser una tarea ardua y compleja en la que también vamos a necesitar de nuestra predisposición a perseverar en momentos frustrantes.
5. **Colaboración** (‘Collaborating’): colaborar significa trabajar con otros para asegurar un mejor resultado. Para la creación de un producto final, en el mundo real no lo crea una sola persona, sino que son varias personas, incluso con diferentes perfiles, las que van creando su propia parte del código para cerrar la solución final. En un caso más sencillo, también se programa por parejas validando la expresión coloquial “cuatro ojos ven más que dos”.

Estados Unidos fue otro de los países pioneros en crear un curriculum Computer Science. A través de su Fundación Nacional para la Ciencia de los Estados Unidos (“National Science Foundation**”**) pone en marcha una serie de acciones formativas para la comunidad educativa, como por ejemplo la iniciativa “CS Principles” (Astrachan y Briggs, 2012), cuyo objetivo es fijar y transmitir las bases de las Ciencias de la Computación al alumnado de Bachillerato y primeros años de universidad. Se destacan las siguientes ideas principales relacionadas con procesos cognitivos y operacionales previa a la que fue su primera definición del pensamiento computacional:

1. Formular problemas de un modo que se haga posible utilizar un ordenador y otras máquinas en su resolución
2. Automatizar soluciones a través del pensamiento algorítmico (una serie de pasos discretos y ordenados)
3. Organizar lógicamente y analizar datos
4. Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más efectiva y eficiente de pasos y recursos
5. Representar datos a través de abstracciones tales como modelos y simulaciones
6. Generalizar y transferir este proceso de solución de problemas a una amplia variedad de situaciones

Otra aproximación distinta es la que realizó el grupo Google for Education (2014), detallando cuales son los 4 fases de procesos cognitivos que trabajan como una rueda circular :

1. Descomposición de un problema o tarea en pasos discretos

Copyright 2026

2. Reconocimiento de patrones (regularidades)
3. Generalización de dichos patrones y abstracción (descubrir las leyes o principios que causan dichos patrones)
4. Diseño algorítmico (desarrollar instrucciones precisas para resolver el problema y sus análogos)

En el mismo artículo se argumenta que el pensamiento computacional implica una serie de habilidades, incluyendo:

1. Formular problemas de una manera que nos permita usar un ordenador y otras herramientas para ayudar a resolverlos
2. Organizar y analizar de forma lógica un grupo de datos.
3. Representación de datos a través de abstracciones como modelos y simulaciones
4. Soluciones automatizadas a través del pensamiento algorítmico (una serie de pasos ordenados)
5. Identificar, analizar e implementar posibles soluciones con el objetivo de lograr la combinación más eficiente y efectiva de pasos y recursos
6. Generalizar y transferir este proceso de resolución de problemas a una amplia variedad de problemas

Lo realmente novedoso en su propuesta es que declaran que estas habilidades están respaldadas y mejoradas por una serie de disposiciones o actitudes que incluyen:

1. Confianza al desenvolverse bien en la complejidad
2. Persistencia al trabajar con problemas difíciles
3. Ser más tolerante con respecto a la ambigüedad
4. Capacidad de hacer frente a problemas abiertos (sin una solución concreta y evidente)
5. Capacidad de comunicarse y trabajar con otros para llegar a una meta-solución común

Y aún van más allá asegurando que los conceptos de pensamiento computacional no son solo los procesos mentales (por ejemplo, abstracción, diseño de algoritmos, descomposición, reconocimiento de patrones, etc.) sino que también producen resultados tangibles (por ejemplo, automatización, representación de datos, generalización de patrones, etc.) asociados con la resolución de problemas en computación. Estos se definen de la siguiente manera:

1. Abstracción: identificación y extracción de información relevante para definir las ideas principales
2. Diseño de algoritmos: creación de una serie ordenada de instrucciones para resolver problemas similares o para realizar una tarea
3. Automatización: Tener computadoras o máquinas para hacer tareas repetitivas.
4. Análisis de datos: dar sentido a los datos mediante la búsqueda de patrones o el desarrollo de ideas

Copyright 2026

5. Recopilación de datos: Recopilación de información
6. Representación de datos: representación y organización de datos en gráficos, cuadros, palabras o imágenes apropiados
7. Descomposición: desglosar datos, procesos o problemas en partes más pequeñas y manejables
8. Paralelización: procesamiento simultáneo de tareas más pequeñas de una tarea más grande para alcanzar de manera más eficiente un objetivo común
9. Generalización de patrones: creación de modelos, reglas, principios o teorías de patrones observados para probar los resultados previstos
10. Reconocimiento de patrones: observación de patrones, tendencias y regularidades en los datos
11. Simulación: Desarrollando un modelo para imitar procesos del mundo real

Bers (2017) amplía los elementos relacionados con el pensamiento computacional, concretándolo como un proceso expresivo organizado en siete habilidades de pensamiento computacional: algoritmos, modularidad, estructuras de control, representación, hardware / software, el proceso de diseño, y la depuración.

Para finalizar, partiendo de la propuesta original de Wing (2006), Olabe, Basogain y Basogain (2015) resumen las principales habilidades del pensamiento asociados al pensamiento computacional:

1. Reformular un problema a uno parecido que sepamos resolver por reducción, encuadrarlo, transformar, simular
2. Pensar Recursivamente
3. Procesar en Paralelo
4. Interpretar código como datos y datos como código
5. Generalizar análisis dimensional
6. Reconocer ventajas y desventajas del solapamiento
7. Reconocer coste y potencia de tratamiento indirecto y llamada a proceso
8. Juzgar un programa por simplicidad de diseño
9. Utilizar Abstracción y descomposición en un problema complejo o diseño de sistemas complejos
10. Elegir una correcta representación o modelo para hacer tratable el problema
11. Seguridad en utilizarlo, modificarlo en un problema complejo sin conocer cada detalle
12. Modularizar ante múltiples usuarios
13. Prevención, protección, recuperarse de escenario peor caso
14. Utilizar razonamiento heurístico para encontrar la solución
15. Planificar y aprender en presencia de incertidumbre
16. Buscar, buscar y buscar más
17. Utilizar muchos datos para acelerar la computación
18. Límite tiempo/espacio y memoria/potencia de procesado

Copyright 2026

Estas tres primeras páginas han servido como base teórica para comprender los procesos de pensamiento computacional con respecto a la resolución de problemas y procesos cognitivos asociados. Dada esta realidad, no es sorprendente que haya aparecido un interés en muchos países por introducir el pensamiento computacional como un conjunto de habilidades de resolución de problemas que los nuevos estudiantes deberían adquirir.

Todos los conceptos y literatura comentadas en este capítulo están extraídos del trabajo de revisión sistemática sobre métodos de evaluación del pensamiento computacional (Ruiz y Bustamente, 2021).

Bibliografía

Astrachan, O., & Briggs, A. (2012). The CS principles project. ACM Inroads, 3(2), 38-42.

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom.* Routledge.

DFE, U. (2013). National curriculum in England: computing programmes of study. Retrieved July, 16, 2014.

Google for Education. (2014). Exploring Computational Thinking. Recuperado 15 de octubre de 2018, de Google for Education website: https://edu.google.com/resources/programs/exploring-computational-thinking/

Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education, 151,* 103872.

Olabe, X. B., Basogain, M. Á. O., & Basogain, J. C. O. (2015). Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje. *Revista de Educación a Distancia (RED),* (46)

Ruiz Reinales, C., & Bustamante, J. C. Pensamiento computacional en educación infantil y primaria: una revisión sistemática.

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

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Copyright 2026

1.4 Saberes básicos, competencias específicas y criterios de evaluación del bloque de pensamiento computacional

Para favorecer la creación de la programación didáctica y la unidad didáctica correspondiente, realizamos una selección de los saberes básicos, competencias específicas y criterios de evaluación que se trabajarán de forma específica en este bloque de contenidos.

Relación con los saberes básicos:

- Estrategias, técnicas y marcos de resolución de problemas en diferentes contextos y sus fases.
- Emprendimiento, resiliencia, perseverancia y creatividad para abordar problemas desde una perspectiva interdisciplinar.
- Algorítmica y diagramas de flujo.

Relación con las competencias específicas:

- CE.PR.1. Abordar problemas tecnológicos con autonomía y actitud creativa, aplicando conocimientos interdisciplinares y trabajando de forma cooperativa y colaborativa, para diseñar y planificar soluciones a un problema o necesidad de forma eficaz, innovadora y sostenible.

Relación con los criterios de evaluación:

- CE.E.1. Abordar problemas tecnológicos con autonomía y actitud creativa, aplicando conocimientos interdisciplinares y trabajando de forma cooperativa y colaborativa, para

diseñar y planificar soluciones a un problema o necesidad de forma eficaz, innovadora y sostenible.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU



Copyright 2026

Situación de aprendizaje 1. ¿Cómo pensamos de forma computacional?

Tras haber realizado al alumnado una presentación sobre el término de pensamiento computacional y las habilidades del pensamiento relacionadas, se propone un primer ejercicio para que puedan ser capaces de reconocerlas a la hora de afrontar la resolución de un problema. En esta primera situación de aprendizaje, no se trata de resolver el problema, sino solo de que el alumnado sea capaz de visibilizar y detectar estas habilidades del pensamiento. Se busca que el alumno afronte mejor la resolución de problemas detectando todas las claves, que como no puede ser de otra manera se encontraran dentro del enunciado.

Vamos a basarnos en el modelo británico y en las siguientes habilidades del pensamiento relacionadas con el pensamiento computacional que ya han sido comentadas en el apartado 1.3: **razonamiento lógico, pensamiento algorítmico, descomposición, patrones, abstracción y evaluación sistemática**.

A continuación planteamos el enunciado del problema a resolver, teniendo en cuenta que sólo hay que identificar algunas de estas habilidades del pensamiento. En el capítulo posterior diseñaremos el diagrama de flujo para este problema, y en el siguiente lo programaremos. Estas serán nuestras 3 fases a la hora de resolver cualquier problema de programación:

1. Interpretación del enunciado
2. Diagrama de flujo
3. Codificación en lenguaje de programación

Enunciado

María es una alumna de 3º ESO que está muy interesada en simular el comportamiento de su calculadora y para ello quiere crear un programa que simule el funcionamiento de una calculadora básica donde introducimos 2 números y 4 operaciones posibles (+, -, /,) teniendo en cuenta que la división tiene un caso especial (el segundo número no puede ser un cero). Tras cada operación matemática exitosa, la calculadora volverá a su estado inicial esperando realizar una nueva*

operación.

El trabajo a realizar es partiendo del enunciado, identificar todas las habilidades del pensamiento computacional vistas en clase para empezar a poner la solución como paso previo a la creación del diagrama de flujo.

Solución:

Seguramente no hay una única solución posible, pero una posible solución sería la siguiente:

Descomposición. El problema general tiene unos cuantos subproblemas a resolver:

- Pedir el primer número (e incluso a la hora de programar, asegurarnos que lo introducido sea un número)
- Pedir el segundo número (e incluso a la hora de programar, asegurarnos que lo introducido sea un número)
- Pedir la operación a realizar (e incluso a la hora de programar asegurarnos que sea una de las 4 operaciones posibles: + - * /)
- En el caso de que la operación sea la división asegurarnos que el segundo número no sea un 0
- Realizar la operación deseada y sacar datos por pantalla

Patrones. Operaciones que se van a repetir dentro de nuestro código:

- Hay un gran patrón que se repite tras realizar la operación de manera exitosa: el programa tiene que volver al principio para realizar una nueva operación. En programación esto habrá que programarlo con un bucle infinito a no ser que introduzcamos una condición de salida
- Después de este gran patrón, podemos encontrar otros patrones como podrían ser un bloque de código para asegurarnos que los números sean números, que sea una de las 4 operaciones posibles y por último que en el caso de la división, el segundo número no sea un cero. En cualquiera de estos 3 caso podríamos incluir un bucle para asegurarnos de que se cumple lo que esperamos en la introducción de datos

Abstracción. Información no relevante para resolver el problema

- La información referente a María y su interés no es relevante a la hora de resolver el problema

Pensamiento **algorítmico**. Secuencia de órdenes para resolver el problema en un orden en concreto:

1. Pedir primero número y asegurarnos de que sea un número

2. Pedir segundo número y asegurarnos de que sea un número
3. Pedir una operación y asegurarnos de que sea una de las permitidas
4. Si es una suma, realizo la operación de suma de los dos números y saco resultado por pantalla
5. Si es una resta, realizo la operación de resta de los dos números y saco resultado por pantalla
6. Si es una multiplicación, realizo la operación de multiplicación de los dos números y saco resultado por pantalla
7. Si es una división, compruebo que el segundo número no sea un cero y en tal caso vuelvo a pedir segundo número hasta que sea distinto de cero. A continuación visualizo el resultado de la división
8. Vuelvo al comienzo para esperar una nueva operación

Como podemos suponer, no hay una única solución muchas veces para el mismo problema, y es genial que el alumnado pueda darse cuenta de ello e incluso se pueda debatir en la pizarra y que cada uno argumente pros y contras. En este caso, una posible solución alternativa sería que antes de pedir el segundo número podría pedir la operación y a continuación pedir el segundo número de tal forma que puedo controlar en ese paso si es un cero o es un número correcto para realizar la división.

Por último, en esta serie de pasos para resolver un problema, podremos a su vez descomponerla en otras órdenes tal y como hemos visto en la parte de descomposición, llegando a hacer esta secuencia de órdenes más detallada.

Evaluación. Antes de ponernos a dar los siguientes pasos, asegurarnos que no nos hemos dejado nada clave en la resolución del problema

Lógica. Habremos aplicado razonamiento lógico para la detección de entender el enunciado, sus requisitos, las salidas esperadas, y por último, los aspectos a controlar y tener en cuenta.

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU



Copyright 2026