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



Revision #4
Created 2022-10-08 07:13:28 CEST by Cristian Ruiz
Updated 2023-01-17 15:49:37 CET by Equipo CATEDU

Copyright 2026