PENSAMIENTO COMPUTACIONAL

Pensamiento algorítmico | Codificación | STEM

Qué involucra

Si queremos preparar a nuestros estudiantes para desarrollar el pensamiento computacional, debemos considerar habilidades como la capacidad de análisis, descomposición y abstracción, el diseño de algoritmos, la codificación y depuración de programas, y la validación de soluciones computacionales.

Computación desconectada

Permite a los estudiantes aprender algunos conceptos fundamentales como el diseño de algoritmos, antes del paso a usar un dispositivo, que involucra elementos adicionales para aprender. Por ejemplo, es posible hacer una actividad en la que los estudiantes diseñen una secuencia de pasos (algoritmo) para salir de un laberinto, donde dicha secuencia sea seguida por uno de los estudiantes antes de programar un dispositivo como el *micro:bit*.

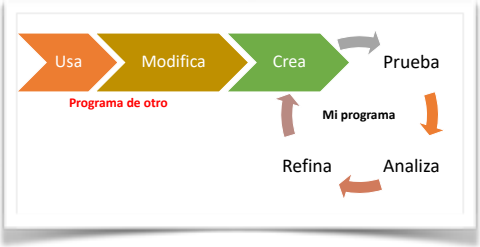
Qué es el pensamiento computacional

https://www.flickr.com/photos/

El pensamiento computacional (*PC*) se define como *el proceso mental de formular problemas y sus soluciones para representarlas de tal manera que puedan ser llevadas a cabo por un agente de procesamiento de información*. La computación ofrece oportunidades únicas para la solución de problemas tales como el procesamiento de grandes cantidades de datos, hacer tareas repetitivas de manera eficiente y la representación de fenómenos complejos a través de simulaciones.

Por qué es importante

La presencia casi ubicua de la computación junto con los problemas complejos a los que nos enfrentamos hoy en día han hecho se desarrollen una multitud de sub-disciplinas de las áreas STEM (*Science Technology, Engineering and Maths*) que utilizan la computación como su aproximación al quehacer disciplinario: Lingüística computacional, Química computacional, Neurociencias computacionales, Física computacional, etc. El PC es parte de las competencias del siglo XXI.



Usa, modifica, crea

Después de una etapa previa de computación desconectada, cuando se quiere pasar a la implementación, se recomienda una estrategia llamada ***usa-modifica-crea***.

Esta estrategia sugiere que para evitar sobrecargas cognitivas, primero hay que permitirles a los estudiantes usar las herramientas y representaciones computacionales para que se familiaricen con éstas y vean su valor. Por ejemplo, suministrar una solución computacional.

En una segunda instancia, los estudiantes modifican estas representaciones computacionales, de tal manera que comiencen a desarrollar una comprensión sobre cómo están construidas y cómo pueden ser adaptadas a otros contextos o problemas.

Finalmente, una vez los estudiantes hayan desarrollado los conocimientos necesarios para hacerlo, es posible pedirles que creen nuevos artefactos y soluciones desde cero.

Esto a su vez ha hecho que gobiernos y educadores de diferentes partes del mundo hagan un llamado para integrar el pensamiento computacional a través de todos los niveles educativos. De hecho, diferentes reportes y agencias internacionales han sugerido que para interactuar con un mundo computacional que nos rodea, el desarrollo del *PC* en todos los niveles educativos es ahora tan importante como aprender matemáticas o lenguaje.

Pensamiento computacional integrado a STEM

El *PC* es más que programar o transferir y probar un código en un dispositivo. El simple uso de editores de texto o de multimedia o uso de aplicativos en un teléfono inteligente no desarrolla *PC*. Las hojas de cálculo cuando se usan para procesar información desarrollan algunas habilidades de *PC*. Involucra también análisis, descomposición y abstracción de un problema, segmentación de tareas, creación de un algoritmo, antes entrar a programar. De hecho, se ha sugerido que éste se aprende mejor cuando se aplica a contextos relevantes para los estudiantes, tales como cursos de ciencias naturales, ciencias sociales, matemáticas y física.

Las áreas STEM se prestan particularmente bien para interactuar con el *PC*, ya que tienen unos fundamentos y prácticas que son similares entre sí: (1) definir los problemas y diseñar soluciones (2) crear y usar modelos computacionales para entender fenómenos o para diseñar y evaluar soluciones y (3) recolectar y analizar datos.

Referencias

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32-37.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. Computers in Human Behavior, 41, 51-61.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology, 25(1), 127-147