Introducción al pensamiento computacional

¿Qué es el pensamiento computacional?

Información basada en "Introduction to computational thinking" de BBC.

Los ordenadores pueden usarse para ayudarnos a **resolver problemas**, sin embargo, antes de que se pueda abordar un problema, es necesario **comprender** el problema en sí mismo y las formas en que podría resolverse.

El pensamiento computacional nos facilita **afrontar** un problema complejo, **comprender** cuál es el problema y **desarrollar** posibles estrategias para resolverlo. Luego podemos presentar estas soluciones de una manera tal que un ordenador, una persona o ambos puedan entenderlo.

Los cuatro pilares del pensamiento computacional

La **clasificación de la BBC** se basa en 4 pilares (Descomposición, reconocimiento de patrones, algoritmos y abstracción) haciendo una mención especial a la **evaluación** . En otros estudios podrás encontrar otras clasificaciones más detalladas. En el módulo 2 de este curso: actividades desenchufadas, se trabaja con estos cuatro pilares mas otros dos: Modelado y simulación, y Evaluación. Te explicamos aquí los fundamentales:

- **Descomposición** - Dividir un problema o sistema complejo en partes más pequeñas y manejables.
- **Reconocimiento de patrones** – Buscar similitudes entre problemas y dentro del problema.
- **Abstracción** – Centrarse sólo en la información importante, ignorando los detalles irrelevantes.
- **Algoritmos** - desarrollar una solución pasos a paso, o establecer reglas a seguir para resolver el problema.

Los cuatro pilares son **importantes**. Son como las patas de una mesa: si falta una pata, la mesa se caerá. La aplicación correcta de las cuatro técnicas ayudará a la hora de programar un ordenador.

El pensamiento computacional en la práctica

Un **problema complejo** es aquel que, a primera vista, no sabemos cómo resolver fácilmente.

El pensamiento computacional implica coger ese problema complejo y dividirlo en una serie de problemas pequeños y más manejables (**descomposición**). Luego, cada uno de estos problemas más pequeños se puede analizar individualmente, considerando cómo se han resuelto problemas similares anteriormente (**reconocimiento de patrones**) y centrándose solo en los detalles importantes, mientras se ignora la información irrelevante (**abstracción**). A continuación se pueden diseñar pasos o reglas simples para resolver cada uno de los problemas más pequeños (**algoritmo**).

Finalmente, estos pasos simples o reglas son usados para **programar un ordenador** o para ayudar a **resolver un problema complejo** de manera más eficaz.

Pensar de forma computacional

Pensar de forma computacional no es programar. Ni siquiera es pensar como un ordenador, ya que los ordenadores no pueden pensar.

En pocas palabras, la programación dice a un ordenador qué hacer y cómo hacerlo. El pensamiento computacional te permite **averiguar exactamente qué decir al ordenador que haga**. Es por tanto una **habilidad humana**, independiente del uso de ordenadores.

Por ejemplo, si quedas en reunirte con tus amigos en algún lugar en el que nunca has estado antes, probablemente planificarás tu ruta antes de salir de casa. Puedes considerar las rutas disponibles y qué ruta es la “mejor”. Esta podría ser la ruta más corta, la más rápida o la que pasa por tu tienda favorita en el camino. Luego seguirías las instrucciones paso a paso hasta llegar allí. En este caso, la parte de planificación es como el pensamiento computacional, y seguir las instrucciones es como programar.

Ser capaz de convertir un problema complejo en uno que podamos entender fácilmente es una **habilidad** que es **extremadamente útil**. De hecho, es una habilidad que ya tienes y que probablemente uses todos los días.

Por ejemplo, puede ser que necesites decidir qué hacer con tu grupo de amigos/as. Si a todos/as os gustan cosas diferentes, tendréis que **decidir**:

- ¿Qué podríamos hacer?
- ¿Dónde podríamos ir?
- ¿Quién quiere hacer qué?
- ¿Qué hemos hecho en el pasado que fue un éxito?
- ¿Con cuánto dinero contamos y cuánto cuesta cada opción posible?
- ¿Qué previsión de tiempo dan?
- ¿De cuánto tiempo disponemos?

Con toda esta **información**, tú y tus amigos/as podréis decidir más fácilmente a dónde ir y qué hacer y que la mayoría esté **conforme**. También podrías usar un ordenador para ayudarte a recopilar y analizar los datos para idear **la mejor solución** al problema, tanto ahora como si volviera a surgir en el futuro.

Otro ejemplo puede ser al jugar a un **videojuego**. Dependiendo del juego, para completar un nivel necesitarás saber:

- ¿Qué artículos necesitas recoger?
- ¿Cómo puedes recogerlos?

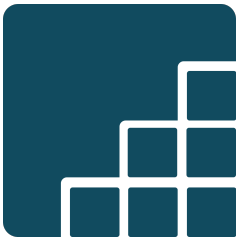
- ¿Cuánto tiempo tienes para recogerlos?
- ¿Dónde está la salida?
- ¿La mejor ruta para llegar a ella en el menor tiempo posible?
- ¿Qué tipo de enemigos hay?
- ¿Cuáles son sus puntos débiles?

A partir de estos detalles puedes elaborar una **estrategia** para completar el nivel de la manera más **eficiente**.

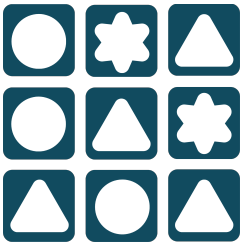
Si tuvieras que crear tu propio juego, estas **preguntas** son exactamente las que tendrías que plantearte y responder antes de programar el juego.

Los dos ejemplos anteriores son ejemplos de cómo se ha utilizado el pensamiento computacional para resolver un problema complejo:

- Cada problema complejo se ha dividido en partes más simples - **descomposición.**

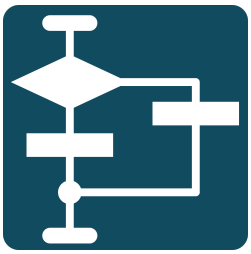


- Se utilizó el conocimiento de problemas similares - **reconocimiento de patrones**.



- Solo se centraron en los detalles relevantes - **abstracción.**

- Para elaborar un plan de acción paso a paso -**algoritmo.**



> Si quieres **saber más,** y practicar tu inglés, mira este vídeo:

https://www.youtube.com/embed/Nc-V948dXWI

> Computational Thinking and problem solving, by Miles Berry Europe Code Week

---

Revision #12
Created 19 June 2023 09:52:06 by Elena López de Arroyabe
Updated 26 June 2023 08:54:47 by Elena López de Arroyabe