Introducción al Pensamiento Computacional

Las tres dimensiones del Pensamiento Computacional

- ***Conceptos*** *computacionales*, que son aquellos que emplean los diseñadores en el trabajo de programación.
- ***Prácticas*** *computacionales*, que son las que desarrollan a medida que programan.
- ***Perspectivas*** *computacionales*, que son las que los diseñadores construyen sobre el mundo que los rodea y sobre sí mismos.

A partir de esta visión, pueden definirse las siguientes maneras de valorar los aprendizajes generados por los jóvenes:

CONCEPTOS	PRÁCTICAS	PERSPECTIVAS
QUE SE EMPLEAN AL PROGRAMAR	QUE SE DESARROLLAN AL PROGRAMAR	QUE SE FORMAN SOBRE EL MUNDO Y SOBRE SÍ MISMOS
• **algoritmos** • **programas** • **secuencias** • **ciclos** • **operadores** • **eventos** • **paralelismo** • **datos y paradigmas**	• **modelado** • **reutilización** • **abstracción** • **evaluación** • **modularización** • **documentación y metodología de desarrollo de proyectos**	• **expresión** • **creatividad** • **trabajo e interacción con pares** • **pensamiento crítico sobre las tecnologías, sus usos y contextos**

Fines principales por los que los estudiantes deberían **desarrollar el Pensamiento Computacional**:

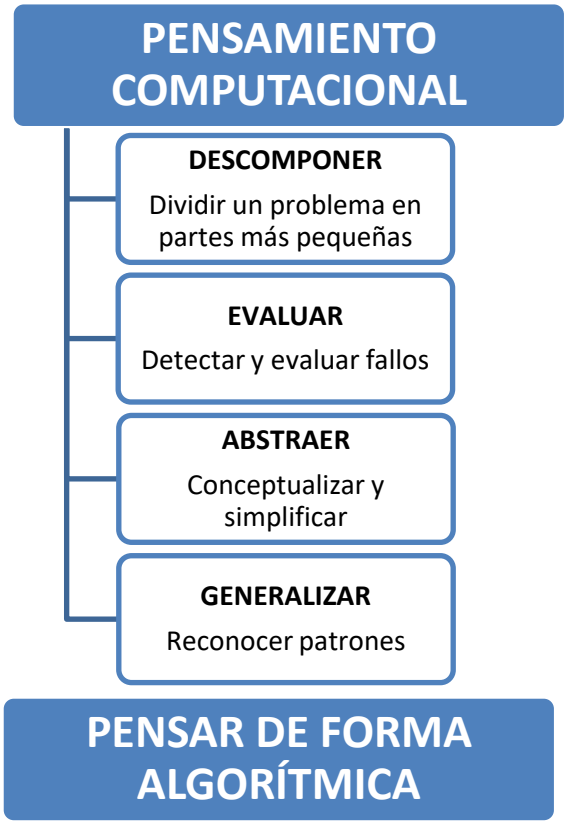
- mejorar la habilidad para solucionar problemas y, además, que estas soluciones puedan traducirse para ser ejecutadas por una computadora.
- que dejen de ser consumidores pasivos de tecnología y que pasen a tener una relación más estrecha, activa y fructífera con recursos y herramientas tecnológicas, con el objetivo de comprender mejor el mundo que los rodea.
- que los nuevos lenguajes informáticos les permitan ampliar sus posibilidades de expresión y poner en juego toda su capacidad creativa.
- que aquellos que ya tengan un principio de vocación por la ingeniería o las ciencias de la computación tengan experiencias de aprendizaje tempranas que les permitan empezar a involucrarse en lo que va a ser parte de su futuro.

Capacidades asociadas al pensamiento computacional

Existe cierto consenso, de creciente relevancia, en relación con que el pensamiento computacional va más allá de programar o codificar e implica todo un proceso previo, de formulación y análisis del problema, como así también de diseño y de evaluación de soluciones.

En este sentido, entendiendo el pensamiento computacional como un proceso cognitivo que implica un razonamiento lógico aplicado a la resolución de problemas, sus elementos clave son los siguientes:

- Capacidad de pensar de forma algorítmica.
- Capacidad de pensar en términos de descomposición.
- Capacidad de pensar en generalizaciones, identificando y haciendo uso de patrones.
- Capacidad de pensar en términos abstractos y elección de buenas representaciones.
- Capacidad de pensar en términos de evaluación.



Los elementos clave del pensamiento computacional involucran el desarrollo de un razonamiento lógico. Este permite que los estudiantes puedan dar sentido a las cosas, lo que sucede por medio del análisis y la comprobación de los hechos a través de un pensamiento claro, detallado y preciso. De esta manera, los estudiantes toman sus propios conocimientos y modelos internos para hacer y verificar predicciones y así obtener conclusiones. El razonamiento lógico es la aplicación del pensamiento computacional para resolver problemas.

Por ejemplo, en actividades relacionadas con el diseño y la tecnología se aplica el razonamiento lógico —en particular, en el diseño de un objeto, la determinación de su forma y funcionalidad, la elección de los materiales a utilizar y los pasos de fabricación—. La ***descomposición*** se aplica al dividir un proyecto o proceso en diferentes partes según un criterio en particular. La ***generalización*** sucede cuando, dada una situación particular, el estudiante es capaz de establecer nuevas conexiones y pensar sobre otras aplicaciones o usos en otros contextos. Al diseñar, se están realizando tareas de predicción dado que se suponen comportamientos. El poder simular comportamientos implica evaluar situaciones futuras y así mejorar el diseño asegurando predicciones correctas. Cuando se escribe una solución a un problema, para que pueda ser implementada por una persona o una computadora, hay tareas relacionadas con la búsqueda y corrección de errores: en esa instancia, también existe un proceso de razonamiento.

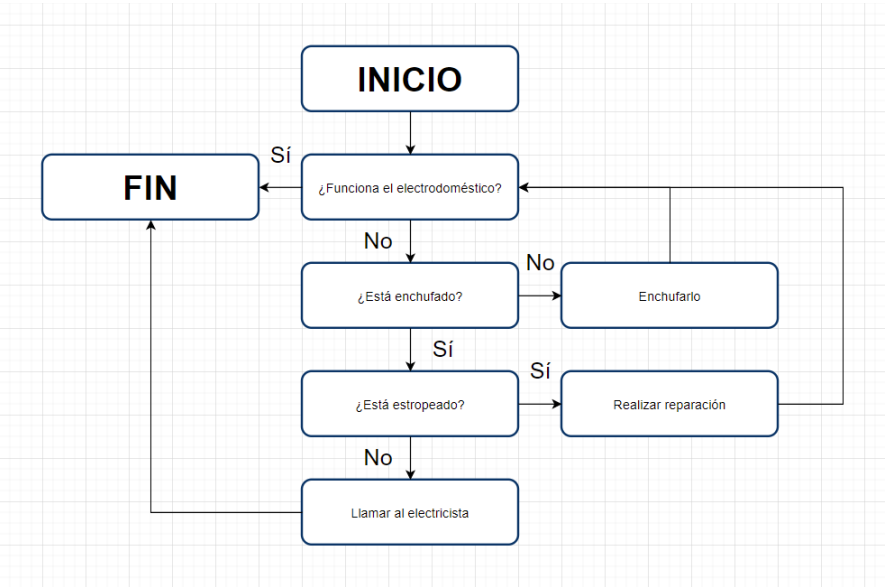
El razonamiento lógico ayuda a explicar por qué sucede algo. Esto es muy importante en ciencias de la computación debido a que las computadoras son predecibles en sus resultados, solo realizan aquello para lo cual están programadas. En virtud de esta cualidad, se utiliza el razonamiento lógico para programarlas y así describir con exactitud las tareas a realizar. Entendido de esta manera, el razonamiento lógico equivale a explicar por qué algo es así.

A continuación, se desarrollan, de manera detallada, cada una de las capacidades básicas que componen el pensamiento computacional:

- **Capacidad de pensar de forma algorítmica**

Un algoritmo, en principio, es un objeto de comunicación compuesto por un conjunto finito de instrucciones que especifican una secuencia de operaciones concretas por realizar en un orden determinado para resolver un problema. El pensamiento algorítmico es una actividad cognitiva asociada a la resolución de problemas, a su especificación y a la comunicación de su solución.

Los siguientes son ejemplos de algoritmos que expresan soluciones a distintos problemas. Nótese que el algoritmo puede expresarse de distintas formas, en estos casos como un gráfico (diagrama de flujo) y como un texto con órdenes:



ALGORITMO PARA PREPARAR UNA SOPA INSTANEA EN EL MICROONDAS

1. Inicio
2. Abrir el envase de sopa
3. Añadir agua a la sopa
4. Introducir en el microondas
5. Establecer el temporizador y la potencia del microondas:
   3 minutos a 800 W
6. Fin

En general, el pensamiento algorítmico se aplica cuando existen problemas semejantes que tienen que ser resueltos con periodicidad, entonces se analizan en conjunto y se desarrolla una solución general que pueda aplicarse cada vez que ocurra el problema.

En nuestra vida cotidiana, recurrimos de manera constante a algoritmos para solucionar problemas y así realizar cosas. Por ejemplo: para resolver una cuenta y obtener un valor, para cocinar una comida o para realizar una extracción de dinero en un cajero automático. En todos los casos mencionados, seguimos una y otra vez un conjunto ordenado de pasos que están almacenados en nuestro cerebro o en algún soporte externo (como en el caso de la receta de cocina que puede ser tomada de un libro o visualizada en YouTube o similar).

Ejemplos de situaciones donde están presentes algoritmos:

- Cuando un cocinero escribe una receta para realizar un plato, está creando un algoritmo dado que otros pueden seguir los pasos y así reproducirla.
- Cuando un amigo anota las instrucciones para llegar a su casa, está especificando una secuencia de pasos (un algoritmo) para que otra persona lo pueda ubicar.
- Cuando un profesor proporciona un conjunto de instrucciones para llevar a cabo un experimento, está especificando un algoritmo, que es seguido por los estudiantes y así obtienen datos para su análisis y aprendizaje.

Podemos definir el pensamiento algorítmico como la capacidad de pensar en términos de secuencias y reglas que sirven para resolver problemas. Es un conocimiento básico que las personas desarrollan cuando aprenden a escribir sus propios programas de computadora, que no son otra cosa que algoritmos traducidos a instrucciones expresadas en un lenguaje que una computadora pueda comprender y ejecutar (por ejemplo, lenguajes informáticos como Scratch, Python, JavaScript, C, Java, etc.).

Cuando se habla de software, se hace referencia directa a los datos digitales y a los programas de computadora. ***Lev Manovich***, autor del libro "*El software toma el mando*", realiza una descripción de su poder y su presencia en la sociedad actual cuando dice: «*El software se ha vuelto nuestra interfaz con el mundo, con otras personas, con nuestra memoria e imaginación; un lenguaje universal mediante el cual habla el mundo, un motor universal mediante el cual*

funciona el mundo» (*Manovich*, 2013) y, por otro lado, nos advierte: «*El software juega, hoy en día, un papel crucial en la confección de elementos materiales y de estructuras inmateriales que, aunados, constituyen la "cultura"*».

En las ciencias de la computación, el trabajo de los científicos se concentra en encontrar los algoritmos más eficientes. Es decir, aquellos que resuelven un problema involucrando los menores recursos posibles (memoria, comunicaciones, tiempo de procesamiento, etc.) de la manera más efectiva al dar la respuesta correcta o la más cercana a ella.

- **Capacidad de pensar en términos de descomposición**

Según el ***Diccionario de la lengua española de la Real Academia Española***, **descomponer** significa *'separar las diversas partes que forman un compuesto'*, entre otras acepciones. En los siguientes gráficos, pueden observarse distintas situaciones en las cuales sucede un proceso de descomposición: una es la descomposición numérica —que ayuda a que los estudiantes entiendan la disposición y las relaciones entre los dígitos de un número— y otra, la descomposición de la luz con un prisma —para obtener el espectro que representa al arcoíris—.

En la escuela es habitual que nos encontremos con actividades de cierta complejidad que tienen que ser descompuestas en tareas más simples para que puedan llevarse a cabo.

Algunos ejemplos:

- Preparar la fiesta de fin de año. Tareas: armado del programa de actividades, difusión del evento, preparación y ensayo de las actividades artísticas, preparación y atención del servicio de cantina, preparación y limpieza del salón de actos, etc.
- Desarrollar un plan que permita asignarle el rótulo de ecológica a una institución educativa. Tareas: definir una estrategia para almacenar y procesar la basura, establecer un plan para reducir el consumo de electricidad, planificar una campaña de concienciación en la comunidad.
- Planificar la publicación de una revista institucional. Tareas: identificar temas y secciones, asignar roles de los colaboradores y responsabilidades asociadas, planificar tiempos y recursos necesarios a los efectos de llevar adelante el proyecto.

En el ámbito informático, al proceso de dividir un problema en partes más pequeñas o sencillas (y, por ende, más manejables) se lo conoce como **descomposición**. Descomponer un problema es una tarea que facilita su resolución debido a que reduce su complejidad.

Para descomponer hay que pensar en términos de partes y componentes, donde cada pieza se debe comprender, evaluar y solucionar por separado. Por otro lado, la solución asociada a cada una de las partes puede encargarse a una persona o a un equipo de trabajo, con lo cual esto permite poder resolver problemas complejos en tiempos más acotados.

En esencia, toda descomposición implica:

- Identificación de las partes de algo
- División de algo en partes más pequeñas

Por ejemplo, una actividad en la que se suele aplicar la técnica de la descomposición es la que los detectives policiales realizan ante un crimen a los efectos de esclarecerlo ya que es un proceso complejo, las tareas asociadas son muchas y, en general, son llevadas a cabo por un equipo de agentes. En general, la técnica policial ante un hecho delictivo divide su quehacer relacionado con la investigación a partir de una serie de preguntas clave (las cuales se podrían ver como subproblemas): ¿qué tipo de crimen se ha cometido?, ¿cuándo se llevó a cabo?, ¿dónde se cometió?, ¿qué evidencias hay?, ¿hubo testigos?, ¿qué observaron?, ¿quién era la víctima?, ¿hay semejanza con otros crímenes registrados? Luego de obtener las respuestas, el equipo policial puede evaluarlas en su conjunto y empezar a pensar en líneas de investigación a partir de indicios concretos. Como se observa en el ejemplo, la descomposición del problema y la generación de tareas de menor complejidad permiten empezar una resolución efectiva y lógica del problema.

También se puede producir un proceso de descomposición en un proyecto de trabajo: por ejemplo, en la creación de un videojuego en torno al tema de cuidados en el uso de redes sociales. Para facilitar la solución del problema, el equipo responsable realiza un primer análisis y evaluación y luego decide usar esta técnica. Se identifican una serie de partes principales y cada una se asigna a un equipo técnico:

- Diseño de la narrativa, lógica y reglas asociadas al juego.
- Diseño de los personajes, objetos y escenarios.
- Programación del videojuego.
- Depuración y búsqueda de errores del producto final (videojuego).

Luego de analizar la situación y de que el gran problema haya podido dividirse en partes de menor complejidad, se asigna la resolución de cada una a una persona o grupo. A lo largo del desarrollo se ensamblan los resultados de cada equipo en pos de dar solución a la situación problema.

- **Capacidad de pensar en generalizaciones, identificando y haciendo uso de patrones**

Para tratar de pensar el concepto de reconocimiento de patrones, supongamos la siguiente situación: una persona está a cargo de un campo con animales y existe comida suficiente y especial para cada uno. Las instrucciones para que un operario alimente a los animales son simples:

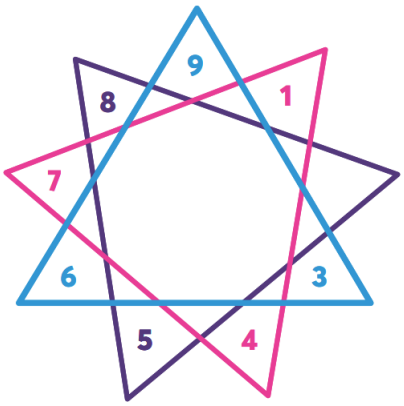
- Para alimentar al perro, poner la comida del perro en el plato del perro.
- Para alimentar al pollo, poner la comida del pollo en el plato del pollo.
- Para alimentar al conejo, poner la comida del conejo en el plato del conejo.

Como podemos observar, existe una estructura subyacente común en cada una de las instrucciones anteriores, es decir un patrón, que podría expresarse de la siguiente manera:

Para alimentar al <*animal*>, poner la comida del <*animal*> en el plato del <*animal*>.

Al detectar un patrón en las instrucciones de alimentación de los animales, se pudo realizar luego un proceso de generalización que simplificó el protocolo de alimentación a partir de una única instrucción genérica.

Ahora vamos a otro caso: observa la siguiente figura y trata de detectar cuál es el patrón que generó el dibujo:



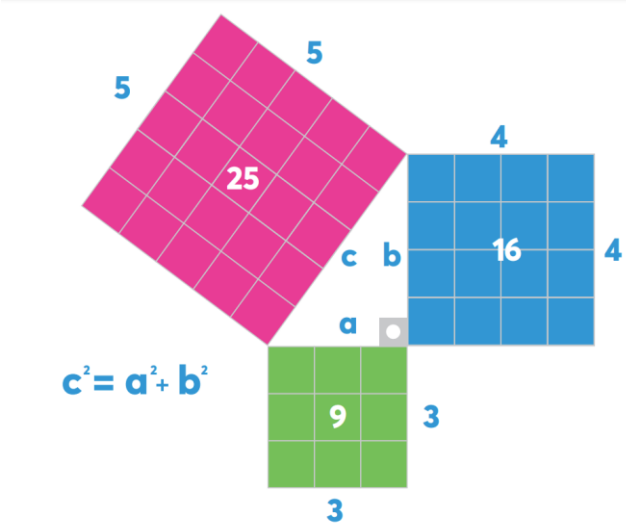
Podríamos decir que, a partir de un triángulo equilátero al que se hizo girar dos veces (*n* grados en cada vez), se obtuvo la imagen anterior, por lo cual el patrón es la rotación del triángulo original *n* grados.

En la práctica, al **descomponer** un problema complejo se suelen encontrar **patrones** entre los subproblemas que fueron definidos. Los patrones se expresan como características compartidas entre los distintos problemas de menor complejidad. Cuando se los detecta, es posible trabajarlos de manera conjunta y así simplificar la tarea de resolución. Los problemas son más fáciles de resolver cuando comparten patrones, debido a que es posible usar soluciones ya diseñadas con anterioridad y aplicarlas a los subproblemas.

Ejemplos de patrones:

- Se buscan patrones respecto de la cantidad de personas y el tiempo que necesitan al ingresar en un banco y elegir una cola para realizar un pago.
- Los conductores buscan patrones de comportamiento en el tráfico para decidir cuándo cambiar de carril e ir más rápido.
- Los inversores buscan patrones en los precios de las acciones para decidir cuándo comprar y cuándo vender.

La generalización es una tarea relacionada con la identificación de patrones, semejanzas y conexiones, para luego realizar una explotación de las características. Se presenta como un método rápido para resolver nuevos problemas sobre la base de clasificarlos como versiones de viejos problemas con solución. Es decir, que se hace uso de la experiencia. En general, la generalización es una tarea que se realiza después de la descomposición.



La fórmula matemática del teorema de Pitágoras es un ejemplo de generalización.

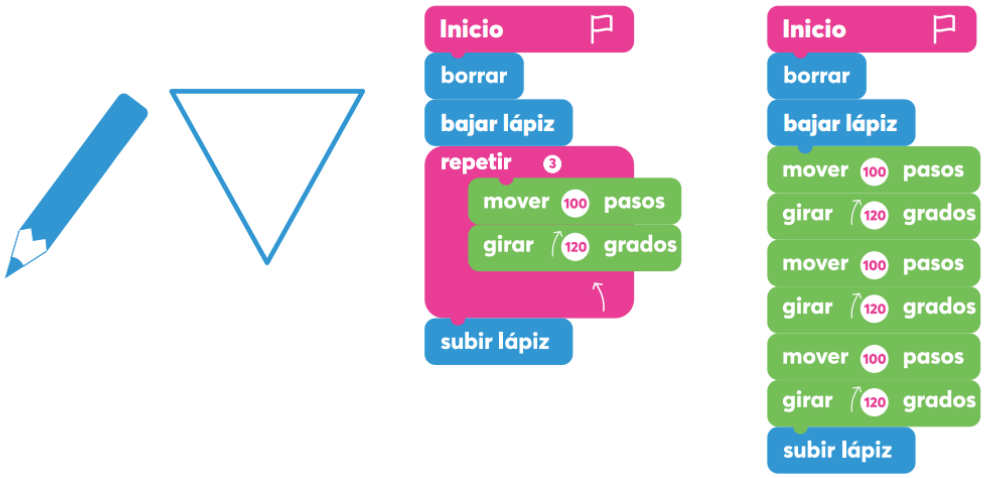
La descomposición y la generalización están relacionadas. Mientras que la descomposición implica dividir un problema en partes más pequeñas, la generalización se basa en combinar esas partes. Así, cuando se realiza la generalización a partir de las partes individuales, no se vuelve a unir el problema de la misma manera en que estaba originalmente. El objetivo de la generalización es observar las partes descompuestas y encontrar formas de facilitar el desarrollo de una solución.

Otro ejemplo de generalización aplicada al desarrollo de programas es el siguiente: supongamos que un estudiante tiene que realizar un algoritmo y luego un programa que dibuje un triángulo equilátero.

Una manera de plantear una solución es el algoritmo (escrito en código de bloques del lenguaje ***Scratch***) que se observa en la imagen siguiente a la derecha. Después de iniciar la tarea, se toma una hoja de trabajo limpia y la herramienta **Lápiz**. Se desplaza la herramienta 100 unidades de longitud (en nuestro caso, son pasos y cada uno tiene una medida fija

determinada) y luego se la hace girar 120 grados en el sentido horario sobre su eje central. De esta manera, se ha dibujado un lado y todo está preparado para dibujar el siguiente. Como se ve en la imagen, luego se repiten dos veces los pares de instrucciones mover y girar. Con eso se completan los tres lados y el algoritmo finaliza con la instrucción de levantar el lápiz.

A la izquierda del algoritmo anterior puede observarse otra solución al problema (y más eficiente en términos de redacción de la solución), donde se ha identificado que el par de instrucciones mover y girar son en sí un patrón y, en este caso, se repiten (una vez por cada lado del triángulo). Para hacer más eficiente la solución, cometer menos errores de escritura y comunicar mejor la tarea se utiliza la instrucción repetir 3 y, en cada ciclo de repetición, se solicita el dibujo de una cara. En este caso, a partir de detectar el patrón lado, se ha podido hacer una generalización simple y así obtener un algoritmo más legible, optimizado en su escritura y seguramente con menos errores.



Los científicos de las ciencias de la computación desean resolver problemas de forma rápida y eficiente, para lo cual suelen reutilizar métodos efectivos creados con anterioridad. Los patrones son elementos comunes en problemas, se identifican a efectos de crear módulos estándar para que se puedan reutilizar en distintos programas para solucionar diferentes problemas. En general, a partir de un patrón detectado se construye un módulo común una sola vez. En términos asociados a los lenguajes de programación, a estos módulos se los conoce como ***procedimientos*** o ***funciones***.

- **Capacidad de pensar en términos abstractos y la elección de representaciones adecuadas**

En principio, la abstracción es un proceso por el cual se simplifica el entendimiento de una situación. Se basa en identificar lo que es importante de algo, sin preocuparse por los detalles, así, de esta manera, se puede administrar la complejidad de un problema. **Todo proceso de abstracción da como resultado** la construcción de **una vista simplificada**, que es **la idea principal de algo**.

El ser humano ha utilizado la abstracción de muchas maneras al retratarse a sí mismo. Desde la forma más icónica y carente de detalles, hasta la representación más fidedigna, todas las formas de representación del ser humano de sí mismo no son sino **abstracciones**.

Jeannette Wing (2006) indica: «*en el proceso de abstracción, decidir qué detalles debemos resaltar y qué detalles podemos ignorar, subyace en el pensamiento computacional*». Teniendo en cuenta que la idea principal de todo no está en los detalles, sino en los grandes rasgos, al abstraer algo se obtiene como resultado una vista más comprensible. La base del trabajo de abstracción consiste en la observación, la detección y la reducción de detalles, al obviar cosas no relevantes.

Diferentes puntos de vista ofrecen diferentes abstracciones. Por ejemplo, los mapamundis solo muestran las características de importancia para un observador particular. Un mapa para asistir a viajantes muestra las rutas, las distancias y las ciudades de una región. Un mapa de distribución de cultivos, realizado para inversores, realza sobre un área específica las áreas asignadas a la producción de cada cultivo con un color asociado e indica el rendimiento. Pero un mapa de esa misma región destinado a ecólogos podría indicar la distribución de cultivos y los riesgos asociados en relación con el uso de agroquímicos.

Como se ha visto, si bien sobre un mismo problema se pueden generar distintas abstracciones, lo importante para avanzar en su solución es realizar la elección de una buena representación. Es necesario tener en cuenta que las diferentes representaciones sobre un mismo problema pueden hacer las cosas más fáciles en algunos casos y más difíciles en otros.

Un proceso de abstracción permite definir la esencia de algo. En la educación, la abstracción suele ser utilizada porque permite mostrar con ejemplos simples sistemas complejos para su estudio y comprensión. Por eso, se aplica en infografías, maquetas, diseños conceptuales, etc.

Son ejemplos de abstracción los mencionados a continuación:

- Una hoja de planificación diaria utiliza la abstracción para representar una semana en términos de días y horas, lo cual la convierte en un objeto útil para organizar el tiempo personal.
- Un mapamundi es una abstracción del planeta. Su composición, en términos de longitud y latitud, ayuda a describir la ubicación y la geografía de un lugar concreto.
- En matemáticas, se escriben fórmulas generales en términos de variables en lugar de utilizar números. Esto sucede para que puedan ser utilizadas en problemas que involucran diferentes valores.
- En música, los pentagramas son soportes de escritura de piezas musicales y se los puede entender como abstracciones que representan sonidos en el tiempo.

- **Capacidad de pensar en términos de evaluación**

Una tarea de evaluación implica hacer juicios sobre algo, de una manera sistemática y sobre la base de criterios previamente definidos para que esta sea lo más objetiva posible. La evaluación es algo que se realiza diariamente: de manera regular hacemos juicios sobre qué hacer, pensando en función de una serie de factores que son parte de un contexto. Por ejemplo, cuando alguien va a comprar un automóvil, en general, antes de hacerlo piensa cuán confortable es, en qué medida es económico, si es caro o barato su mantenimiento, etc., es decir, evalúa aspectos esenciales antes de decidir si lo adquiere o no.

También evaluamos algo en estas situaciones:

- Cuando cocinamos, probamos nuestros platos para poder ajustar su sabor y verificar el estado de cocción.
- Cuando estamos en un nivel de un videojuego y pretendemos pasar al siguiente, probamos y evaluamos distintas estrategias de acción que nos permitan seguir adelante.
- Cuando, como estudiantes, recibimos la devolución de un docente de un trabajo, revisamos las notas añadidas para saber qué estamos haciendo bien y qué cosas aún nos falta por aprender o hemos comprendido de forma errónea.

En el marco del pensamiento computacional, una vez que se ha diseñado una solución, es necesario asegurarse de que sea adecuada para su propósito. La evaluación es el proceso que se aplica a una respuesta en pos de asegurar que esta responde a los requerimientos de diseño y que, además, funciona correctamente, sin errores. Cuando se trabaja con programas de computadora, la evaluación es una tarea sistemática y rigurosa ya que se está juzgando su efectividad y eficiencia.

Cuando una solución se traduce a un algoritmo, deberían verificarse una serie de elementos que son parte de una prueba de evaluación:

- Que se entienda fácilmente: ¿está completamente descompuesto?
- Que sea eficaz: ¿resuelve el problema?
- Que sea eficiente: ¿resuelve el problema, haciendo el mejor uso posible de los recursos disponibles?
- Que cumpla con los criterios de diseño especificados: ¿está en el marco de los requerimientos?

Una vez que un algoritmo supera el proceso de evaluación y cumple con los cuatro criterios anteriores, se entiende que se está ante una solución correcta; por tanto, se puede avanzar a la etapa de programación en algún lenguaje informático. La mayoría de las veces, programar sin antes evaluar dificulta la tarea de programar, lo cual puede llevar a más errores y, por ende, a mayores costos y tiempos en el proyecto.

¿Cómo evaluamos nuestra solución? Como hay muchos métodos desarrollados, en principio y a modo de ejemplo, se podría comenzar la tarea con la orientación de las siguientes tres preguntas:

- ¿Se comprende, de manera completa, cómo se ha resuelto el problema? Es decir, ¿la solución construida está explicitada en su totalidad? Si todavía algo no se sabe, aún no se tiene una solución completa. En caso de que no se sepa claramente cómo hacer algo para resolver el problema, hay que volver a la etapa anterior. Es necesario verificar que todo se haya descompuesto correctamente y que cada parte tenga una solución.
- ¿La solución cubre todas las partes del problema? Aquí se busca validar que la solución propuesta y desarrollada satisfaga plenamente el objetivo a cumplir y que, además, lo haga en el marco de las restricciones impuestas.
- ¿La solución optimiza la repetición de tareas? En caso de respuesta negativa, se debe preguntar si existe alguna forma de reducir tal repetición, para lo cual hay que regresar a la etapa de desarrollo de la solución y eliminar las repeticiones innecesarias.

En ciencias de la computación, se conoce como depuración o ***debbuging*** el proceso de búsqueda y corrección de errores en un programa. Es una tarea de cierta complejidad y tiempo debido a que primero hay que entender el problema y la solución elegida para luego trabajar analizando las instrucciones que la conforman.

Para corregir errores o ***bugs*** en programas de computadora, en principio, se sugiere seguir una secuencia de tres pasos:

- Predecir lo que debe suceder
- Averiguar exactamente qué sucede
- Trabajar nuevamente dónde algo salió mal hasta arreglarlo

En un programa de computadora suele haber dos tipos de errores: los de codificación (se escribió mal alguna instrucción; se produjo un error de sintaxis) y los de lógica (el plan asociado a la solución está equivocado).

Para finalizar con el tema de la evaluación, la historia del primer ***debugging*** fue en la era de las primeras computadoras analógicas y se trató, específicamente, de la ***Mark III*** (construida por la Universidad de Harvard para estudios militares). En esa computadora, la ingeniera contraalmirante de la Marina de los Estados Unidos *Grace Hopper* encontró, en 1947, el primer ***bug*** o error en la historia de la informática. Resulta que la máquina no funcionaba bien y, haciendo una revisión física, halló que en el relé 70 de un panel había una polilla que impedía todo contacto eléctrico. De ahí en adelante se asoció el error informático con la palabra ***bug*** (en inglés, bicho).

Técnicas asociadas con el pensamiento computacional

Más allá de las capacidades que configuran el pensamiento computacional, de manera complementaria existen una serie de técnicas asociadas a su desarrollo cuya función es ordenar y favorecer el trabajo cognitivo. Son las siguientes:

- **Reflexión**: se define como la habilidad de llevar a cabo juicios argumentados sobre situaciones que tengan cierto grado de complejidad. Desde la perspectiva de la informática, las técnicas que promueven la reflexión utilizan criterios sobre la comprensión de las especificaciones de los productos y las necesidades de los usuarios.
- **Análisis**: para analizar una situación problemática, se acude a una serie de herramientas auxiliares como dividir en partes de menor complejidad los componentes de un problema (descomponer), reducir aspectos de complejidad (abstraer), identificar los procesos y buscar elementos patrones (generalizar). Se trata de utilizar la capacidad analítica, apelando a la lógica, para comprender plenamente el problema y así avanzar en su solución.
- **Diseño**: es una técnica que se aplica al proceso de desarrollo de una solución efectiva y eficiente de un problema. El diseño implica creatividad debido a que es necesario ver desde distintos puntos de vista un mismo problema, satisfacer las necesidades o deseos y adaptarse a un contexto que la hace posible y sustentable.
- **Programación**: todo diseño de una solución a un problema debe ser programado para que una computadora pueda automatizar el proceso. A esta traducción de la solución, realizada utilizando los recursos de un lenguaje computacional, se le debe aplicar un proceso de revisión técnica funcional que la evalúe en función de garantizar su funcionamiento correcto en todas las condiciones de trabajo. Asociada al proceso, está la operación de depuración, entendida como la actividad de corrección de errores que solo afectan a la programación, debido a que no provienen de etapas previas.
- **Aplicación**: se basa en la adopción de soluciones existentes para satisfacer las necesidades de otro contexto. Esta habilidad implica el uso de la capacidad de generalizar debido a que se deben identificar patrones y realizar conexiones a efectos de adoptar lo preexistente.

Por otro lado, existen una serie de actitudes adicionales que ayudan a promover el desarrollo del pensamiento computacional. Estas están en relación con aspectos de quienes resuelven los problemas ya que les permitirán enfrentar tales situaciones de una manera más natural, reduciendo la tendencia a la frustración por resultados erróneos o no deseados.

Actitudes vinculadas al pensamiento computacional

- **Perseverancia**: Es una de las capacidades extras por desarrollar junto con el pensamiento computacional ya que estamos tratando con problemas y, utilizando nuestra creatividad, debemos pensar en soluciones. Así, perseverar se entiende como la actitud de continuar sin ceder, ser resistente y tenaz en una tarea.
  En muchas situaciones, al resolver problemas complejos, se necesita voluntad para superar, en ciertas ocasiones, las ganas de abandonar un proyecto y dejarlo inconcluso. La capacidad de perseverar es común a los profesionales de las ciencias de la computación dado que es frecuente que el trabajo intelectual se vea acompañado de fallas y errores que deriven en situaciones de frustración sino se tiene la capacidad mencionada. La tolerancia a los errores, su análisis y comprensión son parte de la disciplina que deben acompañar a todo profesional.

- **Experimentación**: Otra capacidad por desarrollar en los estudiantes tiene que ver con experimentar, lo que sucede frecuentemente a partir de probar cosas (en inglés, ***tinkering***). Es una forma de intervención, asociada a no conformarse con lo establecido y ya hecho, sino con querer dar un valor agregado a las cosas a partir de una mirada y un hacer crítico que genera nuevas versiones de los objetos en estudio.

  Desde la infancia, realizamos actividades de ***tinkering***, a partir de experimentar probando cosas. En los niños pequeños, esta forma de indagación se presenta como una etapa basada en el juego y la exploración con el fin de aprender sobre los objetos del mundo. **Deconstruir** cosas es una actividad básica, desarmarlas para ver cuáles son sus componentes, materiales, uniones, conexiones, etc. es un juego de hackeo, donde el experimentador deja de ser un consumidor pasivo y pasa a ser alguien que se cuestiona su propio mundo y lo quiere intervenir.

  Las comunidades de aprendizaje que experimentan con actividades basadas en el ***tinkering*** tienen la libertad de explorar y, en general, lo realizan en un entorno lúdico y libre de riesgos, lo que les da confianza en sí mismas a partir de las posibilidades de creación que aparecen. Estas actividades están asociadas con el razonamiento lógico; en la práctica, a medida que los estudiantes prueban cosas comienzan a acumular experiencias de tipo causa y efecto (por ejemplo: si muevo esto, entonces sucede aquello). Las modificaciones a un objeto, derivadas de una actividad de ***tinkering***, ayudan a ver que este abordaje de la tecnología pasa por desarrollar la comprensión en vez de obtener respuestas prearmadas y actividades totalmente cerradas. La experimentación brinda a niños y a jóvenes la posibilidad de enfocar algo desde distintas perspectivas, lo cual favorece una compresión holística de las situaciones y suele traer de la mano soluciones no tradicionales.

- **Creatividad**: El trabajo creativo comprende aspectos de originalidad. Estos se ponen en juego al **crear** algo valioso y significativo para quien está involucrado directamente en el proceso. La creatividad se aplica tanto a las cosas tangibles como a las que no lo

son: hoy los medios y las herramientas digitales han expandido significativamente el espacio de creación del hombre. En una actividad típica donde se crea algo, existen etapas asociadas que, en general, tienen que ver con la planificación, el hacer y la evaluación de lo creado. Por ejemplo, la programación de por sí es un proceso creativo.

Ken Robinson ha definido la creatividad como «*el proceso de tener ideas originales que tienen valor*» (***Aprendemos Juntos***, 2018). Por eso, toda actividad de aprendizaje que implique creación debe ser original, debe nacer de los estudiantes, no debe ser algo donde se copie o se complete sobre la base de instrucciones. Por otro lado, una vez terminado el trabajo, al compartirlo, el creador se comunica con otros y va desarrollando la confianza en sí mismo. La originalidad y el valor agregado cooperan en la construcción del conocimiento dado que, al tener fluidez en el manejo de técnicas, herramientas y materiales, se desarrollan habilidades, saberes e independencia. En síntesis, es una manera lúdica, experimental y útil de expresar las ideas.

Consideraciones finales

Existe una serie de conceptos que representan la esencia del pensamiento computacional y que merecen ser tenidos en cuenta cuando se piensa en su aplicación y desarrollo en contextos educativos. Son los siguientes:

- **En el pensamiento computacional se conceptualiza, no se programa**. Esto se debe a que se piensa como un científico de la computación y, por ende, se requiere pensar apelando a múltiples niveles de abstracción.
- **En el pensamiento computacional son fundamentales las habilidades no memorísticas o no mecánicas**. Para programar hace falta una mente imaginativa que haga uso del pensamiento divergente.
- **En el pensamiento computacional lo importante son las ideas, no los artefactos**. En esencia, el objetivo es ayudar a que se desarrolle un estudiante activo, creador de artefactos con tecnología, más que un usuario pasivo que se entretiene con objetos tecnológicos.

El desarrollo del pensamiento computacional en el currículo educativo puede verse como una oportunidad de ayudar a que niños y jóvenes construyan saberes significativos y necesarios para este tiempo. En particular, las oportunidades pueden estar en relación con las siguientes acciones:

- **Actualizar la alfabetización:** Desarrollar usos efectivos de las herramientas y tener fluidez digital se ha vuelto tan importante como leer, escribir y hacer cuentas. Esto, por un lado, permite conectar a los estudiantes con su entorno,

con su realidad diaria, y, por otro lado, al requerir de estudiantes activos, promueve una relación activa y crítica con tal entorno. Aplicar el pensamiento computacional en proyectos resolviendo problemas, generando prototipos de soluciones y analizando críticamente lo hecho permite superar con creces el «*modelo de enseñanza bancario*», según lo definió el pedagogo ***Paulo Freire***, que critica el modelo educativo basado exclusivamente en la transmisión no dialógica. Los estudiantes pasan a ser coautores y no solamente usuarios pasivos. También, el pensamiento computacional puede colaborar para ayudar a comprender qué es lo que ocurre detrás de todo dispositivo digital, ya sea en su faz tecnológica o social.

- **Potenciar el trabajo colectivo:** La mayoría de los objetos tecnológicos que nos rodean son el resultado de procesos de resolución de problemas que involucraron mucha inversión económica y de recursos humanos. En general, no fueron hechos por una única persona en un garaje o en un laboratorio aislado; por lo contrario, son el resultado de la acumulación de muchas soluciones aportadas por distintas personas. El trabajo en equipo es algo fundamental, tanto en la época en que se es estudiante, como a lo largo de toda la vida.

  Los equipos permiten trabajar aspectos colaborativos y cooperativos. Los colaborativos se dan cuando los participantes trabajan todos a la vez sobre una situación determinada, en una ida y vuelta, tratando de enfocar sus esfuerzos en una sola cosa (por ejemplo, diseñar una solución a un problema o evaluar una solución construida). En cambio, el trabajo cooperativo sucede en un equipo cuando las tareas se dividen y cada participante o subgrupo del equipo realiza las que acordó hacer para lograr un determinado resultado (por ejemplo, en un diseño de solución alguien podría desarrollar un *hardware*; otro, fabricar un esqueleto de robot, y un subgrupo, programar las acciones acordadas). Para que se dé un trabajo efectivo en equipo, es necesario construir un marco para la colaboración, donde cada individuo participe en función de sus motivaciones, intereses y capacidades.

- **Conectar los saberes teóricos con la práctica:** El desarrollo efectivo del pensamiento computacional implica que los estudiantes, necesariamente, tienen que articular saberes prácticos con saberes teóricos. No es ni práctica vacía de sentido ni repetitiva, ni teoría desvinculada de la realidad de cada persona. Para diseñar y construir tecnología, en efecto, hay que conectar los elementos del hacer tecnológico con las bases del conocimiento teórico.

- **Apostar por la no directividad:** Es decir, intentar la superación del viejo mandato escolar que sostiene: «*todos deben aprender lo mismo y al mismo tiempo*». En este sentido, la no directividad implica tratar de encontrar nuevos modos de relación con el estudiante donde surja una actitud comprensiva del otro. Así, se intentan encontrar, a partir de una búsqueda compartida, los

intereses de los estudiantes y canalizar la energía que tienen para tratar de aumentar sus posibilidades de autonomía, de confianza en sí mismos y de responsabilidad.

- **Promover la participación genuina y la autonomía:** El trabajo por proyectos, donde las situaciones problemáticas son el centro, es la base de un hacer pedagógico que promueve la construcción genuina de conocimientos ya que existe una apropiación de estos a lo largo del proceso y, también, en el objeto realizado. Pero, para que este proceso sea efectivo, debe estar conectado con las motivaciones e intereses de los educandos. De esta manera, al promover un ambiente de trabajo donde esté presente el manejo autónomo y participativo, se podrán lograr aprendizajes ricos.

En síntesis, hemos presentado el **pensamiento computacional** como una forma alternativa de trabajo cognitivo enriquecida por el pensamiento ingenieril, el pensamiento científico y el pensamiento lógico-matemático, que se enfoca en el estudio de los problemas y sus soluciones, apuntando a que estas últimas puedan traducirse para que puedan ejecutarse por un procesador de información.

El método de trabajo asociado al **PC** (Pensamiento Computacional) plantea que todo problema complejo a trabajar debe descomponerse (proceso de descomposición) en subproblemas, lo cual implica que habrá más problemas, pero más pequeños y manejables, cuyas soluciones combinadas aportarán a la solución final del problema original.

El proceso de **abstracción** permite centrar la atención en las características más importantes del problema, al captar su esencia. Esto se logra al invisibilizar las características no fundamentales y, a la vez, destacar los rasgos más relevantes. Como resultado de la abstracción, se logra desarrollar una representación simplificada del problema.

Luego, se acude a la experiencia, al conocimiento compartido, cuando se procede a realizar el **reconocimiento de patrones**. Se buscan similitudes entre los distintos subproblemas dentro de una misma situación. De esta manera, se utiliza el conocimiento de problemas similares, que han sido resueltos con anterioridad y se lo aplica directamente (**generalización**). Cuantos más patrones se reconozcan, será más fácil y rápido el desarrollo de un proyecto.

Luego, cuando se tiene una solución al problema se la traduce en forma de plan de acción a ejecutar (**algoritmo**) para que sea realizado por una máquina o un agente humano. Finalmente, se realiza una **evaluación** profunda del algoritmo a los efectos de verificar que, por un lado, dé respuesta efectivamente al problema original, y, por otro lado, no contenga errores en sus instrucciones.