Introduciendo conocimientos sobre el Pensamiento Computacional en los primeros años de las carreras de ciencia, tecnología, ingeniería y matemáticas

Pamela Viale, Claudia Deco

Departamento de Investigación Institucional, Facultad de Química e Ingeniería del Rosario,

Pontificia Universidad Católica Argentina (PUCA), Rosario, Argentina.

Resumen

En la actualidad, las computadoras se encuentran accesibles para cualquier persona y permiten asistirle en la resolución de problemas de la vida cotidiana de manera automática y/o semi-automática mediante el uso de aplicaciones. Es por esta razón que los individuos de la sociedad actual deben pasar de ser meros consumidores de aplicaciones a productores o, al menos, conocedores de los fundamentos de la informática, para poder aprovechar el poder de cálculo y procesamiento de las nuevas tecnologías. Teniendo este objetivo en mente se propuso el proyecto actual para enseñar las bases de la informática a los alumnos del primer año de las carreras de ingeniería industrial, ingeniería ambiental, licenciatura en química y licenciatura en tecnología de los alimentos de nuestra universidad, ya que, si bien no son específicos de cada especialidad, constituyen una base de conocimiento esencial para cualquier profesional hoy en día.

Palabras Claves: Pensamiento Computacional, Resolución de Problemas, Automatización, Programación

Abstract

Nowadays, computers are accessible to any person and allow him/her to solve everyday problems in an automatic and/or semi-automatic way through specific applications. It is for this reason that individuals in today's society must transform from being mere consumers of applications to producers or, at least, have knowledge about the fundamentals of computer science, in order to take advantage of the computing and processing power of new technologies. With this objective in mind, the current project was proposed to teach the basics of computer science to the students of the first year of the careers of industrial engineering, environmental engineering, chemistry and food technology from our university. Although these knowledges are not specific to these careers, they constitute an essential knowledge base for any professional today.

Key Words: Computational Thinking, Problem Solving, Automatization, Programming

Introducción

Desde hace ya varios años, en Argentina y en el resto del mundo, quedan miles de puestos de trabajo sin cubrir en las áreas de ciencia, tecnología, ingeniería y matemáticas, usualmente denominadas STEM (Science, Technology, Engineering & Mathematics), por la falta de recursos humanos con la formación y los conocimientos necesarios. Las carreras afines a estas áreas no son de las más populares y existe una gran deserción en los primeros años *[Benotti et al., 2012; Dapozo et al., 2014; Beaubouef et al., 2005]*.

Esto ha sensibilizado a las instituciones educativas a abordar el problema, desde el punto de vista de la formación y aparece el término *Pensamiento Computacional* como un conjunto de conocimientos importantes a desarrollar en todo profesional, independientemente del área en la que vaya a desempeñarse. En 2006 J. Wing, profesora de la Universidad de Carnegie Mellon, utilizó por primera vez este término en un artículo para describir cómo piensa un científico de computadoras y, como acabamos de decir, los beneficios que esta forma de pensar podría tener en todos. Define el Pensamiento Computacional como: *"...el proceso de pensamiento envuelto en formular un problema y sus soluciones de manera que las soluciones sean representadas de una forma en que pueden ser llevadas a un agente de procesamiento de información" [Wing, 2006]*. *[Phillips, 2009]* retoma este concepto y lo describe como *"la integración del poder del pensamiento humano con las capacidades de las computadoras"*. En los últimos años, el Pensamiento Computacional descrito por Wing, ha tenido influencia en las investigaciones relacionadas al entendimiento y desarrollo de los procesos de enseñanza y aprendizaje.

Estado del arte

Según la **ISTE** (International Society for Technology in Education) y la **CSTA** (Computer Science Teacher Association) el Pensamiento Computacional es un proceso para la solución de problemas que incluye pero no se limita a

- La formulación de problemas de forma que se pueda usar una computadora y otras herramientas para ayudar a resolverlos.

- Analizar y organizar los datos de forma lógica.
- Representar los datos de forma abstracta como modelos y simulaciones.
- Automatizar la solución con pensamiento algorítmico
- Identificar, analizar e implementar posibles soluciones con la meta de lograr la más eficaz y eficiente combinación de pasos y recursos.
- Generalizar y transferir este proceso de solución de problemas a otros problemas.

La aprehensión de los conceptos y prácticas involucradas en el Pensamiento Computacional se ve favorecida por la implementación de proyectos en algún lenguaje de programación, o sea, programando. Es por este motivo que numerosos lenguajes y entornos de desarrollo fueron creados con fines didácticos. A continuación, se nombran y describen, muy brevemente, algunos de ellos.

Scratch es un lenguaje de programación visual *[Resnick et al., 2009]*, por bloques, desarrollado por el Grupo Lifelong Kindergarten del MIT Media Lab (2007). Permite el desarrollo de habilidades de razonamiento y lógicas mediante el aprendizaje de la programación sin tener conocimientos profundos sobre el código. Este lenguaje de programación se utiliza con fines didácticos para crear animaciones de forma sencilla y servir como introducción al contenido de programación más avanzado. Sin embargo, Scratch tiene ciertas debilidades *[Harvey et al., 2010]*. Carece de procedimientos recursivos y el soporte de estructuras de datos también es muy básico. Estas debilidades no son descuidos, los diseñadores de Scratch evitaron deliberadamente saturar el lenguaje.

Snap! *[Carrasquer, 2019]* fue desarrollado por Jens Mönig y Brian Harvey en la Universidad de California en Berkeley, con el objetivo de acercar el poder del lenguaje de programación Scheme y otros conceptos fundamentales de ciencias de la computación en un entorno similar al de Scratch. Durante algunos años, Snap! se desarrolló como una versión modificada de Scratch. La posibilidad de construir bloques propios a partir de otros bloques, fue una de las principales extensiones realizadas, y fue por esto que en un primer momento se lo nombró como BYOB por ser las siglas en inglés de "Build Your Own Blocks" (o bien en español, "Crea Tus Propios Bloques"). Hoy en día, Snap! es un proyecto separado, con un código base completamente independiente de Scratch y con numerosas características nuevas y particularidades propias que lo convierten en un lenguaje adecuado para un estudio serio de las ciencias de la computación así como también para ser utilizado en proyectos de investigación.

Alice es un software educativo libre y abierto orientado a objetos con un entorno de desarrollo integrado (IDE), que fue desarrollado por investigadores de la Universidad Carnegie Mellon, entre los que destaca Randy Pausch. Utiliza un entorno sencillo basado en "arrastrar y soltar" para crear animaciones mediante modelos 3D *[UVa Interface Group, 1995]*. Este software acepta tanto el modelo de programación orientada a objetos como el modelo dirigido a eventos.

Racket es un lenguaje multi-propósito, derivado de Scheme y Lisp. Desarrollado desde 1994 por Matthias Felleisen y otros, quienes decidieron crear un entorno de programación pedagógico propio llamado DrRacket *[Felleisen et al. 2001]*. Sus características acogedoras para los estudiantes incluyen soporte para múltiples "niveles de lenguaje" (estudiante principiante, intermedio y así sucesivamente). El lenguaje se utiliza en una variedad de contextos, como scripting, programación de propósito general, enseñanza de computación e investigación. El Massachusetts Institute of Technology (MIT) entre otras universidades, lo emplean como uno de los primeros lenguajes de programación.

Gobstones *[Martínez López, 2013]* es un lenguaje de programación creado en la Universidad Nacional de Quilmes donde su principal referente es el Dr. Pablo E. Martínez López. Se diferencia de otros lenguajes dedicados a tal fin debido a varias características. La más importante es que busca orientar al programador a un pensamiento denotacional (el "qué" de los programas), en lugar del tradicional pensamiento operacional (el "cómo" o secuencia de instrucciones). Otra de esas características es que el pasaje de este lenguaje a otros utilizados en la industria resulta más sencillo, pues posee conceptos fundamentales comunes a todos los lenguajes (pero con una separación mucho más clara que en ellos), y por la utilización de una sintaxis similar a la de éstos. La sintaxis del lenguaje fue diseñada para ser similar a lenguajes como C y Java ya que posee bloques de código encerrados por llaves, definiciones de procedimientos y funciones, parámetros por valor, variables exclusivamente locales, etc..

Existen numerosos referentes con respecto al estudio del pensamiento computacional. Nos concentramos ahora en dos equipos: el equipo de Karen Brennan de la Universidad de Harvard, perteneciente a la Graduate School of Education, Cambridge, Estados Unidos, y el equipo de Marcos Román-González de la Universidad Nacional de Educación a Distancia (UNED), de Madrid, España.

El grupo de trabajo de K. Brennan *[Brennan et al., 2012]* ha desarrollado un marco de referencia para el estudio del pensamiento computacional a través del uso online de la herramienta Scratch. Este estudio de la herramienta y, a su vez, la participación en workshops, les ha permitido proponer una definición propia de pensamiento computacional que involucra tres dimensiones diferentes:

1. **Conceptos computacionales**, conceptos que no son exclusivos de la herramienta en cuestión, sino que son comunes a la mayoría de los lenguajes de programación, como por ejemplo, secuencias, bucles, paralelismo, eventos, condicionales, operadores lógicos y matemáticos, datos, etc.
2. **Prácticas computacionales**, el uso sólo de conceptos no es suficiente sino que deben ser usados siguiendo un conjunto de estrategias o prácticas, entre ellas podemos citar la experimentación e iteración, el testeo y debugging, la reutilización de proyectos anteriores, la abstracción y modularización, entre otras.
3. **Perspectivas computacionales**, esto involucra el cambio de mirada que produce el uso de la herramienta en sus usuarios, dado que cambian su forma de expresarse con respecto al área de las ciencias de la computación, aprecian la importancia de conectarse con otros y producir colaborativamente, y se cuestionan sobre el mundo, lo cual les permite darse cuenta que podrían ayudar a resolver problemas cotidianos mediante el desarrollo de herramientas propias.

Con respecto a Marcos Román-González y su equipo, nuestro principal interés se centra en un test, desarrollado por dicho equipo, que permite evaluar los conocimientos que posee un individuo sobre el pensamiento computacional. La Tabla 1 muestra cómo se relacionan las dimensiones propuestas por el equipo de Karen Brennan y el test de pensamiento computacional propuesto por el equipo de Marcos Román-González *[Román-González et al., 2017]*:

Tabla 1: Relación entre las dimensiones propuestas por el equipo de K. Brennan y el test de pensamiento computacional propuesto por el equipo de M. Román-González

Dimensiones de Karen Brennan	Descripción	Componentes	¿Evaluado por Test de Marcos Román-González?
Conceptos Computacionales	*Conceptos usados por durante la programación*	*Secuencias*	*Sí*
		Bucles	*Sí*
		Eventos	*No*
		Paralelismo	*No*
		Condicionales	*Sí*
		Operadores	*Sí*
		Datos	*No*
Prácticas Computacionales	*Prácticas de resolución de problemas usadas durante el proceso de programación*	*Experimentación e Iteración*	*No*
		Testeo y Debugging	*Parcialmente*
		Reutilización de proyectos anteriores	*Parcialmente*
		Abstracción y Modularización	*Parcialmente*
Perspectivas Computacionales	*Cambios que produce aprender a programar*	*Forma de expresarse*	*No*
		Conexión con otros	*No*
		Cuestionamientos	*No*

El test de Marcos Román-González permite evaluar los conocimientos sobre pensamiento computacional teniendo en cuenta la capacidad de abstracción, de descomposición, de reconocimiento de patrones y la habilidad para comprender/corregir algoritmos.

Este test consta de 28 preguntas. A modo de ejemplo de dichas preguntas en las figuras 1 y 2 se muestran las preguntas 1 y 7 respectivamente.

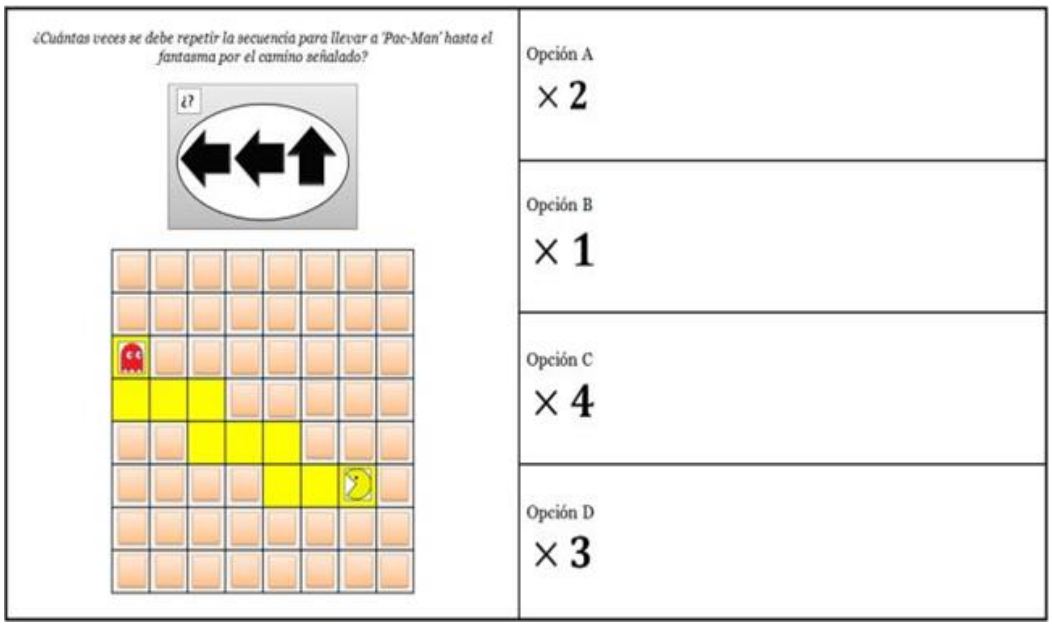

Figura 1: Pregunta 1 del Test de Marcos Román-González

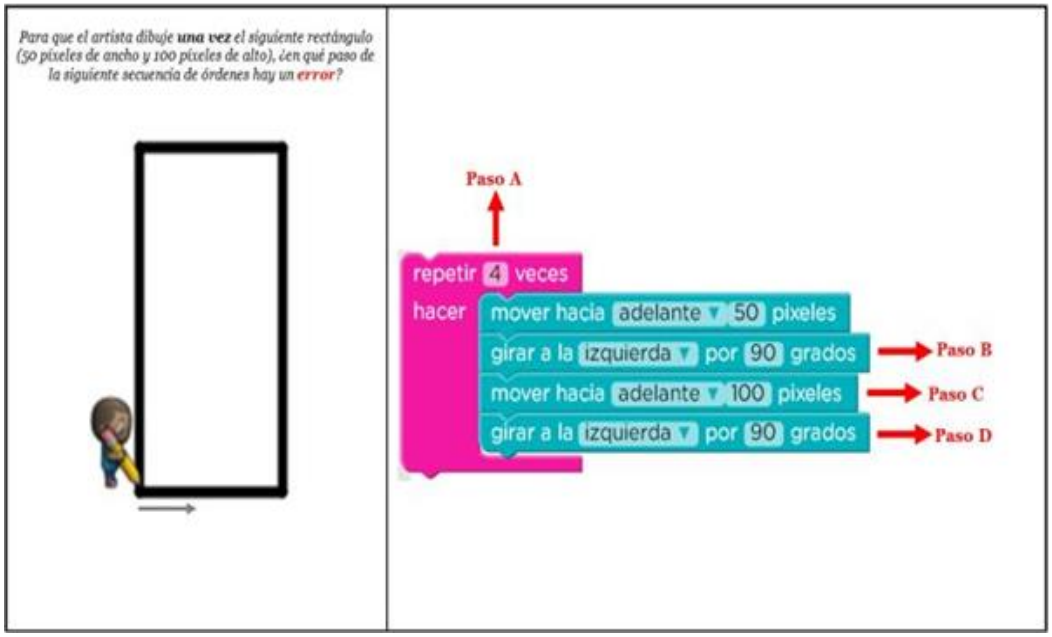

Figura 2: Pregunta 7 del Test de Marcos Román-González.

Cada pregunta evalúa diferentes capacidades que se resumen en la siguiente tabla *[Backmann, 2017]*:

Tabla 2: Capacidades evaluadas por las preguntas del Test de Marcos Román-González.

Pregunta	Abstracción	Descomposición	Reconocimiento de Patrones	Algoritmo
1	x			x
2	x			x
3	x			x
4		x	x	x
5		x	x	x
6		x	x	x
7	x	x		
8			x	x
9			x	x
10		x	x	x
11	x	x	x	x
12	x	x	x	x
13	x	x		x
14	x		x	x
15	x	x	x	x
16				x
17			x	x
18			x	x
19				x
20			x	x
21	x	x		x
22	x	x	x	x
23	x	x	x	x
24				x
25	x	x	x	x
26	x	x	x	x
27	x	x	x	x
28	x	x	x	x

Pensamiento Computacional en nuestras carreras

Desde julio de este año, hemos decidido incluir la enseñanza de los conceptos y prácticas involucradas en el pensamiento computacional en las carreras que se dictan en nuestra facultad. Hemos decidido comenzar concentrándonos en las materias Informática e Informática General que se dictan en el primer año de las carreras de Ingeniería Industrial, Ingeniería Ambiental, Licenciatura en Química y Licenciatura en Tecnología de los Alimentos. Se comenzó proponiendo las primeras 22 preguntas del test propuesto por el equipo de Marcos Román-González a nuestros alumnos durante el primer día de clases, tratando así de obtener una idea de los conocimientos sobre el pensamiento computacional con los cuales llegan a nuestra universidad.

A lo largo de todo el cuatrimestre se han desarrollado los contenidos de las materias en cuestión, haciendo fundamental hincapié en aquellos relacionados al pensamiento computacional. Al finalizar el cuatrimestre se solicitará a los mismos alumnos que respondan nuevamente las mismas preguntas y se evaluará si hubo o no mejoras con respecto a los resultados.

Conclusiones

La inclusión de la enseñanza del pensamiento computacional en diferentes carreras permitirá que los nuevos profesionales puedan hacer uso de las nuevas tecnologías para resolver diferentes situaciones problemáticas que se les presenten en su actividad laboral de manera más eficiente. Mediante este proyecto se tratará de no sólo enseñar sino también evaluar el grado de aprendizaje de estos conocimientos básicos en ciencias de la computación. Se tratará de evaluar la didáctica aplicada y el grado de entendimiento por parte de nuestros alumnos, para proponer cambios donde se detecten problemas de aprehensión de conceptos.

Bibliografía

[Backmann, 2017] C. P. Brackmann. Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica. Universidade Federal do Rio Grande Do Sul, Centro interdisciplinar de novas tecnologias na educação, Programa de Pós-Graduação em informática na educação, 2017.

[Benotti et al., 2012] L. Benotti, M. E. Echeveste, F. Schapachnik. Despertando vocaciones en computación mediante el uso de automatas de chat. In 6tas Jornadas de Vinculación Universidad Industria, 41JAIIO, pp 12-21, 2012.

[Beaubouef et al., 2005] T. Beaubouef, J. Mason. Why the high attrition rate for computer science students: Some thoughts and observations. SIGCSE Bull., 37(2):103-106, June 2005.

[Brennan et al., 2012]. K. Brennan, M. Resnick. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the american educational research association (vancouver: Canada).

[Carrasquer, 2019] B. Romagosa i Carrasquer. The Snap! Programming System , pp 1-10. Springer International Publishing, Cham, 2019.

[Dapozo et al., 2014] G. Dapozo, C. Greiner, G. Pedrozo Petrazzini, J. Chiapello. Vocaciones tic. ¿qué tienen en común los alumnos del nivel medio interesados por carreras de informática? En IX Congreso de Tecnología en Educación & Educación en Tecnología, pp 128-139, 2014.

[Felleisen et al. 2001] M. Felleisen, R. B. Findler, M. Flatt, S. Krishnamurthi. How to Design Programs: An Introduction to Programming and Computing. MIT Press, Cambridge, MA, USA, 2001.

[Harvey et al., 2010] B. Harvey, J. Monig. Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists? Constructionism , 2010, 01/2010.

[Martínez López, 2013] P. E. Martinez López. Las bases de la programación. Publicado electrónicamente por la Universidad Virtual de Quilmes, La Plata, 1era edition, 12/2013.

[Phillips, 2009] P. Phillips. Computational thinking: A problem solving tool for every classroom. Disponible en http://education.sdsc.edu/resources/CompThinking.pdf (accedido Julio 2019), 2009.

[Resnick et al., 2009] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai. Scratch: Programming for all. Commun. ACM , 52(11):60-67, November 2009.

[Román-González et al., 2017] M. Román-González, J. C. Perez-González, C. Jiménez-Fernández. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior, 72: 678-691, 2017.

[UVa Interface Group, 1995] UVa User Interface Group. Alice: Rapid prototyping for virtual reality. IEEE Comput. Graph. Appl., 15(3):8-11, May 1995.

[Wing, 2006] J. M. Wing. Computational thinking. Commun. ACM., 49(3):33-35, March 2006.