Elementos del Pensamiento Computacional aplicados a proyectos de extensión universitaria en Costa Rica

Irene Hernández Ruiz[1], Carolina Gómez Fernández[1]
[1] Universidad Nacional, Heredia, Costa Rica
{irene.hernandez.ruiz@una.ac.cr,
carolina.gomez.fernandez@una.ac.cr }

Abstract. El siguiente trabajo da a conocer un conjunto de elementos relacionados con el Pensamiento Computacional (PC), concepto que fue utilizado para la elaboración de diversos materiales y talleres en el proyecto de extensión universitaria "Creando Capacidades de Programación en Jóvenes y Docentes tanto en Secundaria como en Enseñanza Superior", durante los años 2020 al 2022 y en el proyecto de extensión universitaria Red UNA STEM[1] durante los años del 2020 al 2025. En este documento se presenta la información recopilada sobre la forma en que diversos autores conciben este concepto, además se indican las habilidades desarrolladas por las personas participantes de los talleres realizados por el proyecto, finalmente se presenta un conjunto de resultados y las principales conclusiones

Keywords: ***computational thinking; education; university extension.***

Elements of Computational Thinking Applied to a University Extension Projects in Costa Rica

Resumen. The following work presents a set of elements related to Computational Thinking (CT), a concept that was used for the development of various materials and workshops in the university extension project "Creating Programming Capabilities in Young People and Teachers in both Secondary and Higher Education", during the years 2020 to 2022 and in the university extension project Red UNA STEM during the years 2020 to 2025. This document presents the information collected on the way in which various authors conceive this concept, in addition to indicating the skills developed by the participants in the workshops carried out by the project, finally a set of results and the main conclusions are presented

Palabras clave: pensamiento computacional; educación; extensión universitaria.

[1] https://sites.google.com/una.cr/redunastemcr/inicio

1 Introducción

El proyecto Creando Capacidades de Programación en Jóvenes y Docentes tanto en Secundaria como en Enseñanza Superior de la Escuela de Informática de la Universidad Nacional de Costa Rica, tuvo como propósito el desarrollar un conjunto de talleres de programación tanto por bloques como con la programación de circuitos. Este proyecto pudo impactar a 1191 personas de diferentes zonas del país (Ver Anexo 1). Para lograr el propósito del proyecto se consideró importante definir la estrategia para impartir los talleres, además se realizó una búsqueda por medio en las bases de datos de la IEEExplore y ACM sobre la experiencia en talleres de programación en el mundo, de esta manera se encontró el concepto del Pensamiento Computacional desarrollado primeramente por Wing y posteriormente por otros autores.

Posteriormente, se formó parte del Red UNA STEM, en cual se logró no solo hacer uso del material logrado, sino poder también desarrollar actividades y materiales en el área STEM y aplicando los elementos más importantes del Pensamiento Computacional.

Este trabajo da a conocer el marco teórico desarrollado y utilizado en el proyecto con miras a que pueda ser utilizado en otros contextos y que sea un framework de trabajo con mejores prácticas que otros autores han desarrollado a nivel mundial. Se presenta a continuación los siguientes apartados que han sido desarrollado en este trabajo: elementos del pensamiento computacional, habilidades desarrolladas, entre otros temas.

II. Elementos del Pensamiento Computacional

El desarrollo del pensamiento computacional es de vital importancia para cualquier persona, independientemente de la edad, debido a que como lo indican CSTA e ISTE, citados por Basogain, Olabe y Olabe [1]:

> El Pensamiento Computacional es un enfoque para resolver un determinado problema que empodera la integración de tecnologías digitales con ideas humanas. No reemplaza el énfasis en creatividad, razonamiento o pensamiento crítico, pero refuerza esas habilidades al tiempo que realza formas de organizar el problema de manera que el computador pueda ayudar.

Jeanette Wing indicó en el año 2006 [2] que el pensamiento computacional (PC) implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, basándose en los conceptos fundamentales para la informática, posteriormente en el 2011 [3] aclaró que el PC es el proceso de pensamiento involucrado en formular

problemas, cuyas soluciones sean representadas de una manera que puedan ser efectivamente llevadas a cabo por un agente de procesamiento de información. Para implementar la resolución de problemas se recurre desde la informática al área de la programación, la cual es el conjunto de instrucciones necesarias que se le brinda a una máquina para poder realizar una tarea específica.

Sze Yee Lye y Joyce Hwee Ling Koh indicaron en el año 2014 [4] que el PC se refiere a derivar el proceso cognitivo de resolución de problemas, además consideraron que la programación de computadoras es la forma fundamental en que permite que el pensamiento computacional cobre vida.

Resnick [5] junto con otros autores desarrollaron en el 2009 una definición de pensamiento computacional que involucra tres dimensiones claves:

- conceptos computacionales: los conceptos que los diseñadores emplean mientras programan
- prácticas computacionales: las prácticas que los diseñadores desarrollan a medida que programan
- perspectivas computacionales: las perspectivas que los diseñadores forman sobre el mundo que los rodea y sobre sí mismos

Es importante considerar que el PC puede ser transferido a varios tipos de problemas que no involucran directamente tareas de programación como lo indica Wing [6], es por este motivo que uno de los recursos para el desarrollo del pensamiento computacional es el entorno de programación de Scratch.

Para desarrollar el PC se puede hacer uso del Framework desarrollado por Brennan y Resnick [7]. Brennan y Resnick [8] desarrollaron un Framework en el cual se indican elementos importantes como lo son: el concepto, las prácticas y las perspectivas del pensamiento computacional:

- el concepto da a conocer los temas de: secuencias, bucles, eventos, paralelismo, condicionales, operadores y datos
- las prácticas de pensamiento computacional pueden ser: incremental e iterativo, prueba y depuración, reutilizar y remezclar y resumen y modularización.
- entre las perspectivas se encuentran: expresando, conectando y cuestionando.

En la tabla 1, se mencionan algunas de las conceptualizaciones que hacen otros autores acerca del pensamiento computacional.

TABLA I CONCEPTUALIZACIONES NO REFERENCIALES, EXPLÍCITAS Y COMPONENCIALES DEL PENSAMIENTO COMPUTACIONAL (PC)

Autor (es)	Definición
Bers et al. (2014)[10]	Variables de PC - depuración, secuencias, correspondencia, control de flujo
Yadav et al. (2014) [11]	Conceptos de PC – identificación de problemas, descomposición del problema, abstracción, pensamiento lógico, algoritmos, depuración

Atmatzidou y Demetriadis (2016) [12]	Dimensiones de PC – abstracción, generalización, algoritmos, descomposición, modularidad
Atmatzidou y Demetriadis (2017) [13]	Conceptos de PC - abstracción, generalización, algoritmos, descomposición, modularidad, depuración
Looi et al. (2018) [14]	Habilidades de PC – descomposición, algoritmos, abstracción, generalización, evaluación
Witherspoon et al. (2018) [15]	Conceptos de PC – secuencias, condicionales, iteración
Tran (2019) [16]	Conceptos de PC- secuencias, algoritmos, bucles, depuración, condicionales
Nam et al. (2019) [17]	Formularios de PC – secuencias, solución de problemas
Calderon et al. (2020) [18]	Elementos de PC – abstracción, descomposición, datos, algoritmos, secuencias
Chen et al. (2020) [19]	Elementos de PC – creatividad, valores, simplificación, incrustación, simulación, transformación
Angeli y Valanides (2020) [19]	Elementos de PC – algoritmo, secuenciación, descomposición, depuración
Noh y Lee (2020) [21]	Componentes de PC – colección de datos, análisis de datos, estructuración, descomposición, modelado, algoritmo, automatización, generalización
Yin et al. (2020) [22]	Subhabilidades de PC – descomposición, abstracción, algoritmos, generalización de patrones
Uzumcu y Bay (2020) [23]	Dimensiones de PC – Comprensión de problemas, diagramas de flujo, condicionales, bucles, paralelismo, descomposición, abstracción, patrones, algoritmos, evaluaciones, depuración

En la tabla 1y la tabla 2 se puede observar que hay una coincidencia entre los conceptos que se enseñan con el pensamiento computacional entre ellos se puede encontrar la abstracción, los algoritmos y las estructuras condicionales.

TABLA II DEFINICIÓN DE PENSAMIENTO COMPUTACIONAL (PC) DESDE LA LITERATURA DE REFERENCIA

Artículo referenciado	Artículos referenciados	Encuadre del pensamiento computacional
Brennan y Resnick (2012) [8]	14	Conceptos de PC – secuencias, bucles, paralelismo, eventos, condicionales, operadores y datos
		Prácticas de PC – diseño incremental e

		iterativo, pruebas y depuración, reutilizar y remezclar y abstracción y modularización
		Perspectivas de PC – expresión, conexión y preguntas
Moreno – León et al (2015) [24]	8	Abstracción y descomposición de problemas, pensamiento lógico, sincronización, paralelismo, control de flujo algorítmico, interactividad de usuario, representación de datos. También conocido como Dr. Scratch
Román – González et al (2017) [25]	6	Secuencias, bucles, condicionales, funciones y variables. También conocido como prueba de pensamiento computacional
Dagiené y Sentance (2016) [26]	4	Abstracción, algoritmos, descomposición, evaluación y generalización. También conocido como tarea Bebras
D. Barr et al (2011) [27], ISTE (International Society for Technology in Education) y CSTA (Computer Science Teachers Association)(2011) [28]	2	Resolución de problemas que incorpora características de formulación de problemas, abstracción, pensamiento lógico, algoritmos, eficiencia, generalización y transferencia
Csizmadia et al (2015) [29] , Selby y Woollard (2013) [30]	2	Abstracción, descomposición, algoritmos, evaluación y generalización
Weintrop et al (2016) [31]	1	Práctica de datos, simulación y modelado, solución de problemas, pensamiento sistémico
CMCCT (Camegie Mellon Center for Computational Thinking) (n.d) [32]	1	Abstracción, algoritmos

En la tabla III , se encuentran aquellas estructuras del pensamiento computacional que diversos autores han utilizado. Además, puede observarse que el trabajo del pensamiento computacional ha tenido un gran auge desde el año 2011 y los elementos que han utilizado han permitido tener una gran base de conocimiento para poder ser replicadas.

Tabla III RESUMEN DE ESTRUCTURAS DE PENSAMIENTO COMPUTACIONAL

	Barr y Stephenson (2011) [33]	Brennan y Resnick (2012) [8]	Selby (2012) [34]	Grover y Pea (2013) [35]	Seiter y Foreman (2013) [36]	Kalelioglu, Gülbahar y Kukul (2016) [37]	Angeli et al. (2016) [20]	Repenning, Basawapatna y Escherle (2016) [38]
Abstracción	✓	✓	✓	✓	✓	✓	✓	✓

Algoritmos	✓	✓	✓	✓	✓	✓	✓	
Datos	✓	✓		✓	✓	✓		
Descomposición (desgloce)	✓	✓	✓	✓	✓	✓	✓	
Paralelismo	✓	✓		✓	✓	✓		
Pruebas y depuración	✓	✓		✓		✓		✓
Estructuras de control	✓	✓		✓		✓		
Automatización						✓		✓
Generalización			✓			✓	✓	
Simulación	✓		✓			✓		
Eventos		✓						
Iterativo e incremental		✓						
Expresión, conexión y preguntas		✓						
Reutilización y remezcla		✓						
Eficiencia y restricciones de rendimiento				✓				
Procedimiento sistemático				✓				
Conceptualización						✓		

III. HABILIDADES QUE SE PUEDEN GENERAR CON EL PENSAMIENTO COMPUTACIONAL

Actualmente es de suma importancia que las nuevas generaciones puedan desarrollar un conjunto de habilidades no solo a nivel general sino desde el punto de vista de la programación. Para ello se presenta la tabla 4, la cual resume este tipo de habilidades para lo cual se desarrolló una busqueda de literatura sobre el tema.

TABLA IV HABILIDADES GENERALES Y HABILIDADES EN RELACIONADO EN PROGRAMACIÓN

Categoría	Habilidad	Referencias
Habilidades relacionadas a programación	Solución de Problemas	[40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55]

	Habilidad Matemática	[40], [56], [42], [57], [38], [59], [60], [61], [62], [50], [52], [53], [54], [63], [64]
	Previo conocimiento en programación	[65], [43], [66], [67], [68], [49], [62], [50], [52], [53], [64]
	Abstracción	[56], [69], [44], [57], [70], [64]
Habilidades educativas generales	Conocimiento básico de inglés	[71], [66], [72], [52], [54]
	Pensamiento crítico y habilidad de discusión	[42], [69], [73], [57]
	Creatividad	[69], [57], [55], [74]
	Gestión del tiempo	[40], [70]

IV. CASOS DE ÉXITO DEL PENSAMIENTO COMPUTACIONAL HACIENDO USO DE SCRATCH

Se encontró en la literatura el uso de un entorno de programación llamado Scratch, el cual desde el año 2017 tiene una comunidad en línea que cuenta con proyectos creados por niños y jóvenes con edades entre los 8 y los 16 años, los cuales han compartido más de 2,5 millones de trabajos. Los miembros de la comunidad pueden interactuar con los proyectos (probarlos o descargarlos) y además con otros miembros (dejar comentarios o marcar a alguien como amigo) [8] [75]

En el año 2015 se realizó una experiencia sobre el uso de la herramienta en Chile. Los autores encontraron que el uso de Scratch constituye un instrumento propicio para el desarrollo del pensamiento lógico y algorítmico en los estudiantes, además presenta un ambiente que es motivador y permite la participación en la propuesta de soluciones a las situaciones planteadas, lo que posibilita el análisis de problemas, la propuesta, el desarrollo y la aplicación de soluciones lógicas y algorítmicas [76].

Otro caso de estudio es el presentado por Armoni, Meerbaum-Salant y Ben-Ari, quienes analizaron la transición de estudiar informática con el entorno visual de Scratch a estudiar informática con un lenguaje de programación de texto profesional como C # o Java en la escuela secundaria. “Descubrimos que el conocimiento y la experiencia de programación de los estudiantes que habían aprendido Scratch facilitaron enormemente el aprendizaje del material más avanzado en la escuela secundaria: se necesitaba menos tiempo para aprender nuevos temas, había menos dificultades de aprendizaje y lograron niveles cognitivos más altos de comprensión de la mayoría conceptos” [77]

Bustillo Bayón [78] observó que la inclusión del programa Scratch en la educación primaria facilita la incorporación de nuevas técnicas de aprendizaje, metodologías de enseñanza y recursos. Los estudiantes del Magisterio de Vitoria-Gasteiz, ubi-

cado en España realizaron diferentes experiencias referentes al lenguaje de programación, lo que habilitó la posibilidad de incluir la herramienta como una de las prácticas que aplican los docentes a la hora de enseñar, además permitió que la herramienta Scratch ayudara a que los profesores aprendieran, experimentaran y observaran los resultados que este tipo de actividades generó en sus estudiantes.

El Ministerio de Educación de Colombia [79], incorporó al programa de matemáticas la herramienta Scratch, con el fin de enriquecer el desarrollo del pensamiento lógico matemático de forma divertida y solucionando problemas. El proyecto se desarrolló con niños de tercer grado de primaria de las escuelas públicas, donde se realizaron actividades interactivas sobre las figuras geométricas.

En un estudio realizado por Vázquez y Ferrer [80] de la Universidad Española a Distancia se crearon experiencias educativas con alumnos de bachillerato, donde hicieron videojuegos en el aula utilizando Scratch, obteniendo como resultado de la experiencia estudiantes con la capacidad de diseñar y crear videojuegos complejos con diferentes módulos de programación, aumentando así sus destrezas técnicas y promoviendo una mayor creatividad en el proceso de enseñanza - aprendizaje, además se obtuvo como beneficio, que los profesores pudieron orientar sus clases desde una perspectiva más creativa.

López menciona [81] que el éxito en la aplicación de actividades en clase que favorecen el pensamiento algorítmico se debe a que programar con Scratch no es lo mismo que resolver problemas con Scratch, debido a que los estudiantes deben activar estrategias cognitivas, así como usar recursos y conceptos del pensamiento computacional para poder resolverlos.

V. ESTRATEGIAS UTILIZADAS

El proyecto realizó talleres con las herramientas Scratch, Scratch Jr y Tinkercad utilizando los componentes Arduino y micro:bit. Para el desarrollo de los talleres se analizó la información extraída de la tabla 3, en cuál se trabajó con: la abstracción, algoritmos, fatos, descomposición, pruebas, e iterativo.

En los talleres ser hizo énfasis en los siguientes puntos:

- impartir talleres de programación para la resolución de problemas informáticos.
- Conocer diversos entornos de programación, incluyendo creación de cuentas de usuario, áreas de trabajo, componentes, entre otros.
- Generar capacidades en las personas participantes para que puedan efectuar modificaciones en los proyectos realizados en los talleres.
- Realizar una evaluación de los talleres para conocer la percepción de los participantes

VI. CONCLUSIONES

Se logró en los talleres poner en práctica los conceptos del pensamiento computacional.

Es muy importante la creación de un marco teórico rubusto para poder aplicar los conocimientos en diferentes ambientes. Además, es importante poder conocer de las experiencias de otros casos en el mundo, ya que se puede replicar los trabajos, tropicalizarlos y aprender de la experiencia con otros colegas

Referencias

1. Basogain, X., Olabe, M. A., & Olabe, J. C. (2015). Pensamiento computacional a través de la programación: paradigma de aprendizaje. Revista de Educación a Distancia (RED), 46.

2. Wing, J. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

3. Wing, J. (2011). Research notebook: Computational thinking: what and why? The ink: The magazine of the Carnegie Mellon University School of Computer Science.

4. 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.

5. Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., & Brennan, K. (2009). Scratch: Programming for all. Communications of the ACM, 52(11).

6. Wing, J. (1881). Computational thinking and thinking about computing. Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 366.

7. Brennan, K., Valverde, A., Prempeh, J., Roque, R., & Chung, M. (2011). More than code: the significance of social interactions in young people's development as interactive media creators. In T.

8. Brennan, K., & Resnick, M. (2012). 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.

9. Ezeamuzie, N., & Leung, J. (2021). Computational thinking through an empirical lens: a systematic review of literature. Journal of Educational Computing Research.

10. Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: exploration of an early childhood robotics curriculum. Computers & Education, 72, 145–157.

11. Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. ACM Transactions on Computing Education, 14(1).

12. Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: a study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661–670.

13. Atmatzidou, S., & Demetriadis, S. (2017). A didactical model for educational robotics activities: a study on improving skills through strong or minimal guidance. In Educational Robotics in the Makers Era (pp. 58–72).

14. Looi, C.-K., How, M.-L., Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. Computer Science Education, 28(3), 255–279.

15. Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Shoop, R. (2018). Attending to structural programming features predicts differences in learning and motivation. Journal of Computer Assisted Learning, 34(2), 115–128.

16. Tran, Y. (2019). Computational thinking equity in elementary classrooms: what third-grade students know and can do. Journal of Educational Computing Research, 57(1), 3–31.

17. Nam, K. W., Kim, H. J., & Lee, S. (2019). Connecting plans to action: the effects of a card-coded robotics curriculum and activities on Korean kindergartners. Asia-Pacific Education Researcher, 28(5), 387–397.

18. Calderon, A. C., Skillicorn, D., Watt, A., & Perham, N. (2020). A double dissociative study into the effectiveness of computational thinking. Education and Information Technologies, 25(2), 1181–1192.

19. Chen, G., He, Y., & Yang, T. (2020). An ISMP approach for promoting design innovation capability and its interaction with personal characters. IEEE Access.

20. Angeli, C., & Valanides, N. (2019). Developing young children's computational thinking with educational robotics: an interaction effect between gender and scaffolding strategy. Computers in Human Behavior.

21. Noh, J., & Lee, J. (2019). Effects of robotics programming on the computational thinking and creativity of elementary school students. Educational Technology Research and Development, 68(1), 463–484.

22. Yin, Y., Hadad, R., Tang, X., & Lin, Q. (2019). Improving and assessing computational thinking in maker activities: the integration with physics and engineering learning. Journal of Science Education and Technology, 1–26.

23. Uzumcu, O., & Bay, E. (2020). The effect of computational thinking skill program design developed according to interest driven creator theory on prospective teachers. Education and Information Technologies.

24. Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: automatic analysis of Scratch projects to assess and foster computational thinking. RED-Revista de Educación a Distancia, 46, 1–23.

25. Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? criterion validity of the computational thinking test. Computers in Human Behavior, 72, 678–691.

26. Dagienė, V., & Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In A. Brodnik & F. Tort (Eds.), Informatics in schools: improvement of informatics knowledge and perception (pp. 28–39).

27. Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: a digital age skill for everyone. Learning Leading with Technology, 38(6), 20–23.

28. International Society for Technology in Education, & Computer Science Teachers Association. (2011). Operational definition of computational thinking for K–12 education.

29. Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: a guide for teachers.

30. Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

31. 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.

32. Carnegie Mellon Center for Computational Thinking. (n.d.). What is computational thinking?

33 Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K–12: what is involved and what is the role of the computer science education community? ACM Inroads, 2(1), 48–54.

34. Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

35. Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. Educational Researcher, 42(1), 38–43.

36. Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research.

37. Kalelioğlu, F. (2015). A new way of teaching programming skills to K–12 students: Code.org. Computers in Human Behavior, 52, 200–210.

38. Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., Horses, I. H. M., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., & Repenning, N. (2015). Scalable game design: a strategy to bring systemic computer science education to schools through game design and simulation creation. ACM Transactions on Computing Education, 15(2), 1–31.

39. Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. IEEE Transactions on Education, 62(2), 77–90.

40. Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. In Conference on Innovation and Technology in Computer Science Education, Uppsala, Sweden (pp. 39–44).

41. Santos, A., Gomes, A., & Mendes, A. (2013). A taxonomy of exercises to support individual learning paths in initial programming learning. In Front. Educ. Conf. (FIE), Oklahoma City, OK, USA (pp. 87–93).

42. Gomes, A., & Mendes, A. (2014). A teacher's view about introductory programming teaching and learning: difficulties, strategies and motivations. In Front. Educ. Conf. (FIE), Madrid, Spain (pp. 1–8).

43. Ateeq, M., Habib, H., Umer, A., & Rehman, M. U. (2015). C++ or Python? Which one to begin with: a learner's perspective. Teach. Learn. Comput. Eng. (LaTiCE).

44. Holvikivi, J. (2010). Conditions for successful learning of programming skills. In Key Competencies in the Knowledge Society (IFIP Advances in Information and Communication Technology [AICT], 324, 155–164.

45. Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. Computer Science Education, 21(1), 57–80.

46. Tavares, O. D. L., de Menezes, C. S., & de Nevado, R. A. (2012). Pedagogical architectures to support the process of teaching and learning of computer programming. In Front. Educ. Conf. (FIE), Seattle, WA, USA (pp. 1–6).

47. Thota, N. (2014). Programming course design: a phenomenographic approach to learning and teaching. In Teach. Learn. Comput. Eng. (LaTiCE), Kuching, Malaysia (pp. 125–132).

48. Kranch, D. A. (2011). Teaching the novice programmer: a study of instructional sequences and perception. Education and Information Technologies, 17(3), 291–313.

49. Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The influence of problem solving abilities on students' performance on different assessment tasks in CS1. In 47th ACM Technical Symposium on Computer Science Education, Memphis, TN, USA (pp. 329–334).

50. Silva-Maceda, G., Arjona-Villicaña, P. D., & Castillo-Barrera, F. E. (2016). More time or better tools? A large-scale retrospective comparison of pedagogical approaches to teach programming. IEEE Transactions, 59(4), 274–281.

51. Uysal, M. P. (2014). Improving first computer programming experiences: the case of adapting a Web-supported and well-structured problem-solving method to a traditional course. Contemporary Educational Technology, 5(3), 198–217.

52. Alturki, R. A. (2014). Measuring and improving student performance in an introductory programming course. Informatics in Education, 15(2), 183–204.

53. Hoskey, A., & Maurino, P. S. M. (2011). Beyond introductory programming: success factors for advanced programming. Information Systems Education Journal, 9(5), 61–70.

54. Apiola, M., & Tedre, M. (2012). New perspectives on the pedagogy of programming in a developing country context. Computer Science Education, 22(3), 285–313.

55. Shuhidan, S., Hamilton, M., & D'Souza, D. (2010). Instructor perspectives of multiple-choice questions in summative assessment for novice programmers. Computer Science Education, 20(3), 229–259.

56. Gomes, A. J., & Mendes, A. J. (2010). A study on student performance in first year CS courses. In ITiCSE, Ankara, Turkey (pp. 113–117).

57. Ambrósio, A. P., Costa, F. M., Almeida, L., Franco, A., & Macedo, J. (2011). Identifying cognitive abilities to improve CS1 outcome. In Front. Educ. Conf. (FIE), Rapid City, SD, USA (pp. F3G-1–F3G-7).

58. Yousoof, M., & Sapiyan, M. (2015). Optimizing instruction for learning computer programming—A novel approach, in Intelligence in the Era of Big Data. ICSIIT 2015 (Communications in Computer and Information Science), 516, 128–139.

59. Apiola, M., Moisseinen, N., & Tedre, M. (2012). Results from an action research approach for designing CS1 learning environments in Tanzania. In Front. Educ. Conf. (FIE), Seattle, WA, USA (pp. 1–6).

60. Gomes, A., & Mendes, A. J. (2010). Studies and proposals about initial programming learning. In Front. Educ. Conf. (FIE), Washington, DC, USA (pp. 1–6).

61. Giraffa, L. M. M., Moraes, M. C., & Uden, L. (2013). Teaching object-oriented programming in first-year undergraduate courses supported by virtual classrooms. In 2nd International Workshop on Learning Technology in Education Cloud (pp. 15–26).

62. Petersen, A., Craig, M., Campbell, J., & Tafliovich, A. (2016). Revisiting why students drop. In CS1 in Proc. 16th Koli Calling International Conference on Computing Education Research (pp. 71–80).

63. Ott, C., Robins, A., Haden, P., & Shephard, K. (2015). Illustrating performance indicators and course characteristics to support students' self-regulated learning. In CS1 Comput. Sci. Educ., 25(2), 174–198.

64. Robins, A. (2010). Learning edge momentum: a new account of outcomes. In CS1 Comput. Sci. Educ., 20(1), 37–71.

65. Tafliovich, A., Campbell, J., & Petersen, A. (2013). A student perspective on prior experience in CS1. In SIGCSE 44th ACM Technical Symposium on Computer Science Education, Denver, CO, USA (pp. 239–244).

66. Horton, D., & Craig, M. (2015). Drop, fail, pass, continue: persistence in CS1 and beyond in traditional and inverted delivery. In SIGCSE, Kansas City, MO, USA (pp. 235–240).

67. Porter, L., & Zingaro, D. (2014). Importance of early performance. In CS1: Two conflicting assessment stories, in Proc. SIGCSE, Atlanta, GA, USA (pp. 295–300).

68. Koulouri, T., Lauria, S., & Macredie, R. D. (2014). Teaching introductory programming: a quantitative evaluation of different approaches. ACM Transactions on Computing Education, 14(4), Article 26.

69. Brito, M. A., & de Sá-Soares, F. (2014). Assessment frequency in introductory computer programming disciplines. ACM Comput. Human Behav., 30, 623–628.

70. Bati, T. B., Gelderblom, H., & van Biljonc, J. (2014). A blended learning approach for teaching computer programming: design for large classes in Sub-Saharan Africa. Computer Science Education, 24(1), 71–99.

71. Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. ACM SIGCSE Bulletin, 37(3), 14–18.

72. Yousoof, M., & Sapiyan, M. (2015). Optimizing instruction for learning computer programming—A novel approach, in Intelligence in the Era of Big Data. ICSIIT 2015 (Communications in Computer and Information Science), 516, 128–139.

73. Eltegani, N., & Butgereit, L. (2015). Attributes of students engagement in fundamental programming learning. In Comput. Control Netw. Electron. Embedded Syst. Eng. (ICCNEEE), Khartoum, Sudan (pp. 101–106).

74. Shell, D. F. (2014). Improving learning of computational thinking using computational creativity exercises in a college CSI computer science course for engineers. In Front. Educ. Conf. (FIE), Madrid, Spain (pp. 1–7).

75. Brennan, K., Resnick, M., & Monroy-Hernandez, A. (2010). Making projects, making friends: online community as catalyst for interactive media creation. New Directions for Youth Development, 75–83.

76. Vidal, C. L., Cabezas, C., Parra, H. J., & López, L. P. (2015). Practical experiences for using the programming language Scratch to develop algorithmic thinking of students in Chile.

77. Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). Desde cero hasta la programación real. Transacciones de ACM sobre educación en computación.

78. Bustillo Bayón, J. (2015). Formación del profesorado con Scratch: análisis de la escasa incidencia en el aula.

79. Bolaños, M., Cuero, E., & Villalobos, N. (2017). Uso de Scratch como herramienta para el desarrollo de la competencia matemática.

80. Vázquez, E., & Ferrer, D. (2015). La creación de videojuegos con Scratch en Educación Secundaria. Communication Papers, 63–73.

81. López, J. C. (2014). Actividades de aula con Scratch que favorecen el uso del pensamiento algorítmico.