Revista Iberoamericana de la Educación, Vol - 10 No. 4, October - December 2026
e-ISSN: 2737-632x
Pgs 1-19

Pensamiento computacional y aprendizaje de la programación en estudiantes universitarios de América Latina: revisión crítica de tendencias, estrategias y desafíos en educación superior

How to cite:
Moreira, C., Arias, J., Astudillo, R., Egüez, R. (2026) Pensamiento computacional y aprendizaje de la programación en estudiantes universitarios de América Latina: revisión crítica de tendencias, estrategias y desafíos en educación superior. *Revista Iberoamericana De educación,* 9 (4).

Computational Thinking and Programming Learning among University Students in Latin America: A Critical Review of Trends, Strategies, and Challenges in Higher Education

Andrea Carolina Moreira Cañizares*
Jaime Arturo Arias Méndez*
Richard Vinicio Astudillo Sarmiento*
Rita Carolina Egüez Cevallos

Received: Agosto, 2026
Approved: Septiembre, 2026

http://www.revista-iberoamericana.org/index.php/es

MBA Administración de negocios Universidad de Guayaquil https://orcid.org/0000-0001-7222-3850 andrea.moreirac@ug.edu.ec

Magister en Pedagogía de las Ciencias Experimentales con mención en Matemática y Física. Universidad de Guayaquil, https://orcid.org/0009-0004-4042-602X jaime.ariasm@ug.edu.ec

Magister en Tecnología e innovación educativa. Universidad de Guayaquil https://orcid.org/0000-0001-5863-3087 richardastudillos@ug.edu.ec

Magister en educación informática Universidad Estatal Península de Santa Elena https://orcid.org/0000-0003-0226-3026 reguez8794 reguez8794@upse.edu.ec

Abstract

Computational thinking has become a relevant cognitive framework for problem solving in digitally mediated environments, while programming is one of its most visible means of externalization, practice, and assessment in higher education. In Latin America, the expansion of programs related to computing, engineering, science, education, and digital technologies has increased the need to understand how computational thinking development is connected with introductory programming learning, particularly among university students who move from heterogeneous school experiences to courses that demand abstraction, algorithmic reasoning, debugging, and modeling. This article aims to critically analyze recent literature on the relationship between computational thinking and programming learning among university students in Latin America, identifying conceptual trends, teaching strategies, empirical findings, tensions, and research gaps. A critical bibliographic review was conducted, mainly covering publications from 2021 to August 2026, prioritizing peer-reviewed articles, systematic reviews, Latin American empirical studies, and reports from international organizations. Findings indicate that instruction focused exclusively on syntax is insufficient; the most promising experiences shift attention toward problem solving, algorithm construction, active learning, visual programming, educational robotics, and formative feedback. Prior knowledge, self-efficacy, the quality of instructional scaffolding, and digital divides also condition learning outcomes. The emergence of generative artificial intelligence expands opportunities for personalized support but introduces risks of cognitive dependence and weakened reasoning processes when pedagogical mediation is absent. The review

concludes that Latin American universities need curricular designs that explicitly integrate computational thinking and programming as related but non-equivalent processes, supported by multidimensional assessment, faculty development, and institutional policies focused on equity and deep understanding of code.

Keywords: computational thinking; programming; higher education; university students; Latin America; active learning.

Resumen

El pensamiento computacional se ha consolidado como un marco cognitivo relevante para la resolución de problemas en entornos mediados por tecnologías digitales, mientras que la programación constituye uno de sus medios de exteriorización, práctica y evaluación más visibles en la educación superior. En América Latina, la expansión de carreras vinculadas con informática, ingeniería, ciencias, educación y tecnologías digitales ha incrementado la necesidad de comprender cómo se articula el desarrollo del pensamiento computacional con el aprendizaje inicial de la programación, especialmente en estudiantes universitarios que transitan desde experiencias escolares heterogéneas hacia cursos que demandan abstracción, razonamiento algorítmico, depuración y modelación. El objetivo de este artículo es analizar críticamente la literatura reciente sobre la relación entre pensamiento computacional y aprendizaje de la programación en estudiantes universitarios de América Latina, identificando tendencias conceptuales, estrategias didácticas, hallazgos empíricos, tensiones y vacíos de investigación. Se realizó una revisión bibliográfica crítica de publicaciones principalmente comprendidas entre 2021 y agosto de 2026, priorizando artículos arbitrados, revisiones sistemáticas, estudios empíricos latinoamericanos y documentos de organismos internacionales. Los hallazgos muestran que la enseñanza centrada exclusivamente en la sintaxis resulta insuficiente; las experiencias más prometedoras desplazan el foco hacia la resolución de problemas, la construcción de algoritmos, el aprendizaje activo, la programación visual, la robótica educativa y la retroalimentación formativa. También se evidencia que los conocimientos previos, la autoeficacia, la calidad del andamiaje docente y las brechas digitales condicionan los resultados. La irrupción de la inteligencia artificial generativa amplía las posibilidades de apoyo personalizado, pero introduce riesgos de dependencia cognitiva y de debilitamiento de procesos de razonamiento si no existe mediación pedagógica. Se concluye que las universidades latinoamericanas requieren diseños curriculares que integren explícitamente pensamiento computacional y programación como procesos relacionados, pero no equivalentes,

acompañados de evaluación multidimensional, formación docente y políticas institucionales orientadas a la equidad y a la comprensión profunda del código.

Palabras clave: pensamiento computacional; programación; educación superior; estudiantes universitarios; América Latina; aprendizaje activo.

INTRODUCCIÓN

La transformación digital ha modificado las demandas cognitivas, profesionales y formativas asociadas a la educación superior. En este escenario, programar ya no se reduce a dominar la gramática de un lenguaje computacional, sino que implica formular problemas, representar datos, anticipar comportamientos, diseñar procedimientos, evaluar soluciones y depurar errores. Tales operaciones se aproximan a lo que la literatura contemporánea denomina pensamiento computacional, un constructo que, pese a su diversidad definicional, converge en la capacidad de construir modelos abstractos y procedimientos algorítmicos para resolver problemas mediante procesos susceptibles de ser ejecutados por agentes humanos o computacionales. La meta-revisión de Astor et al. (2026), basada en un amplio conjunto de revisiones sistemáticas y metaanálisis, confirma que la fragmentación conceptual continúa siendo un problema, pero también identifica un núcleo común alrededor de la abstracción, los pasos computacionales y los algoritmos. Esta convergencia es especialmente pertinente para la enseñanza universitaria de la programación, pues ayuda a desplazar la discusión desde el dominio instrumental del código hacia la comprensión de los procesos de razonamiento que preceden, acompañan y evalúan su producción.

En la educación superior, la relación entre pensamiento computacional y programación es estrecha, aunque no debe interpretarse como identidad conceptual. Tikva y Tambouris (2021) muestran, en un mapeo sistemático sobre enseñanza y aprendizaje del pensamiento computacional mediante programación, que el campo universitario se ha organizado en torno a bases de conocimiento, estrategias de aprendizaje, herramientas, factores asociados, evaluación y desarrollo de capacidades. Esta estructura evidencia que programar puede funcionar como medio para desarrollar y observar el pensamiento computacional, pero no garantiza por sí sola su adquisición. El estudiante puede reproducir sintaxis, adaptar fragmentos de código o seguir procedimientos sin comprender la descomposición del problema, la selección de representaciones, la lógica del algoritmo o la generalización de la solución. En

consecuencia, el desafío didáctico consiste en promover una programación orientada al razonamiento y no únicamente a la ejecución técnica.

La problemática adquiere particular relevancia en América Latina. Belmar (2022), al revisar la enseñanza de programación y pensamiento computacional en diferentes regiones del mundo, identifica una expansión desigual y señala que América Latina presenta avances heterogéneos frente a regiones con políticas curriculares más consolidadas. Esta situación se relaciona con condiciones estructurales más amplias. La OECD (2023) advierte que los países latinoamericanos participantes en evaluaciones de habilidades muestran rezagos en alfabetización, numerosidad y resolución de problemas en entornos tecnológicos, mientras que Herrera et al. (2025), desde la CEPAL, sostienen que las brechas digitales regionales ya no pueden analizarse únicamente en términos de conectividad, debido a que persisten desigualdades en usos, competencias y oportunidades de aprendizaje. Aunque estas fuentes no se circunscriben exclusivamente a la población universitaria, ofrecen un marco relevante para comprender la diversidad de trayectorias con las que los jóvenes, de manera referencial en edades frecuentes de 18 a 25 años, ingresan a las instituciones de educación superior.

Las investigaciones universitarias latinoamericanas confirman esa heterogeneidad. Pérez (2021), al estudiar percepciones de estudiantes universitarios venezolanos antes de incorporar formalmente el pensamiento computacional en una asignatura introductoria de programación, encontró concepciones parciales y la ausencia de un marco compartido entre los propios aprendices. En Brasil, Nascimento et al. (2024) identificaron diferencias en habilidades de algoritmo y reconocimiento de patrones entre ingresantes de carreras de computación según su experiencia previa en programación. En México, Narváez Díaz et al. (2023) documentaron una metodología construccionista mediada por Scratch para fortalecer fundamentos de programación, mientras que en Ecuador Prado Ortega et al. (2023) reportaron beneficios de la programación por bloques articulada con robótica educativa y aprendizaje móvil. En Colombia, Ibatá Soto et al. (2024) observaron mejoras asociadas a un modelo activo con ejercicios en Python, especialmente en pensamiento lógico y reconocimiento de patrones. Estas experiencias sugieren que la enseñanza universitaria de la programación constituye un espacio privilegiado para desarrollar pensamiento computacional, pero también revelan que los resultados dependen de la arquitectura pedagógica, los instrumentos de evaluación, los conocimientos de entrada y el contexto institucional.

La relevancia científica del problema se intensifica por dos transformaciones recientes. La primera corresponde al creciente uso

de metodologías activas, entornos visuales, robótica, aprendizaje basado en problemas y sistemas de retroalimentación para reducir la carga cognitiva inicial y favorecer la comprensión conceptual. Calderon et al. (2024), en un mapeo sistemático de metodologías activas para enseñar programación en pregrado, muestran la diversidad de enfoques implementados y la necesidad de observar no solo el rendimiento final, sino las estrategias mediante las cuales el estudiante construye conocimiento. La segunda transformación proviene de la inteligencia artificial generativa. La disponibilidad de asistentes capaces de explicar, corregir y producir código modifica radicalmente la relación entre estudiante, error y solución. Nathaniel et al. (2025) advierten que la integración de estas herramientas puede apoyar el aprendizaje, pero también comprometer habilidades de orden superior y la lógica fundamental de programación cuando se utiliza sin diseño instruccional; Manorat et al. (2025) confirman que la inteligencia artificial se está extendiendo a la planificación de cursos, la implementación de clases, la evaluación, la retroalimentación y el seguimiento del desempeño.

En este contexto, el objetivo del presente artículo es analizar críticamente la literatura reciente sobre pensamiento computacional y aprendizaje de la programación en estudiantes universitarios de América Latina, con énfasis en educación superior, a fin de identificar las relaciones conceptuales entre ambos procesos, las dificultades de aprendizaje más recurrentes, las estrategias didácticas con mayor sustento, los condicionantes regionales y los vacíos que deben orientar futuras investigaciones. El análisis se articula desde la premisa de que una formación universitaria pertinente no debe medir el éxito por la capacidad de producir código funcional de manera aislada, sino por la posibilidad de comprender, explicar, transferir y evaluar los procesos de razonamiento que hacen posible una solución computacional.

MATERIALS AND METHODS

El estudio se desarrolló mediante una revisión bibliográfica crítica y documental de carácter integrador. La finalidad no fue efectuar un metaanálisis ni estimar un tamaño de efecto agregado, sino construir una síntesis analítica capaz de relacionar hallazgos empíricos, revisiones de literatura y documentos regionales con el problema específico de la formación universitaria en pensamiento computacional y programación. La ventana temporal principal comprendió desde 2021 hasta agosto de 2026, lo que permitió concentrar el análisis en la producción de los últimos cinco años y, simultáneamente, incorporar trabajos publicados en 2021 que resultan decisivos para el estado reciente del campo. Se priorizaron

fuentes revisadas por pares y verificables mediante DOI, portales editoriales, SciELO, repositorios institucionales y sistemas de información académica. Asimismo, se integraron informes de la OECD, la CEPAL cuando aportaban evidencia contextual sobre habilidades, transformación digital, equidad o educación superior en América Latina y el Caribe.

La localización de literatura se orientó mediante combinaciones de descriptores en español, inglés y portugués, entre ellos "pensamiento computacional", "computational thinking", "pensamento computacional", "programación", "programming", "higher education", "educación superior", "estudiantes universitarios", "undergraduate students", "Latin America", así como términos asociados a "programación por bloques", "aprendizaje activo", "robótica educativa", "introductory programming", "artificial intelligence" y "generative AI". Los criterios de inclusión consideraron la pertinencia directa con pensamiento computacional, programación o enseñanza universitaria; la presencia de población de educación superior o implicaciones transferibles a dicho nivel; la disponibilidad de información bibliográfica suficiente para comprobar la fuente; y la contribución a alguno de los ejes analíticos del estudio. Se excluyeron materiales de divulgación sin trazabilidad académica, referencias duplicadas, trabajos centrados exclusivamente en aspectos técnicos de ingeniería de software sin dimensión educativa y estudios cuyo objeto no permitía establecer relación con el aprendizaje de la programación o el pensamiento computacional.

El análisis se realizó mediante lectura temática comparativa. Las fuentes fueron examinadas en función de cinco categorías: conceptualización de la relación entre pensamiento computacional y programación; dificultades y factores asociados al aprendizaje inicial; estrategias didácticas y tecnologías mediadoras; evaluación de habilidades y heterogeneidad de entrada; y condiciones regionales e institucionales, incluyendo la emergencia de la inteligencia artificial generativa. Se privilegiaron contrastes entre resultados antes que la acumulación descriptiva de autores. En consecuencia, los hallazgos se interpretaron atendiendo a coincidencias, tensiones metodológicas, limitaciones de muestra, diferencias entre contextos nacionales y vacíos de investigación. Esta decisión es relevante porque una parte de la literatura latinoamericana disponible se apoya en estudios de caso o experiencias institucionales con muestras reducidas, por lo que sus resultados deben comprenderse como evidencia contextual prometedora y no como generalizaciones automáticas para toda la región.

RESULTS

Pensamiento computacional y programación: una relación de complementariedad, no de equivalencia

La revisión confirma que el principal problema conceptual consiste en confundir el pensamiento computacional con la mera capacidad de codificar. La literatura reciente permite superar esa reducción. Astor et al. (2026) identifican que, pese a la multiplicidad de definiciones existentes, el pensamiento computacional puede articularse alrededor de la construcción de representaciones abstractas y secuencias algorítmicas orientadas a la solución de problemas. Desde esta perspectiva, la programación es una práctica privilegiada para materializar tales representaciones, ya que obliga a convertir una intención de solución en estructuras formalizadas que pueden ejecutarse, verificarse y corregirse. Sin embargo, el código es un producto observable, mientras que el pensamiento computacional comprende procesos cognitivos y metacognitivos que pueden manifestarse antes, durante y después de escribirlo.

Tikva y Tambouris (2021) refuerzan esta distinción al mostrar que la investigación universitaria sobre pensamiento computacional mediante programación incluye no solo herramientas y lenguajes, sino también bases de conocimiento, estrategias pedagógicas, factores personales, evaluación y desarrollo de capacidades. Tal amplitud implica que el aprendizaje no puede reducirse a elegir entre Python, C++, Java o Scratch. El lenguaje constituye una mediación semiótica y técnica; la calidad del aprendizaje depende de cómo el estudiante descompone el problema, reconoce patrones, abstrae información irrelevante, diseña un algoritmo, prueba hipótesis y depura errores. Belmar (2022) sitúa precisamente la abstracción, la descomposición, la algoritmización, la depuración y la resolución de problemas entre los componentes reiterados del pensamiento computacional, lo cual permite entender por qué una instrucción excesivamente sintáctica puede producir desempeños frágiles: el estudiante aprende a construir expresiones válidas sin adquirir necesariamente un modelo explicativo del problema ni una estrategia transferible de solución.

Esta tensión aparece de manera explícita en la evidencia latinoamericana. Brugés Romero y Camperos Villamizar (2022), en estudiantes de ingeniería de Colombia, describen el aprendizaje de programación como un proceso que exige deducción, razonamiento y abstracción más allá de la memorización. El hallazgo es coherente con Pérez (2021), quien mostró que estudiantes universitarios podían aproximarse al pensamiento computacional desde concepciones intuitivas o incompletas antes de una integración curricular deliberada. La implicación pedagógica es significativa: si el profesorado presupone que programar automáticamente "produce"

pensamiento computacional, corre el riesgo de invisibilizar las operaciones cognitivas que necesitan enseñanza explícita. Por el contrario, cuando esas operaciones se nombran, modelan y evalúan, la programación puede convertirse en un laboratorio de razonamiento donde los errores dejan de ser fallas finales y pasan a constituir evidencias diagnósticas del proceso de pensamiento.

La distinción también tiene consecuencias curriculares. Sinésio Ferris da Silva y Pontual Falcão (2021), al analizar proyectos pedagógicos de licenciaturas en computación de Brasil, encontraron que la incorporación formal del pensamiento computacional era todavía limitada, aunque los programas que lo incluían tendían a presentarlo como una habilidad transversal de resolución de problemas y no exclusivamente como contenido técnico. Esta constatación resulta relevante para América Latina porque muestra una brecha entre el reconocimiento discursivo del pensamiento computacional y su traducción en resultados de aprendizaje, secuencias didácticas y procedimientos de evaluación. Una integración curricular madura debería explicitar qué dimensiones se pretende desarrollar, mediante qué experiencias de programación se trabajarán y cómo se distinguirá el dominio de sintaxis del razonamiento computacional subyacente.

Dificultades del aprendizaje inicial de programación y heterogeneidad de entrada

La programación introductoria continúa siendo una experiencia académica exigente porque integra conocimientos declarativos, habilidades procedimentales, modelos mentales y procesos de resolución de problemas. La revisión sistemática de Kalelioğlu y Keskinkılıç (2026) identifica entre las dificultades más recurrentes la comprensión insuficiente de conceptos y constructos de programación, las confusiones entre sintaxis y semántica, las limitaciones en resolución de problemas, la incapacidad para construir modelos mentales adecuados del funcionamiento del computador y las debilidades en el proceso de diseño de soluciones. Estos hallazgos permiten interpretar la dificultad no como atributo fijo del estudiante, sino como resultado de la interacción entre conocimiento previo, complejidad conceptual, representación del problema, estrategias de enseñanza y calidad de la retroalimentación.

En América Latina, la heterogeneidad de entrada es especialmente visible. Nascimento et al. (2024) evaluaron habilidades de pensamiento computacional en ingresantes de cursos de computación en Brasil antes de una asignatura introductoria de programación y observaron diferencias entre quienes no tenían experiencia y quienes declaraban conocer dos o más lenguajes. Este resultado no significa que conocer múltiples lenguajes equivalga a dominar pensamiento computacional, pero sí evidencia que la experiencia previa puede ofrecer esquemas de reconocimiento de patrones y algoritmos que

modifican el punto de partida. La consecuencia para el diseño universitario es clara: un curso único, con ritmo uniforme y supuestos homogéneos, puede ampliar las diferencias iniciales si no incluye diagnóstico, rutas de apoyo y actividades escalonadas.

La experiencia de Pérez (2021) aporta otra dimensión de la heterogeneidad: las concepciones. Antes de integrar formalmente el pensamiento computacional en un curso de introducción a la programación, los estudiantes mostraban percepciones que no necesariamente coincidían con los componentes académicos del constructo. Esta distancia entre lenguaje cotidiano y lenguaje disciplinar afecta la forma en que el estudiante interpreta una tarea. Por ejemplo, una actividad de "resolver un problema" puede ser entendida como encontrar rápidamente un código que funcione, mientras que desde una perspectiva de pensamiento computacional implicaría justificar la descomposición, seleccionar una representación, comparar alternativas y explicar por qué el algoritmo resulta adecuado. De allí que la alfabetización conceptual deba acompañar la práctica de programación desde las primeras semanas.

Las dificultades, además, tienen componentes afectivos y motivacionales. La literatura sobre programación introductoria ha documentado que el error reiterado puede disminuir la confianza cuando se percibe como evidencia de incapacidad, mientras que los entornos que convierten la depuración en una práctica normalizada favorecen persistencia y autorregulación. Calderon et al. (2024) muestran que las metodologías activas de programación se han expandido precisamente porque movilizan al estudiante desde una posición receptiva hacia la resolución, experimentación, colaboración y producción. La enseñanza universitaria necesita, por tanto, equilibrar la exigencia cognitiva con andamiajes que permitan al estudiante explicar su razonamiento, recibir retroalimentación oportuna y revisar soluciones sin que la evaluación se limite al resultado binario de "funciona/no funciona".

Estrategias didácticas con evidencia en contextos universitarios latinoamericanos

Las experiencias latinoamericanas revisadas convergen en un desplazamiento desde la exposición tradicional hacia estrategias que hacen visible el proceso de solución. En México, Narváez Díaz et al. (2023) aplicaron una metodología construccionista mediada por Scratch en una asignatura de Algoritmia de Ingeniería de Software. El valor didáctico de la programación visual no radica simplemente en su apariencia gráfica, sino en reducir parte de la carga asociada a errores sintácticos y permitir que el estudiante centre atención en secuencias, condiciones, iteraciones, variables y relaciones causales. Este tipo de entorno funciona especialmente bien cuando se utiliza como puente y no como destino final: su función es favorecer la

construcción de estructuras conceptuales que posteriormente puedan transferirse a lenguajes textuales.

Una lógica similar aparece en Ecuador. Prado Ortega et al. (2023), con estudiantes universitarios de la Universidad Técnica de Machala, analizaron beneficios de la programación por bloques mediante Sphero mini y aprendizaje móvil. Los estudiantes destacaron facilidad de uso, portabilidad, interacción directa con el dispositivo y mayor facilidad para detectar errores.

Desde una interpretación crítica, el aporte central no se encuentra en el recurso tecnológico por sí mismo, sino en la posibilidad de vincular representación algorítmica y efecto observable. Cuando un algoritmo controla el comportamiento de un robot, la abstracción adquiere una consecuencia física inmediata que facilita formular hipótesis, identificar errores y ajustar secuencias. No obstante, el tamaño reducido de la muestra y el carácter contextual del estudio exigen cautela antes de extrapolar resultados.

La robótica también ha sido utilizada como medio de formación integral. Gómez Rodríguez et al. (2024), en universitarios de la Universidad de Guadalajara, incorporaron pensamiento computacional dentro de talleres de robótica mediante una secuencia que incluía comprensión de la situación, identificación de la dificultad, descomposición, reconocimiento de patrones, selección de información relevante y diseño y ejecución de algoritmos. El estudio reportó mejoras asociadas al pensamiento creativo y al trabajo con metodologías activas. Aunque pensamiento creativo y pensamiento computacional no son equivalentes, su articulación resulta teóricamente plausible: diseñar soluciones computacionales implica generar alternativas, representar restricciones y evaluar consecuencias. La evidencia sugiere que las experiencias auténticas o semiauténticas pueden ampliar el aprendizaje más allá de ejercicios descontextualizados de entrada-salida.

En Colombia, Ibatá Soto et al. (2024) compararon un enfoque tradicional con una herramienta de aprendizaje activo basada en más de mil ejercicios de Python, organizados según dimensiones de pensamiento computacional. Los resultados mostraron progresos en el grupo experimental, especialmente en pensamiento lógico y reconocimiento de patrones, al tiempo que las dimensiones de resolución de problemas, algoritmos y pensamiento lógico aparecían entre las más débiles al inicio. Este hallazgo es particularmente significativo porque muestra que el simple aumento de práctica no basta si los ejercicios no están alineados con dimensiones cognitivas explícitas. La práctica deliberada adquiere valor cuando cada actividad permite identificar qué habilidad se moviliza, qué error conceptual se intenta superar y qué tipo de retroalimentación se ofrece.

La evidencia internacional sintetizada por Calderon et al. (2024) respalda esta orientación. Su mapeo sistemático identifica una amplia variedad de metodologías activas utilizadas en pregrado, entre ellas aprendizaje basado en problemas y proyectos, pair programming, gamificación, actividades colaborativas y recursos interactivos. El denominador común es el aumento de la actividad intelectual del estudiante y la creación de oportunidades para explicar, probar y revisar soluciones. Sin embargo, la revisión también sugiere un problema metodológico: los estudios utilizan métricas muy diversas, por lo que comparar efectividad entre estrategias resulta complejo. Para el contexto latinoamericano esto implica que el avance no debe medirse solo por la cantidad de innovaciones implementadas, sino por la calidad de los diseños de investigación y la consistencia de los instrumentos con los constructos evaluados.

En síntesis, las estrategias más prometedoras comparten cuatro propiedades. Primero, reducen el peso inicial de la sintaxis cuando esta interfiere con la comprensión. Segundo, sitúan la resolución de problemas como eje y no como aplicación posterior de contenidos. Tercero, hacen del error una fuente de información mediante depuración y retroalimentación. Cuarto, articulan representaciones múltiples diagramas, bloques, pseudocódigo, código textual, simulaciones o dispositivos físicos para facilitar la transición entre lo concreto y lo abstracto. Estas propiedades explican por qué la discusión no debería formularse en términos de "qué lenguaje es mejor", sino de qué secuencia de representaciones y tareas permite desarrollar una comprensión progresivamente más formal y transferible. (Calderon et al., 2024; Narváez Díaz et al., 2023; Prado Ortega et al., 2023).

Evaluación del pensamiento computacional y del aprendizaje de programación

La evaluación constituye uno de los puntos más sensibles del campo porque aquello que se mide termina orientando aquello que se enseña. Si la evaluación se limita a comprobar que un programa compila y produce una salida esperada, se pueden invisibilizar procesos centrales como descomposición, razonamiento algorítmico, calidad de la representación, depuración, comparación de alternativas y explicación del código. Tikva y Tambouris (2021) ya identificaban la evaluación como una de las áreas de mayor evolución dentro de la investigación universitaria sobre pensamiento computacional. A su vez, Astor et al. (2026) muestran que la literatura de síntesis continúa reportando fragmentación conceptual, lo que afecta directamente la comparabilidad de instrumentos: distintas escalas y pruebas pueden denominar "pensamiento computacional" a combinaciones parcialmente diferentes de habilidades.

El estudio de Nascimento et al. (2024) ilustra el potencial y las limitaciones de las evaluaciones diagnósticas. Al comparar

habilidades de algoritmo y reconocimiento de patrones antes de iniciar programación formal, la investigación permite identificar diferencias de entrada que una calificación final no mostraría. Esta lógica diagnóstica debería ampliarse a la educación superior latinoamericana mediante evaluaciones iniciales breves que distingan experiencia previa, razonamiento lógico, comprensión de problemas y autopercepción. El propósito no sería seleccionar o excluir estudiantes, sino ajustar apoyos, organizar actividades de nivelación y evitar que quienes llegan sin experiencia previa interpreten la diferencia inicial como una imposibilidad de aprender.
La evaluación formativa requiere igualmente evidencias de proceso. En ambientes de programación pueden analizarse versiones sucesivas del código, explicaciones verbales o escritas, diagramas, registros de depuración, pruebas diseñadas por el estudiante y justificaciones sobre decisiones algorítmicas. Este enfoque permite diferenciar entre quien llega a una solución mediante comprensión y quien obtiene un resultado funcional por ensayo aleatorio, copia o asistencia externa. La distinción se vuelve más importante en el contexto de inteligencia artificial generativa, donde producir código correcto ya no constituye evidencia suficiente de dominio. El reto contemporáneo es evaluar comprensión, capacidad de revisión crítica, transferencia y explicación, no solo producción. (Nathaniel et al., 2025; Tikva & Tambouris, 2021).
Condiciones regionales: brechas, currículo y capacidad institucional
La formación en pensamiento computacional no ocurre en un vacío tecnológico ni social. La OECD (2023) señala que los países latinoamericanos analizados presentan debilidades en habilidades básicas y en resolución de problemas en entornos ricos en tecnología, mientras que Herrera et al. (2025) subrayan que la brecha digital regional incluye desigualdades de acceso, uso y desarrollo de competencias. Estas condiciones afectan directamente a la educación superior, incluso cuando las universidades disponen de laboratorios o plataformas. La brecha puede manifestarse en la familiaridad con herramientas, la disponibilidad de equipos personales, la estabilidad de conectividad, el tiempo de exposición previa a programación y la posibilidad de practicar fuera del aula. Por ello, asumir una supuesta homogeneidad de “nativos digitales” resulta pedagógicamente problemático.
La dimensión curricular es igualmente relevante. Sinésio Ferris da Silva y Pontual Falcão (2021) muestran que la incorporación explícita del pensamiento computacional en licenciaturas brasileñas de computación era todavía incipiente. Esta constatación puede leerse como un indicador de transición: la región reconoce cada vez más la importancia del pensamiento computacional, pero aún necesita traducir ese reconocimiento en perfiles de egreso, resultados de aprendizaje, secuencias formativas y mecanismos de evaluación.

La ausencia de explicitación curricular favorece que el desarrollo del pensamiento computacional dependa de iniciativas individuales de docentes, lo que produce experiencias valiosas pero discontinuas y difíciles de institucionalizar.

La formación docente aparece, por tanto, como una condición de sostenibilidad para cualquier estrategia institucional de desarrollo de competencias digitales y computacionales en educación superior. Las brechas regionales no se resuelven únicamente ampliando el acceso a tecnología; también requieren experiencias educativas explícitamente orientadas a formar capacidades de uso pertinente (Herrera et al., 2025; OECD, 2023). No basta con incorporar Scratch, Python, robótica o inteligencia artificial en los programas; se requiere que el profesorado comprenda por qué, cuándo y con qué propósito didáctico emplear cada recurso. Una institución puede modernizar herramientas sin transformar la lógica pedagógica. La innovación relevante ocurre cuando tecnología, actividad cognitiva, evaluación y resultados de aprendizaje permanecen alineados.

Inteligencia artificial generativa: apoyo cognitivo y riesgo de sustitución del razonamiento

Desde 2023, la inteligencia artificial generativa ha alterado profundamente el aprendizaje de programación. Los asistentes basados en modelos de lenguaje pueden explicar conceptos, generar ejemplos, detectar errores, proponer pruebas, traducir entre lenguajes y producir soluciones completas en segundos. Manorat et al. (2025), en una revisión sistemática sobre inteligencia artificial en educación de programación, muestran que las aplicaciones ya abarcan diseño de cursos, implementación en aula, evaluación y retroalimentación, y seguimiento del rendimiento. Estas capacidades son especialmente atractivas en cursos iniciales porque ofrecen apoyo inmediato, una de las necesidades más persistentes del aprendizaje de programación.

Sin embargo, la disponibilidad de respuestas técnicamente plausibles introduce una tensión epistemológica: el estudiante puede obtener una solución sin atravesar los procesos cognitivos que la formación pretende desarrollar. Nathaniel et al. (2025), al revisar la integración de inteligencia artificial generativa en educación de programación, concluyen que los beneficios dependen de estrategias pedagógicas deliberadas, evaluación bien diseñada y procesos de integración estructurados. También advierten sobre la sobredependencia y el posible debilitamiento de pensamiento de orden superior y lógica fundamental de programación. La cuestión, por tanto, no es prohibir o permitir la inteligencia artificial, sino redefinir las tareas para que su uso exija comprensión y juicio.

En términos de pensamiento computacional, la IA generativa debería funcionar como objeto de crítica y herramienta de contraste. Un estudiante puede solicitar una solución y luego identificar supuestos, comparar complejidad, localizar errores, explicar cada decisión,

generar casos de prueba y proponer una alternativa. Tales actividades trasladan el valor educativo desde "producir código" hacia "evaluar y justificar código". Esta reorientación es crucial para la educación superior latinoamericana porque permite aprovechar asistencia automatizada sin renunciar al desarrollo de autonomía cognitiva. También exige políticas institucionales sobre integridad académica, transparencia de uso y diseño de evaluación, particularmente en cursos donde la calificación tradicional se basaba en tareas que ahora pueden resolverse automáticamente.

Los hallazgos permiten sostener que la relación entre pensamiento computacional y aprendizaje de programación en educación superior es bidireccional y condicionada. La programación ofrece un entorno de práctica para operaciones centrales del pensamiento computacional, pero el desarrollo de estas operaciones depende de que sean objeto explícito de enseñanza, reflexión y evaluación. Esta interpretación coincide con Tikva y Tambouris (2021) y con la síntesis conceptual de Astor et al. (2026), pero adquiere un matiz particular al confrontarse con evidencia latinoamericana: en contextos caracterizados por trayectorias educativas heterogéneas y brechas digitales, el acceso al curso de programación no equivale a igualdad de oportunidades para aprender.

La primera tensión identificada es curricular. En numerosas experiencias, pensamiento computacional aparece asociado a asignaturas específicas, talleres o innovaciones docentes, pero no necesariamente como un resultado formativo transversal y progresivo. Sinésio Ferris da Silva y Pontual Falcão (2021) muestran esta debilidad en licenciaturas brasileñas, mientras que experiencias de México, Ecuador y Colombia evidencian que las innovaciones se concentran en contextos institucionales concretos. Esto sugiere un patrón regional de experimentación pedagógica con institucionalización todavía limitada. Para superarlo, la universidad debería definir trayectorias: diagnóstico al ingreso, desarrollo inicial de resolución algorítmica, transferencia a lenguajes textuales, integración en proyectos auténticos y evaluación de procesos de pensamiento a lo largo de la carrera.

La segunda tensión es didáctica. Las evidencias de Narváez Díaz et al. (2023), Prado Ortega et al. (2023), Gómez Rodríguez et al. (2024) e Ibatá Soto et al. (2024) respaldan estrategias activas, visuales y contextualizadas. No obstante, sería metodológicamente incorrecto afirmar que una herramienta concreta garantiza mejores aprendizajes. Scratch puede reducir carga sintáctica, la robótica puede hacer visibles relaciones causa-efecto y Python puede facilitar prototipado, pero el efecto depende de la tarea, el andamiaje, la secuencia y la retroalimentación. La tecnología funciona como mediación; la variable pedagógica decisiva es la calidad del diseño que obliga al estudiante a pensar, explicar, revisar y transferir.

La tercera tensión corresponde a la evaluación. El campo todavía enfrenta problemas para construir medidas comparables de pensamiento computacional, debido a la diversidad conceptual señalada por Astor et al. (2026). En programación universitaria, el problema se agrava porque el código funcional puede ocultar comprensiones diferentes. La evaluación debería combinar desempeño de producto y evidencia de proceso, incorporando razonamiento algorítmico, pruebas, depuración, explicación, transferencia y reflexión. Esta necesidad se vuelve urgente con la inteligencia artificial generativa: cuando producir código deja de ser una actividad exclusivamente humana, la validez de tareas tradicionales disminuye y la evaluación debe desplazarse hacia competencias de interpretación y juicio.

La cuarta tensión es de equidad. Los datos regionales de la OECD (2023) y la CEPAL (Herrera et al., 2025) impiden asumir que todos los estudiantes universitarios ingresan con condiciones digitales equivalentes. Nascimento et al. (2024) muestran que la experiencia previa en programación se asocia con diferencias observables antes del curso introductorio. Por tanto, políticas de nivelación, recursos abiertos, tutorías, laboratorios accesibles y diagnósticos iniciales no son medidas remediales marginales, sino componentes de una pedagogía equitativa. La equidad no implica disminuir exigencia, sino crear condiciones para que la exigencia cognitiva sea alcanzable mediante apoyos diferenciados.

Finalmente, la revisión identifica un vacío de investigación regional. Aunque existen experiencias valiosas, siguen predominando estudios institucionales con muestras pequeñas, diseños de corta duración y métricas heterogéneas. Son necesarios estudios multicéntricos latinoamericanos, diseños longitudinales que acompañen la progresión del pensamiento computacional durante varios semestres, instrumentos con evidencia de validez en contextos culturales diversos y análisis de transferencia hacia asignaturas avanzadas. También se requiere estudiar con mayor precisión la relación entre pensamiento computacional, permanencia estudiantil y rendimiento en cursos iniciales, así como los efectos de la inteligencia artificial generativa sobre comprensión profunda, dependencia, autorregulación y capacidad de depuración. La investigación futura debería abandonar la pregunta simple de si una tecnología “mejora” el aprendizaje y avanzar hacia preguntas sobre para quién, bajo qué condiciones, mediante qué mecanismos y con qué efectos sostenidos.

CONCLUSIONS

El análisis realizado permite concluir que el pensamiento computacional constituye un marco cognitivo relevante para comprender y fortalecer el aprendizaje de la programación en

estudiantes universitarios de América Latina, pero no debe confundirse con el dominio técnico de lenguajes de programación. Su núcleo se expresa en procesos de abstracción, descomposición, reconocimiento de patrones, formulación algorítmica, evaluación y depuración que pueden desarrollarse mediante programación cuando el currículo y la enseñanza los hacen explícitos. Por ello, el éxito de un curso introductorio no debería definirse únicamente por la capacidad de producir código correcto, sino por la posibilidad de construir soluciones explicables, transferibles y susceptibles de revisión crítica.

La literatura reciente muestra que las metodologías activas, la programación visual, la robótica educativa, las representaciones múltiples y la retroalimentación formativa ofrecen condiciones favorables para este desarrollo. Las experiencias de México, Ecuador, Colombia y Brasil demuestran potencial, aunque su alcance debe interpretarse con cautela por la diversidad de diseños y tamaños de muestra. En conjunto, la evidencia apoya una secuencia pedagógica que reduzca inicialmente barreras sintácticas, fortalezca modelos conceptuales y resolución de problemas, y conduzca gradualmente hacia lenguajes textuales y proyectos de mayor complejidad.

También se concluye que la heterogeneidad de conocimientos previos y las brechas digitales constituyen variables estructurales de la formación universitaria latinoamericana. En consecuencia, los diagnósticos de entrada, la nivelación y los apoyos diferenciados deben formar parte del diseño ordinario de los cursos, no operar únicamente como respuestas posteriores al fracaso. De igual manera, la integración curricular del pensamiento computacional requiere formación docente y acuerdos institucionales que definan resultados de aprendizaje, indicadores y mecanismos de seguimiento a lo largo de las carreras.

La inteligencia artificial generativa introduce un punto de inflexión. Su potencial para proporcionar explicaciones y retroalimentación puede favorecer el aprendizaje, pero también puede sustituir procesos de razonamiento cuando las tareas se limitan a obtener una solución funcional. Las universidades deben rediseñar actividades y evaluaciones para exigir explicación, verificación, comparación y transferencia, de manera que la inteligencia artificial se utilice como apoyo a la metacognición y no como reemplazo de la construcción conceptual. En términos de investigación, la región necesita estudios longitudinales, multicéntricos y metodológicamente comparables que permitan establecer cómo progresa el pensamiento computacional, qué estrategias producen efectos sostenidos y de qué modo las condiciones institucionales y tecnológicas median los resultados. La principal implicación es que enseñar programación en

la universidad contemporánea significa enseñar a pensar computacionalmente con, sobre y más allá del código.

REFERENCES

Astor, K., Rönnlund, J., Fawcett, C., & Gredebäck, G. (2026). Computational thinking: A meta-review of systematic reviews and meta-analyses. Educational Research Review, 52, 100794. https://doi.org/10.1016/j.edurev.2026.100794

Belmar, H. (2022). Review on the teaching of programming and computational thinking in the world. Frontiers in Computer Science, 4, 997222. https://doi.org/10.3389/fcomp.2022.997222

Brugés Romero, A. R., & Camperos Villamizar, Y. del P. (2022). Desarrollo del pensamiento computacional a través del aprendizaje de la programación en estudiantes de ingeniería. Revista Investigación & Praxis en CS Sociales, 1(1), 1–16. https://doi.org/10.24054/ripcs.v1i1.1311

Calderon, I., Silva, W., & Feitosa, E. (2024). Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. Informatics in Education, 23(2), 279–322. https://doi.org/10.15388/infedu.2024.11

Gómez Rodríguez, H., González Fernández, M. O., & Aceves Aldrete, C. E. (2024). La creatividad y pensamiento computacional: una experiencia de formación integral a través de talleres de robótica en universitarios. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 14(28), e658. https://doi.org/10.23913/ride.v14i28.1901

Herrera, P., Huepe, M., & Trucco, D. (2025). Educación y desarrollo de competencias digitales en América Latina y el Caribe. CEPAL. https://hdl.handle.net/11362/81377

Ibatá Soto, L. J., Correa Zapata, J. C., Vallejo Córdoba, S. L., & Tamayo Quintero, J. D. (2024). Implementación experimental de modelos activos para la enseñanza de programación en entornos universitarios. Mundo FESC, 14(30), 318–337. https://doi.org/10.61799/2216-0388.1794

Kalelioğlu, F., & Keskinkılıç, F. (2026). Difficulties of learning programming in higher education: A systematic literature review of recent research. Ankara University Journal of Faculty of Educational Sciences, 59(2), 771–835. https://doi.org/10.30964/auebfd.1703365

Manorat, P., Tuarob, S., & Pongpaichet, S. (2025). Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence, 8, 100403. https://doi.org/10.1016/j.caeai.2025.100403

Narváez Díaz, L. E., Escalante Torres, M., González Segura, C. M., Miranda Palma, C., & Canché Euán, M. (2023). Propuesta metodológica para mejorar el desempeño académico de los estudiantes en fundamentos de programación. RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 14(27). https://doi.org/10.23913/ride.v14i27.1714

Nascimento, A. K. C. do, Soares, V. M. A., Oliveira, A. L. S., & Andrade, W. L. (2024). Estimando habilidades de pensamento computacional em ingressantes de cursos de computação. En Anais do XXXV Simpósio Brasileiro de Informática na Educação (pp. 2287–2300). Sociedade Brasileira de Computação. https://doi.org/10.5753/sbie.2024.242356

Nathaniel, J., Oyelere, S. S., Suhonen, J., & Tedre, M. (2025). Literature review on the integration of generative AI in programming education. International Journal of Artificial Intelligence in Education, 35, 2724–2755. https://doi.org/10.1007/s40593-025-00524-3

OECD. (2023). Skills in Latin America: Insights from the Survey of Adult Skills (PIAAC). OECD Publishing. https://doi.org/10.1787/5ab893f0-en

Pérez, J. (2021). Percepción de estudiantes universitarios sobre el pensamiento computacional. REDU. Revista de Docencia Universitaria, 19(1), 111–127. https://doi.org/10.4995/redu.2021.15491

Prado Ortega, M. X., Paucar Córdova, R. J., Valarezo Castro, J. W., Acosta Yela, M. T., & Guaicha Soriano, K. M. (2023). Beneficios de la programación por bloques utilizando Sphero mini mediante aprendizaje móvil en la educación superior. e-Ciencias de la Información, 13(2), 73–96. https://doi.org/10.15517/eci.v13i2.54814

Sinésio Ferris da Silva, I., & Pontual Falcão, T. (2021). Uma pesquisa documental sobre o pensamento computacional no ensino superior: análise dos projetos pedagógicos dos cursos de licenciatura em computação no Brasil. Revista Contexto & Educação, 36(114), 54–71. https://doi.org/10.21527/2179-1309.2021.114.54-71

Tikva, C., & Tambouris, E. (2021). A systematic mapping study on teaching and learning computational thinking through

programming in higher education. Thinking Skills and Creativity, 41, 100849. https://doi.org/10.1016/j.tsc.2021.100849