Universidad Nacional de Costa Rica
Facultad de Ciencias Exactas y Naturales
Escuela de Informática

Replanteamiento de la Enseñanza de Programación Inicial ante la irrupción de la IA Generativa

Un análisis exploratorio del caso CS1_CS2_AI

Asignatura: EIF400 Paradigmas de Programación II-2025
Profesor: Dr. Carlos Loría-Sáenz

Integrantes:

David Alberto Núñez Franco
Amanda Zamora Ramírez
Mariana Villalobos Ramírez
Sofía Lizano Alfaro
Grupo 01-6pm

Noviembre 2025

Resumen (Abstract)

El presente estudio explora el impacto de la Inteligencia Artificial generativa, particularmente los Modelos de Lenguaje Grandes (LLMs), en la enseñanza de la programación inicial universitaria, representada por los cursos CS1 y CS2. Su objetivo es analizar cómo estas tecnologías influyen en los objetivos de aprendizaje tradicionales, las competencias esenciales y las estrategias de evaluación de dichos cursos. Metodológicamente, se adoptó un enfoque exploratorio comparativo, analizando programas locales (UNA, UCR y TEC) y referencias internacionales (ACM/IEEE, UNSW), junto con literatura académica sobre enseñanza de programación y pedagogía asistida por IA. Los hallazgos indican que los LLMs pueden automatizar tareas básicas como la generación de código o la explicación sintáctica, pero no sustituyen la comprensión algorítmica, el razonamiento lógico ni la creatividad en la resolución de problemas. Además, se identificó que la integración responsable de la IA puede enriquecer la retroalimentación, promover el aprendizaje adaptativo y fortalecer la enseñanza práctica si se acompaña de una orientación ética y pedagógica adecuada. Se concluye que la IA generativa debe considerarse un complemento formativo y no un reemplazo del proceso de enseñanza-aprendizaje, impulsando así una actualización curricular que preserve el pensamiento crítico y el desarrollo humano en la educación en computación.

Palabras clave: IA generativa, LLMs, enseñanza de programación, CS1, CS2.

Contents

1 Introducción

Durante las últimas décadas, los cursos CS1 y CS2, correspondientes a la enseñanza inicial de la programación universitaria, han sido reconocidos como una de las etapas más desafiantes en la formación de los futuros profesionales en informática. Diversos estudios —incluido el de *Hidalgo-Suárez et al.* (2021)— evidencian que en estos cursos se registran altas tasas de reprobación y deserción, resultado de la complejidad inherente al aprendizaje del pensamiento algorítmico, la abstracción y la traducción de problemas reales en soluciones computacionales. Factores como la heterogeneidad de los estudiantes, las limitaciones en el acompañamiento docente y la naturaleza progresiva de los conceptos de programación contribuyen a que CS1 sea considerado, en muchas universidades, un curso con "alto índice de mortalidad académica".

Paralelamente, la irrupción de la Inteligencia Artificial (IA) generativa, y en especial de los Modelos de Lenguaje Grandes (LLMs) como ChatGPT, GitHub Copilot y Gemini, ha transformado la manera en que las personas aprenden y producen conocimiento. Estas herramientas son capaces de generar código, explicar errores, diseñar algoritmos e incluso proponer soluciones a problemas complejos en cuestión de segundos. En este nuevo contexto, la enseñanza tradicional de la programación se ve desafiada: los estudiantes pueden apoyarse en la IA para realizar tareas que antes requerían esfuerzo cognitivo sostenido, mientras los docentes deben redefinir la manera de evaluar el aprendizaje y promover el desarrollo de competencias auténticas que trasciendan la mera ejecución automática de código.

El objetivo principal de esta investigación es analizar de manera exploratoria cómo la IA generativa está impactando los cursos CS1 y CS2, identificando qué objetivos de aprendizaje, competencias y estrategias pedagógicas se ven afectados, cuáles deben preservarse sin mediación tecnológica, y qué nuevas habilidades deberían incorporarse al currículo para responder a este entorno educativo en transformación.

La importancia de este estudio radica en que ofrece una base conceptual y comparativa para orientar el rediseño curricular de los cursos introductorios de programación, tanto en el contexto costarricense (UNA, UCR, TEC) como internacional. Comprender el papel de la IA generativa permitirá desarrollar programas más adaptativos, éticos y centrados en el pensamiento crítico, la creatividad y la autonomía del estudiante. De esta forma, la IA no se concibe como una amenaza a la enseñanza tradicional, sino como una oportunidad para replantear las metodologías, actualizar las competencias y fortalecer la calidad del aprendizaje en los primeros niveles de la educación en ciencias de la computación.

2 Marco de referencia conceptual

El presente marco conceptual se sustenta en literatura académica y se centra en dos ejes principales. En primer lugar, se abordan los cursos introductorios de programación universitaria, comúnmente denominados CS1 y CS2, describiendo sus objetivos clásicos y la estructura típica de sus contenidos tanto a nivel local como internacional. En segundo lugar, se presentan los fundamentos de la Inteligencia Artificial (IA) generativa, con énfasis en los modelos de lenguaje de gran tamaño (Large Language Models, LLMs), dada su relevancia en la informática y su potencial en la enseñanza de la programación. A continuación, se desarrollan ambos aspectos en detalle en las subsecciones correspondientes.

La literatura en educación en computación indica que el curso CS1 (Introducción a la Programación) tiene como finalidad que los estudiantes adquieran habilidades básicas de programación y resolución de problemas lógicos mediante el uso de un lenguaje de programación. Es un curso fundamental en carreras de informática o sistemas, pues sienta las bases de pensamiento algorítmico y comprensión de conceptos como variables, estructuras de control y datos básicos. Típicamente, CS1 introduce un paradigma de programación (por ejemplo, funcional u orientado a objetos) y busca que el estudiante desarrolle la capacidad de traducir problemas del mundo real en soluciones computacionales. Según Hidalgo-Suárez et al. (2021), en este nivel inicial "los estudiantes deben adquirir habilidades básicas para aplicar un paradigma de programación en la solución de problemas lógicos y de abstracción, mediante el uso de un lenguaje de programación". Al completar CS1 con éxito, el alumno domina los fundamentos suficientes para avanzar al curso CS2, usualmente dedicado a estructuras de datos y algoritmos básicos. De hecho, en los lineamientos curriculares internacionales se define CS2 como la continuación natural de CS1, enfocada en profundizar en abstracciones de datos y técnicas algorítmicas más avanzadas. En otras palabras, la secuencia CS1-CS2 tradicional cubre desde la introducción a la programación hasta conceptos como estructuras de datos (listas, pilas, colas, árboles, etc.) y algoritmos asociados, formando el núcleo de la formación inicial en computación. Es importante señalar que estos cursos suelen presentar desafíos significativos: se han reportado altos índices de reprobación y deserción en CS1 en diversas instituciones, lo cual sugiere la necesidad de fortalecer las estrategias pedagógicas empleadas. Esto motiva la exploración de mejoras curriculares y metodológicas (por ejemplo, enfoques colaborativos, flipped classroom, entre otros) para lograr que más estudiantes tengan éxito en su primera aproximación a la programación.

Por otra parte, la IA generativa se refiere a la categoría de técnicas de inteligencia artificial capaces de producir contenido original de forma autónoma, ya sea texto, código, imágenes u otros formatos, a partir de modelos entrenados con grandes volúmenes de datos. En el

contexto de esta investigación, interesa particularmente la generación de texto (y código) mediante modelos de lenguaje. Los avances recientes en aprendizaje profundo han dado lugar a modelos de lenguaje de gran tamaño (LLMs) sumamente poderosos. Un LLM es una red neuronal de alta complejidad (con cientos de millones o incluso miles de millones de parámetros) entrenada sobre volúmenes masivos de texto, de modo que aprende las regularidades del lenguaje y puede generar texto coherente y contextual en respuesta a peticiones en lenguaje natural. Tecnológicamente, la mayoría de estos modelos emplean la arquitectura de transformadores, la cual incorpora mecanismos de auto-atención que les permiten capturar relaciones complejas en secuencias de texto. Gracias a ello, los LLM modernos exhiben una asombrosa flexibilidad: con un mismo modelo es posible responder preguntas, resumir documentos, traducir entre idiomas, mantener conversaciones e incluso generar código fuente a partir de descripciones, todo con ajustes mínimos o indicaciones adecuadas por parte del usuario. Un ejemplo emblemático es el modelo GPT-3 de OpenAI, que con 175 mil millones de parámetros puede producir texto en lenguaje natural de manera extremadamente fluida; su sucesor, ChatGPT, ha popularizado el uso de LLMs al demostrar la capacidad de mantener diálogos coherentes, generar respuestas detalladas y asistir en tareas de programación. Por lo que, los LLMs constituyen la base de la IA generativa textual moderna, y su rápida evolución está comenzando a influir en diversos dominios, incluida la educación en ingeniería y ciencias de la computación. Comprender sus fundamentos es crucial para evaluar cómo podrían integrarse estas herramientas en la enseñanza de la programación, mejorando el aprendizaje de los estudiantes de CS1/CS2 o brindando apoyo en la resolución de problemas y la generación de código.

En síntesis, este marco referencial conecta dos ámbitos: la enseñanza inicial de la programación, con sus objetivos formativos clásicos y desafíos bien documentados, y la tecnología emergente de los modelos de IA generativa. Esta vinculación servirá de base para nuestro estudio, ya que nos permite fundamentar qué se espera que aprendan los estudiantes en cursos CS1/CS2 y cómo las nuevas herramientas basadas en LLMs podrían apoyar o transformar dicho proceso de aprendizaje.

2.1 CS1 y CS2: objetivos clásicos y estructura típica

2.1.1 Propósito general

En los planes de estudio internacionales, CS1 introduce el pensamiento algorítmico y las bases de la programación: modelado de problemas, estructuras de control, tipos de datos básicos, pruebas iniciales y depuración. CS2, como continuación, profundiza en estructuras de datos y algoritmos, consolidando el uso de abstracciones, diseño de soluciones más com-

plejas y análisis de eficiencia. Históricamente, esta secuencia CS1→CS2 ha sido reconocida como el núcleo inicial del currículo en computación, abarcando algoritmos, programación y estructuras de datos fundamentales. En esencia, CS1 sienta las bases de la programación, mientras CS2 expande esas bases hacia el manejo eficiente de datos y la resolución de problemas más sofisticados.

2.1.2 Objetivos típicos de CS1 (síntesis)

Los objetivos formativos clásicos de un curso CS1 suelen incluir:

- Comprender el modelo de ejecución de programas y los fundamentos del lenguaje de programación empleado (p. ej., cómo la computadora interpreta instrucciones y maneja la memoria).
- Analizar problemas y diseñar soluciones elementales, a menudo introduciendo metodologías como diseño orientado a objetos (objects-first) o la programación imperativa tradicional (imperative-first).
- Implementar las soluciones y realizar pruebas básicas, escribiendo código en un IDE, ejecutando casos de prueba sencillos y depurando errores; todo ello aplicando buenas prácticas de programación (legibilidad, documentación, estilo de código, etc.).
- Manipular estructuras de datos básicas (principalmente arreglos; en algunos casos también introducir estructuras dinámicas simples como listas enlazadas o árboles binarios) e implementar algoritmos clásicos de búsqueda y ordenamiento.

Por ejemplo, en la Universidad de Costa Rica, el curso Programación 1 (CI-0112) presenta objetivos acordes a este perfil: resolución de problemas con paradigma OO, implementación y pruebas básicas, uso de un IDE y depurador, aplicación de buenas prácticas, además de la manipulación de estructuras de datos simples y algoritmos básicos.

2.1.3 Estructura típica de CS1 (contenidos y evaluación)

Contenidos Los contenidos de un curso CS1 abarcan la sintaxis y semántica del lenguaje de programación escogido, incluyendo tipos de datos, variables, operadores y expresiones. Se cubren los fundamentos de control de flujo (secuenciación lineal, condicionales y bucles) y la modularidad mediante funciones o métodos. También, introduce la programación estructurada hacia la programación orientada a objetos básica (definición de clases, uso de objetos) y nociones de recursión. Muchas veces se presentan estructuras de datos básicas: arreglos (unidimensionales y multidimensionales) y, en cursos más ambiciosos, primeras nociones de

estructuras dinámicas (como listas enlazadas o árboles binarios). Igualmente, se enseñan algoritmos simples de búsqueda (secuencial, binaria) y ordenamiento (burbuja, selección, inserción), junto con la discusión de su corrección y, si el nivel lo permite, su eficiencia. Esta base de contenidos prepara al estudiante en los elementos esenciales para programar y entender problemas computacionales sencillos.

Evaluación La evaluación en CS1 combina prácticas de programación frecuentes con exámenes prácticos. Es común el uso de laboratorios y proyectos cortos durante el semestre, con retroalimentación por parte del profesor, para que el estudiante gane experiencia y confianza. Se fomenta el trabajo colaborativo, ya que discutir y programar en equipo puede mejorar la comprensión. De hecho, un estudio reportó que integrar actividades colaborativas junto con evaluación del código en un curso CS1 redujo el tiempo promedio para resolver tareas y aumentó las calificaciones en un 50%, reforzando las competencias de programación de los estudiantes. Esto evidencia que estrategias pedagógicas (como programar en equipo, usar jueces automáticos de programas, etc.) no solo incrementan la práctica sino que también mejoran el rendimiento y desarrollan habilidades interpersonales importantes para la disciplina. De acuerdo a lo anterior, la evaluación en CS1 suele centrarse en la práctica constante y la retroalimentación inmediata, para afianzar los conceptos básicos y superar las dificultades típicas de los aprendices de programación.

2.1.4 Objetivos típicos de CS2 (síntesis)

En el curso CS2, los objetivos se orientan a profundizar las habilidades de diseño y análisis algorítmico:

- Diseñar e implementar estructuras de datos fundamentales, como listas enlazadas, pilas, colas, árboles y, en muchos programas, grafos básicos. Esto incluye escribir las operaciones asociadas (inserción, eliminación, recorrido, búsqueda) y entender en qué escenarios cada estructura es adecuada.

- Aplicar principios de abstracción y tipos abstractos de datos (ADT) en la resolución de problemas. El estudiante aprende a separar la interfaz de una estructura de datos de su implementación, fomentando un pensamiento más modular. Asimismo, se introduce el análisis de la eficiencia computacional mediante la notación Big-O, para evaluar el costo temporal y espacial de los algoritmos. Temas como la recursión avanzada y gestión de memoria dinámica (punteros, asignación/liberación) suelen afianzarse en este nivel.

- Fortalecer pruebas y depuración de código, ahora con programas más complejos. Se espera que el estudiante escriba pruebas unitarias más exhaustivas y utilice técnicas

de depuración para estructuras de datos dinámicas (p.ej., seguimiento de punteros, detección de memory leak).

- Preparar al estudiante para cursos superiores en la carrera (algoritmos avanzados, arquitectura de computadores, desarrollo de software, etc.). CS2 sienta así la base teórica-práctica para entender estructuras y algoritmos más complejos. Tradicionalmente, los lineamientos curriculares han señalado a CS2 como el espacio formal para dominar estructuras de datos y algoritmos, completando la formación iniciada en CS1. En algunas universidades, CS2 es sinónimo de "Estructuras de Datos y Algoritmos". Incluso es común que se introduzcan nociones de grafos (búsquedas BFS/DFS u otros algoritmos simples) hacia el final del curso si el tiempo lo permite, dando una vista de temas que se verán con mayor profundidad en cursos posteriores de algoritmos.

2.2 Fundamentos de la IA generativa (Modelos de Lenguaje Grandes – LLMs)

La Inteligencia Artificial generativa es un campo de la IA enfocado en la creación de contenido nuevo a partir de patrones aprendidos. A diferencia de los sistemas discriminativos (que se limitan a clasificar o reconocer patrones existentes), un sistema generativo puede producir datos: por ejemplo, generar texto, imágenes, audio o código, en respuesta a una solicitud del usuario. Estos modelos aprenden las estructuras y regularidades presentes en sus datos de entrenamiento y luego utilizan ese conocimiento para originar contenido que mantiene características similares a los datos originales. En sus inicios, la IA generativa buscaba imitar procesos creativos humanos, y hoy en día ha logrado resultados impresionantes en múltiples dominios (desde la redacción de artículos o código fuente, hasta la creación de obras de arte visual o música). Ejemplos destacados de IA generativa son los modelos de difusión de imágenes (p.ej. DALL-E) y, relevante en el contexto de la computación, los modelos de lenguaje como GPT-4, que pueden entablar conversaciones, responder preguntas y producir texto coherente de gran calidad.

Dentro de la IA generativa, una categoría son los LLMs (Large Language Models). Un LLM es un tipo de modelo de IA entrenado con cantidades masivas de datos textuales, diseñado para comprender y generar lenguaje natural de manera cercana a como lo haría un humano. Estos modelos han ganado enorme notoriedad recientemente por su papel en llevar la IA generativa al uso cotidiano, gracias a aplicaciones como los chatbots tipo ChatGPT, que demuestran sus capacidades al gran público. Sin embargo, es importante destacar que los LLMs son el resultado de años de avances en machine learning, en especial en subcampos como el procesamiento de lenguaje natural (PLN) y las redes neuronales profundas. En

particular, la introducción de la arquitectura de Transformers (propuesta por Vaswani et al. en 2017) marcó una división en la construcción de modelos de lenguaje, permitiendo escalar su entrenamiento de forma efectiva. Modelos como GPT-3/GPT-4 de OpenAI, LLaMA, PaLM, son todos LLMs basados en variantes de arquitecturas transformer.

2.2.1 ¿Cómo funcionan los LLMs?

En esencia, aprenden a predecir la siguiente palabra en una secuencia de texto dado un contexto previo. Durante el entrenamiento, se les proporcionan millones de frases y párrafos (obtenidos de libros, artículos, páginas web, código, etc.) y el modelo ajusta sus parámetros internos para estimar cuál sería la palabra (o token) más probable que sigue a una secuencia de palabras dada. Este entrenamiento se realiza de forma auto-supervisada, ya que el propio texto provee las etiquetas (la siguiente palabra) para cada contexto posible, eliminando la necesidad de datos manualmente etiquetados. Los LLM modernos se entrenan con corpus masivos que abarcan miles de millones de palabras, lo que les permite captar patrones estadísticos del lenguaje a gran escala: aprenden gramática, semántica, hechos del mundo, e incluso "sentido común" básico expresado en el texto. Un aspecto clave es el uso del mecanismo de atención en las redes transformer, el cual permite al modelo enfocarse en las partes relevantes del contexto para predecir la siguiente palabra. En la práctica, tras el entrenamiento, un LLM puede tomar un prompt del usuario y generar una respuesta palabra por palabra, eligiendo cada término sucesivo según las probabilidades aprendidas, que aseguran coherencia y relevancia con respecto al contexto dado. El resultado es que el modelo puede producir texto fluido en lenguaje natural, manteniendo la cohesión temática e incluso imitando estilos si así lo sugiere el prompt.

Seguidamente, se sintetizan algunas capacidades y características típicas de los LLMs:

- **Comprensión del lenguaje natural:** Los LLM entienden instrucciones y contexto en lenguaje humano, pudiendo seguir indicaciones dadas en texto de manera flexible (e.g., entender una pregunta y extraer la respuesta adecuada del conocimiento implícito en sus parámetros).

- **Generación de texto coherente:** Pueden producir oraciones y párrafos que respetan reglas gramaticales y sentido lógico, continuando una idea o conversación de forma natura. Esto incluye adaptarse a distintos estilos o tonos si se les indica (formal, coloquial, técnico, creativo, etc.).

- **Polivalencia en tareas de PLN:** Un solo modelo es capaz de realizar múltiples tareas de procesamiento de lenguaje: traducción, resumen, clasificación temática, respuesta a

preguntas, corrección ortográfica/gramatical y más, sin estar especializado únicamente en una de ellas. Esta versatilidad proviene de su amplio entrenamiento, a diferencia de modelos más pequeños entrenados para tareas específicas.

- **Aprendizaje contextual y zero-shot — few-shot:** Los LLM pueden adaptarse a nuevas instrucciones o formatos. Incluso sin entrenamiento adicional específico (zero-shot), pueden intentar resolver una tarea nueva usando solo la indicación del usuario. Mejora su desempeño si se les dan uno o pocos ejemplos en el prompt (few-shot learning), demostrando cierta capacidad de generalización por contexto.

- **Generación de código y contenido estructurado:** Además de lenguaje humano, muchos LLM han visto ejemplos de código fuente y pueden generar código en lenguajes de programación (p. ej. Python, Java) a partir de descripciones, o escribir marcado HTML/JSON estructurado, lo que los hace útiles como asistentes de programación.

- **Escalabilidad con desempeño mejorado:** Mientras más grande el modelo (más parámetros y más datos de entrenamiento), mejor tiende a ser su desempeño y capacidad para capturar sutilezas del lenguaje. Esto ha llevado a una carrera por construir modelos cada vez más grandes (GPT-2 tenía 1.5 millardos de parámetros, GPT-3 175 millardos, GPT-4 se estima aún mayor, etc.). Sin embargo, también aumenta el costo computacional y energético de entrenarlos y ejecutarlos.

  A pesar de sus impresionantes habilidades, los LLMs tienen limitaciones y consideraciones importantes. Primero, no "entienden" el mundo de la misma forma que un humano; operan sobre correlaciones estadísticas del texto. Por ello, pueden cometer errores factuales con mucha seguridad (alucinaciones), generar respuestas que suenan elogiables pero son incorrectas o incluso contradictorias. Para mitigar esto, a menudo se aplica un ajuste fino (fine-tuning) adicional con datos curados e incluso técnicas de aprendizaje con refuerzo a partir de retroalimentación humana (RLHF) para alinearlos con comportamientos deseados (por ejemplo, que sigan instrucciones específicas, eviten sesgos o contenido inapropiado). OpenAI, por ejemplo, empleó RLHF para mejorar la calidad y seguridad de ChatGPT, guiando al modelo a rechazar solicitudes indebidas y reducir la toxicidad en sus respuestas. Otra consideración es la preocupación ética y educativa: dado que un LLM puede generar textos completos y resolver problemas, surge la necesidad de usarlos responsablemente, evitando trampas académicas y manejando correctamente la autoría. De hecho, organismos como la UNESCO han llamado a establecer directrices claras para el uso de IA generativa en educación, promoviendo que estas herramientas se utilicen de forma inclusiva, equitativa y ética.

2.3 Impacto pedagógico de herramientas como ChatGPT, Copilot

La incorporación de asistentes de código que se basan en modelos de lenguaje (LLMs) como ChatGPT o Github Copilot ha tenido un gran impacto en en cursos introductorios de programación. Un estudio realizado por Peng, Kalliamvakou et al, publicado en febrero del 2023, menciona que la IA presenta un aumento en la productividad humana. Este estudio midió task success y task completion time, el éxito de la tarea se midió como el porcentaje de participantes en un grupo que sí completaron la tarea correctamente, y el tiempo de finalización de la tarea se midió como el tiempo transcurrido desde el inicio hasta el final de la tarea. Al finalizar el estudio se mostró que el uso de Copilot redujo el tiempo de finalización de tareas en un 55,8% sin disminuir la calidad del código, aunque se incrementó la probabilidad de errores sin detectar. Más adelante muestra los resultados en porcentajes de cada grupo del estudio, y se concluye que hay una mejor en la productividad pero también existe la necesidad de reforzar la verificación y el juicio crítico.

En el ámbito educativo, Becker et al. (2023) alegan que los LLMs pueden servir como estructuras cognitivas que apoyan la generación de ejemplos, explicaciones y de retroalimentación más personalizada en CS1/CS2. No obstante, de igual forma advierten que el uso sin mediación de un docente, puede descalzar y debilitar la comprensión conceptual y fomentar la dependencia tecnológica. Adicionalmente, la UNESCO (2023) recomienda que estas herramientas se integren sólo dentro de marcos curriculares que incluyan ética, transparencia y evaluación humana.

Por ende, el impacto pedagógico de estas herramientas por un lado potencian la exploración y el aprendizaje, pero a la vez exige rediseñar actividades y evaluaciones para preservar la comprensión profunda y la capacidad de razonamiento.

2.4 Teorías del aprendizaje relevantes: constructivismo, aprendizaje activo, evaluación auténtica

El constructivismo de acuerdo a McLeod (2025), es tanto una teoría del aprendizaje como una filosofía de la educación. Pues menciona que el constructivismo propone que los estudiantes construyan su conocimiento a través de experiencias e interacciones, ya que de acuerdo a esa teoría la educación debe centrarse en la resolución de problemas y el pensamiento crítico, además hace énfasis en que el aprendizaje se centra en el estudiante y el docente debe guiar en lugar de dirigir. En educación en ingeniería y computación, el aprendizaje activo tiene un enorme impacto, y según el análisis de Freeman et al (2014) en PNAS, se muestra que el aprendizaje activo conlleva a mejoras en el rendimiento en los exámenes y que las tasas de fracaso con la enseñanza tradicional aumentan un 55% con

respecto a lo observado con el aprendizaje activo.

Además la evaluación auténtica según Wiggins (1998), busca que se evidencie el desempeño realista y la transferencia de conocimiento por medio de tareas complejas, rúbricas y retroalimentación formativa, y que las evaluaciones no se enfoquen sólo en auditar. En conjunto, un enfoque constructivista y activo con la evaluación auténtica fomenta que los estudiantes no solo "usen" las herramientas de IA, sino que también comprendan y evalúen de forma crítica su funcionamiento, y que así se pueda alinear la práctica enfocada en la comprensión, la creatividad y la ética profesional.

2.5 Ética y evaluación en contextos con IA

La integración de la inteligencia artificial en la educación conlleva desafíos éticos y evaluativos. El ACM Code of Ethics (2018) exige transparencia, honestidad y responsabilidad sobre el uso de estas herramientas automatizadas. De la misma manera la UNESCO (2023) destaca la importancia y la obligación de proteger la privacidad y de asegurar la equidad de acceso, de forma que se eviten sesgos.

En el ámbito educativo, Cotton et al. publicaron un artículo en marzo del 2023, en el cual subrayan que el uso indiscriminado de ChatGPT puede provocar prácticas de plagio inadvertido, y pérdida de aprendizaje significativo. Por esa razón, recomiendan políticas claras y ejercicios de evaluación que evidencien la comprensión personal. Adicionalmente, de acuerdo a Kasneci et al. (2023), la evaluación debería combinar mediciones sin IA y otras con IA regulada. Las mediciones sin IA pueden ser razonamiento manual, trazado de ejecución o análisis de errores, y las mediciones con IA regulada como proyectos, bitácoras de prompts o defensas orales, de manera que se pueda garantizar la integridad académica y el desarrollo de competencias profesionales.

Por lo tanto, la ética y la evaluación convergen en la transparencia (al declarar cuándo y cómo se usa la IA), la autoría y responsabilidad (diferencias el aporte del estudiante) y finalmente, la equidad y seguridad (uso sin exposición de datos). Puesto que la aplicación coherente de estos permitirá que la IA sea una herramienta y no un sustituto del pensamiento crítico.

2.6 Inteligencia Híbrida: un marco para la colaboración humano-IA en CS1/CS2

Mientras que el debate educativo sobre IA generativa se ha centrado en si estas herramientas "reemplazan" el aprendizaje, el concepto de Inteligencia Híbrida (HI) propuesto por Akata et al. (2020) ofrece un marco alternativo más productivo. Los autores definen

HI como "la combinación de inteligencia humana y de máquina, aumentando el intelecto y las capacidades humanas en lugar de reemplazarlas, para alcanzar objetivos que serían inalcanzables por humanos o máquinas por separado" (p. 19). Este enfoque no concibe a los LLMs como amenazas o atajos, sino como prótesis cognitivas que, correctamente integradas, pueden potenciar el desarrollo de competencias fundamentales en programación.

Akata et al. (2020) identifican cuatro dimensiones críticas para sistemas HI efectivos: colaborativa (trabajo sinérgico entre humano y máquina explotando sus habilidades complementarias), adaptativa (aprendizaje mutuo y ajuste continuo), responsable (comportamiento ético alineado con valores humanos) y explicable (capacidad de justificar razonamiento y decisiones). Aplicadas a CS1/CS2, estas dimensiones sugieren que el estudiante debe ser responsable del diseño algorítmico y la validación crítica, mientras la IA asiste en tareas operativas como generación de sintaxis o casos de prueba. Como señalan los autores, "necesitamos enfoques de resolución de problemas cooperativos donde las máquinas y humanos contribuyan a través de una conversación colaborativa" (p. 19), lo cual es precisamente lo que debe modelarse en los cursos introductorios: no el uso pasivo de herramientas, sino la coordinación cognitiva deliberada entre pensamiento humano y capacidades computacionales.

Este marco de HI refuerza las recomendaciones emergentes en educación en computación. Becker et al. (2023) sostienen que los estudiantes deben "pensar como programadores antes de actuar como usuarios de IA", y la UNESCO (2023) llama a integrar estas tecnologías dentro de marcos éticos y pedagógicos claros. La perspectiva de inteligencia híbrida unifica ambas posturas: la pregunta no es si permitir o prohibir LLMs en CS1/CS2, sino cómo diseñar experiencias de aprendizaje donde la IA amplifique — sin sustituir— el desarrollo del pensamiento algorítmico, la creatividad en resolución de problemas y la capacidad de juicio crítico que caracterizan al profesional en computación. Esta investigación explora precisamente ese desafío en el contexto costarricense.

3 Desarrollo: análisis de preguntas generadoras

3.1 ¿Qué objetivos de aprendizaje se ven afectados?

La Inteligencia Artificial Generativa (GenAI) está transformando las prácticas de enseñanza y aprendizaje de la programación mediante el uso de modelos de lenguaje capaces de generar, explicar y corregir código. La incorporación de herramientas de inteligencia artificial generativa, como Chat GPT o GitHub Copilot, ha transformado de manera significativa los objetivos de aprendizaje tradicionales en los cursos CS1 y CS2, centrados históricamente en la producción manual de código y la comprensión de los fundamentos sintácticos de los

lenguajes de programación.

Becker et al. (2023) sostienen que los modelos de lenguaje grandes ya pueden resolver la mayoría de los ejercicios típicos de los cursos introductorios, reduciendo el valor pedagógico de enseñar tareas mecánicas como la escritura de sintaxis correcta, la traducción directa de pseudocódigo a código fuente o la identificación básica de errores de compilación. Sin embargo, en áreas más avanzadas como sistemas distribuidos, ingeniería de software o interacción humano–computadora los estudiantes deben enfrentar retos que exigen síntesis de arquitecturas complejas, optimización del rendimiento y resolución de dilemas éticos, los cuales demandan habilidades de orden superior que superan ciertas capacidades de la IA generativa.

De acuerdo con Alanazi et al. (2025), los estudiantes que emplean herramientas de IA completan tareas más rápido y con mejores resultados en los laboratorios, pero no evidencian mejoras proporcionales en la comprensión conceptual ni en el diseño algorítmico, lo que sugiere que los objetivos de bajo nivel (recordar reglas sintácticas, aplicar estructuras de control, repetir patrones de código o copiar soluciones modelo) son los más afectados y pueden ser automatizados por los sistemas de IA.

Por el contrario, los objetivos de orden superior, como la resolución de problemas no rutinarios, el diseño de algoritmos eficientes, la abstracción de estructuras de datos y la evaluación de la corrección lógica y ética del código, siguen siendo no automatizables y dependen del pensamiento humano (Agbo et al., 2025). Zviel-Girshin (2024) confirma que los estudiantes usan la IA principalmente para comentar código (91.7 %) y corregir errores (80.2 %), lo que demuestra que las tareas operativas y repetitivas se ven desplazadas por la automatización, mientras que las metas asociadas al razonamiento algorítmico y a la creatividad computacional permanecen como pilares irremplazables.

Vera et al. (2025) advierten que, si los cursos no equilibran la automatización con el pensamiento reflexivo, la IA puede conducir a una "comprensión superficial de los conceptos". Lo que puede afectar objetivos esenciales como el análisis de eficiencia algorítmica, la comprobación de resultados mediante prueba de escritorio y la capacidad de depuración intencional. En conformidad, *Guidance on the Future of Computer Science Education in an Age of AI* (2025) recomienda redefinir los objetivos de CS1/CS2 para priorizar la evaluación crítica del código asistido, la interpretación semántica de resultados generados por IA y la integración ética y responsable de estas tecnologías en el aprendizaje.

Desde la experiencia educativa, también se observa que el uso intensivo de herramientas de IA puede reducir las interacciones sociales y emocionales entre estudiantes y docentes, lo que debilita el desarrollo de habilidades blandas como la comunicación, la empatía y el trabajo colaborativo. Estas competencias, fundamentales para el ejercicio profesional en

ingeniería, deberían preservarse mediante actividades presenciales, debates y proyectos en equipo que mantengan el componente humano del proceso formativo.

En conclusión, los objetivos de aprendizaje afectados por la IA generativa pueden clasificarse en aquellos que son automatizables (serán afectados directamente), como la escritura de sintaxis, la detección de errores de compilación, la generación de ejemplos o plantillas y la documentación o comentarios; y los no automatizables (se deben conservar), como el razonamiento algoritmico, diseño de soluciones, el análisis de eficiencia y complejidad, interpretación de errores, creatividad, evaluación crítica y desarrollo de habilidades sociales y éticas. Por lo que el desafío pedagógico consiste en redefinir el equilibrio entre ambos tipos de objetivos, utilizando la IA como apoyo para automatizar lo operativo, sin comprometer el desarrollo de las capacidades cognitivas, críticas y éticas que caracterizan la formación en computación.

3.2 ¿Qué competencias deberían mantenerse sin IA?

A pesar de todos los beneficios que ofrecen las herramientas de inteligencia artificial, la mayoría de las investigaciones coinciden en que hay competencias humanas que no se pueden delegar. Becker et al. (2023) lo dicen de forma directa: los estudiantes deben seguir pensando como programadores antes de actuar como usuarios de IA. En otras palabras, la tecnología puede ayudar, pero no debería reemplazar el proceso mental de analizar un problema, imaginar una solución y luego traducirla en código. Si no se refuerza ese razonamiento algorítmico desde los primeros cursos, se corre el riesgo de formar usuarios de herramientas en lugar de ingenieros capaces de entender lo que están creando.

Zviel-Girshin (2024) subraya la necesidad de mantener viva la práctica de la depuración manual. Muchos estudiantes se acostumbran a que el asistente de IA les diga dónde está el error, pero pierden la capacidad de observar con calma la lógica detrás del código. Entender por qué algo falla es lo que al final enseña, más que ver el resultado correcto. Vera et al. (2025) también advierten sobre esto: cuando el aprendizaje depende demasiado de los asistentes, la confianza en la propia capacidad para resolver empieza a desaparecer, y con ella la autonomía cognitiva que define a un buen ingeniero.

Alanazi et al. (2025) fueron más concretos en sus hallazgos. En sus estudios, los alumnos que usaban IA terminaban las tareas más rápido y con menos errores sintácticos, pero su creatividad y comprensión profunda del código no mejoraban. Esto refuerza la idea de que la IA puede ser una gran ayuda operativa, aunque el pensamiento lógico, el diseño de algoritmos y la validación de resultados siguen necesitando la mente humana, con toda su paciencia y capacidad de ensayo y error.

Además, Agbo et al. (2025) y *Guidance on the Future of Computer Science Education* (2025) llaman la atención sobre algo que suele quedar al margen: la ética profesional. Los cursos introductorios deberían seguir enseñando a los estudiantes a pensar en las consecuencias de su código, a reconocer los sesgos de la IA y a usar la información digital de manera responsable. Son habilidades que no se automatizan ni se aprenden con un clic, pero que marcan la diferencia entre un programador técnico y uno verdaderamente consciente de su impacto social.

En conclusión, las competencias que deben mantenerse al margen de la IA son, precisamente, las que nos hacen humanos frente a la automatización, tales como el razonamiento lógico, la depuración manual, la comprensión de errores, la creatividad en el diseño de soluciones y la ética profesional. Conservarlas es mucho más que una cuestión académica; es asegurarse de que el estudiante entienda no solo cómo obtener una respuesta usando IA, sino también por qué esa respuesta es correcta, segura y aceptable. Resumiendo, la inteligencia artificial puede escribir el código, pero solo la inteligencia humana puede darle propósito.

4 Caso(s) de estudio

4.1 CS1 - FUNDAMENTOS DE INFORMÁTICA

Table 1: CS1 - FUNDAMENTOS DE INFORMÁTICA

Aspecto	**TEC – IC-1803 (Taller de Programación)**	**UNA – EIF200 (Fundamentos de Informática)**	**UCR – CI-0110 (Introducción a la Computación)**	**Análisis Comparativo**
Código del curso	IC-1803	EIF200	CI-0110	-
Nombre del curso	Taller de Pograma-ción	Fundamentos de Informática	Introducción a la Computación	-
Horas semanales	3 presenciales + 6 extraclase = 9 horas totales	4 contacto (2 teoría/2 práctica) + 6 independientes = 10 horas totales	**5 horas semanales** (distribución no especificada)	Cargas similares, UNA con más horas de contacto

Continúa en la siguiente página

Aspecto	**TEC – IC-1803 (Taller de Programación)**	**UNA – EIF200 (Fundamentos de Informática)**	**UCR – CI-0110 (Introducción a la Computación)**	**Análisis Comparativo**
Enfoque principal	**Herramientas de desarrollo, práctica de programación, ejercicios masivos** (URI Online Judge)	Resolución computacional de problemas con énfasis algorítmico; multi-paradigma	**Orientación profesional:** ¿Qué hace un profesional en computación? ¿Para quién trabaja? ¿Qué conocimientos necesita?	**Diferencia radical:** TEC=práctica técnica; UNA=fundamentos programación; UCR=exploración de carrera
Objetivo general	Construir programas con elementos básicos de ambiente de programación; operar sistema computacional; manejar terminología CS	Desarrollar soluciones algorítmicas y con POO usando estructuras básicas; fomentar actitud investigativa y trabajo colaborativo	Conocer labores del profesional en computación y practicar fundamentos básicos para tomar decisiones informadas sobre futuro profesional	TEC: habilidades técnicas; UNA: resolución de problemas; UCR: orientación vocacional
Herramientas de desarrollo	**Enfoque central:** ambiente de desarrollo, componentes de aplicación, módulos, proyectos	IDE Zinjai (C++), debugger, MS Teams, Aula Virtual	IDE mencionados conceptualmente (no se usa uno específico)	TEC más enfocado en herramientas profesionales; UNA específica Zinjai; UCR teórico

Continúa en la siguiente página

Aspecto	**TEC – IC-1803 (Taller de Programación)**	**UNA – EIF200 (Fundamentos de Informática)**	**UCR – CI-0110 (Introducción a la Computación)**	**Análisis Comparativo**
Metodología	**6 horas magistral** (conferencias, videos) + **2 horas laboratorio** + prácticas semanales de código	Aprendizaje activo + trabajo en equipo + laboratorios guiados + ejercicios en papel y computadora	**Práctica de fundamentos básicos** + charlas motivadoras + trabajos y proyectos prácticos + uso de herramientas modernas	TEC más tradicional; UNA más colaborativa; UCR más expositiva/motivacional

Fuente: Elaboración propia con base en los programas de curso IC-1803 (Instituto Tecnológico de Costa Rica, 2021), EIF200 (Universidad Nacional, 2025) y CI-0110 (Universidad de Costa Rica, s.f.-a). Disponible en https://www.ecci.ucr.ac.cr/cursos/ci-0110

La Tabla 1 presenta una comparación de los cursos CS1 en las tres principales universidades públicas costarricenses, evidenciando diferencias significativas en sus enfoques pedagógicos. Mientras el TEC prioriza la práctica técnica intensiva mediante herramientas profesionales y ejercicios masivos, la UNA se centra en fundamentos algorítmicos con enfoque multi-paradigma, y la UCR adopta una orientación más exploratoria y vocacional. Es notable que ninguno de los tres programas menciona explícitamente la integración de herramientas de IA generativa en sus objetivos, metodologías o evaluaciones.

Esta ausencia curricular es particularmente significativa considerando que, según Becker et al. (2023), los LLMs ya pueden resolver la mayoría de ejercicios típicos de CS1. Las metodologías siguen siendo tradicionales (magistrales, laboratorios guiados, ejercicios de codificación manual), sin contemplar cómo preparar a los estudiantes para evaluar críticamente código generado por IA o usar estas herramientas de manera ética y responsable en su proceso de aprendizaje.

4.2 CS2 - PROGRAMACIÓN 1

Table 2: CS2 - PROGRAMACIÓN 1

Aspecto	**TEC – IC-1802 (Introducción a la Programación)***	**UNA – EIF201 (Programación I)**	**UCR – CI-0112 (Programación 1)**	**Análisis Comparativo**
Código	IC-1802	EIF201	CI-0112	-
Horas semanales	3 presenciales + 6 extraclase = 9 total	4 contacto (2 teoría/2 práctica) + 6 independientes = 10 total	5 semanales (distribución no especificada)	Cargas similares
Lenguaje	Python (capítulos 3-8)	C++ (continuidad con EIF200)	No especificado (implícitamente POO, probablemente C++ o Java)	UNA *especifica* C++ explícitamente
Enfoque principal	Algoritmos + paradigma funcional → imperativo → intro POO	**Profundización POO:** relaciones entre clases, colecciones, persistencia	Resolución de problemas usando POO como forma "natural" desde inicio	UNA avanza en POO; UCR lo introduce; TEC lo menciona al final
Objetivo general	Desarrollar algoritmos básicos y realizarlos sistemáticamente en un modelo computacional específico o lenguaje de programación	Diseñar e implementar con POO las principales técnicas de programación y estructuras de datos para software de pequeña a mediana complejidad	Resolver problemas mediante programación utilizando el paradigma de programación orientado a objetos	TEC: algorítmico-teórico; UNA: diseño e implementación POO; UCR: resolución de problemas con POO

Continúa en la siguiente página

Aspecto	**TEC – IC-1802 (Introducción a la Programación)***	**UNA – EIF201 (Programación I)**	**UCR – CI-0112 (Programación 1)**	**Análisis Comparativo**
Herramientas de desarrollo	No especificadas explícitamente (implícitamente Python IDE)	Cada profesor selecciona IDE (Zinjai, Code::Blocks, etc. típicamente para C++)	Objetivo específico: Utilizar ambiente de desarrollo integrado (IDE), compilador, depurador	UCR único que lista IDE/depurador como objetivo formal; UNA flexible; TEC no especifica
Metodología	Magistral por Zoom + alta participación estudiantil + trabajos grupales + lecturas del libro del profesor + llamadas orales y casos	Magistral con acompañamiento intenso: "estudiante se sienta acompañado en todo momento" + ejercicios prácticos + laboratorios semanales + investigación constructiva + presencialidad obligatoria	Resolución de problemas + diseño e implementación OOP + uso de IDE y depurador + prácticas de laboratorio	TEC: teórico-participativo virtual (2021, pandemia); UNA: presencial con acompañamiento estructurado; UCR: práctico orientado a herramientas

Fuente. Elaboración propia con base en los programas de curso IC-1802 (Instituto Tecnológico de Costa Rica, 2021), EIF201 (Universidad Nacional, 2023) y CI-0112 (Universidad de Costa Rica, s.f.-b). Disponible en https://www.ecci.ucr.ac.cr/cursos/ci-0112

La Tabla 2 compara los cursos CS2, revelando diferentes niveles de profundización en Programación Orientada a Objetos: la UNA avanza significativamente en POO con relaciones entre clases y persistencia; la UCR lo introduce desde el inicio como paradigma natural; y el TEC lo menciona al final del curso. Destaca que solo la UCR establece como objetivo formal el dominio de IDEs y depuradores, mientras las otras instituciones no especifican o dejan flexibilidad al docente para seleccionar herramientas.

Al igual que en CS1, ninguno de estos programas ha incorporado formalmente el uso de

asistentes de código basados en IA (como GitHub Copilot o ChatGPT) ni estrategias de evaluación adaptadas a este contexto. Esta omisión genera un desfase entre la formación universitaria y la práctica profesional real, donde estudios como el de Peng et al. (2023) demuestran incrementos del 55.8% en productividad mediante el uso de estas herramientas. La falta de integración curricular de la IA limita la preparación de los estudiantes para el mercado laboral contemporáneo y desaprovecha oportunidades pedagógicas para desarrollar competencias críticas necesarias en la era de la inteligencia artificial generativa.

5 Conclusiones

El presente estudio permitió analizar el impacto de la Inteligencia Artificial generativa, especialmente los Modelos de Lenguaje Grandes (LLMs), en la enseñanza de los cursos introductorios de programación universitaria CS1 y CS2, identificando los principales retos y oportunidades que emergen de su integración en el proceso educativo. Los resultados reflejan que los LLMs poseen un alto potencial para automatizar tareas operativas y repetitivas del aprendizaje de la programación, tales como la escritura de código, la explicación de sintaxis o la generación de ejemplos. Sin embargo, su uso no sustituye las habilidades cognitivas superiores que caracterizan el pensamiento computacional, como el razonamiento lógico, el diseño algorítmico, la depuración conceptual y la toma de decisiones éticas en el desarrollo de software.

Asimismo, se confirma que la IA generativa puede constituir una herramienta pedagógica de apoyo siempre que su implementación esté acompañada de un marco ético y metodológico claro. Al ser empleada correctamente, permite fortalecer la retroalimentación inmediata, personalizar el aprendizaje y fomentar la autonomía del estudiante, contribuyendo a reducir la frustración y aumentar la motivación en los cursos con mayor índice de reprobación.

En cuanto a las recomendaciones generales, se propone que las instituciones universitarias avancen hacia una actualización curricular progresiva que contemple el uso responsable de la IA generativa. Los programas de CS1 y CS2 deberían incorporar actividades que desarrollen tanto las habilidades técnicas como las metacognitivas, promoviendo el análisis crítico del código generado por IA, la ética profesional y la colaboración entre estudiantes. Además, se sugiere capacitar a los docentes en el uso pedagógico de herramientas de IA, de modo que puedan guiar al estudiantado en su aprovechamiento consciente y formativo.

Finalmente, como líneas futuras de investigación, se recomienda profundizar en el estudio del impacto cuantitativo del uso de LLMs en el rendimiento académico y en la evolución de las estrategias didácticas en contextos reales de aula. También sería pertinente explorar el desarrollo de modelos híbridos de enseñanza, en los cuales la IA actúe como asistente

cognitivo complementario al docente, y analizar los efectos de estas tecnologías sobre la motivación, equidad y ética del aprendizaje en la educación en computación. De esta manera, se podrá avanzar hacia un modelo educativo que integre la innovación tecnológica con la formación integral del estudiante, asegurando que la IA contribuya al desarrollo humano y no a su sustitución.

6 Bibliografía

- ACM. (2018). *ACM Code of Ethics and Professional Conduct*. Association for Computing Machinery. Recuperado de https://www.acm.org/code-of-ethics
- Becker, B. A., Hwa, T., et al. (2024). *Computing Education in the Era of Generative AI*. arXiv preprint arXiv:2401.02358.
- Buckland, R. (2009). *CS2: Data Structures and Algorithms – Course Lectures*. University of New South Wales (UNSW). Recuperado de courses.com (UNSW).
- BM. (s. f.). *Modelos de lenguaje grandes (LLM)*. En IBM Think. Recuperado de https://www.ibm.com/es-es/think/topics/large-language-models
- Cotton, D. R. E., Cotton, P. A., & Shipway, J. R. (2023). ChatGPT, generative AI, and the changing landscape of higher education. *Innovations in Education and Teaching International, 60*(5), 505–516. https://doi.org/10.1080/14703297.2023.2190148
- Escuela de Ciencias de la Computación e Informática, UCR. (s.f.). *Programación 1 (CI-0112)*. Recuperado de https://www.ecci.ucr.ac.cr/cursos/ci-0112
- Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410–8415. https://doi.org/10.1073/pnas.1319030111
- Hemmendinger, D. (s.f.). *The ACM and IEEE-CS Guidelines for Undergraduate CS Curricula: A Historical Perspective*. Recuperado de https://cs.union.edu/~hemmendd/History/curric-hist.pdf
- Hidalgo-Suárez, C. G., Bucheli-Guerrero, V. A., Restrepo-Calle, F., & González-Osorio, F. A. (2021). Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1. *Investigación e Innovación en Ingenierías, 9*(1), 50–60. https://doi.org/10.17081/invinno.9.1.4185

- Kasneci, E., Sessler, K., Kitchens, J., et al. (2023). ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education. *Learning and Individual Differences, 103*, 102274. https://doi.org/10.1016/j.lindif.2023.102274
- Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.* arXiv preprint arXiv:2302.06590.
- Simply Psychology, & McLeod, S. (2025, March 31). *Constructivism Learning Theory & Philosophy of Education.* Simply Psychology. Recuperado de https://www.simplypsychology.org/constructivism.html
- UNESCO. (2023). *Guidance for Generative AI in Education and Research.* United Nations Educational, Scientific and Cultural Organization. Recuperado de https://unesdoc.unesco.org/ark:/48223/pf0000385450
- Wiggins, G. (1998). *Educative Assessment: Designing Assessments to Inform and Improve Student Performance.* Jossey-Bass.
- Agbo, F. J., Oyelere, S. S., Suhonen, J., & Tukiainen, M. (2025). *Computing education in the era of generative AI: A systematic literature review. Computers & Education Open, 9*, 100266. https://doi.org/10.1016/j.caeo.2025.100266
- Alanazi, M., Soh, B., Samra, H., & Li, A. (2025). *The influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. Computers, 14*(5), 185. https://doi.org/10.3390/computers14050185
- Becker, B. A., Denny, P., Luxton-Reilly, A., & Guo, P. J. (2023). *Generative AI in introductory programming education: Opportunities and challenges. ACM Curricular Practices Series.*
- Guidance on the Future of Computer Science Education in an Age of AI. (2025). *Computing Research Association & CSTA Joint Report.*
- Vera, D. A., Franco, O. O., & Córdova, L. C. (2025). *Impacto de la inteligencia artificial en el aprendizaje de la programación informática en los estudiantes universitarios. Ciencia y Educación, 6*(3), 33–38.
- Zviel-Girshin, R. (2024). *The good and bad of AI tools in novice programming education. Education Sciences, 14*(10), 1089. https://doi.org/10.3390/educsci14101089

- Escuela de Ciencias de la Computación e Informática. (s.f.-a). *CI-0110 Introducción a la Computación*. Universidad de Costa Rica. Recuperado el 9 de noviembre de 2025, de https://www.ecci.ucr.ac.cr/cursos/ci-0110

- Escuela de Ciencias de la Computación e Informática. (s.f.-b). *CI-0112 Programación 1*. Universidad de Costa Rica. Recuperado el 9 de noviembre de 2025, de https://www.ecci.ucr.ac.cr/cursos/ci-0112

- Ramírez Jiménez, E. (2021). *Programa del curso IC-1802 Introducción a la programación*. Escuela de Computación, Instituto Tecnológico de Costa Rica.

- Ramírez Jiménez, E. (2021). *Programa del curso IC-1803 Taller de programación*. Escuela de Computación, Instituto Tecnológico de Costa Rica.

- Caamaño Polini, S., Solano Orias, M., Murillo Morera, J. D., & Ramírez Jiménez, E. (2023). *Carta al estudiante EIF-201 Programación I*. Escuela de Informática, Universidad Nacional de Costa Rica.

- Gómez Fernández, C., Leitón Arrieta, K., Hernández Villalobos, M., Benavides Arguello, O., Chavarría Nerio, M., Rodríguez Oconitrillo, L. R., & Chaves Barrantes, O. (2025). *Carta al estudiante EIF-200 Fundamentos de Informática*. Escuela de Informática, Universidad Nacional de Costa Rica.

- Z. Akata et al. (2020). "A Research Agenda for Hybrid Intelligence: Augmenting Human Intellect With Collaborative, Adaptive, Responsible, and Explainable Artificial Intelligence," *Computer, 53*(8), 18–28. doi: 10.1109/MC.2020.2996587.
  keywords: {Artificial intelligence; Task analysis; Teamwork; Tools; Adaptive systems; Machine intelligence}

7 Anexos

7.1 Prompts utilizados durante el proceso de investigación

A continuación, se presentan los prompts empleados en las distintas etapas de desarrollo de la investigación, como apoyo en la búsqueda de información, revisión de redacción y comprensión teórica:

- "Recomiendame fuentes fidedignas de las que pueda investigar sobre el impacto de la IA en el ámbito pedagógico"

- “Revisa la redacción de este párrafo. Crees que se puede mejorar?”
- “Háblame del constructivismo, aprendizaje activo, evaluación auténtica, debo entenderlos con facilidad”
- “Tienes información sobre el contenido del libro “Educative Assessment: Designing Assessments to Inform and Improve Student Performance” escrito por Grant Wiggins?”
- “Según las especificaciones de la investigación, recomiéndame cuáles de los siguientes documentos me pueden funcionar”
- “Recuérdame cómo generar una cita dentro del texto en formato apa”
- “Revisa la coherencia de este texto y dame mejoras”