

Generative AI in C Programming Education: Pedagogical Strategies and Educational Risks

Marina Vicentini Secaf¹, Maurício Massaru Arimoto¹

¹Centro de Ciências Tecnológicas – Universidade Estadual do Norte do Paraná (UENP)
Bandeirantes – PR – Brasil

marina.secaf@discente.uenp.edu.br, mauricio@uenp.edu.br

Abstract. *Teaching C programming involves challenges, particularly in pointers and memory management. With the widespread adoption of Generative Artificial Intelligence (GenAI), new learning opportunities have emerged, although these tools may obscure subtle technical errors. This Systematic Mapping Study analyzes studies published between January 2022 and December 2025 to identify pedagogical uses, instructional strategies, and the impacts of GenAI in C programming education. The analysis is based on a corpus of 12 primary studies. The results indicate risks of cognitive passivity when AI is used without instructional mediation and suggest a shift from code production to code auditing. This study synthesizes recent evidence and highlights strategies for integrating GenAI without compromising students' conceptual understanding.*

Resumo. *O ensino de programação em C envolve desafios, especialmente relacionados ao uso de ponteiros e ao gerenciamento de memória. Com a ampla adoção da Inteligência Artificial Generativa (IAG), novas oportunidades de aprendizagem têm surgido, embora essas ferramentas possam ocultar erros técnicos sutis. Este Mapeamento Sistemático da Literatura identifica usos pedagógicos, estratégias instrucionais e impactos da IAG no ensino de programação em C, com base na análise de 12 estudos primários publicados entre janeiro de 2022 e dezembro de 2025. Os resultados indicam riscos de passividade cognitiva quando a IA é utilizada sem mediação instrucional e sugerem uma transição da produção de código para a auditoria de código. Este estudo sintetiza evidências recentes e destaca estratégias para integrar a IAG sem comprometer a compreensão conceitual dos estudantes.*

1. Introduction

The field of Computer Science Education has been significantly influenced by the widespread adoption of Generative Artificial Intelligence (GenAI), enabled by Large Language Models (LLMs) such as ChatGPT and Gemini. As discussed by [Becker, B. A. et al. 2023], the introduction of these technologies represents a transformation comparable to the emergence of compilers, significantly lowering the barrier to entry for code production. However, the large-scale integration of these tools in educational contexts is a recent phenomenon, intensifying only after late 2022. Consequently, the scientific literature on this topic is still emerging, characterized by a limited number of longitudinal studies and consolidated empirical evidence.

Programming is a fundamental component of Computer Science curricula, providing the foundation for students' progression across both theoretical and practical disciplines. Consequently, difficulties in learning programming may affect students' performance throughout the curriculum. However, learning programming remains challenging for novice students, as many of its concepts are inherently complex and often difficult for beginners to understand [Arimoto and Cruz 2020, Arimoto and Oliveira 2019, Santiago and Kronbauer 2017].

The C programming language remains a cornerstone of introductory Computer Science courses, playing a crucial role in developing computational thinking and in understanding low-level concepts such as memory management and pointer arithmetic. GenAI systems can generate syntactically correct solutions that remain conceptually opaque to novice learners, posing significant challenges for the assessment process. The literature warns of the risk of “cognitive passivity,” in which students adopt automatically generated code without understanding the underlying logic, thereby compromising the retention of critical concepts in the discipline [Finnie-Ansley et al. 2022].

Despite the relevance of this topic, a gap remains in the literature regarding the systematization of evidence specifically focused on the teaching of C programming. Although several studies investigate the use of GenAI in high-level languages such as Python and Java, there is a lack of systematic mappings that address the particularities of AI “hallucinations” in memory allocation and the instructional strategies required to mitigate these risks [Pereira Filho et al. 2025]. The absence of consolidated studies highlights the need to organize the dispersed knowledge produced in recent years.

In light of this scenario, this paper presents a Systematic Mapping Study (SMS) aimed at identifying, classifying, and analyzing the pedagogical uses, instructional strategies, and reported impacts of Generative AI in C programming education. The study covers publications indexed between January 2022 and December 2025 and seeks to identify which teaching strategies are being adopted to transform AI from a provider of immediate solutions into a tool that supports learning, as well as to map the main impacts and limitations reported by the academic community.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the methodological procedures adopted in this study. Section 4 presents the analysis and synthesis of the results. Section 5 discusses the findings. Finally, Section 6 presents the conclusions of this work and outlines directions for future research.

2. Related Work

Recent literature on the use of GenAI in Computer Science Education has grown rapidly, with a focus on evaluating LLM performance. According to [Becker et al. 2023], this development represents a paradigm shift comparable to the emergence of compilers. In their analysis, the authors discuss the ethical and pedagogical implications of automatic code generation but adopt a general perspective on the Computer Science curriculum without examining the syntactic characteristics of programming languages such as C.

In the context of performance comparisons between students and AI-based systems, [Finnie-Ansley et al. 2022] conducted a study comparing solutions generated by Codex and GPT-3 with responses from novice students. Although the results indicate that AI

outperforms average students on introductory programming tasks, the study investigates Python and basic programming logic. This perspective tends to overlook challenges in teaching C, where manual memory management and pointer arithmetic may lead to subtler “hallucinations” that are harder to debug than syntax errors in languages such as Python.

Other recent studies investigate the impact of GenAI from the perspectives of automated assessment and educational data mining. The work of [Silva et al. 2024], for example, analyzes the use of GenAI in automated assessment and educational data mining in Brazil, while [Prather, J. et al. 2023] examines students’ perceptions of Copilot. Although these studies contribute to understanding the adoption of this technology, they do not systematize pedagogical mediation strategies specifically for teaching C programming. Existing research has primarily focused on the technical capabilities of these tools, whereas less attention has been given to their pedagogical integration in programming education.

Overall, existing studies predominantly emphasize technical performance and code generation capabilities, whereas less attention has been devoted to pedagogical mediation, instructional strategies, and educational risks associated with low-level programming concepts in C. In particular, there is a lack of studies that integrate three key dimensions: (1) the C programming language, (2) the technical limitations of GenAI systems, such as hallucinations related to memory management, and (3) instructional mediation in the teaching process. Addressing this gap is essential for understanding how GenAI can be effectively integrated into programming education without compromising students’ conceptual learning. This study contributes to this discussion by focusing on pedagogical strategies aimed at mitigating these limitations, thus complementing previous work that prioritized higher-level languages or analyzes focused primarily on tool performance.

Although previous studies analyze tool performance or students’ perceptions, few studies examine how GenAI is pedagogically integrated into C programming education. In particular, there is a lack of studies jointly analyzing pedagogical uses, instructional strategies, and educational impacts in this context.

3. Methodology

This Systematic Mapping Study (SMS) aims to identify and categorize how GenAI has been used in C programming education, focusing on pedagogical strategies, reported impacts, and associated challenges. The study follows the SMS guidelines proposed by [Petersen et al. 2008] and [Kitchenham and Charters 2007], ensuring transparency and enabling replication of the study. The process comprises the planning and definition of (i) research questions, (ii) search strategies, (iii) inclusion and exclusion criteria, and (iv) the execution of the search and selection of studies.

3.1. Research Questions

The research questions (RQs) proposed in this study include:

- **RQ1:** Which GenAI tools and types of pedagogical use have been reported in C programming education?
- **RQ2:** Which pedagogical strategies are associated with the use of GenAI in this context?
- **RQ3:** What impacts, limitations, and educational risks have been reported regarding the use of these tools?

These research questions guide the data extraction and analysis process, enabling the identification of GenAI tools, pedagogical strategies, and reported educational impacts in the selected studies.

3.2. Search Strategy

The search was conducted in December 2025, applying the defined search string to the title, abstract, and keyword fields of the selected databases. The temporal restriction for the beginning of the search (January 2022) is justified by the public availability of Large Language Models (LLMs) accessible to non-specialist users. Although AI technologies have existed for decades, it was only with the release of tools such as GitHub Copilot (June 2022 for the general public) and ChatGPT (November 2022) that their use became widespread in classrooms. Studies published before this period mainly address predictive techniques or traditional automated grading tools, which differ from the generative and explanatory capabilities of current GenAI systems and therefore fall outside the scope of this investigation. Only articles written in English and Portuguese were considered.

The following digital databases were consulted: ACM Digital Library, IEEE Xplore, Google Scholar, RENOTE, and the SBC Digital Library (SBC-OpenLib – SOL), which includes major Brazilian venues such as CBIE, RBIE, SEMISH and WEI. Google Scholar was used as a complementary search source to increase recall. In the IEEE Xplore, SOL, and RENOTE databases, the search string was applied to the main metadata fields (Title, Abstract, and Keywords). In the ACM Digital Library, due to the comprehensive indexing characteristics of its search engine, the application of the string was restricted to the Abstract field to maintain search precision. In Google Scholar, due to the absence of structured metadata filters and the large volume of retrieved results (~8,000), the search was restricted to titles and limited to the 100 most relevant records, a common practice adopted to ensure the feasibility of the screening process in Systematic Mapping Studies (SMS). The search string was defined based on the research questions and iteratively refined through preliminary tests in the selected databases to ensure adequate coverage of relevant studies.

("Generative AI" OR "Large Language Models" OR "LLM" OR "ChatGPT" OR "Gemini" OR "Copilot" OR "Claude") AND ("C programming" OR "C language" OR "introductory programming") AND ("education" OR "teaching" OR "learning").

Table 1 summarizes the distribution of records retrieved from each selected database, considering the search scope adopted in each source.

Table 1. Distribution of retrieved studies across digital databases.

Database	Results (<i>n</i>)
ACM Digital Library	76
IEEE Xplore	56
SBC OpenLib (SOL)	194
RENOTE	9
Google Scholar*	100
Total Records	435

* Search performed in titles only, limited to the 100 most relevant results. This restriction follows common practices in SMS studies due to the instability and high volume of Google Scholar indexing results. All other databases considered title, abstract, and keywords.

3.3. Inclusion and Exclusion Criteria

The inclusion and exclusion criteria were defined to ensure that only studies directly related to the pedagogical use of GenAI in C programming education were considered.

The inclusion criteria (IC) for the identified studies are as follows:

- **(IC1) Temporal Scope:** Studies published between January 2022 and December 2025, covering the period of popularization of GenAI tools;
- **(IC2) Central Topic:** Studies investigating the use of LLMs or GenAI systems (e.g., ChatGPT, Copilot, Gemini);
- **(IC3) Specific Context:** Research explicitly applied to the teaching or learning of programming in the C language (generic studies on “programming” without mention of C were excluded if they did not address low-level concepts);
- **(IC4) Type of Approach:** Studies describing a practical pedagogical application (e.g., tutoring, code generation, refactoring) or analyzing students’ and instructors’ perceptions regarding these tools.

The exclusion criteria (EC) for the identified studies are as follows:

- **(EC1) Focus on Other Languages:** Studies on programming education focused exclusively on high-level languages (Python, Java, Scratch) or visual block-based programming without presenting segregated data or specific discussions of C language;
- **(EC2) Purely Technical Bias:** Articles evaluating the performance of LLMs in professional software engineering benchmarks (e.g., HumanEval) without discussing pedagogical implications, teaching strategies, or impacts on novice learning;
- **(EC3) Publication Type:** Short abstracts, posters, slide presentations, proceedings prefaces, or grey literature (blogs, non-peer-reviewed technical reports) that do not provide sufficient methodological or results-related details;
- **(EC4) Accessibility and Language:** Studies whose full text was not available in the subscribed databases or written in languages other than Portuguese or English;
- **(EC5) Duplication:** In the case of duplicate studies (the same work published in a conference and later in a journal), only the most complete version was considered.

3.4. Study Execution and Selection Process

The study selection process followed a four-step workflow, in accordance with the guidelines for SMS. The initial search across the databases returned a total of 435 studies. Duplicate records identified between the different databases were removed before the screening stage.

1. **Identification and Removal of Duplicates:** In the first stage, criterion EC5 was applied, resulting in the removal of 47 duplicate records indexed across multiple databases. After this process, 388 studies remained for the screening stage.

2. **Screening:** The remaining 388 studies were subjected to title and abstract screening. At this stage, criteria EC1 (focus on other languages) and EC2 (purely technical bias) were applied. Three hundred and forty three studies were excluded because, although they addressed programming, they did not focus on the C language or concentrated only on software engineering metrics without a pedagogical perspective. Forty-five articles remained.
3. **Eligibility:** The 45 candidate studies were read in full to verify compliance with the inclusion criteria. At this stage, 33 studies were excluded, mainly because they did not meet criterion IC4 (failure to describe a clear pedagogical application or to present sufficient data regarding the intervention).
4. **Final Inclusion:** At the end of the process, a set of 12 primary studies was selected for data extraction and synthesis.

The complete workflow of the study selection process, including the stages of identification, screening, eligibility, and inclusion, as well as the application of the inclusion (IC) and exclusion (EC) criteria, is presented in Figure 1.

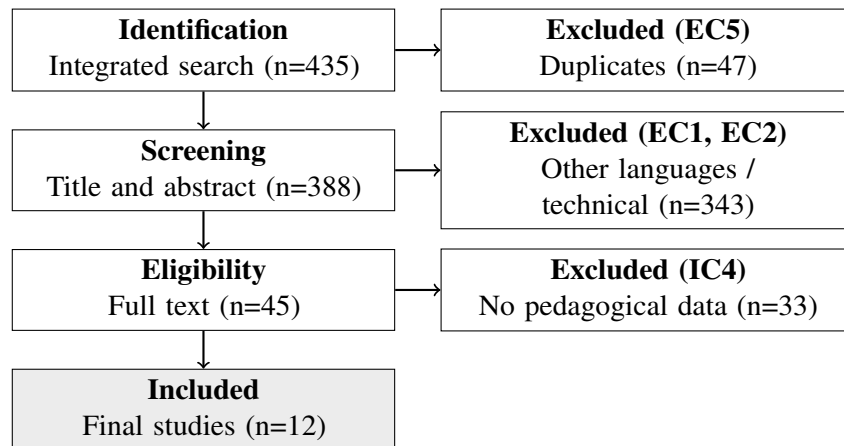


Figure 1. Study selection workflow and application of IC and EC.

For each included study, data were extracted regarding the category of GenAI use, the associated pedagogical strategy, and the reported educational impacts.

The reduction from 435 retrieved records to 12 selected studies reflects the highly specific focus of this investigation. Although the literature on GenAI in education has expanded rapidly since 2022, most studies focus on high-level languages such as Python and JavaScript. Relatively few investigations address C programming specifically.

The scarcity of studies strictly focused on the C programming language may be related to its intrinsic complexity, particularly the requirement for explicit memory management and pointer manipulation, areas in which GenAI systems still present severe technical limitations, such as the generation of memory leaks. Therefore, the reduced number of final studies does not reflect a limitation of the search process, but rather a genuine gap in the current scientific literature, which still lacks structured pedagogical interventions for teaching low-level programming concepts mediated by GenAI.

The complete replication package, including the search strings, retrieved records, screening decisions, extracted data, and study classifications, is publicly available at: <https://mahhhvs.github.io/SBIE-article/>.

4. Analysis and Synthesis of Results

The execution of the mapping made it possible to identify 12 primary studies addressing the use of GenAI in C programming education. The following analysis does not aim to evaluate the quality of the tools, but rather to systematize how the academic community has been adopting these technologies.

4.1. Overview of Studies

Table 2 presents the characterization of the selected studies. A predominance of research focused on Higher Education can be observed, particularly in introductory programming courses. To answer the research questions, the data extracted from the studies were analyzed by thematic categorization and descriptive synthesis, as presented below.

Table 2. Characterization of the 12 Selected Primary Studies

Study	Year	Source	Focus	Category
[Siam et al. 2024]	2024	ACM	Evaluation of AI-generated solutions	Debugging
[Pereira Filho et al. 2025]	2025	RBIE	Evaluation of LLM responses	Debugging
[Junior et al. 2023]	2023	SBIE	Oral explanation strategies	Tutoring
[Xie et al. 2024]	2024	IEEE	Automated feedback	Tutoring
[Andleeb et al. 2025]	2025	ACM	Student vs AI-generated code	Code Generation
[Kavianpour 2024]	2024	IEEE	AI tutoring support	Tutoring
[Abouelenein et al. 2025]	2025	ACM	Prompt engineering framework	Tutoring
[Silva et al. 2024]	2024	RENOTE	AI-assisted assessment	Assessment
[Becker, B. A. et al. 2023]	2023	ACM	Educational implications of AI code generation	Code Generation
[Denny, P. et al. 2024]	2024	ACM	Prompt engineering activities	Assessment
[Finnie-Ansley et al. 2022]	2022	ACM	Codex versus novice programmers	Code Generation
[Prather, J. et al. 2023]	2023	ACM	Student interaction with Copilot	Code Generation

4.2. Types of Pedagogical Uses of Generative AI (RQ1)

To answer the first research question, an inductive thematic analysis was conducted on the extracted data. The usage categories emerged from grouping the practices reported in the 12 primary studies.

Some studies describe hybrid approaches (e.g., code generation followed by explanation). However, for the purposes of systematization and quantitative analysis, each study was classified according to the predominant pedagogical focus of the reported intervention. Based on this criterion, four main categories were identified:

- **AI for Code Generation and Solutions (n=4):** This is the predominant category, in which the tool is used to generate complete syntax from natural language. Studies such as [Becker, B. A. et al. 2023, Finnie-Ansley et al. 2022, Andleeb et al. 2025] employ this approach to compare the quality of AI-generated code with code produced by novice students, while [Prather, J. et al. 2023] analyze student interaction during the automatic generation process.

- **AI as Tutor and for Conceptual Explanation (n=4):** This category focuses on the dialogical aspect and the personalization of instruction. [Xie et al. 2024, Kavianpour 2024] describe systems that provide real-time feedback. [Junior et al. 2023] report the use of AI for oral explanations aimed at mitigating plagiarism, while [Abouelenein et al. 2025] propose frameworks to guide students in formulating conceptual questions.
- **AI in Support of the Assessment Process (n=2):** This category refers to instructional or methodological uses of AI to create new assessment instruments. [Silva et al. 2024] describes the use of AI to assist in generating exam questions, while [Denny, P. et al. 2024] introduce the concept of “Prompt Problems”, reversing the evaluation logic in which students must assess the output produced by AI.
- **AI for Debugging and Error Correction (n=2):** These studies explore the ability of AI to identify logical or memory-related errors. [Pereira Filho et al. 2025, Siam et al. 2024] specifically investigate the effectiveness (and hallucinations) of these tools when attempting to correct code involving segmentation faults and improper pointer usage.

To improve the reliability of this categorization process, two researchers independently reviewed and classified the studies according to their predominant pedagogical focus. Any disagreements were resolved through discussion until consensus was reached.

Most studies (8 out of 12 studies, $\approx 66\%$) focus on the generative and explanatory capabilities of these tools (Generation and Tutoring). This suggests that, in this initial phase, the academic community has prioritized understanding whether AI can effectively support code generation and tutoring activities in C programming. The distribution of studies across these categories is presented in Figure 2.

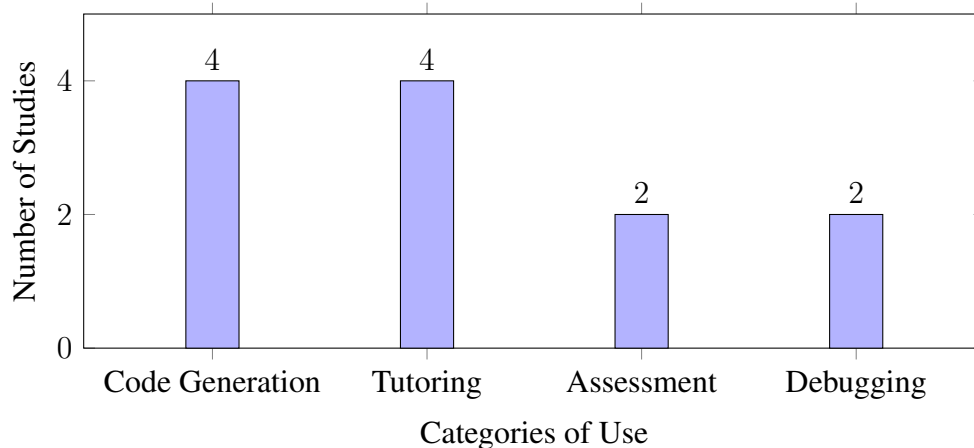


Figure 2. Distribution of primary studies by category of pedagogical use.

4.3. Educational Strategies and Context (RQ2)

The analysis reveals that the use of AI does not occur in isolation, but rather in association with new teaching strategies, predominantly in Higher Education. To categorize these approaches, three main strategic axes were identified:

1. **Code Auditing and Validation (n=5):** The most recurrent strategy, in which the pedagogical activity shifts from “writing from scratch” to “testing and

- correcting” the code generated by AI. Studies such as [Denny, P. et al. 2024, Pereira Filho et al. 2025, Siam et al. 2024] focus on identifying subtle logical failures and memory hallucinations, requiring students to develop critical thinking regarding the output produced by the AI system.
2. **Intelligent Tutoring and Prompt Engineering (n=5):** This strategy focuses on dialogical interaction. It includes the use of AI as a “24-hour tutor” for answering questions [Xie et al. 2024] and the explicit teaching of how to formulate questions. The work of [Abouelenein et al. 2025] stands out, proposing the R5E framework to teach students how to refine their prompts in order to obtain safer C code.
 3. **Instructor Mediation in Assessment (n=2):** Strategies focused on ensuring authorship and avoiding passivity. [Silva et al. 2024] proposes the use of AI to assist instructors in creating assessment items, while [Junior et al. 2023] describes assessment through oral explanation, requiring students to verbalize the memory allocation logic of the generated code in order to demonstrate their understanding of the concepts.

Table 3 summarizes the distribution of instructional strategies (RQ2) across the categories of pedagogical use of GenAI identified in the selected studies (RQ1). Although some studies reported hybrid approaches, each study was classified according to its predominant pedagogical focus and instructional strategy to support quantitative analysis.

Table 3. Distribution of instructional strategies across pedagogical use categories.

Category	Auditing & Validation	Tutoring & Prompt Eng.	Assessment & Mediation
Code Generation	1 (25%)	3 (75%)	0 (0%)
Tutoring	0 (0%)	4 (100%)	0 (0%)
Assessment	1 (50%)	0 (0%)	1 (50%)
Debugging	2 (100%)	0 (0%)	0 (0%)

4.4. Educational Limitations and Risks (RQ3)

The analyzed literature consistently points to specific risks associated with the teaching of C programming, which differ from those observed in high-level languages such as Python:

- **Hallucinations in Memory Management:** [Siam et al. 2024] and [Pereira Filho et al. 2025] warn that AI systems frequently generate syntactically correct code but with serious security flaws, such as forgetting the `free()` command after a `malloc()`, or accessing invalid memory positions.
- **Cognitive Passivity:** [Andleeb et al. 2025] and [Prather, J. et al. 2023] report that unrestricted use may lead students to accept the first response generated by AI without critical evaluation, harming the development of the algorithmic thinking required to understand the complexity of pointer manipulation.

Although these aspects did not explicitly emerge from the analyzed studies, the widespread integration of GenAI into education may also involve additional impact dimensions beyond the classroom context. The continuous inference processes required by

LLMs demand energy-intensive computational infrastructures, potentially contributing to environmental impacts. While this issue was not identified in the analyzed pedagogical studies, it represents a broader concern that has received increasing attention in Computing Education.

Furthermore, the risk of deskilling, understood as the erosion of fundamental competencies, emerges as another critical concern. Excessive reliance on tools capable of automating code generation may compromise the development of essential skills such as manual debugging and algorithmic reasoning, particularly in low-level programming languages such as C, where understanding explicit memory management remains indispensable. The absence of these discussions in the analyzed primary studies reveals a gap that should be addressed in future investigations concerning the pedagogical adoption of GenAI.

It should be noted that many of the analyzed studies were conducted using earlier generations of LLMs, such as Codex and GPT-3.5. Therefore, some technical limitations reported in the literature, particularly those related to memory management and pointer handling, should be interpreted in light of the rapid evolution of these models. Nevertheless, the pedagogical concerns identified in this mapping, such as cognitive passivity and the need for instructional mediation, remain relevant regardless of improvements in model performance.

5. Discussion

Based on the results presented in Section 4, important patterns can be observed in the adoption of GenAI in C programming education, suggesting a reconfiguration of the instructor's role and of assessment methods.

5.1. Impacts on C Assessment and Learning

The disparity identified between the syntactic accuracy (high) and the semantic accuracy (variable) of the tools suggests the need to revise assessment strategies.

In the context of learning, the traditional focus appears to be insufficient in light of the widespread use of these tools. Since GenAI tools can generate solutions to “code completion” tasks almost instantly, the results suggest a possible shift in focus, moving from construction to auditing. Students may need to move beyond a purely code-production role to becoming “critical evaluators of AI-generated code”, capable of identifying whether the solution generated by AI correctly released allocated resources (use of free) or whether it performed improper accesses through pointer arithmetic.

Regarding assessment, the findings indicate limitations of exercises based exclusively on the correct output of the code. If the tool generates the final solution, evaluation must focus on the process. Strategies such as “Oral Explanation” [Junior et al. 2023] and “Prompt Problems” [Denny, P. et al. 2024] show that verifying learning now depends on the student's ability to explain why and how the code works, mitigating cognitive passivity in which the student accepts the AI-generated response without critical reflection.

5.2. Divergence of Perspectives: National vs. International

National and international studies appear to emphasize different aspects of GenAI adoption. International literature (ACM/IEEE) tends to focus on technical benchmarks

[Finnie-Ansley et al. 2022, Siam et al. 2024], often examining the effectiveness of GenAI tools in programming tasks (e.g., whether Copilot correctly solves introductory programming exercises).

In contrast, national literature (SBC) prioritizes methodological aspects and instructor mediation [Pereira Filho et al. 2025, Silva et al. 2024]. This divergence may reflect differences in research priorities. While international studies often focus on validating the capabilities of GenAI tools, Brazilian studies appear more concerned with their pedagogical integration and classroom adoption, particularly regarding instructional mediation and the role of instructors in supporting students' learning processes.

5.3. Limitations and Threats to Validity

The interpretation of the results of this mapping should consider certain limitations inherent to the study design and the emerging stage of the technology investigated.

- **Sample Size ($n = 12$):** The small number of primary studies reflects the novelty of the topic (post-2022). The direct impact of this limitation is the difficulty of statistical generalization of the data; the identified patterns represent trends among early adopters and may not reflect established practice in the global Computer Science Education community. This limited number of studies does not indicate a lack of relevance of the topic, but rather the recent emergence of GenAI tools in educational contexts.
- **Search and Retrieval Bias:** Although the search covered major databases in Computing Education (ACM, IEEE, SBC-OpenLib, RENOTE, and Google Scholar), the restriction to studies published in Portuguese and English may have excluded relevant contributions from other linguistic communities. Furthermore, the exclusion of multidisciplinary indexing services such as Scopus and Web of Science, as well as the limitation of Google Scholar to the 100 most relevant results, may have limited the retrieval of relevant studies. Consequently, the selected corpus may not fully represent all existing research on the pedagogical use of GenAI in C programming education.
- **Technological Volatility:** The rapid obsolescence of GenAI models (e.g., the transition from GPT-3.5 to GPT-4o during the analysis period) may make some of the technical criticisms identified in studies from 2023 (such as the inability to handle complex pointers) less severe at present.
- **Positive Publication Bias:** As GenAI is an emerging and highly visible research topic, studies reporting successful educational applications or innovative instructional practices may be more likely to be published than studies reporting neutral or negative outcomes. Consequently, the literature analyzed in this mapping may overrepresent positive experiences and underestimate challenges associated with the pedagogical adoption of GenAI.

To mitigate these limitations, the application of the snowballing technique is suggested to retrieve studies that may not have been indexed in the main databases. In addition, expanding the search to multidisciplinary databases such as Scopus and Web of Science may help capture a broader spectrum of pedagogical practices.

6. Conclusion

This Systematic Mapping Study achieved its main objective by investigating and classifying the pedagogical use of GenAI in introductory C programming education. Through the selected studies, it was possible to construct an overview of emerging instructional strategies and the technical challenges faced by the academic community.

The synthesized results reveal a duality in the adoption of these tools. While these systems demonstrate high effectiveness as “personalized tutors” for explaining syntax and debugging logical errors, they present limitations in low-level concepts such as manual memory management and pointer arithmetic. The mapping suggests that the growing adoption of AI may require a shift in the student’s role, from code production toward the critical review and validation of AI-generated solutions.

Beyond identifying existing studies, this mapping also provides an organized view of how GenAI has been pedagogically integrated into C programming education. A key contribution of this work lies in the systematization of evidence regarding instructor mediation strategies for the use of GenAI in this context. The analyzed studies report risks such as cognitive passivity and technical hallucinations, as well as strategies adopted to mitigate these issues, including oral code defense and prompt engineering activities.

From an educational perspective, the findings suggest that the integration of GenAI in programming education requires a shift in instructional design. Instead of focusing exclusively on code production tasks, instructors may benefit from adopting activities centered on code auditing, explanation, and critical evaluation of AI-generated solutions.

By organizing recent evidence on the pedagogical use of GenAI in C programming education, this study contributes to supporting educators in designing instructional practices that balance the benefits of AI assistance with the development of students’ conceptual understanding.

As future work, experimental classroom studies are suggested to measure the impact of these mediation strategies on learning dynamic memory allocation. Additionally, the search scope may be expanded to include grey literature and technical reports on new versions of LLMs in order to monitor the evolution of these tools in detecting segmentation faults and memory leaks.

Generative AI Use Statement

During the preparation of this work, the authors used the Gemini tool (Gemini 3 Pro) to assist with translation, grammatical refinement, and structuring of arguments. After using the tool, the authors reviewed and edited the generated content as necessary and assume full responsibility for the final content of the publication.

Acknowledgments

This work was supported by the Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Estado do Paraná (FA) through the Institutional Program for Technological Development and Innovation Scholarships (PIBITI).

References

Abouelenein, Y. A. M. et al. (2025). The R5E pattern: can artificial intelligence enhance programming skills development? *Education and Information Technologies*, 30:22177–22205.

- Andleeb, S., Kantorski, B., and Carver, J. (2025). ChatGPT in Introductory Programming: Counterbalanced Evaluation of Code Quality, Conceptual Learning, and Student Perceptions. In *Proceedings of the 26th ACM Annual Conference on Cybersecurity & Information Technology Education*, pages 180 – 186. <https://doi.org/10.1145/3769694.3771128>.
- Arimoto, M. M. and Cruz, J. H. (2020). Ensino de Lógica e Programação no Ensino Médio por meio de uma Abordagem Lúdica e Gamificada. In *Workshop sobre Educação em Computação (WEI)*, pages 160–170. <https://doi.org/10.5753/wei.2020.11151>.
- Arimoto, M. M. and Oliveira, W. T. (2019). Dificuldades no processo de aprendizagem de programação de computadores: um survey com estudantes de cursos da área de computação. In *Workshop sobre Educação em Computação (WEI)*, pages 244–254. <https://doi.org/10.5753/wei.2019.6633>.
- Becker, B. A. et al. (2023). Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, pages 500–506. <https://doi.org/10.1145/3545945.3569759>.
- Denny, P. et al. (2024). Prompt Problems: A New Type of Programming Exercise. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*, pages 296 – 302. <https://doi.org/10.1145/3626252.3630909>.
- Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., and Prather, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proceedings of the 24th Australasian Computing Education Conference (ACE '22)*, pages 10–19, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3511861.3511863>.
- Junior, S. S., Freitas, R., Morais, M., and Costa, D. (2023). ChatGPT no auxílio da aprendizagem de programação: Um estudo de caso. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1375–1384, Porto Alegre, RS, Brasil. SBC. <https://sol.sbc.org.br/index.php/sbie/article/view/26762>.
- Kavianpour, S. (2024). Teaching programming languages by two teachers: Instructor and chatgpt. In *2024 ASEE Annual Conference & Exposition*, pages 1–16. American Society for Engineering Education (ASEE).
- Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University and Durham University Joint Report, Keele, Staffordshire, UK.
- Pereira Filho, L. C., Souza, T. d. P. C. d., and Paula, L. B. d. (2025). Análise das Respostas de LLMs em Relação ao Conteúdo Introdutório de Programação: um Comparativo entre o ChatGPT e o Gemini. *Revista Brasileira de Informática na Educação*, 33:722–747.
- Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77.
- Prather, J. et al. (2023). “It’s weird that it knows what I want”: Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, 31(1):1–31.

- Santiago, A. D. and Kronbauer, A. H. (2017). Um Modelo Lúdico para o Ensino de Conceitos de Programação de Computadores. *Revista Brasileira de Informática na Educação*, 25:1–29.
- Siam, M. K., Gu, H., and Cheng, J. Q. (2024). Programming with AI: Evaluating ChatGPT, Gemini, AlphaCode, and GitHub Copilot for Programmers. In *Proceedings of the 3rd International Conference on Computing Advancement*, pages 346 – 354. <https://doi.org/10.1145/3723178.3723224>.
- Silva, L., Vidotto, K. N. S., Tarouco, L. M. R., and Silva, P. F. (2024). Inteligência Artificial Generativa no Ensino de Programação: um mapeamento sistemático da literatura. *Revista RENOTE – Novas Tecnologias na Educação*, 22(1):262–272. <https://doi.org/10.22456/1679-1916.141553>.
- Xie, H., Wang, J., and Zhang, Y. (2024). Research on the Application of Intelligent Assisted Teaching System in Programming Courses. In *2024 7th International Conference on Education, Network and Information Technology (ICENIT)*, pages 194–197. IEEE. <https://ieeexplore.ieee.org/document/10873910>.