KARYA ILHAM PUBLISHING
CONNECTING MINDS, BRIDGING IDEAS, INSPIRING READERS

Karya Journal of Emerging Technologies in Human Services

Journal homepage:
https://karyailham.com.my/index.php/kjeths/index
ISSN: 3093-6551

The Challenges of Teaching and Learning Programming in Schools: Insights from A Systematic Literature Review

Efa Elfrieda Abu Bakar[1,*], Noor Dayana Abd Halim1, Mohd Fadzil Abdul Hanid[1], Rita Inderawati[2]

[1] Faculty of Social Sciences and Humanities, Universiti Teknologi Malaysia (UTM), 81310 Johor Bahru, Johor, Malaysia
[2] Fakultas Keguruan dan Ilmu Pendidikan, Universitas Sriwijaya Jl. Raya Palembang - Prabumulih No.KM. 32, Indralaya Indah, Kec. Indralaya, Kabupaten Ogan Ilir, Sumatera Selatan 30862 Palembang Indonesia

ARTICLE INFO

Article history:
Received 20 February 2025
Received in revised 13 March 2025
Accepted 4 April 2025
Available online 25 April 2025

Keywords:
Programming education; teaching challenges; learning challenges; school curriculum

ABSTRACT

The growing importance of programming education in schools is undeniable as digital literacy becomes a critical competency for the 21st century. However, various challenges hinder effective programming education, including pedagogical constraints, cognitive difficulties, and technological barriers. This systematic literature review provides a comprehensive analysis of recent studies to identify and categorize these challenges. Methodologically, a systematic review approach was adopted, involving comprehensive searches across academic databases to identify relevant studies. To achieve this, we conducted an extensive search of scholarly articles from reputable databases such as Scopus and Web of Science, focusing on studies published between 2023 and 2025. The flow of study based on PRISMA framework. The database found (n=26) final primary data was analysed. Numerical results from the selected literature highlight three key themes: (1) Pedagogical Approaches and Strategies in Teaching Programming, (2) Computational Thinking and Cognitive Development in Programming Education, and (3) Technological Integration and Challenges in Programming Education. Findings indicate that 38.5% highlighting the effectiveness of game-based learning, project-based instruction, and peer collaboration in improving engagement and learning outcomes; 30.8% addressing cognitive overload, conceptual difficulties, and the need for structured problem-solving frameworks to enhance students' analytical skills; and 30.8% emphasizing issues such as inadequate infrastructure, limited access to programming tools, and insufficient teacher training. The findings underscore the necessity of standardized pedagogical frameworks, enhanced teacher training programs, and curriculum reforms to bridge the gap between theory and practice. Future research should focus on developing scalable, evidence-based strategies to address these challenges and ensure the effective integration of programming education across diverse school settings.

[*] *Corresponding author.*
E-mail address: efaelfrieda@graduate.utm.my

https://doi.org/10.37934/kjeths.1.1.4863

1. Introduction

In the digital age, programming has become an essential skill, not only for future software developers but also for fostering computational thinking, problem-solving, and logical reasoning among students [1,2] Many countries have integrated programming into school curricula, recognizing its role in preparing students for a technology-driven world. However, the teaching and learning of programming in schools remain a significant challenge. Despite the growing emphasis on digital literacy, numerous obstacles hinder effective instruction, ranging from pedagogical issues to infrastructural limitations and cognitive barriers among students [3]. One of the primary challenges in teaching programming is the inherent complexity of the subject. Unlike traditional subjects, programming requires both theoretical understanding and practical application. Students must grasp abstract concepts such as variables, loops, and algorithms while simultaneously developing problem-solving skills to apply these concepts effectively. The transition from understanding syntax to implementing functional programs can be daunting, leading to frustration and a high dropout rate in computer science courses [4]. Moreover, many students experience cognitive overload due to the multi-layered nature of programming, which involves logical reasoning, debugging, and conceptual abstraction simultaneously [5]. Teachers also face significant challenges in delivering programming education effectively. Many educators lack sufficient training in computer science pedagogy, particularly in schools where programming is a recent addition to the curriculum. Unlike subjects with well-established teaching methodologies, programming requires innovative instructional strategies, including problem-based learning, pair programming, and debugging exercises, which are not always implemented effectively [6]. Additionally, educators often struggle to cater to diverse student needs, as programming proficiency varies widely within a single classroom. Some students grasp concepts quickly, while others require extensive guidance, making differentiated instruction a necessity yet a logistical challenge [3].

Furthermore, the availability of technological resources plays a crucial role in programming education. Many schools, especially in developing regions, lack adequate infrastructure, including computers, stable internet access, and up-to-date software tools [7]. Without these resources, hands-on programming practice becomes difficult, limiting students' opportunities to develop coding skills. Even in well-equipped schools, issues such as outdated curricula and insufficient access to industry-relevant tools can prevent students from acquiring relevant programming competencies [8]. Another significant challenge is student motivation and engagement. Programming can be intimidating for beginners, particularly if they do not see immediate relevance to real-world applications. Studies indicate that students are more engaged when programming tasks are contextualized within meaningful, real-life scenarios [9]. However, many school curricula focus primarily on syntax and basic problem-solving without incorporating projects that align with students' interests, leading to disengagement and a lack of enthusiasm for programming [10]. Finally, assessment methods in programming education pose another hurdle. Traditional assessment techniques, such as written exams and quizzes, may not effectively measure students' programming proficiency. Instead, project-based evaluations, peer reviews, and debugging tasks provide a more comprehensive understanding of students' coding abilities [11]. However, implementing these alternative assessment strategies requires additional effort from educators and institutions, further complicating the teaching process.

Despite the increasing integration of programming into school curricula, there remains a significant research gap in identifying scalable, evidence-based solutions that effectively address pedagogical, cognitive, and technological barriers. While previous studies have explored individual aspects such as computational thinking development, instructional strategies, and technology

integration, there is a lack of a comprehensive framework that synthesizes these dimensions to provide actionable recommendations for educators. This study aims to bridge this gap by systematically reviewing literature from 2023–2025, identifying key challenges, and proposing research-backed interventions that enhance programming education effectiveness.

2. Literature Review

The teaching and learning of programming in schools present multifaceted challenges that stem from pedagogical limitations, students' cognitive difficulties, and constraints in technology integration. Research by Florou *et al*., [12] highlights the role of educators in facilitating self-assessment among students learning programming concepts. Their study identifies major obstacles, including students' struggles with evaluating their own learning progress and teachers' difficulties in incorporating modern educational tools effectively. Additionally, the study underscores the importance of guiding principles that enhance programming education, particularly in primary schools. Sandstrak *et al*., [13] further elaborate on the pedagogical issues by examining different instructional approaches used across campuses. Their research found that varying pedagogical methods, such as top-down and bottom-up approaches, yield no significant differences in student outcomes, suggesting that traditional teaching methods may not fully address the challenges novice learners face. Similarly, Yang *et al*., [14] propose the use of graphic organizers to bridge knowledge gaps in elementary students' computational thinking and programming learning. Their findings indicate that structured visual aids significantly improve students' programming skills and problem-solving abilities, mitigating some of the cognitive difficulties inherent in learning programming at an early age.

Assessment methodologies and student engagement strategies also pose critical challenges in programming education. Sandstrak *et al*., [13] argue that the effectiveness of different assessment methods—ranging from multiple-choice tests to portfolio-based evaluations—varies significantly depending on the instructional context. Their study, conducted across three campuses, found that while alternative assessments such as home exams during the pandemic helped maintain student engagement, they did not necessarily lead to better learning outcomes. The findings suggest that assessment techniques should be carefully aligned with pedagogical strategies to foster meaningful learning experiences. Moreover, Oralbayeva *et al*., [15] emphasize the role of interactive learning environments, such as Montessori-based child-robot interactions, in fostering long-term retention of programming-related concepts. Their study on alphabet acquisition using robotics suggests that interactive and self-directed learning models could be adapted to programming education, addressing engagement-related challenges. In a related study, Ienco *et al*., [16] explore the integration of educational robotics to promote hands-on learning experiences. Their research demonstrates that programming within a real-world context, such as sustainability projects, enhances students' engagement and motivation, reinforcing the need for active learning strategies in programming education.

Technological constraints and curriculum design further complicate programming instruction in schools. Florou *et al*., [12] note that many teachers struggle with integrating modern educational technologies into their teaching due to inadequate training and a lack of institutional support. This issue is compounded by the rapid evolution of programming languages and tools, which often outpace curriculum development. The lack of a standardized approach to incorporating programming into school curricula further exacerbates the problem. Oralbayeva *et al*., [15] highlight how structured, well-designed educational technologies, such as robotics-based learning, can provide students with an adaptive learning environment, yet the successful implementation of such

technologies remains a challenge due to resource limitations. These findings indicate a pressing need for curriculum reforms that ensure programming education is both accessible and adaptable to technological advancements. The integration of programming education in schools presents multiple challenges, ranging from digital competency gaps to pedagogical constraints. Dabengwa *et al*., [17] explored the digital competencies of secondary school teachers in Zimbabwe and identified significant disparities in digital literacy, particularly in rural areas where limited access to ICT tools and infrastructure hinders effective teaching. While students demonstrated proficiency in basic applications such as Microsoft Word and PowerPoint, programming-related competencies remained largely underdeveloped. Similarly, Fante *et al*., [18] highlighted the evolving nature of game-based learning (GBL) in STEM education, emphasizing the shift from technology-centric approaches to strategies that prioritize engagement, motivation, and understanding of complex concepts. Despite the potential of GBL to enhance programming education, its implementation faces obstacles such as inadequate empirical validation and limited integration with computational thinking.

Additionally, Dai *et al*., [19] introduced an embodied, analogical, and disruptive (EAD) approach to AI pedagogy, revealing the effectiveness of interactive learning methods in fostering abstract and systems thinking. However, cognitive overload and communication challenges remain key limitations, which are also relevant to programming education in schools. Another significant issue in programming education is the complexity of aligning teaching methodologies with students' cognitive and affective needs. Cohn *et al*., [20] proposed a human-AI collaborative model to enhance STEM learning through multimodal learning analytics (MMLA). Their findings suggest that AI-generated timelines can facilitate formative feedback, improving students' computational model-building skills. However, AI-based interventions struggle with interpreting student emotions and social interactions, making it challenging to offer real-time, contextually appropriate feedback. Moreover, game-based learning approaches analyzed by Fante *et al*., [21] revealed an increasing focus on emotional and experiential learning components, signifying a broader need for instructional methods that engage students both cognitively and affectively. Dabengwa *et al*., [17] further emphasized the importance of problem-solving and digital content creation in programming education, noting that despite technological advancements, digital safety and security remain major concerns, particularly in underprivileged school environments. These findings indicate that while innovative pedagogical approaches hold promise, their effectiveness is contingent upon addressing foundational digital literacy gaps and ensuring equitable access to technological resources.

The systematic analysis of these studies underscores the necessity for a multifaceted approach to overcoming challenges in teaching and learning programming in schools. The integration of AI, multimodal analytics, and game-based learning can enhance students' engagement and problem-solving abilities, but these innovations require substantial empirical validation and infrastructural support. In addition, policymakers must prioritize the development of digital competencies among educators to ensure the effective implementation of programming curricula. Future research should focus on bridging the digital divide, refining AI-based educational tools, and developing pedagogical models that balance cognitive load with engagement strategies. Addressing these challenges will be critical in fostering computational and reflective thinking skills among students, thereby preparing them for the digital economy. The increasing integration of programming education in schools has revealed numerous challenges related to pedagogy, student engagement, and curriculum alignment. Vrbančič *et al*., [22] investigated the effectiveness of graphical versus textual programming environments in secondary school mechatronics education. Their findings suggest that students who began with textual programming before transitioning to graphical methods exhibited superior learning outcomes and knowledge transfer. This aligns with Luo *et al*., [23] who emphasize the necessity of interdisciplinary pedagogical strategies, including problem-based learning and cognitive

development, to improve programming instruction in business schools. Similarly, Zhang *et al*., [24] explored the embodied learning approach in robotics education and found that students who engaged in hands-on, interactive programming activities demonstrated higher motivation and learning engagement. These studies collectively highlight the pedagogical challenges associated with determining the most effective instructional strategies for programming education, suggesting that a structured, sequential approach from textual to graphical programming may yield better results.

Another significant challenge in teaching programming at the school level is the gap between teachers' preparedness and the theoretical frameworks guiding computational thinking (CT). Holstein *et al*., [25] examined teachers' perceptions of integrating CT through a constructionist approach using Scratch and found that while some educators fully embraced this methodology, others struggled with time constraints and curriculum alignment. This issue is further complicated by the findings of Ismail *et al*., [26] who explored the relationship between Neuro-Linguistic Programming (NLP) and psychological flexibility among secondary school students. Their study suggests that students' adaptability to programming concepts may be influenced by their cognitive flexibility, highlighting the need for tailored pedagogical strategies. Luo *et al*., [23] also reported that business students faced difficulties in error handling and theoretical application, which parallels challenges encountered in secondary education. Collectively, these studies indicate that effective programming education requires well-prepared educators, structured curriculum integration, and strategies that foster both cognitive flexibility and practical engagement.

The emergence of artificial intelligence (AI) tools, such as ChatGPT, presents additional challenges and opportunities in programming instruction. Husain [27] examined programming instructors' perspectives on the integration of AI-based chatbots in education, revealing mixed perceptions regarding its effectiveness. While AI tools can assist in debugging and providing instant feedback, concerns regarding over-reliance and reduced problem-solving skills were prevalent. Zhang *et al*., [24] demonstrated that interactive robotics education fosters engagement, yet AI-driven tools must be integrated carefully to maintain students' cognitive involvement. Additionally, Holstein *et al*., [25] stressed that balancing constructionist approaches with institutional constraints remains a major challenge.

These studies highlight the critical need for educational policies that guide the responsible use of AI in programming instruction, ensuring it supplements rather than replaces fundamental problem-solving and coding skills.

3. Methodology

3.1 Identification

For this study, a large amount of pertinent literature was selected using many crucial phases in the systematic review process. After choosing keywords, relevant terms are looked up using dictionaries, thesaurus, encyclopedias, and previous studies. Following the creation of the search strings for the Scopus and Web of Science databases, all pertinent keywords were chosen (see Table 1). For the current study project, 2444 papers were successfully obtained from both databases during the first stage of the systematic review procedure.

Table 1
The search string

Scopus	TITLE-ABS-KEY (challenges AND teaching AND learning AND programming AND schools) AND PUBYEAR > 2022 AND PUBYEAR < 2026 AND (LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (PUBSTAGE , "final")) AND (LIMIT-TO (SRCTYPE , "j")) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SUBJAREA , "SOCI")) Date of Access: March 2025
Web of Science	challenges AND teaching AND learning AND programming AND schools (Topic) and Article (Document Types) and English (Languages) and Article(Document Types) and English (Languages) and 2025 or 2024 or 2023 (Publication Years) and Article (Document Types) and 2025 or 2024 or 2023(Publication Years) and Education Educational Research (Web of Science Categories) and 6.11 Education & Educational Research (Citation Topics Meso) Date of Access: March 2025

3.2 Screening

The variety of possibly pertinent research materials is examined throughout the screening phase to find information that supports the predetermined research questions. The selection of research materials relevant to the challenges of teaching and learning programming in schools is one of the content-related criteria that are frequently used at this point. Duplicate papers are now removed from the list of papers that were obtained. 2316 articles were excluded in the first screening phase, and 128 papers were examined in the second phase using the various exclusion and inclusion criteria described in this study (see Table 2). As the main source of useful suggestions, literature (research articles) was given precedence. This includes reviews, meta-syntheses, meta-analyses, monographs, book series, chapters, and conference proceedings that have not been addressed in recent studies. Furthermore, the study was limited to English-language literature published between 2023 and 2025. 8 publications were ultimately rejected because of duplication issues.

Table 2
The selection criterion is searching

Criterion	**Inclusion**	**Exclusion**
Language	English	Non-English
Timeline	2023 – 2025	< 2023
Literature type	Journal (Article)	Conference, Book, Review
Publication Stage	Final	In Press
Categories	Education & Educational Research	Besides Education & Educational Research

3.3 Eligibility

The final set of materials for evaluation is prepared once all inclusion and exclusion criteria have been met. To help readers identify the precise research items supporting the study's findings, it is essential to disclose the full list of research items included in this sample. There are 120 things in the third tier, which is known as eligibility. Every article title and noteworthy passage was carefully examined at this stage to make sure it met the inclusion requirements and was pertinent to the goals

of the study. As a result, 94 articles were rejected since their titles and abstracts had no discernible relationship to the objectives of the study. In the end, 26 manuscripts were left for assessment (see Figure 1).

3.4 Data Abstraction and Analysis

A range of research designs (quantitative methodologies) were examined and synthesized using an integrative analysis as one of the assessment strategies. Finding pertinent subjects and subtopics was the aim of the competent study. The initial phase of the theme's development was the data collection stage. The authors carefully examined a collection of 26 articles for claims or information pertinent to the subjects of the current investigation, as seen in Figure 1. The authors then assessed the important recent research on the difficulties of school programming for teaching and learning. Both the research findings and the methods employed in each study are being examined. The author then worked with other co-authors to create themes based on the data in the context of this study. Throughout the data analysis process, a log was maintained to document any analyses, opinions, puzzles, or other ideas pertinent to the interpretation of the data. In order to identify any discrepancies in the theme design process, the authors lastly contrasted the outcomes. The writers debate any disputes between the notions among themselves, which is worth mentioning. Eventually, the generated themes were adjusted to guarantee coherence. Two specialists conducted the analysis selection to ascertain the problems' validity. By establishing the domain validity, the expert review process guarantees each subtheme's appropriateness, significance, and clarity. The questions are as follows below:

1. What are the most effective pedagogical approaches and strategies for teaching programming at different educational levels?
2. How does computational thinking and cognitive development contribute to students' learning outcomes in programming education?
3. What are the challenges and opportunities in integrating emerging technologies (e.g., AI, AR, robotics) into programming education?

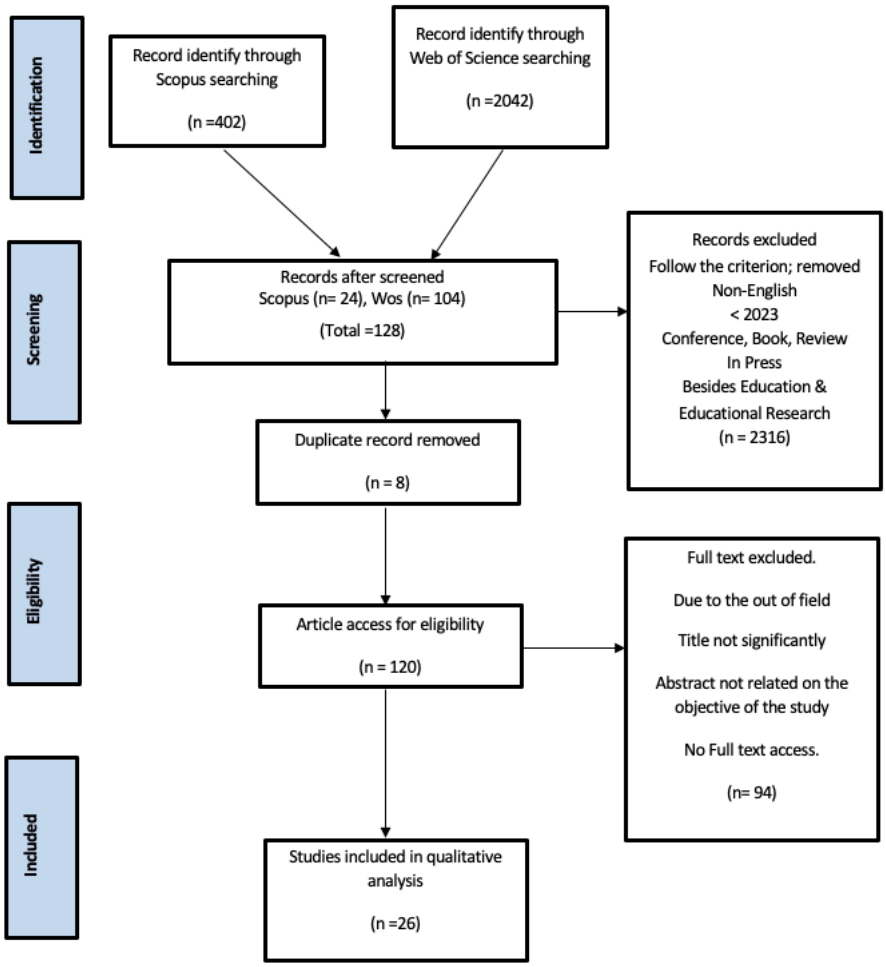

Fig. 1. Flow diagram of the proposed searching study by Moher *et al*., [28]

4. Result and Findings

Numerical results from the selected literature highlight key themes, which are (1) Pedagogical Approaches and Strategies in Teaching Programming, (2) Computational Thinking and Cognitive Development in Programming Education and (3) Technological Integration and Challenges in Programming Education.

4.1 Pedagogical Approaches and Strategies in Teaching Programming

The teaching of programming in schools presents several pedagogical challenges, necessitating innovative strategies to enhance student engagement and learning outcomes. Various studies have explored approaches such as contrasting cases, graphic organizers, self-assessment, and interdisciplinary pedagogies to address these challenges and improve the effectiveness of programming education. One of the fundamental difficulties in teaching programming is engaging novice learners, particularly at the elementary school level. Ma *et al*., [29] investigated the effectiveness of a contrasting cases approach in elementary school programming education. Their study demonstrated that this method improved learning outcomes, engagement, and cognitive load management, thereby facilitating a more structured learning process. Similarly, Yang *et al*., [30] highlighted the benefits of using graphic organizers to assist elementary students in developing

computational thinking and programming skills. Their quasi-experimental study revealed that graphic organizers served as cognitive bridges, enabling students to connect new knowledge with prior learning, ultimately enhancing problem-solving skills. Additionally, Florou *et al*., [31] emphasized the role of educators in facilitating self-assessment in programming learning, identifying the need for structured guidance to help students overcome learning obstacles. These studies collectively underscore the importance of structured instructional strategies in supporting novice programmers.

The integration of programming in diverse educational settings requires tailored approaches to optimize learning. Vrbancic *et al*., [32] investigated the effectiveness of graphical versus textual programming environments in microcontroller programming courses at the secondary education level. Their findings indicated that starting with textual programming before transitioning to graphical environments resulted in better learning gains and knowledge transfer. This aligns with Luo *et al*., [33] who explored interdisciplinary pedagogical approaches for teaching programming in business schools. Their study revealed that integrating problem-based learning, cognitive development strategies, and collaborative learning enhanced student performance and logical reasoning. Zhang *et al*., [34] further expanded on this by demonstrating how an embodied learning-based programming approach in robotics education significantly improved student engagement, motivation, and learning outcomes. These findings suggest that the sequence of instructional methods and interdisciplinary approaches can play a crucial role in improving programming education. Another key challenge in programming education lies in the selection of appropriate teaching materials and methods. Bjursten *et al*., [35] examined the factors influencing technology teachers' choice of programming materials in Swedish primary schools. Their study found that teachers often struggle with balancing curricular demands and the availability of suitable teaching resources, which impacts the effectiveness of programming instruction. Similarly, Zhang *et al*., [34] highlighted the difficulties students face when learning abstract programming concepts in robotics education, suggesting that embodied learning techniques can mitigate some of these challenges. Moreover, Florou *et al*., [31] emphasized the importance of aligning teaching strategies with student self-assessment mechanisms to support personalized learning. These studies highlight the need for adaptable and resource-efficient teaching methods to enhance programming education.

However, several pedagogical interventions have demonstrated success in improving programming education. Cheng *et al*., [36] highlighted how game-based programming environments such as CodeCombat and Scratch improved student engagement and motivation. Grover *et al*., [4] demonstrated that project-based learning, where students develop real-world applications, significantly enhanced their problem-solving abilities and computational thinking. Additionally, Barczak *et al*., [37] emphasized the effectiveness of peer collaboration in reducing learning anxiety and fostering a more supportive learning environment. These case studies underscore the need for diversified instructional approaches tailored to students' varying levels of expertise and learning preferences. In conclusion, the pedagogical challenges of teaching programming in schools require diverse and structured strategies to enhance student learning. Research suggests that employing contrasting cases, graphic organizers, interdisciplinary approaches, and structured self-assessment mechanisms can significantly improve programming education. Additionally, the sequencing of programming environments and the careful selection of instructional materials play crucial roles in optimizing learning outcomes. Future research should explore the long-term impact of these pedagogical strategies on students’ computational thinking and problem-solving abilities.

4.2 Computational Thinking and Cognitive Development in Programming Education

The integration of computational thinking into programming education has been widely acknowledged as crucial for fostering problem-solving skills and cognitive development. However, various challenges persist in effectively teaching programming in school environments. One significant challenge is the need to design engaging and effective teaching methodologies that enhance students' motivation and critical thinking skills. Table 3 shows the findings on the integration of computational thinking into programming education.

Table 3
The findings on the integration of computational thinking into programming education

No	Author Name and Year	Objectives	Methodologies	Findings	Conclusion & Future Research
1	Chang *et al.,*[38] 2023	To investigate the impact of a peer assessment-based Scrum project (PA-SP) learning system on students' learning motivation, collaboration, and communication skills in computer programming education.	Peer assessment-based Scrum project (PA-SP) learning system	PA-SP significantly improved students' learning motivation, collaboration, and communication skills.	The study highlights the effectiveness of PA-SP in enhancing programming education. Future research should explore its long-term impact and applicability across different education levels.
2	Pan *et al.,* [39] 2024	To examine the effectiveness of game-based learning in improving students' computational thinking competency and engagement.	Game-based learning methodology	Game-based learning positively influenced engagement but did not consistently enhance computational thinking competency.	Interactive approaches should be complemented with structured learning objectives. Further research should investigate how to optimize game-based learning for conceptual mastery.
3	Chen *et al.* [40] 2023	To explore self-regulation-based computational thinking learning in Taiwanese primary school students and its impact on engagement.	Self-regulation-based computational thinking learning	Self-paced learning improved engagement but required structured guidance for effective time management.	The study suggests the need for a standardized computational thinking framework. Future research should focus on optimizing self-paced learning structures.
4	Yurdakok *et al.,* [41] 2023	To analyze the effectiveness of physical programming tools like Micro:bit in enhancing computational thinking skills.	Use of Micro:bit as a physical programming tool	Physical programming tools were effective, but their impact depended on structured implementation and teacher expertise.	Teachers require better training in integrating physical programming tools. Future research should assess the scalability of such tools in different educational settings.
5	Hassan *et al.*, [42]	To examine how thinking maps with motivated	Thinking maps combined with	Thinking maps helped mitigate	Structured scaffolding techniques are

	2023	learning strategies affect students' ability to solve programming problems.	motivated learning strategies	cognitive overload and improved problem-solving skills.	necessary. Future research should explore how cognitive load management strategies can be standardized.
6	Holstein *et al.*, [43] 2025	To investigate teachers' perceptions of integrating computational thinking with school subjects using a constructionist approach via Scratch.	Constructionist approach using Scratch	Some teachers embraced the approach, while others struggled with assessment requirements.	Professional development programs are essential to enhance teachers' ability to implement computational thinking concepts. Future research should focus on policy-level interventions.
7	Naya-Varela *et al.*, [44] 2023	To evaluate the effectiveness of the Robobo SmartCity model as an educational tool for computational intelligence learning.	Robobo SmartCity model implementation	The model provided innovative learning experiences, but adoption was hindered by infrastructure limitations and teacher training gaps.	Effective technology integration requires addressing resource constraints. Future research should explore ways to enhance accessibility and teacher readiness for such tools.

Cognitive load theory by Sweller, [45] plays a crucial role in understanding students' difficulties in learning programming. The intrinsic complexity of programming concepts, such as abstraction and algorithmic logic, contributes to cognitive overload, particularly for novice learners. Research suggests that reducing extraneous cognitive load through structured instructional strategies such as worked examples, scaffolding, and adaptive learning pathways can enhance learning efficiency [46]. Additionally, differentiated instruction, including visual programming environments like Scratch for beginners and textual programming for advanced learners, can help accommodate diverse cognitive capacities and learning speeds [34]. Incorporating cognitive load management techniques into programming education is essential for fostering better retention and problem-solving skills.

The reviewed studies collectively emphasize the multifaceted challenges of teaching and learning programming in schools. Key issues include the need for structured and engaging teaching methodologies, standardized curriculum models, cognitive load management, teacher preparedness, student self-efficacy, and effective technology integration. Addressing these challenges requires a collaborative effort among educators, policymakers, and researchers to develop comprehensive strategies that bridge the gap between theoretical approaches and practical classroom implementation.

4.3 Technological Integration and Challenges in Programming Education

The integration of technology in programming education presents significant challenges, particularly in secondary schools, where students often struggle with complex programming concepts and logic structures. Traditional teaching methodologies fail to sustain engagement, necessitating alternative approaches such as open educational resources (OERs) to enhance learning outcomes. Pereira *et al*., [47] highlight the role of REA-LP, an OER designed to facilitate programming

instruction by integrating multimedia content and interactive components. Their empirical findings suggest that students benefit from such tools, exhibiting increased motivation and comprehension. However, challenges persist due to limited access to technology and pedagogical frameworks that fail to optimize digital resources for diverse learning needs. Similarly, Husain [48] underscores the need for well-structured pedagogical designs when incorporating artificial intelligence-driven platforms like ChatGPT in programming instruction, emphasizing that unstructured AI use may lead to reliance on automated responses rather than fostering problem-solving skills. Pappa et al., [49] further corroborate these findings, indicating that teachers in primary education often lack the confidence and training to effectively integrate technology, exacerbating difficulties in programming instruction.

A major issue in programming education is the variability in teachers' preparedness to deliver technology-integrated instruction. In Swedish primary schools, teachers independently select programming learning environments (PLEs) without standardized guidelines, leading to inconsistent implementation [35]. The preference for visual programming languages (VPLs) like Scratch is prevalent, yet many teachers lack adequate pedagogical content knowledge (PCK) to transition students from block-based programming to textual coding. This gap in professional development results in disparities in students' foundational programming knowledge. Pereira *et al*., [47] similarly note that while REA-LP promotes engagement, its effectiveness hinges on instructors' ability to integrate it effectively within curricula. Additionally, Pappa *et al*., [49] report that teachers struggle with unclear curriculum frameworks and insufficient professional development opportunities, further complicating programming instruction. The integration of AI-based instructional tools such as ChatGPT offers potential benefits but also raises concerns regarding dependency and pedagogical efficacy. Husain [48] identifies ChatGPT as a supportive tool for students in coding exercises, debugging, and generating alternative solutions. However, programming instructors' express concerns about students' overreliance on AI-generated code, which may hinder their development of critical thinking skills. Furthermore, immersive environments such as Augmented Reality (AR) and Virtual Reality (VR) have shown promise in making abstract programming concepts more tangible [24]. A more structured technology integration strategy, incorporating both emerging and traditional tools, is essential to mitigate barriers such as limited access to resources and inadequate teacher training.

Pereira *et al*., [47] similarly argue that while OERs provide structured learning opportunities, their success is contingent upon students' ability to engage actively rather than passively consume content. Additionally, Bjursten *et al*., [35] highlight systemic challenges, including the limited availability of structured professional development courses, which leave educators without sufficient resources to incorporate emerging technologies effectively. A critical barrier to effective programming education is the lack of a cohesive strategy for integrating emerging technologies into existing curricula. The absence of well-defined pedagogical frameworks leads to inconsistent technology adoption across educational institutions. Pappa *et al*., [49] emphasize the need for professional development initiatives to equip educators with the necessary skills for integrating programming technologies into their teaching. Moreover, Husain [48] notes that while AI tools can streamline instructional delivery, their adoption should be accompanied by structured curricular changes to mitigate potential drawbacks such as diminished analytical reasoning among students. The reliance on visual programming languages without a clear transition strategy further complicates the learning process, as observed by Bjursten *et al*., [35] who advocate for targeted training in textual programming methodologies to ensure continuity in students' computational skill development.

Given these challenges, a comprehensive approach that includes professional training, curriculum standardization, and effective technological implementation is essential. Pereira *et al*., [47] suggest

that interactive learning environments such as REA-LP can address engagement issues, yet they require structured integration into broader instructional strategies. Similarly, Pappa *et al*., [49] recommend the establishment of teaching communities to facilitate knowledge-sharing and support among educators. Husain [48] underscores the importance of balancing AI integration with traditional pedagogical methods to prevent overreliance on automated solutions. Addressing these concerns will require ongoing research and policy adjustments to ensure that programming education remains both effective and adaptive to technological advancements.

5. Conclusions

By addressing the pedagogical challenges of programming education necessitates the implementation of diverse and structured strategies, such as contrasting cases, graphic organizers, and interdisciplinary approaches, which have been shown to enhance student engagement and learning outcomes. Continued exploration of these methods will be essential for optimizing programming instruction and fostering students' computational thinking and problem-solving skills. The effective integration of computational thinking into programming education faces significant challenges, including the need for engaging teaching methodologies, standardized curricula, and improved teacher preparedness. Addressing these multifaceted issues through collaboration among educators, policymakers, and researchers is essential for enhancing student learning outcomes and fostering a more effective programming education environment.

In summary, the integration of technology in programming education faces significant hurdles, primarily stemming from inconsistent teacher preparedness and a lack of cohesive pedagogical frameworks. To enhance learning outcomes, it is crucial to implement structured professional development, standardized curricula, and effective use of educational resources, ensuring that both educators and students can navigate the complexities of programming in an increasingly digital landscape.

Acknowledgement

This research was not funded by any grant. We sincerely appreciate the invaluable insights and support provided by Associate Professor Dr. Noor Dayana Abd Halim, Dr. Mohd Fadzil Abdul Hanid, Dr. Rita Inderawati, and the Ministry of Education. Their expertise, constructive feedback, and unwavering encouragement were instrumental in shaping the direction of this research and ensuring the successful completion of this manuscript.

References

[1] Angeli, Charoula, and Michail Giannakos. "Computational thinking education: Issues and challenges." *Computers in human behavior* 105 (2020): 106185. https://doi.org/10.1016/j.chb.2019.106185

[2] Maraza-Quispe, Benjamín, Ashtin Maurice Sotelo-Jump, Olga Melina Alejandro-Oviedo, Lita Marianela Quispe-Flores, Lenin Henry Cari-Mogrovejo, Walter Cornelio Fernandez-Gambarini, and Luis Ernesto Cuadros-Paz. "Towards the development of computational thinking and mathematical logic through scratch." *International Journal of Advanced Computer Science and Applications* 12, no. 2 (2021): 332-338. https://doi.org/10.14569/IJACSA.2021.0120242

[3] Zhao, Li, Xiaohong Liu, Chenhui Wang, and Yu-Sheng Su. "Effect of different mind mapping approaches on primary school students' computational thinking skills during visual programming learning." *Computers & Education* 181 (2022): 104445. https://doi.org/10.1016/j.compedu.2022.104445

[4] Grover, Shuchi, and Roy Pea. "Computational thinking: A competency whose time has come." *Computer science education: Perspectives on teaching and learning in school* 19, no. 1 (2018): 19-38. https://doi.org/10.5040/9781350057142.ch-003

[5] Lishinski, Alex, Aman Yadav, Richard Enbody, and Jon Good. "The influence of problem solving abilities on students' performance on different assessment tasks in CS1." In *Proceedings of the 47th ACM technical symposium on computing science education*, pp. 329-334. 2016. https://doi.org/10.1145/2839509.2844596

[6] Saqr, Mohammed, Ville Tuominen, Teemu Valtonen, Erkko Sointu, Sanna Väisänen, and Laura Hirsto. "Teachers' learning profiles in learning programming: The big picture!." In *Frontiers in Education*, vol. 7, p. 840178. Frontiers Media SA, 2022. https://doi.org/10.3389/feduc.2022.840178

[7] Adedoyin, Adeyinka, F. O. Enebe, R. A. Oyekunle, and N. A. Balogun. "Design and implementation of an online teaching and learning management system." *FUDMA Journal of Sciences* 7, no. 1 (2023): 148-155. https://doi.org/10.33003/fjs-2023-0701-1266

[8] Sentance, Sue, and Andrew Csizmadia. "Computing in the curriculum: Challenges and strategies from a teacher's perspective." *Education and information technologies* 22 (2017): 469-495. https://doi.org/10.1007/s10639-016-9482-0

[9] Brennan, Karen, and Mitchel Resnick. "New frameworks for studying and assessing the development of computational thinking." In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, vol. 1, p. 25. 2012.

[10] Batiha, Q., N. Sahari, N. Aini, and N. Mohd. "Adoption of visual programming environments in programming learning." *International Journal on Advanced Science, Engineering and Information Technology* 12, no. 5 (2022): 1921. https://doi.org/10.18517/ijaseit.12.5.15500

[11] C. Watson and F. W. B. Li, "Version of attached le : Failure Rates in Introductory Programming Revisited," *Proceedings of the 2014 conference on Innovation technology in computer science education (ITiCSE'14)*, vol. 44, no. July, 2016. https://doi.org/10.1145/2591708.2591749

[12] Florou, C., G. Stamoulis, A. Xenakis, and A. Plageras. "The role of educators in facilitating students' self-assessment in learning computer programming concepts: addressing students' challenges and enhancing learning." *Education and Information Technologies* (2024): 1-24. https://doi.org/10.1007/s10639-024-13172-2

[13] Sandstrak, Grethe, Bjorn Klefstad, Arne Styve, and Kiran Raja. "Analyzing Pedagogic Practice and Assessments in a Cross-Campus Programming Course." *IEEE Transactions on Education* (2024). https://doi.org/10.1109/TE.2024.3465870

[14] Yang, Tzu-Chi, and Zhi-Shen Lin. "Enhancing elementary school students' computational thinking and programming learning with graphic organizers." *Computers & Education* 209 (2024): 104962. https://doi.org/10.1016/j.compedu.2023.104962

[15] Oralbayeva, Nurziya, Zhansaule Telisheva, Aida Amir, Aida Zhanatkyzy, Arna Aimysheva, and Anara Sandygulova. "Moveable Älıpbi: Design of Montessori-Based Child-Robot Interaction for Long-Term Alphabet Learning." *International Journal of Social Robotics* (2024): 1-16. https://doi.org/10.1007/s12369-024-01189-z

[16] Ienco, Andrea, Bruno Tiribilli, Chiara D'Errico, Armida Torreggiani, Valentina Biasini, Sabrina Gualtieri, and Pietro Galizia. "Sorting Materials using Programmable Lego© Robot: an Educational Activity to Promote Sustainability among Youngsters." In *Conference Proceedings. New Perspectives in Science Education 2024*. 2024.

[17] Dabengwa, Israel Mbekezeli, Sibonile Moyo, Smart Ncube, Tinashe Byron Gashirai, Daga Makaza, Paul Makoni, Notice Pasipamire et al. "Exploring digital competences in Zimbabwean secondary schools using a multimodal view: a hermeneutical phenomenography study." *Cogent Education* 11, no. 1 (2024): 2387911. https://doi.org/10.1080/2331186X.2024.2387911

[18] Fante, Chiara, Fabrizio Ravicchio, and Flavio Manganello. "Navigating the Evolution of Game-Based Educational Approaches in Secondary STEM Education: A Decade of Innovations and Challenges." *Education Sciences* 14, no. 6 (2024): 662. https://doi.org/10.3390/educsci14060662

[19] Dai, Yun, Ziyan Lin, Ang Liu, and Wenlan Wang. "An embodied, analogical and disruptive approach of AI pedagogy in upper elementary education: An experimental study." *British Journal of Educational Technology* 55, no. 1 (2024): 417-434. https://doi.org/10.1111/bjet.13371

[20] Cohn, Clayton, Caitlin Snyder, Joyce Horn Fonteles, Ashwin TS, Justin Montenegro, and Gautam Biswas. "A multimodal approach to support teacher, researcher and AI collaboration in STEM+ C learning environments." *British Journal of Educational Technology* 56, no. 2 (2025): 595-620. https://doi.org/10.1111/bjet.13518

[21] Fante, Chiara, Fabrizio Ravicchio, and Flavio Manganello. "Navigating the Evolution of Game-Based Educational Approaches in Secondary STEM Education: A Decade of Innovations and Challenges." *Education Sciences* 14, no. 6 (2024): 662. https://doi.org/10.3390/educsci14060662

[22] Vrbančič, Franc, and Slavko Kocijančič. "Strategy for learning microcontroller programming—a graphical or a t. extual start?." *Education and Information Technologies* 29, no. 4 (2024): 5115-5137. https://doi.org/10.1007/s10639-023-12024-9

[23] Luo, Xiaojun, and Ismail Adelopo. "Exploring pedagogies, opportunities and challenges of teaching and learning

programming in business school." *Journal of International Education in Business* 18, no. 1 (2025): 26-46. https://doi.org/10.1108/JIEB-05-2024-0060

[24] Zhang, Xinli, Yuchen Chen, Danqing Li, Lailin Hu, Gwo-Jen Hwang, and Yun-Fang Tu. "Engaging young students in effective robotics education: An embodied learning-based computer programming approach." *Journal of Educational Computing Research* 62, no. 2 (2024): 532-558. https://doi.org/10.1177/07356331231213548

[25] Holstein, Simona, and Anat Cohen. "Scratch Teachers' Perceptions of Teaching Computational Thinking with School Subjects in a Constructionist Approach." *Thinking Skills and Creativity* (2025): 101772. https://doi.org/10.1016/j.tsc.2025.101772

[26] Ismail, Ahmed Mohamed Bani, and Faisal Jaber Al-Ajmi. "Neuro-linguistic programming and its relationship with psychological flexibility among secondary school students in Najran region." *Qubahan Academic Journal* 4, no. 1 (2024): 210-223. https://doi.org/10.48161/qaj.v4n1a379

[27] Husain, Anas. "Potentials of ChatGPT in computer programming: Insights from programming instructors." *Journal of Information Technology Education: Research* 23 (2024): 002. https://doi.org/10.28945/5240

[28] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred reporting items for systematic reviews and meta-analyses: the PRISMA Statement," 2009. https://doi.org/10.1371/journal.pmed.1000097

[29] Ma, Ning, Jinglong Qian, Kaixin Gong, and Yao Lu. "Promoting programming education of novice programmers in elementary schools: A contrasting cases approach for learning programming." *Education and Information Technologies* 28, no. 7 (2023): 9211-9234. https://doi.org/10.1007/s10639-022-11565-9

[30] Yang, Tzu-Chi, and Zhi-Shen Lin. "Enhancing elementary school students' computational thinking and programming learning with graphic organizers." *Computers & Education* 209 (2024): 104962. https://doi.org/10.1016/j.compedu.2023.104962

[31] Florou, C., G. Stamoulis, A. Xenakis, and A. Plageras. "The role of educators in facilitating students' self-assessment in learning computer programming concepts: addressing students' challenges and enhancing learning." *Education and Information Technologies* (2024): 1-24. https://doi.org/10.1007/s10639-024-13172-2

[32] Vrbančič, Franc, and Slavko Kocijančič. "Strategy for learning microcontroller programming—a graphical or a textual start?." *Education and Information Technologies* 29, no. 4 (2024): 5115-5137. https://doi.org/10.1007/s10639-023-12024-9

[33] Luo, Xiaojun, and Ismail Adelopo. "Exploring pedagogies, opportunities and challenges of teaching and learning programming in business school." *Journal of International Education in Business* 18, no. 1 (2025): 26-46. https://doi.org/10.1108/JIEB-05-2024-0060

[34] Zhang, Xinli, Yuchen Chen, Danqing Li, Lailin Hu, Gwo-Jen Hwang, and Yun-Fang Tu. "Engaging young students in effective robotics education: An embodied learning-based computer programming approach." *Journal of Educational Computing Research* 62, no. 2 (2024): 532-558. https://doi.org/10.1177/07356331231213548

[35] Bjursten, Eva-Lena, Tor Nilsson, and Gunnar Jonsson. "Factors influencing Swedish grades 4–6 technology teachers' choice of teaching and learning material in programming education." *International Journal of Technology and Design Education* 34, no. 4 (2024): 1275-1303. https://doi.org/10.1007/s10798-023-09860-8

[36] Cheng, Yu-Ping, Chin-Feng Lai, Yun-Ting Chen, Wei-Sheng Wang, Yueh-Min Huang, and Ting-Ting Wu. "Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform." *Computers & Education* 200 (2023): 104794. https://doi.org/10.1016/j.compedu.2023.104794

[37] Barczak, Andre LC, Anuradha Mathrani, Binglan Han, and Napoleon H. Reyes. "Automated assessment system for programming courses: a case study for teaching data structures and algorithms." *Educational technology research and development* 71, no. 6 (2023): 2365-2388. https://doi.org/10.1007/s11423-023-10277-2

[38] Chang, Shao-Chen, and Charoenchai Wongwatkit. "Effects of a peer assessment-based scrum project learning system on computer programming's learning motivation, collaboration, communication, critical thinking, and cognitive load." *Education and Information Technologies* 29, no. 6 (2024): 7105-7128. https://doi.org/10.1007/s10639-023-12084-x

[39] Pan, Yanjun, Elizabeth L. Adams, Leanne R. Ketterlin-Geller, Eric C. Larson, and Corey Clark. "Enhancing middle school students' computational thinking competency through game-based learning." *Educational technology research and development* 72, no. 6 (2024): 3391-3419. https://doi.org/10.1007/s11423-024-10400-x

[40] Chen, Chien-Yu, Shih-Wen Su, Yu-Zhi Lin, and Chuen-Tsai Sun. "The Effect of Time Management and Help-Seeking in Self-Regulation-Based Computational Thinking Learning in Taiwanese Primary School Students." *Sustainability* 15, no. 16 (2023): 12494. https://doi.org/10.3390/su151612494

[41] Yurdakök, Ezgi Arzu, and Filiz Kalelioğlu. "The Effect of Teaching Physical Programming on Computational Thinking Skills and Self-Efficacy Perceptions Towards Computational Thinking." *Journal of Educational Computing Research* 62, no. 3 (2024): 785-815. https://doi.org/10.1177/07356331231220313

[42] Hassan, Nurul Nadia, Siti Salbiah Hamzah, Nurkhuzaimah Fazreen Mohd Jalaluddin, Muhammad Zaffwan Idris, and Che Soh Said. "Mediation of Motivated Strategies for Learning for Thinking Maps Involvement Towards

Metacognitive Awareness." *Asian Journal of University Education* 19, no. 2 (2023): 381-394. https://doi.org/10.24191/ajue.v19i2.22235

[43] Holstein, Simona, and Anat Cohen. "Scratch Teachers' Perceptions of Teaching Computational Thinking with School Subjects in a Constructionist Approach." *Thinking Skills and Creativity* (2025): 101772. https://doi.org/10.1016/j.tsc.2025.101772

[44] Naya-Varela, Martin, Sara Guerreiro-Santalla, Tamara Baamonde, and Francisco Bellas. "Robobo smartcity: An autonomous driving model for computational intelligence learning through educational robotics." *IEEE Transactions on Learning Technologies* 16, no. 4 (2023): 543-559. https://doi.org/10.1109/TLT.2023.3244604

[45] Sweller, John. "Cognitive load during problem solving: Effects on learning." *Cognitive science* 12, no. 2 (1988): 257-285. https://doi.org/10.1016/0364-0213(88)90023-7

[46] Paas, Fred GWC, and Jeroen JG Van Merriënboer. "Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach." *Journal of educational psychology* 86, no. 1 (1994): 122. https://doi.org/10.1037//0022-0663.86.1.122

[47] Pereira, Diego EF, and Rodrigo D. Seabra. "Open Educational Resource for Studying Algorithms and Programming Logic: An Approach to the Technical Level Integrated with Secondary School." *Informatics in Education* 22, no. 3 (2023): 441-462. https://doi.org/10.15388/infedu.2023.17

[48] Husain, Anas. "Potentials of ChatGPT in computer programming: Insights from programming instructors." *Journal of Information Technology Education: Research* 23 (2024): 002. https://doi.org/10.28945/5240

[49] Pappa, Christina Ioanna, Despoina Georgiou, and Daniel Pittich. "Technology education in primary schools: addressing teachers' perceptions, perceived barriers, and needs." *International Journal of Technology and Design Education* 34, no. 2 (2024): 485-503. https://doi.org/10.1007/s10798-023-09828-8