Programming and Algorithm Fundamentals
E-ISSN: 3123-979X

Research Article

Integrating Computational Thinking and Adaptive Curriculum Frameworks to Enhance Problem-Solving Skills in Undergraduate Programming Education Across Diverse Learning Contexts

Nicodemus Rahanra [1], Ahmad Ashifuddin Aqham [2], and Eko Siswanto [2,*]

1 Universitas Satya Wiyata Mandala nicorh73@gmail.com
2 Universitas Sains dan Teknologi Komputer ashif@stekom.ac.id
3 Universitas Sains dan Teknologi Komputer Eko.siswanto@stekom.ac.id
* Corresponding Author : nicorh73@gmail.com

Abstract: This study investigates the integration of computational thinking (CT) principles with adaptive curricula to enhance problem-solving skills in undergraduate programming education. Traditional programming curricula often emphasize syntax and basic concepts, neglecting critical problem-solving strategies. The adaptive curriculum framework used in this study combines CT skills such as decomposition, pattern recognition, abstraction, and algorithmic thinking with personalized learning experiences. A mixed-method approach, combining qualitative and quantitative research, was employed to assess the effectiveness of this integrated approach. The results show significant improvements in students' problem-solving abilities, conceptual understanding, and engagement compared to a control group following a traditional curriculum. Students in the experimental group, which received the adaptive curriculum, demonstrated better performance in applying algorithms and debugging code. Additionally, students expressed higher levels of engagement and motivation, suggesting that the personalized learning environment fostered greater academic involvement. The study highlights the importance of integrating CT principles with adaptive learning frameworks to create a more inclusive and effective learning environment that accommodates diverse learning needs. The findings suggest that adaptive curricula can bridge gaps in traditional education by providing personalized support and ensuring that students progress at their own pace. This approach is especially beneficial for programming education, where both conceptual understanding and practical problem-solving skills are critical for success. Future research should explore the long-term impact of adaptive learning frameworks and investigate how these technologies can be integrated with traditional teaching methods to maximize their effectiveness.

Keywords: Computational thinking; Adaptive learning; Programming education; Problem-solving skills; Curriculum design.

Received: November 20, 2025
Revised: Desember 30, 2025
Accepted: January 14, 2026
Published: January 17, 2026
Curr. Ver.: January 19, 2026

Copyright: © 2025 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY SA) license (https://creativecommons.org/licenses/by-sa/4.0/)

1. Introduction

Undergraduate programming education faces several significant challenges, particularly when it comes to developing problem-solving skills. Many students struggle with debugging, understanding programming fundamentals, and applying abstract concepts. These challenges often result in higher dropout rates among programming students [1], [2]. The abstract nature of programming, combined with the intensive learning required, makes it difficult for students to grasp key concepts, leading to disengagement and lack of progress [2], [3]. Traditional methods, which focus more on theory and syntax, often do not adequately address these difficulties or engage students effectively. Studies on digital learning environments also indicate that strengthening digital literacy and technology-based learning

ecosystems can improve students' engagement and capacity to solve computational problems [4], [5].

Problem-solving is a critical skill in programming, involving the ability to develop algorithms and debug code efficiently. However, many educational approaches place too much emphasis on syntax and less on problem-solving strategies [3], [6]. The lack of focus on problem-solving leaves students ill-prepared for tackling complex programming tasks. One promising approach to addressing this gap is the integration of computational thinking (CT), which includes decomposition, pattern recognition, and algorithmic thinking. By teaching problem-solving strategies explicitly and incorporating CT principles, educators can significantly enhance students' ability to address programming challenges [7], [8]. Personalized learning frameworks and adaptive instructional models have also been shown to support the development of computational thinking and improve students' cognitive engagement in programming courses [9], [10].

Moreover, traditional curricula often fail to accommodate the diverse cognitive abilities and learning needs of students. This rigidity can lead to students having to retake courses, delaying their academic progression [10], [11]. Rigid teaching methods may not be flexible enough to engage students effectively, causing frustration and disengagement, particularly for those with different learning profiles [12]. To address these issues, adaptive curricula that personalize learning to the cognitive characteristics of students have been shown to improve learning outcomes and engagement [7], [9]. In addition, interactive and technology-supported learning approaches such as gamification and hybrid learning models have been reported to increase student motivation, participation, and learning performance in digital education contexts [13], [14].

Integrating computational thinking principles into programming curricula offers a potential solution to these challenges. Computational thinking skills, such as decomposition, pattern recognition, abstraction, and algorithmic thinking, are essential for effective problem-solving in programming [8], [10]. These skills help students break down complex problems into manageable parts, identify patterns, and design algorithms to solve them. By incorporating these principles into an adaptive curriculum framework, it is possible to create a more engaging and effective learning environment that caters to diverse learning needs [8], [15]. In addition, the integration of technology-supported learning and digital literacy frameworks can further enhance the effectiveness of computational thinking instruction in modern educational environments [5].

The goal of this study is to explore the integration of computational thinking principles with adaptive curriculum frameworks to enhance problem-solving skills in undergraduate programming education. This approach seeks to provide a more personalized learning experience that takes into account the cognitive abilities and learning contexts of individual students. By focusing on the development of computational thinking skills and utilizing adaptive learning strategies, the study aims to improve students' ability to solve programming problems and increase their overall engagement with the subject [7], [10]. Previous research also suggests that adaptive and technology-enhanced learning environments can significantly improve student engagement and participation in digital learning contexts [14].

Ultimately, this study aims to address the limitations of traditional programming curricula by providing a more flexible, engaging, and personalized learning experience. By integrating computational thinking and adaptive frameworks, students will be better equipped to tackle programming challenges, leading to improved problem-solving skills, increased academic success, and greater overall engagement in programming courses [10], [15]. Furthermore, the adoption of adaptive digital learning ecosystems and innovative educational technologies may support the development of more resilient and effective learning systems in higher education [5], [16].

2. Literature Review

Computational Thinking in Programming Education

Computational thinking (CT) plays a critical role in programming education by providing a structured approach to problem-solving, system design, and understanding human behavior using key computer science concepts. CT involves breaking down complex problems into manageable parts, recognizing patterns, and creating algorithms to solve them. This process is not only fundamental to programming but also extends to various other disciplines, making

CT an essential skill for success in a technology-driven society. As such, CT is increasingly being incorporated into educational curricula at all levels, from K-12 to higher education, to ensure that students are equipped with the necessary skills to navigate and thrive in the digital age [17]. The integration of digital literacy and technology-supported learning frameworks has also been recognized as an important component in strengthening students' computational and analytical capabilities in modern education systems [5].

Various educational frameworks, including CDIO (Conceive, Design, Implement, Operate), Scratch, and game-based learning, are being employed to teach CT. These frameworks emphasize the integration of theoretical knowledge with practical, hands-on learning experiences, which help make CT more accessible and engaging for students [18], [19]. By blending theory with practice, these frameworks not only improve students' understanding of computational thinking but also foster deeper engagement in programming education. In addition, the integration of interactive and technology-enhanced learning models has been shown to improve student engagement and participation in digital learning environments [4], [14]. This approach encourages students to apply their CT skills in real-world contexts, ultimately enhancing their problem-solving abilities and preparing them for the demands of a technology-driven world.

Adaptive Curriculum Frameworks

Adaptive curricula are designed to meet the unique learning needs of individual students by offering personalized learning experiences. These systems utilize advanced tools such as intelligent tutoring systems, AI chatbots, and machine learning to create customized learning paths based on each student's cognitive abilities and progress [20]. Adaptive learning environments dynamically adjust instructional materials and learning activities to match the learner's progress and understanding. As a result, adaptive learning systems have been widely recognized as an effective approach for improving student-centered learning experiences in modern digital education systems [5].

Adaptive learning systems are proven to significantly enhance academic performance, motivation, and engagement by providing immediate feedback and adjusting the pace of learning to align with the student's needs [21]. By continuously analyzing learner interactions and performance data, these systems can recommend personalized learning activities that strengthen conceptual understanding and problem-solving skills. This approach helps maintain student interest by offering tailored challenges that promote continuous development, ensuring that students can move forward at a comfortable yet challenging pace [22]. In addition, adaptive digital learning environments that integrate interactive technologies have also been shown to increase student engagement and participation in modern education contexts [14].

Despite the benefits, the implementation of adaptive learning systems faces several challenges. Key obstacles include faculty buy-in, the need for robust technical infrastructure, and ensuring equitable access to advanced learning technologies [23]. These challenges can hinder the widespread adoption of adaptive learning tools in educational institutions. Institutional readiness, technological infrastructure, and adequate training for educators are critical factors in ensuring the successful implementation of adaptive learning systems. Furthermore, integrating advanced digital technologies into learning ecosystems requires careful planning to ensure both accessibility and sustainability of educational innovations [16].

Challenges in Undergraduate Programming Education

Undergraduate programming education is frequently challenged by several factors that impede students' success. Key issues include difficulties in problem-solving, debugging, grasping programming fundamentals, and sustaining motivation throughout the course [2], [24]. These challenges are exacerbated by the abstract nature of programming concepts, which can overwhelm students, particularly those with limited exposure to the subject. Moreover, many traditional programming curricula focus primarily on syntax and technical skills, while neglecting the development of essential problem-solving abilities. As a result, students are often unprepared to apply theoretical knowledge to practical, real-world programming challenges [12]. Studies on digital learning environments also indicate that improving digital literacy and integrating technology-supported learning ecosystems can help students better understand complex computational concepts [5].

Furthermore, traditional educational models frequently fail to accommodate the diverse cognitive abilities and learning styles of students, leading to disengagement and frustration.

The rigidity of these curricula does not cater to the varying needs of learners, causing some students to fall behind. In addition, the lack of personalized learning strategies often limits students' opportunities to develop computational thinking and analytical skills in programming courses. To address these issues, there is a growing need for more flexible and adaptive learning approaches that tailor instruction to individual students' needs. Adaptive learning frameworks have shown promise in improving student engagement and performance by providing personalized support and learning paths [25]. In modern educational environments, integrating interactive and technology-enhanced learning models has also been shown to significantly improve student motivation, engagement, and participation in digital learning settings [4], [14]. These approaches are particularly valuable in addressing the challenges faced by students in programming education.

Adaptive Curricula as a Solution to Educational Challenges

Adaptive learning systems present a promising solution to the challenges faced in undergraduate programming education. These systems personalize the learning experience by providing tailored feedback, scaffolding, and problem-solving tasks that meet the unique needs of each student. Studies have shown that adaptive learning strategies, such as guided discovery and adaptive sequencing of learning tasks, significantly enhance problem-solving skills and improve student engagement [21], [24]. By offering a more individualized approach, adaptive curricula allow students to progress at their own pace, reinforcing key programming concepts and building critical problem-solving abilities [25]. Furthermore, these systems help address gaps in traditional curricula, such as the lack of personalized feedback and support for students who struggle with foundational concepts [26], [27]. In addition, the development of adaptive digital learning ecosystems supported by emerging technologies has further strengthened the effectiveness of personalized learning environments in higher education [16].

Empirical evidence supports the effectiveness of adaptive learning strategies in enhancing students' problem-solving skills and overall academic performance. Studies indicate that students in adaptive learning environments perform significantly better in problem-solving tasks, demonstrating enhanced analytical thinking and application of programming concepts [7], [28]. The adaptive learning process, which adjusts to the student's level of understanding, ensures that learning is optimized and that students are continuously challenged at the right level. This fosters a deeper understanding of programming and strengthens students' ability to apply what they have learned to complex problems [21], [22]. Furthermore, the integration of adaptive learning systems with game-based elements, such as those used in platforms like Scratch, has proven effective in engaging students and improving their problem-solving abilities in a more interactive and enjoyable way [26]. Research on technology-enhanced learning environments also indicates that gamification and digital learning platforms can significantly improve learner motivation and engagement in modern educational contexts [14].

Gaps in the Literature

Despite the promising results of adaptive learning systems, significant gaps remain in the literature that need to be addressed. While existing studies highlight the benefits of adaptive learning, more comprehensive research is required to explore the long-term effects of these systems across diverse educational contexts. There is a need to better understand how adaptive learning can be applied in various disciplines and environments to assess its sustained impact on student outcomes [19], [29]. Additionally, further studies should investigate how adaptive learning systems can be integrated with traditional teaching methods to optimize their effectiveness. Combining adaptive learning with existing instructional models could help overcome barriers related to implementation, such as faculty buy-in, infrastructure limitations, and the adaptation of course materials [30]. Previous studies also emphasize the importance of developing robust technological infrastructures and institutional readiness to successfully integrate adaptive learning technologies within higher education environments [24].

Another important area for future research is the scalability and accessibility of adaptive learning technologies. As these systems grow in popularity, it is crucial to ensure that they are accessible to all students, regardless of their socio-economic background or the resources available to them. This includes investigating ways to make adaptive learning technologies affordable and ensuring equitable access to these systems, particularly in underfunded educational institutions [24], [25]. Furthermore, the development of inclusive digital learning

ecosystems supported by emerging technologies has become an important focus in recent educational research, particularly in ensuring that digital learning platforms can support diverse learners and learning contexts [14], [16]. Exploring these areas will help address existing challenges and ensure that adaptive learning can be implemented in a way that benefits a diverse range of students, creating a more inclusive and effective educational experience.

3. Proposed Method

This study employs a mixed-method approach, combining qualitative and quantitative research to evaluate the impact of integrating computational thinking (CT) principles with adaptive curriculum frameworks in undergraduate programming education. The research includes a quasi-experimental design with a control and experimental group, where the experimental group receives the adaptive curriculum. Data is collected through pre- and post-test assessments, student surveys, and classroom observations to measure improvements in problem-solving skills, engagement, and conceptual understanding. Quantitative analysis uses statistical methods like paired t-tests and ANOVA to assess the effectiveness of the intervention, while qualitative analysis involves thematic analysis of student feedback and learning experiences. This approach allows for a comprehensive evaluation of both the academic outcomes and student experiences with the adaptive learning tools.

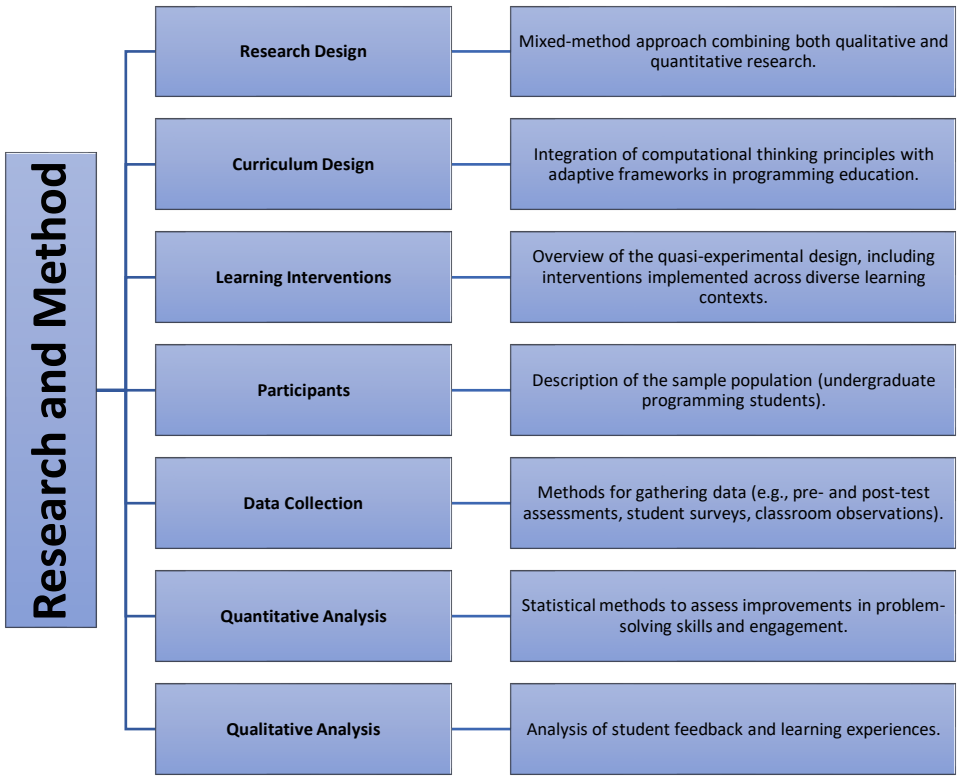


Figure 1. Flowchart structure.

Research Design

This study employs a mixed-method approach, integrating both qualitative and quantitative research methodologies to provide a comprehensive analysis of the effectiveness of integrating computational thinking (CT) principles with adaptive curriculum frameworks in programming education. The use of a mixed-method approach allows for a robust understanding of both the numerical impact on student outcomes (quantitative) and the in-depth exploration of student experiences and feedback (qualitative). This approach is well-suited for educational research as it combines the strengths of both data types to offer a richer, more nuanced understanding of the research problem.

Curriculum Design

The curriculum design for this study integrates computational thinking principles with adaptive frameworks to enhance students' problem-solving skills in programming education. The CT framework focuses on key principles such as decomposition, pattern recognition, abstraction, and algorithmic thinking, which are essential for students to develop strong programming problem-solving skills. These principles are then combined with adaptive learning strategies that personalize the learning experience based on each student's progress and cognitive needs. The adaptive framework utilizes tools like intelligent tutoring systems and AI chatbots to tailor instructional activities to students' individual learning styles and abilities. This blended approach ensures that students receive targeted support, allowing for a more flexible and engaging learning experience.

Learning Interventions

The study adopts a quasi-experimental design to assess the effectiveness of the integrated curriculum. This design includes a control group and an experimental group, with the experimental group receiving the intervention of the adaptive curriculum framework incorporating computational thinking principles. The quasi-experimental design is suitable for this context as it allows for comparisons between groups in real-world classroom settings, where random assignment is not feasible. Interventions are implemented across diverse learning contexts, which include varying classroom environments, teaching methodologies, and student backgrounds, ensuring that the study captures a wide range of student experiences and learning outcomes. The learning interventions focus on delivering personalized, scaffolded problem-solving tasks and using adaptive learning tools to support each student's unique learning needs.

Participants

The sample population for this study consists of undergraduate programming students enrolled in a university-level computer science course. Participants are selected based on their enrollment in introductory programming courses, which are typically characterized by a broad range of prior knowledge and varying levels of experience with programming concepts. This diversity provides a suitable context for evaluating the impact of adaptive curricula on students with different learning needs. The sample size is determined using power analysis to ensure adequate statistical power for detecting significant effects of the intervention. All participants provide informed consent, and ethical considerations are strictly adhered to throughout the study.

Data Collection

Data collection for this study involves multiple methods to capture both the quantitative and qualitative aspects of student learning. Pre- and post-test assessments are used to measure improvements in students' problem-solving skills and conceptual understanding of programming topics. These assessments are designed to evaluate both basic programming knowledge and the application of problem-solving strategies in real-world scenarios. In addition to the assessments, student surveys are conducted to gather data on students' perceptions of the learning experience, including their engagement and motivation levels. Classroom observations are also employed to assess the level of student interaction with the curriculum and their engagement with the adaptive learning tools. These data collection methods ensure that the study captures a holistic view of student progress and experiences.

Quantitative Analysis

Quantitative data analysis involves using statistical methods to assess the improvements in problem-solving skills and engagement among participants. Paired t-tests are employed to compare pre- and post-test scores within each group, allowing for the measurement of changes in problem-solving abilities over the course of the intervention. Additionally, analysis of variance (ANOVA) is used to compare the differences in performance between the experimental and control groups, providing insight into the effectiveness of the adaptive curriculum framework relative to traditional teaching methods. Statistical software such as SPSS or R is used to perform these analyses, ensuring the reliability and accuracy of the results.

Qualitative Analysis

Qualitative data analysis involves a thematic analysis of student feedback and learning experiences. Student surveys and classroom observations provide rich, descriptive data on how students engage with the adaptive learning tools and curriculum. Open-ended survey responses are coded and categorized to identify recurring themes related to student perceptions of the curriculum's effectiveness, their motivation, and their problem-solving strategies. Classroom observations are also analyzed to understand how students interact with the adaptive tools and collaborate with peers. This qualitative analysis complements the quantitative findings, offering deeper insights into the nuances of student learning and engagement. The combination of these qualitative insights with quantitative data allows for a comprehensive understanding of the impact of the integrated curriculum on student outcomes.

4. Results and Discussion

The study found that integrating computational thinking (CT) principles into an adaptive curriculum significantly improved students' problem-solving skills, conceptual understanding, and engagement. The experimental group, which used the adaptive curriculum, showed marked progress in decomposing complex problems, identifying patterns, and applying algorithms, compared to the control group following a traditional curriculum. The adaptive approach also addressed diverse learning needs by personalizing tasks and feedback, allowing students to progress at their own pace and stay engaged. In contrast to traditional programming courses that focus mainly on syntax and theory, the adaptive curriculum enhanced problem-solving abilities and better equipped students to tackle real-world programming challenges, demonstrating its superiority in fostering deeper learning and critical thinking.

Results

The results of the study indicated notable improvements in students' problem-solving skills, conceptual understanding, and engagement after the implementation of the adaptive curriculum framework. The pre- and post-test assessments revealed that the experimental group, which received the adaptive curriculum integrating computational thinking principles, showed significant progress in problem-solving tasks. These students demonstrated improved ability to decompose problems, identify patterns, and develop algorithms, leading to better debugging and programming skills. Additionally, survey responses from the experimental group indicated higher levels of engagement and motivation, suggesting that the personalized and adaptive learning environment successfully maintained student interest throughout the course.

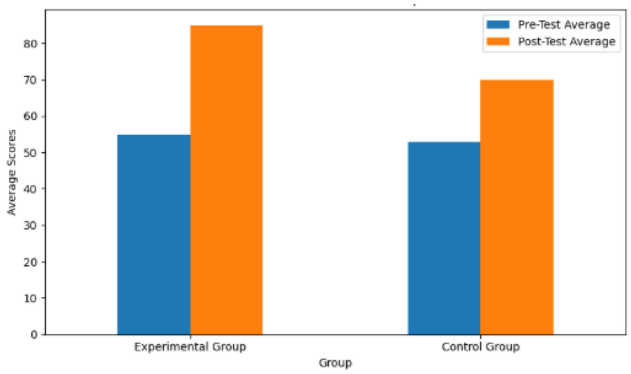


Figure 2. Pre-and Post-Test Results Comparison.

Here is a bar chart comparing the average scores of the experimental and control groups for pre- and post-test assessments. The data shows significant improvements in the problem-solving skills of the experimental group, which received the adaptive curriculum, compared to the control group that followed a traditional curriculum. The experimental group's post-test score significantly increased from 55 to 85, while the control group's score only rose from 53 to 70, indicating the positive impact of the adaptive curriculum.

Furthermore, the data collected from classroom observations confirmed these findings, as students in the experimental group were observed to engage more actively in discussions and collaborative problem-solving tasks. The control group, which followed a traditional curriculum, did not exhibit the same level of improvement, particularly in terms of engagement and application of computational thinking principles. This discrepancy highlights the effectiveness of the adaptive curriculum framework in fostering deeper learning and problem-solving abilities compared to traditional methods.

Discussion

The integration of computational thinking (CT) principles into the adaptive curriculum played a critical role in enhancing students' problem-solving abilities. By emphasizing key CT skills such as decomposition, pattern recognition, and algorithmic thinking, students were able to break down complex programming challenges into manageable parts, making it easier for them to approach and solve problems systematically. These principles provided students with a structured approach to problem-solving, which is essential in programming. As a result, students were not only able to understand programming concepts better but also applied them more effectively in real-world scenarios. The personalized nature of the curriculum, which tailored the level of difficulty to each student's cognitive abilities, ensured that the learning experience was optimally challenging and engaging, promoting higher-order thinking and analytical skills.

The adaptive curriculum framework also proved effective in accommodating diverse learning needs. Traditional programming courses often fail to address the varying cognitive abilities and learning styles of students, which can lead to disengagement or frustration, particularly among those who struggle with foundational programming concepts. However, the adaptive approach allowed students to progress at their own pace, with personalized scaffolding and feedback that met their individual needs. This flexibility helped bridge the gap for students who might have struggled in a traditional, one-size-fits-all curriculum. Additionally, students who demonstrated proficiency in the subject were provided with more advanced tasks, ensuring that they were continuously challenged. This ability to tailor instruction to individual needs significantly enhanced both engagement and academic performance.

In comparison to traditional programming curricula, the findings of this study underscore the superiority of adaptive learning frameworks. Traditional curricula often focus on teaching programming syntax and theoretical knowledge, with less emphasis on problem-solving strategies or real-world application. While these foundational aspects are important, they do not fully equip students with the practical problem-solving skills needed in the field of programming. In contrast, the adaptive curriculum not only reinforced programming fundamentals but also integrated CT principles and personalized learning strategies that promoted the development of essential problem-solving skills. This approach allowed students to not only learn programming but also think critically and solve problems effectively, a key component of successful programming education.

5. Comparison

The comparison between the adaptive curriculum approach and traditional uniform curricula highlights several key differences in terms of student engagement, problem-solving ability, and overall learning outcomes. Traditional curricula often focus on a rigid, one-size-fits-all approach, where the pace and content delivery are fixed for all students, regardless of their individual learning needs or cognitive abilities. This can result in students who struggle with foundational concepts falling behind, while more advanced learners may become disengaged due to a lack of challenge. In contrast, the adaptive curriculum framework implemented in this study provides personalized learning experiences, tailoring the content, pace, and difficulty to each student's needs. This flexibility allows for a more inclusive learning environment, where students can progress at their own pace and receive targeted support based on their individual strengths and weaknesses.

One of the key strengths of adaptive learning is its ability to foster better learning outcomes by offering personalized feedback and scaffolding. In traditional curricula, students often receive generalized feedback, which may not address their specific challenges or provide the depth of understanding necessary to overcome obstacles. Adaptive learning systems, however, provide immediate, targeted feedback that helps students identify and correct

mistakes in real-time. This not only enhances students' problem-solving abilities but also encourages active engagement with the material. As students receive feedback tailored to their progress, they are more likely to remain motivated and persist through challenging concepts, leading to improved learning outcomes. This personalized approach has been shown to significantly improve academic performance, especially in subjects like programming, where hands-on application and problem-solving are key to success.

The integration of computational thinking (CT) principles in the adaptive curriculum contributed significantly to stronger learning outcomes compared to traditional methods. Traditional programming courses often emphasize syntax and basic programming knowledge, without placing enough focus on the cognitive skills required for effective problem-solving. Computational thinking, which includes principles such as decomposition, pattern recognition, abstraction, and algorithmic thinking, provides students with a structured framework to approach complex problems. By integrating CT into the curriculum, students were not only able to better understand programming concepts but were also equipped with the tools to solve real-world problems more effectively. This approach, which fosters critical thinking and analytical skills, resulted in deeper learning and better retention of programming concepts. The ability to think computationally gave students a distinct advantage over those in traditional curricula, where problem-solving was often taught in a more abstract and less structured manner.

6. Conclusions

This study demonstrates that integrating computational thinking principles with adaptive curricula significantly enhances problem-solving skills, conceptual understanding, and student engagement in undergraduate programming education. The key findings indicate that students who participated in the adaptive curriculum framework showed marked improvements in their ability to decompose problems, recognize patterns, and apply algorithms effectively. These students were also more engaged and motivated compared to those in traditional curricula. The integration of computational thinking not only improved students' technical skills but also fostered critical thinking and a deeper understanding of programming concepts.

The importance of integrating computational thinking with adaptive curricula lies in its ability to cater to the diverse learning needs of students. By personalizing the learning experience, adaptive curricula provide targeted feedback and scaffolding that help students progress at their own pace, making the learning process more inclusive and effective. This personalized approach ensures that students are consistently challenged without feeling overwhelmed, which is especially important in subjects like programming that require both conceptual understanding and practical application.

Based on the findings, it is recommended that adaptive curricula be implemented in diverse undergraduate programming courses to accommodate the varied cognitive abilities and learning styles of students. The use of computational thinking principles in these curricula should be prioritized to help students develop the problem-solving skills necessary for success in programming and beyond. Educators and institutions should consider integrating adaptive learning technologies, such as intelligent tutoring systems and AI chatbots, to provide personalized learning experiences and enhance student outcomes.

Future research should focus on exploring the long-term impact of adaptive learning frameworks across different educational contexts. While this study demonstrates the effectiveness of adaptive curricula in enhancing student engagement and performance, further investigation is needed to understand how these frameworks can be scaled and sustained over time. Future studies should also explore the integration of adaptive learning with traditional teaching methods to maximize their potential and address challenges related to implementation, such as infrastructure and faculty adaptation.

References

[1] B. Banic, M. Konecki, and M. Konecki, "Pair Programming Education Aided by ChatGPT," in *2023 46th ICT and Electronics Convention, MIPRO 2023 - Proceedings*, 2023, pp. 911 – 915. doi: 10.23919/MIPRO57284.2023.10159727.

[2] R. Luo, N. Li, M. Leach, E. G. Lim, and S. Saunders, "A Systematic Review of Undergraduate Programming Difficulties: Highlighting AI Tools to Address Learning Challenges," in *2025 International Conference on Artificial Intelligence and Education, ICAIE 2025*, 2025, pp. 30 – 34. doi: 10.1109/ICAIE64856.2025.11158145.

[3] G. Sambe, K. Drame, and A. Basse, "Towards a Framework to Scaffold Problem-solving Skills in Learning Computer Programming," in *International Conference on Computer Supported Education, CSEDU - Proceedings*, 2021, pp. 323 – 330. doi: 10.5220/0010446503230330.

[4] I. Englishtina, H. R. D. Putranti, D. Danang, and A. A. B. Pujiati, "SITENAR CERYA as an Innovation in English Language Learning at SMP Stella Matutina Salatiga: Merging Technology and Folktales," *REKA ELKOMIKA J. Pengabdi. Kpd. Masy.*, vol. 5, no. 3, pp. 241–250, 2024.

[5] D. Danang, H. Haryani, Q. Aini, F. A. Ramahdan, and J. Edwards, "Empowering Digital Literacy Through Blockchain Based Alphasign for Secure and Sustainable E-Governance," 2025.

[6] H. Dawkins *et al.*, "Development of a checklist tool for teaching problem-solving skills," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 2019, p. 301. doi: 10.1145/3304221.3325560.

[7] N. Dehbozorgi and M. Roopaei, "Improving Computational Thinking Competencies in STEM Higher Education," in *2024 IEEE Integrated STEM Education Conference, ISEC 2024*, 2024. doi: 10.1109/ISEC61299.2024.10664997.

[8] E. Çela, N. R. Vajjhala, and P. Eappen, *Foundations of computational thinking and problem solving for diverse academic fields*. 2024. doi: 10.4018/979-8-3693-1974-1.ch001.

[9] Y. Pechorina, K. Anderson, and P. Denny, "Metacodenition: Scaffolding the Problem-Solving Process for Novice Programmers," in *ACM International Conference Proceeding Series*, 2023, pp. 59 – 68. doi: 10.1145/3576123.3576130.

[10] J. Guevara-Reyes, M. Vinueza-Morales, E. Ruano-Lara, and C. Vidal-Silva, "Implementation of personalized frameworks in computational thinking development: implications for teaching in software engineering," *Front. Educ.*, vol. 10, 2025, doi: 10.3389/feduc.2025.1584040.

[11] S. Digennaro and A. Borgogni, "The Bounded Rationality of the Educational Choices," *Encyclopaideia*, vol. 19, no. 41, pp. 21–36, 2015, doi: 10.6092/issn.1825-8670/5045.

[12] Q. Batiha, N. Sahari, N. Aini, and N. Mohd, "Adoption of Visual Programming Environments in Programming Learning," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 12, no. 5, pp. 1921 – 1930, 2022, doi: 10.18517/ijaseit.12.5.15500.

[13] D. Danang, F. A. Prasetya, and E. Siswanto, "Design of Intelligent Street Lighting Systems Based on Motion and Ambient Light Sensors," *J. Multidiscip. Res. Technol.*, vol. 2, no. 1, pp. 65–79, 2026.

[14] H. R. Putranti, R. Retnowati, A. A. Sihombing, and D. Danang, "Performance Assessment through Work Gamification: Investigating Engagement," *South African J. Bus. Manag.*, vol. 55, no. 1, pp. 1–12, 2024.

[15] S. Ganesan, A. Lionelle, C. Gill, and C. Brodley, "Does Reducing Curricular Complexity Impact Student Success in Computer Science?," in *SIGCSE TS 2025 - Proceedings of the 56th ACM Technical Symposium on Computer Science Education*, 2025, pp. 360 – 366. doi: 10.1145/3641554.3701915.

[16] D. Danang, A. B. Santoso, and M. U. Dewi, "CICA Framework: Harnessing CSR, AI, and Blockchain for Sustainable Digital Culture," *Int. J. Adv. Comput. Sci. \& Appl.*, vol. 16, no. 11, 2025.

[17] C. Proctor, *Computational thinking*. 2022. doi: 10.1016/B978-0-12-818630-5.13078-7.

[18] N. Hoić-Božić, J. Mezak, and K. Tomljenović, "Enhancing teachers' computational thinking skills through game based learning," in *CEUR Workshop Proceedings*, 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85075288263&partnerID=40&md5=fc098b92d104efca0f7201502f77bd9e

[19] A. Kaplan *et al.*, "Teaching programming at scale," in *CEUR Workshop Proceedings*, 2020, pp. 2 – 6. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85080865407&partnerID=40&md5=47bc2683e55ccc2c4fb4dbad6509587b

[20] X. Chen, A. Mitrovic, and M. Mathews, "Does adaptive provision of learning activities improve learning in SQL-tutor?," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10331 LNAI, pp. 476 – 479, 2017, doi: 10.1007/978-3-319-61425-0_44.

[21] X. Chen, A. Mitrovic, and M. Mathews, "Exploring adaptive strategies for providing learning activities," in *UMAP 2018 - Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, 2018, pp. 139 – 145. doi: 10.1145/3209219.3209221.

[22] S. Sato, Y. Sato, S. Nakamura, and Y. Miyadera, "Proposal of a System Enabling Adaptive Support Based on Transition of Each Learner's Source Codes," in *Proceedings - 2022 International Conference on Computational Science and Computational Intelligence, CSCI 2022*, 2022, pp. 2145 – 2146. doi: 10.1109/CSCI58124.2022.00389.

[23] H. Khosravi, S. Sadiq, and D. Gasevic, "Development and adoption of an adaptive learning system reflections and lessons learned," in *SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 58 – 64. doi: 10.1145/3328778.3366900.

[24] D. Ginting, D. Sabudu, Y. Barella, A. Madkur, R. Woods, and M. K. Sari, "Student-centered learning in the digital age: in-class adaptive instruction and best practices," *Int. J. Eval. Res. Educ.*, vol. 13, no. 3, pp. 2006 – 2019, 2024, doi: 10.11591/ijere.v13i3.27497.

[25] V. Hegde, R. J. Shastry, and A. R. Pai, "Predicting the Grade by Exploring the Relationship Between Computational Thinking and Academic Performance Through Learning Analytics," *Lect. Notes Networks Syst.*, vol. 520, pp. 667 – 677, 2023, doi: 10.1007/978-981-19-5331-6_68.

[26] A. M. De Jesus and I. F. Silveira, "A collaborative game-based learning framework to improve computational thinking skills," in *Proceedings - 2019 International Conference on Virtual Reality and Visualization, ICVRV 2019*, 2019, pp. 161–166. doi: 10.1109/ICVRV47840.2019.00038.

[27] A. S. Najar, A. Mitrovic, and B. M. McLaren, "Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum," *User Model. User-adapt. Interact.*, vol. 26, no. 5, pp. 459 – 491, 2016, doi: 10.1007/s11257-016-9181-y.

[28] D. Maryono, Sajidan, M. Akhyar, Sarwanto, B. T. Wicaksono, and N. P. T. Prakisya, "NgodingSeru.com: an adaptive e-learning system with gamification to enhance programming problem-solving skills for vocational high school students," *Discov. Educ.*, vol. 4, no. 1, 2025, doi: 10.1007/s44217-025-00581-9.

[29] C. Núñez-Hernández, F. Avilés-Castillo, and J. Buele, "Adaptive Learning Platforms and Their Influence on Higher Education: A Scoping Review," *Lect. Notes Comput. Sci.*, vol. 15806 LNCS, pp. 358 – 370, 2025, doi: 10.1007/978-3-031-93564-0_22.

[30] G. Deckard and K. K. Seo, *A critical review of two automated adaptive teaching systems in mathematics and chemistry: Cognitive Tutor and ALEKS*. 2024. doi: 10.1163/9789004702813_010.