INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

Problem-solving Techniques and Tools for Computer Programming at Higher Education: A Review

Krishna Palod[1]*, Dr. Nishita Parekh[2], Dr. Pravin Shrinath[3]

1 Research Scholar, Technology Management Department, Mukesh Patel School of Technology Management & Engineering, NMIMS (Deemed-to-be-University), Mumbai, Maharashtra, India
2 Assistant Professor, Technology Management Department, Mukesh Patel School of Technology Management & Engineering, NMIMS (Deemed-to-be-University), Mumbai, Maharashtra, India
3 Associate Professor, Computer Engineering Department, Mukesh Patel School of Technology Management & Engineering, NMIMS (Deemed-to-be-University), Mumbai, Maharashtra, India

OPEN ACCESS

Received: 18/06/2025
Accepted: 31/10/2025
Published: 01/12/2025

Citation: Palod K, Parekh N, Shrinath P (2025) Problem-solving Techniques and Tools for Computer Programming at Higher Education: A Review. Indian Journal of Science and Technology 18(43): 3432-3444. https://doi.org/10.17485/IJST/v18i43.922

* **Corresponding author**.

Krishna.Palod01@nmims.in

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: Computer programming is essential in today's world, yet it remains challenging to teach and learn, especially for novices. Problem-solving skills are a core predictor of programming success. This review examines how problem-solving practices can enhance programming education. **Methods:** The review synthesizes findings from 37 research studies focused on problem-solving in context to programming education published between 2018 to 2025. The selected studies were analysed and classified into two categories: instructional techniques and supporting tools. The review also examines how these categories intersect to enhance students' problem-solving abilities in programming. **Findings:** The analysis identified several effective instructional approaches for enhancing problem-solving abilities in programming education, including problem-based learning, puzzle-based learning, game-based learning, flowchart-based learning, and mind mapping. A variety of educational tools were identified to support the implementation of these techniques. A key contribution of this review is the establishment of explicit relationships between these pedagogical techniques and the corresponding tools, offering a novel framework for instructional design in programming education. However, most studies emphasized short-term learning outcomes, with limited exploration of long-term retention, scalability, or suitability for advanced learners. Drawing on these insights, the review provides practical recommendations for educators to effectively develop students' foundational programming and problem-solving skills. It also highlights research gaps and suggests directions for future studies in programming education. **Novelty:** This review introduces an original framework by mapping specific problem-solving techniques to associated educational tools, offering actionable guidance for improving instructional practices in programming education through targeted strategy-tool integration.

Keywords: Computer Programming; Programming Competency; Problem-solving; Problem-solving Techniques; Problem-solving Tools

1 Introduction

Programming is widely regarded as one of the most challenging subjects within computer science, often associated with high dropout and failure rates among novice programmers[1]. The primary challenge in programming education lies in the concurrent acquisition of various skill sets like mastering fundamental programming concepts, understanding the semantics and syntax of the programming language, and cultivating effective problem-solving skills[2]. Traditional classroom instruction often emphasizes programming concepts and language syntax, typically through well-defined problems. However, when students are confronted with ill-defined or open-ended tasks, many struggle to apply their knowledge due to insufficient problem-solving ability[3]. Numerous studies have indicated problem-solving as a key predictor of proficiency in programming[4],[5]. Research by Eteng et al.[6] suggests that incorporating problem-solving training prior to formal programming instruction can significantly improve the students' programming ability.

Despite the recognized importance of problem-solving in programming education, existing systematic literature reviews exhibit notable limitations. For instance, Medeiros's et al.[7] review studies research articles published between 2010 and 2016 exhaustively, but its relevance to recent developments in problem-solving methodologies is limited requiring further research. Their findings lack conclusive evidence supporting the effectiveness of the identified methods and tools. Similarly, Santos et al.[8] focused on predictive cognitive skills between 2011 and 2020, identified and implemented most of the problem-solving techniques and tools in solo institutional contexts with more focus on school level education limiting its generalizability for higher studies. Another review by Santos et al.[9] underscored the need for integrated teaching frameworks and longitudinal studies to assess the sustained impact of innovative instructional approaches and covered the studies from 2012 to 2018. A recent review by Zhu *et al.*[10] addresses the use of large language models in programming education, is restricted to the years 2021-2024 and focusing primarily on AI tools rather than established problem-solving methodologies. These reviews collectively lack a comprehensive and up-to-date synthesis of established problem-solving techniques and tools for programming education in higher education, and they fail to systematically assess their effectiveness across diverse learning contexts.

Therefore, this review addresses the need to consolidate recent studies, evaluate the effectiveness of problem-solving approaches, categorize them into techniques and tools, and establishes their interrelationships. Addressing this gap, the present study introduces an integrated dual-layer framework that explicitly maps problem-solving techniques to their corresponding educational tools. This comprehensive synthesis provides educators and higher education institutions with actionable insights for enhancing programming competency and preparing students for industry demands. By linking instructional strategies with technological resources, the framework offers a consolidated and practical model for programming education, marking a clear departure from earlier fragmented reviews.

To address the identified research gaps and attain the stated contributions, this systematic literature review is guided by the following research questions:

RQ1: What problem-solving techniques have been employed in undergraduate programming education, and what evidence exists for their implementation and effectiveness?

RQ2: What tools have been used to foster problem-solving skills in programming education and what evidence exists about their implementation and efficiency?

2 Review Methodology

This Review is conducted to gain insights into problem-solving skills for computer programming, techniques and tools for fostering these skills. The review methodology followed is according to the prescribed guidelines by Kitchenham and Brereton[11]. All stages of literature screening, study selection, and data extraction were conducted by a single reviewer. This approach ensured consistency across the review process and followed established guidelines for systematic literature reviews.

Search Process:

For this review a literature search on prominent online databases, namely IEEE Xplore, Science Direct, Web of Science, ACM Digital Library, and Springer Link, known for their high relevance to Computer Science. Our search spanned journal or conference publications from January 1, 2018, to May 31, 2025, with a focus on English-language papers. The search process involved exploring titles, keywords, and abstracts to identify relevant literature. Through iterative refinement based on trial and error across diverse databases, we crafted a consolidated search phrase that precisely addressed our research area:

((”problem solving” OR ”problem-solving skill”) AND (”computer programming” OR ”learning coding” OR ”Learning programming” OR ”teaching programming” OR ”novice programming” OR ”introductory programming” OR ”CS1” OR ”Programming course”) AND (”higher education” OR ”university” OR ”college”))

The search for relevant literature led to the identification of 1178 publications, all of which underwent additional screening. Table 1 contains details about the initial search on databases.

Table 1. Details of study search and selection by database

Database	Search result
IEEE Xplore	254
ScienceDirect	110
SpringerLink	544
Web of Science	27
ACM Digital library	243
TOTAL	**1178**

Finally, 37 articles were selected for this review based on the search process as described in Figure 1.

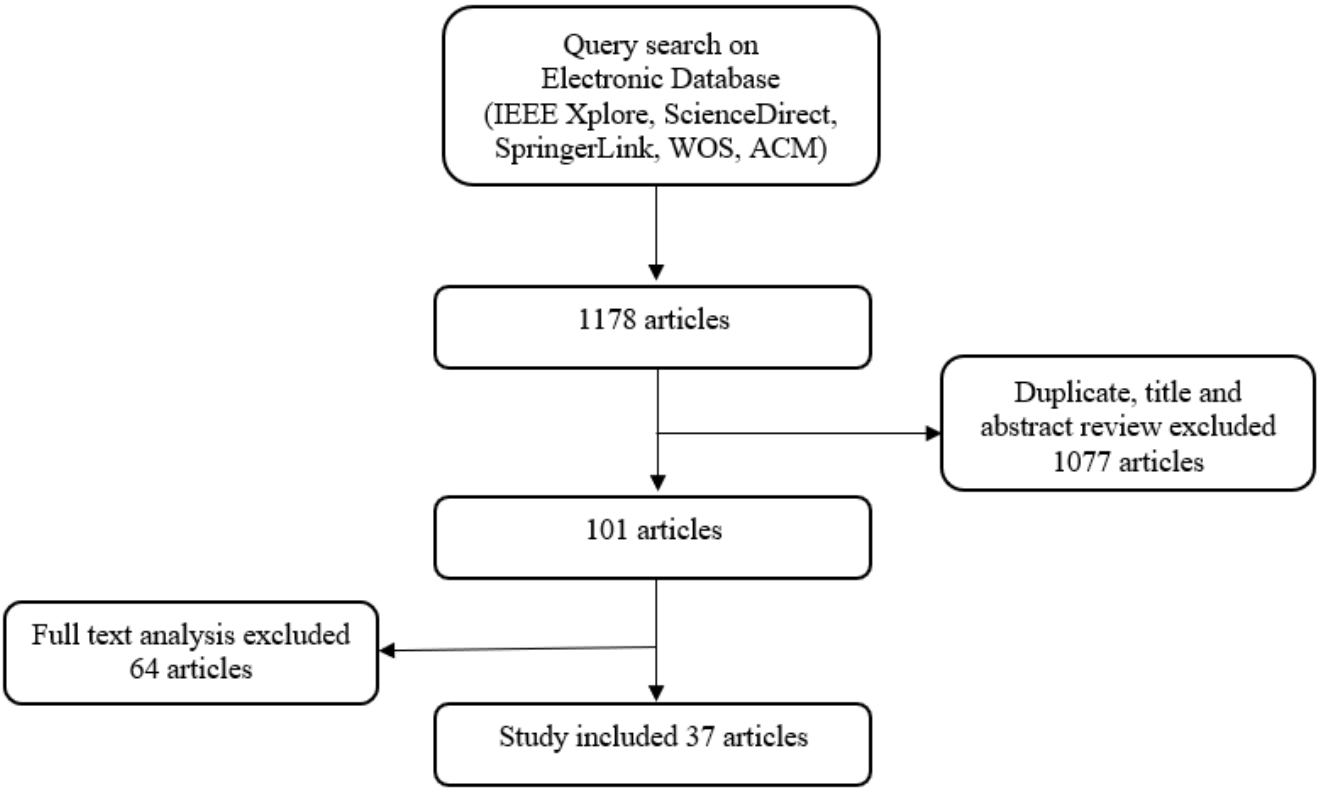

Fig 1. Search Process

Table 2 outlines the criteria for selecting which literature papers should be taken into consideration. In addition to the standard criteria of language, availability of the complete text, research field, and publishing date, the determination of the article's content relevancy was also crucial. This review has examined exclusively articles addressing problem-solving as a predictor for computer programming, excluding studies involving problem-solving techniques or tools applied at the school level.

3 Overview

3.1 Problem-solving Techniques for Computer Programming: Evidence and Implementation (RQ1)

This review identifies five distinct problem-solving techniques across the literature focusing on undergraduate programming education between 2018 and 2025. Table 3 list the details about articles reviewed for each technique and Figure 2

Table 2. Criteria for selection of literature review

Criteria	What's included in the selection of literature
Period of studies	2018 – 2025
Publication Type	Peer-reviewed journal articles, conference proceedings, online articles
Contents	The article addresses problem-solving as a predictor for Computer Programming. Methods, techniques, and tools for teaching problem-solving in the context of Computer Programming. Only studies carried out for higher education are included.
Language	English
Availability	Available online as full-text

visualizes the frequency of each technique in the reviewed programming studies. The graph indicates that Game-Based Learning/Gamification and Problem-Based Learning are the most widely used approaches in recent studies, with Puzzle-Based Learning appearing third. Mind Mapping and Flowcharting, though less frequently studied, are developing as an important supplementary technique.

Table 3. Teaching Techniques for Problem-solving

Techniques	Reviewed Articles
Problem-Based Learning	Tareq and Yusof (2024) [12], Aires et al. (2023) [13], Hegade et al. (2023) [14], Hawa et al. (2022) [15], Aires et al. (2021) [16], Skalka & Drlík (2020) [17], Souza and Bittencourt (2019) [18], Janpla & Piriyasurawong (2018) [19]
Flowchart-Based learning	Zhou et al. (2024) [20], Zhang et al. (2023) [21], Rahman et al. (2020) [22]
Game-Based Learning/Gamification	Salvatierra et al (2025) [23], Wong et al (2025) [24], Mellado & Cubillos (2024) [25], Johan et al (2024) [26], Tun et al. (2023) [27], Videnovik et al. (2023) [28], Uiphanit et al (2023) [29], Cuervo-Cely et al. (2022) [30], Khaleel et al (2019) [31]
Puzzle-Based learning	Hou et al. (2024) [32], Ericson et al. (2023) [33], Ericson et al. (2022) [34], Oyelere (2019) [35]
Mind Mapping	Kefalis et al. (2025) [36], Gul et al. (2023) [37], Liu et al. (2018) [38]

3.1.1. Problem-Based Learning (PBL)

In recent research, Problem-based learning (PBL) is defined as constructivist, student-centric instructional approach that engages learners in collaboratively investigating and solving complex, real-world or ill-structured problems, rather than focusing solely on well-defined textbook problems [14], [16]. In a programming context, PBL usually involves students collaborating in small teams to analyse, investigate, discuss, and develop solutions to open-ended programming problems, guided in structured phases, from problem identification and brainstorming to self-learning and reflection, while the instructor acts as a facilitator or coach rather than a traditional lecturer [12], [17], [18]. Out of eight studies reviewed four studies incorporated experimental methodology [12], [15], two studies followed qualitative study [16], [17], and two articles implemented mixed-method research [18], [19]. Study by Aries et al. [13] reported overall pass rate increased by 81% post experiment. One of the studies, reported that students' interest in programming was increased by 50% after the intervention [14].

As reported by all studies, PBL improved students' grasp of core programming concepts and their ability to relate theory to practical problem-solving. Active, real-world problem scenarios and collaborative settings increased motivation, persistence, and willingness to tackle complex programming tasks [13], [16], [19]. Students developed the capacity to transfer classroom learning to authentic situations, often designing or coding solutions relevant to daily technology use or industry practices. Group-based activities fostered teamwork, communication skills, and mutual support among students [13], [18]. Implemented PBL frameworks addressed multiple cognitive domains, including application, analysis, synthesis, and evaluation; learning activities mapped well to all levels of Bloom's taxonomy [17].

For successful implementation of PBL requires skilled instructor facilitation, which may not be scalable to larger classes or resource-limited institutions [12], [14], [15], [18]. Certain studies did not include a traditional-lecture control group, limiting direct comparison of learning gains attributable to PBL [13], [14], [15]. Many interventions spanned at single institute as single course

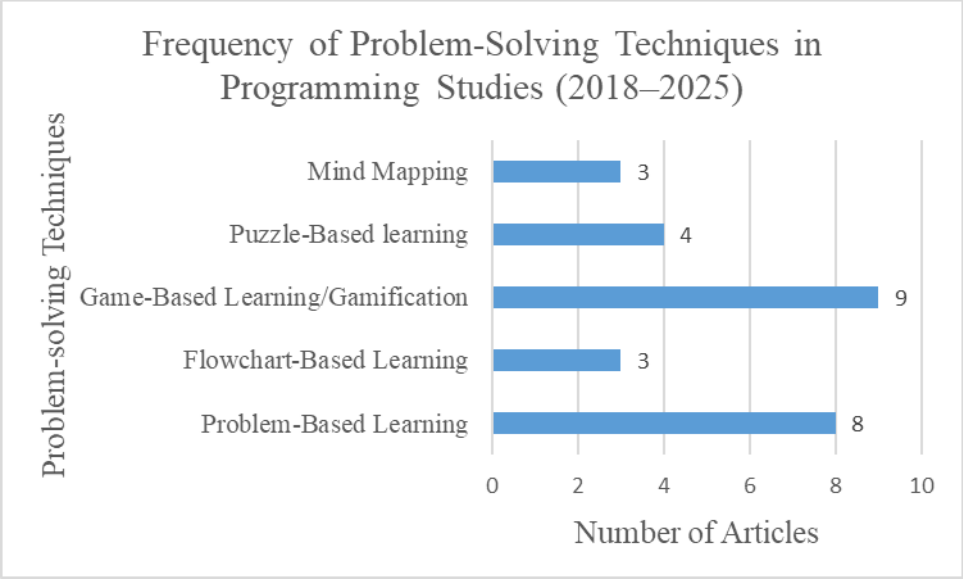

Fig 2. Problem-solving Techniques Studies

or semester or as workshop, providing insights into immediate but not long-term impacts[13],[14],[15],[18],[19]. Consistently assessing process-oriented outcomes, individual team member contributions, and higher-order thinking skills remains difficult, as standardized quantitative measures of PBL effectiveness are not yet consistently established across contexts[14],[16],[18],[19].

3.1.2. Flowchart-Based Learning

Flowcharts visually represent the step-by-step overview of a process or algorithm, making them especially helpful for beginners tackling programming problems.[20]. They support novice programmers by highlighting progress, clarifying next steps, and breaking complex problems into manageable parts[21]. A study by Rahman et al.[22] used flowchart-based platforms like RAPTOR and demonstrated significant improvements in students' problem-solving, coding skills and motivation for programming. All the studies used a quasi-experimental design, where groups of students experience different instructional strategies, such as traditional coding versus flowchart-based visual programming environments and reported positive enhancement in programming.

Students have report increased motivation and comfort in tackling coding problems after using flowchart tools[22]. This approach inspires learners to emphasis on algorithmic thinking and logical structure rather than syntax memorization[20]. Flowcharts are best viewed as a complementary approach, and its limitations highlight studies conducted using specific cohorts, programming languages, or educational contexts may not generalize to all student populations or advanced programming courses[21],[22]. With the prompt progress in artificial intelligence and machine learning, contemporary research has largely shifted away from flowchart-based methods, focusing instead on leveraging these cutting-edge technologies in programming education.

3.1.3. Game-based learning and Gamification

Game-based learning utilizes real games to render the learning experience making it enjoyable and interactive, whereas gamification include game elements to non-game context to encourage students. Both approaches enhance problem-solving skills, but GBL involves playing games as part of the lesson, while gamification transforms the learning process into a game altogether[24],[26]. Students can boost their programming knowledge by playing a game or by designing a game as part of the course[28]. The majority used experimental study as research methodology, among which two studies employed mixed method approach. Most of the studies had control group to compare the results.

In Wong et al.[24] research, experimental group utilized a Sokoban-inspired game-based learning tool for improving programming sills, and statistical analysis indicated greater learning gains for experimental group in comparison with control group, which was delivered conventional instructional approaches. One study discovered that using gamified Moodle exercises with points, leaderboards, and badges improve learning outcomes in data structures programming courses, irrespective of task

difficulty. Students in the gamified group reported lively, more enjoyable and helpful learning experience. The experimental group's post-test scores were significantly higher than the control group's, even after adjusting for pre-test scores. The research emphasizes positive impact of gamification and quality instructional design on programming education [25]. Research by Uiphanit et al [29] indicated students in the experimental group scored significantly higher on the learning achievement test (mean = 25.04) compared to the control group (mean = 21.36).

Benefits identified by the review of studies are game-based learning/Gamification enrich learning effectiveness by engaging students actively, which leads to better understanding and retention of concepts in the short term [27], [31]. The interactive nature of games also boosts enthusiasm and engagement, making students more motivated and invested in their studies. Game-based learning/Gamification boosts creativity through open-ended challenges and helps students connect academic concepts to real-world applications [30].

There are notable limitations to game-based learning/Gamification. One challenge is limited evidence for sustained long-term knowledge retention; although students may improve quickly, maintaining those gains over time often requires additional reinforcement [24], [30]. Scalability across institutions also presents difficulties, as resources, training, and consistent implementation may vary among different educational settings [23], [24], [28]. The lack of a universal teaching framework for game-based learning/gamification hinders its widespread use and can cause inconsistency with curriculum goals, making strategic planning and regular evaluation essential during implementation [31].

3.1.4. Puzzle-Based Learning

Puzzle-based learning is an instructional technique that uses puzzles like code jigsaws to develop students' problem-solving and analytical abilities. It engages learners through structured, often game-like tasks that encourage them to apply concepts, recognize patterns, and develop solutions [34]. It promotes active learning and deeper understanding, especially in STEM subjects like computer programming [33].

A study by Hou et al. [32] applied puzzle-based learning by developing CodeTailor, a tool that uses Large Language Models (LLMs) to create personalized Parsons programming puzzles. This study experimentally reported that students found tool considerably more engaging than receiving AI-generated code alone, with 88% of them preferring it for assistance as it fosters deeper learning and improved problem-solving in introductory programming education.

The study by Oyelere et al. [35] reported that using the MobileEdu-Puzzle app significantly enhanced students' understanding of programming concepts, with 90.2% reporting increased motivation. Students also expressed a strong interest in adopting the Puzzle-Based Learning Technique (PbLT) in other courses, underlines its effectiveness in enhancing engagement and learning-based outcomes.

Few limitations identified in the reviewed articles include the need for high-quality pedagogical design and expert guidance to implement puzzle-based learning effectively. Infrastructure and technology constraints, especially in resource-limited settings, can hinder its adoption [33]. While well-suited for teaching entry-level and procedural programming, puzzles may not scale easily to advanced topics like recursion, concurrency, or object-oriented programming [35]. Most studies were short-term, showing immediate learning gains, but the long-term retention and transfer of skills remain uncertain [35]. Additionally, the effectiveness of puzzle-based learning can vary depending on the programming language, subject matter, and educational context [34]. There is also a risk that students may focus more on completing puzzles than understanding underlying concepts or applying them to real-world programming tasks [32].

3.1.5. Mind-Mapping

A Mind mapping is a visual technique that organizes information around a central idea using branches to show relationships, helping to simplify complex topics and enhance understanding. A systematic review done by Kefalis et al. [36], which included 50 papers, examined the role and impact of mind maps in programming education, focusing on algorithmic and procedural learning. The study found that mind mapping enhances learning, problem-solving abilities, improves memory retention and understanding, and boosts students' motivation and engagement. However, more quantitative and longitudinal studies are needed to confirm its long-term benefits and determine its optimal use access different educational contexts. If mind maps become overly complex or crowded with information, they may overwhelm rather than assist learners, particularly those new to the strategy.

Gul et al. [37] investigated the effectiveness of mind maps in enhancing problem-solving skills in programming education through a quasi-experimental post-test-only design involving 160 undergraduate students. The study compared text-based and block-based programming, both with and without the integration of mind maps. Results showed that students who used mind maps significantly outperformed those who did not. Solution accuracy improved from 33% to 80% for text-based programming

and from 40% to 75% for block-based programming. Qualitative survey responses also indicated that mind mapping positively contributed to students' problem-solving abilities.

Liu et al.[38] integrated mind mapping into college programming courses to help students better understand abstract and complex concepts. The qualitative study found that mind mapping improved students' logical thinking, engagement, and ability to organize programming ideas. It also fostered broader learning skills and motivation. However, the findings are based solely on qualitative data, limiting their generalizability and highlighting the need for further quantitative research. The effectiveness of mind mapping relies on students receiving proper instruction in how to create and use them; without guidance, the technique may seem confusing or unintuitive, especially for learners who prefer linear, text-based, or traditional note-taking methods over graphical formats.

Table 4 provides analysis of various problem-solving techniques implemented in programming education, highlighting the benefits and limitations of each technique. Addressing the first research question, the table shows that techniques such as game-based learning/gamification, problem-based learning, puzzle-based learning, mind mapping and flowchart-based learning contribute significantly to enhancing both programming competency and programming interest. These techniques promote a range of competencies including problem-solving, logical thinking, creativity, which are crucial for computer science students.

Table 4. Benefits and Limitations of Problem-solving Techniques

Technique	Methodology	Benefits	Limitations
Problem-Based Learning	Experimental study	Enhanced Conceptual Understanding	Facilitation Dependency
	Qualitative study	High Student Engagement & Motivation	Short team Implementation
	Mixed-method research	Real-world problem understanding	Assessment Complexity
Flowchart-Based Learning	Quasi-experimental design	Visual Representation	Difficulty Handling Advanced Programming Constructs
		Gain in programming self-efficacy	
		Deeper conceptual understanding	Limited Error Diagnosis
		Smooth Transition to Coding	Generalization
Game-based Learning/ Gamification	Experimental study	Improves Learning & creativity	No Long-team retention
		Enhanced motivation & engagement	Scalability across institutions
	Mixed-method study	Bridge the gap between theoretical knowledge and practical application	No universal pedagogical framework, Educator's Expertise
Puzzle-Based Learning	Experimental Design	Enhances Conceptual Understanding & Motivation	Resource and Design Challenges
	Systematic review	Immediate Feedback and Active Learning	Limited Coverage of Advanced Concepts
	Qualitative study	Improved Problem-Solving Skill	Generalizability and Scope
Mind Mapping	Systematic review	Boosts Problem-Solving & Creativity	Limited Quantitative Evidence
	Experimental study	Supports Organizational and Metacognitive Skills	Learner Preferences & Guidance
	Qualitative study	Improves Memory Retention	Potential for Cognitive Overload

3.2 Problem-solving Tools for Computer Programming: Evidence and Implementation (RQ2)

Studies which focus on tools which can enhance problem-solving for computer programming are listed in Table 5.

3.2.1. Algotaurus

Research by Attila et al.[39] created Algotaurus, an educational puzzle-based game that aims to introduce fundamental programming principles through fun, interactive learning. It is intended for novices with little or no coding experience and focuses on algorithmic thinking in an interactive environment. Qualitative finding indicated that the game increased engagement and motivation, helped learners quickly grasp control structures, and made programming more accessible through its visual, interactive interface. It also supported self-paced, trial-and-error learning. Though the findings are promising, the

Table 5. Tools for enhancing Problem-solving

Tool	Reviewed Article
Algotaurus	Attila et al (2021) [39]
Code Adventure	Uiphanit et al (2023) [29]
Code Saga	Tacouri & Nagowah (2021) [40]
Codeseum	Johan et al (2024) [26]
CodeTailor	Hou et al. (2024) [32]
Flow2Code	Ray et al (2019) [41]
LariJava	Mutiawani et al. (2019) [42]
Lightbot 2.0	Law (2018) [43]
Mind mapping tools- Coggle, MindMeister, Ayoa, Xmind, MindNode, GitMind and many more	Bhattacharya & Mohalik (2020) [44], Myre & Abbamonte (2025) [45]
MobileEdu	Oyelere et al. (2019) [35]
NoBug's SnackBar	Vahldick et al. (2020) [46]
PROBSOL	Malik et al. (2019) [47]
RAPTOR	Rahman et al. (2020) [22]
TextCode	Corno et al. (2021) [48]

authors suggest that further empirical research using more rigorous experimental designs is needed to confirm the long-term educational impact.

3.2.2. Code Adventure

Code Adventure is a serious educational game aimed at teaching Java programming to first-year undergraduate students. The game is based on the action genre and integrates key elements such as educational goals, fun, challenges, rewards, and feedback. It revolves around a hero who collects points for answering questions at every level, hence creating an immersive experience that makes students engage with the content throughout the game [29].

The experimental results showed that educational game can be a powerful tool for teaching programming, leading to higher engagement, satisfaction, and improved learning outcomes compared to traditional lecture-based methods. The authors call for future research with larger and more diverse samples and long-term assessments to validate and extend these positive findings.

3.2.3. Code Saga

Code Saga, a mobile serious game designed to teach introductory programming concepts in an interactive and engaging way. It includes a number of mini-games featuring engaging 2D adventure tile-based gameplay such as Treasure Hunt and JavaWars. SortingAlgo provides visualizations of sorting algorithms, while CodeBlock offers a puzzle-like coding experience. The game also incorporates performance measurement activities like quizzes and the Wheel Spinner to gauge players' comprehension. Acceptance testing with university-level IT students revealed high satisfaction with the game's effectiveness. Through gamification of fundamental concepts using challenges and narrative on a mobile platform, the tool reduces the barrier to entry for learners and promotes continued engagement. More empirical research involving wider participant groups and long-term testing, nonetheless, is required to determine its effectiveness and scalability across varied educational environments [40].

3.2.4. Codeseum

Codeseum is a virtual reality game developed in Unity. It is an educational game that is used to teach introductory programming concepts. Codeseum drops players into a virtual gallery that merges famous artworks with programming concepts and encourages players to solve puzzles by applying their coding knowledge in order to reach the ultimate destination [26].

Codeseum demonstrates that VR-based educational games can provide higher levels of engagement, enjoyment, and focused attention when teaching programming fundamentals, compared to traditional desktop games. Practical limitations, small-scale evaluation, and the requirement for longer-term as well as larger studies, however, indicates that additional research is required to validate its general educational effects and to address scalability for varied group of learners.

3.2.5. CodeTailor

CodeTailor is an AI-powered learning tool that helps novice programmers by generating personalized Parsons puzzles using large language models (LLMs). Instead of providing full code solutions, it encourages active problem-solving by analysing student errors and creating tailored puzzles. If students struggle, it offers hints and gradually increasing support. A user study found CodeTailor more engaging and more effective for recall and concept application than traditional AI-generated solutions, making it a scalable and interactive approach to programming education [32].

3.2.6. Flow2Code

Flow2Code is tool that transform hand-drawn flowcharts into executable code. By leveraging sketch recognition and mobile devices, the tool makes algorithmic thinking and coding more accessible, especially to novice programmers. While challenges remain in handling complex inputs and ensuring broad language support, the approach represents an innovative step toward integrating visual problem-solving tools with hands-on coding practice for enhanced learning outcomes [41].

3.2.7. LariJava

LariJava, a web-based MVC application prototype designed to support learning of fundamental programming concepts through the problem-based learning method. The platform provides students with examples and exercises related to introductory Java programming, allowing them to practice and check their answers against pre-defined solutions and truth table-based test cases [42].

The study found that using LariJava helped students effectively acquire basic programming skills through problem-solving techniques. A usability assessment of the prototype, conducted using the System Usability Scale (SUS) questionnaire, resulted in a final SUS score of 72, indicating good usability and overall acceptability.

3.2.8. Light Bot 2.0

Light Bot 2.0 is puzzle-based game tasks players with guiding a robot to illuminate all blue tiles within a specified area using command sets that represent core programming concepts like sequential execution, functions, recursion, and conditional flows. The findings showed that the students were indeed effectively learning the principles of programming while systematically playing through the game. A significant majority of the test subjects recognized that classroom gaming methodologies are enjoyment and make complicated concepts more easily understood [43].

3.2.9. Mind mapping tools

The reviewed studies have identified more than 25 mind mapping tools, including Coggle, MindMeister, Ayoa, XMind, MindNode, and others. Some of these tools, such as GitMind, are available for free online, while others operate on a subscription basis. XMind is well-suited for individual use, whereas MindMeister and Coggle offer real-time collaboration features. Ayoa is a digital mind mapping platform that enables both individuals and teams to construct unique mind maps with ease.

While certain tools focus exclusively on creating mind maps, others support additional diagram types, such as flowcharts, concept maps, tree maps, and fishbone diagrams. The study by Bhattacharya & Mohalik [44] outlines the steps for creating digital mind maps using these software applications, while Myre & Abbamonte [45] provide an overview of the advantages and disadvantages of several popular tools. These findings can assist educators in selecting the most suitable mind mapping tool for their specific needs.

3.2.10. MobileEdu

MobileEdu is a mobile learning app designed to support computer programming education. In a study with 142 third-year computer science students, those using MobileEdu outperformed peers in programming assessments and reported higher engagement, motivation, and positive attitudes. The app also improved access to learning materials. However, its long-term impact and scalability across institutions remain unassessed [35].

3.2.11. NoBug's SnackBar

Vahldick et al. [46] developed an innovative serious game titled NoBug's SnackBar, aimed at fostering problem-solving skills essential for introductory programming courses. The block-based game was implemented and evaluated across four academic semesters, allowing for iterative refinement based on user feedback and instructional outcomes. These experimental cycles resulted in the identification of fourteen key insights and recommendations that inform the effective design and implementation of game-based learning approaches to support novice programming learners.

3.2.12. PROBSOL

PROBSOL, is a pseudo-code-based application designed to enhance problem-solving skills among novice programmers in an introductory programming course. Available as both a mobile (M-PROBSOL) and web-based (E-PROBSOL) tool, the application presents randomized pseudo-code sequences that students must arrange correctly to solve programming problems, with feedback provided for incorrect attempts. By abstracting syntax, PROBSOL enables learners to focus on logical reasoning and problem-solving strategies.

A study by Malik et al[47], employed student surveys, instructor interviews, and analysis of final exam results over two semesters. Findings indicated a positive reception from both students and instructors, improved engagement, cognitive development, and enhanced logic and problem-solving abilities. Comparative grade analysis also showed increased academic performance and reduced attrition following the implementation of PROBSOL.

3.2.13. RAPTOR

RAPTOR is a flowchart-based programming environment, aimed at aiding students in understanding algorithms visually and navigating syntactic errors. It stands for a Rapid Algorithmic Prototyping Tool for Ordered Reasoning. Programmers can visually follow flowcharts while executing programs to trace the code. It offers the capability to convert flowcharts, to a limited extent, into various programming languages like Ada, C#, C++, Java, and Standalone. Rahman et al.[22] employed RAPTOR to teach Python as part of an introductory CS course. The student's survey was conducted and results revealed favourable outcomes, with students reporting enhanced problem-solving and greater motivation toward programming.

3.2.14. TextCode

Corno et al.[48] introduced TextCode, an Integrated Development Environment (IDE) designed to foster problem-solving ability among university students learning programming. The tool guides developers through the process of comprehending the problem being solved, breaking it down into easily manageable sub-problems, identifying different ways to implement each sub-problem, and then integrating such implementations into a consistent solution. By prioritizing problem-solving steps in its design, starting with understanding the problem statement, TextCode aims to enhance metacognitive awareness among novices and clarify misconceptions about problem-solving as non-essential activity.

Some of the limitations of the tool described in this review relate to the fact that most of the tools were evaluated in single-institution studies with small sample sizes, which limits the generalizability of the findings. In addition, much of the research focused on short-term learning outcomes, providing little evidence of long-term retention or transfer of knowledge. Widespread use of these tools might also face challenges related to technical requirements, the need for instructor training, and limited alignment with existing curricula. Finally, these tools are largely designed for novices, with limited significant evidence supporting their effectiveness for advanced learners.

Educators should be aware of these constraints while choosing and integrating programming tools. Further research, modification of tools to suit varied teaching environments, and continual evaluation are necessary actions to promote continued student learning and interest.

3.3 Key Findings

The reviewed literature mainly emphasizes either problem-solving techniques or instructional tools, with relatively few studies addressing both in an integrated manner. To bridge this gap, the present review seeks to establish the relationship between reviewed techniques and tools, as illustrated in Table 6. This association is meant to guide instructors in the effective adoption and proper implementation of appropriate approaches and resources in enhancing the programming ability of students. For example, to implement a puzzle-based learning approach, educators may employ tools such as Algotaurus, Code Sage, CodeTailor, Lightbot 2.0, or MobileEdu, which are specifically designed around the principles of puzzle-based learning. Some tools are versatile and can be applied across multiple instructional techniques. For example, the Code Sage tool is suitable for both game-based learning or gamification, as well as puzzle-based learning approaches.

Figure 3 shows the diagram of tools related to each technique. The tools in same colour as technique are unique to each technique whereas the pattern filled circles (tools) like Algotaurus, Code Saga, and Lightbot 2.0, between game-based learning and puzzle-based learning shows them applicably to both techniques.

This integrative approach specifies effective problem-solving approaches in programming education and illustrates their application with the help of educational technologies. By mapping techniques such as puzzle-based learning, game-based learning, and mind mapping to tools such as CodeTailor, Codeseum, and MindMeister, educators can move beyond routine learning to enable deeper conceptual understanding in interactive cum interesting manners. These strategies promote student's

Table 6. Tools for problem-solving techniques

Techniques	Tools
Problem-Based Learning	LariJava, PROBSOL, TextCode
Flowchart-Based Learning	Flow2Code, RAPTOR
Game-Based Learning/Gamification	Algotaurus, Code Adventure, Code Saga, Codeseum, NoBug's SnackBar, Lightbot 2.0
Puzzle-Based learning	Algotaurus, Code Saga, CodeTailor, Lightbot 2.0, MobileEdu,
Mind Mapping	Coggle, MindMeister, Ayoa, Xmind, MindNode, GitMind and many more

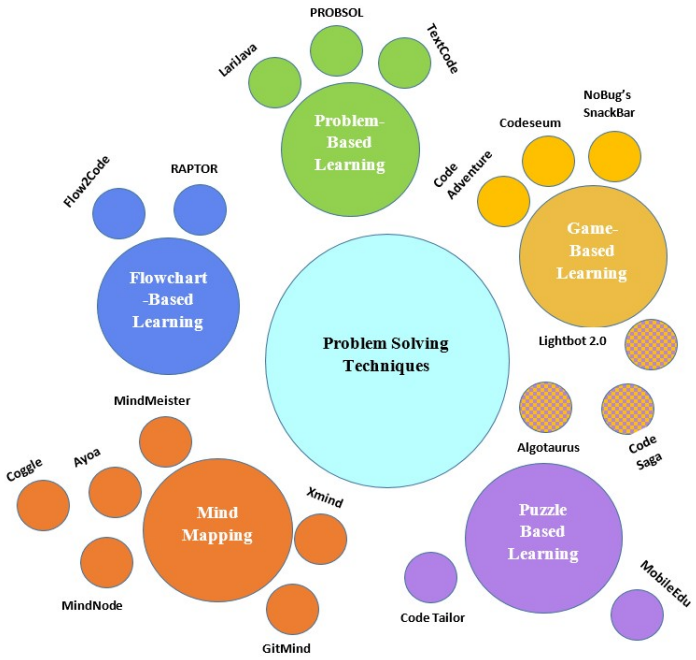

Fig 3. Techniques and Tool Mapping

visualization of problems, investigation of alternate solution paths, and strengthen logical reasoning. Utilizing interactive tools offers enjoyable, hands-on learning experiences that facilitate active construction of knowledge. Moreover, these tools facilitate self-paced and trial-and-error learning, which reinforces student confidence. This model also reinforces learner independence and enables the acquisition of self-directed learning skills. It also strengthens the other higher order thinking skills of the students, such as creative thinking, logical thinking, and critical thinking. Finally, it facilitates a student-centric context that advances engagement and long-term programming achievements.

The review emphasizes that in order to achieve effective incorporation of problem-solving methods and tools in programming education, the upskilling of educators is necessary. Focused training in emerging technologies and pedagogies equips teachers with the confidence to embrace innovative teaching methods, which is in sync with the study by Mills et al.[49]. Continuing professional development connects theory and practice and enhances ongoing improvement and flexibility. This ultimately improves the quality of instruction and better equips instructors to assist a diverse group of learners in acquiring problem-solving and programming competencies.

4 Conclusion

Despite ongoing advancements in educational technologies and methodologies, teaching and learning computer programming continue to pose significant challenges, particularly for novice learners. Among the competencies essential for success, problem-solving skills consistently emerge as a core predictor of programming performance. This review synthesizes insights from 37 peer-reviewed studies and makes an original contribution by mapping a comprehensive range of problem-solving approaches to corresponding digital tools used in programming instruction.

By categorizing and analysing these techniques—such as problem-based learning, puzzle-based learning, flowchart-based learning, game-based learning, and mind mapping—and aligning them with tools like PROBSOL, CodeTailor, Lightbot 2.0, MindMeister, and RAPTOR, the review provides educators with a practical framework. This integrative strategy highlights which approaches work best and how they can be implemented in technology-enhanced classrooms to promote deeper conceptual understanding, engagement, and independent learning. However, the review also identifies notable limitations in the existing body of research, including small sample sizes, single-institution studies, and short intervention durations. Consequently, most findings emphasize short-term learning gains rather than long-term retention or applicability across diverse learner groups.

This review makes a distinctive contribution by presenting an integrated dual-layer framework that systematically links problem-solving techniques with their corresponding educational tools. The resulting synthesis offers a cohesive and actionable model for programming educators, setting it apart from earlier fragmented reviews. The framework also provides practical direction for selecting instructional strategies: game-based and puzzle-based learning are effective for smaller or introductory classes to build motivation and foundational problem-solving skills, while problem-based learning and mind mapping are more suitable for larger or advanced cohorts to promote independent thinking and conceptual integration. Educators can thus align techniques with course goals, class size, and student proficiency to achieve optimal outcomes.

To advance programming education further, future research should prioritize longitudinal studies, broader participant diversity, and professional development opportunities that help educators integrate appropriate tools and active learning strategies. Additionally, the ethical and pedagogical implications of emerging technologies such as artificial intelligence and adaptive learning systems warrant exploration. Addressing these areas can make programming education more effective, inclusive, and aligned with the evolving demands of the modern digital world.

5 Declarations

All authors declare that they have no conflicts of interest.

References

1) Schnedlitz F, Kerschbaumer D, Steinmaurer A, Lowe D, Gütl C. To Complete or Not to Complete: Prediction and Classification of Dropouts in a Cs1 Course. 2025.
2) Cheah CS. Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*. 2020 May;12(2):ep272. https://doi.org/10.30935/cedtech/8247.
3) Kurniawan O, Jégourel C, Lee NT, Mari MD, Poskitt CM. Steps before syntax: Helping novice programmers solve problems using the PCDIT framework. *arXiv preprint arXiv:210908896*. 2021 Sep. https://arxiv.org/pdf/2109.08896.
4) Medeiros RP, Ramalho GL, Falcão TP. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*. 2018 Aug;62(2):77–90. https://doi.org/10.1109/TE.2018.2864133.
5) Santos JS, Andrade WL, Brunet J, Melo MR. A systematic literature review on predictive cognitive skills in novice programming. 2022.
6) Eteng I, Akpotuzor S, Akinola SO, Agbonlahor I. A review on effective approach to teaching computer programming to undergraduates in developing countries. *Scientific African*. 2022 Jul;16:e01240. https://doi.org/10.1016/j.sciaf.2022.e01240.
7) Medeiros RP, Ramalho GL, Falcão TP. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*. 2018 Aug;62(2):77–90. https://doi.org/10.1109/TE.2018.2864133.
8) Santos JS, Andrade WL, Brunet J, Melo MR. A systematic literature review on predictive cognitive skills in novice programming. 2022.
9) Santos SC, Tedesco PA, Borba M, Brito M. Innovative approaches in teaching programming: A systematic literature review;vol. 1. 2020. https://doi.org/10.5220/0009190502050214.
10) Zhu M, Xu L, Ericson B. A systematic review of research on large language models for computer programming education. 2025. https://arxiv.org/pdf/2506.21818.
11) Kitchenham B, Brereton P. A systematic review of systematic review process research in software engineering. *Information and Software Technology*. 2013 Dec;55(12):2049–2075. https://doi.org/10.1016/j.infsof.2013.07.010.
12) Tareq ZA, Yusof RJ. Modelling a problem-solving approach through computational thinking for teaching programming. *IEEE Transactions on Education*. 2024 Feb;67(2):282–291. https://doi.org/10.1109/TE.2024.3354425.
13) Aires JP, Aires SB, Pereira MJ, Alves LM. Using the methodology problem-based learning to teaching programming to freshman students. *International Journal of Information and Education Technology*. 2023;13(3):1825. https://doi.org/10.18178/ijiet.2023.13.3.1825.
14) Hegade P, Aryan OP, Netekal M, Shettar A. A Mini-Ethnography Study of Teaching C Programming to First Year Non-Circuit Branch Students. *Journal of Engineering Education Transformations*. 2023 Jan;36(Special Issue 2):504–511. https://doi.org/10.16920/jeet/2023/v36is2/23077.
15) Hawa DM, Ghoniem E, Saad AM. Integrating problem-based learning into blended learning to enhance students' programming skills. *J Positive School Psychol*. 2022;6(8):4479–4497. https://doi.org/10.53555/ks.v10i1.3554.
16) Aires JP, Aires SBK, Pereira MJV, Alves LM. Active methodologies in incoming programming classes (short paper);vol. 91. 2021.
17) Skalka J, Drlík M. Educational model for improving programming skills based on conceptual microlearning framework. 2020.
18) Souza SM, Bittencourt RA. Motivation and engagement with PBL in an introductory programming course. 2019.
19) Janpla S, Piriyasurawong P. The development of problem-based learning and concept mapping using a block-based programming model to enhance the programming competency of undergraduate students in computer science. *TEM J*. 2018 Nov;7(4):708. https://doi.org/10.18421/TEM74-02.

20) Zhou R, Xie C, He X, Li Y, Fan Q, Yu Y, et al. Effect of Different Flow Design Approaches on Undergraduates' Computational Thinking During Pair Programming. *Journal of Educational Computing Research*. 2024 Dec;62(7):1645–1675. https://doi.org/10.1177/07356331241268474.
21) Zhang JH, Meng B, Zou LC, Zhu Y, Hwang GJ. Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interactive Learning Environments*. 2023;31(6):3792–3809. https://doi.org/10.1080/10494820.2021.1943687.
22) Rahman MM, Sharker MH, Paudel R. An effective approach to teach an introductory computer science course with computational thinking and flow-chart based visual programming. 2020.
23) Castillo-Salvatierra L, Cárdenas-Cobo J, Fuente-Burdiles CDL, Vidal-Silva C. Programming competencies in university students through game development. *Frontiers in Education*. 2025;10:1585602. https://doi.org/10.3389/feduc.2025.1585602.
24) Wong YS, Yatim MHM, Rahman MHA, Zain NZM, Rosli AN, Lee HY, et al. Sokoban: A Game-Based Learning Game for Enhancing Programming Skill Among Undergraduate Students in Malaysia. *J Adv Res Des*. 2025;131(1):104–116. https://doi.org/10.37934/ard.131.1.104116.
25) Mellado R, Cubillos C. Gamification improves learning: Experience in a training activity of computer programming in higher education. *J Computer Assisted Learning*. 2024;40(4):1959–1973. https://doi.org/10.1111/jcal.13000.
26) Ekman J, Solsona J, Quintero L. Codeseum: Learning introductory programming concepts through virtual reality puzzles. 2024. https://doi.org/10.1145/3639701.3656306.
27) Tun ST, Hutasingh A, Aphiratsakun N. Developing steps for learning programming through gamification. 2023.
28) Videnovik M, Vold T, Kiønig L, Bogdanova AM, Trajkovik V. Game-based learning in computer science education: a scoping literature review. *Int J STEM Educ*. 2023;10:54. https://doi.org/10.1186/s40594-023-00447-2.
29) Uiphanit T, Janpla S, Chutosri T, Liangyoo P, Wattanaprapa N, Kingsuwankul P, et al. Code Adventure: An Educational Game for Learning JAVA Programming. *International Journal Interactive Mobile Technologies (iJIM)*. 2023;17(22):26–37. https://doi.org/10.3991/iJIM.v17i22.42307.
30) Cuervo-Cely KD, Restrepo-Calle F, Ramírez-Echeverry JJ. Effect of gamification on the motivation of computer programming students. *Journal of Information Technology Education: Research*. 2022;21:1–23. https://doi.org/10.28945/4917.
31) Khaleel FL, Azhari NS, Wook TS. An empirical study on gamification for learning programming language website. *Jurnal Teknologi- Sciences & Engineering*. 2019 Feb;81(2). https://doi.org/10.11113/jt.v81.11133.
32) Hou X, Wu Z, Wang X, Ericson BJ. CodeTailor: Llm-powered personalized parsons puzzles for engaging support while learning programming. 2024. https://doi.org/10.1145/3657604.3662032.
33) Ericson BJ, Pearce JL, Rodger SH, Csizmadia A, Garcia R, Gutierrez FJ, et al. Conducting multi-institutional studies of Parsons problems. 2023. https://doi.org/10.1145/3587103.3594211.
34) Ericson BJ, Denny P, Prather J, Duran R, Hellas A, Leinonen J, et al. Parsons problems and beyond: Systematic literature review and empirical study designs. 2022. https://doi.org/10.1145/3571785.3574127.
35) Oyelere SS, Agbo FJ, Sanusi IT, Yunusa AA, Sunday K. Impact of puzzle-based learning technique for programming education in Nigeria context;vol. 2161. 2019.
36) Kefalis C, Skordoulis C, Drigas A. A Systematic Review of Mind Maps, STEM Education, Algorithmic and Procedural Learning. *Computers*. 2025 May;14(6):204. https://doi.org/10.3390/computers14060204.
37) Gul S, Asif M, Nawaz Z, Aziz MH, Khurram S, Saleem MQ, et al. Sustainable Learning of Computer Programming Languages Using Mind Mapping. *Intelligent Automation & Soft Computing*. 2023 May;36(2). Available from: https://www.techscience.com/iasc/v36n2/51134.
38) Liu Y, Tong Y, Yang Y. The application of mind mapping into college computer programming teaching. *Procedia Computer Science*. 2018 Jan;129:66–70. https://doi.org/10.1016/j.procs.2018.03.047.
39) Krajcsi A, Csapodi C, Stettner E. Algotaurus: an educational computer programming game for beginners. *Interactive Learning Environments*. 2021 May;29(4):634–647. https://doi.org/10.1080/10494820.2019.1593862.
40) Tacouri H, Nagowah L. Code Saga–A mobile serious game for learning programming. 2021.
41) Mutiawani V, Chaidir M, Afidh RP, Irvanizam I, Amalia MM, Juwita J. Reconstruction of LariJava Learning Programming Website Using MVC Concept. 2019.
42) Mutiawani V, Chaidir M, Afidh RP, Irvanizam I, Amalia MM, Juwita J. Reconstruction of LariJava Learning Programming Website Using MVC Concept. 2019.
43) Law R. Introducing novice programmers to functions and recursion using computer games. 2018.
44) Bhattacharya D, Mohalik R. Digital mind mapping software: A new horizon in the modern teaching-learning strategy. *Journal of Advances in Education and Philosophy*. 2020 Oct;4(10):400–406. https://doi.org/10.36348/jaep.2020.v04i10.001.
45) Myre M, Abbamonte K. The best mind mapping software in 2025. *Zapier Blog*. 2025 Jul. Available from: https://zapier.com/blog/best-mind-mapping-software/.
46) Vahldick A, Farah PR, Marcelino MJ, Mendes AJ. A blocks-based serious game to support introductory computer programming in undergraduate education. *Computers in Human Behaviour Reports*. 2020 Aug;2:100037. https://doi.org/10.1016/j.chbr.2020.100037.
47) Malik SI, Mathew R, Al-Nuaimi R, Al-Sideiri A, Coldwell-Neilson J. Learning problem solving skills: Comparison of E-learning and M-learning in an introductory programming course. *Education and Information Technologies*. 2019 Sep;24(5):2779–2796. https://doi.org/10.1007/s10639-019-09896-1.
48) Corno F, Russis LD, Sáenz JP. TextCode: a tool to support problem solving among novice programmers. 2021.
49) Mills KA, Cope J, Scholes L, Rowe L. Coding and computational thinking across the curriculum: A review of educational outcomes. *Review of Educational Research*. 2025 Jun;95(3):581–618. https://doi.org/10.3102/00346543241241327.