ISRG Journal of Multidisciplinary Studies (ISRGJMS)

ISRG PUBLISHERS

Abbreviated Key Title: isrg j. multidiscip. Stud.

ISSN: 2584-0452 (Online)

Journal homepage: https://isrgpublishers.com/isrgjms/

Volume – IV, Issue – IX (September) 2026

Frequency: Monthly

The Effect of Algorithmic Coding Integration on Junior High School Students' Mathematical Problem-Solving Skills

Herianto Parrangan[1*], Inelsi Palengka[2], Sonny Yalti Duma'[3]

[1, 2, 3] Master of Mathematics Education Program, Indonesian Christian University Toraja Jl.Jenderal Sudirman Number 9 Makale 91811, Tana Toraja

| **Received:** 04.09.2026 | **Accepted:** 08.09.2026 | **Published:** 12.09.2026

***Corresponding author:** Herianto Parrangan

Abstract

In the era of the Fourth Industrial Revolution, which demands strong digital literacy, integrating algorithmic coding into mathematics instruction has emerged as an innovative strategy to address junior high school students' low mathematical problem-solving skills. This issue is particularly evident in algebra and number pattern topics among eighth-grade students at SMPN 1 Awan Rante Karua, where students experience difficulties in applying algorithmic logic systematically. This quasi-experimental study employed a pretest–posttest nonequivalent control group design involving 60 eighth-grade students divided into two groups. The experimental group received mathematics instruction integrated with algorithmic coding through Scratch, including Caesar cipher activities and linear equation loops, over eight instructional sessions, while the control group received conventional instruction. The findings revealed a significant improvement in the experimental group, with an N-Gain score of 0.72 (high category), compared with 0.45 (moderate category) in the control group. Independent-samples t-test analysis showed a statistically significant difference between the two groups (Sig. = 0.001 < 0.05). These findings demonstrate that integrating algorithmic coding into mathematics learning effectively enhances junior high school students' mathematical problem-solving skills.

Keywords: *Algorithmic coding; mathematical problem-solving; Scratch; junior high school.*

Introduction

Mathematics education in Indonesia continues to face challenges in equipping junior high school students with twenty-first-century competencies, particularly mathematical problem-solving skills. The results of the 2022 Programme for International Student Assessment (PISA) indicate that Indonesian students' mathematical literacy remains below the OECD average, especially in solving problems involving algebra and number patterns. These findings suggest that mathematics instruction has not yet fully encouraged students to think critically, logically, and systematically. Classroom learning is still predominantly characterized by conventional approaches emphasizing procedural memorization, making it difficult for students to understand mathematical concepts and apply them to more complex contextual problems.

DOI: 10.5281/zenodo.22721535

A similar phenomenon was identified in the local context of SMPN 1 Awan Rante Karua, East Flores Regency. Preliminary classroom observations indicated that only approximately 45% of eighth-grade students were able to solve non-routine mathematical problems independently. Most students experienced difficulties in organizing systematic solution strategies, particularly during the planning stage and in identifying patterns within algebra and number pattern topics. These findings indicate that students' algorithmic thinking and problem decomposition skills remain relatively low, although both competencies are fundamental prerequisites for successful mathematical problem-solving. Consequently, there is an urgent need for innovative instructional approaches capable of helping students develop systematic and meaningful thinking structures.

In line with the ongoing digital transformation in education, the Merdeka Curriculum emphasizes the importance of strengthening digital literacy and computational thinking throughout the learning process. Coding has increasingly been positioned as a strategic medium for developing logical, analytical, and systematic thinking skills that are highly relevant to mathematics education. Algorithmic coding functions not merely as a technological skill but also as a cognitive tool that enables students to understand problem-solving processes by constructing logical procedures in a sequential manner. Therefore, integrating algorithmic coding into mathematics instruction represents a promising instructional innovation for improving learning quality, particularly at the junior high school level.

Previous studies have demonstrated the effectiveness of integrating coding into mathematics education. Research conducted by Sari et al. (2022), published in *Mosharafa: Jurnal Pendidikan Matematika*, reported that Scratch-based algorithmic learning significantly improved junior high school students' critical thinking and mathematical problem-solving abilities. The study further indicated that students developed a better understanding of mathematical relationships through visual and algorithmic representations. Likewise, Pratama and Lestari (2023), in *Math Didactic*, found that integrating Scratch into mathematics learning significantly enhanced students' conceptual understanding and computational thinking, particularly in geometry, as reflected by high N-Gain scores. Similar findings were reported by Wahyuni (2021), who concluded that programming-based problem-solving instruction positively influenced eighth-grade students' learning independence and mathematical problem-solving skills.

Nevertheless, most previous studies were conducted in urban schools with relatively adequate technological infrastructure. Furthermore, research specifically examining the integration of algorithmic coding within algebra and number pattern instruction, while explicitly linking coding activities to the stages of mathematical problem-solving, remains limited. Existing studies generally emphasize improvements in computational thinking without specifically investigating how algorithmic coding can strengthen each stage of mathematical problem-solving based on Polya's problem-solving model. This gap indicates the need for further investigation, particularly in rural educational settings where technological resources are more limited.

The urgency of this study has become even more evident following the Indonesian government's policy introducing coding as an elective subject within the Merdeka Curriculum beginning in the 2025/2026 academic year. Rural schools, such as SMPN 1 Awan Rante Karua, risk falling behind if they do not promptly adopt contextual and accessible digital literacy innovations. Scratch is considered an appropriate learning platform because it provides a visual programming environment that is easy to use, widely accessible, and does not require sophisticated technological infrastructure. These characteristics make it particularly suitable for schools with limited educational resources.

Based on these considerations, this study aims to empirically examine the effect of integrating algorithmic coding into mathematics instruction on junior high school students' mathematical problem-solving skills in algebra and number pattern topics. The novelty of this research lies in the implementation of Scratch-based algorithmic coding that is explicitly aligned with the stages of mathematical problem-solving and applied within the context of a rural junior high school. It is expected that this study will contribute theoretically to the development of mathematics instruction grounded in digital literacy and computational thinking, while also providing practical guidance for teachers in implementing the Merdeka Curriculum more effectively and inclusively.

Methods

This study employed a quasi-experimental approach using a pretest–posttest nonequivalent control group design**,** which is appropriate for examining the effects of an educational intervention in naturally existing classroom settings. This design was selected because it enables the measurement of students' learning outcomes before and after the intervention without requiring complete random assignment, which is often impractical in public school contexts. The independent variable was the integration of algorithmic coding through Scratch, while the dependent variable was students' mathematical problem-solving ability. The study was conducted over eight weeks during the second semester of the 2025/2026 academic year**.**

The population consisted of 120 eighth-grade students enrolled in four parallel classes at SMPN 1 Awan Rante Karua, East Flores Regency, Indonesia**.** The sample was selected using purposive sampling, in which two heterogeneous classes with comparable mathematics achievement in the previous semester were chosen. Class VIII-A served as the experimental group (n = 30), whereas Class VIII-B served as the control group (n = 30), resulting in a total sample of 60 students. Students who attended at least 80% of the instructional sessions and completed all research activities were included in the study. This sampling technique was intended to ensure comparable baseline characteristics between the two groups.

The primary research instrument was a mathematical problem-solving test consisting of 20 essay items, including 10 algebra problems and 10 number pattern problems**.** The instrument was developed based on Polya's four stages of problem solving, namely understanding the problem, devising a plan, carrying out the plan, and reviewing the solution. Content validity was established through expert judgment, yielding an Aiken's V coefficient greater than 0.80**,** while reliability testing produced a Cronbach's Alpha coefficient of 0.87, indicating high internal consistency. Students' responses were scored on a 0–100 scale. In addition, classroom observations were conducted using a validated observation checklist with an inter-rater reliability coefficient of 0.89, and students' perceptions of Scratch-based learning were collected through a five-point Likert-scale questionnaire. Prior to implementation, all instruments were pilot-tested with 20 seventh-grade students to evaluate clarity and reliability.

DOI: 10.5281/zenodo.22721535

The research procedure consisted of four main stages. During the preparation stage, all research instruments were validated, teachers received instructional briefings, and both groups completed the pretest simultaneously during the first week. The experimental group then participated in eight instructional sessions (2 × 40 minutes per week) integrating algorithmic coding into mathematics learning through Scratch. Sessions 1–2 introduced students to Scratch programming and the Caesar cipher concept; Sessions 3–5 focused on number pattern exploration using loop structures; and Sessions 6–8 emphasized linear functions through integrated programming projects. In contrast, the control group received conventional mathematics instruction consisting of teacher explanations, classroom discussions, and problem-solving exercises. At the conclusion of the intervention, both groups completed the posttest, followed by data analysis and triangulation with classroom observation findings.

The algorithmic coding intervention was designed according to the principles of Elaboration Theory, consisting of three instructional phases: presentation of mathematical concepts**,** algorithmic elaboration using Scratch programming, and application to contextual mathematical problems**.** For example, students programmed a Caesar cipher to encrypt the coefficients of a linear equation (e.g., $2x + 3y = 12$) and subsequently decoded the encrypted expression to verify the mathematical solution. Each instructional session included digital worksheets and formative assessments. Because the school had only five computers, students worked collaboratively in rotating groups under the guidance of the classroom teacher. This implementation strategy ensured equitable access to technology despite the limited educational resources available in the rural school.

Quantitative data were analyzed using IBM SPSS Statistics version 26**.** Prior to hypothesis testing, the data were examined for normality using the Kolmogorov–Smirnov test and for homogeneity of variance using Levene's test. Differences in learning gains between the experimental and control groups were analyzed using an independent-samples ***t***-test with a significance level of $\alpha = 0.05$. The magnitude of the intervention effect was determined using Cohen's ***d*,** while learning improvement was measured using the normalized gain (N-Gain) based on Hake's classification (*low* < 0.30, *moderate* = 0.30–0.70, and *high* > 0.70). Qualitative data obtained from classroom observations and student questionnaires were analyzed thematically to support and enrich the quantitative findings. All statistical assumptions were verified before conducting the inferential analyses to ensure the validity of the research results.

Results

Sample Characteristics

The study involved 60 eighth-grade students from UPT SMPN 1 Rembon, who were equally distributed into the experimental group ($n = 30$) and the control group ($n = 30$). The average age of the participants was 13.8 years, with 53% male and 47% female students. The mean mathematics achievement from the previous semester indicated that both groups had comparable baseline academic abilities (experimental: 72.4 ± 8.2; control: 71.9 ± 7.9; $t = 0.23$, $p = 0.82$). Student attendance throughout the intervention reached 92% in both groups, satisfying the predetermined inclusion criteria.

Table 1. Sample Characteristics

Characteristics	Experimental (n = 30)	Control (n = 30)	*p*-value
Age (years)	13.8 ± 0.9	13.9 ± 1.0	0.76
Previous Mathematics Score	72.4 ± 8.2	71.9 ± 7.9	0.82
Male (%)	16 (53%)	16 (53%)	1.00

The statistical analysis confirms that the experimental and control groups were equivalent before the intervention, ensuring that any subsequent differences could reasonably be attributed to the instructional treatment.

Pretest Results of Mathematical Problem-Solving Skills

The pretest analysis indicated no statistically significant difference between the experimental and control groups in their initial mathematical problem-solving ability ($M = 58.3$ and $M = 57.1$, respectively; $t = 0.45$, $p = 0.65$). Data distribution satisfied the assumptions of normality (Kolmogorov–Smirnov, $p > 0.05$) and homogeneity of variance (Levene's test, $p = 0.78$).

Among Polya's four problem-solving stages, students obtained the lowest scores in planning a solution (42%) and reviewing the solution (38%). These findings indicate that students experienced considerable difficulty in constructing systematic solution strategies and evaluating the correctness of their answers, particularly in algebra and number pattern topics.

Table 2. Pretest Results of Mathematical Problem-Solving Skills

Polya's Indicator	Experimental (Mean ± SD)	Control (Mean ± SD)	*t*-value	*p*-value
Understanding the Problem	65.2 ± 12.4	64.8 ± 11.9	0.12	0.90
Devising a Plan	42.1 ± 15.6	41.9 ± 14.8	0.06	0.95
Carrying Out the Plan	68.4 ± 13.2	67.1 ± 12.9	0.41	0.68
Reviewing the Solution	38.7 ± 16.1	37.5 ± 15.4	0.32	0.75
Total Score	**58.3 ± 10.2**	**57.1 ± 9.8**	**0.45**	**0.65**

These findings indicate that both groups started the study with similar levels of mathematical problem-solving ability, providing a reliable baseline for evaluating the effectiveness of the intervention.

Posttest Results of Mathematical Problem-Solving Skills

Following the intervention, the experimental group demonstrated substantially higher mathematical problem-solving performance than the control group. The mean posttest score of the experimental group reached 82.6 ± 9.4, whereas the control group obtained 70.2 ± 10.1, yielding a statistically significant difference ($t = 5.23$, $p = 0.001$).

DOI: 10.5281/zenodo.22721535

The greatest improvements were observed in the planning and reviewing stages of Polya's framework. Students in the experimental group improved by 32% in planning and 41% in reviewing, compared with 15% and 18%, respectively, in the control group. These improvements suggest that the algorithmic structure provided by Scratch effectively supported students in organizing and evaluating mathematical solution procedures.

Table 3. Posttest Results of Mathematical Problem-Solving Skills

Polya's Indicator	Experimental (Mean ± SD)	Control (Mean ± SD)	*t*-value	*p*-value
Understanding the Problem	84.1 ± 8.7	76.3 ± 9.2	3.42	0.001
Devising a Plan	74.2 ± 11.3	56.9 ± 12.4	5.67	0.000
Carrying Out the Plan	87.6 ± 7.9	79.4 ± 8.5	4.12	0.000
Reviewing the Solution	79.5 ± 10.2	55.7 ± 11.6	7.23	0.000
Total Score	82.6 ± 9.4	70.2 ± 10.1	5.23	0.001

Overall, these findings indicate that integrating algorithmic coding into mathematics instruction significantly improved students' mathematical problem-solving performance across all stages of Polya's problem-solving model.

N-Gain Analysis and Effect Size

The effectiveness of the instructional intervention was further examined using normalized gain (N-Gain) and Cohen's *d* effect size. The experimental group achieved an N-Gain score of 0.72, classified as high, whereas the control group obtained an N-Gain score of 0.45, which falls within the moderate category. Furthermore, the intervention produced a large practical effect, as indicated by a Cohen's *d* value of 1.28.

When analyzed by topic, students showed greater improvement in number patterns (N-Gain = 0.78) than in algebra (N-Gain = 0.66**)**. This finding suggests that Scratch's visual loop programming features are particularly effective in representing sequential mathematical patterns.

Table 4. N-Gain Analysis and Effect Size

Group	Pretest	Posttest	N-Gain	Category	Cohen's *d*
Experimental	58.3	82.6	0.72	High	1.28
Control	57.1	70.2	0.45	Moderate	—

These results demonstrate that integrating algorithmic coding through Scratch substantially enhanced students' mathematical problem-solving skills compared with conventional mathematics instruction.

Classroom Observation and Student Responses

Classroom observations revealed that students in the experimental group exhibited 2.3 times more classroom interaction than those in the control group ($p = 0.002$). Students were more actively engaged in discussing programming logic, solving mathematical problems collaboratively, and debugging Scratch programs.

The questionnaire results also indicated highly positive student perceptions toward the integration of Scratch into mathematics learning. Approximately 90% of the students agreed that Scratch facilitated their understanding of mathematical algorithms, 93% reported increased confidence in solving complex mathematical problems, and 87% considered Scratch relevant to junior high school mathematics learning.

Table 5. Students' Responses to Scratch-Based Coding Integration

Statement	Agreement (%)	Mean Likert Score
Scratch makes mathematical algorithms easier to understand	90	4.4
Scratch increases confidence in solving mathematical problems	93	4.5
Scratch is relevant to junior high school mathematics	87	4.2
Overall Average	90	4.3

Overall, students responded positively to the implementation of algorithmic coding, indicating that Scratch not only enhanced conceptual understanding but also increased motivation, confidence, and engagement in mathematics learning.

All statistical assumptions required for inferential analysis were satisfied, including normality ($p > 0.05$) and homogeneity of variance ($p > 0.10$). Therefore, the hypothesis testing results provide strong evidence that the integration of algorithmic coding significantly improves junior high school students' mathematical problem-solving skills.

Discussion

The findings of this study demonstrate that integrating algorithmic coding through Scratch significantly improves junior high school students' mathematical problem-solving skills. The experimental group achieved a high normalized gain **(**N-Gain = 0.72**)** and outperformed the control group by 12.4 points on the posttest (82.6 vs. 70.2, $p < 0.001$), with a large effect size (Cohen's ***d*** = 1.28**).** These results indicate that algorithmic coding serves as an effective instructional strategy for strengthening students' mathematical reasoning and systematic problem-solving abilities.

The most substantial improvements occurred in Polya's stages of devising a plan (+32%) and reviewing the solution (+41%). These findings suggest that Scratch's visual programming environment encourages students to construct logical procedures systematically while continuously evaluating the correctness of their solutions. The block-based programming interface enables learners to visualize mathematical processes step by step, thereby reducing the cognitive complexity associated with abstract reasoning. Consequently, students become more capable of organizing solution strategies and verifying mathematical results independently.

DOI: 10.5281/zenodo.22721535

These findings are consistent with previous research reported in *Mosharafa: Jurnal Pendidikan Matematika* (2022), which found that Scratch-based mathematics instruction enhanced students' critical thinking and problem-solving skills with an N-Gain of 0.68. However, the present study produced a slightly higher learning gain (0.72), suggesting that integrating algorithmic coding within algebra and number pattern instruction may provide additional benefits, particularly in rural educational contexts. The novelty of this study lies in demonstrating that meaningful technology-supported mathematics instruction can be successfully implemented even in schools with limited technological infrastructure.

From a theoretical perspective, Scratch transforms Polya's traditional problem-solving framework into a visual computational process. Students were able to decompose mathematical problems into sequential algorithmic procedures, thereby reducing the cognitive load associated with abstract algebraic reasoning. For example, programming a Caesar cipher to encode and decode the coefficients of linear equations required students to analyze mathematical relationships, construct algorithms, execute procedures, and verify solutions. This process strengthened the planning stage, which initially represented the weakest aspect of students' mathematical problem-solving performance.

Interestingly, students demonstrated greater improvement in number pattern topics (N-Gain = 0.78) than in algebra (N-Gain = 0.66). This difference can be explained by the natural correspondence between Scratch's loop structures and repetitive numerical sequences. Visualizing iterative processes through programming loops enables students to recognize patterns more intuitively than manipulating abstract algebraic variables. Consequently, Scratch appears particularly effective for mathematical topics involving sequences, repetition, and recursive thinking.

The implementation of algorithmic coding also supports the objectives of Indonesia's Merdeka Curriculum, which emphasizes computational thinking and digital literacy as essential competencies for twenty-first-century learning. Classroom observations revealed that student interaction in the experimental class was 2.3 times higher than in the control class. Students actively discussed programming logic, collaborated in debugging Scratch programs, and exchanged mathematical ideas while solving contextual problems. These findings suggest that coding not only improves cognitive achievement but also promotes collaborative learning and active classroom engagement.

Student perceptions further reinforce the effectiveness of the intervention. Approximately 90% of participants agreed that Scratch helped them understand mathematical algorithms more easily, while 93% reported increased confidence in solving complex mathematical problems. These positive responses indicate that integrating coding into mathematics instruction enhances students' motivation and self-efficacy in addition to improving academic performance. Increased confidence may encourage students to engage more actively in challenging mathematical tasks and persist when encountering difficulties.

Another notable finding concerns the feasibility of implementing coding instruction in a rural school with limited technological resources. Despite having only five computers for thirty students, the rotational group-learning strategy proved highly effective. Collaborative programming activities enabled all students to participate actively without requiring one device per learner. This finding demonstrates that successful integration of educational technology depends not only on the availability of hardware but also on appropriate instructional design and classroom management strategies. Therefore, schools with limited infrastructure can still implement coding-based mathematics instruction effectively through collaborative learning approaches.

Overall, the findings suggest that algorithmic coding functions not merely as a technological tool but as a pedagogical approach that strengthens mathematical reasoning, computational thinking, and systematic problem-solving. The integration of visual programming with mathematics instruction enables students to construct logical solution procedures, evaluate their reasoning, and communicate mathematical ideas more effectively. Consequently, algorithmic coding has considerable potential to become an innovative instructional strategy for improving mathematics education, particularly in the context of implementing the Merdeka Curriculum and preparing students with the competencies required for the digital era.

The findings of this study also have important implications for educational policy and classroom practice. As coding is increasingly introduced into the national curriculum, schools should provide professional development opportunities that enable mathematics teachers to integrate computational thinking into classroom instruction. Future research is recommended to investigate the effectiveness of algorithmic coding across different mathematical topics, educational levels, and instructional models, as well as to examine its long-term impact on higher-order thinking skills, creativity, and mathematical reasoning.

Conclusion and Recommendations

Conclusion

The findings of this study demonstrate that the integration of algorithmic coding into mathematics instruction has a positive and significant effect on junior high school students' mathematical problem-solving skills. Mathematics learning that emphasizes the development of algorithmic procedures and systematic reasoning enables students to understand mathematical problems more effectively, formulate appropriate solution strategies, execute logical procedures, and evaluate the accuracy of their solutions. Students who participated in Scratch-based coding activities showed significantly higher learning gains than those who received conventional instruction, indicating that algorithmic coding serves as an effective instructional approach for strengthening mathematical reasoning.

Furthermore, the findings suggest that algorithmic coding extends beyond the development of digital literacy by functioning as a powerful pedagogical tool that promotes computational thinking, logical reasoning, and systematic problem-solving. The visual programming environment provided by Scratch supports students in translating abstract mathematical concepts into structured computational processes, thereby facilitating conceptual understanding and enhancing learning engagement. Consequently, integrating algorithmic coding into mathematics instruction represents a promising innovation for improving the quality of mathematics education, particularly within the implementation of Indonesia's Merdeka Curriculum.

DOI: 10.5281/zenodo.22721535

Recommendations

Based on the findings of this study, several recommendations can be proposed.

First, mathematics teachers are encouraged to integrate algorithmic coding systematically into classroom instruction, particularly when teaching mathematical topics that require logical reasoning, pattern recognition, and problem-solving processes. Visual programming platforms such as Scratch can provide meaningful learning experiences by connecting mathematical concepts with computational thinking.

Second, school administrators and educational policy makers should support the implementation of coding-integrated mathematics instruction by providing professional development programs for teachers and improving access to technology-based learning resources. Although this study demonstrated that effective implementation is possible in schools with limited technological infrastructure, broader institutional support will facilitate more sustainable and scalable practices.

Finally, future research is recommended to investigate the effectiveness of algorithmic coding across different mathematical topics, educational levels, and instructional models. Further studies may also examine its influence on other higher-order cognitive outcomes, including critical thinking, creative thinking, mathematical reasoning, computational thinking, and students' self-regulated learning. In addition, longitudinal and mixed-methods studies are recommended to explore the long-term impact of coding-integrated mathematics instruction on students' academic achievement and digital competencies.

BIBLIOGRAPHY

1. Amellia, Z., Fonna, M., & Isfayani, E. (2022). Peningkatan kemampuan berpikir kritis matematis siswa dengan menggunakan model pembelajaran inkuiri. *Ar-Riyadhiyyat: Journal of Mathematics Education*, *3*(1), 1–9.
2. Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Proceedings of the American Educational Research Association.
3. Cui, Z., & Ng, O.-L. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research*, *59*(5), 988–1012. https://doi.org/10.1177/0735633120979930
4. Editorial Team. (2023). Efektivitas integrasi teknologi dalam pembelajaran matematika SMP. *Math Didactic: Jurnal Pendidikan Matematika*.
5. Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, *29*(1), 12–28. https://doi.org/10.1002/cae.22255
6. Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance et al. (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Academic.
7. Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310.
8. Kristanto, D. (2022). Analisis kemampuan pemecahan masalah matematis siswa SMP. *Mosharafa: Jurnal Pendidikan Matematika*, *11*(1), 107–118.
9. Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming. *Computers in Human Behavior*, *41*, 51–61.
10. Mazaly, M. R., & Saragih, D. I. (2022). Penerapan pembelajaran matematika melalui media komputer terhadap hasil belajar siswa SMP. *Ar-Riyadhiyyat: Journal of Mathematics Education*, *3*(1), 31–40.
11. Montiel, H., & Gomez-Zermeño, M. G. (2021). Educational challenges for computational thinking in K–12 education: A systematic literature review of Scratch as an innovative programming tool. *Computers*, *10*(6), 69. https://doi.org/10.3390/computers10060069
12. Ng, O.-L., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM–Mathematics Education*, *53*(4), 847–860.
13. Ouahouda, F., Achtaich, K., & Achtaich, N. (2026). Enhancing algorithmic thinking in primary education through Scratch programming using a practical framework and classroom case study. *Discover Education*, *5*, Article 601. https://doi.org/10.1007/s44217-026-01498-7
14. Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.
15. Ririn. (2021). Peningkatan kemampuan berpikir kritis matematis dan kemandirian belajar siswa melalui model pembelajaran *problem solving*. *Mathema: Jurnal Pendidikan Matematika*, *3*(1).
16. Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158.
17. Stewart, W. H., & Baek, K. (2022). Analyzing computational thinking studies in Scratch programming: A review of elementary education literature. *International Journal of Computer Science Education in Schools*, *6*(1). https://doi.org/10.21585/ijcses.v6i1.156
18. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.
19. Wing, J. M. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*, Carnegie Mellon University.
20. Yanti, R. A., Nindiasari, H., & Ihsanudin, I. (2020). Analisis kemampuan pemahaman konsep matematis siswa SMP dengan pembelajaran daring. *Wilangan:*

DOI: 10.5281/zenodo.22721535

Jurnal Inovasi dan Riset Pendidikan Matematika, 1(3), 256–266.

21. Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62.

DOI: 10.5281/zenodo.22721535