Jurnal Pendidikan Islam 12 (1) (2026) 75-88
DOI: https://doi.org/10.15575/jpi.v12i1.42091
http://journal.uinsgd.ac.id/index.php/jpi

p-ISSN: 2355-4339
e-ISSN: 2460-8149

COMPUTATIONAL THINKING AND MATHEMATICAL PROBLEM SOLVING IN MADRASAH TSANAWIYAH STUDENTS

I Made Suarsana[1], Tatang Herman[1], Elah Nurlaelah[1]*, Didi Suryadi[1], Irianto[2], Al Jupri[1], Asep Bayu Dani Nandiyanto[3], Zulkaidah Nur Ahzan[4]
[1]Department of Mathematics Education, Universitas Pendidikan Indonesia, Bandung, Indonesia
[2]Department General Education, Rabdan Academy, Abu Dhabi, United Arab Emirates
[3]Department of Chemistry, Universitas Pendidikan Indonesia, Bandung, Indonesia
[4]Mathematics Education Study Program, Universitas Timor, Nusa Tenggara Timur, Indonesia
*Corresponding Email: elah_nurlaelah@upi.edu

Received: December 2024. Accepted: May 2025. Online First: January 2026. Published: June 2026

ABSTRACT
The integration of computational thinking into mathematics learning is increasingly important in preparing students for problem-solving in the digital era, and visual programming tools such as Flowgorithm offer potential pedagogical support. This study analyzes the effect of Flowgorithm-assisted instruction on students' computational thinking skills and mathematical problem-solving abilities. A quasi-experimental pretest–posttest control group design was employed and implemented in two phases: a programming phase and a mathematics learning phase. The population consisted of all eighth-grade students at Madrasah Tsanawiyah in Bali Province, Indonesia, from which 34 students were selected using cluster random sampling (20 in the experimental group and 14 in the control group). Data on computational thinking and mathematical problem-solving abilities were collected using achievement tests and analyzed using MANCOVA, followed by post hoc tests, with effect sizes reported using partial eta squared. The findings reveal significant differences between the experimental and control groups, with post hoc analysis indicating that Flowgorithm-assisted mathematics learning significantly improved students' computational thinking skills, while no significant difference was found in mathematical problem-solving abilities. Despite a non-significant effect on problem-solving, the integration of Flowgorithm showed no negative impact, suggesting that Flowgorithm-assisted instruction can be a viable approach for embedding computational thinking concepts into mathematics curricula and instructional practices.

Keywords: Computational Thinking, Flowgorithm, Problem-Solving, Programming, Madrasah

INTRODUCTION
The significance of computational thinking (CT) abilities in 21st-century life skills has been widely recognized (Haseski et al., 2018). CT is a skill essential for individuals to succeed in global competition; therefore, efforts should be made to integrate CT into schools (Miterianifa et al., 2021). Thus, the CT concept, initially developed in computer science, needs to be adapted and expanded to the field of education, especially learning in schools. Initially, Jeannette Wing defined CT as a thinking process for formulating problems and solutions, representing them effectively through information processing agents (Varela et al., 2019). After that, the definition of CT developed rapidly and tended to diverge. A unique CT framework for mathematics learning was revealed by Weintrop et al. (2015), who divided CT into four categories: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices. Concerning the CT capability aspect, Selby and Woollard (2014) and Hoyles and Noss (2015) break it down into four aspects, also known as PRADA (pattern recognition, abstraction, decomposition, algorithm). The current research trend in CT integration for mathematics teaching utilizes the PRADA framework (Irawan et al., 2024).

Other trends in CT research within mathematics education were further unveiled by Ye et al. (2023) through a systematic literature review. Their findings indicated that a common intervention to foster CT in math education is the inclusion of programming activities. The current potential for integrating programming tasks into school curricula, particularly within junior high school mathematics education, presents a significant opportunity to enhance CT skills and mathematical performance (Chan et al., 2022), particularly in the context of problem-solving (Robins et al., 2003; Ye et al., 2023). Prior researchers had undertaken various studies on incorporating programming into math education. Cameto et al. (2019), Carboni et al. (2018), and Zeng et al. (2023) integrated the MateFun programming language into function learning in junior high schools, noting no substantial difference in math abilities compared to traditional methods, yet students gained fundamental programming knowledge. MateFun, a text-based language utilizing function commands as primary computing units, was employed. Yuana et al. (2015) demonstrated that blending science-based programming with virtual robots in 10th-grade math classes facilitates the comprehension of math concepts while introducing programming principles. Park and Manley (2024) observed that text-based programming integration enhanced 11th-grade students' grasp of mathematical concepts. Iwamoto and Matsumoto (2019) found that utilizing a visual block editor in junior high math education significantly improved math learning outcomes. Furthermore, Spencer et al. (2023) discovered that integrating visual programming languages into elementary math education could enhance student engagement with mathematical concepts, support problem-solving and reasoning skills, and encourage mathematical discourse.

Overall, previous research demonstrated that integrating programming into mathematics education, whether visual-based or text-based, did not negatively influence it. Furthermore, a significant benefit was the early introduction of computational thinking concepts. Literature reviews indicated that research on programming integration in junior high school mathematics had predominantly involved visual or block-based programming (Karaliopoulou et al., 2018; Ye et al., 2023; Yi & Lee, 2018), such as Scratch (Park & Shin, 2022). Visual programming tools like Scratch appealed to school-aged students because they made learning enjoyable; however, challenges can arise when transitioning from visual to text-based coding (Yadav et al., 2016; Papadakis & Kalogiannakis, 2019). Text-based programming languages were preferred for developing coding skills, although they were often considered complex for students at the school level (Weintrop & Wilensky, 2017). Flowgorithm, a user-friendly graphical flowchart-based programming language for beginners (Gajewski & Smyrnova-Trybulska, 2018), served as a potential solution for bridging the gaps between visual and text-based programming (Xu et al., 2019). Students could focus on programming concepts through flowcharts, facilitating a smoother transition to actual coding (Ho et al., 2021). Additionally, flowcharts created could be directly converted into over 18 languages, including C#, C++, Java, JavaScript, Lua, Perl, Python, Ruby, Swift, Visual Basic .NET, and VBA.

The integration of Flowgorithm, a combination of block-based and text-based programming, in mathematics learning has been an area of research that has received limited attention. Studies on the effectiveness of using flowchart programming were mainly conducted in computer science to introduce programming concepts and enhance problem-solving abilities (Hooshyar et al., 2015). Flowgorithm has been demonstrated as an effective tool in programming classes for presenting algorithms and their results (Gajewski & Smyrnova-Trybulska, 2018), clearly distinguishing between programming (creating algorithms) and coding (translating algorithms into specific programming languages). To bridge this gap, the present study aimed to explore the integration of Flowgorithm into mathematics learning within the unique context of Madrasah Tsanawiyah (MTs), investigating its potential impact on both computational thinking and mathematical problem-solving abilities, areas that had received limited attention in prior research. Consequently, a quasi-experimental study was carried out with a population of grade 8 students, specifically targeting students from MTs in Bali Province. These MTs are formal schools that focus on Islamic education, equivalent to junior high level, resembling public schools with Islamic characteristics (Maryati et al., 2023). No previous studies have investigated the integration of programming into mathematics learning at Islamic boarding schools. Hence, the novelty of this research lies in integrating hybrid programming into mathematics education and involving research subjects from formal schools with Islamic characteristics. Another novel aspect examined the influence of programming integration on mathematical problem-solving abilities. Prior research had only revealed the impact of programming activities on general mathematics performance, including understanding mathematical concepts and processes, interest in mathematics (Goldenberg & Carter, 2021), and self-efficacy (Chiang et al., 2022). Despite indications from Kuen (2011) that programming activities had the potential to develop problem-solving skills, no experimental research had explicitly examined the effect of programming activities on problem-solving abilities

METHOD

The research involved all grade 8 students from 36 MTs in Bali Province. Using a simple cluster random sampling technique, the researchers randomly selected 2 classes from the 8th-grade population, yielding a sample of 2. These classes were from MTS Mambaus Sholihin Jembrana, comprising 18 students, and MTs Kalifa Nusantara, Denpasar, with 26 students in the 8th grade. Subsequently, randomization was conducted to assign participants to the control and experimental groups. Initially comprising 44 participants, there was a loss of 10 participants whose data had to be excluded as they did not engage fully in the research phase. Consequently, the research samples comprised the control group from MTs Mambaus Sholihin (14 members; 6 males, 8 females) and the experimental group from MTs Kalifa Nusantara (20 members; 7 males, 13 females). Neither group possessed prior knowledge or skills in programming with Flowgorithm.

The research design for this study involved a quasi-experimental approach with two phases implemented in both the experimental and control groups. More details are presented in Figure 2. The first stage, the programming phase, involved acquiring fundamental knowledge and skills necessary for programming using Flowgorithm. This phase was administered to both groups, each lasting 90 minutes, totaling three sessions. The CT concepts covered in this phase included basic sequences, loops, iteration, conditionals, functions, and variables (Román-González et al., 2017). These CT concepts were deemed suitable for students aged 12-14 years (grades 7 and 8) (Tsarava et al., 2022) and were aligned with the CT framework proposed by Brennan and Resnick (2012) as well as the computer science standards for grades 6-9 established by the Computer Science Teachers Association (Seehorn et al., 2011).

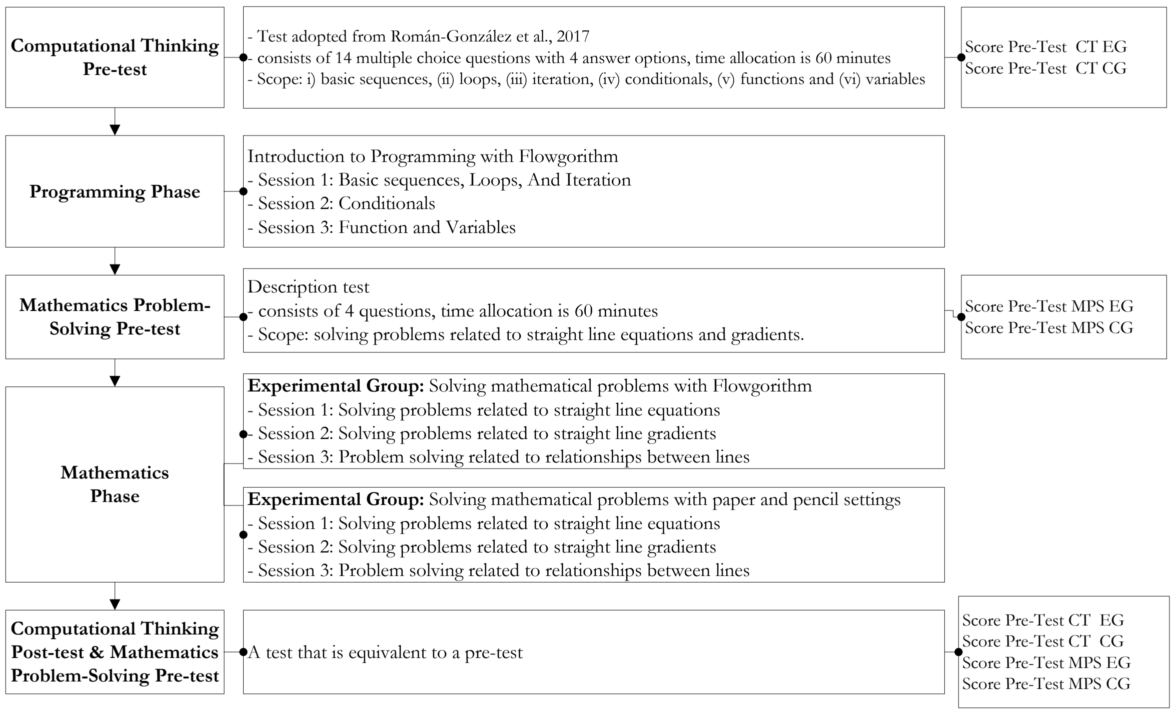
Computational Thinking Pre-test
- Test adopted from Román-González et al., 2017
- consists of 14 multiple choice questions with 4 answer options, time allocation is 60 minutes
- Scope: i) basic sequences, (ii) loops, (iii) iteration, (iv) conditionals, (v) functions and (vi) variables
Score Pre-Test CT EG
Score Pre-Test CT CG
Programming Phase
Introduction to Programming with Flowgorithm
- Session 1: Basic sequences, Loops, And Iteration
- Session 2: Conditionals
- Session 3: Function and Variables
Mathematics Problem-Solving Pre-test
Description test
- consists of 4 questions, time allocation is 60 minutes
- Scope: solving problems related to straight line equations and gradients.
Score Pre-Test MPS EG
Score Pre-Test MPS CG
Mathematics Phase
Experimental Group: Solving mathematical problems with Flowgorithm
- Session 1: Solving problems related to straight line equations
- Session 2: Solving problems related to straight line gradients
- Session 3: Problem solving related to relationships between lines
Experimental Group: Solving mathematical problems with paper and pencil settings
- Session 1: Solving problems related to straight line equations
- Session 2: Solving problems related to straight line gradients
- Session 3: Problem solving related to relationships between lines
Computational Thinking Post-test & Mathematics Problem-Solving Pre-test
A test that is equivalent to a pre-test
Score Pre-Test CT EG
Score Pre-Test CT CG
Score Pre-Test MPS EG
Score Pre-Test MPS CG

Figure 1. Research Procedure

In the second phase, known as the mathematics phase, the emphasis shifted to problem-solving tasks, with Flowgorithm used as an instructional aid. Students participated in 3 sessions focused on solving problems involving straight-line equations, aligning with the learning objectives outlined in the Indonesian curriculum. Session 1 focused on determining the gradient of a straight line. Session 2 involved solving problems to establish the relationship between two straight lines. In Session 3, students engaged in problem-solving exercises to identify the equation of a straight line.

In contrast to the first phase, the second phase involved distinct treatments for the experimental and control groups. The experimental group utilized a Flowgorithm to tackle assigned tasks, whereas the control group completed the tasks without the aid of a Flowgorithm, relying on traditional tools like paper and pencil. Each session occurred twice a week.

Before the programming phase, both groups underwent a pre-test to evaluate their initial CT skills. Similarly, before the commencement of the mathematics phase, students also took a pre-test to gauge their initial mathematical problem-solving abilities. All assessments conducted in this study lasted 60 minutes and were conducted concurrently for both groups. At the end of the second phase, both groups of students participated in two post-tests to assess their CT and mathematical problem-solving abilities. Contrasting the pre-test and post-test results will enable an analysis of the progress in acquiring CT concepts and mathematical problem-solving skills under each experimental condition.

The study utilized two instruments: a CT test and a mathematics problem-solving test. The CT test, adapted from Román-González et al. (2017), comprised 14 multiple-choice items targeting seven computing concepts, including loops, conditionals, and simple functions, designed for grade 7 and 8 students. The test demonstrated validity ($0.27 < r < 0.44$) and reliability (CCronbach'sAlpha $\alpha =$ 0.793), with items arranged by increasing difficulty. Equivalent pre-test and post-test versions were administered online via Google Forms, differing only in question descriptions while maintaining consistent difficulty levels. The mathematics problem-solving test focused on straight-line equations and comprised four multiple-choice questions with four answer options. Two versions of the same test with identical structures and features were administered for both the pre-test and post-test. The researcher developed the problem-solving ability tester and underwent validation by experts, alongside field trials that confirmed the validity of the items ($0.378 < r < 0.9233$) and a CCronbach'sAlpha reliability of $\alpha = 0.760$. Additionally, the test included a Certainty of Response Index (CRI) questionnaire to distinguish between students who misunderstood the problem and those who did not comprehend it (Putri et al., 2021). Students were required to indicate their confidence level in each response, ranging from 100% guessing to 100% certainty.

Data analysis was performed on post-test score data with pre-test scores used as covariates through multivariate analysis of covariance (MANCOVA). MANCOVA was selected because it provided a stronger hypothesis test compared to analysis of variance (Dugard & Todman, 1995). The MANCOVA calculations were carried out using the IBM SPSS Statistics 29.0.1.0 application, employing general linear model multivariate statistics with the Bonferroni correction. Effect size was calculated using Cohen's formula (d) and partial eta squared statistics (η_p^2). Effect size interpretation, based on Cohen's d and partial eta squared (η_p^2), is categorized as negligible $(0 < d < 0.2, 0 < \eta_p^2 < 0.01)$, small ($0.2<d\leq0.5$, $0.01<\eta_p^2\leq0.06$), medium ($0.5<d\leq0.8$, $0.06<\eta_p^2\leq0.14$), and large ($d>0.8$, $\eta_p^2>0.14$)(Gignac & Szodorai, 2016; Stephen & Rockinson-Szapkiw, 2021).

RESULTS AND DISCUSSION

The pre-test and post-test results for each aspect of the CT concept are displayed in Table 1. The optimal maximum score for each aspect of the CT Concept was 20, with 140 being the ideal maximum score for the entire test. The scores for each CT concept showed that the more complex the concept, the lower the average score. This pattern was consistent in the pre-test and post-test for both the control and experimental groups. The average CT post-test scores for the control and experimental groups were 83.571 and 104.500, respectively. There was a descriptive increase in the CT score in each group compared to the pre-test score. Using Cohen's formula, the effect size (d) in the control group and experimental group were 0.489 (small) and 0.716 (medium), respectively. The improvement observed in the experimental group was notably larger in quantity and quality.

Both groups exhibited increased CT scores when analyzed based on the CT concept aspect. In the control group, the most significant increase in CT score was in the "loop repeat until" aspect, while the smallest increase was noted in the "basic direction and sequence" aspect. Conversely, in the experimental group, the most substantial increase in CT scores was seen in the "If simple conditional" aspect, with the smallest increase also occurring in the "basic direction and sequence" aspect. These results suggest that teaching basic CT concepts using Flowgorithm in Phase I could enhance students' CT abilities. Moreover, inferential statistics are required to evaluate the impact of various treatments in Phase II on CT capabilities.

Table 1. CT Pre-test and Post-Test Score

CT Concept	**Control Group**				**Experiment Group**			
	Pre-Test		**Post-Test**		**Pre-Test**		**Post-Test**	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**	**Mean**	**SD**
Basic directions and sequences	18.571	3.631	19.286	2.673	20.000	0.000	20,000	0,000
Loops repeat times	15.714	5.136	17.857	4.258	18.000	4.104	19,500	2,236
Loops repeat until	11.429	8.644	15.000	6.504	13.000	6.569	17,000	4,702
If simple conditional	9.286	6.157	11.429	6.630	11.000	7.182	16,500	4,894
If/else complex conditional	6.429	6.333	8.571	8.644	10.500	6.048	14,000	5,026
While conditional	5.000	5.189	7.857	5.789	7.500	5.501	10,000	6,489

CT Concept	Control Group				Experiment Group			
	Pre-Test		Post-Test		Pre-Test		Post-Test	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Simple functions	2.857	4.688	3.571	4.972	6.000	5.982	7,500	6,387
Total	69.286	31.736	83.571	26.489	86	29.629	104.500	21.392
Cohen's (d)	0.489				0.716			

The results of the pre-test and post-test assessments of problem-solving abilities in the straight-line equation material have been outlined in Table 2. The optimal maximum score for each problem-solving item was 10, with the total test score ideally reaching 40. Upon reviewing the scores per indicator, it became evident that many students encountered challenges when solving problems related to determining points on a line with a known gradient/equation. This pattern was consistent across the pre-test and post-test assessments for both the control group and experimental group. The average post-test scores for problem-solving in the control group and experimental group were 22.857 and 27.000, respectively. Descriptively, there was an increase in problem-solving ability scores in each group compared to the pre-test results.

The effect size in the control group was d=1.022 (large), while in the experimental group, it was d=1.301 (large). Quantitatively, the effect size in the experimental group exceeded that of the control group, although both groups fell within the "large" effect size category. The problem-solving test scores improved in both groups for each question item, indicating that problem-solving activities using both Flowgorithm and traditional methods can enhance students' problem-solving abilities. Inferential statistics are essential to ascertain which treatments can significantly impact the development of problem-solving skills.

Table 2. Mathematics Problem Solving Pre-test and Post-Test Score

Problem-Solving Indicator	Control Group				Experiment Group			
	Pre-Test		Post-Test		Pre-Test		Post-Test	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Determine the equation of a straight line if the gradient and the point through which it passes are known.	0.714	2.673	3.571	4.972	1.500	3.663	5.500	5.104
Determine the point through which a line passes if the gradient/equation is known	0.714	2.673	1.429	3.631	0.500	2.236	2.500	4.443
Determine the gradient of a line if the magnitude of the change in the horizontal and vertical directions is known	7.143	4.688	10.000	0.000	6.500	4.894	9.000	3.078
Determine the gradient relationship of two parallel lines	5.714	5.136	7.857	4.258	8.000	4.104	10.000	0.000
Total	14.286	7.559	22.857	9.139	16.500	9.333	27.000	6.569
Cohen's (d)	1.022				1.301			

The post-test instrument for assessing mathematical problem-solving abilities was also complemented with the CRI questionnaire. The results from the analysis, detailing accuracy and students' confidence levels, are presented in Table 3. In the control group, 12.50% of respondents provided correct answers with a high confidence level, while 30.36% offered incorrect answers with high confidence. This suggests that, among the 100 respondents in the control group, 13 individuals comprehended and applied the straight-line equation accurately, 31 misunderstood the concept, and the remaining 56 were unaware of it, often resorting to guesses in their responses.

Conversely, in the experimental group, 23.75% of respondents answered correctly with high confidence, and 28.75% answered incorrectly with high confidence. This indicates that out of 100 respondents in the experimental group, 24 grasped and effectively applied the concept in problem-solving, 29 misunderstood the concept, and 47 were unaware. The proportion of students exhibiting a correct understanding in the experimental group exceeded that in the control group, indicating that incorporating Flowgorithm in problem-solving activities facilitated a deeper understanding of concepts than traditional paper-and-pencil methods. Moreover, misconceptions among students in the experimental group were lower than in the control group. Utilizing flowgorithm programming in mathematical problem-solving activities proved to be more efficacious in mitigating misconceptions among students than conventional paper-and-pencil methods.

Table 3. Percentage of Respondents Based on Answer Criteria

Answer Criteria		**% Respondents**				
		Problem 1	Problem 2	Problem 3	Problem 4	All
Control Group						
True Answer	High CRI Value (>2.5)	21,43	0,00	7,14	21,43	12,50
	Low CRI Value (< 2.5)	7,14	7,14	42,86	28,57	21,43
Wrong Answer	Low CRI Value (< 2.5)	35,71	50,00	28,57	28,57	35,71
	High CRI Value (>2.5)	35,71	42,86	21,43	21,43	30,36
Experiment Group						
True Answer	High CRI Value (>2.5)	30,00	15,00	35,00	15,00	23,75
	Low CRI Value (< 2.5)	15,00	20,00	25,00	10,00	17,5
Wrong Answer	Low CRI Value (< 2.5)	30,00	20,00	40,00	30,00	30
	High CRI Value (>2.5)	25,00	45,00	0,00	45,00	28,75

The descriptive outcomes mentioned above were subjected to significance testing for differences or effects using inferential statistics. Prior to conducting the MANCOVA inferential test, a prerequisite test was performed. In SPSS, a normality assessment was conducted on the residual variables of the CT post-test and mathematics problem-solving post-test using the Shapiro-Wilk test. The Shapiro-Wilk statistical value for the CT post-test residual variable was 0.977 with a significance of 0.673. Similarly, for the mathematics problem-solving post-test residual, the Shapiro-Wilk statistical value was 0.960 with a significance of 0.244. As both significance values were >0.05, the sample was concluded to be drawn from a normally distributed population.

Subsequently, the homogeneity of the covariance matrix was assessed using Box's test as another prerequisite. The SPSS analysis indicated that Box's M value was 2.582, $F = 0.800$, df1=3, df2=50,117 with a significance of 0.494 > 0.05, signifying that the covariance matrix among groups was homogeneous. The third prerequisite involved testing the homogeneity of variance using Levene's test. The SPSS computations revealed that the F statistical value for the post-test variable was 0.002, a significance of 0.964, and for the mathematics problem-solving post-test variable, the F statistical value was 0.151 with a significance of 0.700. Given that both significance values were >0.05, it was inferred that the two population variances were homogeneous. Consequently, all prerequisite tests for the MANCOVA analysis were satisfied, leading to the subsequent testing of hypothesis H0.

H0: There is no significant difference in the use of Flowgorithm on students' CT abilities and mathematical problem-solving skills after controlling for the pre-test covariate.

H1: There is a significant difference in the use of Flowgorithm on students' CT abilities and mathematical problem-solving skills after controlling for the pre-test covariate.

The results of the MANCOVA test conducted using SPSS Statistics 29.0.1.0 with a general linear multivariate model approach are presented in Table 4.

Table 4. Multivariate Tests

Effect		Value	F	Hypothesis df	Error df	Sig.	Partial Eta Squared
Group	Pillai's Trace	0.256	4.988	2.000	29.000	0.014	0.256
	Wilks' Lambda	0.744	4.988	2.000	29.000	0.014	0.256
	Hotelling's Trace	0.344	4.988	2.000	29.000	0.014	0.256
	Roy's Largest Root	0.344	4.988	2.000	29.000	0.014	0.256

The Wilks' Lambda statistic value is 0.744, and the effect size value. η_p^2 is 0.256 with a significance of 0.014, which is less than 0.05. Therefore, it can be concluded that using Flowgorithm has a significant impact on both computational thinking abilities and mathematical problem-solving, while controlling for initial abilities. The effect size, as indicated by the η_p^2 value is considered "large". The post hoc test utilized the Benferroni test to identify specific differences between groups individually. The outcomes of the Benferroni post hoc test are detailed in Table 5.

Table 5. Tests of Between-Subjects Effects

Source	Dependent Variable	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Group	Computational Thinking Post-test	425.913	1	425.913	6.523	0.016	0.179
	Mathematics Problem Solving Post-test	59.900	1	59.900	2.219	0.147	0.069

The results of the Tests of Between-Subjects Effects on each dependent variable indicated that the utilization of Flowgorithm in mathematics education had a significant impact on CT abilities ($F = 6.523, \rho = 0.016, \eta_p^2 = 0.179$) but did not yield a significant effect on mathematical problem-solving abilities ($F = 2.219, \rho = 0.147, \eta\rho^2 = 0.069$). The effect size of employing Flowgorithm for each dependent variable was assessed using the partial eta squared value as outlined in Table 1. The effect size for CT ability due to Flowgorithm implementation fell in the "large" category, whereas for mathematical problem-solving ability, it was "medium". Therefore, although the statistical significance of Flowgorithm's impact on problem-solving abilities was not established, the descriptive assessment indicated a "medium" level of influence.

The results showed that the use of a Flowgorithm in mathematics learning significantly influenced the CT abilities of grade VIII MTs students. However, it did not significantly affect their ability to solve mathematical problems. Although it did not significantly improve problem-solving abilities, this study still provides valuable insights into integrating programming activities into mathematics learning. Furthermore, the research findings are elaborated through in-depth analysis, interpretations based on relevant theories, and comparisons with previous studies to identify consistencies and differences.

The significant increase in CT ability in the experimental group demonstrates the effectiveness of using Flowgorithm in supporting mathematics learning. The average CT ability score in classes that integrated Flowgorithm in mathematics learning ($M = 104.50$) was higher than the average CT ability score in mathematics classes that did not use Flowgorithm ($M = 83.57$). The large effect size ($\eta\rho^2 = 0.179$) falls into the "large" category. The results of the multivariate post hoc test, as shown in Table 5, yielded an F value of 6.523 and a significance value of $\rho = 0.016$, which indicates a significant difference in students' CT ability between mathematics classes that used Flowgorithm and those that did not. These results can be attributed to the fact that integrating Flowgorithm facilitates students' understanding of concepts in a more structured and systematic manner through an interactive visual approach. This allows key aspects of CT, such as decomposition, abstraction, pattern recognition, and algorithms, to develop effectively. This finding is further reinforced by the results shown in Table 2, which indicate that all aspects of CT ability improved in the experimental and control groups. However, the increase in CT ability in the control group was lower and less consistent. This suggests that providing students with opportunities to solve mathematical problems using Flowgorithm effectively strengthens their CT ability, as they can directly practice CT concepts (basic sequences, loops, iterations, conditionals, functions, and variables) as tools for solving mathematical problems

By engaging in mathematical problem-solving activities using Flowgorithm, students are exposed to various elements that contribute to CT development, such as task contextualization, collaborative learning, and scaffolding in programming tasks (Sue Sentance & Csizmadia, 2016). Students also learn to recognize patterns, decompose problems, abstract information, and construct algorithms to solve mathematical problems (Miller, 2019). Flowgorithm allows students to construct algorithms more effectively than visual or text-based programming environments (Gajewski & Smyrnova-Trybulska, 2018). In contrast, students in the control class did not engage in problem-solving activities using Flowgorithm as an information-processing tool. This makes it more difficult for students to develop CT skills and solve problems effectively (Veerasamy et al., 2018). Furthermore, the absence of Flowgorithm-based problem-solving activities limits students' opportunities to enhance their metacognitive skills (Abdullah et al., 2017).

The findings of this study align with previous research, which indicates that integrating programming activities into mathematics learning can enhance students' CT abilities (Chan et al., 2022; S Sentance & Csizmadia, 2017). Various programming applications have been employed in prior studies, including Scratch (Maraza-Quispe et al., 2021; Park & Shin, 2022; Rodríguez-Martínez et al., 2020), MateFun (Cameto et al., 2019; Carboni et al., 2018), Visual Block Editor (Iwamoto & Matsumoto, 2019), virtual robots (Yuana et al., 2015), and Code.org (Oluk & Çakır, 2021). Regarding mathematics learning in junior high schools, programming integration is predominantly characterized by visual or block-based programming approaches (Karaliopoulou et al., 2018; Ye et al., 2023; Yi & Lee, 2018). The primary distinction between this study and previous research lies in the use of Flowgorithm, which emphasizes the transition from visual to text-based programming. This approach better prepares students for text-based programming environments, as proposed by Weintrop and Wilensky (2017). Thus, this study's findings complement previous research by demonstrating that the integration of hybrid programming applications, such as Flowgorithm, in mathematics learning can significantly enhance the CT abilities of junior high school students, similar to the effects observed with visual-based programming. Moreover, this study provides additional insights into the effectiveness of programming integration in mathematics learning in regular and formal schools with strong Islamic characteristics.

Different findings emerged from further tests on the aspect of problem-solving ability. Although the average problem-solving ability score in classes that integrated Flowgorithm into mathematics learning ($M = 27.00$) was higher than in mathematics classes that did not use Flowgorithm ($M = 22.857$). The effect size ($\eta\rho^2 = 0.069$) reached the “medium” category, and the MANCOVA post hoc test results yielded an $\mathbf{F}$ value of 2.219 and a significance level of $\rho = 0.147$. This indicates no significant difference in students’ mathematical problem-solving abilities between classes that used Flowgorithm and those that did not. As shown in Table 2, scores increased across all problem-solving indicators in both the control and experimental groups. However, the improvements were relatively small. For example, in the indicator *“Determining the points passed by the line if the gradient is known,”* scores only increased from 0.5 to 2.5, suggesting that students struggle with questions requiring high-level thinking skills. This finding is not consistent with the results of previous studies. Robins et al. (2003), Ye et al. (2023), Costa et al. (2017), and Spencer et al. (2023)found that programming activities in mathematics learning have a significant effect on students’ mathematical problem-solving abilities. The discrepancy in findings is likely due to differences in the complexity of the material addressed in the studies. While the present study focused on relatively simple material, Spencer et al. (2023) utilized more complex problem-solving tasks.

Although no significant effect was found, the results of this study demonstrated that using flow Algorithms in mathematics learning did not produce any negative impact. The effect size of using flow algorithms on problem-solving ability ($\eta\rho^2 = 0.069$) falls within the moderate category. While a positive effect was observed, it did not reach statistical significance. Another factor worth considering is analyzing the percentage of respondents based on the answer criteria using the CRI, as shown in Table 3. In the experimental group, 23.75% of students answered correctly with high confidence, compared to only 12.50% in the control group. Additionally, Table 3 indicates that the experimental group exhibited fewer misconceptions (28.75%) than the control group (30.36%). These findings suggest that, despite the lack of statistical significance, using the Flowgorithm in the experimental group supported a deeper understanding of mathematical concepts and helped reduce misconceptions. Mathematical problem-solving activities using Flowgorithm provide a dynamic and interactive learning environment, allowing students to apply mathematical concepts more practically (Grover et al., 2015). Moreover, students can visualize abstract mathematical ideas, experiment with different solutions, and receive immediate feedback, which fosters a deeper understanding of the material (Witherspoon et al., 2017).

Based on the findings above, the Flowgorithm did not produce any negative effects; instead, it provided the advantage of introducing the concept of computational thinking (CT) earlier. This finding creates opportunities for a cross-disciplinary approach integrating CT concepts with mathematics learning. Training and dissemination of CT integration in mathematics education are necessary to ensure that mathematics teachers develop their CT skills and ability to integrate CT into their teaching practices. This is essential because the responsibility for fostering students’ CT skills should not rest solely with informatics teachers. Integrating programming activities into mathematics learning can help students better understand mathematical logic and algorithms. Teachers can utilize Flowgorithm to create more interactive learning environments and reduce the risk of misconceptions. Education policymakers should consider integrating programming activities into the mathematics curriculum as a strategic effort to develop 21st-century skills such as computational thinking.

Further research is needed to compare various programming approaches, block-based, text-based, or hybrid, to identify which form is most suitable for integrating into junior high school mathematics learning. Such research will provide valuable insights into the strengths and weaknesses of each programming approach. Additionally, future studies should explore the effects of CT integration on other mathematical abilities, such as reasoning, communication, and mathematical connections.

CONCLUSION

The effect of integrating programming activities in mathematics learning in MTs through Flowgorithm was examined. The findings showed a significant difference in the impact of the Flowgorithm on CT abilities and students' mathematical problem-solving abilities after controlling for pre-test covariates. A significant difference was observed in the influence of the Flowgorithm on CT abilities, whereas no significant impact was observed on mathematical problem-solving abilities. Furthermore, no negative effects were identified from the utilization of Flowgorithm. The effect size was positive and classified as "large." The explicit introduction of CT concepts through learning and providing ample opportunities for students in mathematical problem-solving using CT concepts successfully empowered students' computational thinking optimally.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support provided by the Indonesia Endowment Fund for Education (LPDP), Ministry of Finance, Republic of Indonesia, which made this research possible.

BIBLIOGRAPHY

Abdullah, A. H., Rahman, S. N. S. A., & Hamzah, M. H. (2017). Metacognitive Skills of Malaysian Students in Non-Routine Mathematical Problem Solving. *Bolema Boletim De Educação Matemática*, *31*(57), 310–322. https://doi.org/10.1590/1980-4415v31n57a15

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American educational research association*, 1, 25. Retrieved from https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Cameto, G., Carboni, A., Koleszar, V., Méndez, M., Tejera, G., Viera, M., & Wagner, J. (2019). Using functional programming to promote math learning. *Proceedings - 14th Latin American Conference on Learning Technologies, LACLO 2019*, 306–313. https://doi.org/10.1109/LACLO49268.2019.00059

Carboni, A., Koleszar, V., Tejera, G., Viera, M., & Wagner, J. (2018). MateFun: Functional Programming and Math with Adolescents. *Proceedings - 2018 44th Latin American Computing Conference, CLEI 2018*, 849–858. https://doi.org/10.1109/CLEI.2018.00106

Chan, S. W., Looi, C.-K., Ho, W. K., & Kim, M. S. (2022). Tools and Approaches for Integrating Computational Thinking and Mathematics: A Scoping Review of Current Empirical Studies. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331221098793

Chiang, F. K., Zhang, Y., Zhu, D., Shang, X., & Jiang, Z. (2022). The influence of online STEM education camps on students' self-efficacy, computational thinking, and task value. *Journal of science education and technology*, 31(4), 461-472. https://doi.org/10.1007/s10956-022-09967-y

Costa, E. J. F., Campos, L. M. R. S., & Guerrero, D. D. S. (2017). Computational thinking in mathematics education: A joint approach to encourage problem-solving ability. *Proceedings - Frontiers in Education Conference, FIE*, *2017-Octob*, 1–8. https://doi.org/10.1109/FIE.2017.8190655

Gajewski, R. R., and Smyrnova-Trybulska, E. (2018). Algorithms, programming, flowcharts and flowgorithm. *E-Learning and Smart Learning Environment for the Preparation of New Generation Specialists*, 393–408. Retrieved from http://www.studio-noa.pl/ig/pub/us/E-l-10/10-393.pdf

Gignac, G. E., & Szodorai, E. T. (2016). Effect size guidelines for individual differences researchers. *Personality and Individual Differences*, *102*, 74–78. https://doi.org/https://doi.org/10.1016/j.paid.2016.06.069

Goldenberg, E. P., & Carter, C. J. (2021). Programming as a language for young children to express and explore mathematics in school. *British Journal of Educational Technology*, 52(3), 969-985. https://doi.org/10.1111/bjet.13080

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

Haseski, H. İ., Ilic, U., & Tugtekin, U. (2018). Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. *International Education Studies*. https://doi.org/10.5539/ies.v11n4p29

Ho, W. K., Looi, C. K., Huang, W., Seow, P., & Wu, L. (2021). Computational thinking in mathematics: To be or not to be, that is the question. In T. L. Toh & L. H. Santos (Eds.), In *Mathematics—Connection and beyond: Yearbook 2020 association of mathematics educators* (pp. 205-234). World Scientific. https://doi.org/10.1142/9789811236983_0011

Hoyles, C., & Noss, R. (2015). A Computational Lens on Design Research. *ZDM*, *47*, 1039–1045. https://doi.org/10.1007/s11858-015-0731-2

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. J. (2015). A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of computer assisted learning*, 31(4), 345-361. https://doi.org/10.1111/jcal.12099

Irawan, E., Rosjanuardi, R., and Prabawanto, S. (2024). Research trends of computational thinking in mathematics learning: A bibliometric analysis from 2009 to 2023. *Eurasia Journal of Mathematics, Science and Technology Education*, *20*(3), em2417. https://doi.org/https://doi.org/10.29333/ejmste/14343

Iwamoto, T., & Matsumoto, S. (2019). Development of Web-Based Programming Learning Support System with Graph Drawing of Mathematics as a Learning Task. *Proceedings - 2019 8th International Congress on Advanced Applied Informatics, IIAI-AAI 2019*, 302–305. https://doi.org/10.1109/IIAI-AAI.2019.00067

Karaliopoulou, M., Apostolakis, I., & Kanidis, E. (2018). Perceptions of Informatics Teachers Regarding the Use of Block and Text Programming Environments. *European Journal of Engineering and Technology Research*. https://doi.org/10.24018/ejers.2018.0.cie.638

Maraza-Quispe, B., Sotelo-Jump, A. M., Alejandro-Oviedo, O. M., Quispe-Flores, L. M., Cari-Mogrovejo, L. H., Fernandez-Gambarini, W. C., & Cuadros-Paz, L. E. (2021). Towards the Development of Computational Thinking and Mathematical Logic through Scratch. *International Journal Of Advanced Computer Science And Applications*, *12*(2), 332–338. https://dx.doi.org/10.14569/IJACSA.2021.0120242

Maryati, S., Lestarika, L., Idi, A., & Tri Samiha, Y. (2023). Madrasah as an Institution of Islamic Education and Social Change. J*urnal Konseling Pendidikan Islam*, 4(2), 317–326. https://doi.org/10.32806/jkpi.v4i2.11

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: using coding to identify mathematical structures and patterns. *ZDM-MATHEMATICS EDUCATION*, *51*(6), 915–927. https://doi.org/10.1007/s11858-019-01096-y

Miterianifa, M., Ashadi, A., Saputro, S., & Suciati, S. (2021). *Higher Order Thinking Skills in the 21st Century: Critical Thinking*. https://doi.org/10.4108/eai.30-11-2020.2303766

Oluk, A., and Çakır, R. (2021). The Effect of Code. Org Activities on Computational Thinking and Algorithm Development Skills. *Journal of Teacher Education and Lifelong Learning*, *3*(2), 32–40. https://doi.org/10.51535/tell.960476

Park, Y., & Shin, Y. (2022). Text Processing Education Using a Block-Based Programming Language. *Ieee Access*. https://doi.org/10.1109/access.2022.3227765

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating a course for teaching advanced programming concepts with Scratch to preservice kindergarten teachers: A case study in Greece. In D. Farland-Smith (Ed.), *Early childhood education*. IntechOpen. https://doi.org/10.5772/intechopen.81714

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*. https://doi.org/10.1076/csed.13.2.137.14200

Rodríguez-Martínez, J. A., González-Calero, J. A., and Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, *28*(3), 316–327. https://doi.org/10.1080/10494820.2019.1612448

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in human behavior, 72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Selby, C., & Woollard, J. (2014). *Refining an understanding of computational thinking*. eprints.soton.ac.uk. https://eprints.soton.ac.uk/372410

Sentance, S, & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. In *Education and Information Technologies*. Springer. https://doi.org/10.1007/s10639-016-9482-0

Sentance, S, & Csizmadia, A. (2016). Computing in the Curriculum: Challenges and Strategies From a Teacher's Perspective. *Education and Information Technologies*, *22*(2), 469–495. https://doi.org/10.1007/s10639-016-9482-0

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., & Verno, A. (2011). *CSTA K--12 Computer Science Standards: Revised 2011*. ACM.

Spencer, D., Mark, J., Reed, K., Goldenberg, P., Coleman, K., Chiappinelli, K., and Kolar, Z. (2023). Using Programming to Express Mathematical Ideas. *Mathematics Teacher: Learning and Teaching PK-12*, *116*(5), 322–329. https://doi.org/https://doi.org/10.5951/MTLT.2022.0354

Stephen, J. S., & Rockinson-Szapkiw, A. J. (2021). A high-impact practice for online students: the use of a first-semester seminar course to promote self-regulation, self-direction, online learning self-efficacy. *Smart Learning Environments*, *8*(1), 6. https://doi.org/https://doi.org/10.1186/s40561-021-00151-0

Tsarava, K., Moeller, K., Román-González, M., Golle, J., Leifheit, L., Butz, M. V., & Ninaus, M. (2022). A cognitive definition of computational thinking in primary education. *Computers & Education*, 179, 104425. https://doi.org/10.1016/j.compedu.2021.104425

Varela, C., Rebollar, C., Garcia, O., Bravo, E., & Bilbao, J. (2019). Skills in computational thinking of engineering students of the first school year. *HELIYON*, *5*(11). https://doi.org/10.1016/j.heliyon.2019.e02820

Veerasamy, A. K., D'Souza, D., Lindén, R., & Laakso, M. (2018). Relationship Between Perceived Problem-solving Skills and Academic Performance of Novice Learners in Introductory Programming Courses. *Journal of Computer Assisted Learning*, *35*(2), 246–255. https://doi.org/10.1111/jcal.12326

Weintrop, D. (2015). *Comparing Text-Based, Blocks-Based, and Hybrid Blocks/Text Programming Tools*. https://doi.org/10.1145/2787622.2787752

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 1–25. https://doi.org/https://doi.org/10.1145/3089799

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing Computational Thinking through a Virtual Robotics Programming Curriculum. *ACM Transactions on Computing Education*, *18*(1). https://doi.org/10.1145/3104982

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. Computer Science Education, 29(2–3), 177–204. https://doi.org/10.1080/08993408.2019.1565233

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. C*omputer Science Education*, 26(4), 235–254. https://doi.org/10.1080/08993408.2016.1257418

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education, 10(*1), 3. https://doi.org/10.1186/s40594-023-00396-w

Ye, H., Ng, O.-L., & Cui, Z. (2023). Conceptualizing Flexibility in Programming-Based Mathematical Problem-Solving. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331231209773

Yi, S., & Lee, Y.-J. (2018). An Educational System Design to Support Learning Transfer From Block-Based Programming Language to Text-Based Programming Language. *International Journal on Advanced Science Engineering and Information Technology*. https://doi.org/10.18517/ijaseit.8.4-2.5735

Yuana, R. A., Faisal, M., Pangestu, D., & Putri, Y. R. L. (2015). Math thematic learning through the introduction of basic science-based programming games virtual robot for high school students. *Advanced Science Letters*, *21*(7), 2235–2238. https://doi.org/10.1166/asl.2015.6318

Zeng, Y., Yang, W., and Bautista, A. (2023). Teaching programming and computational thinking in early childhood education: a case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology*, 14, 1252718. https://doi.org/10.3389/fpsyg.2023.1252718