Uluslararası Avrasya Sosyal Bilimler Dergisi / *The International Journal of Eurasia Social Sciences*
Cilt/Volume: 16, Sayı/No: 59, Yıl/Year: 2025, ss. / pp.: 80-97
E-ISSN: 2146-1961
URL: https://rrpubs.com/index.php/ijoess

The impact of using various programming tools on pre-service teachers' self-efficacy beliefs and computational thinking skills

Ali Kürşat ERÜMİT[1], Sefa ÖZMEN[2], Hasan Yiğit CEBECİ[3]

[1]Trabzon University, Fatih Education Faculty, Trabzon, Turkiye
[2]Trabzon University, Çarşıbaşı Vocational School, Trabzon, Turkiye
[3]Trabzon University, Distance Education Application and Research Center, Trabzon, Turkiye

Araştırma Makalesi/*Research Article*		**DOI:** 10.70736/ijoess.540
Gönderi Tarihi/ *Received:*	Kabul Tarih/ *Accepted:*	Online Yayın Tarihi/ *Published*:
12.09.2024	09.02.2025	15.03.2025

Abstract

This study investigates the effects of different programming tools on the self-efficacy beliefs and computational thinking skills of pre-service elementary mathematics teachers. The growing integration of computational thinking in education necessitates research on how programming education influences teachers' confidence and problem-solving abilities, particularly in mathematics teaching. The study was conducted over 12 weeks with 47 university students enrolled in an Elementary Mathematics Teacher Education Program. A quasi-experimental research design was employed, with pre-test and post-test measurements to assess the impact of programming education. Participants were divided into two experimental groups: one using a block-based programming environment (Scratch) and the other using a text-based programming language (C#). The structured implementation of programming instruction allowed for a systematic comparison of how different approaches influence computational thinking and self-efficacy in teaching mathematics. The results indicate a strong positive correlation between computational thinking skills and self-efficacy beliefs in mathematics teaching. Both programming tools contributed significantly to the improvement of these skills, demonstrating that exposure to algorithmic thinking and coding enhances teachers' confidence in their ability to teach mathematics. However, no statistically significant difference was found between the groups, suggesting that both block-based and text-based programming approaches are effective in fostering computational thinking and self-efficacy. These results imply that regardless of the programming tool used, engaging in coding activities strengthens essential teaching competencies. These findings highlight the importance of integrating programming education into teacher training programs to prepare future educators for technology-enhanced teaching environments. Additionally, the study emphasizes the role of programming in developing higher-order thinking skills, such as problem-solving, logical reasoning, and adaptability.

Keywords: Computational thinking, mathematics teaching pre-service teachers, programming teaching, self-efficacy beliefs

Farklı programlama araçlarının öğretmen adaylarının öz-yeterlik inançları ve bilgi işlemsel düşünme becerileri üzerindeki etkisi

Öz

Bu çalışma, farklı programlama araçlarının sınıf öğretmeni adaylarının öz-yeterlik inançları ve bilgi işlemsel düşünme becerileri üzerindeki etkilerini incelemektedir. İlköğretim Matematik Öğretmenliği Programına kayıtlı 47 üniversite öğrencisiyle 12 hafta süren araştırmada, ön test-son test ölçümlerine dayalı yarı deneysel bir desen kullanılmıştır. Katılımcılar, blok tabanlı programlama (Scratch) ve metin tabanlı programlama (C#) kullanan iki deney grubuna ayrılmıştır. Araştırma, farklı programlama yaklaşımlarının bilgi işlemsel düşünme ve matematik öğretiminde öz-yeterlik üzerindeki etkilerini sistematik olarak karşılaştırmayı amaçlamaktadır. Bulgular, bilgi işlemsel düşünme becerileri ile öz-yeterlik inançları arasında güçlü bir pozitif ilişki olduğunu göstermektedir. Her iki programlama aracı da bu becerilerin gelişimine önemli katkı sağlarken, gruplar arasında anlamlı bir fark bulunmamıştır. Bu sonuçlar, programlama etkinliklerinin öğretmenlik için temel yetkinlikleri geliştirdiğini ortaya koymaktadır. Öğretmen yetiştirme programlarında programlama eğitiminin entegrasyonu, teknoloji destekli öğretim ortamlarına hazırlık açısından kritik öneme sahiptir. Ayrıca, programlamanın problem çözme, mantıksal akıl yürütme ve uyum sağlama gibi üst düzey düşünme becerilerinin gelişimine katkıda bulunduğu vurgulanmaktadır.

Anahtar Kelimeler: *Bilgi işlemsel düşünme, matematik öğretimi, öğretmen adayları, öz-yeterlik inançları, programlama öğretimi*

Sorumlu Yazar/ Corresponded Author: Ali Kürşat ERÜMİT, **E-posta/ e-mail:** kerumit@trabzon.edu.tr

INTRODUCTION

The concept of self-efficacy, first introduced by Bandura in 1977 as part of Social Learning Theory, plays a fundamental role in shaping individuals' belief in their ability to achieve desired outcomes through their own actions. In the context of education, self-efficacy is particularly significant for teachers, as it influences not only their teaching practices but also their motivation, resilience, and capacity to engage with students in meaningful ways (Swars et al., 2007). The impact of self-efficacy is especially critical for pre-service teachers, who are in the process of developing the skills and confidence necessary to manage classroom dynamics and facilitate effective learning. Moreover, with the growing emphasis on computational thinking and problem-solving skills in modern education, it is essential to explore how these competencies can interact with teachers' self-efficacy beliefs, particularly in the domain of mathematics instruction.

Recent research highlights the interconnectedness between self-efficacy, problem-solving skills, and computational thinking, especially in the training of future mathematics educators. As pre-service teachers are increasingly required to engage with programming and algorithmic thinking, it is crucial to understand how these skills contribute to their overall teaching efficacy. The ability to solve complex problems, think critically, and integrate technological tools into the classroom is essential not only for the development of teachers' instructional practices but also for fostering these skills in their students (Hu, 2023; Santos-Trigo, 2020). Against this backdrop, this study seeks to explore how programming education can influence both the self-efficacy beliefs and computational thinking skills of pre-service elementary mathematics teachers, thereby addressing a critical gap in current educational research.

Self-efficacy

The concept of self-efficacy, introduced by Bandura (1977) as part of Social Learning Theory, refers to an individual's belief in their ability to plan, organize, and successfully execute tasks to achieve specific objectives (Bandura & Wessels, 1994). In simpler terms, it represents confidence in one's ability to face challenges, solve problems, and accomplish goals. People with high self-efficacy are more likely to persist in difficult situations, as they believe in their capability to manage and overcome obstacles. Each time an individual successfully solves a problem, their confidence grows, enhancing their ability to deal with similar challenges in the future (Özdemir et al., 2018). Increasing awareness of one's capabilities through improved self-efficacy supports better decision-making in both professional and personal development contexts. The literature consistently demonstrates that self-efficacy plays a crucial

role in predicting success across various fields (Betz & Luzzo, 1996; Sarantseva, 2024; Shin, 2018).

In the field of education, self-efficacy is particularly important for teachers because it directly impacts their teaching practices, attitudes, and perception of their ability to interact effectively with students. Research highlights a strong relationship between teachers' self-efficacy and their instructional effectiveness. For instance, Swars et al. (2007) found that mathematics teachers with high self-efficacy beliefs are more open to adopting new teaching methods and innovations in their classrooms. Such teachers are also more inclined to modify their teaching strategies to better accommodate students' diverse learning needs, particularly those who struggle with mathematics concepts. Similarly, Klassen and Tze (2014) stated that self-efficacy has a significant effect on teachers' teaching performance. Furthermore, teachers with higher levels of self-efficacy are less likely to experience burnout or stress, as they feel more capable of managing classroom challenges and fostering a productive learning environment (Zuya et al., 2016). This positive sense of control and capability allows them to maintain motivation and resilience in their teaching practice.

For pre-service teachers, who are still in the process of developing their teaching skills and instructional approaches, self-efficacy is equally important. Their confidence in their ability to teach will shape not only how they plan and deliver lessons but also how they interact with students (Ma et al., 2022). Pre-service teachers with strong self-efficacy beliefs tend to engage more actively with their students, creating a more dynamic and interactive learning environment. These behaviours, in turn, influence the students' attitudes towards learning, especially in subjects like mathematics, where teacher support and encouragement are critical (Göloğlu-Demir, 2011). From the learners' perspective, higher levels of self-efficacy in teachers have been linked to better academic outcomes (Shin, 2018). Therefore, boosting the self-efficacy of pre-service mathematics teachers is an essential factor in enhancing their future success in the classroom (Ampofo, 2019).

Problem solving skills

Problem-solving is a central skill in mathematics teaching and a fundamental aspect of developing mathematical literacy (Santos-Trigo, 2020). It involves the ability to analyse and approach problems systematically, using logical reasoning and mathematical concepts. In Turkey, pre-service elementary mathematics teachers are required to take the "Algorithm and Programming" course, a crucial part of their curriculum. This course is designed to equip future

teachers with essential problem-solving skills that they will later apply in their teaching practice. The course covers fundamental topics such as defining problems, developing algorithms, problem-solving strategies, and basic programming concepts (YÖK, 2018).

Research highlights that algorithm and programming teaching not only develops students' problem-solving abilities but also fosters critical 21st-century skills such as collaboration, communication, critical thinking, and digital literacy (Hu, 2023; Sayın & Seferoğlu, 2016). These competencies are increasingly important in today's education landscape, where teachers must prepare students for a rapidly evolving, technology-driven world. By teaching future mathematics teachers how to approach complex problems using algorithms, programming teaching helps them develop the cognitive flexibility and adaptability needed to address a variety of mathematical challenges in their future classrooms. This, in turn, enhances their ability to foster problem-solving skills in their students (De-La-Pena et al., 2021).

Computational thinking

Computational thinking is a critical skill in the 21st century and refers to the mental processes involved in formulating problems and solutions in a way that a computer could execute (Thomas et al., 2015). It includes competencies such as problem decomposition, pattern recognition, abstraction, and algorithmic thinking (ISTE, 2022). While computational thinking originates from the field of computer science, it has become increasingly recognized as a valuable skill across a wide range of disciplines. Lye and Koh (2014) argue that computational thinking is not just for computer scientists but is a form of logical thinking that everyone should acquire to tackle complex problems efficiently (Sırakaya, 2019). This skill is particularly relevant in education, where teachers are expected to help students develop higher-order thinking skills that transcend traditional academic boundaries. Incorporating computational thinking into teacher education programs ensures that pre-service teachers are prepared to integrate these approaches into their future classrooms, thereby fostering students' critical thinking and problem-solving abilities (Peracaula-Bosch & González-Martínez, 2022).

Programming teaching

Programming teaching offers numerous cognitive, academic, and practical benefits. However, it is often perceived as challenging, especially for beginners, due to its abstract nature and the difficulty of learning programming languages (Erdem, 2018, Krpan et al., 2015). To overcome these challenges, block-based programming environments such as Scratch, Mblock, MakeCode, and Code.org have been developed and provide a more user-friendly introduction to programming. These platforms simplify the learning process by allowing students to use

visual blocks instead of writing complex text-based code, making abstract programming concepts more accessible (Medlock-Walton et al., 2014). Research shows that block-based environments not only make programming easier for beginners but also improve problem-solving skills and enhance students' motivation and attitudes towards learning programming (Lai & Yang, 2011; Wang et al., 2014).

However, despite the advantages of block-based programming, it has limitations. One of the main criticisms is that it does not provide enough opportunities for students to develop more advanced programming skills, as it lacks the complexity of text-based coding languages (Puente, 2022). When transitioning from block-based platforms to text-based programming environments like Python, Java, or C++, students often struggle to adapt due to the increased difficulty and abstract thinking required (Gomes & Mendes, 2007; Yükseltürk & Curaoğlu, 2019). Text-based programming, on the other hand, offers students the chance to tackle more sophisticated programming tasks and helps develop the flexibility and problem-solving skills required for real-world software development (Kandemir, 2018). Programming environments that involve textual coding also encourage learning through trial and error, fostering resilience and adaptability in learners (Weintrop & Wilensky, 2017).

Considering the significance of self-efficacy in teaching (Martin & Mulvihill, 2019) and its relationship with computational thinking and problem-solving skills, the key question arises: How can programming teaching be structured to not only improve pre-service teachers' computational thinking skills but also enhance their self-efficacy beliefs regarding mathematics teaching? Furthermore, is there a direct relationship between pre-service teachers' self-efficacy beliefs and their computational thinking skills? This study aims to address these questions by investigating the effects of programming teaching using various tools on the self-efficacy beliefs and computational thinking skills of pre-service elementary mathematics teachers. Within the framework of this question, the research problems were determined as follows.

1. "Is there a relationship between the computational thinking skills of pre-service elementary mathematics teachers and their self-efficacy beliefs towards mathematics teaching?"
2. "What are the effects of programming teaching with different programming tools on pre-service elementary mathematics teachers' self-efficacy beliefs towards mathematics teaching?"

3. "What are the effects of programming teaching with different programming tools on pre-service elementary mathematics teachers' computational thinking skills?"

METHOD

Research model

The objective of this study was to evaluate the effects of a standardized training program on two groups, each assigned different programming tools. To ensure comparability, a pre-test was administered to both groups before the intervention, and changes were assessed both within and across the groups. This research followed a quasi-experimental design, which included pre-test and post-test measurements to collect quantitative data. In quasi-experimental research, causal relationships between variables are explored by controlling certain variables and observing their effects on others (Reichardt, 2009).

Study group

This study involved 47 second-year students from the Elementary Mathematics Teaching Program at the Faculty of Education of a state university in Türkiye. All participants were enrolled in the "Algorithm and Programming" course. The students were divided into two sections, labelled A and B. Section A originally had 39 students, while section B had 31. However, by the conclusion of the study, only 27 students from section A and 20 from section B had completed the necessary data collection instruments. The students were randomly placed into two experimental groups for the research. Demographic information for the students in sections A and B is presented in Table 1.

Table 1. Demographic information of participants

Feature		f_a (N=27)	f_b (N=20)
Gender	Man	3	2
	Woman	24	18
Computer use competencies	Very good	1	0
	Good	7	3
	Middle	13	14
	Bad	6	2
	Very Bad	0	1
GPA (Grade point averages)	2.00-2.50	1	0
	2.51-3.00	6	4
	3.01-3.50	16	13
	3.51-4.00	4	3

When Table 1 is analysed, students in both classes are concentrated at the middle level. Looking at the GPA, it can be stated that both classes have equivalent achievement levels.

Data collection tools

In this study, two primary tools were employed for data collection: the Self-Efficacy Beliefs Towards Mathematics Teaching Questionnaire and the Computational Thinking Scale. "Ethics committee permission was obtained from Trabzon University Social and Human Sciences Scientific Research and Publication Ethics Board with the decision dated 26.06.2024 and number E-81614018-050.04-2400029261."

Self-efficacy beliefs towards mathematics teaching questionnaire

This questionnaire was developed by Göloğlu-Demir (2011). It is structured as a 5-point Likert scale and assesses self-efficacy in teaching mathematics across three factors: "effort-based self-efficacy belief, academic background competence belief, and ability to use academic background belief". The instrument includes 19 items, of which 7 are negatively worded, and 12 are positively worded. Items are rated from "Strongly agree" to "Strongly disagree". The reliability of the scale was confirmed through a sample of 286 participants, yielding a Cronbach's Alpha value of 0.88, with sub-factor internal consistency scores of 0.82, 0.78, and 0.81, respectively.

Computational thinking scale

Developed by Korkmaz, et al., (2017), the Computational Thinking Scale consists of 29 items, which are grouped into five factors: "creativity, algorithmic thinking, collaboration, critical thinking, and problem-solving". This tool, too, uses a 5-point Likert scale ranging from "Strongly agree" to "Strongly disagree". The scale was validated with a sample of 1306 participants, demonstrating an overall reliability (Cronbach's Alpha) of 0.82, while the sub-factors had internal consistency coefficients of 0.84, 0.86, 0.88, 0.78, and 0.72, respectively.

Data analysis

In this study, the two experimental groups were compared to examine the impact of varying programming environments on their self-efficacy beliefs related to mathematics teaching and their computational thinking skills. To ensure that the groups were comparable before the intervention, participants completed the "Self-Efficacy Beliefs Towards Mathematics Teaching Questionnaire" and the "Computational Thinking Scale." The data were evaluated to confirm normal distribution. Initial group comparisons were made using an independent samples t-test.

Post-intervention, a two-factor ANOVA for mixed measures was performed to analyse differences in pre-test and post-test scores, both within each group and between the groups. In

this study, since multiple factors were analysed simultaneously and both their main effects and interactions were evaluated, a two-factor ANOVA test was used for mixed measurements instead of a dependent groups t-test. The detailed results of these analyses are provided in the findings section.

Implementation process

The implementation took place during the "Algorithm and Programming" course, which is part of an undergraduate elementary mathematics teaching program at a state university in Türkiye. Initially, the mathematics curriculum of the Turkish Ministry of National Education (MEB, 2018) was reviewed to identify programming-related learning outcomes. These objectives were aligned with the course content to design appropriate activities. The process spanned 12 weeks, with details on the activities, learning outcomes, and tools used each week outlined in Table 2.

The main focus of the course was to teach essential concepts such as algorithm development, programming structures, problem-solving, and the basic components of programming. The course covered topics like algorithm design (including flowcharts, variables, decision structures, loops, and arrays) and the coding of designed algorithms.

The course took place over 12 weeks, with two weekly class hours. Both groups of students used computers in a lab setting. In the first week, the Self-Efficacy Beliefs Towards Mathematics Teaching Questionnaire and the Computational Thinking Scale were administered as pre-tests. These tests were repeated at the end of the course as post-tests to assess the outcomes quantitatively.

Table 2. Implementation process

Week	Subject	Lesson Process	
	Collection of pre-test data		
1	Introduction of Flow Diagram Algorithm and flowchart applications	Paper, Pencil activities	
2	Input-output concepts Algorithm and flowchart applications	Paper, Pencil activities	
3	Loops Algorithm and flowchart applications	Paper, Pencil activities	
4	Decision structures Algorithm and flowchart applications	Paper, Pencil activities	
5	Decision making and development of sample algorithms suitable for loop problems	Paper, Pencil activities	
6	Decision making and development of sample algorithms suitable for loop problems	Paper, Pencil activities	
7	Introducing the Programming Environment Coding of the created algorithms and applications	C# applications	Scratch applications

Week	Subject	Lesson Process	
8	Coding of the created algorithms and applications	C# applications	Scratch applications
9	Coding of the created algorithms and applications	C# applications	Scratch applications
10	Coding of the created algorithms and applications	C# applications	Scratch applications
11	Coding of the created algorithms and applications	C# applications	Scratch applications
12	Collection of post-test data		

The implementation process designed for the sample group commenced with the collection of pre-test data, followed by explanations of flowcharts and the interpretation of symbols used in diagrams. Flowchart examples were developed, such as "finding the average of two entered numbers" and "indicating the success status of the calculated average," aimed at enhancing the sample group's understanding of symbol usage in flowcharts and their relevance to their respective fields. While preparing the activities for the course, learning objectives from the Mathematics Curriculum of The Turkish Ministry of National Education (MEB, 2018) were used as a guide, and objective numbers were assigned to the corresponding activities. In the second week, flowchart examples were created for tasks like "finding the largest number among 10 entered numbers" and "calculating the sum of odd numbers up to a specified number." The third week introduced flowchart examples for tasks such as "calculating the factorial of an entered number" and "determining whether an entered number is prime." Fourth-week examples included flowcharts for tasks such as "displaying the current hour, minute, and second values on screen upon program initiation" and "outputting the multiplication table to the screen." In the fifth week, flowchart examples were developed for programs like "calculating and displaying the average of a person's name and two scores until the 'H' key is pressed, then displaying the class average" and "calculating and displaying the sum of digits and number of digits of an entered number." The final week's activities, conducted with paper and pencil, involved creating flowcharts for programs such as "performing division without using the division operation" and "determining whether three entered numbers can form a triangle." After this 6-week period, which was conducted using paper and pencil, a midterm exam was held. Before the midterm exam, both groups were introduced to flow diagrams and worked on similar activities. However, after the midterm, the groups diverged in terms of tools: Group A used the C# text-based programming language. Group B used the Scratch block-based programming tool. Although both groups completed the same activities, the tools they used differed after the midterm exam. Transitioning to computer-based activities after this six-week process, sample groups were introduced to programming environments (C# or Scratch) they would use

throughout the process. The activities conducted throughout the process were consistent with those developed through paper and pencil flowchart exercises.

FINDINGS

Before moving on to the quantitative findings, the normality of the distribution between groups was evaluated. A normality test was applied to pre-test scores for both groups on the computational thinking and self-efficacy beliefs towards mathematics teaching scales. These results are displayed in Table 3.

Table 3. Normality results of groups

	Groups	Shapiro-Wilk		
		Statistic	d.f.	p
Computational Thinking	Group A	0.969	27	0.570
	Group B	0.920	20	0.097
Self-efficacy	Group A	0.953	27	0.256
	Group B	0.971	20	0.782

Upon examining normal distribution of groups scores, the Shapiro-Wilk test results for the pre-test scores indicate that both groups demonstrated a normal distribution ($p > 0.05$) for the self-efficacy beliefs related to computational thinking and mathematics teaching. An independent samples t-test was then performed to examine if there was a notable difference in the pre-test scores of the groups, which were normally distributed, on the self-efficacy beliefs scale regarding computational thinking and mathematics teaching before the teaching process began. The results are summarized in Table 4.

Table 4. Pre-test independent sample t test

Scale	Groups	N	X	S.d.	d.f.	t	p
Computational Thinking	Group A	27	110.67	11.576	45	-0.399	0.692
	Group B	20	112.05	11.998			
Self-efficacy	Group A	27	73.25	8.877	45	-0.745	0.460
	Group B	20	75.05	7.037			

The comparison between the C# group and the Scratch group revealed no statistically significant difference in the mean pre-test scores for computational thinking and self-efficacy beliefs [$p>0.05$]. This suggests that both groups had similar levels at the start of the study.

The relationship between students' self-efficacy beliefs towards mathematics teaching and computational thinking skills

A Pearson correlation test was conducted to investigate the association between the pre-test scores for self-efficacy beliefs in mathematics teaching and computational thinking skills among pre-service elementary mathematics teachers. These results are presented in Table 5.

Table 5. The relationship between pre-service teachers' self-efficacy beliefs towards mathematics teaching and computational thinking skills

	N	Pearson correlation	p
Self-efficacy - Computational thinking	47	0.608	0.001

According to Table 5, there is a moderate, positive, and statistically significant relationship between pre-test scores of self-efficacy beliefs towards mathematics teaching and computational thinking skills ($r=0.608$, $p<0.05$). According to this finding, it can be said that there is a significant and positive relationship between students' self-efficacy beliefs in teaching mathematics and their computational thinking skills.

Self-efficacy beliefs towards mathematics teaching

A "Two-Factor ANOVA Test for Mixed Measures" was applied to determine whether the pre-test and post-test score changes of the self-efficacy beliefs scale for teaching mathematics showed a significant difference from each other. The findings are shown in Table 6 and Figure 1.

Table 6. Results of the two-factor Anova test for mixed measures of self-efficacy beliefs scale for mathematics teaching

Source of Variance	Sum of squares	d.f.	MS	F	p	η^2
Between Groups	5354,106	46				
Group	4,025	1	4.025	0.034	0.885	0.001
Error	5350,081	45	118.891			
Within Groups	1059,375	47				
Factor	98,673	1	98.673	4.840	0.152	0.045
Group*Factor	43,269	1	43.269	2.122	0.033	0.097
Error	917,433	45	20.387			
Total	6413,481	93				

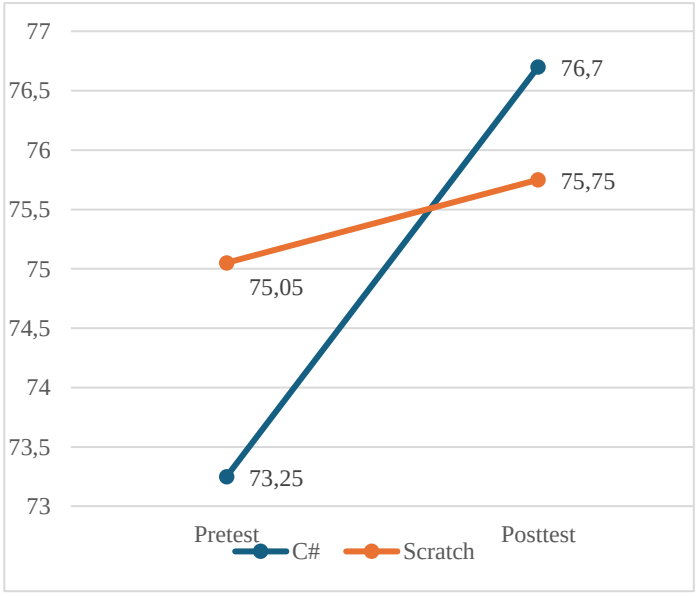

Figure 1. Anova test plot of self-efficacy beliefs scale for mathematics teaching

Table 6 and Figure 1 illustrate the results of the Two-Factor ANOVA Test for Mixed Measures, which was performed to determine if there were any differences in self-efficacy beliefs related to mathematics teaching between different programming environments. The pre-test and post-test results revealed a significant difference within the groups [$F(1.45) = 2.12$, $p < 0.05$, $\eta^2 = 0.097$], but no significant difference was observed between the groups [$F(1,45) = 0.034$, $p > 0.05$, $\eta^2 = 0.001$]. These findings suggest that although there was a significant change within each group over time, the differences between the two groups were not significant. The effect size ($\eta^2 = 0.097$) indicates that this observed change was small.

Computational thinking skills

A "Two-Factor ANOVA Test for Mixed Measures" was also carried out to examine the changes in pre-test and post-test scores for computational thinking skills across the groups. The results are presented in Table 7 and Figure 2.

Table 7. Anova test results of the computational thinking scale

Source of Variance	Sum of Squares	d.f.	M.S.	F	p	η^2
Between Groups	1624,002	46				
Group	0.102	1	0.102	0.000	0.984	0.991
Error	11031.600	45	245.147			
Within Groups	1961.802	47				
Factor	289.589	1	289.589	8.025	0.253	0.029
Group*Factor	48.313	1	48.313	1.339	0.007	0.151
Error	1623.900	45	36.087			
Total	3585.804	93				

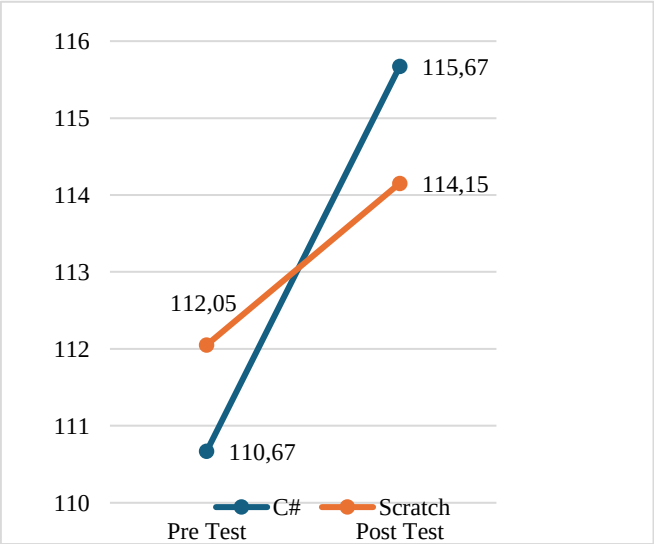

Figure 2. Anova test plot of the computational thinking scale

Upon analyzing Table 7 and Figure 2, it is observed that the mean computational thinking score for the C# group was 110.67, while the Scratch group had a mean score of 112.05. In the post-test, the C# group's score increased to 115.67, and the Scratch group's score was 114.15. According to Table 7 the results of the Two-Factor ANOVA Test for Mixed Measures on the computational thinking scale showed that the pre-test and post-test scores demonstrated a significant difference within the groups [$F(1.45) = 1.33$, $p < 0.05$, $\eta^2 = 0.151$], but no significant difference between the groups [$F(1.45) = 0.00$, $p > 0.05$, $\eta^2 = 0.991$]. These findings indicate that while there was a significant change within the groups over the course of the study, the differences between the groups were not statistically significant. The effect size ($\eta^2 = 0.151$) suggests that the observed difference within the groups was of small magnitude.

DISCUSSION AND CONCLUSION

This study aimed to investigate the relationship between pre-service elementary mathematics teachers' computational thinking skills and their self-efficacy beliefs regarding mathematics teaching, as well as the impact of different programming tools (block-based and text-based) on these domains. The findings offer valuable insights into how programming teaching can contribute to the enhancement of both computational thinking and teaching self-efficacy in the field of mathematics.

The first research question examined whether there exists a relationship between pre-service teachers' computational thinking skills and their self-efficacy beliefs in mathematics teaching. The results show that developing computational thinking skills through programming instruction affects self-efficacy beliefs in teaching mathematics. This aligns with previous literature suggesting that computational thinking, which involves applying logical processes to solve complex problems, can enhance individuals' self-confidence in teaching mathematical concepts (Rich et al., 2021; Román-González et al., 2018; Zhao et al., 2020). Given that computational thinking skills are applicable to mathematical problem-solving, it follows that strengthening these skills would also bolster pre-service teachers' beliefs in their capacity to effectively teach mathematics. As earlier research has shown, self-efficacy in mathematics teaching plays a crucial role in shaping teachers' perceptions of their competence (Bandura & Wessels, 1994; Jaipal-Jamani & Angeli, 2017).

The second research question explored the influence of different programming tools (block-based and text-based) on pre-service teachers' self-efficacy beliefs related to mathematics teaching. The findings revealed a substantial increase in self-efficacy for teaching

mathematics among participants who received algorithm and programming training, irrespective of the programming tool utilized. This corroborates earlier studies that demonstrated how integrating programming into the mathematics curriculum positively affects students' self-efficacy (Jiang et al., 2022; Psycharis & Kallia, 2017). The relationship between coding and mathematics seems to be mutually reinforcing; as students enhance their coding skills, their interest and confidence in mathematics also grow, and vice versa. The present study reinforces this connection by showing that both block-based and text-based programming tools can positively impact teaching self-efficacy.

The third research question focused on the effect of different programming tools on pre-service teachers' computational thinking skills. The results showed a significant improvement in computational thinking skills following programming teaching, regardless of whether block-based or text-based tools were used. This finding aligns with previous research, which has consistently demonstrated the positive impact of programming teaching on the development of computational thinking (Hsu et al., 2018; Seow et al., 2017; Subramaniam et al., 2022). Both block-based and text-based programming approaches foster the development of logical thinking, problem-solving, and creativity, which are fundamental components of computational thinking.

In conclusion, this study demonstrates that programming teaching, whether block-based or text-based, significantly enhances pre-service teachers' computational thinking skills and self-efficacy beliefs towards mathematics teaching. Integrating programming into mathematics teacher education programs not only strengthens their technical and problem-solving skills but also boosts their confidence and motivation for teaching mathematics. These findings suggest that programming should be a vital component of teacher preparation programs to better equip future educators for the demands of contemporary mathematics education.

Recommendations

It is recommended that algorithm and programming training for pre-service teachers outside the field of computer science begin with paper and pencil activities, followed by the introduction of a programming language. At this stage, participants may be encouraged to solve programming problems using pseudo-code rather than working directly in a programming environment. Additionally, aligning the programming teaching activities with the specific subject areas of pre-service teachers is expected to support their professional development and mitigate the perceived difficulty of programming. In this study, C# was employed as a text-

based programming language. However, the Python programming language, which has recently gained popularity and offers a wide range of libraries related to mathematics, is considered more suitable for sample groups with a focus on mathematics. Hence, researchers aiming to work in this area may opt to use Python.

REFERENCES

Ampofo, C. (2019). Relationship between pre-service teachers' mathematics self-efficacy and their mathematics achievement. *African Journal of Educational Studies in Mathematics and Sciences*. https://doi.org/10.4314/ajesms.v15i1.3

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*. https://doi.org/10.1016/0146-6402(78)90002-4.

Bandura, A., & Wessels, S. (1994). Self-efficacy. In V. S. Ramachaudran (Ed.), *Encyclopedia of human behavior, 4,* 71-81. New York: Academic Press.

Betz, N., & Luzzo, D. (1996). Career assessment and the career decision-making self-efficacy scale. *Journal of Career Assessment*, *4*, 413 - 428. https://doi.org/10.1177/106907279600400405

De-La-Peña, C., Fernádez-Cézar, R., & Solano-Pinto, N. (2021). Attitude toward mathematics of future teachers: How important are creativity and cognitive flexibility?. *Frontiers in Psychology, 12*. https://doi.org/10.3389/fpsyg.2021.713941

Erdem, E. (2018). *The investigation of different teaching strategies during teaching programming process in block based environment in terms of different factors.* (Unpublished master's thesis). Başkent University, Institute of Educational Sciences, Ankara.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. *In International Conference on Engineering Education–ICEE*, *7*.

Göloğlu-Demir, C. (2011). *Analysis of self-efficacy beliefs and attitudes of students at the department of elementary mathematics education regarding mathematics education.* (Unpublished master's thesis). Gazi University, Ankara.

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 literature. *Computers & Education, 126,* 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Hu, L. (2023). Programming and 21st century skill development in K-12 schools: A multidimensional meta-analysis. *Journal of Computer Assisted Learning*, *40*, 610-636. https://doi.org/10.1111/jcal.12904

ISTE (2022). *Computational Thinking Competencies*. ISTE. https://iste.org/standards/computational-thinking-competencies

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology*, *26*, 175-192. https://doi.org/10.1007/s10956-016-9663-z

Jiang, H., Turnbull, D., Wang, X., Chugh, R., Dou, Y., & Chen, S. (2022). How do mathematics interest and self-efficacy influence coding interest and self-efficacy? A structural equation modeling analysis. *International Journal of Educational Research, 115,* 102058. https://doi.org/10.1016/j.ijer.2022.102058

Kandemir, C. M. (2018). *Text-based programming.* Pegem Citation Index, 267-294.

Klassen, R., & Tze, V. (2014). Teachers' self-efficacy, personality, and teaching effectiveness: A meta-analysis. *Educational Research Review*, 12, 59-76. https://doi.org/10.1016/J.EDUREV.2014.06.001

Korkmaz, Ö., Çakir, R., & Özden, MY (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72*, 558-569. https://doi.org/10.1016/j.chb.2017.01.005

Krpan, D., Mladenović, S., & Rosić, M. (2015). Undergraduate programming courses, students' perception and success. *Procedia-Social and Behavioral Sciences, 174*, 3868-3872. https://doi.org/10.1016/j.sbspro.2015.01.1126

Lai, A. F., & Yang, S. M. (2011, September). The learning effect of visualized programming learning on 6 th graders' problem solving and logical reasoning abilities. In *2011 international conference on electrical and control engineering* (pp. 6940-6944). IEEE. https://doi.org/10.1109/ICECENG.2011.6056908

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior, 41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Ma, K., McMaugh, A., & Cavanagh, M. (2022). Changes in pre-service teacher self-efficacy for teaching in relation to professional experience placements. *Australian Journal of Education*, *66*(1), 57-72. https://doi.org/10.1177/00049441211060474

Martin, L. E., & Mulvihill, T. M. (2019). Voices in education: Teacher self-efficacy in education. *The Teacher Educator, 54*(3), 195-205. https://doi.org/10.1080/08878730.2019.1615030

MEB (2018). *Mathematics curriculum of the turkish ministry of national education*. Turkish Ministry of National Education. https://mufredat.meb.gov.tr/Dosyalar/201813017165445-MATEMATİK%20ÖĞRETİM%20PROGRAMI%202018v.pdf

Medlock-Walton, P., Harms, K., Kraemer, E., Brennan, K., & Wendel, D. (2014). Blocks-based programming languages: simplifying programming for different audiences with different goals. Proceedings of the 45th ACM technical symposium on Computer science education. https://doi.org/10.1145/2538862.2538873

Özdemir, A., Yaman, C., & Vural, R. A. (2018). Development of the teacher self-efficacy scale for STEM practices: A validity and reliability study. *Adnan Menderes University Journal of Social Sciences Institute*, *5*(2), 93-104. https://doi.org/10.30803/adusobed.427718

Peracaula-Bosch, M., & González-Martínez, J. (2022). Developing computational thinking among preservice teachers. *Qwerty - Open and Interdisciplinary Journal of Technology, Culture and Education*, *57* https://doi.org/10.30557/qw000049

Psycharis, S., Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*, 583–602. https://doi.org/10.1007/s11251-017-9421-5

Puente, E. (2022). Effect of the use of block-based languages in programming learning. 2022 International Symposium on Computers in Education (SIIE), 1-6. https://doi.org/10.1109/SIIE56031.2022.9982348

Reichardt, C. S. (2009). Quasi-experimental design. *The SAGE handbook of quantitative methods in psychology*, 46-71.

Rich, P.J., Mason, S.L., & O'Leary, J. (2021). Measuring the impact of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers & Education, 168*, 104196. https://doi.org/10.1016/j.compedu.2021.104196

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the computational thinking test. *International Journal of Child-Computer Interaction, 18*, 47-58. https://doi.org/10.1016/j.ijcci.2018.06.004

Santos-Trigo, M. (2020). Problem-solving in mathematics education. *Encyclopedia of mathematics education*, 686-693. https://doi.org/10.1007/978-3-030-15789-0_129

Sarantseva, S. (2024). Self-Efficacy in the context of professional activity. *Obshchenauchnyy zhurnal (General Science Academic Journal)*. https://doi.org/10.33920/nik-04-2401-07

Sayın, Z., & Seferoğlu, S. S. (2016). Yeni bir 21. yüzyıl becerisi olarak kodlama eğitimi ve kodlamanın eğitim politikalarına etkisi. *Akademik Bilişim Konferansı, 3*(5), 1-13.

Seow, P., Looi, C.K., Wadhwa, B., Wu, L., Liu, L., Kong, S.C., & Li, K. (2017). Computational thinking and coding initiatives in Singapore. In Conference Proceedings of International Conference on Computational Thinking Education (pp. 164-167).

Shin, M. H. (2018). Effects of Project-Based Learning on Students' Motivation and Self-Efficacy. *English teaching, 73*(1), 95-114. https://doi.org/10.15858/engtea.73.1.201803.95

Sırakaya, D. A. (2019). Programlama öğretiminin bilgi işlemsel düşünme becerisine etkisi. Türkiye Sosyal Araştırmalar Dergisi, 23(2), 575-590.

Subramaniam, S., Maat, S. M., & Mahmud, M. S. (2022). Computational thinking in mathematics education: A systematic review. *Cypriot Journal of Educational Sciences, 17*(6), 2029-2044. http://dx.doi.org/10.18844/cjes.v17i6.7494

Swars, S., Hart, L.C., Smith, SZ, Smith, M.E., & Tolar, T. (2007). A longitudinal study of elementary pre-service teachers' mathematics beliefs and content knowledge. *School science and mathematics, 107*(8), 325-335. http://dx.doi.org/10.1111/j.1949-8594.2007.tb17797.x

Thomas,J., Odemwingie, O.C., Saunders, Q. & Watlerd, M. (2015). Understanding the difficulties African-American middle school girls face while engaging computational algorithmic thinking in the context of game design. *Journal of Computer Science and Information Technology, 3*(1), 15-33. http://dx.doi.org/10.15640/jcsit.v3n1a2

Wang, H. Y., Huang, I., & Hwang, G. J. (2014). Effects of an integrated scratch and project-based learning approach on the learning achievements of gifted students in computer courses. In *Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on (pp. 382-387). IEEE.* https://doi.org/10.1109/IIAI-AAI.2014.85

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/10.1145/3089799

YÖK. (2018). *New teacher training undergraduate programs, primary mathematics teaching undergraduate program.* https://www.yok.gov.tr/Documents/Kurumsal/egitim_ogretim_dairesi/Yeni-Ogretmen-Yetistirme-Lisans-Programlari/Ilkogretim_Matematik_Lisans_Programi.pdf

Zhao, M., Zhao, M., Wang, X.H., & Ma, H.L. (2020, December). Promoting teaching self-efficacy in computational thinking of school teachers. In *2020 Ninth International Conference of Educational Innovation through Technology (EITT)* (pp. 235-239). IEEE. https://doi.org/10.1109/EITT50754.2020.00048

Zuya, HE, Kwalat, SK, & Attah, BG (2016). Pre-service teachers' mathematics self-efficacy and mathematics teaching self-efficacy. *Journal of Education and Practice, 7*(14), 93-98.

KATKI ORANI ***CONTRIBUTION RATE***	**AÇIKLAMA** ***EXPLANATION***	**KATKIDA BULUNANLAR** ***CONTRIBUTORS***
Fikir ve Kavramsal Örgü *Idea or Notion*	Araştırma hipotezini veya fikrini oluşturmak *Form the research hypothesis or idea*	Ali Kürşat ERÜMİT
Tasarım *Design*	Yöntem ve araştırma desenini tasarlamak *To design the method and research design.*	Ali Kürşat ERÜMİT
Literatür Tarama *Literature Review*	Çalışma için gerekli literatürü taramak *Review the literature required for the study*	Sefa ÖZMEN Hasan Yiğit CEBECİ
Veri Toplama ve İşleme *Data Collecting and Processing*	Verileri toplamak, düzenlemek ve raporlaştırmak *Collecting, organizing and reporting data*	Ali Kürşat ERÜMİT Sefa ÖZMEN Hasan Yiğit CEBECİ
Tartışma ve Yorum *Discussion and Commentary*	Elde edilen bulguların değerlendirilmesi *Evaluation of the obtained finding*	Sefa ÖZMEN Hasan Yiğit CEBECİ

Destek ve Teşekkür Beyanı/ *Statement of Support and Acknowledgment*

Bu çalışmanın yazım sürecinde herhangi bir katkı ve/veya destek alınmamıştır.

No contribution and/or support was received during the writing process of this study.

Çatışma Beyanı/ *Statement of Conflict*

Araştırmacıların araştırma ile ilgili diğer kişi ve kurumlarla herhangi bir kişisel veya mali çıkar çatışması bulunmamaktadır.

Researchers do not have any personal or financial conflicts of interest with other people and institutions related to the research.

Etik Kurul Beyanı/ *Statement of Ethics Committee*

Bu araştırma Trabzon Üniversitesi Etik Kurulu'nun 24.06.2024 tarih ve E-81614018-050.04-2400029261 sayılı kararı ile gerçekleştirilmiştir.

This research was conducted with the decision of Trabzon University Ethics Committee dated 24.06.2024 and numbered E-81614018-050.04-2400029261.