IJETS@R

(ISSN: 2587-0238)

Çalışkan, E. (2026). Effects of Instructional Coding Strategies on Elementary and Middle-School Students' Computational Thinking, *International Journal of Education Technology and Scientific Researches, 11(32), 110-130.*

DOI: http://dx.doi.org/10.35826/ijetsar.798

Article Type: Research Article

EFFECTS OF INSTRUCTIONAL CODING STRATEGIES ON ELEMENTARY AND MIDDLE-SCHOOL STUDENTS' COMPUTATIONAL THINKING

Erkan ÇALIŞKAN

Assoc. Prof. Dr., Niğde Ömer Halisdemir University, Niğde, Türkiye, erkancaliskan@ohu.edu.tr

ORCID: 0000-0002-2309-1406

Received: 10.12.2025 Accepted: 16.02.2026 Published: 15.03.2026

ABSTRACT

Coding education plays a critical role in preparing students for participation in an increasingly digital and technology-driven society. Beyond teaching technical skills, coding instruction supports the development of higher-order cognitive abilities such as problem solving, logical reasoning, abstraction, and algorithmic thinking. These competencies are generally conceptualized within the broader framework of computational thinking (CT), which has become a central objective of contemporary computer science and digital literacy education. To foster CT skills among young learners, educators use various instructional approaches that differ in pedagogical structure, level of technology use, and learning environments. Among the most implemented approaches are block-based programming environments, unplugged coding activities that teach computational concepts without computers, and educational robotics that combines programming with physical devices. This study examined the effects of three widely used coding instructional strategies—block-based coding, unplugged coding, and educational robotics—on students' computational thinking skills. A 2×3 factorial model was tested in which grade level (elementary and middle school) and instructional approach served as the independent variables. The research was conducted using a ten-week quasi-experimental non-equivalent groups pretest–posttest design implemented as an after-school program. In total, 123 students participated in the study. To assess students' computational thinking skills, a study-specific multiple-choice assessment instrument was developed and psychometrically validated. Reliability analysis indicated very high internal consistency (KR-20 = 0.969). The findings showed that all three instructional approaches produced statistically significant improvements in students' CT scores from pretest to posttest across both grade levels. Although elementary school students initially had lower CT scores than middle school students, the instructional interventions reduced this difference over time. After the intervention, both grade levels demonstrated similar levels of improvement. Furthermore, when pretest scores were statistically controlled, no significant interaction was found between grade level and instructional approach on posttest CT scores. Overall, the results suggest that unplugged activities, block-based programming, and educational robotics can each effectively support the development of computational thinking when implemented with a structured curriculum and consistent instructional design.

Keywords: Block-based coding, computational thinking, educational robotics, skill test, unplugged coding.

Corresponded Author: Assoc. Prof. Dr., Niğde Ömer Halisdemir University, erkancaliskan@ohu.edu.tr

Ethics Committee Approval: 10.02.2021 dated and 2021/03-12 numbered approval was obtained from Niğde Ömer Halisdemir University Ethics Committee to conduct the study.

Plagiarism/Ethics: This article has been reviewed by at least two referees and has been confirmed to comply with research and publication ethics, containing no plagiarism.

INTRODUCTION

Given the rapid advancements in information technology and software and their impact on daily life, it is crucial for students to possess the skills to utilize emerging technologies. This will prepare them to solve future, currently unknown problems (Durak & Sarıtepeci, 2018). Furthermore, as stated in the 2016 World Economic Forum report, many students will pursue careers that do not currently exist. Computer programming is now widely acknowledged as a new form of literacy, commonly referred to as 'code literacy' (Prensky, 2008; Rushkoff, 2012), because software now underpins many everyday technologies (Román-González et al., 2018). As technology becomes increasingly integrated into daily life, programming is becoming an important skill in many professional fields. In this sense, programming can be viewed as a "language" that enables individuals to communicate with digital systems and express computational ideas, forming a component of digital literacy (Nolan, 2021). The Computer Science Teachers Association (CSTA, 2003) suggests that to fully benefit from computational activities and computer programming, educational systems should integrate coding and programming across curricula at all grade levels. An increasing number of countries are incorporating coding instruction into their education programs and strategic goals to enhance their students' computer programming and coding skills (Sayın & Seferoğlu, 2016). Competitions such as coding Olympiads in the United States provide students with opportunities to develop coding and problem-solving skills while raising public awareness of computer science education (Sayın & Seferoğlu, 2016; USA Computing Olympiad, 2019). Robotics and programming are introduced to children as young as five years old in England (Demirer & Sak, 2016). Numerous coding and robotics events and competitions are organized in Turkey by schools, universities, and institutions affiliated with the Ministry of National Education and other public organizations. For example, national initiatives such as TEKNOFEST technology competitions and robotics and coding project festivals organized within the DENEYAP Technology Workshops bring together thousands of students across the country. Similar competitions are also organized internationally, such as programming contests, robotics olympiads, and hackathons that encourage students to develop computational thinking and coding skills.

Technology will play a significant role, and creative thinking will be necessary to address uncertain situations, such as flexibility and problem-solving (Broza et al., 2023). Changes in professional qualifications require individuals to acquire more skills than ever before. This necessitates educational systems that focus on developing critical thinking skills and the ability to approach problem-solving in different ways (Dağ et al., 2023). Not only is programming useful for writing computer programs, but it is also valuable for problem-solving and gaining a deeper understanding of the world (Ramírez de Arellano Falcón et al., 2023). Therefore, in today's digital society, it is essential for all individuals to possess computational thinking (CT) skills (Kaila et al., 2018). According to Metin et al. (2024), supporting the acquisition of CT has become essential. This skill is considered a 21st-century skill and new literacy. CT has intellectual benefits that can be applied to any field (Wing, 2014). Children who possess well-developed CT skills will be better equipped for their future (Ball & Zorn, 2015; Gülbahar, Kert & Kalelioğlu, 2019; Vaca-Cárdenas et al., 2015). The acquisition of computational thinking skills will enable future citizens to not only consume but also produce, with advanced problem-solving abilities (Kalelioğlu, 2015; Çiftçi & Topçu, 2023). CT involves algorithmic problem-solving and computational methods, which should be introduced in K-12 education (Sen, 2023).

Since Jeannette Wing redefined CT, many countries have incorporated the teaching of CT into their curricula in various ways (Munoz et al., 2023). However, there is still a lack of holistic educational approaches to develop students' CT skills (Gupta & Tiwari, 2022). Studies indicate that countries tend to benefit from applications such as computer science or coding (Arslan Namli & Aybek, 2022; Yagci, 2019). There are numerous pedagogical strategies available to promote the development of children's CT skills and knowledge Piedade & Dorotea, 2023; Yurdakök & Kalelioğlu, 2023). Researchers (Piedade & Dorotea, 2023; Tonbuloglu, B., & Tonbuloglu, I., 2019) suggest that techniques such as unplugged applications, block-based programming, and robotic coding are effective for teaching. Educational robotics, block-based programming tools, and unplugged activities are utilized in various age groups and educational settings (Kirçali & Özdener, 2023; Pala & Mıhcı Türker, 2021). For instance, primary and early childhood mostly utilize unplugged coding, while computer-based activities (such as block-based programming) are rarely used (Dağ et al., 2023). Educational robotics offers an attractive method for developing young children's CT skills. Children can interact directly with a robot and observe the immediate effects of their actions on the robot's behavior (Angeli & Valanides, 2020).

As Fessakis et al. (2013) points out, many questions arise in the teaching of coding and programming that need systematic examination. However, the literature does not provide sufficient evidence to support the claim that teaching programming improves computational thinking skills (Yurdakök & Kalelioğlu, 2023). It is unclear how coding activities can develop CT skills in diverse learning settings (Ching & Hsu, 2023). This study investigates the effects of different coding instructional strategies on the CT skills of middle childhood students, to contribute to the growing body of evidence and reduce uncertainty within the domain of computational thinking education.

Successful incorporation of CT into the K-12 education system requires emphasis on its measurement and evaluation (Sun et al., 2021). Assessment of learning is a fundamental aspect of education (Munoz et al., 2023). Although various techniques can be used to evaluate the development of CT skills, tests are one of the most useful tools for measuring CT development (Dağ et al., 2023). This study is among the first to investigate the impact of the most used coding instruction on CT skills, using the same curriculum and test. In this significant context, this research aims to answer the following research questions:

- Does robotic coding instruction increase students' CT skills scores?
- Does unplugged coding instruction increase students' CT skills scores?
- Does block-based coding instruction increase students' CT skills scores?
- How do coding instruction and school level affect students' CT skills scores?

THEORETICAL BACKGROUND

Computational Thinking

Seymour Papert introduced CT while creating the Logo programming language. The concept involves children developing algorithmic thinking through programming. Papert's idea was to promote logical thinking in children through the use of the Logo programming language. The goal was to find a way to engage students with math and geometry concepts by programming products to acquire knowledge. Jeannette M. Wing redefined CT as a domain

that draws on fundamental computing concepts to solve problems, design systems, and understand human behavior. Her explanation brought new attention to CT, leading to further research (Angeli & Valanides, 2020; Broza et al., 2023; Gupta & Tiwari, 2022).

CT is a popular concept in today's world. Many organizations and companies suggest that everyone should acquire computational thinking skills. The need for these skills in the 21st century has influenced national education policies, and countries are updating their policies to incorporate the computational thinking approach to improve students' skills (Çiftçi & Topçu, 2023). However, it can be ambiguous to some extent (Broza et al., 2023). Although there is no consensus on the definition of CT, researchers agree that it involves finding effective and efficient algorithmic solutions to problems, with or without the aid of computers. CT is a type of thinking and behavior that can be applied to real-world problems by being aware of the possibilities in computer science. CT involves multidimensional problem-solving skills across disciplines and is an essential skill across domains, rather than just being a programming skill used only by computer scientists (Çiftçi & Topçu, 2023; Dağ et al., 2023; Gupta & Tiwari, 2022; Piedade & Dorotea, 2023). CT is a multidimensional thinking skill that integrates algorithmic, creative, and critical thinking styles with communication and problem-solving skills, while also incorporating technology (Dağ et al., 2023). It is a valuable tool for problem-solving and decision-making in various fields. The solutions produced are reusable in different contexts (Çiftçi & Topçu, 2023).

Wing (2011) defines CT as the thought processes involved in formulating problems and their solutions in a way that utilizes computer science to design, understand, model, and communicate a solution to any situation in everyday life. CSTA and ISTE, in collaboration with leaders from higher and K-12 education, and industry, define computational thinking as a problem-solving process. This process involves formulating problems in a way that enables the use of a computer and other tools to generate solutions. It involves organizing and analyzing data logically, representing data through abstractions like models and simulations, automating solutions through algorithmic thinking, identifying, analyzing, and implementing possible solutions to achieve the most efficient and effective combination of steps and resources, and generalizing and transferring this process to other types of problems (Angeli & Valanides, 2020; Yılmaz İnce & Koc, 2021). The UK Royal Society (2014: as cited in Yılmaz İnce & Koc, 2021) has defined computational thinking as the process of recognizing aspects of computation in the world around us and applying tools and techniques from computer science to understand, reason, and solve problems related to both natural and artificial systems and processes.

The field of CT has been defined in various ways and broadened the term to include core concepts from the discipline of computer science (Gupta & Tiwari, 2022). Subdimensions of Computational thinking have been discussed in various studies. Although various dimensions of CT were emphasized in these studies, the most common were abstraction, algorithmic thinking, automation, debugging, decomposition, pattern generalization, problem-solving, and visualization (Angeli & Valanides, 2020; Dağ et al., 2023; Munoz et al., 2023; Pala & Mıhcı Türker, 2021). Researchers have come to accept that CT is a thought process that utilizes the elements of abstraction, generalization, decomposition, algorithmic thinking, and debugging (Grover & Pea, 2013). This study considers abstraction, algorithmic steps, decomposition, error correction, and heuristic reasoning as sub-dimensions of computational thinking. In this framework, 'algorithmic steps' corresponds to algorithmic thinking,

'error correction' aligns with debugging, and 'heuristic reasoning' encompasses pattern recognition and generalization.

Abstraction

CT is primarily concerned with abstracting complex tasks and breaking them down into smaller component tasks (Wing, 2011). Abstraction is the reduction of complexity so that the main idea can be the focus of attention (Çiftçi & Topçu, 2023). To understand the problem, abstraction is used. Abstraction and generalization are closely related: abstraction reduces complexity by hiding irrelevant details, while generalization reduces complexity by replacing multiple entities that perform similar functions with a single construct (Angeli & Valanides, 2020; Palts & Pedaste, 2020). Abstraction means focusing only on important information and ignoring irrelevant details. This process requires students to hide unnecessary details, identify key elements in a problem, and choose a representation of a system.

Algorithmic Steps

The other important elements of computational thinking include algorithmic notions of sequencing and flow control (Angeli & Valanides, 2020). Algorithms are step-by-step instructions for performing a solution or process, and taking these orderly steps is essential to solving a problem or achieving a goal (Çiftçi & Topçu, 2023; Palts & Pedaste, 2020). Algorithmic thinking involves developing a step-by-step solution or set of rules to solve a problem. Students with this skill think in terms of sequences and rules, executing and creating algorithms.

Decomposition

Decomposition is the process of breaking down a problem or task into smaller, more solvable and manageable parts (Angeli & Valanides, 2020; Broza et al., 2023; Çiftçi & Topçu, 2023; Palts & Pedaste, 2020). Decomposition involves breaking down a complex problem or system into smaller, more manageable parts. Students need to break down tasks and think about problems in terms of their component parts. They should make decisions about dividing tasks into sub-tasks with integration in mind, such as deduction.

Error Correction

This process deals with finding and fixing bugs (Çiftçi & Topçu, 2023). Correcting errors is known as debugging, a process that also involves evaluation. Debugging is the implementation of a possible solution and its verification (Angeli & Valanides, 2020; Broza et al., 2023). Debugging is the skill to recognize when actions do not correspond to instructions, and the skill to fix errors (Angeli & Valanides, 2020). Evaluations to assess the effectiveness of the solution solution (Palts & Pedaste, 2020). Evaluation aims to determine if a solution works well and effectively. Students strive to find the best solution, make decisions about the appropriate use of resources, and assess fitness for purpose.

Heuristic Reasoning

Heuristic reasoning to identify patterns/rules, similarities, and repetitive structures in data, processes, or problems (Çiftçi & Topçu, 2023). Pattern recognition involves identifying similarities and connections within and among problems. This skill requires identifying patterns and determining when they are not established, as well as extrapolating or interpolating data and organizing repeated instructions into loops or functions.

Computational thinking skills are valuable not only to computer scientists but also to individuals in various domains, including literacy, art, journalism, biology, engineering, mathematics, and science (Angeli & Valanides, 2020). Various disciplines and strategies can be employed to enhance students' critical thinking skills. Typically, computer programming is utilized to foster and cultivate CT skills (Dağ et al., 2023).

Coding Instructional Strategies

Robotic Coding

Robotics is a popular method for teaching programming to students, helping them develop positive attitudes and achieve success in learning tasks (Mikropoulos & Bellou, 2013). Theory, research, and practice suggest that robotics have value in education because they provide hands-on, mind-on learning opportunities for students. Robotics programming contributes to the development of creative design, computational thinking, and problem-solving skills (Hudson & Baek, 2022). Programming robots can be accomplished through drag-and-drop interfaces or text-based coding languages. The coding process can range from simple numeric input through a robot's keypad to more complex graphical interfaces that use interconnecting blocks as a visual programming metaphor (Stewart et al., 2021). Robotics hardware and block-based programming are suitable for lower-level students due to their user-friendly nature (Numanoğlu & Keser, 2017). Computer-assisted robotic coding plays an important role in developing abstract learning skills at the elementary and middle school levels (Cavas, B. & Cavas, H., 2005; Ersoy et al., 2011). Robotics and programming with motors and sensors make abstract and mechanical concepts like loops and variables concrete and enjoyable (Üçgül, 2018). The use of robots in programming education can help make abstract concepts more tangible, facilitating the development of problem-solving and computational thinking skills in students. This approach can lead to faster and easier acquisition of these skills (Numanoğlu & Keser, 2017). Problems in robotics are open-ended, allowing for multiple solutions and approaches. Robotics provides opportunities for learning problem-solving techniques and processes, integrates multiple domains, exposes realistic constraints and issues, and allows for creativity (Hudson & Baek, 2022). Robotics curricula can focus on basic coding activities to control a robot's physical operation. This can be engaging for children and young students as they interact directly with the electrical and mechanical components of the machine (Stewart et al., 2021). To reduce the cognitive load, it is recommended to break down the process into meaningful parts. For instance, presenting the parts first and then their interrelationships within the whole process. For instance, if programming a robot that moves without obstacles, a holistic application can be implemented. This application consists of meaningful parts, such as motor movement, sensor data reception, loop and decision structures (Şişman & Küçük, 2018). In the studies of educational robotics, students could create original products, which can increase their motivation to learn and make the learning process more effective (Karim et al., 2015; Lin et al., 2012). Teachers

and students share positive views on the process (Aksu, 2019). Constructionist principles are utilized in the process of building and programming robotics models. Real-time feedback is used to analyze and improve the design and code in a cyclical manner (Ching & Hsu, 2023; Hudson & Baek, 2022).

Block-Based Coding

Block-based visual programming languages can be beneficial for teaching coding at the elementary and middle school levels, as they can help overcome obstacles to learning and cognitive development (Wilson & Moffat, 2010). Text-based programming tools require effort to learn syntax, including parentheses, commas, special symbols, or task-specific words to control a program. Children who have not yet been introduced to algorithmic and programming logic may experience difficulties later. Therefore, block-based programming tools that simplify complex programming languages and make teaching suitable for children may be a solution to this problem. These tools enable individuals without programming knowledge to perform programming tasks (Kirçali & Özdener, 2023). Sayın and Seferoğlu (2016) conducted a literature review and found that visual programming structures allow young students to create applications without the need to learn complex code structures of traditional programming languages. These structures are designed for children according to their developmental levels, and enable them to create interactive games, animations, simulations, and stories. They also facilitate the creation of new creations. These tools can also enhance interest in programming due to their engaging structure (Kirçali & Özdener, 2023).

Scratch and Code.org are frequently used block-based programming tools that offer advantageous features compared to other tools (Baz, 2018). Çalışkan (2019) conducted a study on coding instruction and found that teachers primarily utilized Scratch, Code.org, and Blockly. The study also revealed that coding learning significantly improved students' problem-solving and logical thinking skills. However, the study identified loops as the most challenging subject to teach. At the conclusion of Kalelioğlu's (2015) study, analysis of student and teacher feedback revealed that block-based coding platforms have motivating features, useful educational materials and lesson plans, and are a viable option for teaching programming to beginners and supporting students in acquiring fundamental computer science skills.

Unplugged Coding

Unplugged coding instruction is one of the other approaches used to improve children's CT skills (Relkin & Strawhacker, 2021). Unplugged programming activities can overcome challenges such as limited access to hardware devices such as computers and the lack of early programming skills among students. It is particularly important for the equity and universality of programming education in developing countries (Sun et al., 2021). Unplugged coding includes activities that do not require the use of digital equipment (Brackmann et al., 2017). With unplugged coding, students learn through engaging games, algorithmic puzzles, maps, and similar activities (Kalelioğlu, 2015). Playing cards and creating scenarios are among the techniques used in unplugged coding instruction. These unplugged activities can teach basic computer science concepts such as artificial intelligence, human-computer interfaces, and graphics, as well as programming activities involving algorithms and other programming languages (Kirçali & Özdener, 2023).

A recent innovation in educational environments is the widespread use of technological toys (Erkoç, 2018). Technology toys are widely used in coding education. However, although many toys on the market have learning elements embedded in their design to attract the attention of parents and teachers, it is not known exactly which developmental areas are being targeted and how (Bergen, 2012). Therefore, technological toys to be used in the coding instructional process should be examined and their effects should be observed. Especially considering that they require significant investments in terms of cost, it would be useful to determine the level of impact in terms of cost-effectiveness. It is stated that educational robots primarily enable learners to work with concrete objects, they are motivating, and learners develop with the sub-dimensions of information-processing thinking with robotic activities (Üçgül, 2018). In addition, unplugged coding activities implemented in early and middle childhood provide a more equitable foundation for learning CT (Çiftçi & Topçu, 2023).

METHOD

Research Design

The study used a ten-week quasi-experimental non-equivalent groups pretest–posttest design, implemented as an after-school program. It aimed to compare the effects of different coding instructional strategies on the CT skills of elementary and middle-school students. A quasi-experimental design was chosen because random assignment was not feasible within the school setting (Cohen et al., 2018). School organization often precludes random assignment of classes (Piedade & Dorotea, 2023). In a quasi-experimental design, groups are determined without random assignment of participants. All groups are administered the same scale as a pre-test before implementation and as a posttest after implementation. This is done simultaneously for all groups (Creswell, 2002). The study employed a 2x3 factorial design, with school level and coding style as independent variables and CT skill level as the dependent variable. The impact of two factors on the dependent variable was investigated: the level of schooling (elementary and middle) and the coding instructional strategy used to enhance students' computational thinking skills. The research design consisted of three levels of the second factor: unplugged coding, block-based coding, and robotic coding, for each level of the first factor. The study was approved by the Ethics Committee of Niğde Ömer Halisdemir University. Participants voluntarily took part in the study and provided informed consent. The parents of the students were also informed about the study and its activities by the school management and the author.

Implementation

Course Design

To teach coding at elementary and middle schools, Çalışkan (2024) developed forty-nine outcome statements divided into seven categories: Problem Solving, Algorithms, Variables, Controls, Branching and Iteration, Program Development, and Debugging. This strategy aims to reduce cognitive load by presenting the process in meaningful parts rather than all at once to students. He developed teaching modules and designed learning activities suitable for them. A total of twelve courses were designed for this study. The first course required students to complete a CT skill test (pretest) to assess their previous CT skills. The same instructor (author of this

article) conducted 10-week coding activities for the students. In the final class, the participating students were required to complete another CT skill test (posttest). The CT test was developed by the author within the scope of this study.

Table 1. The Distribution of The Study Group by Class, Group, and Gender

Coding Group	Elementary School					Middle School					TOTAL
	3rd Grade		4th Grade		Total	6th Grade		7th Grade		Total	
	Girls	Boys	Girls	Boys		Girls	Boys	Girls	Boys		
Robotic	3	7	5	5	20	4	5	6	6	21	41
Block-based	5	5	6	4	20	3	8	5	5	21	41
Unplugged	2	8	7	4	21	4	6	4	6	20	41

Development of the Computational Thinking Skill Test

When assessing CT, researchers use various methods (Kastner-Hauler et al., 2022). Román-González et al. (2017) argue that traditional instruments for measuring problem-solving skills are inadequate for assessing computational ability. They suggest that specific instruments are necessary for these situations. Many of the measurement tools in the literature concentrate on problem-solving, algorithmic, or coding skills. One reason for the varying results in studies examining the effects of programming and coding teaching processes on programming self-efficacy, problem-solving, and computational thinking skills is the use of different measurement tools. To assess computational thinking, coding, and programming skills and competencies, new methods and measurement tools are necessary (Gülbahar, 2018). A multiple-choice test covering all CT subdimensions was developed.

The computational thinking skills test was developed following the steps recommended by experts in the field (Turgut & Baykul, 2010).

1. Determination of the test purpose. The purpose of developing the CT skill test is to create a tool for measuring the level of CT skills in elementary and middle school students. Therefore, the study aimed to investigate the impact of various coding instructional strategies. Research indicates that measurement tools for determining CT skills are often focused on the affective domain, such as attitude, anxiety, and competence scales, or are dependent on a particular coding platform. Therefore, it is crucial to develop a valid and reliable measurement tool that can assess CT skills with sub-dimensions objectively. To assess CT skills, multiple-choice questions were preferred. The use of multiple-choice questions in testing has several advantages. It allows for testing of many people simultaneously, has high content validity, is based on statistical foundations, and can be administered to individuals at different educational levels. Additionally, it is an economical use of answering time, even if there is a possibility of guessing correctly. Furthermore, it takes a short time to score and is less susceptible to scorer bias (Turgut & Baykul, 2010).

2. Determination of the characteristics to be measured. The characteristics to be measured by the CT skill test were determined to cover the abstraction, algorithmic steps, decomposition, error correction, and heuristic reasoning sub-dimensions of CT skills. In addition, this study also utilized the coding learning outcomes developed by Çalışkan (2024).

3. Writing the test items. During the item writing process, the studies and applications in this field were examined. A set of 75 multiple-choice questions was prepared, with 15 questions for each sub-dimension of computational thinking skills. Before the pilot application, a small group was given the draft form to determine the response time and comprehensibility of the statements. The researchers used the convenience sampling method to select the sample due to time constraints (Yıldırım & Şimşek, 2011). The survey was administered to 20 elementary and middle school students. Interviews were conducted with eight students selected based on their high and low scores. The survey underwent several modifications, including the removal of some items, modification of items and distractors, and addition of new items. A final trial form comprising 75 items was created due to problems with unclear phrasing and the identification of distractors that were either too easy or too difficult.

4. Review of the items. After the 75-item test was prepared, its validity was assessed, spelling and punctuation errors were checked, scientific accuracy was ensured, and its comprehensibility was evaluated. Validity refers to the ability of a measurement tool or method to accurately measure the intended feature without any confusion with other features (Özçelik, 2010). The validity of a test depends on its purpose, application conditions, the group it is applied to, and the scoring method. Correct interpretation of the results is crucial (Ellez, 2011). To ensure the validity of the test, expert opinions in the field are often cited in the literature (Çalık & Ayas, 2003). Tavşancıl (2010) defines content validity as the expert opinion on the feature being measured by the tool. In this study, five experts were consulted to establish the construct and content validity of the CT skill test. Two of the experts were from the field of Computer Science; one was from the field of measurement and evaluation, and two were from the field of gifted education. The experts hold doctoral degrees and are interested in computational thinking. The experts received a form with guidelines to ensure consistency between the audits, and necessary corrections were made to some items in line with the criticisms and suggestions received. The aim of the study was to determine whether the questions accurately represented the domains they were intended to assess and to evaluate their appropriateness for the target age groups. Yurdugül (2005) suggests that items with negative or zero content validity ratios should be removed from the test. In this study, five such items were removed after expert opinion. The remaining items had high content validity ratio values, ranging from 0.6 to 1. According to Şencan (2005), a Kappa Coefficient of Concordance less than 0.20 indicates poor agreement, while a coefficient between 0.20-0.40 is considered acceptable. A coefficient between 0.40-0.60 indicates moderate agreement, while a coefficient between 0.60-0.80 indicates good agreement. Finally, a coefficient between 0.80-1.00 indicates very good agreement. The high Kappa agreement coefficients indicate a high level of consistency between experts when evaluating the findings in this context.

In addition to validity, another important feature of a good measurement tool is reliability. Gönen et al. (2011) state that a reliable test can be applied in other trials. Reliability refers to the consistency or repeatability of measurements obtained from a test or measurement tool when applied to a specific population or sample. Bademci (2011) also defines reliability as the criterion for a test to be consistent or repeatable. It is the degree to which measurement results are free from random errors. Turgut and Baykul (2010) describe reliability as the degree of freedom of measurement results from random errors. After applying the measurement tool once, the reliability coefficient for the obtained scores is calculated using methods based on item variance (Ellez, 2011). In this study, the KR-20 Reliability Coefficient was used to calculate reliability. The KR-20 coefficient is utilized to determine

the reliability of tests with two scoring categories: '1' for correct answers and '0' for incorrect answers (Baykul, 2010). A reliability coefficient value above .70 indicates high reliability of the scale (Büyüköztürk, 2007). In this study, the KR-20 reliability coefficient was calculated as 0.969. Further details are provided in the relevant section below.

5. Preparation of the trial form. During the preparation of the test form, to minimize fatigue, items from the same category were distributed evenly throughout the test. It was ensured that the first questions were easier than the others. Attention was also paid to font size, spelling, and punctuation rules during item writing. At the start of the test, informative and concise instructions were provided. The test, which comprised of 70 items, was then prepared for pilot application.

6. Implementation of the trial form. The test was administered to a total of 532 students, consisting of 124 third graders, 141 fourth graders, 160 fifth graders, and 107 sixth graders. Of these, 332 were boys and 200 were girls. Prior to the test, the students received a brief explanation of its purpose, duration, and scope. They were instructed to read the questions carefully and select the option they believed to be correct. The test lasted for 60 minutes.

7. Scoring of application results, item analysis, and item selection. After administering the test, the correct and incorrect answers of the students were determined. Items that were answered correctly received 1 point, while those that were left blank or answered incorrectly received 0 points. The scores were then ranked from highest to lowest. Next, the upper and lower 27% groups were formed and item analyses were conducted using Microsoft Excel. Item selection was based on item discrimination index (rjx) and item difficulty index (pjx). Item difficulty is defined as the number of correct answers divided by the number of respondents, and item discrimination is defined as the degree to which an item distinguishes between individuals who exhibit the tested behavior and those who do not (Özçelik, 2010). Item difficulty is measured on a scale from 0 to 1. A lower difficulty value indicates a more challenging item with a lower response rate, while a higher value indicates an easier item with a higher response rate. A discrimination value closer to 0 indicates a decrease in discrimination, while a value closer to 1 indicates an increase in discrimination. Item discrimination ranges from -1 to +1. According to Gönen et al. (2011), an item with a negative discrimination value does not serve the purpose of the test and reduces its reliability. The criteria used to determine which items were accepted for the test are as follows:

The item discrimination index (rjx) determines whether an item is accepted or not.

- If the value is 0.19 or less, the item is not accepted.
- If it falls between 0.20 and 0.29, the item should be corrected.
- If it is between 0.30 and 0.39, it is considered a good item and is accepted.
- If it is 0.40 or higher, it is a very good item and is accepted (Özçelik, 2010).

8. Finalize the test and statistics. After analyzing the items, 10 were removed and 4 were revised, while the remaining items remained unchanged. It is recommended that the average item difficulty index of the test be around 0.50 (Turgut & Baykul, 2010). The item difficulty level of the test developed in this study was calculated to be 0.639, which is close to the desired level. Therefore, the test can be considered to be of medium difficulty.

The discrimination levels of the items in the test indicate a strong item structure. Additionally, the test items were of moderate difficulty and demonstrated high reliability (KR-20 = .969).

Coding Instructional Materials

The study distributed elementary and middle school students equally among groups that received training in robotic coding, block-based coding, and unplugged coding. The Lego Education SPIKE Prime set was used for robotic coding training, while Scratch was used for block-based coding training. The Matatalab Pro set was used for unplugged coding activities.

Lego Education SPIKE Prime is an adaptable, hands-on learning system that engages students through playful problem-solving and storytelling. The system allows children to program and move their own robots using drag and drop. This approach helps students learn difficult abstract concepts, such as loops and conditions, by doing, experiencing, and seeing. The set comprises a programmable hub, motors, distance, power, and color sensors, as well as LEGO bricks for building. The programmable hub can store up to 20 programs and has a customizable 5x5 light matrix, a built-in gyroscope, and six ports with input/output auto-detection. It also includes two medium motors and one large motor with a new design that helps save space in your builds and allows for easy positioning.

Although there are several block-based programming environments available for cultivating CT skills, Scratch is the most used, especially for young students (Yılmaz İnce & Koc, 2021). Scratch was developed by the Massachusetts Institute of Technology (MIT) in the United States to help children aged 8-16 learn key 21st century life skills such as creative thinking, causal reasoning, and teamwork. It is a free platform used in over 150 countries and available in more than 40 languages. Scratch is a user-friendly program that is often familiar to elementary school students. It has a very simple interface for dragging and dropping code blocks and characters. It facilitates meaningful learning processes by providing a playful environment that encourages experiential and exploratory learning, allowing for the freedom to play with ideas. Like game-based learning environments, it invites learners with varying competencies to enjoyably discover problem-solving strategies (Broza et al., 2023).

The Matatalab Pro Coding Set enables students to determine probabilities and create algorithms by controlling Matata Bot with code blocks. The written code provides instant feedback, facilitating a fast-learning process. The set incorporates physical activity and art plugins to support student learning. The package includes the MatataBot, Command Tower, Control Card, 99 Code Blocks, one double-sided playground, and six Art and Music Warm-up Cards.

FINDINGS

Skewness and kurtosis for the pretest were 0.079 and −1.413, respectively; for the posttest they were −1.337 and 1.358, respectively. These values fall within the acceptable range (±1.5), suggesting approximate normality (Tabachnick & Fidell, 2012). The pretest data for CT skills had a mean of 30.16, a mode of 12.00, and a median of 29.00. Similarly, the posttest data for CT skills had a mean of 46.53, a mode of 56.00, and a median of 49.00. These values suggest a normal distribution for the test.

Table 2. Pretest and Posttest CT Scores by Group (Paired t-Tests)

School Level	Coding Instruction Group	Pretest			Posttest			t	Cohen's d	p (two-tailed)
		N	X	S.D.	N	X	S.D.			
Elementary	Robotic	20	26.70	11.85	20	44.65	13.24	-6.154	-1.376	<.001
	Block-based	20	26.20	11.66	20	46.05	9.36	-6.703	-1.499	
	Unplugged	21	24.67	12.99	21	43.67	11.41	-7.218	-1.575	
Middle	Robotic	21	33.19	16.72	21	48.00	12.86	-4.643	-1.013	
	Block-based	21	34.95	16.22	21	48.43	11.00	-5.808	-1.267	
	Unplugged	20	35.15	16.32	20	48.35	11.86	-6.016	-1.345	

Prior to the application, a T-test analysis for independent samples was conducted to determine if there were significant differences in students' computational thinking skill scores based on their school level. An independent sample t-test revealed that the two groups had significantly different initial CT skills scores ($t = -3.336$, $p < .001$) prior to the interventions. Specifically, the mean CT skill score of the middle school group (34.42) was higher than that of the elementary school group (25.84). One-way ANOVA analysis was conducted to determine if there was a statistically significant difference in pre-implementation computational thinking skill scores among students in different coding instruction groups at each school level. At the elementary school level, no statistically significant difference was found between the groups' pre-test scores ($F(2, 58) = .156$, $p = .856$). The mean scores for the robotic ($M = 26.7$, $SD = 11.85$), block-based ($M = 26.20$, $SD = 11.66$), and unplugged coding ($M = 24.67$, $SD = 12.99$) groups were comparable before the intervention. Similarly, at the middle school level, the groups also demonstrated comparable baseline CT skills, with no significant differences detected ($F_{(2, 59)} = .090$, $p = .914$) among the robotic ($M = 33.19$, $SD = 16.72$), block-based ($M = 34.95$, $SD = 16.22$), and unplugged coding ($M = 35.15$, $SD = 16.32$) groups. This indicates that middle school students possess higher computational thinking skills than elementary school students, but there is no difference between the coding groups created at each school level.

To address the research questions, paired groups T-test analysis was utilized to compare pretest and posttest scores of students in each coding instruction type. Following the interventions, there was a statistically significant difference between the pretest and posttest scores, with a large effect size observed in all group settings. There was a significant increase in the CT skills scores of elementary and middle school students from pretest to posttest. Table 2 presents the statistical data.

No statistically significant differences were found among instructional groups, indicating comparable effectiveness across strategies both before and after the intervention. Based on the results of the one-way ANOVA analysis, the analysis revealed no statistically significant differences in CT skill scores among the three instructional groups at the elementary school level after the intervention ($F_{(2, 58)} = .224$, $p = .800$). Similarly, no statistically significant difference was found among the coding groups after the intervention ($F_{(2, 59)} = 0.008$, $p = 0.992$).

The difference seen between elementary and middle school students before the application was not observed after the coding instructional process. An independent sample t-test revealed no statistically significant difference was observed between the groups, suggesting a comparable level of effectiveness in CT skill scores between the two groups after the interventions ($t = -1.679$, $p = .096$). However, the mean CT skill score of the middle school level

(48.26) remained higher than that of the elementary school level (44.77). This demonstrates that elementary school students were able to improve and close the gap with the given coding instruction.

A two-way between-groups ANCOVA was conducted to examine the difference between posttest scores while controlling for pre-test scores. The analysis included school level and coding group type as factors. The results indicated that there was no statistically significant difference between students in different school levels and coding group settings after controlling for their pretest scores on CT ($F_{(2, 116)} = .121$, $p = .886$)..

CONCLUSION and DISCUSSION

This study explored the impact of three distinct instructional approaches—robotic coding, block-based programming environments, and unplugged coding activities—on the development of computational thinking (CT) skills among elementary and middle school students. By implementing and comparing these instructional methods in educational settings, the study aimed to identify their relative contributions to students' CT skill development and to examine potential differences in learning outcomes. It is worth noting that a universally accepted definition of CT has not yet been established (Kastner-Hauler et al., 2022). Additionally, there is a dearth of research on effective teaching methods to enhance students' CT abilities (Sun et al., 2021).

In this study, a valid and reliable measurement tool was developed. Three different coding instructional processes were designed and applied to elementary and middle school students. Although middle school students had higher computational skills than elementary school students before the process, the findings indicate that robotic, block-based, and unplugged coding interventions yielded comparable gains in students' CT development, ultimately closing the initial gap found between school levels. Studies have shown that robotic, block-based, and unplugged coding instruction types all have a significant impact on students' computational skills and increase their scores. These results suggest that each instructional strategy—robotic, block-based, and unplugged coding—is equally efficacious in fostering students' computational thinking skills. It is particularly noteworthy that different coding instructional strategies used separately have been found to be similarly effective in other studies. Research has consistently shown that teaching robotics has a positive impact on students' computational thinking skills (Yurdakök & Kalelioğlu, 2023). In their study, Hudson and Baek (2022) found that the process of teaching robotic coding improved students' computational thinking skills. According to Yılmaz İnce and Koc (2021), robotics and block-based coding have a positive impact on students' computational thinking skills. Oluk and Korkmaz (2016) and Arslan Namli and Aybek (2022) found that using Scratch develops students' CT skills. Another study (Kastner-Hauler et al., 2022) suggests that block-based programming could be a promising approach to promote CT skills in lower school grades. According to Kirçali and Özdener (2023), unplugged coding tools are equally effective as robotic and block-based tools in the coding process. Various studies have concluded that unplugged coding enhances the CT of students across different grade levels (Angeli & Valanides, 2020; Dağ et al., 2023; Sun et al., 2021; Tonbuloglu, B., & Tonbuloglu, I., 2019).

After statistically controlling for baseline differences in students' pre-test scores, it was determined that the combination of school level and coding instructional strategies did not have a significant impact on students' CT skills. Therefore, it can be concluded that no single coding instruction method demonstrated a statistically

significant advantage over the others in improving CT skills. According to Kirçali and Özdener (2023), instructional processes develop students' computational thinking skills regardless of group structure. Various studies (Delal & Öner, 2020; Kirçali & Özdener, 2023) have concluded that factors such as gender and grade level do not significantly affect the process. The important factor is the coding instructional process itself. Based on this study, it is suggested that various instructional activities are beneficial for cultivating CT skills. This underscores the critical role of instructional design and teacher facilitation in optimizing learning outcomes, irrespective of the technological medium employed. Robotics, block-based software, and technological toys support the teacher's role as teaching materials.

RECOMMENDATIONS

This study offers several recommendations for fostering students' computational thinking (CT) skills through coding instruction, based on the findings of the study.

1. Since the results indicated that the type of coding approach (robotic, block-based, or unplugged) did not produce a statistically significant difference in students' final CT skill levels, greater attention should be paid to the design of the instructional process. In particular, elements such as the duration of activities, the structure of tasks, opportunities for problem-solving, and the role of the teacher as a facilitator may play a more critical role in supporting CT development.
2. Based on the observed learning process, teachers may begin coding instruction with unplugged coding activities before transitioning to block-based and robotic coding environments. This gradual progression may help students first understand fundamental CT concepts without the cognitive load associated with technological tools.
3. As hands-on learning experiences appeared to increase student engagement and motivation during the instructional process, coding activities that emphasize product creation, design tasks, and collaborative problem-solving may be incorporated more frequently into classroom practices.
4. Since the instructional process and the teacher's guidance play a significant role in student development, teachers may benefit from in-service training that focuses not only on the use of robotic, block-based, and unplugged coding tools but also on effective instructional design strategies for integrating these tools into learning activities.
5. Finally, future research may investigate how different instructional designs, activity durations, and teacher facilitation strategies influence the development of CT skills across various subject areas and grade levels.

REFERENCES

Aksu, F. N. (2019). *Bilişim teknolojileri öğretmenleri gözünden robotik kodlama ve robotik yarışmaları* [Yayımlanmamış Yüksek Lisans Tezi]. Balıkesir Üniversitesi.

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*, 105, 105954. https://doi.org/10.1016/j.chb.2019.03.018

Arslan Namli, N., & Aybek, B. (2022). An investigation of the effect of block-based programming and unplugged coding activities on fifth graders' computational thinking skills, self-efficacy and academic performance. *Contemporary Educational Technology*, *14*(1), 341. https://doi.org/10.30935/cedtech/11477

Bademci, V. (2006). Tartişmayi sonlandirmak: Cronbach'in alfa katsayisi, iki değerli [0,1] ölçümlenmiş maddeler ile kullanilabilir. *Atatürk Üniversitesi Kazım Karabekir Eğitim Fakültesi Dergisi, 13*, 438-446.

Ball, T. & Zorn, B. (2015). Teach foundational language principles. *Communications of the ACM*, 58(5), 30–31. https://doi.org/10.1145/2663342

Baykul, Y. (2010). *Eğitimde ve psikolojide ölçme: Klasik test teorisi ve uygulaması (2. Baskı)*. Pegem Akademi Yayıncılık.

Baz, F. Ç. (2018). Çocuklar için kodlama yazılımları üzerine karşılaştırmalı bir inceleme. *Current Research in Education, 4*(1), 36–47. https://www.researchgate.net/publication/324138421_Cocuklar_icin_kodlama_yazilimlari_uzerine_karsilastirmali_bir_inceleme

Bell, T., Witten, I. H., & Fellows, M. (2015). CS unplugged. An enrichment and extension programme for primary-aged students. New Zealand: University of Canterbury. CS Education Research Group. https://classic.csunplugged.org/documents/books/english/CSUnplugged_OS_2015_v3.1.pdf

Bergen, D. (2012). Play, technology toy affordances, and brain development-research needs and policy issues. In L.E. Cohen & S. Waite-Stupiansky (Eds.) *Play: A polyphony of research, theories, and issues* (pp. 163-173). University Press of America Inc.

Brackmann, C. P., Roman-Gonzalez, M., Robles, G., Moreno-Leon, J., Casali, A., & Barone, D. (2017, November 8-10). *Development of computational thinking skills through unplugged activities in primary school* [Conference presentation abstract]. 12th Workshop on Primary and Secondary Computing Education, Nijmegen, The Netherlands. https://doi.org/10.1145/3137065.3137069

Broza, O., Biberman-Shalev, L., & Chamo, N. (2023). "Start from scratch": Integrating computational thinking skills in teacher education program. *Thinking Skills and Creativity, 48*, 101285-. https://doi.org/10.1016/j.tsc.2023.101285

Büyüköztürk, Ş. (2007). *Sosyal bilimler için veri analizi el kitabı. (8. baskı)*. Pegem Yayıncılık.

Çalık, M., & Ayas, A. (2003). Preparation and application of concept achievement test in solutions. *Pamukkale University Faculty of Education Journal, 2*(14), 1-17. https://dergipark.org.tr/tr/pub/pauefd/issue/11129/133088

Çalışkan, E. (2019, April 25-28). Examining the coding teaching process in secondary school through the eyes of information technologies teachers [Conference presentation abstract]. 28. International Education Sciences Congress, Ankara, Türkiye. https://fs.hacettepe.edu.tr/egitim/Bildiri%20Kitapları/icesozet.pdf

Çalışkan, E. (2024, February 08-10). *Learning outcomes and activity development for teaching coding at primary and secondary school levels* [Conference presentation abstract]. 11th International "Başkent" Congress on Social, Humanities, Administrative, and Educational Sciences, Ankara, Türkiye. https://drive.google.com/file/d/1yhN3m2M0ObxEY9yuMihr47DfbZ2rBoqc/view?usp=sharing

Cavas, B. & Cavas, H. P. (2005, February 2–4). *Technology based learning: Robotics club*. [Conference presentation abstract]. Seventh Academic Informatics Conference, Gaziantep University, Gaziantep, Turkey. https://ab.org.tr/ab05/

Ching, Y.-H., & Hsu, Y.-C. (2023). Educational robotics for developing computational thinking in young learners: A systematic review. *TechTrends*, 1–12. https://doi.org/10.1007/s11528-023-00841-1

Çiftçi, A., & Topçu, M. S. (2023). Improving early childhood pre-service teachers' computational thinking skills through the unplugged computational thinking integrated STEM approach. *Thinking Skills and Creativity, 49*, 101337. https://doi.org/10.1016/j.tsc.2023.101337

CodeCrush (2024). *CodeCrush web page*. https://codecrush.unomaha.edu/

Cohen, L., Manion, L., & Morrison, K. (2018). *Research methods in education (8th edition)*. Routledge.

Creswell, J. W. (2002). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Dağ, F., Sumuer, E., & Durdu, L. (2023). The effect of an unplugged coding course on primary school students' improvement in their computational thinking skills. *Journal of Computer Assisted Learning, 39*(6), 1902–1918. https://doi.org/10.1111/jcal.12850

Delal, H., & Öner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. *Informatics in Education, 19*(1), 1–13. https://doi.org/10.15388/INFEDU.2020.01

Demirer, V. & Sak, N. (2016). Programming education and new approaches around the world and in Turkey, *Journal of Theory Practice Education, 12*(3), 521–546. https://dergipark.org.tr/en/pub/eku/issue/26697/280853

Durak, H.Y. & Sarıtepeci, M. (2018). Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers & Education, 116, 191-202*. https://doi.org/10.1016/j.compedu.2017.09.004

Ellez, A. M. (2011). Ölçme araçlarının sahip olması gereken özellikler. A. Tanrıöğen (Ed.) Bilimsel araştırma yöntemleri (2. baskı) (pp. 165-190). Anı Yayıncılık.

Erkoç, M. F. (2018). Bilgi işlemsel düşünme ve teknolojik oyuncaklar. Y. Gülbahar (Ed.) *Bilgi İşlemsel Düşünmeden Programlamaya (2. baskı)* (*pp.*207-236). Pegem Yayıncılık.

Ersoy, H., Madran, R. O., & Gulbahar, Y. (2011, February 2–4). *A model proposed for teaching programming languages: robotic programming* [Conference presentation abstract]. XIII. Academic Informatics Conference, Inonu University, Malatya, Turkey. https://ab.org.tr/ab11/

Gönen, S., Kocakaya, S. & Kocakaya, F. (2011). A Study on developing an achievement test with validity and reliability on dynamics. *Journal of the Faculty of Education of Yüzüncü Yıl University, 8*(1), 40-57. https://dergipark.org.tr/tr/pub/yyuefd/issue/13707/165951

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning in school* (p. 20-38). Bloomsbury Publishing.

Gülbahar, Y., Kert, S. B., & Kalelioğlu F. (2019). Self-efficacy perception scale for computational thinking skills: A validity and reliability study. *Turkish Computer and Mathematics Education Journal, 10*(1), 1-29. https://doi.org/10.16949/turkbilmat.385097

Gupta, S., & Tiwari, A. A. (2022). A design-based pedagogical framework for developing computational thinking skills. *Journal of Decision Systems*, *31*(4), 433–450. https://doi.org/10.1080/12460125.2021.1943880

Hudson, M.A., & Baek, Y. (2022) Increasing elementary students' computational thinking skills using a multifaceted robotics-based intervention. *Computers in the Schools*, *39*(1), 16-40, https://doi.org/10.1080/07380569.2022.2037295

Kaila, E., Laakso, M. J., & Kurvinen, E. (2018, May 21-25). *Teaching future teachers to code - Programming and computational thinking for teacher students* [Conference presentation abstract]. 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia. https://doi.org/10.23919/MIPRO.2018.8400127

Kalelioğlu, F. 2015. A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200-210. https://doi.org/10.1016/j.chb.2015.05.047

Karim, M. E., Lemaignan, S., & Mondada, F. (2015, July 1–3). *A review: Can robots reshape K-12 STEM education?* [Conference presentation abstract]. 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO 2015), Lyon, France. http://infoscience.epfl.ch/record/209219

Kasalak, İ. (2017). *Robotik kodlama etkinliklerinin ortaokul öğrencilerinin kodlamaya ilişkin öz-yeterlik algılarına etkisi ve etkinliklere ilişkin öğrenci yaşantıları* [Yayımlanmamış Yüksek Lisans Tezi]. Hacettepe Üniversitesi.

Kastner-Hauler, O., Tengler, K., Sabitzer, B., & Lavicza, Z. (2022). Combined effects of block-based programming and physical computing on primary students' computational thinking skills. *Frontiers in Psychology*, *13*, 875382–875382. https://doi.org/10.3389/fpsyg.2022.875382

Kirçali, A. Ç., & Özdener, N. A. (2023). Comparison of plugged and unplugged tools in teaching algorithms at the k-12 level for computational thinking skills. *Tech Know Learn*, *28*, 1485–1513. https://doi.org/10.1007/s10758-021-09585-4

Lin, C. H., Liu, E. Z. F., & Huang, Y. Y. (2012). Exploring parents' perceptions toward educational robots: Gender and socioeconomic difference. *British Journal of Educational Technology*, *43*(1), E31–E34. https://doi.org/10.1111/j.1467-8535.2011.01258.x

Metin, Ş., Başaran, M., Seheryeli, M. Y., Relkin, E., & Kalyenci, D. (2024). Adaptation of the computational thinking skills assessment tool (TechCheck-K) in early childhood. *Journal of Science Education and Technology*, *33*, 365-382. https://doi.org/10.1007/s10956-023-10089-2

Mikropoulos, T. A. & Bellou, I. (2013). Educational robotics as mindtools. *Themes in Science and Technology Education, 6*(1), 5–14. https://files.eric.ed.gov/fulltext/EJ1130925.pdf

Munoz, R. F. Z., Alegria, J. A. H., & Robles, G. (2023). Assessment of Computational Thinking skills: A systematic review of the literature. *IEEE-RITA*, *18*(4), 1–1. https://doi.org/10.1109/RITA.2023.3323762

Nolan, A. (2021). Artificial intelligence, its diffusion and uses in manufacturing. *OECD going digital toolkit notes*. OECD Publishing. https://doi.org/10.1787/249e2003-en

Numanoglu, M. & Keser, H. (2017). Robot usage in programming teaching - Mbot example. *Bartın University Journal of Faculty of Education, 6*(2), 497–515. https://doi.org/10.14686/buefad.306198

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' Scratch skills with their computational thinking skills in terms of different variables. *International Journal of Modern Education and Computer Science, 8*(11), 1-7. https://doi.org/10.5815/ijmecs.2016.11.01

Özçelik, D. A. (2010). Test hazırlama kılavuzu (4. baskı). Pegem Akademi Yayıncılık.

Pala, F. K., & Mıhcı Türker, P. (2021). The effects of different programming trainings on computational thinking skills. *Interactive Learning Environments*, *29*(7), 1090–1100. https://doi.org/10.1080/10494820.2019.1635495

Palts, T., & Pedaste, M. (2020). A model for developing computational thinking skills. *Informatics in Education*, *19*(1), 113–128. https://doi.org/10.15388/INFEDU.2020.06

Piedade, J., & Dorotea, N. (2023). Effects of Scratch-based activities on 4th-grade students' computational thinking skills. Informatics in Education, 22(3), 499–523. https://doi.org/10.15388/infedu.2023.19

Prensky, M. (2008). *Programming is the new literacy*. https://www.edutopia.org/literacy-computer-programming

Ramírez de Arellano Falcón, B., del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2023). Is gamification always productive? A study of the effectiveness of Bebras cards in promoting primary students' computational thinking skills. *Asia Pacific Education Review, 26*, 117-131. https://doi.org/10.1007/s12564-023-09905-6

Relkin, E., & Strawhacker, A. (2021). Unplugged learning: Recognizing computational thinking in everyday life. In M. Bers (Ed.), *Teaching computational thinking and coding to young children* (pp. 41–62). IGI Global.

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

Rushkoff, D. (2012). *Code literacy: A 21st-century requirement*. https://www.edutopia.org/blog/code-literacy-21st-century-requirement-douglasrushkoff

Sayın, Z., & Seferoğlu, S.S. (2016, February 3-5). *Coding education as a new 21st century skill and its impact on education policies* [Conference presentation abstract]. XVIII. Academic Informatics Conferences, Adnan Menderes University, Aydın, Türkiye. https://yunus.hacettepe.edu.tr/~sadi/yayin/AB16_Sayin-Seferoglu_Kodlama.pdf

Sen, S. (2023). Relations between preservice teachers' self-efficacy, computational thinking skills and metacognitive self-regulation. *European Journal of Psychology of Education*, *38*(3), 1251–1269. https://doi.org/10.1007/s10212-022-00651-8

Şencan, H. (2005). *Sosyal ve davranışsal ölçümlerde güvenirlik ve geçerlik*. Seçkin Yayıncılık.

Şişman, B. & Küçük, S. (2018). Pre-service teachers' flow, anxiety and cognitive load levels in robotics programming. Educational Technology Theory and Practice, 8(2), 108–124. https://doi.org/10.17943/etku.366193

Stewart, W. H., Baek, Y., Kwid, G., & Taylor, K. (2021). Exploring Factors That Influence Computational Thinking Skills in Elementary Students' Collaborative Robotics. *Journal of Educational Computing Research*, *59*(6), 1208–1239. https://doi.org/10.1177/0735633121992479

Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity*, *42*, 100926. https://doi.org/10.1016/j.tsc.2021.100926

Tabachnick, B. G., & Fidell, L. S. (2012). *Using multivariate statistics (6th ed.)*. Pearson.

Tavşancıl, E., (2010). *Tutumların ölçülmesi ve SPSS ile veri analizi (4. Baskı).* Nobel Akademik Yayıncılık.

Tonbuloglu, B., & Tonbuloglu, I. (2019). The effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education*, *18*(2), 403–426. https://doi.org/10.15388/infedu.2019.19

Turgut, M. F. & Baykul, Y. (2010). *Eğitimde ölçme ve değerlendirme (2nd Edition)*. Pegem Akademi Yayıncılık.

Üçgül, M. (2018). Eğitsel robotlar ve bilgi işlemsel düşünme. Y. Gülbahar (Ed.) *Bilgi işlemsel düşünmeden programlamaya (2. baskı)* (pp.295-314). Pegem Akademi Yayıncılık.

USA Computing Olympiad, (2019). *USA computing olympiad*. http://www.usaco.org/

Vaca-Cárdenas, L.A., Bertacchini, F., Tavernise, A., Gabriele, L., Valenti, A., Olmedo, D.E., ... Bilotta, E. (2015, September 20-24). *Coding with Scratch: The design of an educational setting for Elementary pre-service teachers* [Conference presentation abstract]. 2015 International Conference on Interactive Collaborative Learning, Florence, Italy. https://doi.org/10.1109/ICL.2015.7318200

Wing, J. M. (2011). *Research notebook: Computational thinking - what and why?* http://www.cs.cmu.edu/link/research-notebookcomputational-thinking-what-and-why

Wing, J. M. (2014). Computational thinking benefits society. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

World Economic Forum. (2016). The future of jobs: Employment, skills and workforce strategy for the fourth industrial revolution. *Geneva, Switzerland: World Economic Forum*. http://www3.weforum.org/docs/WEF_Future_of_Jobs.pdf

Yagci, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, *24*(1), 929–951. https://doi.org/10.1007/s10639-018-9801-8

Yilmaz Ince, E., & Koc, M. (2021). The consequences of robotics programming education on computational thinking skills: An intervention of the Young Engineer's Workshop (YEW). *Computer Applications in Engineering Education*, *29*(1), 191–208. https://doi.org/10.1002/cae.22321

Yıldırım, A., & Şimşek, H. (2011). *Sosyal bilimlerde nitel araştırma yöntemleri (8. baskı)*. Seçkin Yayıncılık.

Yurdakök, E. A., & Kalelioğlu, F. (2023). The effect of teaching physical programming on computational thinking skills and self-efficacy perceptions towards computational thinking. *Journal of Educational Computing Research*. https://doi.org/10.1177/07356331231220313

Yurdugül, H. (2005, September 28-30). *Using Content validity indexes for content validity in scale development studies* [Conference presentation abstract]. XIV. National Congress of Educational Sciences, Pamukkale University Faculty of Education, Denizli, Türkiye. https://yunus.hacettepe.edu.tr/~yurdugul/3/indir/PamukkaleBildiri.pdf

Ethics Statement: In this article, the journal writing rules, publication principles, research and publication ethics rules, and journal ethics rules have been complied with. The responsibility for any violations that may arise regarding the article belongs to the author. The article's ethics committee approval was obtained 10.02.2021 dated and 2021/03-12 numbered approval Niğde Ömer Halisdemir University Ethics Committee.

Declaration of Author(s)' Contribution Rate: The author's contribution rate is 100%.

CONTRIBUTION RATE	CONTRIBUTORS
Idea or Notion	Erkan ÇALIŞKAN
Literature Review	Erkan ÇALIŞKAN
Yöntem	Erkan ÇALIŞKAN
Data Collecting	Erkan ÇALIŞKAN
Data Analysis	Erkan ÇALIŞKAN
Findings	Erkan ÇALIŞKAN
Discussion and Commentary	Erkan ÇALIŞKAN

Funding: This study was supported by a grant from the Research Projects Unit of Niğde Ömer Halisdemir University (Grant No. EBT 2022/1-BAGEP).

Informed Consent Statement: Informed consent forms were obtained from all participants in the study.

Data Availability Statement: For questions regarding data sets, etc., the corresponding author should be contacted.

Conflict of Interest: The author has no conflict of interest with any other individuals, institutions, or organizations involved in this research.

This study is licensed under CC BY (https://creativecommons.org/licenses/by/4.0/deed.en).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of IJETSAR and/or the editor(s). IJETSAR and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.