Collaboration of Unplugged and Plugged Activities for Primary School Students: Developing Computational Thinking with Programming

Semra FİŞ ERÜMİT

semrafiserumit@ktu.edu.tr

Karadeniz Technical University, Trabzon, Türkiye

Abstract

This study investigates the contribution of plugged and un-plugged activities to primary school students' development of computational thinking skills. The plugged and unplugged activities were used together in this study. In the implementation, in addition to the un-plugged activities prepared by the "Ministry of National Education," activities prepared by the researcher were also used. Plugged activities were also determined and implemented on the code.org website according to the age of the students and subjects. A quasi-experimental design was used with a single group to determine the changes before and after learning and to investigate the research questions. The measurements were performed with the Bebras tasks both before and after the implementation. Bebras consists of internationally valid tasks that measure computational thinking. The results showed that the combination of plugged and unplugged activities helped improve students' computational thinking skills. Our findings show that using a combination of unplugged and plugged activities is beneficial for primary school students. Further research is needed to evaluate these activities separately and their role in providing gains. Additionally, the effects of using different teaching methods in programming education can be examined.

Keywords: Primary School, Teaching Programming, Computational Thinking, Un-plugged activities, Plugged activities

1. Introduction

When Wing (2006) first referred to computational thinking (CT), it was defined as analysing problems, using abstraction to make their structures understandable, and logically developing solutions to them. In later years, Wing (2017) defined CT as the skill to find and pursue solutions to problems in a manner compatible with computer operations; in other words, approaching problems as computer scientists would. Accordingly, Grover and Pea (2013) defined CT as the process of formulating problems in a format that can be solved by computer programming. The International Society for Technology in Education (ISTE) determined the concept of CT as a combination of "algorithmic, creative, and logical thinking and problem-solving skills" (ISTE, 2015). Correspondingly, in many studies, the concept of CT is construed as the ability to create solutions to problems using algorithmic thinking to analyse, abstract, and transform information with computer applications, and to use modelling skills in succession (Durak & Saritepeci, 2018; Tsarava et. al., 2022). Different definitions continue to be established regarding the concept of CT (Shute et al., 2017).

Today, it is widely accepted that in addition to cognitive skills, learners should develop skills such as problem-solving, critical thinking, communication, cooperation, and self-management, which are referred to as 21st-century skills (Nouri et al., 2020). It is assumed that individuals who have these skills will become inquiring, analytical, and productive citizens that our times require. In this context, the improvement of CT is directly related to developing problem-solving and critical-thinking skills (Kong, 2016). Considering that computer science interacts with multiple fields, it may be inferred that CT proficiency affects the skills of people in different science and mathematics disciplines, including problem-solving, algorithmic thinking, creative thinking, analytical and logical thinking skills (Popat & Starkey, 2019; Tsarava et. al., 2022). Acquiring these skills beginning in the early grades will facilitate learners' development of these skills throughout their schooling and prepare future generations for the rapid change characteristic of a technology-driven society. For this reason, the importance of beginning training in coding and CT in primary school has been emphasized (Durak & Sarıtepeci, 2018), and important issues include

the kinds of activities that should be used for learning programming at primary school, which activities can develop CT skills, and what planning should be done in the execution of the activities.

Within this context, programming training was provided to primary school students, and the development of both programming sub-skills and CT were examined. Recently, many studies have been conducted on the development of CT through programming teaching (Ching, Hsu, & Baldwin, 2018; Tikva & Tambouris, 2021). Studies on this topic have either focused on plugged activities (Armoni, Meerbaum-Salant, & Ben-Ari, 2015; S'aez-L'opez et al., 2016) or unplugged activities (Brackmann et al. 2017; Tsarava, Moeller, & Ninaus, 2018), as well as activities that compare both approaches (del Olmo-Muñoz et al., 2020; Erümit & Sahin, 2020; Sigayret, Tricot, & Blanc, 2022; Kirçali & Özdener, 2022), or have applied both methods together (Jiang & Wong, 2019; Tsarava et al., 2017). Studies generally focus on various purposes, such as activities, the effects of coding activities on learning and motivation, the improvement of CT, and the practice of different methods, such as game-based activities. In this study, unlike previous studies, a different perspective for primary school students is presented by investigating what plugged and unplugged activities can be at primary school, especially how these activities can be used together, and how these activities affect students' CT. In addition, measuring CT with internationally accepted Bebras' activities will contribute to evaluation studies in this field.

1.1. The Significance of Developing CT

Nowadays, it is considered necessary for K-12 students to develop 21st-century skills to be successful in their lives (Partnership for 21st Century Skills, 2013). Acquiring such skills at a young age enables them to develop the flexibility to entertain multiple perspectives and produce different solutions to open-ended problems, which will support their success in professional and social lives (Chalkiadaki, 2018). Therefore, developing problem-solving skills is directly related to programming training and accordingly, the acquisition of CT in children.

Learning coding, an important digital literacy skill in today's digital world, is also a means of developing CT (Gretter & Yadav, 2016). Programming is not only about creating a computer program but also about structuring problems and producing appropriate solutions (Shin et al., 2013), which calls for computational and CT, such as reasoning, systematic thinking, and evaluation of evidence. Therefore, programming is interrelated with problem-solving, creativity, and CT, which is now seen as essential throughout K-12 education (Wong & Cheung, 2020). Many countries (Australia, the UK, Sweden, South Korea, the United States, and Macedonia) have included computer science topics in their primary school curricula, and some (Estonia, Finland, and Norway) have included programming education as a compulsory course in primary schools (Balanskat & Engelhardt, 2015; Hijón-Neira et al., 2017). Because educators globally accept 21st-century skills as necessary for children, many other countries have also started to provide programming education in the early grades (Wong et al., 2015; Manches & Plowman, 2017; Webb et al., 2017). Additionally, it has been stated that CT approaches will become the main topic in all disciplines and that advances in informatics will allow students to design strategies for problem-solving and control of solution steps in both the digital and real world. Weintrop et al. (2016) stated that activities that support critical thinking have been used in mathematics and science courses. The study emphasizes the importance of including CT in new-generation science standards as a basic scientific practice. It has been stated that there is a strong connection between coding, CT, and problem analysis strategies in different content areas such as Science, Technology, Engineering, and Mathematics (STEM) (Tsarava et al., 2017).

Providing programming activities, especially in primary schools, can greatly contribute to students' development of creativity skills (Denner et al., 2012) and CT at different stages of coding (DeJarnette, 2012), and debugging activities help students develop problem-solving strategies (Mishra & Yadav, 2013). Thus, primary school students should be given training to improve CT in addition to basic lessons such as reading, writing, and mathematics (Hsu et al., 2018), which can begin with teaching programming (Kong, 2016; Webb et al., 2017). However, there is a need to support research on how to develop suitable activities, how to teach CT-related subjects, and which activities should be used for K-12 (Tran, 2020; Rehmat et al., 2020).

1.2. Use of Plugged and Unplugged Activities in Programming Teaching

Different types of tools, such as plugged activities, block-based tools, and online applications, are used to help students acquire CT. Unplugged activities, such as block-based tools or online applications, include coding activities with a computer, and unplugged activities include coding activities without the use of digital tools (Brackmann et al., 2017). Currently, many block-based applications for children, such as Scratch, are used. These applications provide easy-to-use teaching opportunities for children with simple syntax and drag-and-drop features (Fessakis et al., 2013). Lin and Weintrop (2021) examined 46 block-based programs and specified areas where

these programs were used separately. Game and simulation design, data science, physical computing, and multimedia are the main applications. These block-based programs are the most suitable programs that can be used to teach programming to children. In particular, block-based coding tools, which are widely used to teach children programming, are easy to use (Papadakis et al., 2019). There are many block-based coding platforms for teaching programming to children. Code.org, such as Alice, Blockly Games, and Kodu (Kalelioğlu, 2015). Alice is a block-based environment in which students can create animations, interactive stories, and simple games while learning basic programming concepts (Costa & Miranda, 2017). Blockly Games are platforms that allow users to organize and interlock graphical elements or blocks (Shih, 2017). Code.org, which is based on object-oriented programming, is a coding platform that is widely used around the world and supported by many large companies such as Apple, Microsoft, Facebook, and Google, which provide support in 63 languages. On this platform, users carry out the assigned tasks gradually by dragging and dropping the code blocks to the workspace. After the students completed the task, the next task appeared. If a student is unsuccessful, a hint screen will appear, providing the help needed to solve the problem (Kale & Yuan, 2020).

Unplugged activities, in which students learn CT and computer science concepts without using computers, offer an alternative method for easy teaching of difficult subjects and are used for teaching programming, especially for children (Caeli & Yadav, 2020). In unplugged activities, role-playing to simulate programming processes can be carried out in such ways as bodily actions with objects, such as papers and cards, that allow students to explore fundamental ideas about programming (Aranda & Ferguson, 2018). Tsarava et al. (2018) found that third- and fourth-grade students can comprehend CT processes by engaging in unplugged activities. Although many studies on CT have been conducted for middle and high school students (Cheng, Wang, & Ritzhaupt, 2023), the current focus on CT activities for primary school students is still at the beginning (del Olmo-Muñoz et al., 2020). It was also stated that unplugged activities should be supported by plugged activities to develop students' CT. It seems more appropriate to provide unplugged and plugged activities together, particularly to improve programming skills. For students to understand the programming processes and what computers can do in this process, plugging activities should be undertaken. Algorithms must be implemented using a machine to test problem solutions and computational ideas (Denning, 2017; Caeli & Yadav, 2020). Unplugged activities should be prepared by relating them to real life with concrete examples and increasing student motivation. For this reason, it is appropriate to prepare activities that will attract the attention of primary school students and enable them to follow topics without getting bored with teaching programming (Duncon, 2019). It is quite common to use unplugged activities in many countries for teaching programming to children, both for this purpose and because of their cognitive level (Bell et al., 2009; Tsarava et al., 2018). These activities provide the development of an appropriate CT at the beginning of programming teaching. There are studies aiming to improve CT using only plugged activities (Yildiz Durak, 2018; Kalelioğlu & Gülbahar; 2014; Kale & Yuan, 2020), comparing plugged and unplugged activities (Polat & Yilmaz, 2022; Sigayret et al., 2022), and using both activities together (Lee et al., 2021; Saxena et al., 2020; Tsarava et al., 2017). However, because unplugged and plugged activities were seen as more appropriate to be given together to reinforce the topics, this study was planned in which both activity types were used together. At this point, it is important to determine the kinds of activities that should be used for programming teaching in primary school, which activities can develop CT skills, and what kind of planning should be done in the execution of the activities. This study will guide the planning of the process and which activities can be used in programming training for primary school students.

1.3. Purpose of the Study

As both "plugged" and "unplugged" activities can be used to develop programming and CT skills in primary school children, there remains a need for more research on how programming should be taught to them and with which activities. Accordingly, this study aims to find out the effect of applying both plugged and unplugged activities for teaching programming to primary school students on students' CT skills. Therefore, the research questions of this study are as follows.

RQ1. How does incorporating "plugged" and "unplugged" activities together in primary school students' learning affect their CT skills?

RQ2. What are the effects of programming teaching using "plugged" and "unplugged" activities together on the primary school development of students' programming skills?

2. Method

2.1. Research Design

In this study, a one-group pre-test and post-test quasi-experimental design was used. The purpose of this method is to determine the improvement of CT in students at the end of the training process, in which plugged and unplugged activities are applied together. Thus, the suitability of the training program for CT development of CT will be understood. The research process is illustrated in Fig. 1.
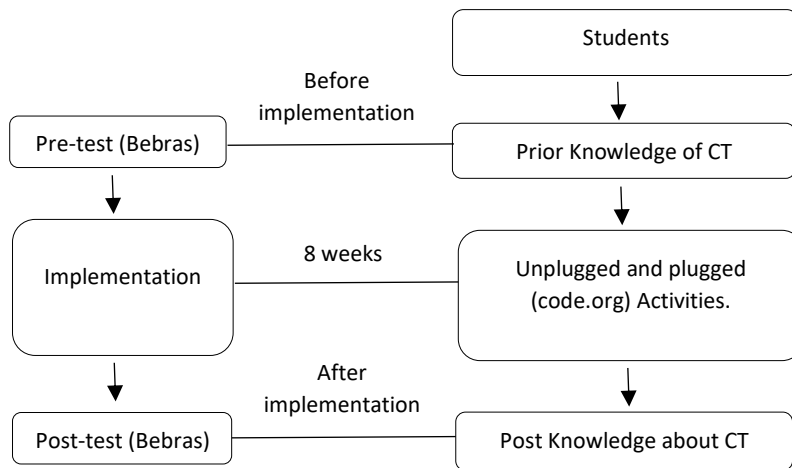

Figure 1. Research Study

During implementation, the students were first unplugged and then plugged. At the end of the training, the measurement tool at the beginning of the implementation was applied again to evaluate the progress of the group's CT. In this quasi-experimental design, measurements were made using the same tool before and after training. When the group's post-test and pre-test scores were compared, implementation was considered effective if the post-test scores were significantly higher (Creswell, 2012). Accordingly, the Bebras tasks were administered to the students before starting the implementation, and this process was repeated after the implementation was completed.

2.2. Sample

The sample was determined by convenience sampling, which is a purposeful sampling method. Convenience sampling is a type of non-random sampling that meets practical criteria such as easy access to the target group, geographical proximity, and accessibility at a certain time (Etikan et al., 2016). In this study, a close and accessible sample was chosen from the university where the researcher works. In addition, the researcher taught coding education to children at the university and could easily access the sample. The research was conducted with 18 primary school students (11 girls and 7 boys) who participated in programming training at a university in Türkiye (Table 1).

Table 1. Information of sample

Student	Age	Class	Girl	Boy
S1	8	3	✓	
S2	8	2	✓	
S3	8	3	✓	
S4	8	2		✓
S5	8	3		✓
S6	8	3		✓
S7	9	3		✓
S8	8	3	✓	
S9	8	2	✓	
S10	7	2		✓
S11	8	3	✓	
S12	8	3	✓	
S13	8	3	✓	
S14	7	2		✓
S15	8	3	✓	
S16	8	3	✓	
S17	8	3	✓	
S18	8	3	✓	

S: Student

The coding and robotics training in which the students participated for a fee was given at the research and application centre of a university as part of a program for primary "grade 1-4," middle "grade 5-8," and high school "grade 9-12" students. There are no prerequisites for such training programs. Applications made at Code.org are designed in such a way that each student can perform the activities in the education centre using a computer. The students participating in the study were from different primary schools and voluntarily sought programming and robotics training. The students had not previously studied computer science or programming.

2.3. Procedure

Coding and robotics training consists of different skill modules, including visual and robotic programming skill training. Before starting special skill modules, all primary school students first completed a module on general topics related to programming, critical thinking, logic, and algorithms.

In this introductory module, children learn to develop strategies for solving different problems, create problem-solving steps, create algorithms for the solution paths they determine, and write basic codes. The activities in this module guide students in developing strategies and steps to solve problems they encounter in their daily lives and mathematical and logical problems. In addition, students are prepared for subsequent modules, particularly visual programming and robotics, so they can integrate problem-solving steps, writing algorithms, and basic programming logic into their work in these modules. The current implementation was conducted in the first training module (Figure 2).

Figure 2. Students' implementations of unplugged activities

Within the context of the implementation, the main concepts and approaches to problem-solving, suggestions for solutions to problems in daily life, problem analysis, operators, using expressions and equations, creating algorithms, and flowchart components were taught. At the end of the initial unplugged activities, the course content (Course D), prepared for primary school students aged 7-11, was selected on code.org, and an account was opened

for each student. The students completed applications of sequencing, events, loops, and conditional sections in this course, and the completed applications were checked by the teacher (Figure 3).
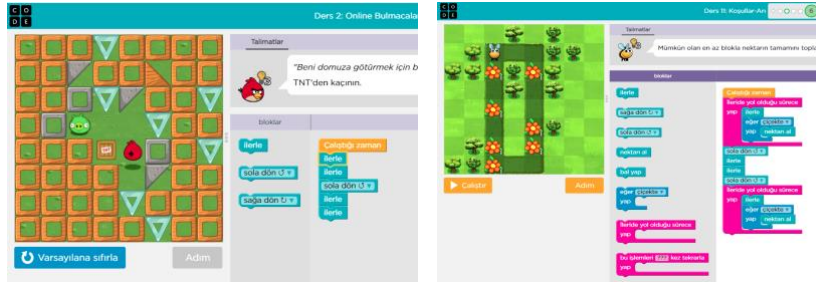

Figure 3. Plugged activities on Code.org

In the first module, a total of 24 hours of training was given for eight weeks. The training activities were conducted at the centre where the training was held. The planning of the training contents and some of the activities were prepared by the researcher who provided training at the centre. In addition, some of the Keşf@ Teacher Portal (www.kesfetprojesi.org) activities, implemented in collaboration with Google and the Ministry of National Education of the Republic of Türkiye (MoNE) in 2014, were used to prepare the contents. Before implementation, Bebras tasks were administered to the students to measure their CT (see 2.4). In the six-week part of the implementation, 23 unplugged activities were provided.
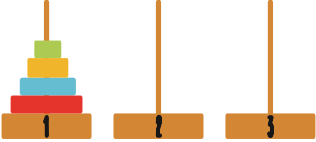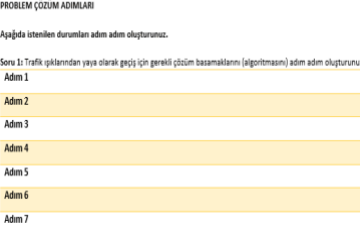
The content of the 12 activities prepared by the researcher was checked by a team of four experts in coding and programming education, all of whom had more than 15 years of experience in this field. Other activities were chosen from those prepared by the MoNE. The purpose of the researcher's planning of unplugged activities in the study is to support MoNE activities to improve programming and CT and to increase examples of unplugged activities that can be applied. In addition, code.org activities were conducted to support programming gains with computer applications and to convert an algorithm into program code and observe the results. The programming subjects, activities related to these subjects, their relationship with CT, and learning outcomes are shown in Table 2.
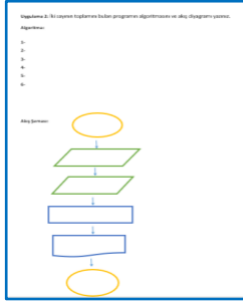
Table 2. Contents of programming training

Week	Activity Content	Activities	Programming Gains	Learning Outcomes
1	Identifying the Problem-Solving Strategies	"Cat-Dog-Mouse" Activity (MoNE)	• Abstraction • Algorithmic Thinking	• Recognizes problem-solving steps. • Analyses a problem.
		Tower of Hanoi (MoNE)		
		"Fishbone" Activity (MoNE)	• Dealing with Uncertainty • Algorithmic Thinking • Algorithm Design • Decomposition • Generalization	
		"What Should I Do Now" Activity (MoNE)		
2	Identifying the Problem-Solving Strategies	"Mixed situations" Activity (MoNE)		• Recognizes problem-solving steps. • Analyses a problem. • It offers solutions to problems in daily life. • Solves problems using appropriate solutions
		Tangram (MoNE)		
		Creating Problem Solving Steps-1 (Researcher)	• Algorithmic Thinking • Algorithm Design	
		Creating Problem Solving Steps-2 (Researcher)		
3	Algorithm and Strategy	"Karobot" Activity (Researcher)	• Sequencing	
		Navigating with a map (Researcher)		
		Writing Algorithms (Researcher)		
4	Flowchart Preparation	Creating a Flowchart (Researcher)	• Algorithmic Thinking • Algorithm design	• Writes an algorithm and creates a flowchart for this algorithm
		Writing Algorithms and Creating Flowchart (Researcher)		
		"Flowcharts mixed up" Activity (MoNE)		
5	Concepts Used in Programming "loops, conditionals, mathematical and logical operators"	Mathematical and logical operators (Researcher)	• Algorithmic Thinking • Logical questioning	• Gives examples of the use of operators in problem-solving. • Uses operators to solve a problem. • Understands Loops and Conditionals
		Loops and conditionals (Researcher)		
		"Choosing Occupation" Activity (Researcher)		
		"Colors of Nature" Activity (MoNE)		
		"Winning a Scholarship" Activity (Researcher)		
6	Concepts Used in Programming (variable-constant)	"Who Stays Here" Activity (MoNE)	• Decomposition • Data Analysis	• Explains the "variables", "constants", and "operations" used for problem-solving. • Explains data types.
		"Breakfast Habits" Activity (MoNE)		
		"Making a cake" Activity (MoNE)		
		"Variable-Constant in Our Lives" Activity (Researcher)		
7	Converting Algorithm to Program Code	Making applications on Code.org	• Creating program code	• Implementing applications involving algorithms, conditionals, and loops on the computer
8	Converting Algorithm to Program Code	Making applications on Code.org	• Creating program code	• Implementing applications involving algorithms, conditionals, and loops on the computer

First, problem-solving strategies were taught at the beginning of training. In teaching this subject, "Cat-Dog-Mouse", Tower of Hanoi, "Fishbone", "Mixed situations", Tangram, and "Creating Problem-Solving Steps" activities were used. In the 3rd week, activities were conducted to write algorithms and determine strategies for solving problems. In this context, writing algorithms for a problem given in daily life, "Karobot" and "Navigating with a map" activities were used. In the 4th week of training, flow-chart preparation education was provided. In this context, step-by-step algorithm writing for problem-solving in daily life and a flowchart of the algorithms were created. In the 5th week, exercises related to the concepts of mathematical and logical operators, conditionals, and loops, and their use were performed. In the 6th week, the concept of the variable constant was taught, and practices related to the subject were implemented. In the 7th and 8th weeks, the plugged activities on the code.org site were applied individually by each student. Examples of the applied activities are listed in Table 3.

Table 3. Relationship of activities with CT and implementation steps

Activities	CT and Programming Skills	Implementation
Activity name: "Cat-Dog-Mouse" Activity (MoNE)	• Abstractions • Algorithmic thinking	This activity is a different version of the "wolf-lamb-grass" problem. The students are asked to find the fewest solution steps that will enable the farmer and the objects to cross by boat.
Activity name: Tower of Hanoi (MoNE)	• Algorithmic thinking Decomposition • Generalization	Students are given towers of Hanoi in the classroom and asked to move the rings from the 1st column to the 3rd column. Students are asked to move first 3 rings and then 4 rings to the 3rd column, respectively.
Activity name: Tangram (MoNE)	• Algorithmic thinking	The tangram pieces were handed out to the students in the classroom. Students individually created shapes that were projected on the screen.
Activity name: Creating an Algorithm (Researcher)	• Algorithm design Sequencing	Students are asked to list the problem-solving steps and write the algorithm by giving problems from daily life. The given problems are in the form of describing a day at school, describing the activities carried out on the weekend, adding and subtracting 2 numbers, and describing the formation of day and night.
Activity name: Karobot (Researcher)	• Sequencing • Algorithmic thinking	Activity papers are distributed to the students. On the paper, they are asked to move the robot to the specified points in order. While writing the steps, they are asked to use the "forward-turn right-turn left" commands.

Activity name: Creating Algorithms and Flowcharts (Researcher)	• Sequencing • Algorithm design • Loops	Students are given examples of flowchart shapes with explanation of they do. Then they are asked to write the algorithms of the given problems and create a flowchart. Papers on which they can write the algorithm and flowchart for each problem separately are distributed to the students. The problems given are going to the market to buy the ingredients to make a cake, cross the road, getting food in a cafeteria, adding and subtracting two numbers, and going out in rainy weather.
Activity name: Career choice (Researcher)	• Algorithmic thinking • Conditionals • Mathematical and logical operators Decomposition	Activity sheets are distributed to the students, on which different occupations are shown and instructions are given. Using the "and," "or," and "not" operators, the students are asked to find the occupation described in the given statement.
Activity name: Winning a Scholarship (Researcher)	• Algorithmic thinking • Mathematical and logical operators • Conditionals	The students are given an activity paper with a table showing their age and scholarship status. Students are asked use operators to write an algorithm to identify 11-year-old recipients of scholarships and then to write the names given in the table that meet these criteria.
Activity name: Variable-Constant in Our Lives (Researcher)	• Decomposition • Generalization • Variable-constant	After receiving an explanation of "constant" and "variable," students, are given examples from daily life (school, shopping mall, hide and seek game) and asked to determine the variables and constants. Finally, students are asked to compare two numbers and write the algorithm necessary for finding the larger number, by specifying the variables.

2.4. Computational Thinking Test (Bebras)

Although there are various opinions on how to measure CT in children, there is still no consensus regarding this issue (del Olmo-Muñoz et al.,2020). Selby and Woollard (2013) state that CT development is determined by measuring CT sub-skills. In this study, Bebras tasks for primary school students, which have international validity, were used as data tools. Bebras is an international contest created in Lithuania to encourage K-12 students to learn about information technologies and develop CT (Cartelli et al., 2012; Dagiene & Stupuriene, 2016). The International Bebras Committee has many members, with 52 full members and 22 provisional members, and the number of members increases every year. Türkiye was also a full member of this community (https://www.bebras.org/community.html). Three committees have been established to manage the Bebras events. These committees include the National Bebras Organization, International Bebras Community, and Bebras Board. The National Bebras Organization is responsible for an all-year Bebras contest in a country. This committee has duties such as preparing and presenting new events, reviewing and evaluating events, selecting events from the international pool, translating them into the native, and arranging the challenges of the events. The selection of the

activities to be held in Türkiye, the translation of the activities into Turkish, and the organization of the activities are done by the faculty members in charge of this committee. Determining and scoring the difficulties of the activities were also performed by this board (Gülbahar et al., 2020). Bebras tasks are intended to measure the sub-skills of CT including "algorithmic thinking," "abstraction," "decomposition," "generalization," and "evaluation."

For this study, the activities were selected from the Turkish version of the problem set for the second and third grades. These activities can be solved by students who have no prior knowledge in the field of informatics, but they must have high-level critical thinking skills, such as making calculations and decisions, analytical thinking, and problem-solving. The tasks are related to CT, such as algorithms, condition and comparison, and pattern recognition. The tasks selected for this program included low, medium, and high difficulty levels. Low-difficulty tasks scored 6 points, medium tasks scored 9 points, and difficult questions scored 12 points. Two points were deducted for incorrect solutions to low-difficulty activities, three points for incorrect solutions to medium-difficulty activities, and four points for incorrect solutions to high-difficulty activities (Gülbahar et al., 2020). For this study, 12 activities corresponding to specific lesson contents were selected from the three levels of difficulty at the primary school level (Table 4).

Table 4. Contents of Bebras tasks used for pre & post-test

Task Number	Task Name	Difficulty Level	Description of the Problem	Programming Skills
1	Footsteps	Easy (6 points)	Students are requested to find a solution by comparing a defined pattern with other patterns. Similar processes are also used in the areas of pattern recognition and image detection in Informatics.	Abstractions and pattern recognition
2	Table Preparation	Easy (6 points)	Students are requested to find the order of the tableware. This problem involves decomposing and changing the order of different elements through layers.	Sequencing
3	Choosing Food	Easy (6 point)	Students are requested to establish a condition by finding similarities and differences in the food.	Conditionals Algorithm design
4	Vehicle Transfer	Easy (6 points)	Students are given priority rules regarding production priorities for vehicles and are requested to use these rules to order vehicles according to their priority ratings.	Sequencing Algorithm design are given to facilitate coordination Loops
5	Geometric Bracelet	Easy (6 points)	Students are requested to verify a solution concerning the order of the shapes on a bracelet.	Abstractions and pattern recognition
6	Faces and Glasses	Easy (6 points)	Students are requested to choose glasses suitable for their face shape according to the given condition.	Conditionals Mathematical and logical operators
7	Crazy Stars	Easy (6 points)	Students are requested to rank the stars by finding a common feature.	Abstractions and pattern recognition Sequencing
8	Ice cream	Easy (6 points)	Students are requested to find the ice cream order in the cone by the given condition.	Sequencing
9	Directions	Easy (6 points)	Students are requested to give directions for the shortest route from one point to another point	Algorithm design
10	Honeypot	Medium (9 points)	Students are requested to interpret the data in the visual given and make predictions to find the shortest way to reach the honeypot.	Algorithm design Conditionals
11	Clothes	Medium (9 points)	Students are requested to find the order of folding clothes according to the instructions	Sequencing Algorithm design
12	Similar Foods	Hard (12 points)	Students are requested to establish a condition by finding similar and different materials used in the given meals.	Conditionals Algorithm design

Before implementation, the Bebras tasks were applied to the students in writing, and no positive or negative feedback was given to the students about their answers and solutions. The students were not informed that these activities would be re-applied or that the Bebras tasks were applied. After the implementation process was

completed, the same activities were applied again in writing, and the students' overall scores on this activity and their CT subskills were calculated.

2.5. Data Analysis

Bebras tasks were used to measure the development of students' programming and CT. For this reason, the consistency of the scores obtained from the questions also expresses reliability (Golafshani, 2003). Cronbach's alpha coefficient for the scale was 0.782. Cronbach's alpha coefficient was above 0.7, indicating that the reliability of the measurement tool is high (George & Mallery, 2003).

The Wilcoxon signed-rank test was used to compare students' total scores on CT and their scores on programming knowledge (abstractions and pattern recognition, sequencing, algorithm design, and conditionals). The Wilcoxon signed-rank test was used to test the significance of the difference between the scores of the two related measurement sets in non-parametric measurements (Büyüköztürk, 2007). This test method was chosen because the sample size was insufficient for parametric tests, and the scores of the two related measurement sets were compared.

3. Findings

3.1. The Effect of Teaching Plugged and Unplugged Activities on Students' CT Skills

First, the change in the students' CT skills was evaluated by examining the pre- and post-test scores obtained from the Bebras scores. When looking at the changes in the total scores of 18 students in the Bebras tasks, it was observed that most students increased their scores (Figure 4).
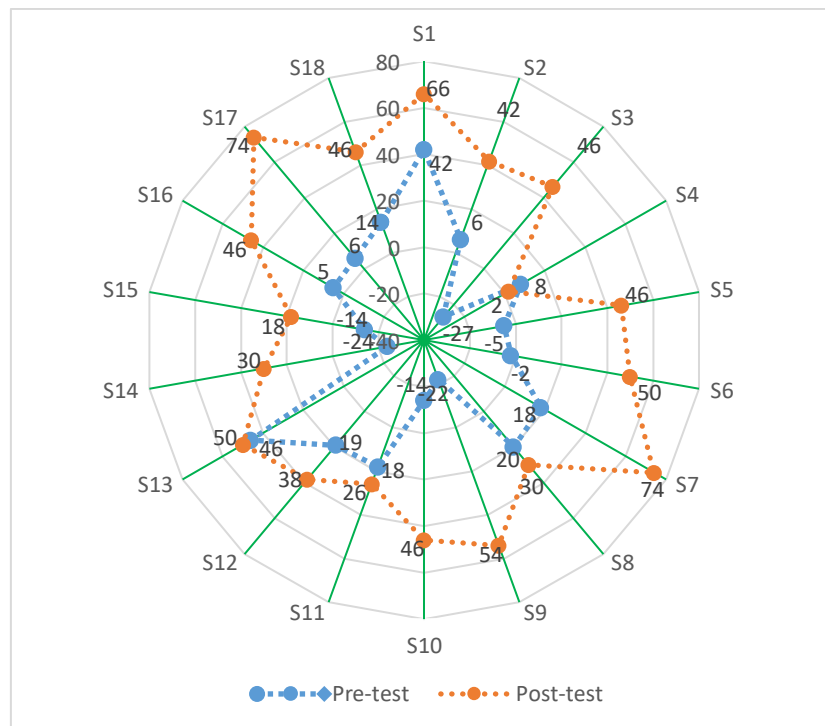
S: Student
Figure 4. Comparison of pre-and post-test scores

Figure 4 graphically illustrates the comparison of the students' pre-test and post-test scores, which shows that their post-test scores are higher than their pre-test scores, with an average increase of 38.47. The lowest total score that could be obtained from the questions was -30, and the highest score was 90. The lowest pre-test score was − -27 (S3), and the highest score was 46 (S13). The lowest post-test score was 2 (S4) and the highest score was 74 (S7, S17). Table 4 provides a statistical comparison of students' pre-test and post-test scores. The difference between

the pre-test and post-test scores of the group is shown in Table 5, and Figure 5 shows the change in pre-test post-test averages.

Table 5. Descriptive statistical results of CT

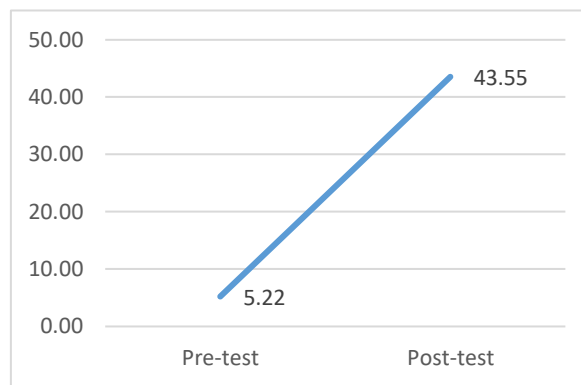| Group			Pre-test				Post-test		
	N	$\mathcal{X}$	S	Min	Max	$\mathcal{X}$	S	Min	Max
Experimental Group (Unplugged Activities- Code.org)	18	5.22	20.81	-27	46	43.55	18.26	2	74

Figure 5. Improvement in Bebras pre- and post-test score averages

Table 5 shows the min (-27) and max (46) scores obtained by the students from the pre-test and the min (2) and max (74) scores from the post-test. The results prove that the scores increased in favour of the post-test. The Wilcoxon signed-rank test was applied to determine whether the increase in the minimum and maximum scores was significant. By comparing the pre-and post-test scores, the effect of the activities on the CT skills of the students was determined due to the difference in the CT scores determined by the Bebras. The results of a Wilcoxon signed-rank test comparing the Bebras pre- and post-test scores are shown in Table 6.

Table 6. Results of Wilcoxon Signed-Rank Test regarding the CT skills

	Pre-test - Post-test	N	Mean Rank	Sum of Rank	z	p	Effect size (r)
Bebras Score	Negative ranks	1	2.00	2.00	-3.637*	0.000	0.857
	Positive ranks	17	9.94	169.00			
	Ties	0^c					

*Based on negative ranks.

The results of the analysis show that the post-test score (M = 43.55, SD = 18.26) was significantly higher than the pre-test score (M = 5.22, SD = 20.81), indicating that the implementation had a significantly positive effect on students' CT skills (z=-3,637, p<.05). It is seen that there is one student whose score decreased after the training and 17 students whose score increased. The average rank value of the scores of the students whose scores increased was determined as 169. A statistically significant difference was detected between the average success scores of the students before and after the training (p <.05).

In this study, the effect size of the comparison results of Bebras pre- and post-test scores was calculated. Effect size is useful because it provides an objective measure of the importance of the effect (Field, 2009). Calculating and interpreting effect size values in hypothesis tests increases the comprehensibility of the results (Büyüköztürk, 2010). Pearson's correlation coefficient r is an effect value coefficient. The r value takes a value between 0 (no effect) and 1 (perfect effect). The r value is evaluated independently of its sign. An r value of 0.1 is considered a

small effect, 0.3 is considered a medium effect, and 0.5 is considered a large effect (Field, 2009). The square of the r coefficient (r^2) expresses how much of the total variance it explains. The r^2 value shows how much of the change the independent variable explains on the dependent variable. In this study, the effect size of the comparison results of Bebras pre- and post-test scores was found to be r = 0.857 and the variance was r^2 = 0.734. This finding shows that the difference obtained has a large effect and 73% of the total variance is explained by the independent factor (coding training).

3.2. The Effect of Teaching Plugged and Unplugged Activities on Students' Programming Skills.

It was determined that training in which plugged and unplugged activities were implemented together improved the CT skills of the students. To determine which programming sub-skills the activities applied to the students developed, the pre- and post-test scores for the programming sub-skills in Bebras were compared. The Wilcoxon signed-rank Test was applied to compare the pre- and post-test scores for the programming sub-skills (Table 7).

Table 7. Results of Wilcoxon Signed-Rank Test regarding the plugged and unplugged activities on programming skills

Comparisons	Pre-test - Post-test	N	Mean Rank	Sum of Rank	z	p	Effect size (r)
Abstractions and Pattern Recognition Scores	Negative ranks	3[a]	5,83	17.50	-2.228[*]	0.026	
	Positive ranks	11[b]	7,95	87.50			0.525
	Ties	4[c]					
Sequencing Scores	Negative ranks	5[a]	5.80	29.00	-2.251[*]	0.024	
	Positive rank	12[b]	10.33	124.00			0.53
	Ties	1[c]					
Algorithm Design Scores	Negative ranks	1[a]	2.50	2.50	-3.616	0.000	
	Positive rank	17[b]	9.91	168.50			0.85
	Ties	0[c]					
Conditional Structures Scores	Negative ranks	0[a]	0.00	0.00	-3.523[*]	0.000	
	Positive rank	16[b]	8.50	136.00			0.83
	Ties	2[c]					

* Based on negative ranks.
[a] Pre-test score > post-test score
[b] Pre-test score < post-test score
[c] Pre-test score = post-test score

In Table 7, the Wilcoxon signed-rank test results of the analysis show a significant increase in the students' post-test scores after the implementation. When the pre- and post-test scores of the students on abstraction and pattern recognition (tasks 1, 5, and 7) were compared, a significant difference was found (z=-2.228, p<.05). Similarly, when the pre- and post-test sequencing (tasks 2, 4, 7, 8, and 11) scores of the groups were compared, there was a significant difference between the scores (z=-2.251, p<.05). Likewise, when students' algorithm design (tasks 3, 4, 9, 10, 11, and 12) and pre- and post-test scores were compared, a significant difference was observed between the scores (z=-3.616, p<.05). Finally, there was a significant difference in the students' pre- and post-test scores in the conditional structures (tasks 2, 6, 10, and 12) (z=-3.523, p<.05). When the effect size and variances of the pre- and post-test scores in the programming sub-skills were examined in this study, r = 0.525 and variance was found to be r^2 = 0.275 for abstractions and pattern recognition. This finding shows that the difference obtained has a large effect and approximately 28% of the total variance is explained by the independent factor (coding training). It was found that r=0.53 and variance was found to be r^2=0.28 for sequencing. This finding shows that the difference obtained has a large effect and approximately 28% of the total variance is explained by the independent factor. For algorithm design, r = 0.85 and variance was found to be r^2 = 0.726. This finding shows that the difference obtained has a large effect and approximately 73% of the total variance is explained by the independent factor. For conditional structures, r=0.83 and variance was found to be r^2=0.689. This finding shows that the difference obtained has a large effect and approximately 69% of the total variance is explained by the independent factor.

Accordingly, it is seen that education in which plugged and unplugged activities are implemented together contributes to the development of students in all programming sub-skills. The increase in students' post-test scores for all programming subskills is shown in Figure 6.
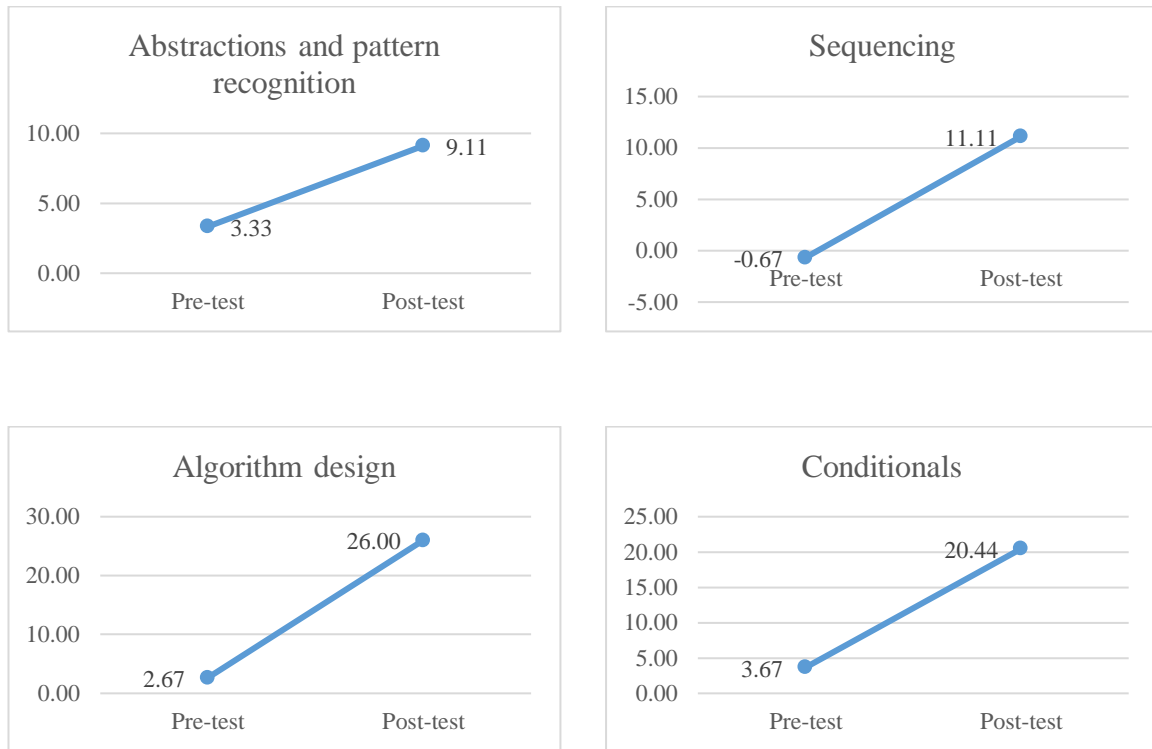
Figure 5. Improvement in scores for programming sub-skills

These increases in students' post-test scores on abstractions and pattern recognition, sequencing, algorithm design, and conditionals indicate that the students' programming knowledge levels in these areas increased after the implementation. The highest score increase was in the algorithm design (23.33), and the lowest score increase was in abstractions and pattern recognition (5.78).

4. Discussion and Conclusion

In this study, an eight-week training program for primary school students on programming and CT with "plugged" and "unplugged" activities was implemented. Unplugged activities prepared by the MoNE and designed according to these activities by the researcher were used. It has been stated in the literature that written, visual, and applied activities include sequencing, creating algorithms, visual presentations, video demonstrations, game activities, puzzles (Bell et al., 2009), finding solutions to daily life problems, map activity, drawing with instructions, and finding a route between two nodes (Brackmann et al., 2017) have been used.

Regarding the RQ1, it was observed that children's CT significantly improved. Tsarava et al. (2018) stated that performing plugged activities after unplugged activities not only improved CT skills but also increased students' motivation to learn coding. Olmo-Munoz et al. (2020) compared only unplugged activities with both unplugged and plugged activities among primary school students and concluded that the application of both will improve CT better than the application of only unplugged activities. The current study also confirms that the collaboration between plugged and unplugged activities helps improve primary school students' CT. It has also been stated that applying plugged activities to students after unplugged activities is beneficial not only in terms of students' CT, but also in increasing their motivation (del Olmo-Munoz et al., 2020; Tsarava et al., 2018). Due to their cognitive abilities, it is more beneficial for primary school students to start programming processes with unplugged activities, which are more fun and tangible, for the development of their CT. Because it is stated that the development of CT depends on the development of cognitive skills, therefore the challenges experienced in cognitive processes will negatively affect the development of CT (Ambrosio et al., 2014; Marinus et al., 2018; Tsarava et al., 2022). In this study, students mostly carried out unplugged activities in which they experienced the processes concretely, and after these activities, they had the chance to transfer what they learned to the digital environment with plugged activities. The improvement in students' CT in all sub-skills was an indicator of this.

Regarding the RQ2, the results of the comparison of the students' pre- and post-test scores showed they became better able to perform the applications of creating patterns, algorithms, loops, and conditionals correctly with fewer errors. The activities prepared for the curriculum in this study were matched to the students' cognitive levels to support their understanding of concepts they were learning for the first time, such as algorithms, flowcharts, loops, and conditional structures, and concrete examples from daily life were constantly provided. Hsu et al. (2018) stated that the content, methods, and approaches used in CT teaching should be adapted to learners' cognitive levels. However, it is difficult for children to understand and apply certain concepts. Although the success of the group increased in the post-tests, some individual students were unsuccessful. Concerning Piaget's cognitive theory (1962) development, abstract thinking skills develop after the age of 11 (Babakr et al., 2019), suggesting that the low or lack of increase in the scores of some students was due to their level of cognitive development. Although plugged activities are concretely associated with daily life, when students' scores from the post-test are examined, it is seen that they have difficulty interpreting advanced applications of the concepts of "*sequencing, algorithm design, abstractions, conditionals, mathematical and logical operators*" and associating them with plugged activities. It is possible that this is because students' abstract thinking skills are not fully developed.

Unplugged activities associated with daily life have drawn attention in the field of programming teaching. In this study, the activities prepared were similarly related to daily life and included written, visual, and in-class implementations. In the literature, it is stated that algorithms in unplugged activities can include comprehensive activities that present the procedures in our daily life as a sequence of steps. However, while learning about algorithms, it is also very important that they are designed in such a way that a machine can understand. Daily life examples such as describing addresses, preparing meals, or making a cake can be useful to illustrate and teach the algorithm. However, these agents should not be used alone. Unplugged activities gain meaning with plugged activities used together (Caeli, & Yadav, 2020). Plugged activities were applied using code.org. in this study. Although unplugged and plugged activities on code.org are powerful methods on their own, applying these methods alone poses the risk of finding solutions to real problems and not solving original problems. Therefore, unplugged activities and code.org activities were used together, and the equivalents of algorithms and solutions on the machine were observed.

5. Limitations and Suggestions

The study was limited to a small number of primary school students attending a university's coding training. A similar application can be applied to a larger group of students in primary school. In addition, plugged activities were applied after unplugged activities in this study. The effects of these applications on CT can be compared by conducting comparative studies in which only unplugged activities, only plugged activities, and both activities are applied together.

Although the activities implemented in the study were included in the MoNE curriculum, the activities prepared by the researcher were controlled by an expert team, and they were associated with increases in skills; thus, their effects could not be definitively determined. Therefore, more research is needed to determine how each of the unplugged activities used in the study contributes to the gains students achieve after implementation. Studies can be conducted on the evaluation of these activities separately, their role in providing the gains, and students' thoughts about these activities. No interviews were conducted regarding the practices and processes in which the students had difficulty or their thoughts. Through interviews, students' opinions were obtained about the questions on which they increased their scores less, or about the positive or negative practices they experienced regarding CT sub-skills and programming processes. Students' opinions on activities can be obtained in different studies.

The study has become an example of how to provide programming training by using "unplugged" and "plugged" activities together in primary school. The results also showed that this training improved the students' CT. However, no analysis was conducted of the teaching methods used in the study. Future studies can examine the effects of using different teaching methods in programming education. This study focused on the types of activities but did not focus on the types of teaching methods and their effects on the process. The literature states that there are many learning strategies for CT and programming teaching (Hsu et al., 2018). Studies can be conducted on the effects and contributions of different learning strategies, such as game-based learning, collaborative learning, and individual and group activities, on CT and programming teaching, especially for primary school children.

In addition to increasing educational activities and practices, measurement tools should also be developed to determine the development of students' programming and CT (Roman-Gonzalez et al., 2017). Although international Bebras tasks were used to measure programming and CT in the study, measurement tools associated with the gains need to be developed or increased for students of all age groups. In this study, it was difficult to

identify questions related to the subject or gain. Therefore, it is necessary to conduct more studies on the development of programming and CT measurement tools for primary school students.

Acknowledgments

References

Aranda, G. & Ferguson, J. P. (2018). Unplugged Programming: The future of teaching computational thinking? *Pedagogika, 68*(3). https://doi.org/10.14712/23362189.2018.859

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to "real" programming. *ACM Transactions on Computing Education (TOCE), 14*(4), 1–15. https://doi.org/10.1145/2677087

Ambrosio, A. P., da Silva Almeida, L., Macedo, J., & Franco, A. (2014). Exploring core cognitive skills of computational thinking. In *Psychology of programming interest group Annual conference 2014 proceedings, july, 25–24.* http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf. http://citeseerx.ist.psu. edu/viewdoc/download? doi=10.1.1.698.1911&rep=rep1&type=pdf.

Babakr, Z. H., Mohamedamin, P., & Kakamad, K. (2019, June). Piaget's Cognitive Developmental Theory: Critical Review. *Education Quarterly Reviews, 2*(3), 517-524. https://doi.org/10.31014/aior.1993.02.03.84

Balanskat, A. & Engelhardt, K. (2015). Computing our future computer programming and coding priorities, school curricula and initiatives across Europe. European Schoolnet. Retrieved from http://fcl.eun.org/documents/10180/ 14689/Computing+our+future_final.pdf

Bell, T, Alexander, J, Freeman, I., & Grimley, M. (2009). Computer science unplugged: school students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20-29. https://www.citrenz.ac.nz/jacit/

Büyüköztürk, Ş. (2010). *Sosyal Bilimler İçin Veri Analizi El Kitabı*, 11. Baskı, Pegem Akademi, Ankara.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017, November). Development of computational thinking skills through unplugged activities in primary school. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education, Nijmegen, Netherlands, November 8–10, 2017 (WiPSCE '17) (pp. 65-72). https://doi.org/10.1145/3137065.3137069

Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A historical perspective. *TechTrends, 64*(1), 29-36. https://doi.org/10.1007/s11528-019-00410-5

Cartelli, A., Dagiene, V., & Futschek, G. (2010). Bebras contest and digital competence assessment: Analysis of frameworks. *International Journal of Digital Literacy and Digital Competence (IJDLDC), 1*(1), 24-39. https://doi.org/10.4018/jdldc.2010101902

Chalkiadaki, A. (2018). A systematic literature review of 21st century skills and competencies in primary education. *International Journal of Instruction, 11*(3), 1-16. https://doi.org/10.12973/iji.2018.1131a

Ching, Y.-H., Hsu, Y.-C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *TechTrends, 62*, 563–573. https://doi.org/10.1007/s11528-018-0292-7

Cheng, L., Wang, X., & Ritzhaupt, A. D. (2023). The Effects of Computational Thinking Integration in STEM on Students' Learning Performance in K-12 Education: A Meta-analysis. *Journal of Educational Computing Research, 61*(2), 416-443. https://doi.org/10.1177/07356331221114183

Citt`a, G., Gentile, M., Allegra, M., Arrigo, M., Conti, D., Ottaviano, S.Sciortino, M., … (2019). The effects of mental rotation on computational thinking. *Computers & Education, 141*, 103613. https://doi.org/10.1016/J.COMPEDU.2019.103613.

Costa, J. M. & Miranda, G. L. (2017). Relation between Alice software and programming learning: A systematic review of the literature and meta-analysis. *British Journal of Educational Technology, 48*(6), 1464-1474. https://doi.org/10.1111/bjet.12496

Creswell, W. J. (2012). *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. Boston, United States of America: Pearson Education.

Dagiene, V. & Stupuriene, G. (2016). Bebras--A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in education, 15*(1), 25-44. https://doi.org/10.15388/infedu.2016.02

DeJarnette, N. K. (2018). Implementing STEAM in the Early Childhood Classroom. *European Journal of STEM Education, 3*(3), 18. https://doi.org/10.20897/ejsteme/3878

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020, June). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education, 150*, 103832. https://doi.org/10.1016/j.compedu.2020.103832

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10. 1145/2998438.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? Computers & Education, 58(1), 240-249. https://doi.org/10.1016/j.compedu.2011.08.006

Duncan, C. (2019). Computer science and computational thinking in primary schools. Doctoral Dissertation. University of Canterbury, New Zealand.

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* (January 2018), 191-202. https://doi.org/10.1016/j.compedu.2017.09.004

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics*, 5(1), 1-4. https://doi.org/10.11648/j.ajtas.20160501.11

Erümit, A. K., & Sahin, G. (2020). Plugged or Unplugged Teaching: A Case Study of Students' Preferences for the Teaching Programming. *International Journal of Computer Science Education in Schools, 4*(1), 1-14.

Fessakis, G., Gouli, E. & Mavroudi, E. (2013, April). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Field, Andy. (2009) Discovering Statistics Using SPSS, (Third Edition), Sage Publications Ltd., London

George D, & Mallery P. (2003). *SPSS for Windows step by step: A simple guide and reference.* 11.0 update (4th ed.). Boston: Allyn & Bacon.

Golafshani N. (2003). Understanding reliability and validity in qualitative research. *The qualitative report, 8*(4), 597-606. https://doi.org/10.46743/2160-3715/2003.1870

Grover, S. & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. E*ducational Researcher, 42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Gülbahar, Y., Kalelioğlu, F., Doğan, D., & Karataş, E.(2020). Bilge Kunduz: Enformatik ve bilgi-işlemsel düşünmeyi kavram temelli öğrenme için toplumsal bir yaklaşım. *Ankara Üniversitesi Eğitim Bilimleri Fakültesi Dergisi, 53*(1), 241-272. https://doi.org/10.30964/auebfd.560771

Hijón-Neira, R., Santacruz-Valencia, L., Pérez-Marín, D., & Gómez-Gómez, M. (2017, November). An analysis of the current situation of teaching programming in Primary Education. In 2017 International Symposium on Computers in Education (SIIE) (pp. 1-6). IEEE.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018, November). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

ISTE (2015). CT leadership toolkit. http://www.iste.org/docs/ctdocuments/ct-leadershipttoolkit.pdf?sfvrsn=4.

Jiang, S., & Wong, G. K. (2019). Primary school students' intrinsic motivation to plugged and unplugged approaches to develop computational thinking. *International Journal of Mobile Learning and Organisation, 13*(4), 336-351.

Kale, U. & Yuan, J. (2021). Still a new kid on the block? Computational thinking as problem solving in Code. org. *Journal of Educational Computing Research, 59*(4), 620-644. https://doi.org/10.1177/0735633120972050

Kalelioğlu, F. (2015, November). 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

Kalelioğlu, F., & Gülbahar, Y. (2014). The effects of teaching programmin*g via scratch on problem solving skills: A discussion from learners' perspective.* Informatics in Education, 13(1), 33–50.

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic J. Modern Computing, 4*(3), 583-596. https://www.bjmc.lu.lv/

Keşfet Projesi, (2018). Keşf@ Öğretmen Portalı, Kodlamayı Keşfediyorum. Retrieved from June 10, 2020, from https://kesfetprojesi.org/kodlamayi-kesfediyorum

Kirçali, A. Ç., & Özdener, N. (2023). A comparison of plugged and unplugged tools in teaching algorithms at the K-12 level for computational thinking skills. *Technology, Knowledge and Learning, 28*(4), 1485-1513. https://doi.org/10.1007/s10758-021-09585-4

Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, *3*(4), 377-394. https://doi.org/10.1007/s40692-016-0076-z

Lin, Y., & Weintrop, D. (2021). The landscape of Block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming. *Journal of Computer Languages, 67,* 101075. https://doi.org/10.1016/j.cola.2021.101075

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year olds. In *Proceedings of the 2018 ACM conference on international computing education research - ICER '18, august* (pp. 133–141). https://doi.org/10.1145/3230977.3230984

Manches, A., & Plowman, L. (2017). Computing education in children's early years: A call for debate. *British Journal of Educational Technology*, *48*(1), 191-201. https://doi.org/10.1111/bjet.12355

Mishra, P. & Yadav, A.(2013). Rethinking technology & creativity in the 21st century. *TechTrends, 57*(3), 10-14. https://doi.org/10.1007/s11528-013-0655-z

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

Piaget, J. (1962). *Play, dreams, and imitation in childhood*. New York, US: W Norton & Co.

Polat, E., & Yilmaz, R. M. (2022). Unplugged versus plugged-in: examining basic programming achievement and computational thinking of 6th-grade students. *Education and Information Technologies*, 1-35. https://doi.org/10.1007/s10639-022-10992-y

Popat, S. & Starkey, L. (2019, January). Learning to code or coding to learn? A systematic review. *Computers & Education, 128,* 365-376. https://doi.org/10.1016/j.compedu.2018.10.005

Partnership for 21st Century Skills, (P21). (2013). Framework For 21st Century Learning. Retrieved from November 06, 2020, from http://www.p21.org/about-us/p21-framework

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2019). The appropriateness of scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. In Early Childhood Development: Concepts, Methodologies, Tools, and Applications (pp. 797-819). IGI Global.

Rehmat, A. P., Ehsan, H., & Cardella, M. E. (2020). Instructional strategies to promote computational thinking for young learners. *Journal of Digital Learning in Teacher Education, 36*(1), 46-62. https://doi.org/10.1080/21532974.2019.1693942

Roman-Gonzalez, M., Perez-Gonzalez, J.-C., & Jimenez-Fernandez, C. (2017, July). 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.

S´aez-L´opez, J. M., Rom´an-Gonz´alez, M., & V´azquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two years case study using "scratch" in five schools. *Computers & Education, 97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate *computational thinking: An exploratory study in early childhood education. The* Asia-Pacific Education Researcher, 29(1), 55-66. https://doi.org/10.1007/s40299-019-00478-w

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definition. http://eprints.soton.ac.uk/id/eprint/356481.

Shin, S., Park, P., & Bae, Y. (2013). The Effects of an information-technology gifted program on friendship using Scratch programming language and clutter. *International Journal of Computer and Communication Engineering, 2*(3), 246-249. https://doi.org/10.7763/IJCCE.2013.V2.181

Shih, W. C. (2017, June). Mining learners' behavioral sequential patterns in a blockly visual programming educational game. In 2017 International Conference on Industrial Engineering, Management Science and Application (ICIMSA) (pp. 1-2). IEEE.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017, November). Demystifying computational thinking. *Educational Research Review, 22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Sigayret, K., Tricot, A., & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education, 184*, 1-14. https://doi.org/10.1016/j.compedu.2022.104505

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education, 162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research*, *57*(1), 3-31. https://doi.org/10.1177/0735633117743918

Tsarava, K., Moeller, K., & Ninaus, M. (2018). Training computational thinking through board games: The case of Crabs & Turtles. *International Journal of Serious Games, 5*(2), 25-44. https://doi.org/10.17083/ijsg.v5i2.248

Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017, October). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. In *European conference on games based learning* (pp. 687-695). Academic Conferences International Limited.

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

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when?. *Education and Information Technologies*, *22*(2), 445-468. https://doi.org/10.1007/s10639-016-9493-x

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). *Defining computational thinking for mathematics and science classrooms. Journal of science education and technology, 25* (1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

Wing, J. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, *25*(2), 7-14. https://www.learntechlib.org/p/183466/.

Wong, G. K. W. & Cheung, H.Y. (2020). Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming, *Interactive Learning Environments, 28*(4), 438-450. https://doi.org/10.1080/10494820.2018.1534245

Wong, G. K., Cheung, H. Y., Ching, E. C., & Huen, J. M. (2015, December). School perceptions of coding education in K-12: A large scale quantitative study to inform innovative practices. In *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 5-10). IEEE.

Yıldız Durak, H. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning, 25*(1), 179-195. https://doi.org/10.1007/s10758-018-9391-y