



# Unplugged versus plugged-in: examining basic programming achievement and computational thinking of 6th-grade students

Elif Polat<sup>1</sup> · Rabia Meryem Yilmaz<sup>2</sup>

Received: 9 December 2021 / Accepted: 7 March 2022 / Published online: 25 March 2022  
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

The aim of this study is to compare the effects of unplugged and plugged-in activities on academic achievement and computational thinking (CT) skills of sixth-grade students. Mixed-method research was carried out to explore whether there were differences between the groups, and to learn the students' opinions and experiences regarding the practices. For the quantitative phase, a quasi-experimental design was used with two groups. For qualitative phase, 12 students were interviewed. The participants were 84 sixth-grade students (between the ages of 10 and 11). The intervention was designed on a selection/construction of activities from seven different basic programming web platforms for the plugged-in group and the proposed national curriculum unplugged activities for the unplugged group. The results showed that significant differences between groups in academic achievement favoring the unplugged activities, but not in CT skills. Development in CT skills contributed to the unplugged group's academic achievement. In addition, qualitative results showed that the plugged-in group perceived their activities as fun and entertaining, but not exactly like a lesson; in contrast, the unplugged group did not experience anxiety or boredom since they perceived the activities as educational. CT explained 27 percent of the variance in academic achievement, suggesting that this skill is important for academic achievement in basic programming. These results suggest that students can improve their academic achievement and maintain the level of CT acquisition across unplugged and plugged-in activities. This article contributes to the body of knowledge about the positive impact of unplugged activities on teaching CT and programming fundamentals.

**Keywords** Unplugged activities · Plugged-in activities · Computational thinking · Intervention study · Mixed methods

---

✉ Rabia Meryem Yilmaz  
rkufrevi@atauni.edu.tr; rabia.kufrevi@gmail.com

Extended author information available on the last page of the article

## 1 Introduction

In today's world, it is necessary to go beyond the mere use of information technologies that have given shape to the future. Computer programming hereafter, programming, as a means of information technologies, is the current method of producing and seeking solutions to problems (Digital Promise, 2021, March). It's clear that today's younger generation must have programming skills, because their interaction with computers goes beyond routine tasks like posting blogs, writing documents, or archiving photos. In our increasingly computational world, students need to learn to create, modify, customize, and adapt tasks via computers. As one of the twenty-first century skills, the basic programming and sub-learning objectives related to these basic concepts and thinking skills inherent in programming have gained priority in today's world (Balanskat & Engelhardt, 2015). In primary school, students mainly need to learn computer basics, fundamental concepts to computing (Wing, 2006), and how computers process algorithms rather than memorizing the syntax of programming languages. As a concept that meets these needs, Wing (2014, para. 5) defined computational thinking (CT) as "the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer-human or machine- can effectively carry out". CT must be thought of more broadly than programming or computer science (Israel-Fishelson, & Hershkovitz, 2022; OECD, 2020). It includes skills such as problem solving, decomposing, pattern abstraction and recognition, algorithm design, algorithmic thinking, debugging, and evaluation which skills are fundamental components of programming basics, and that are needed in programming (Eight Reasons Why Every Child, 2021; Yadav et al., 2016). In same way, the relevant literature reports that these skills are directly related to performance in programming (Buitrago Flórez et al., 2017; Tsarava et al., 2017; Wing, 2006). Academic achievement in programming, which measures the acquisition of these key concepts mostly captured by the CT skill, is associated with CT. Thus, learning basic programming concepts and then to code is associated with acquiring the skill CT.

CT and concepts are indispensable prerequisites in today's world like other basic skills (International Society for Technology in Education (ISTE), 2021). CT, which has been incorporated into most national curricula, is not only fundamental to computer science, but is also needed independently to solve real-life problems (Bitesize, 2021; Gülbahar & Kalelioğlu, 2018). In addition, Kite et al. (2021) points out that code-centered one-size-fits-all present instructions of CT is not appropriate for all students and the risk that CT is not accessible to all students could be eliminated by integrating it into the core academic curriculum. In order to meet the need to expand the teaching and learning of programming and computer science, new pedagogical approaches are being explored that promote deeper conceptual understanding (Grover et al., 2019). To equip students with basic programming skills, block-based programming applications, other computer-based applications, and unplugged activities are mostly preferred (Brackmann et al., 2017; Merino-Armero et al., 2022). Recently, more and more studies

have shown that basic programming learning objectives can achieve through unplugged activities where the computer is not essential (Balanskat & Engelhardt, 2015; Bocconi et al., 2016; Brackmann et al., 2016; Kirçali, & Özdener, 2022). With the help of these positive developments, it is possible to teach the conceptual framework of programming through unplugged activities to students who may not have access to a computer under various circumstances, in order to prepare them for programming classes (Brackmann et al., 2017). Unplugged activities are activities that support the teaching and learning of computer science using cards, pieces of string, crayons, puzzles, games, and crosswords (CS Unplugged, 2021; Gülbahar & Kalelioğlu, 2018). Moreover, digital resources and tools used in education are costly (Merino-Armero et al., 2022), and they may not be accessible to all students (Kite, et al., 2021; Unnikrishnan et al., 2016), and also students may feel uncomfortable using computers (Bell et al., 1998). However, unplugged activities can be a way to avoid some other disadvantages (power outages or an Internet interruption, technical details of computers, the failure of hardware, radiation emissions, etc.) that can occur when using digital resources in the classroom. They prefer unplugged activities in countries where there are insufficient resources for computer education and if teachers' training in programming is inadequate (Brackmann et al., 2016; Pérez-Marín et al., 2018; Tsortanidou et al., 2022). Accordingly, the digital gap is still there. Therefore, it is possible to teach students in the lower grades the basics of programming concepts without computers (Merino-Armero et al., 2022). One of the key approaches we have for this is unplugged pedagogy. But the question of whether unplugged activities are interchangeable with plugged-in activities requires answers for many different aspects. This study seeks to examine effective strategies for teaching foundational CT practices and basic programming concepts. To this end, this study addresses if plugged-in activities can be replaced with unplugged activities based on the learning objectives.

## 1.1 Theoretical Background

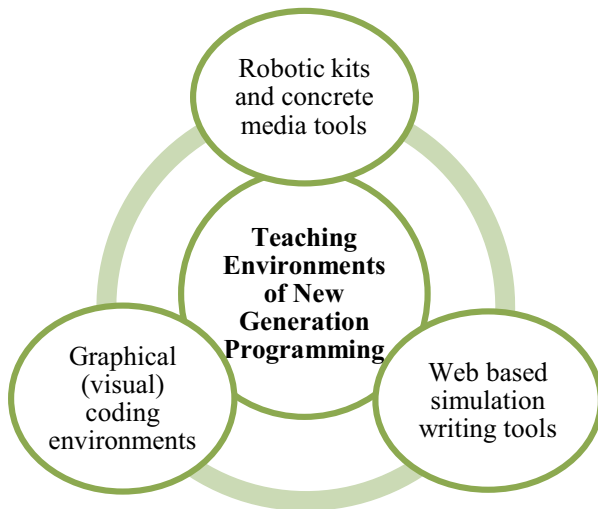
### 1.1.1 Computer Programming and Computational Thinking Skill

Computer programming used to be taught by teaching programming languages (e.g. C#, Python, Java, etc.) that are used to programme computers. However, since computers are used by everyone in all fields, beyond its importance for professionals who need to know programming languages to do programming work, but also to understand how computers work (Wing, 2006). Instead of teaching specific programming languages, teaching programming logic, computational thinking, algorithmic thinking and problem-solving skills have become increasingly important (Ramadhan, 2000). In a review study by Falkner and Vivian (2015) of programming education curricula in Australia, it found that the focus is more on course content than on pedagogy, while pedagogical support is neglected. Similarly, Lahtinen et al. (2005) reported that it limited novice programmers to superficial “line-by-line” programming rather than using meaningful programming structures that required

computational thinking skills relevant to programming. Although computer education offers a fun and effective learning experience, there is no single pedagogical solution for all grade levels (Garneli et al., 2015). While learning programming, when a programmer transition from process learning to synthesize creative programming in an interdisciplinary way, it will create a difference in terms of the digital ability needs of today (Futschek, 2006; Resnick & Siegel, 2015; Romero & Barberà, 2014). Del Olmo-Muñoz, et al. (2020) claimed that a mixed approach that mixes unplugged and plugged-in activities is better for the early years of primary education than only through plugged-in activities. In this way, besides text-based programming languages, there are some computer tools that are widely used in both programming and programming instruction. One such classification is made by BuitragoFlórez et al. (2017) into three subgroups, as shown in Fig. 1.

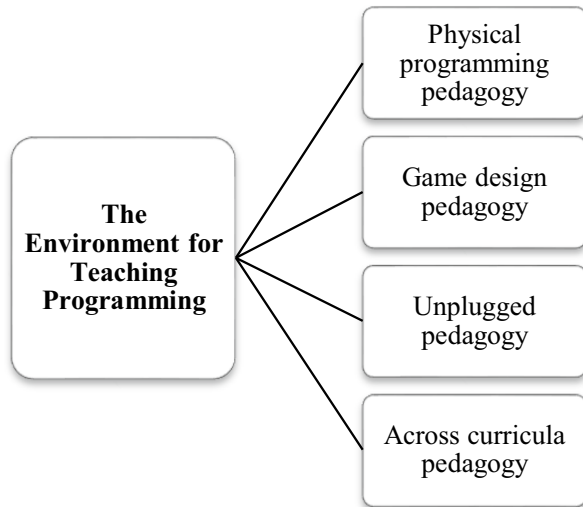
*Robotic kits and concrete media tools* (Arduino, GogoBoards etc.), *web-based simulation writing tools* (Agentsheets and Agentcubes) and *graphical (visual) programming environments* (Scratch, Alice, Kodu, Greenfoot, LightBot, etc.) can be mentioned as examples. Newly developed tools can form additional groups as well as be integrated into existing ones. With the increasing knowledge of visual programming tools and the growing importance given to teaching programming related concepts to students at younger ages, these studies are even conducted with pre-school children (Fessakis et al., 2013). In addition, a review study of information and computer technology (ICT) instructional pedagogy by Waite (2017) indicates that learning models and instructional techniques need to be identified at the initial stage and then the environment where the instruction will take place are listed under four headings in Fig. 2 (Waite, 2017).

With Papert (1980) starting to use some tools that inspired him within the scope of physical programming pedagogy, robots that could be programmed started to be



**Fig. 1** A classification of the tools used for teaching programming

**Fig. 2** The characteristics of the environment for teaching programming



used in the teaching of programming. Physical programming is a process where software and hardware come together and students make "repairs" with wires, sensors, LEDs, and other elements (Meyers, 2019). On the other hand, researches on unplugged computer science have attracted as much attention as alternative tools (Bell et al., 2009; Brackmann et al., 2017; Grover et al., 2019). These are kinesthetic opportunities designed to help students relate complex concepts to their own lives and internalize them (CS Fundamentals Curriculum Guide, 2021). They increase students' motivation, thinking skills, imagination, and computational thinking (Nishida et al., 2008; Sun et al., 2021). In addition, unplugged activities in unplugged implementation pedagogy familiarize students with most basic concepts that are distinct from distracting features and technical details of computers such as game playing and the Internet surfing during instruction (Computer Science Unplugged, n.d.). Indeed, as recommended by the Computer Science Unplugged movement, direct computer programming is not essential for developing computational thinking concepts in students (CS Unplugged, 2021). In the research carried out with plugged-in and unplugged activities in two groups, the increase in CT skills level of the unplugged group in the first phase of instruction was more remarkable (Del Olmo-Muñoz, et al., 2020). There are also recommendations to go beyond text-based programming languages when teaching programming to secondary school students (Szlávi & Zsakó, 2006). As indicated by research studies, everyday practical materials for teaching basic programming should be developed that take into account the development of learners at each stage (Lister, 2016).

Even though programming is at the heart of data processing and computer science, they should teach it to develop CT skills, not in terms of passive application of syntax (Buitrago Flórez et al., 2017; Futschek, 2006). In a review study by Lye and Koh (2014) on teaching and learning the skill of CT, they noted students should not only practice this skill but also reflect on it. The statements about CT in the literature, a fundamental skill for everyone, anticipate new learning objectives that

include CT for early stages in schools (Wing, 2008). CT refers to the idea of finding a solution to a problem by following an algorithmic solution after examining the problem and identifying its variables and invariants (Wing, 2006, 2010). It is not only essential for teaching programming skills, but also includes cognitive understanding, which is crucial for computer literacy (Buitrago Flórez et al., 2017; Tsaravaet al., 2017), as well as all basic concepts and applications (Korucu et al., 2017). Digital Promise (2021, March) stating that CT is a skill set that need to be integrated with the core academic subject matter. Due to these characteristics, CT, which is increasingly used in many fields, is attracting the attention of governments and educational institutions, which are working towards including it in primary school curricula (Bougot-Robin et al., 2016; Tsarava et al., 2017; Ministry of National Education (MoNE), 2018a, p.27; Wing, 2014). Contrarily, CT is still mystical to many educators (Digital Promise, 2021, March). Its practical applications are introduced even in early childhood (Lee et al, 2022). Therefore, it is important to know more about which core school subjects are relevant to CT for primary school students. For this reason, the correlation and regression between CT and the acquisition of basic programming skills, as well as the extent to which the components of CT are related to the acquisition of basic programming skills and their contribution to the formation of academic achievement scores, are considered significant data.

### 1.1.2 Related Studies on Unplugged and Plugged-in Activities in Teaching Programming

Learning computational concepts and developing process thinking will help students acquire more sustainable knowledge and skills rather than the rapidly changing technology (Strnad, 2018) because considering that students can carry out functions by using their own mental tools like a computer can enable them to acquire CT via unplugged activities (Kim et al., 2013). In addition, rather than aiming to teach programming to children, by using unplugged activities or challenging class activities toward CT, it is possible to obtain real positive effects at primary (del Olmo-Muñoz et al., 2020) and secondary school levels even with teachers with low level of teaching experience (Moreno-León & Robles, 2015; Ung et al. (2022); through a preliminary investigation, they revealed that teachers do not seem to know what computational thinking means in general, similar the study done by Tsortanidou et al. (2022). In research by del Olmo-Muñoz et al. (2020) concluded that unplugged activities seem beneficial in terms of development of CT skills and motivation of 84 s-grade students. They used activities extracted from Code.org courses for instructional intervention.

A study comparing high school students on block-based and text-based platforms found that students on the former platform developed higher levels of learning objectives and more positive attitudes toward programming courses (Weintrop & Wilensky, 2019). In a study by Duncan et al. (2014) was found that 28 of the 47 introduction-to-programming courses used the block-based platforms. Similarly, Bers (2018) stated that ScratchJr is the most popular free introductory programme for kids. On the other hand, positive results regarding the development of the CT were obtained in studies related to project based Arduino educational robot

applications (Karahmetoğlu & Korkmaz, 2019), Minecraft-based coding activities (Kutay & Oner, 2022), programmable floor robot named Bee-Bot (Angeli & Valanides, 2020), the application of unplugged activities (del Olmo-Muñoz et al., 2020; Merino-Armero et al., 2022), the Scratch application (Bers, 2018; Rodríguez-Martínez et al., 2020), and the learning game, Zoombinis (Asbell-Clarke et al., 2020). As an example, in a study by Yallihep, & Kutlu (2020) a significant development was revealed in the achievement levels regarding programming concepts of 5<sup>th</sup> grade students studying in the Lightbot platform when compared to those whose studies were strictly guided by the MoNE ITS (Information Technology and Software) curriculum (MoNe, 2018b). It seems that an attempt was made to teach the basic programming concepts with only one tool, and thus to evaluate only one tool.

On the other hand, in terms of both computer programming and basic ICT life skills, CT is a skill that needs to be developed at a young age (Conde et al., 2017; ISTE, 2021). Thus, the methods and approaches that can be effective in teaching this skill should be studied according to the principles of accessibility, cost-effectiveness and suitability for students, and lessons should be derived from the results. It can be observed that unplugged activities in lower grades are significantly beneficial at lower costs when access to computers is limited and when disadvantages related to technical failures and some computer-related problems (child screen time out, the distraction of attention, radiation emission, etc.) are to be avoided, and even in conditions where teachers do not have sufficient programming skills (Moreno-León & Robles, 2015; Nishida et al., 2008). In addition, some studies have found positive results on unplugged activities in relation to the studied variables such as motivation, CT, gender, academic achievement (del Olmo-Munoz et al., 2020; Delal, & Oner, 2020; Nishida et al., 2008; Kırçali & Özdener, 2022). By reviewing studies, it is observed that unplugged activities were developed, introduced (Bell et al., 2009; Sendurur, 2019) adapted to the grade level in the subject (Meyers, 2019; Nishida et al., 2008), or their effect was investigated alone (Delal & Oner, 2020) or compared to a computer application (del Olmo-Munoz et al., 2020; Kırçali & Özdener, 2022). It cannot be guaranteed that all learning objectives of a curriculum or syllabus will be achieved using block-based or visual programming tools with their unique interfaces, activity flows, and scopes, as the CT involves different thinking skills (CS Unplugged, 2021; Yadav et al., 2016). CT exploration is limited in Primary Education.

The aim of the present study was to compare the unplugged activities with their equivalent plugged-in activities used in teaching programming basics in 6th-grade information and software technologies (IST) courses to find their effect on students' academic achievement in basic programming, computational thinking, and opinions. To this end, the answers to the following research questions were sought:

1. Is there a significant difference between the pre and post achievement tests in basic programming within the plugged-in group and the unplugged group?
2. Is there a significant difference between plugged-in and unplugged groups in terms of academic achievement in basic programming?
3. Is there a significant difference between the plugged-in and unplugged groups in terms of computational thinking skills?

4. Is students' computational thinking ability a predictor of academic achievement in basic programming?
  - 4a. Is each sub-dimension of the computational thinking scale a meaningful predictor of academic achievement in programming?
5. What are the opinions of the participating students about the plugged-in or unplugged activities that were conducted in their group?

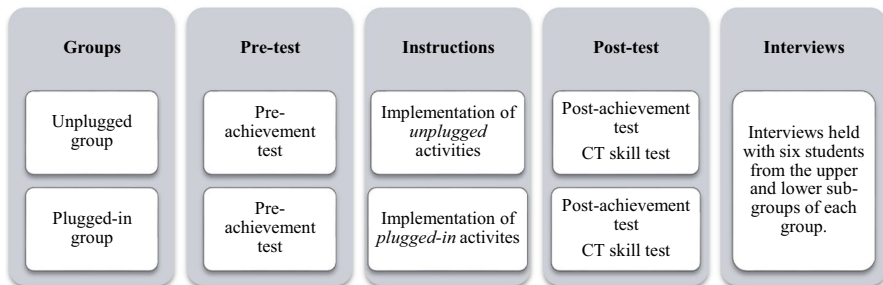
## 2 Method

### 2.1 Design

In this study, sequential-explanatory research design, one of the mixed-methods, was used since the quantitative data were collected and analyzed first, and then the results were examined and used to plan the qualitative phase. The purpose of this design is to use qualitative data to further explain quantitative results (Creswell, 2013). In this study, for the quantitative phase, a quasi-experimental design was designed with two groups. It was conducted to examine whether there were differences between the groups in computational thinking skills and academic achievement of basic programming. A pre-test academic achievement was administered to test the equivalence of the two groups before intervention. For qualitative phase, 12 students were interviewed, three students from the upper and lower subgroups of each group. Interviews were conducted to learn students' opinions of the implementations and to reveal their experiences. The research design processes of the study are shown in Fig. 3.

### 2.2 Sample

The study was conducted with a total of 84 participants consisting of 42 girls and 42 boys 6<sup>th</sup>-grade students (between the ages of 10 and 11) in two different schools. The sample was obtained through non-random convenience sampling method while the classes were randomly assigned to the groups. Table 1 shows the homogeneity



**Fig. 3** Details regarding the research design employed in the study



**Table 1** Demographic Characteristic of the Participants

Groups	Students responding to the quantitative scales			Students responding to the interview protocols				
	Female	Male	Total	Female		Male		Total
				Lower group	Upper group	Lower group	Upper group	
Unplugged	23	21	44	1	1	2	2	6
Plugged-in	19	21	40	2	2	1	1	6
<i>Total</i>	42	42	84	3	3	3	3	12

of the groups. According to Piaget (1977), children between the ages of seven and eleven can solve problems that relate to concrete objects, but not abstract concepts or phenomena. In fact, unplugged activities are more related to concrete materials and concrete phase of child development. Moreover, the "unplugged" activities of the national curriculum for 6th grade were designed for this stage of development and compared with the "plugged-in" activities. Accordingly, sixth grade students are the participants in this study.

## 2.3 Instruments

Based on the aim of the study, three different instruments were used to collect data. While the academic achievement test and interview protocol were developed by the researchers of the present study, the Computational Thinking Skills Scale was developed by Korkmaz et al., (2015a, b).

### 2.3.1 Academic Achievement Test

For the development of the academic achievement test, the purpose of the test was initially identified, the learning objectives were specified, and subsequently the table of levels of objectives was prepared, accordingly the test items were written, revised and the pilot test form was prepared. The pilot test was conducted with 104 6th grade students who responded to the preliminary implementation form, which consisted of 33 questions. There are studies in the literature on the development of achievement tests with as many or fewer respondents as in this study, e.g. Djambong et al. (2018), Lin (2018), and Marie and Sreekala (2015). It was conducted on an easily accessible sample near the teacher, one of the authors. The number of questions was limited by the amount of testing time available for nearly 10 years old students, 40 min. Item difficulty and item discrimination analysis in the achievement test was calculated to make the test items discriminant for high and low scorers. Based on the preliminary item analysis results of the pilot study, 5 items were removed from the form; the mean score of the 28 items remaining in the form was 15.67. The coefficients of skewness and kurtosis of the data were within acceptable ranges, 0.31 and -0.52, respectively (Tabachnick & Fidell, 2013). The standard deviation was found to be

5.49, and a Cronbach alpha reliability coefficient of 0.82 indicated that the test was reliable.

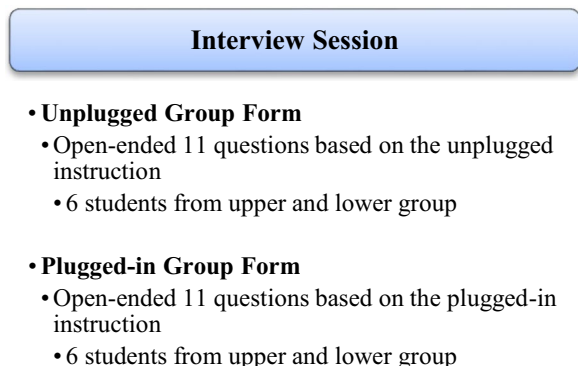
### 2.3.2 The Computational Thinking Scale

The original version of the computational thinking scale, which was adapted to the secondary school level in this study, was developed by Korkmaz et al. (2017). The final version of the scale consists of 5 sub-dimensions with a total of 22 items and is based on a 5-point Likert scale, with “5” indicating the most positive and “1” the most negative. In the present study, the reliability coefficients of the post-test administered to the unplugged group and the plugged-in group were found to be 0.82 and 0.71, respectively.

### 2.3.3 Interview protocols

Two open-ended interview sheets were prepared for each group with 11 questions related to the activities in their group (See in Fig. 4). The aim of the interview questions was to conduct an in-depth analysis of the students’ opinions regarding the challenges they encountered during class, their expectations, feelings, performance levels, anxiety and satisfaction, the methods they used in solving problems, the solution methods they applied to algorithms and problems, and their self-assessment of their own learning. In accordance with the purpose of the study, the design of the interview forms was made using the literature and taking into account that the questions must not be predictable, must be easy to understand without leading to short answers, must not provoke a negative reaction from the respondent, and must encourage a detailed and explanatory answer (Yıldırım & Şimşek, 2016). It is important that each question focuses on a single dimension so that in-depth responses can be obtained (Creswell, 2013). Since students have difficulty giving long answers, more questions were written. Then, it was ensured that two academics reviewed the form and two students who were not part of the sample responded.

**Fig. 4** Interview procedure of groups



## 2.4 Procedure

The intervention process of the study was conducted in the 6<sup>th</sup>-grade ICT course in four different classes over 7 weeks by the same teacher. It extended over a total of 14 lessons—2 h per week. The steps followed in the intervention process are shown in Fig. 5.

Here, the basic programming learning objectives of the intervention were part of the curriculum of the 6th grade MoNE curriculum (MoNE, 2018b). Its unplugged lesson plans were used in the unplugged group as teaching materials for the study. Besides, some of the basic programming teaching tools (code.org, scratch, compute.it; lightBot) and some other applications that enable computer interactions (kahoot.it; draw.io; learningapp.org) were used in the plugged-in group. During the 7-week intervention process, unplugged and plugged-in activities that differ from the common presentation and narrative are listed by objectives for both groups in Table 2. Based on these objectives, plugged-in activities have been developed or selected to correspond to the unplugged activities in the program, and presented weekly in Table 2.

The activities, B1 worksheets, that were performed in week 5 and week 7 in the unplugged group is presented as an example (See Fig. 6). These activities are part of MoNE curriculum of information written by Gülbahar Güven et al. (2018).

The other common and different implementation processes in the groups are shown in Fig. 7.

The unplugged group followed both common and unique instructions in the traditional classroom setting, as listed in Fig. 7; activities were either done in groups or pairs or individually in class or completed as homework. In the plugged-in group, computer applications were used to address the targeted learning objectives (See in Table 2). These included various block-based programming applications as well as Draw.io (2021), a drawing platform, and a platform for preparing interactive educational materials, as well as other platforms such as LearningApp and Kahoot. In the plugged-in group, the computer applications were also used

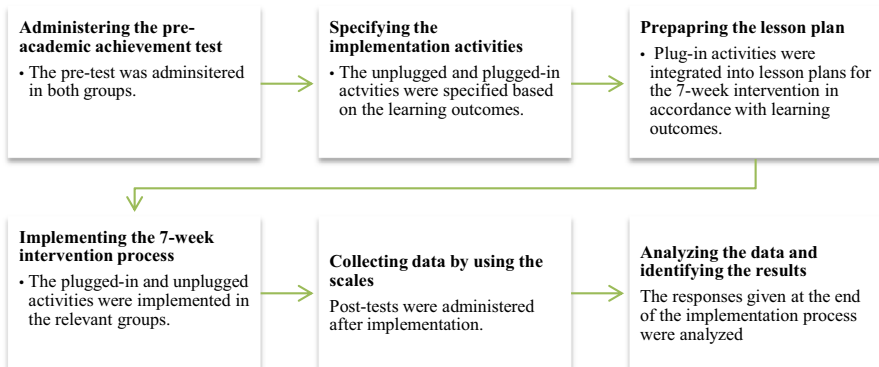


Fig. 5 The implementation process of study

**Table 2** Plugged-in and Unplugged Teaching Activities and Practices Used for Instruction Intervention

Weeks	Plugged-in Activities*	Learning objectives
1	A1—Presentation: An entertaining bridge from mathematics to computers B1—Animation: How does a computer work? C2—Poster: Types of data Plugged-in activity: <a href="https://ozgurseremet.com">https://ozgurseremet.com</a> ( <a href="https://ozgurseremet.com/ulke-veri-tablosu-oyunu/">https://ozgurseremet.com/ulke-veri-tablosu-oyunu/</a> )	6.5.1.1.1. Collect data and classify them according to their types 6.5.1.1.10. Discuss the relationship between mathematics and computer science
2	A1—Song: I grow A2—Presentation: Constant-Variable Confusion Plugged-in activity: <a href="https://code.org">Code.org</a> ( <a href="https://code.org/">https://code.org/</a> )	6.5.1.5 Develop an algorithm to solve a problem 6.5.1.6. Tests the solution of an algorithm 6.5.1.8. Correct a faulty algorithm so that it works correctly 6.5.1.10. Discuss the relationship between mathematics and computer science
3	For Simple and Complex Problems Plugged-in activity: Kahoot ( <a href="https://create.kahoot.it/">https://create.kahoot.it/</a> )	6.5.1.2. Uses constants and variables when solving problems 6.5.1.3 Divides a problem into subproblems
4	For making Algorithms Plugged-in activity: LearningApp ( <a href="https://learningapp.ps.org/">https://learningapp.ps.org/</a> )	6.5.1.2. Uses constants and variables when solving problems 6.5.1.3 Divides a problem into subproblems
5	For making algorithm procedure: Draw.io ( <a href="https://www.draw.io/">https://www.draw.io/</a> )	6.5.1.2. Uses constants and variables when solving problems 6.5.1.3 Divides a problem into subproblems 6.5.1.5 Develop an algorithm to solve a problem
6	For correcting mistakes: Scratch ( <a href="https://scratch.mit.edu/">https://scratch.mit.edu/</a> ) Code.org ( <a href="https://code.org/">https://code.org/</a> )	6.5.1.8. Correct a faulty algorithm so that it works correctly 6.5.1.9. Generalize the solution of the problem to similar problems
7	Compute.it ( <a href="http://compute-it.toxiccode.fr/">http://compute-it.toxiccode.fr/</a> ) LightBot ( <a href="https://lightbot.com/">https://lightbot.com/</a> )	6.5.1.4. Uses basic functions in the problem-solving process 6.5.1.7. It tests different algorithms and selects the fastest and most accurate solution
<b>Weeks</b>	<b>Unplugged Activities</b>	<b>Learning objectives</b>
1	A1—Presentation: An entertaining bridge from mathematics to computers B1—Animation: How does a computer work? C1—Worksheet: <i>Country Data Table</i> C2—Poster: Types of data E1—Worksheet: <i>Determination and evaluation of data types</i>	6.5.1.1. Collect data and classify them according to their nature 6.5.1.10. Discuss the relationship between mathematics and computer science
2	A1—Song: I grow A2—Presentation: Constant-Variable Confusion B1—Worksheet: <i>Who stays here? C1—Flowcharts: Is there a problem?</i>	6.5.1.2. Uses constants and variables when solving problems

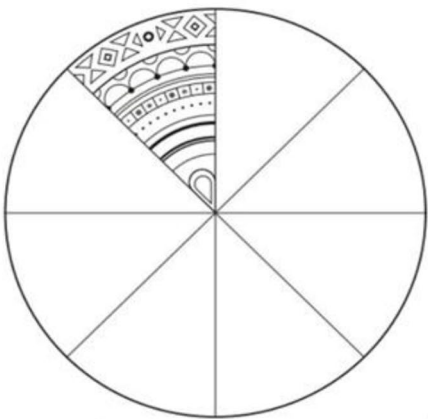
**Table 2** (continued)

Weeks	Plugged-in Activities*	Learning objectives
3	A1— <i>Worksheet: Simple and Complex Problems</i> B1— <i>Poster: Drawing Machine C1—Worksheet: Gülse's Story</i>	6.5.1.3. Divides a problem into subproblems 6.5.1.4. Uses basic functions in the problem-solving process
4	B1— <i>Worksheet: Solving Problems with Algorithm</i>	6.5.1.5 Develops an algorithm to solve the problem 6.5.1.6. Tests the solution of an algorithm
5	A1— <i>Worksheet: Labyrinth of Glory</i> B1— <i>Worksheet: The Shortest Path</i>	6.5.1.5 Develops an algorithm to solve the problem 6.5.1.6. Tests the solution of an algorithm 6.5.1.7. It tests different algorithms and selects the fastest and most accurate solution
6	A1— <i>Worksheet: Let's sort it!</i> B1— <i>Worksheet: Let's Notice the Difference</i> C1— <i>Worksheet: What kind of is This Story? D1—Worksheet: Extract Rice Stone</i>	6.5.1.8. Correct a faulty algorithm so that it works correctly
7	B1— <i>Worksheet: Let's Draw Mandala</i> C1— <i>Worksheet: A Little Math Now</i>	6.5.1.9. Generalize the solution of the problem to similar problems

\* Sample weekly lesson plans for plugged-in group can be found in the supplemental files.

## LET'S DRAW A MANDALA

The mandala pattern you have is only one part of the mandala. I want you to complete this mandala to form a Whole. After completing the pattern at the end of the activity, you can color your mandala as you wish. Examine the patterns carefully before drawing them. Discover the pattern, symmetry and harmony in the patterns. Then you can start your drawing.



6.2.7. B1 Lets Draw a Mandala

---

## Shortest Path

Below you will find 3 different directions you will use to get from A to B. Examine the algorithms under each instruction. Determine the shortest and longest path using the algorithms.

**1. Walk from A to B, stopping at green and blue.**

a) Go 2 units east, 1 unit south, 3 units east, 1 unit south, 3 units west, 1 unit south, 3 units east.

b) Go 3 units south, 2 units east, 2 units north, 3 units east, 2 units south.

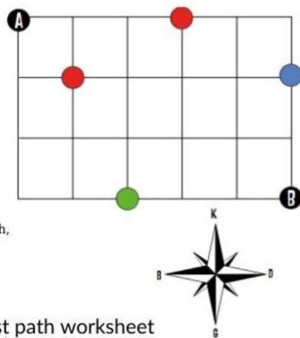
c) Go 2 units east, 1 unit south, 3 units east, 3 units west, 2 units south, 2 units east.

**2. Walk from A to B by stopping at two red squares.**

a) Go 3 units east, 1 unit south, 2 units west, 1 unit south, 4 units east, 1 unit south.

b) Go 1 unit south, 1 unit east, 1 unit north, 2 units east, 3 units south, 2 units east.

c) Go 3 units east, 3 units west, 1 unit south, 1 unit east, 1 unit south, 4 units east, 1 unit south.



6.2.5.B1 shortest path worksheet

**Fig. 6** Pattern Recognition (6.2.7.B1) and Algorithmic thinking (6.2.5.B1): Sample activity for unplugged group

for summative assessment, with the goal of continuing the computer interactions. After the implementations, the scales were administered to both groups as a post-test. Following this, the interviews were conducted.

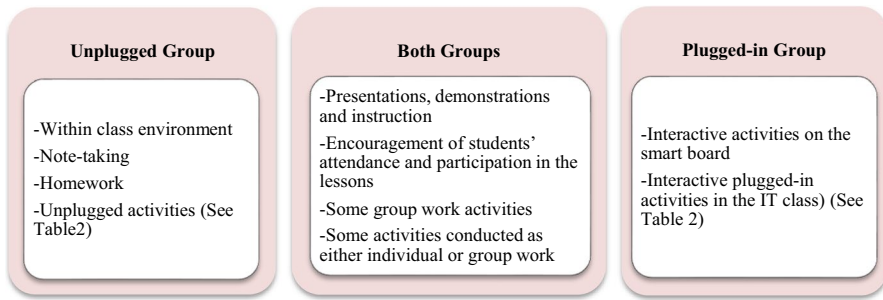


Fig. 7 The characteristics of the teaching activities used in the groups

## 2.5 Data Analysis

In the present study, data were collected before and after the 7-week intervention process. The dependent variables were academic performance in basic programming and computational thinking, while the independent variables were the unplugged and plugged-in instructions. Instead of removing students with no answer (NA) from our analysis, we run the missing data analysis and assigned mean values to the missing data. This is because we do not know why the students did not answer them. It may be due to various circumstances, for example, that he/she skips the question because this happens to young students. Then the normality tests were conducted and the assumptions for the inferential statistics were supplied. The Shapiro–Wilk test for the pre-test achievement scores yielded values of  $p=0.118$  and  $p=0.090$  in the groups, respectively, ( $p>0.05$ ). In addition, the “Hedges’  $g$ ” value was also calculated to explain the effect size of the implementation. In the sub-dimensions of the CTs and overall, the skewness and kurtosis values were found to fall between  $\pm 2$ , which indicate a normal distribution. The values that the regression analysis yielded met the required assumptions, and the test was conducted accordingly. For the analysis of quantitative data, students’ responses were transcribed into written form. A descriptive analysis was conducted to identify student responses in both groups.

## 3 Results

### 3.1 Comparison of academic achievement scores before and after instruction

The present study examined whether there was a significant difference among the groups in terms of academic achievement and CT skills. First, the mean scores of the

**Table 3** Independent Sample t-Test of Pre-Achievement Test Scores

Score	Groups	N	M	SD	df	t	p
Pre-test	Unplugged	44	40.664	14.433	82	1.308	0.195
	Plugged-in	40	36.630	13.760			

**Table 4** Dependent Sample t-Test of Pre-test and Post-test Scores for Academic Achievement

Unplugged	N	M	SD	df	t	p
Pre-test	44	40.664	14.433	43	9.297	0.000
Post-test	44	57.923	17.248			
Plugged-in	N	M	SD	df	t	p
Pre-test	40	36.630	13.760	39	5.649	0.000
Post-test	40	48.510	19.667			

**Table 5** Independent Sample t-Test of Post-Achievement Test Scores

Score	Groups	N	M	SD	df	t	d	p
Post-achievement test	Unplugged	44	57.923	2.600	82	2.337	0.51	0.022
	Plugged-in	40	48.510	3.110				

pre-academic achievement test before the intervention were compared between the groups using a t-test for independent samples; the results are presented in Table 3.

As can be seen in Table 3, pre-test scores did not significantly differ among the groups ( $p > 0.05$ ). Accordingly, the groups' pre-implementation level of basic programming was equivalent. Next, Table 4 summarizes the difference between students' within-group pre-test and post-test academic achievement scores using dependent sample t-Test.

Table 4 shows that there was a significant statistical difference between pre-test and post-test academic achievement scores in favor of the post-test of students in both, the unplugged group ( $t=9.30$ ,  $p < 0.05$ ) and the plugged-in group ( $t=5.65$ ,  $p < 0.05$ ). In this way, both sets of activities had a significant effect on students' academic performance in basic programming.

### 3.2 The comparison of the achievement scores of the groups after the intervention

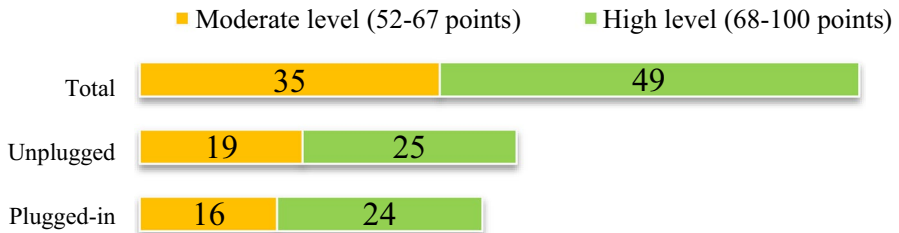
Table 5 shows that the results of the comparison between the academic achievement scores of the students in the unplugged and plugged-in groups after the intervention were examined using the Independent Sample t-Test.

According to Table 5, the post-test mean score of the unplugged group ( $M_{\text{Unplugged}}=57.923$ ,  $SD=2.600$ ) was higher than that of the plugged-in group ( $M_{\text{Plugged-in}}=48.510$ ,  $SD=3.110$ ). Moreover, there was a significant difference among the groups' post-achievement test scores ( $t_{(82)}=2.337$ ;  $p < 0.05$ ). This evidences that the



**Table 6** Descriptive Statistics related to the Computational Thinking Scale after the intervention

Name of the scale	Groups	N	M	SD	Min	Max
The Computational Thinking Scale	Unplugged	44	71.638	13.062	52.48	97.60
	Plugged-in	40	69.767	10.065	52.48	94.55

**Fig. 8** The CT levels of the students in the groups

use of unplugged activities in the classroom supported greater improvement and had a more positive effect on student academic achievement than the use of plugged-in activities. Moreover, with a value of  $d=0.51$ , the Hedges'  $g$  effect size had a moderate level of practical significance.

### 3.3 The Across-Group Comparison of the Computational Thinking Scale Scores after the Intervention

The level of students CT was investigated by analyzing the descriptive data in the computational thinking scale, the results of which are presented in Table 6.

The total scale means of the unplugged group and that of the plugged-in group can be seen in Table 6. The visual descriptive CT levels of the students in both groups are shown in Fig. 8.

In both groups, the majority of the students had a high level of CT ( $N_{\text{unplugged}}=25$ ;  $N_{\text{plugged-in}}=24$ ). After controlling for pre-test achievement scores, there was not a significant effect of teaching condition on computational thinking skill scores ( $F_{(1,81)}=0.108$ ,  $p=0.743$ ,  $\eta^2<0.01$ ). Estimated marginal means were similar in the plugged-in ( $M=3.516$ ,  $SE=0.089$ ) and unplugged ( $M=3.557$ ,  $SE=0.085$ ) conditions. Pre-test achievement was significantly related to computational thinking skill test scores ( $F_{(1,81)}=8.948$ ,  $p=0.004$ ,  $\eta^2=0.099>0.01$ ).

### 3.4 The computational thinking skill as a predictor of academic achievement in the unplugged group

No significant correlation was identified between the examined variables in relation to the students in the plugged-in group ( $p>0.05$ ). In the unplugged group, there

**Table 7** The Predictor Model of the Academic Achievement of Students in the Unplugged Group

Model	B	<i>t</i>	<i>p</i>	<i>F</i>	<i>p</i>	AdjustedR <sup>2</sup>
(Constant)	7.192	0.575	0.569	16.956	0.000	0.271
The Computational Thinking Scale	0.536	4.118	0.000			

Academic achievement model = 0.54\* Computational thinking.

is a significant positive correlation of moderate level between students’ academic achievement and their CT ( $r=0.536, p<0.01$ ). Therefore, a basic linear regression analysis was performed for the unplugged group to identify whether the level of CT predicted academic achievement, between which a moderate degree of positive linear correlation was found. The model developed was found to be significant ( $F_{(1, 42)}=16.956, p<0.05, R^2=0.271$ ) (see Table 7).

Based on the model, it was revealed that the levels of CT significantly predicted students’ post-test achievement scores. The model accounts for 27% of the academic achievement.

### 3.5 Examining the Correlation between Academic Achievement Scores and the Computational Thinking Scale Subdimensions for the Unplugged Group

When the correlation between the academic achievement scores of the students in the unplugged group and the sub-dimensions of the computational thinking scale is examined, a significant correlation is observed between some variables ( $p<0.05$ ). The obtained results are presented in Table 8.

Accordingly, it was found that there was a positive moderate correlation between academic achievements and the sub-dimensions creativity, algorithmic thinking and problem solving of the computational thinking scale, while there was a positive but

**Table 8** The correlation between the subdimensions of the computational thinking scale and academic achievement in the unplugged group

	AA	Cr	AT	Cp	CT	PrS
AA	1					
Cr	<b>0.480**</b>	1				
AT	<b>0.401**</b>	<b>0.303*</b>	1			
Cp	0.229	<b>0.433**</b>	<b>0.371*</b>	1		
CT	<b>0.332*</b>	<b>0.386**</b>	<b>0.554**</b>	<b>0.447**</b>	1	
PrS	<b>0.414**</b>	<b>0.258</b>	<b>0.335*</b>	<b>0.369*</b>	<b>0.244</b>	1

\*\* . Correlation is significant at the 0.01 level (2-tailed). \* . Correlation is significant at the 0.05 level (2-tailed).

AA: Academic Achievement, Cr: Creativity, AT: Algorithmic thinking

Cp: Cooperativity, CT: Critical Thinking, PrS: Problem Solving

**Table 9** The Predictor Model of the Academic Achievement of Students in the Unplugged Group by the Computational Thinking Scale Subdimensions

Model	B	t	p	F	p	Adjusted R <sup>2</sup>
(Constant)	0.322	0.025	0.980	5.446	0.001	0.293
Creativity	0.344	2.427	0.020			
Algorithmic Thinking	0.196	1.232	0.226			
Critical Thinking	0.029	0.180	0.858			
Problem Solving	0.253	1.826	0.075			

Academic achievement model = (0.34 \* Creativity).

weak correlation between critical thinking and academic achievement. Then, the results obtained by the multiple regression analysis are presented in Table 9.

The hypotheses predicting academic achievement for subdimensions other than creativity was rejected. The hypothesis that only the creativity subdimension of the computational thinking scale is a predictor of academic achievement was confirmed by the resulting regression model ( $p < 0.05$ ). Creativity explained 29% of academic achievement according to the resulting model.

### 3.6 Findings Regarding Students' responses to interview

The responses received from six students from each of the groups – plugged-in and unplugged – were categorized. The groups were coded as S<sub>1\_x</sub> (plugged-in) and S<sub>2\_x</sub> (unplugged). In Table 10, the codes for the themes and the frequencies are presented for each group separately, and the responses that were commonly expressed are indicated with a different bullet point.

In both groups, students liked the classroom activities. They rated the teacher's classroom management as positive. One student in plugged-in group said, "*I was really excited when we were going to play games. I did not feel sad at all. The games were fun* (S<sub>1\_4</sub>)." In the plugged-in group, feelings of excitement and a sense of collaboration were emphasized. Students in the unplugged group, on the other hand, indicated that the lessons were educational and that they had more freedom during the lessons. One of them said, "*I felt happy. We learned how to solve the questions.*" (S<sub>2\_4</sub>). They highlighted that they learned the material and said, "*I felt happy. We learned how to solve the questions.*" (S<sub>2\_4</sub>).

As a barrier to learning, anxiety must be kept at a low level. In fact, all students who participated in both groups of activities indicated that they felt no anxiety throughout the learning process. They attributed the lack of anxiety to the fact that they were able to ask their questions easily and had no difficulty learning. One of the unplugged groups said, "*I was not anxious because I could ask the questions that were on my mind. The teacher explained them right away*" (S<sub>2\_3</sub>). Furthermore, making lessons palatable to students makes learning conducive. In fact, students in both groups found the learning process enjoyable. Some students in unplugged group mentioned that it was fun to learn certain topics. Especially in the plug-in group, playing games was found to be entertaining. One in the plugged-in group

**Table 10** The process-related opinions of the students in the plugged-in and unplugged groups

Themes	Codes	f	Themes	f	Codes	f
General opinions about the course	✓ Finding the course good	10	Using algorithms to solve problems	5	✓ Makes the work easier	5
	✓ The integration of games into the lessons	9		1	■ Enabling the work to be done more quickly	1
	✓ Statements about liking the course	4		1	■ Enabling the prevention of errors	1
	■ Liking the course owing to learning something in the course	5		3	✓ Enabling the prevention of confusions	3
	✓ Positive teacher approach	11		5	■ Enabling doing the steps of the work in a sequence	5
Feelings in class	• Feeling happy	2		4	■ Enabling doing the work in a planned way	4
	• Feeling a sense of freedom	3		2	• No opinion stated	2
	• Being active	1	The effect on problem solving methods	5	✓ Makes it easier for me to understand the problem	5
	• Feeling relaxed	1		3	✓ I am doing the work step by step	3
	■ Feeling excited	3		1	• Enabled me to understand the mathematics and science courses better	1
Anxiety	■ Feeling a sense of cooperation	1		2	■ I plan [my work]	2
	• No anxiety owing to the opportunity to ask questions	2		4	✓ It did not seem to affect the problem-solving method	4
	✓ No reason expressed for not feeling any anxiety	4	The effect on developing/finding solutions to new problems	1	• Understanding how to solve the problem	1
	■ No anxiety owing to lack of learning difficulties	6		4	• Solving them more easily	4
	✓ Finding some topics fun to learn	6		2	✓ Solving them in a step-by-step manner	2
Enjoyment	■ The enjoyment of playing games	6		2	✓ It didn't seem to affect the method of finding solutions to prob	2
	• The lessons made enjoyable by the teacher	5				

**Table 10** (continued)

Themes	Codes	f	Themes	f	Codes
Challenge	<ul style="list-style-type: none"> <li>■ The facilitating nature of learning technology</li> <li>● Its being practical; understanding how to do it</li> <li>✓ Experiencing difficulty when not knowing how to do them</li> </ul>	4 5 3	Opinions regarding learning more effectively via activities	2	<ul style="list-style-type: none"> <li>✓ Understanding more effectively</li> <li>✓ More effective learning through visual aids</li> <li>● The idea of not being able to get a higher score</li> </ul>
Boredom	<ul style="list-style-type: none"> <li>● Expressions of boredom in class</li> <li>✓ No reason stated for not getting bored</li> <li>● Boredom in face of challenges</li> <li>■ I understood the topics explained well</li> </ul>	1 7 2 1		1 1 1	<ul style="list-style-type: none"> <li>● Being fond of pen and paper activities</li> <li>■ Being able to use the computer better</li> <li>■ Making solutions easier by trial and error</li> </ul>
Effect of the activities on the final course grade (Effect observed)	<ul style="list-style-type: none"> <li>■ Because learned while engaging in the activities</li> <li>■ Good to see how it is done on the computer</li> </ul>	4 2	The effect of the activities on final course grade (No effect)	1 2	<ul style="list-style-type: none"> <li>■ Feeling the sense of a real, typical lesson</li> <li>■ The opinion of not having a tight lesson</li> </ul>
				1	<ul style="list-style-type: none"> <li>● The opinion of not being taught the details</li> <li>■ The opinion of being able to get a higher score if there were no games</li> </ul>

- Unplugged group
- Plugged—in group
- ✓ Both unplugged and plugged—in groups

said, *"I had fun because I learned how to solve the questions; I learned how to use the computer and I had fun when we played games"* (S<sub>2\_3</sub>).

Another consideration is that if the instructions are challenging, this makes learning difficult. Fortunately, the plugged-in group expressed the opinion that learning technology made their lives easier. In the unplugged group, it was pointed out that the activities facilitated learning because they were practical, but three students indicated that they faced challenges when they did not understand how to perform the activities. Because block-based programming environments for children allow trial and error, they are not afraid to make mistakes. Therefore, plugged-in activities challenged them rather less. Students were also not bored because of the gaming effect of the plugged-in activities. However, two students in the unplugged group reported boredom and one said, *"I was bored. I got bored solving the algorithm."*(S<sub>2\_2</sub>). However, all students in the plugged-in group indicated that they were not bored. The reasons that they were not bored are that they were able to express their opinions freely and understood the topics covered in class well.

Topics mainly include understanding and creating algorithms and related concepts. For example, students were asked about the effect of solving problems using algorithms. Three students in the unplugged group indicated that using algorithms to solve problems makes things easy, saying, *"Algorithms are a very simple method. More explanatory."* (S<sub>2\_3</sub>). Similarly, five students in the plugged-in group stated that using algorithms was useful as it enabled one to do the steps in the work in a sequential and organized way. They also stated that they planned and carried out the work step by step, using a problem-solving method that they improved by using an algorithm. Similarly, two students from the unplugged group used the word "nice", and one student from each group used the word "good", stating that they were able to understand how to solve the problems, how to develop problem solving methods like finding the solution by following a step-by-step approach. It seems that the unplugged group developed a better understanding of how to use algorithms to solve problems, saying, *"I can find the answers to questions instantly. I learned how everything is or will be."*(S<sub>2\_3</sub>).

After that comes the question about their activities. The unplugged group rated activities as engaging in activities, learning more effectively, enjoying using pen and paper, and learning with visual aids. They said *"They enabled me to learn more effectively because I like it when I do them on paper."* (S<sub>2\_2</sub>) *"They enabled me to learn more effectively because when you assign an activity, I remember it. I still study with activity worksheets."* (S<sub>2\_6</sub>). Similarly, plugged-in group reported learning more effectively through trial and error, using the computer more proficiently and learning with visual aids. Furthermore, students then gave their opinions on the impact of the activities on their academic performance. The majority of the unplugged group stated that the activities contributed to their achievement and that they learned during the activities. One said, *"The activities in the course allowed me to get a slightly higher score; I learned the algorithm better through these activities."*(S<sub>2\_1</sub>). One student said, *"The activities in the course did not help me get a higher score because I did not feel like it was real lessons. We did not do a tight class"* (S<sub>1\_1</sub>). The conclusion is that the computer activities and subsequent paper and pencil exam might make it difficult for the plugged-in group to transfer their knowledge. But the unplugged group had already learned with paper and pencil and kinesthetic activities. This is also consistent with the quantitative results of academic achievement scores.

## 4 Discussion

### 4.1 Within-Group and among-groups Differences in Academic Achievement

The purpose of the present study was to compare the plugged-in and unplugged activities in terms of programming academic achievement, hereafter academic achievement, and CT skill in two groups. There was no significant difference in pre-test academic achievement scores among unplugged and plugged groups. This shows the equivalence of the groups' preliminary knowledge prior to the intervention. It was found that the difference within-group pre and post-academic achievement test scores, was in favor of the post-test, suggesting that both unplugged and plugged-in activities led to learning. The intervention resulted in improved academic achievement and CT skill. Importantly, a significant difference was found in academic achievement in favor of unplugged activities. Similarly, there are studies in the related literature that report a positive impact of unplugged activities on achievement in basic programming (Tsortanidou et al., 2022; del Olmo-Munoz et al., 2020; Gülbahar & Kalelioğlu, 2018; Kalelioğlu, 2018; Moreno-León & Robles, 2015; Rodriguez, 2015). Unplugged activities facilitate the understanding of basic concepts related to computer science and the concept of programming (Kalelioğlu, 2018; Rodriguez, 2015). In the present study, the students in the unplugged group stated that what they learned became clear in their minds, that they arrived at solutions more easily, learned algorithm more effectively and found the topics easy, which verifies its significant difference among the two groups. This shows that unplugged activities help students focus better on the subject matter by eliminating the destructive effects of computers or other digital tools. Besides that, the plugged-in platforms are standardized and this limits the adaptability of them to students' local experiences in the classroom. However, the ease of adapting and redesigning unplugged activities could possibly support the learning of the unplugged group. In a study by Pérez-Marín et al. (2018), the results showed that it was possible to teach basic programming concepts in a short time through the use of unplugged activities, and they observed a development of CT skills. Moreover, in the present study, students in the unplugged group commented on the reasons that positively affected their learning, such as the fact that they liked the pen and paper activities and the effectiveness of learning through visual posters. Showing relationships between concepts and how concepts interact with each other on visual aids or posters seems to be very useful for teaching basic concepts.

In fact, students who participated in the plug-in activities cited as reasons for this difference that some of them felt they were not participating in a typical course, that the course was not taught effectively, and that they could achieve higher scores if there were no computer games in the classroom. However, this may not be a barrier to the fact that, as Rodríguez-Martínez, et al. (2020) found, integrating CT with Scratch applications resulted in higher student academic achievement compared to students exposed to the traditional method. In the present study, each platform used was designed or selected by considering various

characteristics of the various plugged-in activities that could address the corresponding learning objective, in order to diversify and adapt the activities as much as possible to the level of knowledge of the students. If the teaching practice of computer programming is adapted to the age level of the younger students, they will internalize it more easily (Pérez-Marín et al., 2018). In conclusion, sixth grade students can learn easily when activities and metaphors are adapted to the level and experience of them and topics that are not related to the learning objectives are eliminated. Brown et al. (1989) stated that when teaching simple and concrete computer science concepts, students could work collaboratively, solve problems, and think creatively using unplugged activities. Zhan et al. (2022) supports this situation for unplugged activities, also states that it offers more interaction. During the interviews of the plugged-in group in the present study, the students indicated that they felt enthusiastic while playing, that they were happy, and that they were not bored or anxious. Even though students viewed the process as positive, they did not feel they were in a typical course. In a study by Yallihep and Kutlu (2020) which investigated the effect of games on students' understanding of programming concepts, it was reported that teaching through games did not have a positive effect on students' attitude towards the course. Meerbaum-Salant et al. (2013), who evaluated the teaching materials they developed via Scratch with secondary school students, reported that they experienced difficulties in teaching certain topics such as compilation, variables, and flow. However, to avoid bias, for which platform the academic achievement test was developed should be taken into consideration when evaluating these results.

On the other hand, there are certain negative effects related to unplugged activities. For example, Tsarava et al. (2022) and Jun (2018) stated that unplugged activities are gaining popularity, but standards were lacking in the assessment of these activities. Similarly, students in the present study attributed their inability to achieve higher scores to the fact that several assessment questions were different from classroom activities. It may be that students were unable to relate the hands-on activities to those in the pen-and-paper exams. Related to this point, Falkner and Vivian (2015) reported that there is a further need for unplugged materials to be prepared in a standardized manner and for assessment and critical thinking materials to be developed for primary school students as part of the programming process. Rodríguez-Martínez et al. (2020) made it clear that students had little prior experience with CT and they implied that the development of computational concepts does not happen by chance at this stage of students' education. The long time it took the students to complete the activities also showed us that. When they get into it, it goes faster. In the present study, the finding that the impact of the teacher's classroom management approach was viewed positively by the students shows a consistency with the findings reported in other studies that unplugged activities contribute positively to students' long-term memory span and motivation level due to the interactions they have with each other and with their teachers (Zhan et al., 2022), and that close and effective supervision by teachers can facilitate more effective learning than leaving students alone with the devices (Meerbaum-Salant et al., 2013).



## 4.2 The Difference among the Plugged-in and Unplugged Groups in terms of the Computational Thinking Skill

No significant difference was found between the two groups of activities. Five sub-dimensions of the computational thinking scale were addressed in our study namely creativity, algorithmic thinking, cooperativity, critical thinking and problem solving. Both groups scored approximately the same mean scores in these sub-dimensions.

In both groups, the students' CT skills were at an intermediate or high level. The unplugged group achieves slightly higher mean scores only in the sub-dimensions algorithmic thinking, cooperativity and problem solving of the computational thinking scale. Similar results have also been reported in related literature (Atman Uslu et al., 2018; Brackmann et al., 2017; Chen et al., 2017; Çelik&Özdener, 2019; Davies, 2008; Harris, 2018; Pérez-Marín et al., 2018; Zhan et al., 2022). Similarly, Merino-Armero et al. (2022) said that the unplugged approach contributed to students CT skills. In a study by Brackmann et al. (2017), it was reported that students who engaged in unplugged activities developed more skills than those who engaged in plugged-in activities. In an experimental study, it was found that the game-based unplugged group showed better academic performance (Kuo & Hsu, 2019). In a study of teachers by Harris (2018), it was highlighted that the pre- and post-intervention groups showed similar levels of skill and that teachers learned in a more holistic, enjoyable, and confident manner through the use of unplugged activities.

The finding that students gain depth of interaction in the unplugged activities (Kuo & Hsu, 2019; Zhan et al., 2022) supports the high means score in favor of the unplugged group in the cooperativity sub-dimension in our study. One of the reasons could be that unplugged activities provide more opportunities for collaboration. On the other hand, Yıldız et al. (2017), in their study investigating whether there was a difference between students' computational skills in relation to the type of digital games they played, found that the level of critical thinking skills was low among secondary school students. This result is consistent with the critical thinking sub-dimension being rated low in both groups compared to the other sub-dimensions and the overall scale.

The fact that there was no significant difference among the groups in terms of intervention activities should not be viewed negatively. Thus, in the posttest, the level of CT was high in both groups. Similarly, there are studies that report that unplugged activities develop these skills at least as much as plugged-in applications and that they motivate students (Atman Uslu et al., 2018; Brackmann et al., 2017; Çelik & Özdener, 2019; Davies, 2008; Pérez-Marín et al., 2018). While block-based programming activities have a positive effect on the development of CT (Karaahmetoğlu & Korkmaz, 2019; Turan, 2019), there are also studies that find no significant difference between the compared applications (Atman Uslu et al., 2018; Gülbahar & Kalelioğlu, 2014). So, based on these results, it can be claimed that when students have no prior familiarity with computers, their interest in how computers operate can be developed based on knowledge of unplugged activities. Because they are focused on the subject and the task and are not afraid of breaking the computer or don't want to play games on the other windows of the computer. In parallel with these results, students with unplugged activities are able to explore

computer science concepts and fundamentals more deeply without the presence of a computer, which they usually consider as a tool or toy (Bell et al., 2009).

As in the present study, Hsu, et al. (2018), who conducted a literature review study on how CT should be taught and researched and the reasons why no significant difference was found in the CT, recommended that the method and content of instruction should vary as students grow older and progress to higher grade levels. The group activities and active learning activities that Zeybek (2019) recommended to develop students' cooperative characteristics seem to indicate unplugged activities. For example, the students in the unplugged group expressed in their opinions that the lessons and games were integrated and that they felt happy, relaxed and had a sense of freedom. Zhan et al. (2022) also found that the unplugged group was more engaged and this improved their classroom interaction and communication. Indeed, having students engage in unplugged activities as a group or in pairs seems to enhance communication, interaction, and learning. Merino-Armero et al. (2022) discussed how CT can be introduced in schools. Tsortanidou et al. (2022) highlight the explicit need to integrate CT and related concepts into the primary curriculum. We believe that the preliminary readiness of teachers is a precondition to integrate it into the primary curriculum explicitly. Sendurur (2019) analyzed the unplugged activities designed by prospective computer teachers. He pointed out that the design process showed that the teacher candidates' designs were incompetent. In fact, we have also seen how important it is for students in the classroom that teachers place value on unplugged activities. When teachers were familiar with unplugged activities and knew how to use and adopt them, students felt freer and perceived the lessons as more fun and educational, they said. Students in the unplugged group may have felt freer because the way activities were conducted was based on creativity and the possibility of different solutions. However, for rural students who mostly have little experience with computers, unplugged activities can be an effective way to improve their CT. In fact, both pre-service and in-service teachers need to know how to design an unplugged activity and how to develop possible solutions to potential implementation challenges. In our implementation experience, the students' reading level and the time required for the activity were very critical to their performance. Similarly, Tsarava et al. (2022) indicated that CT has a positive association with students' linguistic thinking ability.

### 4.3 The Correlation between the Variables in Groups

In the present study, a moderate positive correlation was found between academic achievement and CT in the unplugged group. Academic achievement posttest scores were significantly predicted by CT. This correlation indicates that development in the CT increases academic achievement in basic programming classes. There are also other studies that found this correlation (Chen et al., 2017; Conde et al., 2017; Donley, 2018; Korkmaz et al., 2018; Korkmaz et al., 2015a, b; Román-González et al., 2018; Taylor et al., 2010).

A study by Korkmaz et al. (2018) found a correlation between achievement in programming and CT. A study by Román-González et al. (2018) has empirically

supported the statement that CT and coding achievement are cognitively correlated in middle school. In addition, Donley (2018) found a moderate significant correlation between academic achievement and creativity. This finding shows consistency with the finding in our study regarding the correlation between academic achievement and its predictor, the CT, and the creativity subdimension. Korkmaz et al. (2018) also showed a moderate correlation between achievement and problem solving-another subdimension of CT. Similarly, Chen et al. (2017) reported that students who did not follow the codes in their observations found a more effective and creative solution method and showed effective CT. Moreover, Korkmaz et al., (2015a, b) found that programming performance can be predicted by the CT. Thus, as creativity increases, academic achievement is also expected to increase, since the only subdimension of CT, which is accounting for academic achievement, is creativity.

#### 4.3.1 The correlation between academic achievement scores and subdimensions of the computational thinking scale for the unplugged group

Since the CT is a predictor of academic achievement, the correlation between academic achievement and the subdimensions of the CT was examined for the unplugged group. Accordingly, it was found that there was a positive moderate correlation between academic achievement and the sub-dimensions creativity, algorithmic thinking and problem solving of the computational thinking scale, while there was a positive but weak correlation between critical thinking and academic achievement. With the exception of the cooperativity sub-dimension of CT, all sub-skills can be considered effective in enhancing performance in the basic concepts of programming related to unplugged activities. Based on the developed model, 29% of academic achievement is predicted by creativity sub-dimension alone. This could mean that CT is a skill closely related to creativity. We can emphasize that creativity-enhancing teaching practices are needed to improve students' basic programming skills. Also, in a study by Atiker (2019), a significant difference in the creativity sub-dimension was found between groups in favor of the experimental group. Likewise, Berry (2014) argues that creativity has informed ICT teaching for many years and that it would be wrong to neglect this important factor within CT. By focusing on creative thinking within CT, students are helped to engage with new ideas (Brooks, 2019). Similarly, students in the present study felt that developing new methods of solving problems through unplugged activities helped them understand how to solve problems in a relatively easier way than before. In this way, children learn how to analyze problems and improve solutions. Learning problem-solving skills can also decrease their anxiety about the subject and teaching. In addition, they believed that algorithms helped in getting the job done faster and saved them from making mistakes. Atun and Usta (2019) stated that programming instruction developed students' CT as well as problem solving skills. Finally, Nouri et al. (2020) supported with evidence that there is a strong correlation between programming and creativity. It must be noted because computer programming is the way of utilizing of utilizing of computers to form solutions of crucial problems of human being, it gives emphasis on the creativity and problem-solving skills.

## 5 Conclusions

The aim of the present study was to compare the unplugged activities and their equivalent plugged-in activities used in teaching basic programming in 6th grade ICT course and to reveal student opinions. Programming basics and CT are rapidly being integrated into primary education curricula, and tools and methods are being developed to teach them. As the findings of this study show, the literature recommends unplugged activities as an effective option for making these skills accessible to all basic education students. This study makes a contribution to the impact of unplugged activities for teaching basic computer skills, especially in countries and regions with limited access to computers and Internet technology. In doing so, it demonstrates that digital tools such as computers and tablets are not essential for the acquisition of these skills by comparing unplugged activities with plugged-in activities based on learning objectives. The platforms that could address the related learning objective was selected for plugged-in group by taking into consideration different features of various plugged-in activities; attempts were made to add variety to the activities and to adapt to the students' level of knowledge as much as possible. As a recommendation, the type of teaching activities must be selected according to the conditions, interests, level of knowledge of the students, etc. This study has some limitations. Because of physical circumstances of classrooms (the limited numbers of computers or other materials), some activities were conducted in groups or pairs. In addition, analysis of item difficulty and item discrimination in the achievement test was conducted for 104 respondents.

A significant difference in academic performance in basic programming was found in favor of the unplugged activities and that the unplugged approach-maintained students' CT skill compared to the plugged-in group. Also, the creativity sub-dimension of CT explains 29% of academic achievement in the unplugged group. This suggests the importance of CT to academic achievement in basic programming. Creativity subdimension of the computational thinking scale is a predictor of academic achievement was confirmed by the resulting regression model ( $p < 0.05$ ). Also, the students in the unplugged group expressed their positive perspective that lesson and games were integrated, and that they felt happy, relaxed, and a sense of freedom. In the present study, students in the unplugged group indicated that what they learned became clear in their minds, they arrived at solutions more easily, learned algorithms more effectively, and found the topics easily, confirming the significant difference between the two teaching practices. However, students who participated in plugged-in activities indicated that they did not feel they were taking a regular course, that the course was not taught effectively, and that they would score higher if there were no games in the course. Indeed, the use of unplugged activities for the basic ICT skills development needs to be extended especially to learning environments in rural areas where there is limited access to computers so as not to widen the digital gap.

## 6 Recommendations

Taking into account the findings of the present study and the knowledge and experience gained during the research, planning, implementation and follow-up phases, some recommendations were made. The goal of extending algorithmic thinking skills to lower grades requires studies and research on more effective and comprehensive content designs. Therefore, it can be suggested that more unplugged activities can be developed or adapted to the cultures.

- Researchers can develop standardized assessment instruments that measure basic thinking skills independent of instructional content, thereby eliminating some uncertainties and biases. Standardized tests can be developed to assess academic achievement that incorporates unplugged methodology.
- Depending on the nature of the learning objectives, both researchers and teachers should consider the different pedagogical characteristics, target audiences, and teaching methods of tools to select.
- It is recommended that studies should be conducted on the method and content of different unplugged activities for students of different age groups in different grade levels.
- A study can be conducted to investigate how much learning is achieved when the unplugged activities are presented individually in one group and in groups or pairs in another group. In conclusion, may be recommended to use a larger sample size to examine the CT and other variables.
- Students with computer experience and students with limited experience can be compared on study variables with the same unplugged activities. So, in which group CT and programming basics would be developed more through unplugged activities.

**Acknowledgements** This study was carried out as part of the master thesis entitled “Effect of unplugged activities and computer applications used in secondary school basic programming education on academic achievement and computational thinking” (Thesis Number: 634096).

## Declarations

**Conflict of Interest** The authors declare that they have no conflict of interest.

**Ethical approval** All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

**Informed consent** Informed consent was obtained from all individual participants included in the study.

## References

- 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>
- Asbell-Clarke, J., Rowe, E., Almeda, V., Edwards, T., Bardar, E., Gasca, S., Baker, R. S., Scruggs, R. (2021). The development of students' computational thinking practices in elementary- and middle-school classes using the learning game Zoombinis. *Computers in Human Behavior*, *115*, 106587. <https://doi.org/10.1016/j.chb.2020.106587>
- Atiker, B. (2019). *The effects of middle school students' computational thinking skills on success in programming teaching (Publication No. 561543) [Doctoral dissertation, İstanbul University]*. YÖK National Thesis Center. Available at <https://tez.yok.gov.tr/UlusalTezMerkezi/>
- Atman Uslu, N., Mumcu, F., & Eğin, F. (2018). Görsel programlama etkinliklerinin ortaokul öğrencilerinin bilgi-işlemsel düşünmebecerilerine etkisi. *Ege Eğitim Teknolojileri Dergisi*, *2*(1), 19–31.
- Atun, H., & Usta, E. (2019). The effects of programming education planned with TPACK framework on learning outcomes. *Participatory Educational Research*, *6*(2), 26–36. <https://doi.org/10.17275/per.19.10.6.2>
- Balanskat, A. & Engelhardt, K. (2015). *Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe*. Retrieved August 26, 2021, from [http://www.eun.org/c/document\\_library/get\\_file?uuid=3596b121-941c-4296-a760-0f4e4795d6fa&groupId=43887](http://www.eun.org/c/document_library/get_file?uuid=3596b121-941c-4296-a760-0f4e4795d6fa&groupId=43887)
- Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.
- Bell, T. C., Witten, I. H., & Fellows, M. (1998). *Computer science unplugged: Off-line activities and games for all ages*. CiteSeer. Retrieved March 14, 2021, from <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.6827&rep=rep1&type=pdf>
- Berry, M. (2014). *Computational thinking and creativity*. Retrieved April 11, 2021, from <http://milesberry.net/2014/08/computational-thinking-and-creativity/>
- Bers, M. U. (2018). Coding and Computational Thinking in Early Childhood: The Impact of Scratch Jr in Europe. *European Journal of STEM Education*, *3*(3), 08. <https://doi.org/10.20897/ejsteme/3868>
- Bitesize. (2021). *Introduction to computational thinking*. Retrieved September 30, 2021, from <https://www.bbc.com/bitesize/guides/zp92mp3/revision/1>
- Bocconi, S., Chiocciariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice (No. JRC104188)*. In P. Kamyllis, & Y. Punie (Eds.). Seville, Spain: European Commission Joint Research Centre. <https://doi.org/10.2791/792158>
- Bougot-Robin, K., Paget, J., Atkins, S. C., & Edell, J. B. (2016). Optimization and design of an absorbance spectrometer controlled using a raspberry pi to improve analytical skills. *Journal of Chemical Education*, *93*(7), 1232–1240. <https://doi.org/10.1021/acs.jchemed.5b01006>
- Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017, November 8–10). Development of computational thinking skills through unplugged activities in primary school [Conference session]. The 12th Workshop on Primary and Secondary Computing Education, 65–72. ACM, Nijmegen, Netherlands. <https://doi.org/10.1145/3137065.3137069>
- Brackmann, C., Barone, D., Casali, A., Boucinha, R., & Muñoz-Hernandez, S. (2016, September 13–15). Computational thinking: Panorama of the Americas [Conference session]. The International Symposium on Computers in Education (SIIE), Salamanca, Spain. <https://doi.org/10.1109/SIIE.2016.7751839>
- Brooks, J. (2019). The creative power of computational thinking. Retrieved January 28, 2021, from <https://medium.com/tech-based-teaching/the-creative-power-of-computational-thinking-24dae04bce93>
- Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher*, *18*(1), 32–42. <https://doi.org/10.3102/2F0013189X018001032>
- BuitragoFlórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834–860. <https://doi.org/10.3102/2F0034654317710096>

- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- Computer Science Unplugged. (n.d.). *Computer science unplugged*. Retrieved May 25, 2021 from <https://www.stem.org.uk/cx5u7>
- Conde, M. Á., Fernández-Llamas, C., Rodríguez-Sedano, F. J., Guerrero-Higueras, Á. M., Matellán-Olivera, V., & García-Peñalvo, F. J. (2017, October 18–20). Promoting computational thinking in K-12 students by applying unplugged methods and robotics [Conference session]. Cádiz Spain, TEEM 2017. Retrieved September 26, 2021, from <https://repositorio.grial.eu/bitstream/grial/1070/1/a07-Conde.pdf>
- Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approach*. Sage.
- CS Fundamentals Curriculum Guide. (2021). Retrieved May 26, 2021, [https://code.org/curriculum/docs/csf/CSF\\_Curriculum\\_Guide\\_2018\\_smaller.pdf](https://code.org/curriculum/docs/csf/CSF_Curriculum_Guide_2018_smaller.pdf)
- CS Unplugged. (2021). Computational Thinking and CS Unplugged. Retrieved July 15, 2021, from <https://csunplugged.org/en/computational-thinking/>
- Çelik, A., & Özden, N. (2019). Bilgisayarlı ve bilgisayarsız programlama etkinliklerinin güdülenme üzerindeki etkisi. *Akademik Sosyal Araştırmalar Dergisi*, *88*, 651–669. <https://doi.org/10.16992/ASOS.14692>
- Davies, S. (2008). *The effects of emphasizing computational thinking in an introductory programming course [Conference session]*. Saratoga Springs, NY, 38th ASEE/IEEE Frontiers in Education Conference. <https://doi.org/10.1109/FIE.2008.4720362>
- Del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education*, *150*, 103832. <https://doi.org/10.1016/j.compedu.2020.103832>
- Delal, H., & Oner, 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>
- Digital Promise. (2021, March). *Powerful learning with computational thinking: Our why, what, and how of computational thinking*. Digital Promise. <https://doi.org/10.51388/20.500.12265/115>
- Djambong T., Freiman V., Gauvin S., Paquet M., Chiasson M. (2018). Measurement of Computational Thinking in K-12 Education: The Need for Innovative Practices. In: Sampson D., Ifenthaler D., Spector J., Isaias P. (eds) *Digital Technologies: Sustainable Innovations for Improving Teaching and Learning*. Springer, Cham. [https://doi.org/10.1007/978-3-319-73417-0\\_12](https://doi.org/10.1007/978-3-319-73417-0_12)
- Donley, K. S. (2018). *Coding in the curriculum: learning computational practices and concepts, creative problem solving skills, and academic content in ten to fourteen year-old children [Doctoral dissertation, the Temple University Graduate Board]*. ProQuest. Retrieved July 16, 2021, from <https://www.proquest.com/dissertations-theses/coding-curriculum-learning-computational/docview/2100068134/se-2?accountid=13014>
- Draw.io (2021). *About us*. Retrieved September 06, 2021, from <https://about.draw.io/about-us/>
- Duncan, C., Bell, T., & Tanimoto, S. (2014, November 5 - 7). Should your 8-year-old learn coding? [Conference session]. WiPSCE'14. ACM, New York, NY, USA. <https://doi.org/10.1145/2670757.2670774>
- Eight Reasons Why Every Child. (2021). *Teachyourkidscode*. Retrieved July 15, 2021, from <https://teachyourkidscode.com/>
- Falkner, K., & Vivian, R. (2015). A review of computer science resources for learning and teaching with K12 computing curricula: An Australian case study. *Computer Science Education*, *25*(4), 390–429. <https://doi.org/10.1080/08993408.2016.1140410>
- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *87*–97. <https://doi.org/10.1016/j.compedu.2012.11.016>
- Futschek, G. (2006). *Algorithmic thinking: the key for understanding computer science*. In: Mittermeir, R.T. (Ed.), ISSEP 2006, LNCS, 4226, 159–168. [https://doi.org/10.1007/11915355\\_15](https://doi.org/10.1007/11915355_15)
- Gal-Ezer, J. (1995). Computer science teachers' certification program. *Computers and Education*, *25*(3), 163–168. [https://doi.org/10.1016/0360-1315\(95\)00040-2](https://doi.org/10.1016/0360-1315(95)00040-2)
- Gal-Ezer, J., Vilner, T., & Zur, E. (2004). Teaching algorithm efficiency at CS1 Level: A different approach. *Computer Science Education*, *14*(3), 235–248. <https://doi.org/10.1080/0899340042000302736>

- Garneli, B., Giannakos, G. & Chorianopoulos, K. (2015). Computing education in K–12 schools. A review of the literature [Conference session]. The 2015 IEEE Global Engineering Education Conference (EDUCON), Tallinn, Estonia. <https://doi.org/10.1109/EDUCON.2015.7096023>
- Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, 1–30. <https://doi.org/10.1080/08993408.2019.1568955>
- Gülbahar, Y. (2017). Bilgi işlemsel düşünme ve programlama konusunda değişim ve dönüşümler. Y. Gülbahar (Ed.), *Bilgi İşlemsel Düşünmeden Programlamaya* (1. Baskı, pp. 395–410). Pegem Akademi. <https://doi.org/10.14527/9786052411117>
- Gülbahar Güven, Y. (Ed.), Kalelioğlu, F., Kert, S. B., İliş, E. B., Kızıman Demirhan, E., Yurdakök, E. A., & Karaosmanoğlu, G. (2018). *6. Sınıf Bilişim Teknolojileri ve Yazılım Dersi Öğretmen Rehberi [6th Grade Information Technologies and Software Lesson Teacher's Guide]*. MEB Press. Retrieved October 16, 2021, from [www.eba.gov.tr](http://www.eba.gov.tr)
- Gülbahar, Y., & Kalelioğlu, F. (2014). The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education*, 13(1), 33–50.
- Gülbahar, Y. & Kalelioğlu, F. (2018). Bilişim teknolojileri ve bilgisayar bilimi: Öğretim programı geliştirme süreci. *Millî Eğitim Dergisi*, 47(217), 5–23. Retrieved June 5, 2021, from [https://dhgm.meb.gov.tr/yayimlar/dergiler/Milli\\_Egitim\\_Dergisi/217.pdf](https://dhgm.meb.gov.tr/yayimlar/dergiler/Milli_Egitim_Dergisi/217.pdf)
- Harris, C. (2018). *Computational thinking unplugged: Comparing the impact on confidence and competence from analog and digital resources in computer science professional development for elementary teachers (Paper 374)*[Doctoral dissertation, St. John Fisher College]. Fisher Digital Publications. Available at [https://fisherpub.sjfc.edu/education\\_etd/374/](https://fisherpub.sjfc.edu/education_etd/374/)
- Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126, 296–310. <https://doi.org/10.1016/j.compedu.2018.07.004>
- Huang, W., & Looi, C.-K. (2020). A critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education. *Computer Science Education*, 1(29). <https://doi.org/10.1080/08993408.2020.1789411>
- Israel-Fishelson, R., & Hershkovitz, A. (2022). Studying interrelations of computational thinking and creativity: A scoping review (2011–2020). *Computers & Education*, 176, 104353. <https://doi.org/10.1016/j.compedu.2021.104353>
- ISTE (2021). *Computational thinking competencies*. Retrieved February 24, 2020, from <https://www.iste.org/standards/computational-thinking>
- Jun, W. (2018). *A study on development of evaluation standards for unplugged activity [Conference session]*. ICTC 2018, Jeju, Korea (South). <https://doi.org/10.1109/ICTC.2018.8539505>
- Kalelioğlu, F. (2018). Bilgisayarsız Bilgisayar Bilimi (B3) Öğretimi. In Y. Gülbahar (Ed.), *Bilgi İşlemsel Düşünmeden Programlama Kitabı* (pp. 183–204). PEGEM Akademi.
- Karahmetoğlu, K., & Korkmaz, Ö. (2019). The effect of project-based arduino educational robot applications on students' computational thinking skills and their perception of basic stem skill levels. *Participatory Educational Research*, 6(2), 1–14. <https://doi.org/10.17275/per.19.8.6.2>
- Korkmaz, Ö., Karaçaltı, C. & Çakır, R. (2018). Öğrencilerin Programlama Başarılarının Bilgisayarca-Eleştirel Düşünme ile Problem Çözme Becerileri Çerçevesinde İncelenmesi. *Amasya Üniversitesi Eğitim Fakültesi Dergisi*, 7(2), 343–370. Retrieved February 10, 2022 from <https://dergipark.org.tr/pub/amauefd/issue/41157/413487>
- Kırçalı, A.Ç., Özden, N. (2022). A Comparison of Plugged and Unplugged Tools in Teaching Algorithms at the K-12 Level for Computational Thinking Skills. *Tech Know Learn* (2022). <https://doi.org/10.1007/s10758-021-09585-4>
- Kite, V., Park, S., Wiebe, E. (2021). The code-centric nature of computational thinking education: A review of trends and issues in computational thinking education research. *SAGE Open*, 11(2), 215824402110164. <https://doi.org/10.1177/21582440211016418>
- Kim, B., Kim, T., & Kim, J. (2013). Paper-and-pencil programming strategy toward computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, 49(4), 437–459. <https://doi.org/10.2190/2FEC.49.4.b>
- Korkmaz, Ö., Çakır, R., & Özden, M. Y. (2015a). Bilgisayarca Düşünme Beceri Düzeyleri Ölçeğinin (BDBD) ortaokul düzeyine uyarlanması. *Gazi Eğitim Bilimleri Dergisi*, 1(2), 67–86.
- Korkmaz, Ö., Çakır, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). *Computers in Human Behavior*, 72. <https://doi.org/10.1016/j.chb.2017.01.005>



- Korkmaz, Ö., Çakır, R., Özden, M. Y., Oluk, A., & Sarioğlu, S. (2015b). Bireylerin bilgisayarca düşünme becerilerinin farklı değişkenler açısından incelenmesi. *Ondokuz Mayıs Üniversitesi Eğitim Fakültesi Dergisi*, 34(2), 68–87. <https://doi.org/10.7822/omuefd.34.2.5>
- Korucu, A. T., Gençtürk, A. T., & Gündoğdu, M. M. (2017). Examination of the computational thinking skills of students. *Journal of Learning and Teaching in Digital Age*, 2(1), 11–19. Retrieved January 18, 2021, from <https://dergipark.org.tr/tr/pub/joltida/issue/55466/760079>
- Kuo, W.-C., & Hsu, T.-C. (2019). Learning computational thinking without a computer: How computational participation happens in a computational thinking board game. *The Asia-Pacific Education Researcher*, 29, 67–83. <https://doi.org/10.1007/s40299-019-00479-9>
- Kutay, E., & Oner, D. (2022). Coding with Minecraft: The Development of Middle School Students' Computational Thinking. *ACM Transactions on Computing Education (TOCE)*, 22(2), 1–19. <https://doi.org/10.1145/3471573>
- Lahtinen, E., Mutka, K., & Jarvinen, H. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14–18. <https://doi.org/10.1145/1151954.1067453>
- Lee, J., Joswick, C., & Pole, K. (2022). Classroom play and activities to support computational thinking development in early childhood. *Early Childhood Education Journal*, 1–12. <https://doi.org/10.1007/s10643-022-01319-0>
- Lin, S. (2018). Item Analysis of English Grammar Achievement Test. *Mandalay University of Foreign Languages Research Journal* 9(1). Retrieved July 2, 2021, from <https://mufu.edu.mm/pdf/Vol9/Sandar%20Lin%20journal%202018.pdf>
- Lister, R. (2016). Toward a developmental epistemology of computer programming [Paper presentation]. *11th Workshop in Primary and Secondary Computing Education* (pp. 5–16). ACM. Münster, Germany. <https://doi.org/10.1145/2978249.2978251>
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Marie, S. M., Sreekala, E. (2015). Relevance of Item Analysis in Standardizing an Achievement Test in Teaching of Physical Science in B.Ed Syllabus, *Journal of Educational Technology* 12(3). <https://doi.org/10.26634/JET.12.3.3743>
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education*, 23(3), 239–264. <https://doi.org/10.1080/08993408.2013.832022>
- Merino-Armero, J. M., González-Calero, J. A., Cózar-Gutiérrez, R., & del Olmo-Muñoz, J. (2022). Unplugged Activities in Cross-Curricular Teaching: Effect on Sixth Graders' Computational Thinking and Learning Outcomes. *Multimodal Technologies and Interaction*, 6(2), 13. <https://doi.org/10.3390/mti6020013>
- Meyers, E. M. (2019). Guest editorial. *Information and Learning Sciences*, 120(5/6), 254–265. <https://doi.org/10.1108/ILS-05-2019-139>
- MoNE (2018a). *Ministry of national education of Turkey 2023 Education Vision*. MEB Yayınları. Retrieved February 10, 2022 from <https://2023vizyonu.meb.gov.tr/>
- MoNE (2018b). *Ministry of national education - information technology and software (ITS) (6th grade) curriculum*. MEB Yayınları. Retrieved February 10, 2022 from [www.eba.gov.tr](http://www.eba.gov.tr)
- Moreno-León, J., & Robles, G. (2015). *Analyze your scratch projects with Dr. Scratch and assess your computational thinking skills [Paper presentation]*. Scratch2015AMS, Amsterdam, Netherlands. Retrieved May 15, 2021, from <http://jemole.me/replication/2015scratch/InferCT.pdf>
- Nishida, T., Idosaka, Y., Hofuku, Y., Kanemune, & S., Kuno, Y. (2008). New methodology of information education with “Computer science unplugged”. In: Mittermeir R.T., Syslo M.M. (eds) *Informatics Education - Supporting Computational Thinking. ISSEP 2008. Lecture Notes in Computer Science*, vol 5090. Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-540-69924-8\\_22](https://doi.org/10.1007/978-3-540-69924-8_22)
- Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21<sup>st</sup> century skills when learning programming in K-9. *Education Inquiry*, 11(1), 1–17. <https://doi.org/10.1080/20004508.2019.1627844>
- OECD. (2020). *OECD Skills Outlook 2019: Thriving in a Digital World*. OECD Publishing. <https://doi.org/10.1787/df80bc12-en>
- Pala, F. K., & Mihçı Türker, P. (2019). The effects of different programming trainings on the computational thinking skills. *Interactive Learning Environments*, 1–11. <https://doi.org/10.1080/10494820.2019.1635495>
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books Inc.

- Pérez-Marín, D., Hijón-Neira, R., Babelo, A., & Pizarro, C. (2018). Can computational thinking be improved by using a methodology based on metaphors and Scratch to teach computer programming to children? *Computers in Human Behavior*. <https://doi.org/10.1016/j.chb.2018.12.027>
- Piaget, J. (1977). *The development of thought: Equilibration of cognitive structures*. (Trans A. Rosin). Viking.
- Ramadhan, H. A. (2000). Programming by discovery. *Journal of Computer Assisted Learning*, 16(1), 83–94. <https://doi.org/10.1046/j.1365-2729.2000.00118.x>
- Resnick, M., & Siegel, D. (2015). *A different approach to coding*. Bright/Medium.
- Rodriguez, B. (2015). *Assessing computational thinking in computer science unplugged activities [Master's thesis, Colorado School of Mines]*. Mountain Scholar. Retrieved February 10, 2022 from [https://mountainscholar.org/bitstream/handle/11124/169998/Rodriguez\\_mines\\_0052N\\_10899.pdf?sequence=1](https://mountainscholar.org/bitstream/handle/11124/169998/Rodriguez_mines_0052N_10899.pdf?sequence=1)
- Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: An experiment with sixth-grade students. *Interactive Learning Environments*, 28(3), 316–327. <https://doi.org/10.1080/10494820.2019.1612448>
- Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2018). Can computational talent be detected? Predictive validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*. <https://doi.org/10.1016/j.ijcci.2018.06.004>
- Romero, M., & Barberà, E. (2014). Computer-Based Creative Collaboration in Online Learning. *New Horizons in Web Based Learning* (pp. 330–336). Springer.
- Sendurur, P. (2019). Investigation of pre-service computer science Teachers' CS-unplugged design practices. *Education and Information Technologies*, 24, 3823–3840. <https://doi.org/10.1007/s10639-019-09964-6>
- Strnad, B. (2018). Introduction to the world of algorithmic thinking. *Journal of Electrical Engineering*, 6, 57–60. <https://doi.org/10.17265/2328-2223/2018.01.009>
- 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>
- Szlávi, P., & Zsakó, L. (2006). *Programming versus application*. In: Mittermeir, R.T. (Ed.), ISSEP 2006, LNCS 4226, 48–58. Retrieved May 13, 2021, from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.9408&rep=rep1&type=pdf>
- Tabachnick, B. G., & Fidell, L. S. (2013). *Using Multivariate Statistics* (6th ed.). Pearson.
- Taylor, M., Harlow, A., & Forret, M. (2010). Using a computer programming environment and an interactive whiteboard to investigate some mathematical thinking. *Procedia social and behavioral sciences*, 8, 561–570. <https://doi.org/10.1016/j.sbspro.2010.12.078>
- Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017). *Training computational thinking: Game-based unplugged and plugged-in activities in primary school [Conference session]*. 11th European Conference on Game-Based Learning ECGBL 2017, At Graz, Austria. Retrieved September 2, 2021, from [https://www.researchgate.net/publication/320491120\\_Training\\_Computational\\_Thinking\\_Game-Based\\_Unplugged\\_and\\_Plugged-in\\_Activities\\_in\\_Primary\\_School](https://www.researchgate.net/publication/320491120_Training_Computational_Thinking_Game-Based_Unplugged_and_Plugged-in_Activities_in_Primary_School)
- 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, <https://doi.org/10.1016/j.compedu.2021.104425>.
- Tsortanidou, X., Daradoumis, T., & Barberá-Gregori, E. (2022). Unplugged computational thinking at K-6 education: Evidence from a multiple-case study in Spain. *Education*, 3–13, 1–18. <https://doi.org/10.1080/03004279.2022.2029924>
- Turan, B. (2019). *The effect of problem based learning on problem solving and computational thinking skills in robot and game projects developed by secondary school students (No. 545841) [Master's thesis, Van Yüzüncü Yıl University]*. YÖK National Thesis Center. Available at <https://tez.yok.gov.tr/UlusaITezMerkezi/>
- Ung, L. L., Labadín, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education*, 177. <https://doi.org/10.1016/j.compedu.2021.104379>
- Unnikrishnan, R., Amrita, N., Muir, A., & Rao, B. (2016). Of Elephants and Nested Loops: How to Introduce Computing to Youth in Rural India. ACM Press, 137–146. <https://doi.org/10.1145/2930674.2930678>
- Waite, J. (2017). *Pedagogy in teaching Computer Science in schools: A Literature Review*. (After The Reboot: computing education in UK Schools). The Royal Society. Retrieved May 19, 2021, from <https://royalsociety.org/-/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf>

- Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to Professional programming languages in high school computer science classrooms. *Computers & Education*, 142, 103646. <https://doi.org/10.1016/j.compedu.2019.103646>
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. Retrieved October 13, 2021, from <https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, 366(1881), 3717–3725. <https://doi.org/10.1098/rsta.2008.0118>
- Wing, J. M. (2010). *Computational thinking: What and why?* Retrieved October 7, 2021, from <https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>
- Wing, J. M. (2014). *Computational thinking benefits society*. Retrieved December 24, 2018, from <http://socialissues.cs.toronto.edu/index.html%3Fp=279.html>
- Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in k-12 classrooms. *TechTrends*, 60(6), 565–568. <https://doi.org/10.1007/s11528-016-0087-7>
- Yallihep, M., & Kutlu, B. (2020). Mobile serious games: Effects on students' understanding of programming concepts and attitudes towards information technology. *Education and Information Technologies*, 25(2). <https://doi.org/10.1007/s10639-019-10008-2>
- Yıldırım, A. & Şimşek, H. (2016). *Sosyal bilimlerde nitel araştırma yöntemleri* (10th ed.). Seçkin Yayıncılık.
- Yıldız, H. D., Yılmaz, F. G., & Yılmaz, R. (2017). Examining the relationship between digital game preferences and computational thinking skills. *Contemporary Educational Technology*, 8(3), 359–369. <https://doi.org/10.30935/cedtech/6205>
- Zeybek, G. (2019). Determination of level of 21st Century learner skills use of high school students. *International Journal of Social Sciences and Education Research*, 5(2). <https://doi.org/10.24289/ijsser.505263>
- Zhan, Z., He, W., Yi, X., & Ma, S. (2022). Effect of Unplugged Programming Teaching Aids on Children's Computational Thinking and Classroom Interaction: with Respect to Piaget's Four Stages Theory. *Journal of Educational Computing Research*. <https://doi.org/10.1177/2F07356331211057143>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Authors and Affiliations

Elif Polat<sup>1</sup>  · Rabia Meryem Yılmaz<sup>2</sup> 

Elif Polat  
elifpolat.pltplt@gmail.com

<sup>1</sup> Rectorate Office, Middle East Technical University, 06800 Ankara, Turkey

<sup>2</sup> Department of Software Engineering, Engineering Faculty, Ataturk University, 25240 Erzurum, Turkey