Educational Policy Analysis and Strategic Research, V19, N4, 2024

Investigation of the Effect of Block-Based Applications on Computational Thinking Skills[1]

Aysegul YILMAZ, [2]
Ministry of National Education, Ankara, Turkey
Devkan KALECİ,[3]
İnönü University, Malatya, Turkey

Abstract

The research aims to explore the acquisition of Computational Thinking (CT) sub-skills among 5th and 6th grade secondary school students in Turkey through a block-based programming application, code.org. It seeks to understand if mastering these skills is essential for students globally. This study involved seven volunteer students selected through a mixed-methods suitability sample for the academic year 2021-2022. It utilized multiple tools: the Computational Thinking Skill Level Scale (CTSLS), observations, and interviews. Each student completed six tailored lessons via distance learning, which were recorded for observation and followed by interviews. Analysis of CT skills was conducted descriptively, considering the CTSLS scores before and after the lessons, observation of CT sub-skills during each session, and individual progress. Despite individual variations in CT skill display, there was no significant difference in the CTSLS scores pre- and post-intervention or between students of middle and high skill levels. The findings suggest that while individual progress in CT skills can be noted, the overall impact of the intervention on enhancing CT skills as measured by CTSLS is limited. This highlights the need for further investigation into the methods and effectiveness of teaching CT skills through block-based programming platforms.

Keywords: Computational Thinking, Computational Thinking Skills, Block-Based Applications, Code.org, Secondary School, Turkey
DOI: 10.29329/epasr.2024.1099.3

Submitted: 31 May 2024 **Accepted:** 13 September 2024 **Published:** 31 December 2024

[1] This article is derived from a master's thesis conducted by the first author under the guidance of the second author.

[2] İnönü University, School of Graduate Studies, Computer and Instructional Technologies Education, Malatya, Turkey, ORCID: https://orcid.org/ 0000-0001-8995-9709 **Correspondence**: aysegul.yilmaz@meb.gov.tr,

[3] Assoc. Prof., İnönü University, Faculty of Education, Computer and Instructional Technologies Education, Malatya, Turkey, ORCID: https://orcid.org/0000-0001-5642-4396 Email: devkan.kaleci@inonu.edu.tr,

Introduction

The 21st century is characterized by the capabilities of calculation, programming, and design, as it is an era that is progressing with the notion that "everything is programmable". The Institute for the Future (IFTF, 2020) categorized 21st century skills into six shifts, and according to emerging technological advancements, the majority of the 10 skills required in the future are technology-based. Among these skills is Computational Thinking (CT), which underscores the significance of programming. (Davies, Fidler & Gorbis, 2011) Therefore, CT is considered one of the essential skills for the 21st century. (Wing, 2010; Davies, Fidler & Gorbis, 2011; Mohaghegh, McCauley, 2016). In this context, CT stands out as one of the most important skills of the 21st century, and programming is considered as a basic tool in the development of this skill (Wong & Cheung, 2018). Research shows that programming activities can contribute to the development of problem solving, critical thinking and creative skills (Cetinkaya, 2019; Kukul & Çakır, 2020).

According to the Association of Computer Science Teachers (CSTA) and the International Association for Technology in Education (ISTE), CT defines skills as an approach to problem-solving that can be applied to a computer. It is a problem-solving methodology that enables situations to be automated, transferred, and applied and has the potential to be produced from a wide range of disciplines. The power of CT extends to all other reasoning skills, allowing for the development of useful computational tools and advancements in quantum physics, biology, and human-computer systems. (CSTA; HERE; Barr and Stephenson, 2011)

Through a program supported by the National Science Foundation (NSF), CSTA and the ISTE collaborated to address future educational needs. According to this study, all students must demonstrate proficiency in basic Computational Thinking Skills (CTS) by the time they graduate high school (CT Leadership Toolkit, 2011).

Many organizations, such as ISTE, NSF, CSTA, Google, Microsoft, and universities, are actively working on teaching CTS (Wing, 2010). Additionally, many countries have included CT skill-based training programs in their curricula and are conducting extensive research on this topic. Several studies have suggested that CTS should be incorporated into a 12-year curriculum. (Wing, 2010; NSF, 2010; Google, 2010; CSTA-ISTE, 2011)

Currently, there is a lack of comprehensive evaluations of CT abilities. (CT Teacher Resources, 2011) This study aimed to address the questions raised by Wing regarding the teaching of computer science concepts (Wing, 2017).

- What is the appropriate scope of computer science skills for students of various ages and developmental stages?
- At what age can the students effectively comprehend and apply these skills?
- At what point did they typically acquire computer science concepts in their education?

- What is the most effective approach for teaching computational thinking to students at the K-12 level?
- How has the teaching of computational thinking progressed thus far, and has the intended outcomes been achieved?

The primary objective of this research was to investigate the acquisition of sub-skills related to CT by secondary school students in Turkey through the use of a block-based application (code.org), which aims to promote the development of these skills among a broad audience. Specifically, this study sought to determine the extent to which students acquire various sub-skills of CT and the specific sub-skills that they learn.

To fulfill the objectives of this research, it is necessary to address the following questions:

- What constitutes CTS?
- What sub-skills are included within CTS?

When breaking down a problem into smaller components for resolution, the following questions arise.

- How much development do students experience in terms of skills using this application, based on their individual needs?
- How do students acquire skills tailored to their individual needs and development by using this application?
- Does the application's provision of CTS enhance students' CTS?

Method

Research Design

This study employed a mixed method which is a research approach that integrates the collection and analysis of both quantitative and qualitative data in a single study or series of studies (Creswell et al., 2006). In this study, a sequential transformational design was employed, in which quantitative data were first collected and analyzed, followed by the collection and analysis of qualitative data. All data collected in the interpretation and discussion sections were combined and evaluated (Creswell et al. 2003).

Participants

For this study, 5th and 6th grade students attending secondary schools in Turkey during the academic year 2021-2022 were selected as a sample. Since the study covered an extensive period, required in-depth research, and the age group of the students was small, the sample selection was voluntary as "convenience sampling." (Teddlie and Yu, 2007)

Six students (5 girls and 2 boys) participated in the research, 5 of them being in the 5th grade and 2 in the 6th grade. Of the six students, 4 had taken information technology courses and 2 had

previously received coding training. All but one student reported knowing how to use a computer. As the courses were provided through distance education, the students were selected from different schools resulting in a diverse mix of social status and academic success among the students. This increased the diversity of the study and allowed for individual differences among participants.

Data Collection

To enhance the validity and reliability of the study, observations, interviews, and survey data-collection techniques were employed for each sub-problem, which complemented each other. The triangulation method was used to conduct both observations and interviews, using different data collection techniques and methods (Holloway & Wheeler, 1996; Mays & Pope, 2000).

Computational Thinking Skill Levels Scale (CTSLS)

The "Computational Thinking Skill Levels Scale" developed by Korkmaz, Çakır, and Özden (2015) was utilized in this study. This scale was specifically designed for secondary school students and comprised a five-point Likert-type questionnaire containing 22 items. The Cronbach's alpha reliability coefficient for the scale was 0.809, and the values for its sub-dimensions ranged from 0.640 to 0.867. These results indicate that this study is appropriate for the intended purpose.

The Likert-type scale used in this study was rated on a scale ranging from 1 (never) to 5 (always). The raw scores obtained from the students' answers were converted into standard scores, which were then used to determine the CTS. The formula used to convert the raw score to the standard score is as follows: × standard score = (× raw score/number of measuring items) × 20. Standard scores were then used to determine the corresponding CTS levels, which were classified as follows: 20-51 for Low Level, 52-67 for Intermediate CTS, and 68-100 for High Level CTS.

Structured Observation and Interview Form

The items on the observation form were categorized based on the concepts of CT outlined in Table 1, and the observable behaviors on the code.org platform were assessed based on these classifications.

Interviews were conducted with the students at the conclusion of the course, during which their feedback on the observation form was obtained. To this end, interview form questions were developed based on problem situations and observation form items. These questions and the behavioral items were developed after consulting an expert and five Information Technology teachers who had previously provided coding instructions to secondary school students.

Each entry on the form was designated as "Observed (O), Not Observed (N), or Partially Observed (P)" using a Likert-type scale with three levels of agreement.

Table 1. CT Concepts constituting the Observation Form items

CT Concepts	Definition of Concepts
Data Collection	Proper Information Collection Process
Data Analysis	Make sense of information, draw patterns and draw conclusions
Data Representation	Organize and depict information through appropriate graphs, charts, words, or images
Problem Decomposition	Break down tasks into smaller manageable chunks
Abstraction	Define the main idea by reducing complexity
Algorithms and Procedures	A series of sequential steps to solve a problem or to achieve an end
Automation	Having machines or computers to do tedious or repetitive tasks
Simulation	A notation or transactional model. It also involves providing experiments that use models.
Parallelization	Organize resources to perform concurrent tasks aimed at achieving a common goal

Data Analysis

The quantitative and qualitative data obtained were analyzed separately and interpreted together in the findings section. Based on the data obtained from the CTSLS, the raw score was converted into a standard score according to the leveling formula presented in this study, and student levels were determined. The observation and interview forms were analyzed using a descriptive analysis method.

Application and Code.org Lessons

Prior to the commencement of the course, which comprised seven individual courses as shown in Figure-1, the demographic characteristics of the students were assessed and the corresponding profiles were established. After the initiation of the course, CTSLS was implemented and the CT proficiency levels of the students were determined. Each session was documented, and the entire curriculum was tracked using a structured observation form. At the conclusion of the course, a structured interview form was administered to students to obtain feedback. Independent observations and interviews were conducted for each course. At the conclusion of the program, the CTSLS was administered once more to the students and changes in their CT levels were examined.

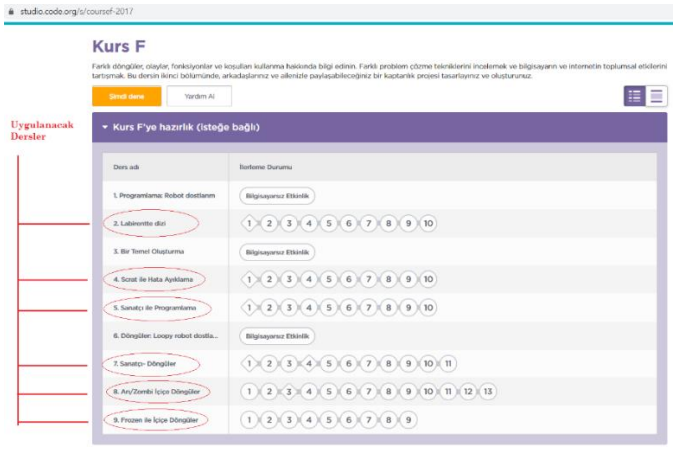

Figure 1: Course F to apply (code.org screen)

Findings and Discussion

The study's results were analyzed based on the CTSLS, the length of each course completed individually, the CTS of the students for each course, the CT sub-skills exhibited by the students at each CTS level, and the students' individual progress.

Table 2.

CTSLS student first test results

Student	Level	Scale Standard Score
Student-1	High Level	**79,09**
Student-2	Intermediate	**61,82**
Student-3	Intermediate	**56,36**
Student-4	Intermediate	**62,73**
Student-5	High Level	**68,18**
Student-6	High Level	**68,18**
Student-7	High Level	**65,45**

Table 3.

CTSLS student final test results

Student	Level	Scale Standard Score
Student-1	High Level	**74,55**
Student-2	High Level	**71,82**
Student-3	Intermediate	**54,55**
Student-4	Intermediate	**55,45**
Student-5	Intermediate	**64,55**
Student-6	High Level	**76,36**
Student-7	High Level	**77,27**

20-51: Low Level; 52-67: Intermediate; 68-100: High Level.

Based on CTSLS data, the students in this study were categorized as having intermediate or high levels. As indicated in Tables 2 and 3, the outcomes of the initial and concluding examinations demonstrated that although some students exhibited changes in their levels, others did not experience any changes. Nevertheless, there was a noticeable improvement in the standardized scores of all students.

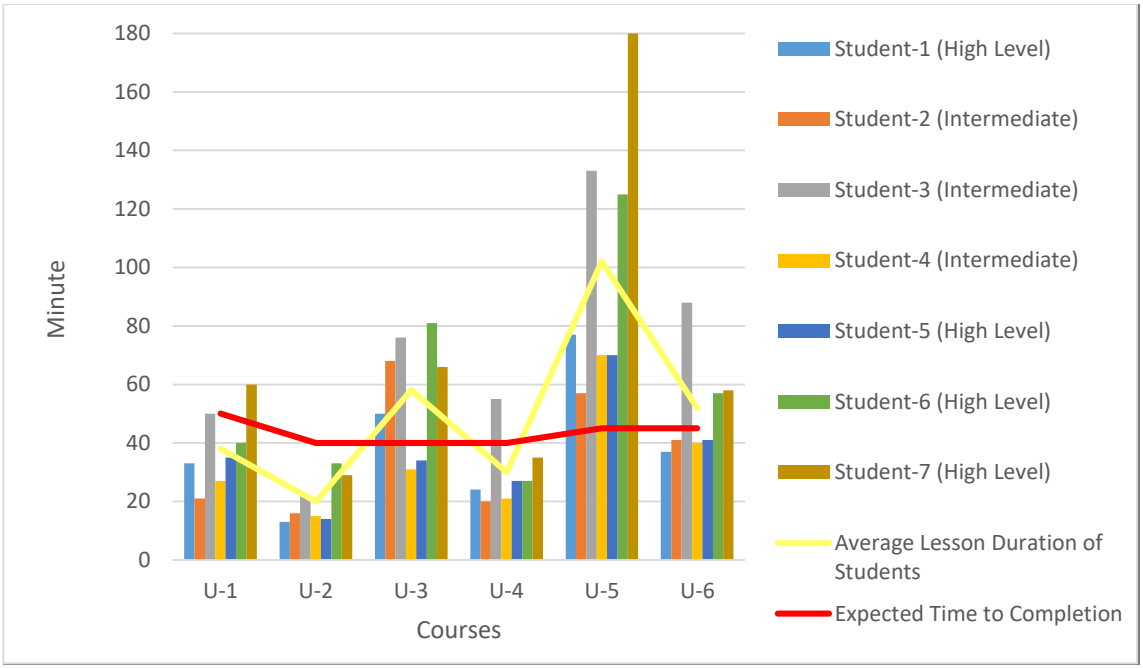

Figure 2: Students' course practice times

The data presented in Figure 2 indicate that the majority of the students completed the U-1 and the U-2 lessons focusing on debugging ahead of the schedule within the expected. They completed the U-4 course including basic loops faster than expected. However, students generally struggled with other courses and continued at a slower pace. Notably, all students struggled with the nested-loop concept in U-5 course which is longest time spent.

Table 4. Findings of CTS Observation Items of All Students in Practice Courses

Measurement Values	Observed (**O**) Partially observed (**P**) Not observed (**N**)	**Application Duration**	**Data Collection**	**Data Analysis**	**Data Representation**	**Problem Decomposition**	**Abstraction**	**Algorithms and Procedures**	**Automation**	**Simulation**	**Parallelization**
Lessons											
U-1		38’	**P**	**O**	**O**	**P**	**P**	**P**	**N**	**O**	**P**
U-2		20’	**O**	**O**	**O**	**O**	**O**	**O**	**N**	**O**	**O**
U-3		58’	**O**	**O**	**O**	**O**	**P**	**O**	**P**	**O**	**O**
U-4		30’	**O**	**O**	**O**	**O**	**P**	**O**	**O**	**O**	**O**
U-5		102’	**O**	**O**	**O**	**P**	**P**	**O**	**P**	**O**	**P**
U-6		52’	**O**	**P**	**O**	**P**	**P**	**O**	**O**	**O**	**P**

According to Table 4, the students in the U-1 course exhibited skills in data analysis, representation, and simulation as well as in adapting what they had learned to new situations, parsing problems, abstraction, and algorithms. However, only a partial observation was made regarding this abstraction. In the U-2 course, students made progress after being applied to real-world situations, and all skills except automation were observed. In the U-3 course, students demonstrated abstraction by visualizing events in their minds, and despite struggling with automation, they were able to showcase all skills. This demonstrates that the students progressed through learning. As students had to use their knowledge of mathematics in the course, it took longer than expected, and the course structure was different from that of previous courses, prolonging the time it took to become accustomed to. Although the U-3 course was the first in a loop structure, students demonstrated all skills except for partial abstraction in the U-4 course. The U-5 course, which included nested loops, was the longest course, and students who made progress in CTS demonstrated some skills. Students who could parse problems, automate, and adapt to new situations struggled with abstraction. In the U-6 course, where nested loops became even more complex, students were challenged to analyze and abstract data, partially demonstrating how to parse and parallelize problems. It can be concluded from Table 5 that the students demonstrated the partial processes of problem decomposition, abstraction, automation, and parallelization throughout the course.

Table 5. Overall Evaluation of CTS Observation Items of All Students in Practice Courses

Measurement Values Observed (O) Partially Observed (P) Not observed (N)	**Measurement Values**	**Data Collection**	**Data Analysis**	**Data Representation**	**Problem Decomposition**	**Abstraction**	**Algorithms and Procedures**	**Automation**	**Simulation**	**Parallelization**
Frequency Values	O	**5**	**5**	**6**	**3**	**1**	**5**	**2**	**6**	**3**
	P	**1**	**1**	**-**	**3**	**5**	**1**	**2**	**-**	**3**
	N	**-**	**-**	**-**	**-**	**-**	**-**	**2**	**-**	**-**
	Avg. Duration									
Overall Rating	405'	O	O	O	P	P	O	P	O	P

The CTS results of students across different courses and based on their CTSLS levels were analyzed. It was found that in the U-1 course, there was no difference in the CTS displayed by middle- and high-level students, indicating similar skill levels. In the U-2 course, high-level students demonstrated problem decomposition, abstraction, and algorithms, while all students, except one, partially exhibited some of these skills. It was observed that higher-level students in the U-2 course performed better than intermediate students in terms of these skills. In the U-3 course, higher-level

students showed superiority in data collection, analysis, and representational skills compared with intermediate students, although they exhibited similar traits in other skills. In the U-4 course, intermediate students demonstrated full proficiency in automation, simulation, and parallelization skills, whereas there were no significant differences in other skills between higher and intermediate students. Additionally, there were no significant differences between intermediate- and higher-level students in U-5 and U-6 courses.

Table 6. Evaluation of CTSLS Observation Items of Each Students in All Courses

Measurement Values Observed (**O**) Partially observed (**P**) Not observed (**N**)	**Application Duration (minutes)**	**Data Collection**	**Data Analysis**	**Data Representation**	**Problem Decomposition**	**Abstraction**	**Algorithms and Procedures**	**Automation**	**Simulation**	**Parallelization**
Students										
Student-1 (High Level) (6 Grade)	234'	**O**	**P**	**O**	**P**	**P**	**O**	**P**	**O**	**O**
Student-2 (Intermediate) (5 Grade)	223'	**O**	**O**	**O**	**O**	**O**	**O**	**P**	**O**	**O**
Student-3 (Intermediate) (6 Grade)	424'	O	P	O	P	P	P	P	P	P
Student-4 (Intermediate) (5 Grade)	204'	**O**	**O**	**O**	**P**	**P**	**O**	**P**	**O**	**P**
Student-5 (High Level) (5 Grade)	221'	**O**	**O**	**O**	**O**	**P**	**O**	**O**	**O**	**O**
Student-6 (High Level) (5 Grade)	363'	**O**	**O**	**O**	**O**	**P**	**O**	**P**	**O**	**O**
Student-7 (High Level) (5 Grade)	428'	**O**	**P**	**O**	**P**	**P**	**P**	**P**	**P**	**P**

According to Table 6, which provides an overview of each student's proficiency in various skills, all students exhibited data collection and data representation abilities across all courses. However, their proficiencies in data analysis, algorithm building, simulation, and parallelization are partial. Furthermore, only two students, one at an intermediate level and one at a high level, demonstrated all skills, except for one in the overall course. These students almost fully exhibited most of their skills. One high-level student fully exhibited all skills, whereas only two skills were partially demonstrated by this student. Additionally, two students, one at a higher level and one at an intermediate level, exhibited

four skills partially and the remaining skills completely. Finally, two students in total, including one at an intermediate level and one at a high level, fully demonstrated the two skills and partially demonstrated the others. They achieved the lowest performance by demonstrating their skills.

Upon evaluating the individual performance of students in courses, it becomes clear that certain skills pose difficulties for all students collectively, whereas others exhibit distinct behaviors. Upon closer examination of the overall situation of Student-1 (S1), it becomes evident that S1 encounters difficulties in gathering and processing data, but remains proficient in presenting it. However, S1 faced challenges in abstraction, parsing problems, and displaying automation skills. With regard to the algorithm, simulation, and application of the learned skills to new situations, S1 demonstrated competency. When faced with an unfamiliar task, S1 resorts to trial and error to achieve a correct outcome. S1 recognizes the difficulty in abstraction and consequently immerses themselves in the context by utilizing their hands or mouse to solve the problem step-by-step. The construction of nested loop structures poses a challenge for S1 because of its inability to decipher the intricate problem structures. Only after comprehending the appropriate course of action can S1 employ an algorithmic framework to reach a conclusion.

Upon examining the general situation of Student-2 (S2), it is evident that S2 possesses the skills of collecting data, analyzing and representing the collected data, abstracting problems, developing algorithms, and adapting learned knowledge through simulation to novel situations. S2 demonstrates the understanding and application of the loop structure, although difficulties arise with complex nested-loop structures. Despite this, S2 remains open to exploring new problem situations and can devise solutions through logical reasoning even in the absence of prior exposure to loop structures. With teacher guidance, S2 was able to independently apply a repeat block within the loop structure. Ultimately, S2 advanced their learning by mastering the CTS.

Upon evaluating the general circumstances of Student-3 (S3), it is apparent that the student possesses the ability to collect data but encounters difficulties in effectively analyzing it. While S3 demonstrated proficiency in applying the problem situation as required, challenges arose when attempting to view the problem comprehensively when solving intricate issues. Nonetheless, S3 divides the problem into more manageable components. However, S3 experienced difficulties in constructing algorithms step by step and creating loop structures. Consequently, S3 frequently makes errors when applying prior solutions to similar situations because of its inability to visualize the problem as a whole. Additionally, S3 displayed reluctance in the approach when faced with new situations and when solving complex problems. Research conducted by İbili and Günbatar (2020) revealed that the utilization of block-based programming tools has a favorable impact on the self-confidence of students who have easy access to computers and possess higher self-efficacy perceptions in CTS. However, students such as S3, who did not take IT classes or block-based programming lessons, may face disadvantages.

Upon evaluating the overall performance of Student-4 (S4), it is evident that the students possessed proficient skills in data collection and analysis. However, challenges in abstraction have hindered S4's ability to demonstrate problem-solving skills in problem discrimination to the fullest extent. Moreover, students face difficulties in algorithm development, automation, and in adapting their learned skills to new situations. Despite these challenges, S4 demonstrated its ability to effectively perform simulations. Although students have attempted to break down complex problem structures into smaller components, they have struggled to determine the appropriate placement of blocks when creating nested loop structures. Additionally, S4 faced difficulties in applying the learned skills to new situations and was unable to fully exhibit the ability to parallelize.

Upon analyzing the overall performance of Student-5 (S5), it can be observed that the student possesses the necessary abilities for data collection and analysis. Although S5 may have some challenges with abstract thinking, the student can develop algorithms by breaking down problems and utilizing automation. Additionally, S5 demonstrated the capacity to adapt and apply the learned concepts in new situations. Initially, students struggled in the early lessons, but S5 was able to showcase their skills more effectively as they progressed. Despite the difficulties encountered in the abstraction process, S5 simplified the complex problem structures using an algorithmic approach. Ultimately, Student-5 was successful in completing the CTS course.

Upon analyzing the general circumstances of Student-6 (S6), it becomes evident that the student possesses the ability to gather and interpret data, as well as the capacity to resolve problems systematically by breaking them down into smaller components. In addition, students can create or write algorithms to arrive at solutions. Although S6 may struggle to comprehend the broader picture in complex situations, the student demonstrated an effort to attain the desired outcome by drawing or erasing their steps and identifying errors. This indicates that S6 successfully addressed the problems by utilizing concrete drawings, as students may have difficulty with abstract concepts and thoughts. This methodical approach and students' step-by-step progress have enabled them to simplify complex situations. Ultimately, it was observed that Student-6 encountered challenges in exhibiting CTS but made commendable progress through diligent efforts.

Upon analyzing the general circumstances of Student-7 (S7), it becomes evident that the student possesses the capacity to gather data but struggles with interpreting the information collected. When faced with a problem, S7 endeavored to solve it by breaking it down into manageable steps. However, students encounter difficulties in identifying solutions in complex problem situations and often resort to applying previous solutions in a similar manner. Furthermore, S7 had trouble creating a sequence of algorithmic steps as the student struggled to visualize the problem at hand. To better comprehend the problem, S7 often attempted to concretize it through drawings or writing. Nevertheless, students encounter challenges in adapting existing state data to new situations as well as applying what was previously learned to diverse contexts.

Conclusion and Recommendations

According to the pretest–post-test analysis, there was no significant discrepancy between the students, which is consistent with the findings of several previous studies (Grover et al., 2016; Uslu, 2018). It was determined that the Code.org courses and CTS training received by the students did not prove to be beneficial in a significant manner. In addition, there was no marked difference in CTS between middle- and high-level students.

Upon examination of the students' completion times, all students were able to finish the initial course within the anticipated timeframe. They completed the courses in a longer time than expected because they had difficulty with new problem situations in other courses. The most time was spent in the course that included the nested loop structure.

In general, the students were successful in collecting, analyzing, and representing the data, designing algorithms, and simulating them. However, they faced challenges in parsing problems, particularly in the areas of abstraction, automation, and adapting what they learned to similar situations. Just like another study, students generally struggled with the loop structures because of the significance of CT and abstraction skills in mathematical thinking (Grover et al., 2016). In conclusion, it is evident that students struggle to visualize situations in their minds while abstracting, which hinders their ability to separate problems and perceive them as a whole.

Based on the findings of this study, the following recommendations were provided for teaching CTS:

- For the nested loop structure, lessons should be more detailed and simplified by distributing them over time.
- Students at the secondary school level have difficulties in abstraction. They should be encouraged to embody situations by drawing, writing, or using body language.
- A scale should be developed for CTS in order to incorporate personal creative projects, illuminate transaction processes, control different stages, and value multiple ways of knowing of students (Brennan & Resnick, 2012). Additionally, this can be supported by artificial intelligence.
- Conducting a similar study including grade level would be beneficial to examine the difference in development according to age.

Policy Implications

According to findings of this study, the suggestions and implications for how CTS can contribute to educational policies are categorized as follows:

- Integration of CTS into education programs

CTS should be integrated not only in the information technologies course but also in all courses. Like STEM and STEAM studies that support the interdisciplinary approach, CT has a structure that can support many fields such as Mathematics, Science and Art. Skills such as problem solving and analytical thinking, which are sub-skills of CT, are among the skills needed in all fields. Especially as seen in this study, students developed various skills for inferencing, putting themselves in the character's shoes and repetitive operations. These skills are needed in many areas of the curriculum and it is recommended to develop and support them with CTS.

CTS need to be integrated into the curriculum in a way that is appropriate to the level of the student with the spiral learning. This study with secondary school students has shown that they have difficulty in demonstrating the necessary skills according to their age level. However, for students who develop abstract thinking skills with age, the level of demonstrating these skills becomes easier and more understandable. For this reason, it would be more effective to support effective learning if CTS are not only aimed at students at a certain level, but also distributed to all levels in accordance with the level of the student.

Studies should be carried out in order to make learning permanent by giving feedback with measurement and evaluation methods of computational thinking skills. In all learning environments, feedback and tracking learning is an important phenomenon. For this reason, CT should be integrated into the curriculum in a well-structured way in terms of measurement and evaluation.

- CTS in teacher education

In the light of this study, in order to create the guidance and learning environment that students need to develop their CTS, teachers need to be trained in this direction and to have this skill. If CTS are to be integrated into all disciplines, it is necessary to include them in the programs of all faculties of educational sciences. For this reason, curricula should be created in faculties of education to enable students to acquire CTS and teachers should graduate with this ability.

All educators who are currently teaching should be supported with in-service training programs to have CTS and policies should be developed in this direction.

- Developing CTS with artificial intelligence support

As seen in this study, CTS are difficult to assess and monitor because they involve many high-level thinking skills. For this reason, presenting CT with artificial intelligence systems for a more systematic, accurate measurement and feedback will be a supportive element for learning.

In order to understand the digital world, CTS are expected to support the development of students to get used to artificial intelligence systems, one of the technologies of the future, and to achieve artificial intelligence literacy. Students and teachers who have an analytical logic structure and computer logic are expected to adapt to the use and literacy of artificial intelligence more easily. For this reason, efforts can be made to provide CTS to teachers and students in this direction.

With the integration of these policy approaches into education, teachers and students with 21st century skills will respond to the requirements of the age.

Conflict of Interest

The authors declare no conflict of interest.

Funding Details

No funding was received for conducting this study.

Ethical Statement

This study was approved by the Ethics Commission of İnönü University and published in National Thesis Center (approval number: 10435700) in 2021. All procedures were carried out in accordance with the relevant guidelines and regulations. Informed consent was obtained from all study participants.

Credit Author Statement

Yılmaz contributed to the design and implementation of the research and to the analysis of the results and to the writing of the manuscript. Kaleci supported and supervised the research.

References

Akcaoglu, M., & Koehler, M. J. harr(2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. *Computers & Education, 75*, 72-81.

Aydoğdu, E. (2019). Bilgisayarsız etkinlikler sürecinde öğrencilerin algoritmik düşünme becerilerinin incelenmesi. *Trabzon Üniversitesi. Eğitim Bilimleri Enstitüsü, BÖTE Anabilim Dalı (Yüksek lisans tezi).*

Baki, A., & Gökçek, T. (2012). Karma yöntem araştirmalarina genel bir bakiş. *Elektronik Sosyal Bilimler Dergisi*, 11(42), 1-21.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1), 48-54.

Başkale, H. (2016). Nitel araştırmalarda geçerlik, güvenirlik ve örneklem büyüklüğünün belirlenmesi. *Dokuz Eylül Üniversitesi Hemşirelik Fakültesi Elektronik Dergisi*, 9(1), 23-28.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada* (Vol. 1, p. 25).

Brown, W. (2015). Introduction to algorithmic thinking.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing, 1*(2), 67-69.

Cansu, F. K., & Cansu, S. K. (2019). An overview of computational thinking. *International Journal of Computer Science Education in Schools*, 3(1), 17-30.

Cetinkaya, L. (2019). The effects of problem based mathematics teaching through mobile applications on success. Egitim ve Bilim, 44(197).

Code.org. (2021, August 4). *Code.org*. https://code.org/

Creswell, J. W., Clark, V. L. P., Gutmann, M. L., & Hanson, W. E. (2003). Advanced mixed. *Handbook of mixed methods in social & behavioral research*, 209, 209-240.

Creswell, J. W., Shope, R., Plano Clark, V. L., & Green, D. O. (2006). How interpretive qualitative research extends mixed methods research. *Research in the Schools, 13*(1), 1-11.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers.

CSTA, I. (2011). Computational thinking teacher resources. *National Science Foundation under Grant No. CNS-1030054.*

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/~ CompThink/resources/TheLinkWing. pdf.*

Curzon, P., Black, J., Meagher, L. R., & McOwan, P. W. (2009). cs4fn. org: Enthusing students about Computer Science. *Proceedings of Informatics Education Europe IV*, 73-80.

Davies, A., Fidler, D., & Gorbis, M. (2011). Future work skills 2020. *Institute for the Future for University of Phoenix research Institute*, 540.

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30.

Erdoğan, İ. (2003). Pozitivist metodoloji: Bilimsel araştırma tasarımı, İstatiksel yöntemler, Analiz ve yorum. Erk.

Erümit, A. K., Şahin, G., & Karal, H. (2020). YAP Programlama Öğretim Modelinin Öğrencilerin Bilgi-İşlemsel Düşünme Becerilerine Etkisi. *Kastamonu Education Journal, 28*(3), 1529-1540.

Fidler, D., & Williams, S. (2016). Future skills. Update and literature review. *Institute for the Future.*

Futschek, G. (2006, November). Algorithmic thinking: The key for understanding computer science. *In International conference on informatics in secondary schools-evolution and perspectives* (pp. 159-168). Springer, Berlin, Heidelberg.

Glesne, C. (2016). *Becoming qualitative researchers: An introduction.* Pearson. One Lake Street, Upper Saddle River, New Jersey 07458.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Grover, S., Pea, R., & Cooper, S. (2016, February). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 552-557).

Holloway, I., & Wheeler, S. (1996). *Qualitative research for nurses.* (No Title).

Ibılı, E., & Günbatar, M. S. (2020). Computational thinking skills self-efficacy perceptions in secondary education: A review of the effectiveness of the New Information Technology and Software Curriculum. *Trakya Eğitim Dergisi*, *10*(2), 303-316.

ISTE, I., & CSTA, C. (2011). Operational definition of computational thinking for K-12 education. *National Science Foundation.*

ISTE, C. (2011). Computational Thinking in K–12 Education leadership toolkit. *Computer Science Teacher Association: http://csta. acm. org/Curriculum/sub/CurrFiles/471.11 CTLeadershipt Toolkit-SP-vF. pdf*

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Educational Researcher*, *33*(7), 14-26.

Korkmaz, Ö., Çakır, R., & Özden, M. Y. (2015). Bilgisayarca düşünme beceri düzeyleri ölçeğinin (BDBD) ortaokul düzeyine uyarlanması. *Gazi Eğitim Bilimleri Dergisi*, 1(2), 143-162.

Kukul, V., & Çakır, R. (2020). Exploring the development of primary school students' computational thinking and 21st century skills through scaffolding: Voices from the stakeholders. International Journal of Computer Science Education in Schools, 4(2), 36-57.

Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 260-264).

Mays, N., & Pope, C. (2000). Assessing quality in qualitative research. *BMJ*, *320*(7226), 50-52.

Mohaghegh, D. M., & McCauley, M. (2016). Computational thinking: The skill set of the 21st century.

Özçinar, H. (2017). Hesaplamali düşünme araştirmalarinin bibliyometrik analizi. *Eğitim Teknolojisi Kuram ve Uygulama*, 7(2), 149-171.

Özyol, B. (2019). *Bilgi-işlemsel düşünme becerisinin kazandırılmasına yönelik bir ortam tasarımı ve geliştirilmesi* (Master's thesis).

Seehorn, D., & Clayborn, L. (2017, March). CSTA K-12 CS Standards for all. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 730-730).

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definition.

Serim, E. Ü. (2019). *Oyunlaştırma yöntemiyle tasarlanan kodlama eğitimi ile öğrencilerin hesaplamalı düşünme becerileri ve kodlamaya ilişkin öz-yeterlik algılarının incelenmesi* (Master's thesis, Balıkesir Üniversitesi Fen Bilimleri Enstitüsü).

Sneed, J., & Ace Morgan, D. (1999). Evaluating the verbal, quantitative, and problem-solving skills of students entering the accounting curriculum. *Management Research News*, 22(4), 22-27.

Yıldırım, A., Şimşek, H., & Sosyal Bilimlerde Nitel Araştırma Yöntemleri, S. (2003). *Yayıncılık. Baskı, Ankara*.

Uslu, N. A. (2018). Görsel programlama etkinliklerinin ortaokul öğrencilerinin bilgi-işlemsel düşünme becerilerine etkisi. *Ege Eğitim Teknolojileri Dergisi, 2*(1), 19-31.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

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.

Wing, J. M. (2014). Computational thinking benefits society. *40th anniversary blog of social issues in computing*, 2014, 26.

Wing, J. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7-14.

Wong, G.K., & Cheung, H. (2018). Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. Interactive Learning Environments, 28, 438 - 450.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 1-16.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, 60(4), 55-62.

Yıldırım, A., & Şimşek, H. (2011). Sosyal bilimlerde araştırma yöntemleri. *Ankara: Seçkin Yayınları.*

Yıldız, M., Çiftçi, E., & Karal, H. (2017). Bilişimsel düşünme ve programlama. *Eğitim Teknolojisi Okumaları, 5*, 75-86.

Zsakó, L., & Szlávi, P. (2012). ICT competences: Algorithmic thinking. *Acta Didactica Napocensia, 5*(2), 49-58.