DOI: 10.5961/highereducsci.936833

Does Teaching Programming Have an Effect on Computational Thinking Skill? A Longitudinal Study

Programlama Öğretiminin Bilgi İşlemsel Düşünme Becerisi Üzerine Etkisi Var mıdır? Boylamsal Bir Araştırma

Mithat ELÇİÇEK

ABSTRACT

This study aims to examine the effect of teaching programming at a higher education level on computational thinking skills and its sub-dimensions. In the study, an explanatory design, one of the mixed-method research types in which quantitative and qualitative data are used together, was used. In this context, the change of the computational thinking skill variable for a single group taken from a starting point in certain time intervals was examined with a quasi-experimental method. The study was conducted with a total of 42 undergraduate students including 18 males and 24 females. Quantitative data were collected with the computational thinking skills scale developed for university students, and qualitative data were collected through semi-structured focus group interviews. The data were analyzed using the Friedman test, Wilcoxon signed-rank test, and descriptive content analysis techniques. In the measurements made, it was concluded that teaching programming had a significant effect on computational thinking skill and its creativity, algorithmic thinking, and problem-solving sub-dimensions, but had no significant effect on collaborative and critical thinking sub-dimensions.

Keywords: Teaching programming, Computational thinking skill, Longitudinal study

ÖZ

Bu çalışmanın amacı, yükseköğretim düzeyinde gerçekleştirilen programlama öğretiminin bilgi işlemsel düşünme becerisi ve alt boyutları üzerindeki etkisini incelemektir. Çalışmada, nicel ve nitel verilerin birlikte kullanıldığı karma yöntem araştırma türlerinden açıklayıcı desen kullanılmıştır. Bu kapsamda bilgi işlemsel düşünme becerisi değişkeninin tek bir grup için, bir başlangıç noktasından alınarak belirli zaman aralıklarındaki değişimi yarı deneysel yöntemle incelenmiştir. Çalışma 18 erkek, 24 kadın olmak üzere toplam 42 lisans öğrencisiyle yürütülmüştür. Nicel veriler üniversite öğrencilerine yönelik geliştirilen bilgi işlemsel düşünme becerisi ölçeğiyle, nitel veriler ise yarı yapılandırılmış odak grup görüşmeleriyle toplanmıştır. Veriler Friedman testi, Wilcoxon işaretli sıralar testi ve betimsel içerik analizi teknikleri doğrultusunda analiz edilmiştir. Yapılan ölçümlerde programlama öğretiminin bilgi işlemsel düşünme becerisi ve bilgi işlemsel düşünme becerisinin yaratıcılık, algoritmik düşünme ve problem çözme alt boyutları üzerine anlamlı bir etkisinin olduğu, işbirliklilik ve eleştirel düşünme alt boyutları üzerine ise anlamlı bir etkisinin olmadığı sonucuna ulaşılmıştır.

Anahtar Sözcükler: Programlama öğretimi, Bilgi işlemsel düşünme becerisi, Boylamsal çalışma

Elçiçek M., (2022). Does teaching programming have an effect on computational thinking skill? A longitudinal study. *Journal of Higher Education and Science/Yükseköğretim ve Bilim Dergisi, 12*(1), 40-50. https://doi.org/10.5961/highereducsci.936833

Mithat ELÇİÇEK (✉)
ORCID ID: 0000-0003-1845-7271

Siirt University, Department of Graphic Design, Siirt, Turkey
Siirt Üniversitesi, Grafik Tasarım Bölümü, Siirt, Türkiye
mithat_elcicek@siirt.edu.tr

Received/*Geliş Tarihi* : 12.05.2021
Accepted/*Kabul Tarihi* : 14.01.2022

INTRODUCTION

With digital technologies becoming a part of daily life, problem situations encountered in daily life have turned into a more complex structure compared to the past. In the face of this complex structure, individuals are expected to have a set of new competencies that are qualified as 21st-"century skills. Basic skills such as life, profession, learning, innovation, media, and technology literacy are expressed as 21st-century skills (Partnership for 21st Century Skills, 2009). These skills, which do not have fixed content, vary according to living conditions. One of these skills that researchers and educators have emphasized especially in recent years is computational thinking skill (Saavedra & Opfer, 2012). It is stated that although it was first put forward by Seymour Papert in 1980, the popularization process started with the work of Janette Wing in 2006. This is observed in and supported by systematic review studies on computational thinking skills. It is observed that there has been a significant increase in the number of studies conducted in the last decade (Hsu, Chang, & Hung, 2018; Kalelioğlu, Gülbahar, & Kukul, 2016; Tosik-Gün & Güyer, 2019).

Computational thinking skill is considered as a kind of problem-solving skill in terms of the processes it involves (Yadav, Hong, & Stephenson, 2016). For, it includes the processes of making problems solvable with the help of computers or other tools, organizing data, analyzing, automating solutions, and generalizing by transferring them to different problems (Doleck et al., 2017). On the other hand, Wing (2006) stated that computational thinking skill is not a skill that only concerns computer scientists and that everybody should have skills such as reading, writing, and basic math. Researchers have reached a consensus over the years that computational thinking skills should be acquired by students (Aho, 2012; Barr, Harrison, & Conery, 2011; Román-González, Pérez-González and Jiménez-Fernández, 2017; Korkmaz, Çakır, & Özden, 2015). Researchers agree that computational thinking skill should be given as a basic literacy skill at all levels of education from pre-school to higher education (Barr & Stephenson, 2011; Román-González, Pérez-González, & Jiménez-Fernández, 2017). Therefore, it has been inevitable that the computational thinking skill, which is considered to be so necessary and has a wide impact, has been included in international standards. In the list of standards that will guide 21st-century education published by the International Educational Technologies Association (ISTE), it is stated that computational thinking skill is one of the seven characteristics students are desired to have, and in the standards list published by the Computer Science Teachers Association (CSTA) in 2016, it is stated as a method that can be used in all other disciplines.

Studies show that ensuring that computational thinking skills are acquired by students in the education process will provide various benefits for students (Lee et al., 2011). One of these benefits is that it allows students to analyze the problems they encounter in daily life from different angles and develop solution strategies (Allan et al.," 2010). Within the framework of these contributions, one may notice that countries update their curriculum by adding content for computational thinking skills. In this context, England stands out as the first country to teach content for computational thinking skills at primary and secondary school levels since 2014 (Gülbahar & Kalelioğlu, 2018). Likewise, France started updating the curriculum in 2016, creating a structure for teaching computational thinking skills at all age levels (Özyol, 2019). In Norway, computational thinking subjects are taught in the updated curriculum as an elective course (Gülbahar & Kalelioğlu, 2018). In 2017, the Board of Education and Discipline in Turkey decided to gradually implement the Information Technologies and Software course enriched with contents for computational thinking skills in primary education institutions.

All of these make it clear that computational thinking skills can be acquired through different approaches and techniques (Weinberg, 2013). Among these approaches are "without using a computer", "block-based learning", "game or robot programming", and "interdisciplinary applications". In this context, it is stated that computational thinking skills can be acquired by using basic programming principles with the "game and robot programming" approach (Weintrop & Wilensky, 2015). Programming is the application and development process performed with various instruction sets to solve problems, provide human-computer interaction, and perform a specific task by computers (Prensky, 2008; Vee, 2013). Studies conducted within this scope show that teaching programming can affect the computational thinking skills of individuals by activating their metacognitive thinking skills (Çınar & Tüzün, 2017; Lye & Koh, 2014). Computational thinking skill consists of process stages such as "breaking down complex structures", "showing operations step by step", "observing repetitive structures", and "abstraction and debugging" (Barr, Harrison, & Conery, 2011; Israel, Pearson, Tapia, Wherfel & "Reese, 2015; Wing, 2008). With this aspect, it can be said that computational thinking skill is similar to the programming process (Israel et al., 2015). Considering that programming is also a problem-solving task, this situation seems to be reinforced (Fessakis, Gouli & Mavroudi, 2013).

There are studies examining the effect of teaching programming on computational thinking skills (Alsancak Sırakaya, 2019; Akçay, Atmatzidou, & Dimetriadis, 2017; Djambong & Freiman, 2016; Nouri, Zhang, Mannila, & Noren, 2019; Oluk & Korkmaz, 2016; Karahan and Türk, 2019). It is stated in these studies that programming and computational thinking skill is an issue that should be taken into consideration both in terms of the educational process and researchers, which suggests that the programming teaching process may have an effect on computational thinking skills. However, unlike previous studies, there is a growing in the importance of the idea that determining the effects of teaching programming on computational thinking skills through in-depth studies spread over larger periods can lead to more comprehensive results. As is known, longitudinal studies are long-term studies that are carried out in long periods. Therefore, this study aims to give an idea to the relevant educators and researchers by longitudinally examining the effect of teaching programming in higher education on the development of computational

thinking skills. In this context, the aim of the study is to examine the effect of teaching programming at the higher education level on computational thinking skills and its sub-dimensions. The problem of the study is has been set as "does teaching programming at the higher education level have an effect on computational thinking skills and its sub-dimensions?".

METHOD

The convergence of evidence that emerges with multiple methods increases the validity and strength of the findings (Creswell, 2014). For this reason, an explanatory design, one of the mixed methods in which quantitative and qualitative data are used together, was used in the present study. The reason why the descriptive design was preferred is to support quantitative data and strengthen research findings. For, as is known, qualitative data are collected and analyzed based on the analysis of quantitative data in descriptive research studies (Johnson, Onwuegbuzie, & Turner, 2007). In this context, the change of the computational thinking skill variable for a single group taken from a starting point in certain time intervals was examined with a quasi-experimental method. In educational research, it is often not possible to do real experimental work. The most important reason for this is that it is very difficult to distribute individuals to groups in school and classroom environments. Therefore, the experimental group from the pre-determined university and classroom was chosen impartially. This type of model is called quasi-experimental and is often used in educational research. One of the most important factors in ensuring internal validity in quasi-experimental studies is the impartial and random selection of the study group (Shadish, Cook & Campbell, 2002). Within the scope of the research, the working group was chosen impartially and randomly. In order to increase the external validity of the study, attention has been paid to the fact that the class in which the study is conducted is a traditional class, meaning that no material or technological special tool is used. However, validity in qualitative research is directly related to the consistency of data collection tools and analyzes (Whittemore, Chase & Mandle, 2001). Therefore, data that will go with and verify each other within the scope of the research have been collected and analyzed. To provide these, the relevant field experts were asked their opinions.

Participants

The participants of the study consist of students enrolled at the Department of Computer Educational and Instructional Technology of the Faculty of Education at a state university in Turkey. A total of 42 undergraduate students, 18 males and 24 females, enrolled during the 2016-2017 academic year, were selected on a voluntary basis. Convenience sampling method was preferred while determining the study group. Since all the participants in the study group are given programming courses, the control group could not be determined. Typical case sampling, one of the purposeful sampling methods, was used in the study. In this context, not all participants were invited to semi-structured focus group discussions. 8 randomly selected volunteer students were invited to semi-structured focus group interviews. However, the scale was applied to all participants.

Data Collection Tools

In this study, quantitative data were used in the "Computer Thinking Scale" developed by Korkmaz, Çakır, and Özden (2017) for university students. The scale is a five-point Likert type consisting of five factors including creativity, algorithmic thinking, collaboration, critical thinking, and problem-solving, and 29 items. The Cronbach Alpha reliability coefficient of the scale was calculated as 0.82. Cronbach's alpha values for the sub-dimensions of the scale vary between 0.72 and 0.86. The Cronbach Alpha reliability coefficient of the scale, which was applied to the participants three times within the scope of this study, ranges between 0.75 and 0.81. For the sub-dimensions of the scale, it changes between 0.70 and 0.83. The scale items used in this study were scored from the most positive (5) to the most negative (1) for each item level. Negative items were scored by reversing. The raw scores obtained in response to the responses of the participants to the five-point Likert-type scale were converted into standard scores between 20 and 100 in line with the suggestions of the researchers who developed the scale, and the scale level ranges were Low Level (20-51), Medium Level (52.67), and High Level (68- 100).

The qualitative data of the study were collected with a semi-structured focus group interview form (Appendix 1) developed by the researcher based on the "Computer Thinking Scale" sub-factors. The semi-structured focus group interview is a partially structured flexible group interview technique to reveal the knowledge and ideas of pre-selected participants within the framework of a specific topic (Çokluk, Yılmaz, & Oğuz, 2011). Five open-ended questions were included in the semi-structured focus group interview form to determine the participants' opinions on the factors in the scale. Open-ended questions were tested with a pilot application after the expert opinion was obtained. The order and dimensions of the questions were prepared in a way to allow them to be changed during the interview. The questions, which were corrected after the pilot interview, were applied in three sessions. After the pilot interview, the corrected questions were applied separately for each of the preliminary, intermediate, and final interviews.

Data Analysis

For the analysis of quantitative data, the researcher first controlled whether the parametric test assumptions were met or not. During the controls, it was observed that the sample size was greater than 30 (Büyüköztürk, 2007) and the kurtosis-skewness coefficients varied between ±2 (George & Mallery, 2010), but the Kolmogorov-Smirnov results ($p > 0.05$) of the intermediate test scores did not provide the assumption of normality. For this reason, nonparametric tests were used in the analysis of quantitative data. In the analysis of quantitative data, the Friedman test was used for the nonparametric response of the one-way ANOVA test. The purpose of Friedman analysis is to test whether the mean scores of three or more related sets of measures differ significantly from each other (Büyüköztürk, 2007). The Wilcoxon signed-rank test was used to determine which measurements differ significantly between

the Friedman test results. The descriptive analysis technique was used in the analysis of qualitative data. In the descriptive analysis technique, the data obtained from the interviews are interpreted and explained according to previously determined themes. Therefore, direct quotations were frequently included, allowing the participants to reflect their views.

Application Process

It took 3 years to complete the study with measurements made in 3 different time intervals. The first measurement data were collected at the beginning of the fall semester of the 2017-2018 academic year (2nd-grade), the second measurement data in the fall term of the 2018-2019 academic year (3rd -grade), and the third measurement data at the end of the spring semester of the 2019-2020 academic year (4th-grade). During the first measurement, the students had not taken any courses for teaching programming yet. During the second measurement, the students took "Programming Languages –I", "Programming Languages –II" courses, and continue to take the "Internet-Based Programming" course. During the third measurement, the courses students were supposed to take were completed. Each of the "Programming Languages –I", "Programming Languages –II" and "Internet-Based Programming" courses were given in different terms for 14 weeks, 5 hours a week, by face-to-face teaching method. C++ and C# programming languages were taught within the scope of these courses.

The researcher works as a lecturer in the department where the study was conducted. The researcher closely knows the participants in the study group, which allowed the participants' to reflect their real views and thoughts during the repeated measurements. "Computer Thinking Scale" and" semi-structured focus group interview form developed by the researcher were used at all stages of the study. The scale was used by the paper-and-pencil test, and the semi-structured focus group interview form developed by the researcher was applied in face-to-face interviews. The interviews were recorded with a voice recorder. The interviews were conducted in a conversational mood. The researcher returned to the questions left unanswered by the students and enabled them to focus on the relevant topic. Each interview lasted approximately 50 minutes. During the data collection process, permission was obtained from the ethics commission and the research was conducted by taking into account the publication ethics.

RESULTS

Descriptive statistics of pre-test, intermediate test and post-test scores of students' computational thinking skills are shown in Table 1.

Table 1: Descriptive Statistics of Pre-Test, Intermediate Test and Post-Test Scores of Students' Computational Thinking Skills

Measurements	f	x̄	ss
Pre-test	42	58,63	6,38
Intermediate test	42	76,02	7,81
Post-test	42	77,15	8,48

Table 1 reveals that students' computational thinking skill scores are at the medium level (52-67) in the pretest measurement stage (x Medium = 58.63), and at the high level (68-100) in the intermediate test and posttest measurement stages (x̄ = 76.02; x̄ = 77.15). It is observed that there is an arithmetical increase in students' computational thinking skills scores (x̄ = 58.63; x̄ = 76.02; x̄ = 77.15). In order to determine whether this increase was significant, the Friedman test analysis was performed for repeated measures. The Friedman test results are shown in Table 2.

Table 2: Friedman Test Results Regarding Pre-Test, Intermediate Test and Post-Test Scores of Students' Computational Thinking Skills

Measurements	x̄	Chi-Square (χ^2)	sd	P	Effect Size (r)
Pre-test	58,63	48,00	2	0,000	0,81
Intermediate test	76,02				
Post-test	77,15				

Table 2 reveals that there is a statistically significant difference between the pre-test, intermediate test and post-test scores of students' computational thinking skills as a result of Friedman's chi-square test (p = 0.000). The effect size of this difference was calculated as r = 0.81 (r = Z / √N). In order to determine the measurement stage at which this difference occurred, Wilcoxon signed-rank test analysis was performed (with Bonferroni adjusted alpha values). Therefore, firstly, the data obtained before teaching programming (Pre-test) and during the teaching programming process (Intermediate test) were analyzed. Then, the data obtained during the teaching programming process (Intermediate test) and the data after teaching programming (Post-test) were analyzed. This required the test to be applied twice and therefore the alpha level was calculated as 0.025 by dividing it into two (0.05 / 2 = 0.025) (Rosenthal, 1994). Wilcoxon signed-rank test results are shown in Table 3.

Table 3 reveals that there is a significant difference between the students' pre-test and intermediate test scores (p <0.025), and there is no significant difference between the students' intermediate test and post-test scores (p> 0.025), showing that there was a significant increase between the scores of the students before they started taking programming courses and their scores after they started taking programming courses. There was an increase between the average scores of students who continued to take programming courses and their average scores after taking programming courses, but this increase did not occur at a significant level.

The Friedman test analysis was applied for repeated measures to examine whether there is a significant difference between the pre-test, intermediate test and post-test scores of the sub-factors of the computational thinking skill scale. The analysis results are shown in Table 4.

Table 3: Wilcoxon Signed-Rank Test Results of Students' Pre-Test, Intermediate Test and Post-Test Scores

Measurements		n	Mean rank	Rank sum	z	p
Pre-test-Intermediate test	Negative rank	38	23.03	875	-5.29	0.000
	Positive rank	4	7	28		
	Equal	0				
Intermediate test- Post-test	Negative rank	13	12.46	162	-2.13	0.033
	Positive rank	7	6.86	48		
	Equal	22				
Pre-test- Post-test	Negative rank	38	23.08	877	-2.13	0.000
	Positive rank	4	6.50	26		
	Equal	0				

Table 4: Friedman Test Results Regarding Sub-Factors of Computational Thinking Skill

Factors	Measurements	Means	Chi-Square ($\chi 2$)	sd	P	Effect Size (r)
Creativity	Pre-test	56.96	34.31	2	0.000	0.69
	Intermediate test	76.96				
	Post-test	77.44				
Algorithmic Thinking	Pre-test	52.06	51.08	2	0.000	0.80
	Intermediate test	76.34				
	Post-test	78.17				
Problem-Solving	Pre-test	51.03	70.99	2	0.000	0.86
	Intermediate test	81.03				
	Post-test	81.98				
Critical thinking	Pre-test	68.28	2.16	2	0.339	0.14
	Intermediate test	70.28				
	Post-test	71.61				
Collaboration	Pre-test	71.19	5.00	2	0.082	0.13
	Intermediate test	73.33				
	Post-test	74.76				

Table 4 reveals that there is a significant difference between the pre-test, intermediate test and post-test scores of the creativity, algorithmic thinking and problem-solving sub-factors of the students' computational thinking skills ($r = 0.69$, $p < 0.025$; $r = 0.80$, $p < 0.025$; $r = 0.86$, $p < 0.025$). As a result of the Wilcoxon signed-rank test conducted to determine the direction of this difference, it was seen that the difference occurred between the pre-test and intermediate test scores. There was no significant difference between the pre-test, intermediate test and post-test scores of the Critical Thinking and Collaboration sub-factor ($r = 0.14$, $p > 0.025$; $r = 0.13$, $p > 0.025$). This situation shows that teaching programming causes a significant increase in the scores of creativity, algorithmic thinking, and problem-solving skills of the students, but it does not cause a significant increase in the scores of critical thinking and collaboration.

The qualitative data collected by asking the questions in the semi-structured focus group interview form shown in Appendix 1 to the participants are described within the framework of the creativity, algorithmic thinking, collaboration, critical thinking, and problem-solving factors of the computational thinking skill scale. The data obtained revealed that the participants stated a total of 130 opinions, 15 in the pre-test stage, 53 in the intermediate test stage, and 62 in the post-test stage. It was determined that the categories in which the participants expressed opinions most were problem-solving (36), algorithmic thinking (35), creativity (33), critical thinking (17), and collaboration (9). The codes obtained as a result of the analysis of the participants' opinions on the creativity category are shown in Table 5.

It was determined that the participants stated 5 opinions under 3 different codes in the pre-test stage, 13 opinions in the intermediate test stage under 5 different codes, and 15 opinions in the post-test stage under 5 different codes. In the pre-test interview on the creativity sub-factor, the participants stated that teaching programming had an effect on reasoning,

Table 5: Codes of the Participants' Opinions on the Creativity Category

Category (Factor)	Codes (Opinions)	Pre-test	Intermediate Test	Post-test
Creativity	To develop imagination		(S1,S6)	(S1,S4,S8)
	To develop new ideas		(S3,S4,S8)	(S2,S8)
	To reason	(S3,S6)	(S3,S7,S2)	(S1,S6,S3)
	To produce original products	(S5,S4)	(S1,S5)	(S1,S6,S7,S2)
	To try new methods	(S8)	(S1,S6,S8)	(S1,S4, S5)

Table 6: Codes of the Participants' Opinions on the Category of Algorithmic Thinking

Category (Factor)	Codes (Opinions)	Pre-test	Intermediate Test	Post-test
Algorithmic Thinking	To solve problems step by step	(S1,S2)	(S4,S1,S6,S7,S2)	(S7,S8,S3,S2)
	To make it easier to reach results		(S1,S6,S8)	(S4,S6,S2)
	To simplify the solution			(S3,S6)
	To facilitate complex operations	(S8)	(S5,S8,S3)	(S5,S6,S1)
	To divide the subject into small pieces	(S5,S6)	(S3,S6,S8,S2)	(S1,S6,S8)

developing original products and encouraging them to try new methods for solutions. Some of the opinions of the participants are as follows:

S6: While coding, you decide everything yourself, you write the plot in some way, which inevitably improves your power of reasoning...

S8: There is more than one way to solve a problem in programming, you constantly try new methods, after a while, you realize that you are doing the same thing in daily life.

In the interview test interview, the participants stated that, in addition to what they revealed in the pre-test, teaching programming had an effect on developing imagination and supporting the development of new ideas. In the post-test interview, they did not express a new opinion in addition to that in the intermediate test. Some opinions are as follows:

S7: Frankly, after learning programming, I realized that I am more confident that I can do more about some subjects…

S1: We constantly coded new and original programs in the programming languages lesson, so I can say that this has increased our imagination.

The codes obtained as a result of the analysis of the opinions of the participants regarding the algorithmic thinking category are shown in Table 6.

It was determined that the participants stated 5 opinions under 3 different codes in the pre-test stage, 15 opinions under 4 different codes in the intermediate test stage and a total of 15 opinions under 5 different codes in the post-test stage. In the pre-test interview regarding the algorithmic thinking category, the participants stated that reaching programming had an effect on the step-by-step solution of problems, facilitating complex operations and solving the problem by dividing it into small steps. Some of the opinions of the participants are as follows:

S1: … programs are usually written to find a solution to a problem, but to do this; you need to specify the problem step by step. Imagine that a programmer does it all the time, and of course s/he is likely to do it in real life over time.

S8: When we do things that we think are very difficult for us, we see that it is not that difficult actually. Building an algorithm is the same...

In the intermediate test interview, the participants stated that in addition to what they indicated in the pre-test, teaching programming had an effect on facilitating reaching the result. In the post-test interview, they expressed an opinion that it had an effect on simplifying the solution in addition to what they indicated in the intermediate test. Some of the opinions of the participants are as follows.

S3: I can say that I am making fewer mistakes now, if you ask why, it is because we encountered so many errors while writing code such as commas, dot types ... It is easier to reach the result after learning programming ...

S6: After learning how to code, I can solve the complex problems I encounter in normal life much more easily by simplifying the solution.

The codes obtained as a result of the analysis of the participants' opinions on the category of collaboration are shown in Table 7.

It was determined that the participants stated 1 opinion under 1 code during the pre-test stage, 3 opinions under 2 different codes in the intermediate test stage, and 5 opinions under 2 different codes in the post-test stage. In the pre-test interview regarding the collaboration category, the participants stated that teaching programming partially supported teamwork. In the mid-test interview, in addition to the opinion the participants expressed in the pre-test, they sometimes expressed an opinion that they made common cause, and they did not make any new opinion in the post-test interview. Some of the opinions of the participants are as follows:

Table 7: Codes of the Participants' Opinions on the Category of Collaboration

Category (Factor)	Codes (Opinions)	Pre-test	Intermediate Test	Post-test
Collaboration	To make common cause with someone		(S1,S2)	(S1,S2)
	To support teamwork	(S7)	(S7)	(S5,S8,S3)

Table 8: Codes of the Participants' Opinions on the Category of Critical Thinking

Category (Factor)	Codes (Opinions)	Pre-test	Intermediate Test	Post-test
Critical Thinking	To develop different perspectives			(S8,S2)
	To make logical inferences	(S1)	(S1,S7)	(S6,S2)
	To identify differences and similarities		(S5,S6)	(S1,S6)
	To notice inconsistencies		(S8,S4,S3)	(S4,S5,S3)

Table 9: Codes of the Participants' Opinions on the Category of Problem-Solving

Category (Factor)	Codes (Opinions)	Pre-test	Intermediate Test	Post-test
Problem-solving	To address the problems		(S1,S8,S3)	(S1, S5,S3, S4)
	To find the source of the problem	(S4)	(S5,S2,S6,S4)	(S3, S5,S2)
	To increase belief in solution		(S6,S2,S7,S4,S1)	(S1,S5,S8,S4,S7,S6)
	To be aware of your abilities	(S7,S2)	(S4,S2,S8)	(S1,S2,S8,S4,S5)

S7: I think that learning programming is a job that one can tackle on his/her own, we are constantly working with our friends and the professor, maybe it can contribute positively to group work with this aspect.

S2: Normally, I don't like to work with others, but we come together for the same purpose many times in programming language lessons...

The codes obtained as a result of the analysis of the participants' opinions on the critical thinking category are shown in Table 8.

It was determined that the participants stated 1 opinion under 1 code during the pre-test stage, 7 opinions under 3 different codes in the intermediate test stage, and 9 opinions under 4 different codes in the post-test stage. In the pre-test interview regarding the critical thinking category, the participants stated that teaching programming partially supported making logical inferences. In the intermediate test interview, they stated that it helped to identify differences and similarities and to notice inconsistencies in addition to the opinion they revealed in the pretest. In the post-test interview, the participants stated that it did not have much effect on developing different perspectives in addition to the opinions they expressed in the intermediate test. Some of the opinions of the participants are as follows:

S1: Programming has a logical system in itself. In order for programmers to code, they need to be able to make comparisons and sometimes establish a cause-and-effect relationship. But this does not mean that teaching programming improves this over time.

S5: Each code that will be written while coding should be consistent with the next one, as time passes, you sometimes start doing this in your daily life, but it does not allow you to notice all kinds of inconsistencies immediately ...

The codes obtained as a result of the analysis of the participants' opinions on the problem-solving category are shown in Table 9.

3 opinions were expressed under 2 different codes in the pre-test stage for the problem-solving category, 15 opinions were expressed under 4 different codes in the intermediate test stage, and 18 opinions were expressed under 4 different codes in the post-test stage. In the pre-test interview regarding the problem-solving category, the participants stated that teaching programming had an effect on finding the source of the problem and being aware of their own abilities. In the intermediate test interview, it was determined that in addition to the opinions they expressed in the pre-test, they expressed that it encouraged them to address the problems and increased the belief in solving the problems and that they did not express further opinion in the post-test interview. Some of the opinions of the participants are as follows:

S4: I can say that teaching programming has important benefits for me in finding the source of real-life problems because finding the source of the error is one of the most common problems faced when writing code.

S2: Being a programmer is a matter of talent. Three years passed, frankly, I did not know that I was this talented...

When the average scale scores and the number of opinions of the pre-test, intermediate test and post-test measurements regarding the effect of teaching programming on computational thinking skills are examined, it is seen that the findings obtained from the quantitative and qualitative data of the re-

search overlap with each other numerically. Both qualitative and quantitative data showed a similar increase in the pre-test, intermediate test, and post-test measurements.

DISCUSSION and CONCLUSION

Within the scope of the study, the effect of teaching programming at a higher education level on computational thinking skills and its sub-dimensions was examined. In this context, measurements were made in three different time intervals: pre-test (before teaching programming), intermediate test (during teaching programming) and post-test (after teaching programming). In the measurements made, it was concluded that teaching programming had an effect on computational thinking skills. This result of the study overlaps some research results "(Alsancak Sırakaya, 2019; Atmatzidou & Demetriadis, 2017; Djambong & Freiman, 2016; Korkmaz, Karaçaltı, & Çakır, 2018; Nouri, Zhang, Mannila & Noren, 2019; Oluk & Korkmaz, 2016; Pérez -Marín, Hijón-Neira, Bacelo, & Pizarro, 2018; Portelance & Bers, 2015; Witherspoon et al., 2017; Yinnan & Chaosheng, 2012). It also differs from some limited research results (Ataman-Uslu, Mumcu & Eğin, 2018; Aydoğdu, 2020; Lai & Yang, 2011). This difference may be related to the duration of the studies, the age level of the participants, and the types of activities performed. As a matter of fact, considering some of the studies in which the difference is observed, one may notice that the application period is mostly limited to 4-12 weeks and block-based programming activities are generally used. However, it is thought that this may be due to the fact that the average scores of the computational thinking skills of the participants in the studies in which the difference occurred was higher before the application. It is supported by the fact that there was no statistically significant difference between the intermediate test and post-test mean scores of the students in the present study.

It was concluded that there was a statistically significant difference between the pre-test and intermediate test scores of students' computational thinking skills. Students' computational thinking skill levels increased from medium to high after the teaching programming process. This shows that the teaching programming process has a positive effect on students' computational thinking skills. In this context, the fact that programming teaching processes were mostly preferred in studies conducted to develop computational thinking skills (Alsancak Sırakaya, 2019; De Araujo et al., 2016; Lockwood & Mooney, 2018; Yinnan & Chaosheng, 2012) supports this result. In the systematic literature review conducted by Tosik-Gün and Güyer (2019) on the evaluation of computational thinking skills, it is stated that programming components are mostly used in the evaluation of computational thinking skills. It was also observed that block-based programming was used in the study conducted by Portelance (2015), visual-based programming in the study by Alsancak-Sırakaya (2019), and computer-free coding activities in the study by Akçay, Karahan, and Türk (2019). In the study conducted by Alsancak-Sırakaya (2019), it was concluded that visual programming education improves students 'computational thinking skills, and in the study conducted by Korkmaz, Karaçaltı, and Çakır (2018), a positive correlation was found between students' programming course success and their computational thinking skill scores.

According to the results of the research, it was concluded that there was an increase in favor of the post-test between the students' intermediate test and post-test scores, but this increase was not statistically significant. There was no statistically significant change between the computational thinking skill scores of the students continuing the teaching programming process and their post-programming scores. This shows that the computational thinking skill that students acquired during the teaching programming process did not change after the teaching programming process was completed. In other words, no significant increase was observed in the computational thinking skill scores of the students, which increased with the teaching programming process, after the programming teaching process was completed.

The findings obtained from the quantitative data of the present study coincide with the findings obtained from the qualitative data. As a matter of fact, it was seen that the students stated 15 opinions under 10 different codes in the pre-test interview, 54 opinions under 19 different codes in the intermediate test interview, and 67 views under 23 different codes in the post-test interview. When the data obtained from the qualitative findings were examined, it was determined that the computational thinking skill of teaching programming had an effect on the sub-dimensions of algorithmic thinking, creativity, problem-solving, critical thinking and collaboration, respectively. When the quantitative and qualitative findings of the study were examined together, it was concluded that teaching programming had a significant effect on the algorithmic thinking, creativity and problem-solving sub-dimensions of the computational thinking skill, but had no significant effect on the sub-dimensions of collaboration and critical thinking.

Teaching programming is expressed as a long-term process that includes teaching algorithms (Erümit, Beyaz, Aksoy, Aksoy, & Şahin, 2017). Therefore, the development of algorithmic thinking skills after teaching programming, which includes the teaching algorithm process, is considered as an expected result. Studies show that computational thinking skill is an expression of creativity (ISTE, 2015), while the teaching programming process improves creativity (Yecan, Özçınar, & Tanyeri, 2017). From this point of view, it is pointed out that the opportunity to develop different perspectives depending on the abstract and complex structure of the teaching programming process (Durak, 2018; Lockwood & Mooney, 2017) may have had an effect on the creativity sub-dimension of computational thinking skill."

Programming is defined as compiling and processing codes for the solution of a problem in a computer environment (Arabacıoğlu, Bülbül, & Filiz, 2007). Therefore, it is an expected result that the teaching programming process, which is basically a problem-solving job (Prensky, 2008; Vee, 2013), will be effective in the development of problem-solving skills. In this respect, both the current study and other studies in the

literature (Chen et al.2017; Kafai & Burke, 2014; Fanchamps, Slangen, Hennissen, & Specht, 2019) show that teaching programming is effective on problem-solving skills. According to another result of the study, there was no significant increase in the scores of the students' collaboration and critical thinking sub-dimensions during and after the teaching programming process. While this result of the study is similar to some research results (Alsancak-Sırakaya, 2019; Ataman-Uslu, Mumcu & Eğin, 2018; Kaucic & Asic, 2011), it differs from others (Doğan & Kert, 2016; Pierce, 2011). Based on the different results obtained from the study, it can be said that the effect of teaching programming on collaboration and critical thinking may vary depending on variables such as different sample, application process, program carried out in the application process, the type of activities performed, and the role of the educator.

As a result, this study investigated whether teaching programming in higher education has an effect on computational thinking skills, concluding that it has a statistically significant effect. Although the findings of the study are limited to the scope of the research, it is seen that the subject is open to discussion in the literature. It can be stated that more experimental studies are needed on this subject.

Appendix.1 Semi-structured focus group interview questions

1. What are your thoughts on the effect of teaching programming on the development of some of our skills, thoughts, ideas or skills that we need in daily life?
2. What can you say about the effect of teaching programming on creative thinking skills?
3. What are your thoughts on the effect of teaching programming on problem-solving and our strategies for solving problems we encounter in daily life?
4. What can you say about the effects of teaching programming on the teamwork process?
5. What are your thoughts on the effect of teaching programming on our process of deciding what we need and how much we need in daily life?

REFERENCES

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

Akçay, A. O., Karahan, E., & Türk, S. (2019). Bilgi işlemsel düşünme becerileri odaklı okul sonrası kodlama sürecinde ilkokul öğrencilerinin deneyimlerinin incelenmesi. *Eskişehir Osmangazi Üniversitesi Türk Dünyası Uygulama ve Araştırma Merkezi Eğitim Dergisi*, *4*(2), 38-50. https://dergipark.org.tr/tr/pub/estudamegitim/issue/50016/626608

Alsancak Sırakaya, D. (2019). Programlama öğretiminin bilgi işlemsel düşünme becerisine etkisi. *Turkiye Sosyal Arastirmalar Dergisi*, *23*(2), 575-590. https://dergipark.org.tr/tr/pub/tsadergisi/issue/47639/448409

Ataman-Uslu, N., Mumcu, F., & Eğin, F. (2018). Görsel programlama etkinliklerinin ortaokul öğrencilerinin bilgi-işlemsel düşünme becerilerine etkisi. *Ege Eğitim Teknolojileri Dergisi*, *2*(1), 19-31. https://dergipark.org.tr/tr/pub/eetd/issue/38495/410699

Atmatzidou, S., Demetriadis, S., & Nika, P. (2018). How does the degree of guidance support students' metacognitive and problem solving skills in educational robotics. *Journal of Science Education and Technology*, *27*(1), 70-85. https://doi.org/10.1007/s10956-017-9709-x

Aydoğdu, Ş. (2020) Blok tabanlı programlama etkinliklerinin öğretmen adaylarının programlamaya ilişkin öz yeterlilik algılarına ve hesaplamalı düşünme becerilerine etkisi. *Eğitim Teknolojisi Kuram ve Uygulama*, *10*(1), 303-320. https://doi.org/10.17943/etku.649585

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, 20-23. https://eric.ed.gov/?id=EJ918910

Büyüköztürk, Ş. (2007). *Sosyal bilimler için veri analizi el kitabı*. Ankara: Pegem A Yayıncılık.

Çakır E., (2017). *Ters yüz sınıf uygulamalarının fen bilimleri 7. sınıf öğrencilerinin akademik başarı, zihinsel risk alma ve bilgisayarca düşünme becerileri üzerine etkisi* (Yayınlanmamış yüksek lisans tezi), Ondokuz Mayıs Üniversitesi, Samsun. https://tez.yok.gov.tr/UlusalTezMerkezi

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

Çokluk, Ö., Yılmaz, K., & Oğuz, E. (2011). Nitel bir görüşme yöntemi: Odak grup görüşmesi. *Kuramsal Eğitimbilim Dergisi, 4*(1), 95-107. https://dergipark.org.tr/en/pub/akukeg/issue/29342/313994

Creswell, J. W. (2014). *A concise introduction to mixed methods research*. SAGE publications.

De Araujo, A. L. S. O., Andrade, W. L., & Guerrero, D. D. S. (2016). A systematic mapping study on assessing computational thinking abilities. *Proceedings of Frontiers in Education Conference*, 1-9. https://ieeexplore.ieee.org/abstract/document/7757678

Djambong, T., & Freiman, V. (2016). Task-Based Assessment of Students' Computational Thinking Skills Developed through Visual Programming or Tangible Coding Environments. *International Association for Development of the Information Society*. https://eric.ed.gov/?id=ED571389

Doğan, U., & Kert, S. B. (2016). Bilgisayar oyunu geliştirme sürecinin, ortaokul öğrencilerinin eleştirel düşünme becerilerine ve algoritma başarılarına etkisi. *Boğaziçi Üniversitesi Eğitim Dergisi*, *33*(2), 21-42. https://dergipark.org.tr/en/pub/buje/issue/29693/319507

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, *4*(4), 355-369. https://doi.org/10.1007/s40692-017-0090-9

Durak, H. Y. (2018). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning*, 1-17. https://doi.org/10.1007/s10758-018-9391-y

Fanchamps, N. L., Slangen, L., Hennissen, P., & Specht, M. (2019). The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction. *International Journal of Technology and Design Education*, 1-20. https://doi.org/10.1007/s10798-019-09559-9

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*, *63*, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

George, D., & Mallery, P. (2010). SPSS for Windows step by step. A simple study guide and reference. *GEN, Boston, MA: Pearson Education, Inc*.

Gülbahar, Y., & Kalelioğlu, F. (2018). Bilişim teknolojileri ve bilgisayar bilimi: öğretim programı güncelleme süreci. *Millî Eğitim Dergisi, 47*(217), 5-23.

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

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education,* 82, 263-279. https://doi.org/10.1016/j.compedu.2014.11.022

ISTE (2015). ISTE-NETS Standards for learning, teaching, and leading in the digital age. International Society for Educational Technology. Retrieved January 12, 2019. https://www.iste.org/standards

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research*, *1*(2), 112-133. https://doi.org/10.1177/1558689806298224

Kafai, Y. B. & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming*. The MIT Press.

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4*(3), 583-596. http://hdl.handle.net/11727/3831

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

Korkmaz, Ö., Karaçaltı, C., & Çakır, R. (2018). Öğrencilerin programlama başarılarının bilgisayarca-eleştirel düşünme ile problem çözme becerileri çerçevesinde incelenmesi. *Amasya Üniversitesi Eğitim Fakültesi Dergisi, 7*(2), 343-370. https://dergipark.org.tr/en/pub/amauefd/issue/41157/413487

Lai, A. F., & Yang, S. M. (2011, September). The learning effect of visualized programming learning on 6 th graders' problem solving and logical reasoning abilities. In 2011 International Conference on Electrical and Control Engineering (pp. 6940-6944). IEEE. https://ieeexplore.ieee.org/abstract/document/6056908

Lockwood, J., & Mooney, A. (2017). A pılot study ınvestıgatıng the ıntroductıon of a computer-scıence course at second level focusıng on computatıonal thınkıng. *The Irish Journal of Education/Iris Eireannach an Oideachais*, *42*, 108-127. https://www.jstor.org/stable/26607242

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

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

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' scratch skills with their computational thinking skills in terms of different variables. *I.J. Modern Education and Computer Science*, *8*(11), 1-7. https://eric.ed.gov/?id=ED582994

Özçınar, H. (2017). Hesaplamali düşünme araştirmalarinin bibliyometrik analizi. *Eğitim Teknolojisi Kuram ve Uygulama*, *7*(2), 149-171. https://doi.org/10.17943/etku.288610

Özyol, B. (2019). *Bilgi-işlemsel düşünme becerisinin kazandırılmasına yönelik bir ortam tasarımı ve geliştirilmesi* (Master's thesis, Fen Bilimleri Enstitüsü). https://tez.yok.gov.tr/UlusalTezMerkezi

Partnership for 21st Century Skills (2009). Curriculum and instruction: A 21st century skills implementation guide. The Partnership for 21st Century Skill. http://www.p21.org/storage/documents

Pérez-Marín, D., Hijón-Neira, R., Bacelo, 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*, 105, 849. https://doi.org/10.1016/j.chb.2018.12.027

Pierce, T. S. (2011). *Introductory computer programming courses used as a catalyst to critical thinking development*. (Doctoral Dissertations). Ball State Unıversıty, Muncıe, Indıana. http://liblink.bsu.edu/catkey/1660953

Portelance, D. J., & Bers, M. U. (2015, June). Code and Tell: Assessing young children's learning of computational thinking using peer video interviews with ScratchJr. In *Proceedings of the 14th international conference on interaction design and children* (pp. 271-274). https://doi.org/10.1145/2771839.2771894

Prensky, M. (2008). Programming is the new literacy. *Edutopia magazine*. https://www.edutopia.org/print/5142

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, *72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Rosenthal, R., Cooper, H. ve Hedges, L. (1994). Etki büyüklüğünün parametrik ölçüleri. *Araştırma Sentezi El Kitabı*, *621* (2), 231-244.

Saavedra, A. R., & Opfer, V. D. (2012). Learning 21st-century skills requires 21st-century teaching. *Phi Delta Kappan*, *94*(2), 8-13. https://doi.org/10.1177/003172171209400203

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference.* Boston: Houghton Mifflin.

Talim ve Terbiye Kurulu Başkanlığı (2018*). Milli Eğitim Bakanlığı Bilişim Teknolojileri ve Yazılım Dersi Öğretim Programı: 5. ve 6. Sınıflar*. Ankara. https://ttkb.meb.gov.tr/

Tosik-Gün, E., & Güyer, T. (2019). Bilgi işlemsel düşünme becerisinin değerlendirilmesine ilişkin sistematik alanyazın taraması. *Ahmet Keleşoğlu Eğitim Fakültesi Dergisi*, *1*(2), 99-120. https://doi.org/10.38151/akef.597505

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64. http://d-scholarship.pitt.edu/id/eprint/21695

Whittemore, R., Chase, S. K., & Mandle, C. L. (2001). Validity in qualitative research. *Qualitative health research*, *11*(4), 522-537. https://doi.org/10.1177/104973201129119299

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

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

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 1-20. https://doi.org/10.1145/3104982

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

Yecan, E., Özçınar, H., & Tanyeri, T. (2017). Bilişim teknolojileri öğretmenlerinin görsel programlama öğretimi deneyimleri. *Elementary Education Online*, *16*(1), 377-393. http://dx.doi.org/10.17051/io.2017.80833

Yinnan, Z., & Chaosheng, L. (2012, July). Training for computational thinking capability on programming language teaching. In *2012 7th International Conference on Computer Science & Education (ICCSE)* (pp. 1804-1809). IEEE. http://dx.doi.org/10.1109/ICCSE.2012.6295420