International Journal of Computer Science Education in Schools, January 2020, Vol. 3, No. 3
ISSN 2513-8359

The Effect of Algorithm Education on Students' Computer Programming Self-Efficacy Perceptions and Computational Thinking Skills

Pınar Mıhcı Türker[1]
Ferhat Kadir Pala[1]

[1]Aksaray University, Faculty of Education

DOI: 10.21585/ijcses.v3i3.69

Abstract

In this study, the effect of algorithm education on pre-service teachers' computational thinking skills and computer programming self-efficacy perceptions were examined. In the study, one group pretest posttest experimental design was employed. The participants consisted of 24 (14 males and 10 females) pre-service teachers, majoring in Computer Education and Instructional Technology (CEIT). In order to determine the pre-service teachers' computer programming self-efficacy perceptions, the Computer Programming Self-Efficacy Scale was used, whereas Computational Thinking Skills Scale was used to determine their computational thinking skills. The Wilcoxon Signed-Rank Test was used to analyze the differences between pretest and posttest scores of students' computer programming self-efficacy perceptions and computational thinking skills. Throughout the practices, 10 different algorithmic problems were presented to the students each week, and they were asked to solve these problems using flow chart. For 13 weeks, 130 different algorithmic problems were solved. Algorithm education positively and significantly increased students' simple programming tasks, complex programming tasks and programming self-efficacy perceptions. On the other hand, algorithm education had a positive and significant effect only on students' algorithmic thinking sub-dimension but did not have any effect on other sub-dimensions and computational thinking skills in general.

Keywords: algorithm education, computational thinking, computer programming self-efficacy, pre-service teachers, computer education and instructional technology

1. Introduction

The process of creating software that brings technological equipment to life, in other words, the programming process, has gained importance today and has emerged as an important skill that every individual should have. Teaching processes have been shaped in this framework, and it has been ensured that programming education takes place as a course in schools starting from elementary school (Sayın, 2017; International Society for Technology in Education [ISTE], 2018). In Turkey, coding education has been included in Information Technologies and Software course starting from the 5th grade (11 years old) since 2012. Recently, it has come to the agenda to further reduce the age range of coding education and to include this content in all age groups from 1st grade to 4th grade (between 7-10 years old).

In short programming can be said as a series of codes written to make electronic devices perform certain functions in order to produce solutions to problems (Arabacıoğlu, Bülbül & Filiz, 2007; Blackwell, 2002; Ersoy, Madran & Gülbahar, 2011; Yükseltürk & Altıok, 2016). Programming skill is not only a skill required to produce software for computers. Students' high-level skills develop during the programming process. For example, Akçay and Çoklar (2016) describe these skills as critical thinking, algorithmic thinking, analytical thinking, problem solving, multidimensional thinking, creativity and questioning. There are also studies showing that programming skills have positive effects on computational thinking skills (Lye & Koh, 2014; Pala & Mıhcı Türker, 2019; Yıldız & Çiftçi, 2017). Computational thinking is a kind of analytical thinking, which includes elements such as problem solving, system design and understanding of human behavior based on the concepts of computer science (Wing, 2006). Korkmaz, Çakır & Özden (2017) gathered the computational thinking skills in five sub-dimension on the scale they developed. These skills are creativity, algorithmic thinking, collaboration, critical thinking and problem solving. In this study, these sub-dimension of computational thinking skills were used.

The programming process takes place in the development stage of the software development process. The programming process in this stage consists of four stages. Algorithm development is at the second stage of these processes (Çamoğlu, 2018). Akçay and Çoklar (2016) states that the algorithm as the process of developing a design that reveals how the computer should work and the process of finding a solution to a problem. According to Gökoğlu (2017), the operations performed to solve the problem during the programming process vary according to the programming languages but the logic does not change. To this end, students should be given the logic of programming before moving on to any programming language teaching. Algorithms form the basis of programming logic. In other words, without any programming language, students are taught the logic of programming with algorithm. With algorithm, it is possible for the individual to write the processing steps of the program in his or her own language, that is, to create the flowchart of the program with the so-called code. When the basic steps of algorithm development are examined, individuals should first identify the problem, define the inputs and outputs and determine possible solutions. Then, they should parse these solutions into steps and connect these steps to each other, that is, building an algorithm (Çamoğlu, 2018). In this process, it is possible that the students will benefit from some sub-dimensions of computational thinking skills such as algorithmic thinking, problem solving and creativity.

However, many studies indicated that programming is a difficult process and there are some problems in programming education (Arabacıoğlu, Bülbül and Filiz, 2007; Gomes and Mendes, 2007; Esteves and Mendes, 2004; Hongwarrittorn and Krairit, 2010; Ozoran, Çağıltay and Topalli, 2012; Robins, Rountree and Rountree, 2003; Saygıner and Tüzün, 2017). The complex structure of the programming language is listed as one of these problems (Gomes and Mendes, 2007; Kalelioğlu, 2015). Aşkar and Davenport (2009) state that the programming course at the university is perceived as quite difficult by the beginner-level students. Students' initial acceptance of programming as difficult can cause them to fail in this course. For example, Altun and Mazman (2012) emphasize that students may fail due to the fact that students accept programming as difficult and have low self-efficacy perceptions.

In the light of this information, it can be said that with the algorithm education, the student will get the logic of programming and focus on the solution process of the problems. In other words, without the complex language of programming, it is provided to comprehend the logic of programming and enter the programming course. On the other hand, algorithm education is one of the computer science unplugged activities. With computer science unplugged activities, it is aimed to teach computer science to students without computers.

Bell (2014) states that with computer science unplugged activities, students interact directly with content and not bothered with unnecessary details. In addition, computer science unplugged enables them to learn how computing principles work, providing students with the opportunity of direct observation and experimentation. Computer science unplugged activities include many subjects such as algorithms, human computer interaction, artificial intelligence, computer graphics, data compression and encryption (Bell, 2014).

Bell, Alexander, Freeman, and Grimley (2009) state that engaging in activities away from the computer might be effective for students because they might see the computer as a tool or toy. Students may think of problems encountered as a computer programmer by getting away from the computer. They can learn subjects such as algorithms, data compression, graphics algorithms, interface design, and models of computing without technical experience. In most cases, students find programming topics impressive. For example, in a study conducted by Mıhcı Türker and Pala (2018) with 5th and 6th graders, students stated that coding helped with developing games, moving characters, making movies, making robots and having fun. However, although the students have positive views on programming, the difficulty of learning programming should be explained appropriately. Therefore, computer science unplugged provides students with the opportunity to interact with the topic they find impressive without getting into complex programming topics, provides more fun and lasting learning opportunities in different contexts such as game-based learning and learning by discovery and help them attain computational thinking skills (Kalelioglu, 2017). Kalelioğlu (2015) addressed the difficulties of programming languages during the programming process and stated that thinking skills should be supported with different methods for programming. In this context, computer science unplugged is important for developing these skills.

In line with these views, it is possible that students will learn the logic of programming and develop their programming self-efficacy through algorithm education by moving away from the complex structure of programming languages. On the other hand, students are expected to produce a solution to a problem without programming language during this process and to use high-level thinking skills in this way. Therefore, in this study, the effect of algorithm education on pre-service teachers' computational thinking skills and computer programming self-efficacy perceptions was examined and answers to the following questions were investigated:

1) Is there a significant difference between the pretest and posttest scores of students' computer programming self-efficacy perceptions?

2) Is there a significant difference between the pretest and posttest scores of students' computational thinking skills?

1.2 Related Literature

In this study, the effect of algorithm education on pre-service teachers' computer programming self-efficacy perceptions and computational thinking skills were examined. For example, in a study conducted by Tsai (2019), visual programming language was used to teach basic programming concepts to university students. Accordingly, students' learning performances and computer programming self-efficacy regarding perceptions were examined. After the implementation, students' performances in basic programming concepts and programming self-efficacy increased positively.

Özmen and Altun (2014) conducted a study on students' perceptions of self-efficacy regarding programming. In this study, the reasons of failure of teacher candidates in programming course were examined and 12 students were interviewed for this purpose. The interview results revealed that students had difficulties in processes such as programming knowledge, programming skills, understanding the logic of the program and debugging. Students stated that the main reasons for their failure in programming were lack of practice and knowledge and not being able to develop algorithms. On the other hand, it is seen that students with high programming experience have high programming achievement and high self-efficacy perceptions.

In another study conducted by Mazman and Altun (2013), self-efficacy perceptions and pre-experience characteristics of Computer Education and Instructional Technology Education (CEIT) students were examined. In this context, programming courses were given to the students who had taken programming courses before, scales were applied before and after the course and the obtained data were analyzed. Accordingly, the programming course significantly increased computer programming self-efficacy perceptions in both pre-experienced and experienced groups, and this increase was found to be higher in the group without prior experience. In addition, the difference between self-efficacy perceptions between pre-experienced and experienced groups decreased at the end of the programming course. The researchers stated that students' self-efficacy perceptions and their previous knowledge played an important role in their achievement in this course.

In a study conducted by Davidson, Larzon and Ljunggren, (2010), the effect of Introduction to Programming course on the change in self-efficacy perceptions of students was examined. Accordingly, individuals' self-efficacy scores did not show a significant change at the beginning and at the end of the programming course. However, there was an increase in students' sub-skills such as analysis and solution of simple problems, debugging and working principle of computer.

In the light of the findings obtained in the studies, it is concluded that the students' perceptions of self-efficacy increased with the increase of their experience in programming and consequently their success in programming increased. Based on these data, it is possible to obtain programming logic and to increase self-efficacy perceptions positively without facing problems related to language or abstract structure in programming.

Pala and Mıhcı Türker (2019) examined the effect of programming education on the computational thinking skills of pre-service teachers. In this context, the researchers found that robotic-based programming significantly affected students' creativity, algorithmic thinking and critical thinking dimensions and that text-based programming had no effect. In a different study, Oluk and Korkmaz (2016) determined a positive relationship between programming skills and computational thinking skills. As students' programming skills increased, their computational thinking skills also increased. Similarly, as a result of implementations done using programming, Fadjo (2012) determined that implementations play an important role in the development of students' computational thinking skills and concept knowledge.

In addition, there are studies showing the effectiveness of programming education on sub-skills comprising the computational thinking skills such as problem solving (Kalelioğlu & Gülbahar, 2014; Kukul & Gökçearslan, 2014; Somyürek, 2014; Robinson, 2005), creativity (Kobsiripat, 2015).

2. Method

Examining the effect of algorithm education on pre-service teachers' computer programming self-efficacy perceptions and computational thinking skills, the present study employed one group pretest posttest experimental design, one of the quasi-experimental research designs. In this design, the significance of the difference between the pretest and posttest values of the groups is tested (Cohen, Manion & Morrison, 2002). In the study, unplugged implementations in the algorithm course made up the independent variable, whereas students' computer programming self-efficacy perception and computational thinking skill made up the dependent variable.

2.1 Study Group

The study was conducted within the scope of Algorithm course during the Spring semester of 2018-2019 academic year. The participants consisted of 24 (14 males and 10 females) pre-service teachers studying at the CEIT department of a university in Central Anatolia Region, Turkey. The age range of the participants ranged from 19 to 26, and all of them had programming knowledge.

2.2 Data Collection Tool

In order to determine the pre-service teachers' computer programming self-efficacy perceptions, the Computer Programming Self-Efficacy Scale, adapted to Turkish by Altun and Mazman (2012), was used. Computational Thinking Skills Scale (CTSS), developed by Korkmaz, Çakır and Özden (2017) for undergraduate students, was used to determine pre-service teachers' computational thinking skills.

The original computer programming self-efficacy scale consists of 32 items and four factors. However, after the adaptation and analysis by Altun and Mazman (2012), the Turkish form consists of nine items and two factors, simple programming tasks and complex programming tasks. Internal consistency for the scale was .90 for the first factor, .94 for the second factor, and .92 for the overall scale, and these nine items explained 80.81% of the total variance.

Computational Thinking Skills Scale (CTSS), developed by Korkmaz, Çakır and Özden (2017) for undergraduate students, was used to determine pre-service teachers' computational thinking skills. The scale has 29 items and five dimensions, creativity (eight items), algorithmic thinking (six items), collaboration (four items), critical thinking (five items) and problem solving (six items). The scale is a five-point Likert one. Internal consistency coefficient Cronbach's alpha values were .84 for the creativity dimension, .86 for the algorithmic thinking dimension, .86 for the collaboration dimension; .78 for the critical thinking dimension, .72 for the problem solving dimension and .82 for the overall scale. The explained variance for all factors was 56.1%.

2.3 Data Analysis

Russell and Purcell (2009) stated that parametric tests should not be used with groups less than 30, and that when the size of the group is smaller (n<30), the data do not meet normality. Gosling (1995) stated that when the distribution of the universe is unknown and when the group number is small (n<30), the normality cannot be adequately met, and suggested the use of non-parametric tests. Ploger and Yasukawa (2003) suggested that parametric techniques should be used when the groups are large (n> 30) and the normality is met. Since the number of groups included in the study was less than 30, non-parametric tests were used to compare between pretest and posttest scores of students. In this respect, the Wilcoxon Signed-Rank Test was used to analyze the differences between pretest and posttest scores of students' computer programming self-efficacy perceptions and computational thinking skills.

2.4 Implementation Process

The implementation process of the study was conducted with 24 pre-service teachers who were studying in the CEIT department of a university in Central Anatolia during the 2018-2019 academic year. First, scales were administered to the pre-service teachers, and then the process was continued by presenting various algorithm problems for 13 weeks. Each week, 10 different algorithmic problems were presented to the students, and they were asked to solve these problems using flowchart. A total of 130 different algorithmic problems were solved.

The problems were prepared by researchers using three different algorithm books (Çamoğlu, 2018; Tungut, 2016; Vatansever, 2015). The problems were approved by two experts. According to this, questions about sequential operations, conditional states and repetitive structures are included in the algorithm problems. The students exchanged information with their groupmates during the solution process and discussed the solution with them. At the end of the process, the scales were applied to the participants again, and the data obtained were organized for analysis.

3. Results

3.1 Is there a significant difference between the pretest and posttest scores of students' computer programming self-efficacy perceptions after the algorithm education?

Within the scope of the study, in order to determine pre-service teachers' computer programming self-efficacy perceptions, Computer Programming Self-Efficacy Scale was administered. The scale was comprised of two dimensions and nine items. Descriptive analyzes were conducted in accordance with the responses given by the pre-service teachers to the items, and the results are presented in Table 1.

Table 1. Descriptive statistics regarding pre-service teachers' programming self-efficacy perceptions

	Dimension	Item Number	*N*	Lowest	Highest	$\bar{X}$	*ss*
Before Implementation	Simple programming tasks	3	24	5	21	14.41	5.44
	Complex programming tasks	6	24	6	36	20.29	9.10
Total	Programming self-efficacy perceptions	9	24	11	57	34.70	13.76
After Implementation	Simple programming tasks	3	24	12	21	17.79	3.07
	Complex programming tasks	6	24	7	40	24.79	7.56
Total	Programming self-efficacy perceptions	9	24	21	61	42.58	9.99

According to Table 1, while pre-service teachers' mean scores of simple programming tasks were $\bar{X}$=14.41 before the implementation and $\bar{X}$=17.79 after the implementation, pre-service teachers' mean scores of complex programming tasks were $\bar{X}$=20.29 before the implementation and $\bar{X}$=24.79 after the implementation. In addition, pre-service teachers' computer programming self-efficacy perceptions were 11 at the lowest before the implementation and 21 after the implementation, whereas pre-service teachers' computer programming self-efficacy perceptions were 57 at the highest 57 before the implementation and 61 at the highest after the implementation.

Table 2. Pre-service teachers' programming self-efficacy perception pretest posttest score analysis

Dimension	Test	Ranks	N	Mean	Total	*Z*	*p*
Simple Programming Tasks	Post	Negative Ranks	4				
	Pre	Positive Ranks	16	4.25	17	-3.30	.001*
		Equal	4	12.06	193		
		Total	24				
Complex Programming Tasks	Post	Negative Ranks	8				
	Pre	Positive Ranks	16	8.13	65	-2.43	.015*
		Equal	0	14.69	235		
		Total	24				
Programming Self-Efficacy Perception	Post	Negative Ranks	6				
	Pre	Positive Ranks	17	6.42	38.5	-3.03	.002*
		Equal	1	13.97	237.5		
		Total	24				

* $p < .05$

Table 2 presents the results of Wilcoxon Signed-Rank Test done to determine whether pre-service teachers' computer programming self-efficacy pretest and posttest scores differ significantly. Accordingly, there is a significant difference in pre-service teachers' pretest and posttest scores in all dimensions. However, since positive ranks were taken as baseline and the z value was negative, the difference was in favor of the posttest (p<.05; z_{simple}: -3.30; $z_{complex}$:-2.43; z_{total}: -3.03). In other words, algorithm education significantly increases pre-service teachers' computer programming self-efficacy perceptions.

3.2 Is there a significant difference between the pretest and posttest scores of students' computational thinking skills after the algorithm education?

Within the scope of the study, in order to determine pre-service teachers' computational thinking skills, CTSS was administered. The scale was comprised of five dimensions and 29 items. Descriptive analyzes were conducted in accordance with the responses given by the pre-service teachers to the items, and the results are presented in Table 3.

Table 3. Descriptive statistics of students' computational thinking skills

	Dimension	Item Number	N	Lowest	Highest	$\bar{X}$	ss
Before Implementation	Creativity	8	23	24	40	34.21	3.84
	Algorithmic Thinking	6	23	6	29	17.26	6.94
	Collaboration	4	23	6	30	13.82	5.95
	Critical Thinking	5	23	4	20	14.69	3.92
	Problem Solving	6	23	12	25	18.00	3.60
Total	Computational Thinking	29	23	82	142	98.00	14.31
After Implementation	Creativity	8	23	26	40	34.86	3.55
	Algorithmic Thinking	6	23	6	30	20.34	7.13
	Collaboration	4	23	8	27	12.56	5.22
	Critical Thinking	5	23	13	20	14.17	3.66
	Problem Solving	6	23	6	25	19.26	4.01
Total	Computational Thinking	29	23	79	125	101.21	12.92

Table 3 shows that one pre-service teacher's scale results were considered invalid, and the analysis was continued with 23 pre-service teachers. According to this, pre-service teachers' mean scores of computational thinking skills were $\bar{X}$=98.00 before the implementation and $\bar{X}$=101.21 after the implementation. In addition, pre-service teachers' computational thinking skills were 82 at the lowest before the implementation and 79 after the implementation, whereas pre-service teachers' computational thinking skills were 142 at the highest 57 before the implementation and 125 at the highest after the implementation.

Table 4. Pre-service teachers' computational thinking skills pretest posttest score analysis

Dimension	Test	Ranks	N	Mean	Total	*Z*	*P*
Creativity	Post	Negative Ranks	6		59.00		
	Pre	Positive Ranks	11	9.83	94.00	-.831	.406
		Equal	6	8.55			
		Total	23				
Algorithmic	Post	Negative Ranks	3				
	Pre	Positive Ranks	17	11.50	34.50	-2.64	.008*
		Equal	3	10.32	175.50		
		Total	23				
Collaboration	Post	Negative Ranks	14	9.43	132.00	-1.49	.135

	Pre	Positive Ranks	5	11.60	58.00		
		Equal	4				
		Total	23				
Critical Thinking	Post	Negative Ranks	9				
	Pre	Positive Ranks	10	11.50	103.50	-.343	.732
		Equal	4	8.65	86.50		
		Total	23				
Problem Solving	Post	Negative Ranks	4				
	Pre	Positive Ranks	15	12.75	51.00	-1.77	.075
		Equal	4	9.27	139.00		
		Total	23				
Total	Post	Negative Ranks	8				
	Pre	Positive Ranks	14	9.88	79.00	-1.54	.123
		Equal	1	12.43	174.00		
		Total	23				

* *p < .05*

Table 4 presents the results of Wilcoxon Signed-Rank Test done to determine whether pre-service teachers' computational thinking skills pretest and posttest scores differ significantly. Accordingly, there is a significant difference in pre-service teachers' algorithmic thinking pretest and posttest scores. Since positive ranks were taken as baseline and the z value was negative, the difference was in favor of the posttest (*p*<.05; $z_{algorithmic}$: -2,64). However, there was no significant difference in pre-service teachers' computational thinking skills and sub-dimension comprising this skill (*p*>.05).

4. Discussion

In this study, the effect of algorithm education on pre-service teachers' computational thinking skills and self-efficacy perceptions related computer programming were investigated. In this context, a 13-week implementation process was carried out with 24 pre-service teachers (14 males, 10 females). Each week, 10 different algorithmic problems were presented to the students and they were asked to solve these problems using flow chart. A total of 130 different algorithmic problems were solved. The students exchanged information with their groupmates during the solution process and discussed the solution with them.

In this study, the effect of direct programming logic on pre-service teachers' computational thinking skills and computer programming self-efficacy perceptions were examined without any programming language because many programming languages cause negative attitudes in students with their abstract and complex structure (Gomes & Mendes, 2007; Kalelioğlu, 2015).

In this direction, the study results show that algorithm education positively and significantly increases students' perceptions of simple programming tasks, complex programming tasks and programming self-efficacy. Similarly, related literature put forth that students' computer programming self-efficacy perceptions are positively affected as a result of adequate understanding of programming logic by students (Jegede, 2009; Mazman & Altun, 2013; Özmen & Altun, 2014; Tsai, 2019). On the other hand, it is possible to come across studies that do not show a significant change in students' self-efficacy after programming education (Davidson, Larzon & Ljunggren, 2010).

Bell, Alexander, Freeman & Grimley, (2009) stated that students do not deal with the difficulty of programming and mobilize their computational thinking skills through computer science unplugged. Similarly, Kalelioğlu (2015) pointed out the importance of computer science unplugged in the development of high-level thinking skills. Within the scope of the present study, algorithm education was offered using computer science unplugged. At the end of the implementation process, it was found that the algorithm education positively and significantly affected the pre-service teachers' algorithmic thinking skills. However, it had no effect on other sub-skills and computational thinking skill in general. Significant differences in algorithmic thinking are expected. On the other hand, it is surprising that there was no significant difference in other sub-dimensions and overall score. It was expected that an increase in algorithmic thinking skill would affect creativity, collaboration, critical thinking, problem solving and computational thinking skill in general. Yet, no significant difference was found because during the algorithm education, students were allowed to work together in a collaborative environment, and they were able to find solutions to problems together. During the process, students discussed the solutions with a critical eye, and they created new solutions. This is believed to be due to the abstract progress of the implementations. It is thought that more clear data can be obtained by comparing the implementations in which the students can be more active in different researches and the implementations in which the process of solving algorithmic problems can be carried out passively.

The studies examining the relationship between programming education and computational thinking skills show that robotic programming education (Grover & Pea, 2013; Lee et al., 2011; Pala & Mıhcı Türker, 2019; Penmetcha, 2012; Repenning, Webb and Ioannidou, 2010) and visual-based programming education (Fadjo, 2012; Howland & Good, 2015) are effective on students' computational thinking skills. In addition, both robotic and visual-based programs (Khanlari, 2013; Kobsiripat, 2015) have positive effects on algorithmic thinking sub-dimension (Penmetcha, 2012), critical thinking skill sub-dimension (Blanchard, Freiman & Lirrete-Pitre, 2010; Chambers, Carbonaro, Rex & Grove, 2007), collaboration sub-dimension (Khanlari, 2013; Miller & Nourbakhsh; 2016) and problem solving skill dimension (Atmatzidou & Demetriadis, 2012; Khanlari, 2013; Petre & Price, 2004; Robinson, 2005; Rogers & Portsmore, 2004). This is believed to be due to the fact that these platforms have a more concrete structure and are easier to understand programming logic. In future studies, it is possible to compare all three methods of implementation by using different groups. In addition, this study was conducted with a limited sample because of the size of class. However, it is assumed that other latent variables such as projects in other courses taken during the study period do not affect dependent variables. Similar implementations can be carried out with larger experiment groups and control groups.

In addition, in different studies, algorithm development through digital applications can be investigated. Thus, the effect of digital applications on pre-service teachers' self-efficacy and computational thinking skills can be examined. Similar studies can be reported in different age groups. Then, the success of these students towards programming can be examined.

5. Conclusion

It is possible to increase the self-efficacy of programming positively by learning the logic of programming by moving away from the complex structure of programming languages. In this respect, algorithms that are included in computer science unplugged activities and aim to give students the logic of programming without complex language structure gain importance. The student is required to identify the problem, to define the inputs and the outputs, and to determine the possible solutions during the algorithm creation stage. The next step is to parse these solutions into steps and associate these steps with each other. In this process, it is possible to utilize and develop various high-level thinking skills. Computational thinking is among these high-level thinking skills. In the light of this process, the effect of algorithm education, which is one of the computer science unplugged activities, on students' self-efficacy perception and computational thinking skills was examined in this study.

The results show that; algorithm education increases students' self-efficacy perceptions related programming in a positive and meaningful way. In this way, after the algorithm education, self-efficacy related programming can be increased and students can start programming with more positive perspectives. On the other hand, there was no significant difference in the students ' computational thinking skills. However, only the algorithmic thinking, a sub-dimension of computational thinking, increased significantly and positively.

References

Akçay, A. & Çoklar, A. N. (2016). Bilişsel becerilerin gelişimine yönelik bir öneri: Programlama eğitimi. In A. İşman, H. F. Odabaşı & B. Akkoyunlu (Eds.), *Eğitim Teknolojieri Okumaları 2016* (pp. 121-139). Ankara, Turkey: The Turkish Online Journal of Educational Technology (TOJET).

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin öz yeterlilik algısı ölçeğinin Türkçe formunun güvenirlik ve geçerlik çalışması, *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi*, *3*(2), 297-308.

Arabacıoğlu, C., Bülbül, H. & Filiz, A. (2007, January). *A new approach to computer programming teaching*. Presented at the 2007 Academic Informatics Conference, Dumlupinar University, Kütahya, Turkey. Retrieved from https://ab.org.tr/ab07/kitap/arabacioglu_bulbul_AB07.pdf.

Aşkar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming among engineering students. *Online Submission*, *8*(1), 26-32.

Atmatzidou, S., & Demetriadis, S. N. (2012, July). Evaluating the role of collaboration scripts as group guiding tools in activities of educational robotics: Conclusions from three case studies. In *2012 IEEE 12th International Conference on Advanced Learning Technologies* (pp. 298-302). IEEE.

Bell, T. (2014). Ubiquity Symposium: The science in computer science: unplugging computer science to find the science. *Ubiquity,* March*,* 1-6. http://dx.doi.org/10.1145/2590528.2590531.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13(1), 20-29.

Blackwell, A. F. (2002, June). What is programming? Proceedings of the *14th workshop of the Psychology of Programming Interest Group,* Brunel University, Middlesex, UK. pp. 204-218.

Blanchard, S., Freiman, V., & Lirrete-Pitre, N. (2010). Strategies used by elementary schoolchildren solving robotics-based complex tasks: Innovative potential of technology. *Procedia-Social and Behavioral Sciences*, *2*(2), 2851-2857.

Chambers, J. M., Carbonaro, M., Rex, M., & Grove, S. (2007). Scaffolding knowledge construction through robotic technology: A middle school case study. *Electronic Journal for the Integration of Technology in Education*, 6, 55-70.

Cohen, L., Manion, L., & Morrison, K. (2002). *Research methods in education (6th ed.).* Abingdon, UK: Routledge

Çamoğlu, K. (2018). *Algorithm (6th ed.).* Kodlab: İstanbul, Turkey.

Davidson, K., Larzon, L. & Ljunggren, K. (2010). *Self-Efficacy in programming among STS students.* Retrieved from http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports.

Ersoy, H., Madran, R. O., & Gülbahar, Y. (2011, February). *A model proposed for teaching programming languages: Robotic programming*. Presented at the 2011 Academic Informatics Conference, İnönü University, Malatya, Turkey. Retrieved from https://ab.org.tr/ab11/kitap/ersoy_madran_AB11.pdf.

Esteves, M., & Mendes, A. J. (2004, October). A simulation tool to help learning of object oriented programming basics. In *34th Annual Frontiers in Education, 2004. FIE 2004*. (pp. F4C-7). IEEE.

Fadjo, C. L. (2012). *Developing computational thinking through grounded embodied cognition* (Unpublished dissertation). Columbia University, New York, NY, USA.

Gomes, A. & Mendes, A. J. (2007, September). Learning to program - difficulties and solutions. In *Proceedings of the International Conference on Engineering Education*. Coimbra, Portugal. Retrieved from http://icee2007.dei.uc.pt/proceedings/papers/411.pdf.

Gosling, J. (1995). *Introductory statistics*. Leichhardt, Australia: Pascal Press.

Gökoğlu, S. (2017). Algorithm perception in programming education: A metaphor analysis. *Cumhuriyet International Journal of Education,* 6(1), 1-14.

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

Hongwarrittorn, N., & Krairit, D. (2010, July). *Effects of program visualization (jeliot3) on students' performance and attitudes towards java programming.* Presented at the 8th International Conference on Computing, Communication and Control Technologies, Delhi, India.

Howland, K., & Good, J. (2015). Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education*, *80* (2015), 224-240.

International Society for Technology in Education- ISTE (2018). *Computational thinking for all.* Retrieved from https:// www.iste.org/explore/articleDetail?articleid=152.

Jegede, P. O. (2009). Predictors of java programming self-efficacy among engineering students in a Nigerian University. *International Journal of Computer Science and Information Security,4*(1&2). Retrieved from https://arxiv.org/ftp/arxiv/papers/0909/0909.0074.pdf.

Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education,* 13(1), 33-50.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, 52, 200-210.

Kalelioğlu, F. (2017). Bilgisayarsız bilgisayar bilimi (B3) öğretimi. Gülbahar, Y. (Ed.) *Bilgi İşlemsel Düşünmeden Programlamaya.* (I. Baskı) (ss. 183-206) Ankara, Turkey: Pegem Akademi.

Khanlari, A. (2013). Effects of robotics on 21st century skills. *European Scientific Journal (ESJ)*, 9 (27). 26-36.

Kobsiripat, W., (2015). Effects of the media to promote the Scratch programming capabilities creativity of elementary school students. *Procedia-Social and Behavioral Sciences,* 174 (2015), 227-232.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the Computational Thinking Scales (CTS). *Computers in Human Behavior*, 72, 558-569.

Kukul, V., & Gökçearslan, Ş. (2014, September). *Investigatıng the problem solving skills of students attended scratch programming course*. Presented at the 8th International Computer & Instructional Technologies Symposium, Trakya University, Edirne, Turkey.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32-37.

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.

Mazman, S. G., & Altun, A. (2013). The effect of introductory to programming course on programming self-efficacy of CEIT students. *Journal of Instructional Technologies and Teacher Education,* 2 (3), 24-29.

Mıhcı Türker, P. & Pala, F. K. (2018). Opinions of Secondary School Students, Teachers and Parents About Coding. *Elementary Education Online*, 17(4), 2013-2029.

Miller, D. P., & Nourbakhsh, I. (2016). Robotics for education. In *Springer handbook of robotics* (pp. 2115-2134). Springer, Cham.

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,* (11)*,* 1-7.

Ozoran, D., Çağıltay, N. E., & Topallı, D. (2012, November). Using scratch in introduction to programming course for engineering students. In *2nd International Engineering Education Conference (IEEC2012)* (Vol. 2, pp. 125-132).

Özmen, B., & Altun, A. (2014). Undergraduate students' experiences in programming: difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 1-27.

Pala, F. K., & Mıhcı Türker, P. (2019). The effects of different programming trainings on the computational thinking skills. *Interactive Learning Environments*, 1-11. https://doi.org/10.1080/10494820.2019.1635495.

Penmetcha, M. R. (2012). *Exploring the effectiveness of robotics as a vehicle for computational thinking* (Doctoral dissertation), Purdue University, West Lafayette, IN, USA.

Petre, M., & Price, B. (2004). Using robotics to motivate 'back door'learning. *Education and Information Technologies*, 9 (2), 147-158.

Ploger, B. J., & Yasukawa, K. (2003). Introduction to statistics. In *Exploring Animal Behavior in Laboratory and Field* (pp. 415-446). San Diego, CA, USA: Academic Press.

Repenning, A., Webb, D., & Ioannidou, A. (2010, March). *Scalable game design and the development of a checklist for getting computational thinking into public schools.* Presented at the 41st ACM technical symposium on Computer science education, Milwaukee, Wisconsin, USA

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, 13(2), 137-172.

Robinson, M. (2005). Robotics-driven activities: Can they improve middle school science learning? *Bulletin of Science, Technology & Society*, 25 (1), 73-84.

Rogers, C., & Portsmore, M. (2004). Bringing engineering to elementary school. *Journal of STEM Education. Innovations & Research*, 5 (3), 17-28.

Russell, B., & Purcell, J. (2009). *Online research essentials: designing and implementing research studies* (Vol. 19). Hoboken, New Jersey, USA: John Wiley & Sons.

Saygıner, Ş. & Tüzün, H. (2017, December). *Difficulties in programming education and solutions.* In Proceedings of the International Computer and Instructional Technology Symposium, Malatya, Turkey. (pp 72-84).

Sayın, Z. (2017). Bilgisayar bilimi eğitimi kapsamı. Gülbahar, Y. (Ed.) *Bilgi İşlemsel Düşünmeden Programlamaya*. (I. Baskı) (ss. 133-154) Pegem Akademi, Ankara, Turkey.

Somyürek, S. (2015). An effective educational tool: construction kits for fun and meaningful learning. *International Journal of Technology and Design Education*, *25*(1), 25-41.

Tsai, C. Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*, 95, 224-232.

Tungut, H. B. (2016). *Algorithm and Logic of Programming (6th ed.).* Kodlab: İstanbul, Turkey.

Vatansever, F. (2015). *Algorithm development and introduction to programming.* Seçkin: Ankara, Turkey.

Yıldız, M. & Çiftçi, E. (2017). Bilişimsel Düşünme ve Programlama. In H. F. Odabaşı, B. Akkoyunlu ve A. İşman (Eds.). *Eğitim Teknolojileri Okumaları 2017*, (Chapter 5, pp. 75-86). The Turkish Online Journal of Educational Technology (TOJET) and Sakarya University, Adapazarı, Turkey.

Yükseltürk, E. & Altıok, S. (2016). Pre-Service information technology teachers` perceptions about using Scratch tool in teaching programming. *Mersin University Journal of the Faculty of Education, 12*(1), 39-52.