Journal of Teacher Education and Lifelong Learning (TELL)

Cilt: 3 Sayı:2 Yıl: 2021 | Research Article | ISSN: 2687-5713

The Effect of Code.Org Activities on Computational Thinking and Algorithm Development Skills

Ali Oluk [1] Recep ÇAKIR [2]

[1] Amasya University, Faculty of Education, CEIT, Amasya, Turkey
alioluk85@gmail.com
[2] Amasya University, Faculty of Education, CEIT, Amasya, Turkey
recepcakir@gmail.com

Article Info

Article History
Received: 30/06/2021
Accepted: 26/11/2021
Published: 31/12/2021

Keywords:
Computational Thinking, Code.Org, Algorithm Development.

ABSTRACT

With the sub-skills covered, there are many studies aimed at providing students with computational thinking skills that are known to be an important skill for today's students. In this study, it is aimed to investigate the effect of code.org applications on the development of computational thinking and algorithm development skills of the students. In this study, quasi experimental research design with pre-test and post-test control group was used. A total of 67 middle school of 6th grade students, 32 of who were in the control group and 35 in the experimental group, participated in the study. The study was planned to cover 6 weeks of information technology and software courses with students. The course was enriched with the applications in Code.Org site for the experimental group students. The control group was treated appropriately course curriculum to their students. In the study, the scale of computational thinking skill levels and algorithm development achievement test were applied to the students as pre-test and post-test. When the data obtained in the study is examined, it is seen that there is no significant difference between the pre-test results of algorithm development achievement test and computational thinking skill levels scale. However, when the differences between pre-test and post-test scores of both tests were examined, it was seen that there was a significant difference in favor of the experimental group. As a result, it can be said that code.org applications used by experimental group students have positive effect on developing algorithms and computational thinking skills of students.

Citation: Oluk, A. & Çakır, R. (2021). The effect of code.org activities on computational thinking and algorithm development skills. *Journal of Teacher Education and Lifelong Learning*, *3*(2), 32-40.

INTRODUCTION

The world around us is changing rapidly, and countries are updating their education systems to keep up with that change. Today, students are expected to develop skills that were not even heard of in the past. Computational thinking is one of those skills, which has been introduced to curricula over the past years. Researchers predict that computational thinking will be one of the fundamental skills (e.g., reading, writing, and basic math) by mid 21st century (Wing, 2006; Wing, 2014).

Therefore, every student needs to acquire computational thinking skills (Barr & Stephenson, 2011; Grover & Pea, 2013). However, getting students to develop computational thinking skills presents challenges to education systems (Wing, 2008). To overcome those challenges, many countries update their education systems and broaden their curricula to include activities tailored to computational thinking skills (Angeli & Valanides, 2019; Grover & Pea, 2013; León & Robles, 2015). Some examples are the computer science curriculum in the USA (Collage Board, 2013; Collage Board, 2016); four-tier curriculum for 5-16-year-olds in the UK (Department for Education, 2013; Royal Society, 2012); and the information technology and software (ITS) course (MEB, 2017a) and the computer science course (MEB, 2017b) in Turkey.

Computational thinking is a critical skill because it helps students recognize and solve problems (Czerkawski & Lyman, 2015). However, computational thinking is a multidimensional concept that involves algorithmic thinking, critical thinking, communication, cooperative learning, and creative thinking (ISTE, 2015; Korkmaz et al., 2017; İbili et al., 2020; Yağcı, 2019). Students today are expected to develop those subskills as well (Günüç, Odabaşı & Kuzu, 2013). Therefore, we should provide students with programming education to help them acquire creative thinking, critical thinking, and problem-solving skills (Akpınar & Altun, 2014; Karabak & Güneş, 2013; Monroy-Hern´andez & Resnick, 2008; Shin et al., 2013;). In other words, programming education is a powerful tool by which students can develop computational thinking skills (Lye & Koh, 2014; Oluk & Korkmaz, 2016; Oluk et al., 2018; Sayın, 2020).

Programming education is challenging for beginners, and therefore, it is more suited to students with a certain level of proficiency in algorithms and coding (Genç & Karakuş, 2011). However, some visual software programs (e.g., Code.org, Microsoft Small Basic, Scratch, and Alice) allow less code-savvy students to learn to program easily (Çatlak et al., 2015; Yılmaz, 2019). Code.org is designed to help anyone learn basic programming in an easy and fun way (Demirer & Sak, 2016). Teachers can use Code.org to teach beginners how to code (Yecan et al., 2017). It is already a popular tool commonly used in block-based and computerless coding activities (Sayın, 2020).

Funded by big companies (Microsoft, Facebook, and Google), Code.org was launched in 2013 to promote computer science education (Code.Org, 2019). Code.org is a website where anyone interested in programming can learn how to code by completing activities and drag-and-drop tasks for all levels. The website also allows teachers to monitor their students' progress and provides a certificate to those who complete all stages of training.

There is a large body of research investigating the relationship between computational thinking skills and programming education (Atmatzidou & Demetriadis, 2016; Bers et al., 2014; Brennan & Resnick, 2012; Lye & Koh, 2014; León & Robles, 2015; Oluk & Korkmaz, 2016; Oluk et al., 2018). Oluk and Korkmaz (2016) gave fifth graders block-based programming education within the scope of the ITS course. They concluded that students who developed programming skills were more likely to acquire computational thinking skills. Oluk et al. (2018) determined that Scratch, a block-based visual programming language, helped fifth graders develop computational thinking skills. Atmatzidou and Demetriadis (2016) also found that robotic coding education helped students pick up computational thinking skills. Therefore, research shows that educators often provide programming education to help students learn computational thinking skills.

Computational thinking skills are a prerequisite not only for people interested in computer science but for all those interested in other branches of science (Guzdial, 2008; Korkmaz el al., 2015; Yadav et al., 2014). Research shows that programming education plays a crucial role in developing computational thinking skills (Lye & Koh, 2014). People with high-level thinking and problem-solving skills are more likely to pick up programming skills (Yükseltürk & Altıok, 2015). However, such people need to know the logic of algorithms to be able to acquire those skills. Therefore, programming teaching involves algorithms and flowcharts (Köse & Tüfekçi, 2015).

Code.org is a promising tool for programming education. This paper investigated whether Code.org helped beginner students develop computational thinking and algorithm development skills. Studies address different aspects of computational thinking. For example, they define the concept (Bundy, 2007; Voogt et al., 2015; Wing, 2006; Wing, 2011), focus on the evolution of computational thinking research (Kalelioğlu et al., 2016; Şahiner & Kert, 2016), incorporate it into curricula (Barr & Stephenson, 2011; Lye & Koh, 2014), examine its relationship with computer science and other sciences (Barcelos & Silveira, 2012; Czerkawski & Lyman, 2015; Liu & Wang, 2010; Mishra & Yadav, 2013; Orton et al., 2016; Weintrop et al., 2016), and programming (Atmatzidou & Demetriadis, 2016; Brennan & Resnick, 2012; Bers et al., 2014; Oluk & Korkmaz, 2016). This is the first experimental study to look into the effect of Code.org activities on computational thinking and algorithm development skills in secondary school students. The research questions are as follows:

1. Do Code.org activities help secondary school students develop computational thinking skills?
2. Do Code.org activities help secondary school students develop algorithm development skills?

METHOD

This quantitative study adopted a quasi experimental pretest-posttest control group design, which is employed to determine the effect of an intervention on dependent variables. The intervention in this study was a set of Code.org activities within the scope of the ITS course. The activities had three learning outcomes: (1) learning the logic of algorithm development, (2) choosing the right algorithm, and (3) editing faulty algorithms. The experimental group took part in the Code.org activities, while the control group received education according to the current curriculum.

Study Group

The sample consisted of 67 sixth graders divided into two groups: experimental (n=35; 18 girls and 17 boys) and control (n=32; 16 girls and 16 boys). Table 1 shows the gender distribution of the groups.

Table 1. *Gender distribution by groups*

Group	Gender				
	Girl		Boy		Total
	N	%	N	%	N
Control	16	50	16	50	32
Experimental	18	51.4	17	48.6	35
Total	34	50.7	33	49.3	67

Procedure

The information technology and software course was a 2-hour course for sixth graders. The experimental group participated in Code.org activities within the scope of the ITS course for six weeks. The control group did class based on the current curriculum involving lecturing and practicing examples on the board.

The Code.org activities focused on the basics of algorithms, such as conditions, variables, loops,

and nested loops. The experimental group participants were handed out pieces of paper for computerless activities. They completed the computer-based activities in a computer lab.

The teacher provided the control group participants with examples of algorithms and flow diagrams and delivered the lectures based on the current curriculum. The participants solved the examples in front of the class so that all students could follow the process.

The experimental group participants performed the computerless and computer-based activities in the "Introduction to Computer Science- Express Course" on code.org. Each activity has an example situation, blocks, and a workspace to move the blocks to. When clicking the run button, the user can drag and drop the blocks to make the program run. When clicking the "show code" button, the user can see the assembled blocks in JavaScript. These activities aim to help users acquire fundamental algorithm skills in a progressive fashion.

Research Instruments and Processes

The information technology and software course was a 2-hour course for sixth graders. The experimental group participated in Code.org activities within the scope of the ITS course for six weeks. The control group did class based on the current curriculum involving lecturing and practicing examples on the board.

The Code.org activities focused on the basics of algorithms, such as conditions, variables, loops, and nested loops. The experimental group participants were handed out pieces of paper for computerless activities. They completed the computer-based activities in a computer lab.

The teacher provided the control group participants with examples of algorithms and flow diagrams and delivered the lectures based on the current curriculum. The participants solved the examples in front of the class so that all students could follow the process.

The experimental group participants performed the computerless and computer-based activities in the "Introduction to Computer Science- Express Course" on code.org. Each activity has an example situation, blocks, and a workspace to move the blocks to. When clicking the run button, the user can drag and drop the blocks to make the program run. When clicking the "show code" button, the user can see the assembled blocks in JavaScript. These activities aim to help users acquire fundamental algorithm skills in a progressive fashion.

Computational Thinking Skill Levels Scale

The Computational Thinking Skill Levels Scale (CTSLS) was used as a pretest-posttest. The instrument was developed by Korkmaz et al. (2015) to determine secondary school students' computational thinking skill levels. The instrument consists of four subscales (problem-solving, critical thinking, creativity, collaboration and algorithmic thinking) and 22 items scored on a five-point Likert-type scale. The CTSLS has item test correlation coefficients of 0.655 to 0.862 and regression values of 0.507 to 0.872. These values indicate that the CTSLS is a valid and reliable instrument to assess computational thinking skills.

Algorithm Development Achievement Test

The Algorithm Development Achievement Test (ADAT) was developed by Oluk, Korkmaz, and Oluk (2018) to measure three algorithm-related skills: (1) comprehending the logic of algorithms, (2) choosing the best algorithm, and (3) editing faulty algorithms. ADAT consists of 20 items. It has an item discrimination index of 0.33 to 0.48, an item difficulty index of 0.66, and a KR-20 internal consistency coefficient of 0.85 (Oluk et al., 2018).

Data Analysis

The data were analyzed using the Statistical Package for Social Sciences (SPSS). First, a normality test was conducted. The results showed that the data were normally distributed. Second, an independent groups t-test was used to determine the differences in CTSLS and ADAT scores between the groups.

RESULTS

CTSLS and ADAT Pretest Scores

An independent t-test was used to determine whether there was a significant difference in CTSLS pretest scores between the groups. Table 2 shows the results.

Table 2. *CTSLS pretest scores*

Group	N	$\overline{X}$	S	Sd	t	P
Experimental	**35**	85.66	16.95	65	1.15	.254
Control	**32**	89.94	13.03			

There was no statistically significant difference in CTSLS pretest scores between the experimental ($\overline{X}$ =85.66) and control ($\overline{X}$ =89.94) groups [t(65)=1.15, p>.05] (Table 2).

An independent sample t-test was performed to determine whether there was a significant difference in ADAT pretest scores between the groups. Table 3 shows the results.

Table 3. *ADAT pretest scores*

Group	N	$\overline{X}$	S	Sd	t	P
Experimental	**35**	26.71	13.00	65	1.46	.149
Control	**32**	22.19	12.31			

There was no statistically significant difference in ADAT pretest scores between the experimental ($\overline{X}$ =26.71) and control ($\overline{X}$ =22.19) groups [t(65)=1.46, p>.05] (Table 3).

Algorithm Development Skills

There was no statistically significant difference in ADAT pretest scores between the groups. Therefore, improvement scores (posttest score minus pretest score) were calculated, and then, between-group differences were determined using an independent t-test. Table 4 shows the results.

Table 4. *Analysis of ADAT improvement scores*

Group	N	$\overline{X}$	S	Sd	t	P
Experimental	35	38.91	8.67	65	5.21	.000
Control	32	51.14	10.53			

The experimental group had a significantly higher ADAT improvement score ($\overline{X}$ =51.14) than the control group ($\overline{X}$ =38.91) [t(65)=5.21, p<.01] (Table 4). This result showed that the Code.org activities were better at helping students develop algorithm development skills than the current curriculum.

Computational Thinking Skills

An independent t-test was used to determine whether there was a statistically significant difference in CTSLS improvement scores between the groups. Table 5 shows the results.

Table 5. *Analysis of CTSLS improvement scores* subscale

	Group	N	$\overline{X}$	S	Sd	t	P
Creativity	**Experimental**	35	35	1.80	3.79	65	0.51
	Control	32	32	1.38	2.88		
Algorithmic thinking	**Experimental**	35	2.46	3.68	65	2.22	.000
	Control	32	0.72	2.58			
Collaboration	**Experimental**	35	3.03	3.90	65	2.01	.04
	Control	32	1.31	3.06			
Problem-solving	**Experimental**	35	1.94	3.78	65	0.76	.45
	Control	32	1.28	3.35			
Critical thinking	**Experimental**	35	3.45	5.90	65	2.92	.005
	Control	32	-1.94	9.04			
Total	**Experimental**	35	12.69	12.68	65	3.42	.001
	Control	32	2.75	10.97			

The experimental group had a significantly higher CTSLS improvement score ($\overline{X}$ =12.69) than the control group ($\overline{X}$ =2.75) [t(65)=3.42 p<.05]. There was no statistically significant difference in CTSLS "creativity" improvement scores between the experimental ($\overline{X}$ =1.80) and control groups ($\overline{X}$ =1.38) [t(65)=0.51 p>.05]. The experimental group had a significantly higher CTSLS "algorithmic thinking" improvement score ($\overline{X}$ =2.46) than the control group ($\overline{X}$ =0.72) [t(65)=0.03 p<.05]. The experimental group had a significantly higher CTSLS "collaboration" improvement score ($\overline{X}$ =3.03) than the control group ($\overline{X}$ =1.31) [t(65)=0.04 p<.05]. There was no statistically significant difference in CTSLS "problem-solving" improvement scores between the experimental ($\overline{X}$ =1.94) and control groups ($\overline{X}$ =1.28) [t(65)=0.45 p>.05]. The experimental group had a significantly higher CTSLS "critical thinking" improvement score ($\overline{X}$ =3.45) than the control group ($\overline{X}$ =-1.94) [t(65)=0.005 p>.05] (Table 5).

CONCLUSION AND DISCUSSION

There was no statistically significant difference in ADAT pretest scores between the experimental and control groups. However, the experimental group had a significantly higher ADAT improvement score (posttest score minus pretest score) than the control group. This result showed that the Code.org activities were better at helping students develop algorithm development skills than the current curriculum. Visual programming tools are easy tools for teaching concepts, such as logical structures, loops, and variables (Yükseltürk & Altıok, 2016). Code.org is a useful tool for beginners (Yecan et al., 2017). It helps users learn the logic of algorithms and algorithm-related concepts (e.g., condition, loop, and variable) (Code.org, 2019). Code.org is popular among teachers interested in teaching their students the logic of algorithms (Dönmez Usta & Turan Güntepe, 2019). Code.org also helps students learn how

to figure out coding problems (Arfe et al., 2020). Therefore, we can state that Code.org provides students with the opportunity to develop algorithm development skills.

There was no significant difference in CTSLS pretest scores between the experimental and control groups. However, the experimental group had significantly higher CTSLS posttest scores than the control group. This result showed that the Code.org activities helped students develop computational thinking skills. Research, in general, shows that students who learn to program are more likely to develop computational thinking skills (Lye & Koh, 2014; Oluk et al., 2018; Rijke et al., 2018). For example, Brennan and Resnick (2012) used a drag-and-drop programming tool to help students acquire computational thinking skills. The researchers concluded that the students who participated in the programming activities had higher computational thinking skills than those who did not. Oluk et al. (2018) also found that block-based programming tools helped students develop computational thinking skills. Oluk and Korkmaz (2016) provided students with a training program in which they used a visual programming tool to develop a project. The researchers determined that the training program improved the participants' computational thinking skills. As part of the "Coding Week" in Turkey, teachers from different branches offer programming training to their students, who find a chance to take part in activities tailored to computational thinking skills (Sayın, 2020). All these results indicate that programming education and block-based programming tools help students develop computational thinking skills.

Code.org is a drag-and-drop programming tool used to teach students of all ages the fundamentals of programming and computation. Our results show that code.org activities help students develop computational thinking and algorithm development skills. Therefore, we think that activities on algorithm development skills within the scope of the ITS course should be integrated with block-based drag-and-drop programming tools (e.g., Code.org) to provide students with the opportunity to acquire computational thinking skills as well. We also think that young students should be encouraged to use block-based programming tools to develop algorithm development skills so that they can put those skills into practice in text-based programming languages. All courses should incorporate appropriate programming tools to allow students to improve their computational thinking skills.

REFERENCES

Akpınar, Y., & Altun, A. (2014). Bilgi Toplumu Okullarında Programlama Eğitimi Gereksinimi. İlköğretim Online, 13(1), 1-4.

Angeli, C., & Valanides, N. (2019). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. Computers in Human Behavior, 105(2020), 1-13.

Arfe, B., Vardanega, T. & Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. Computers Education, 148(2020), 1-16.

Atmatzidou, S., & Demetriadis, S. (2016). Advancing Student' Computational Thinking Skills Through Educational Robotics: A Study On Age And Gender Relevant Differences. Robotics and Autonomous System, 75(2016), 661-670.

Barcelos, T., & Silveira, I. (2012). Teaching computational thinking in initial series an analysis of the confluence among mathematics and computer sciences in elementary education and its implications for higher education. Conferencia Latinoamericana En Informatica, Medellin, Colombia.

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? ACM In Roads, 2(1), 48-54.

Bers, M., Flannery, L., Kazakoff, E., & Sullivan, A. (2014). Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum. Computer & Education, 72(2014), 145-157.

Brennan, K., & Resnick, M. (2012). New Frameworks For Studying And Assessing The Development of Computational Thinking. American Educational Research Association, Vancouver, Canada.

Bundy, A. (2007). Computational Thinking Is Pervasive. Journal of Scientific and Practical Computing, 1(2), 67-69.

Code.Org Hakkında. Code Org. Retrived November 15, 2019, from https://code.org/international/about

Czerkawski, B., & Lyman, E. (2015). Exploring Issues About Computational Thinking in Higher Education. TechTrends, 59(2), 57-65.

Çatlak, Ş., Tekdal, M., & Baz, F. (2015). Scratch Yazılımı İle Programlama Öğretiminin Durumu: Bir Döküman İnceleme Çalışması. Journal of Instructional Technologies & Teacher Education, 4(3), 13 - 25.

Demirer, V., ve Sak, N. (2016). Programming education and new approaches around the world and in Turkey/Dünyada ve Türkiye'de programlama eğitimi ve yeni yaklaşımlar. Eğitimde Kuram ve Uygulama, 12(3), 521-546.

Department For Education. (2013). Computing programmes of study: key stages 1 and 2 National curriculum in England. Retrieved February 12, 2017, https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study

Dönmez Usta, N. ve Turan Güntepe, E. (2019). Bilişim Teknolojileri Rehber Öğretmenlerinin Programlama Araçlarına İlişkin Deneyimlerinin İncelenmesi. Amasya Üniversitesi Eğitim Fakültesi Dergisi, 8(2), 373-396.

Genç, Z. ve Karakuş, S. (2011). Tasarımla öğrenme: Eğitsel bilgisayar oyunları tasarımında Scratch kullanımı. 5. Internatiol Computer & Instructional Technologies Symposium, Fırat Üniversitesi, Elazığ.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38-43.

Guzdial, M. (2008). Education Paving the Way for Computational Thinking. Communications Of The ACM, 51(8), 25-27.

Günüç, S., Odabaşı, H., & Kuzu, A. (2013). 21. Yüzyıl Öğrenci Özelliklerinin Öğretmen Adayları Tarafından Tanımlanması: Bir Twitter Uygulaması. Eğitimde Kuram ve Uygulama, 9(4), 436-455.

ISTE. (2015). Retrieved February 12, 2015, http://www.iste.org/docs/ct-documents/ctleadershipt-toolkit.pdf?sfvrsn=4.

İbili, E., Günbatar, M. S., & Sırakaya, M. (2020). Bilgi-işlemsel düşünme becerilerinin incelenmesi: Meslek liseleri örneklemi. Kastamonu Education Journal, 28(2), 1055-1067.

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.

Karabak, D., & Güneş, A. (2013). Ortaokul Birinci Sınıf Öğrencileri için Yazılım Geliştirme Alanında Müfredat Önerisi. Eğitim ve Öğretim Araştırma Dergisi, 2(3), 175-181.

Korkmaz, Ö., Çakır, R., & Özden, M. (2015). Bilgisayarca Düşünme Beceri Düzeyleri Ölçeğinin (BDBD) Ortaokul Düzeyine Uyarlanması. Gazi Eğitim Bilimleri Dergisi, 1(2), 143-162.

Köse, U.,& Tüfekçi, A. (2015). Algoritma ve Akış Şeması Kavramlarının Öğretiminde Akıllı Bir Yazılım Sistemi Kullanımı. Pegem Eğitim ve Öğretim Dergisi, 5(5), 569-586.

León, J., & Robles, G. (2015). Analyze Your Scratch Projects With Dr. Scratch And Assess Your Computational Thinking Skills. 7th International Scratch Conferance, Amsterdam, Netherlands.

Liu, J., & Wang, L. (2010). Computational Thinking in Discrete Mathematics. Second International Workshop on Education Technology and Computer Science, 2010(1). 413-416.

Lye, S., & Koh, J. (2014). Review On Teaching And Learning Of Computational Thinking Through Programing: What Is NextFor K - 12?,Computers In Human Behavior, 51-61.

MEB. (2017 a). Bilişim Teknolojileri ve Yazılım Dersi Öğretim Programı. Retrieved February 12, 2017, http://mufredat.meb.gov.tr/Default.aspx

MEB. (2017 b). Bilgisayar Bilimi Dersi Öğretim Programı Kur1 - Kur2. Retrieved February 12, 2017, http://ttkb.meb.gov.tr/www/ogretim-programlari/icerik/72

Mishra, P., & Yadav, A. (2013). Of Art and Algorithms: Rethinking Technology & Creativity in the 21st Centruy. TechTrends, 57(3), 10-14.

Monroy-Hernández, A., & Resnick, M. (2008). Feature empowering kids to create and share programmable media. Interactions, 15(2), 50-53.

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.

Oluk, A., Korkmaz, Ö., & Oluk, H. A. (2018). Scratch'ın 5. sınıf öğrencilerinin algoritma geliştirme ve bilgi-işlemsel düşünme becerilerine etkisi. Türk Bilgisayar ve Matematik Eğitimi Dergisi, 9(1), 54-71.

Orton, K., Weintrop, D., Beheshti, E., Horn, M., Jona, K., & Wilensky, U. (2016). Bringing Computational Thinking Into High School Mathmatics and Science Clasrooms. ICLS 2016 Proceedings, (s. 705-712).

Rijke, W. J., Bollen, L., Eysink, T. H. S., & Tolboom, J. L. J. (2018). Computational thinking in primary school: An examination of abstraction and decomposition in different age groups. Informatics in Education, 17(1), 77–92.

Royal Society. (2012). Shut Down Or Restart? The Way Forward For Computing In UK Scholls. London: The Royal Academy Of Engineering.

Sayın, Z. (2020). Öğretmenlerin Kodlama Eğitiminde Eğilimlerinin Belirlenmesi. Öğretim Teknolojileri ve Öğretmen Eğitimi Dergisi, 9(1), 52-64.

Shin, S., Park, P., & Bae, Y. (2013). The Effects of an Information-Technology Gifted Program on Friendship Using Scratch Programming Language and Clutter. International Journal of Computerand Communication Engineering,

2(3), 246-249.

Şahiner, A., & Kert, S. (2016). Komputasyonel Düşünme Kavranı ile İlgili 2006 - 2015 Yılları Arasındaki Çalışmaların İncelenmesi. Avrupa Bilim ve Teknoloji Dergisi, 5(9), 38-43.

The Collage Board. (2013). Retrieved February 12, 2016, http://www.csprinciples.org/home/about-the-project/docs/csp-cf-2013.pdf?attredirects=0&d=1

The Collage Board. (2016). AP Computer Science Principles Including the Cirriculum Framework. Retrieved February 12, 2016, https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Educ Inf Techno, 20(2015), 715-728.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining Computational Thinking for Mathmatics and Science Classrooms. J Sci Educ Technol, 25(1), 127-147.

Wing, J. M. (2014). Computational thinking benefits society. 40th Anniversary Blog of Social Issues in Computing, 2014.

Wing, J. (2011). Computational Thinking: What and Why? The Link Magazine, 6, 20-23Wing, J.

(2006). Computational Thinking. Commun. ACM, 33-35.

Wing, J. (2008). Computational Thinking and Thinking About Computing. Phil. Trans. R. Soc. A, 3717-3725, doi:10.1098/rsta.2008.0118.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. (2014). Computational Thinking in Elementary and Secondary Teacher Education. ACM Transactions on Computing Education, 14(1), 1-5.

Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. Education and InformationTechnologies, 24(1), 929-951.

Yecan, E., Özçınar, H., & Tanyeri, T. (2017). Bilişim Teknolojileri Öğretmenlerinin Görsel Programlama Öğretimi Deneyimleri. İlköğretim Online, 16(1), 377-393.

Yılmaz, Ş. (2019). Scratch programı öğretiminde birlikte öğrenme tekniği kullanımının öğrencilerin akademik başarısına ve öz yeterlik algısına etkisi. Unpublished Master Dissertation, Afyon Kocatepe Üniversitesi, Fen Bilimleri Enstitüsü, Afyon.

Yükseltürk, E., & Altıok, S. (2015). Bilişim Teknolojileri Öğretmen Adaylarının Bilgisayar Programlama Öğretimine Yönelik Görüşleri. Amasya Üniversitesi Eğitim Fakültesi Dergisi, 4(1), 50-65.

Yükseltürk, E., ve Altıok, S. (2016). Bilişim teknolojileri öğretmen adaylarının programlama öğretiminde Scratch aracının kullanımına ilişkin algıları. Mersin Üniversitesi Eğitim Fakültesi Dergisi, 12(1), 39-52.