OPEN ACCESS

CONTEMPORARY EDUCATIONAL TECHNOLOGY

ISSN: 1309-517X (Online) 2022, 14(1), ep341, https://doi.org/10.30935/cedtech/11477

Research Article

An Investigation of The Effect of Block-Based Programming and Unplugged Coding Activities on Fifth Graders' Computational Thinking Skills, Self-Efficacy and Academic Performance

Nihan Arslan Namli
Department of Computer Technologies, Dortyol Vocational School of Higher Education, Iskenderun Technical University, Turkey
ORCID: 0000-0002-5425-1468

Birsel Aybek
Curriculum and Instruction Department, Faculty of Education, Cukurova University, Turkey
ORCID: 0000-0001-5846-9838

Received: 5 Aug 2021 Accepted: 19 Oct 2021

Abstract

This paper investigated the effect of block-based programming and unplugged coding teaching activities on fifth graders' computational thinking skills, self-efficacy, and academic performance. The teaching activities were conducted within the scope of the "Problem-Solving and Programming" unit of the Information Technologies and Software (ITS) course. The sample consisted of 82 fifth graders of three public middle schools in the academic year of 2020-2021. Participants were recruited using random sampling. The study adopted an embedded mixed design. The quantitative stage employed a pretest-posttest randomized control group design, while the qualitative staged employed a case study. Quantitative data were collected using the Computational Thinking Self-efficacy Scale (CTSES), the International Informatics and Computational Thinking Activity Task Test (IICTATT), and a Computational Thinking Performance Test (CTPT) developed by the researcher. Qualitative data were collected using a semi-structured interview questionnaire. The quantitative data were analyzed using the Kruskal Wallis H, paired sample t-test, and ANCOVA test on the Statistical Package for Social Sciences (SPSS). The qualitative data were analyzed inductively using MAXQDA. There was no significant difference in CTSES scores between groups. Experimental 2 had higher IICTATT and CTPT scores than Experimental-1 and control groups. The qualitative findings were grouped into seven categories.

Keywords: computational thinking, computerless computer education, block-based programming, programming education, self-efficacy

INTRODUCTION

National and international research institutions make great efforts for the rapid development of information and communication technologies and the global economy in order to identify 21st-century skills and competencies. Today, students should turn into empowered learners, digital citizens, knowledge builders, innovative designers, computational thinkers, creative communicators, and global collaborators in an increasingly digital world (International Society for Technology in Education, 2016). According to the Institute for the Future (2011), computational thinking (CT) will be one of the key skills in business life by 2020. Different researchers classify 21st-century skills differently. Therefore, CT is classified under the categories of digital age literacy, information, media, and technology skills. This shows that CT is a 21st-century skill.

Although Papert coined the term "computational thinking," Wing (2006) made it known by his three-paper article. According to Wing, CT is composed of four skills: (1) decomposing complex problems into solvable forms; (2) using variables to recognize patterns of a system; (3) abstracting and analyzing the patterns; and (4) putting heuristic reasoning into practice. Mannila et al. define CT as a process in which terms and processes in informatics are transferred to formulate and solve problems. On the other hand, Riley and Hunt (2014) focus on cognitive processes and defines CT as a process in which one reflects on problems like a computer scientist. Cuny et al. (2010) define CT as a thinking process in which an information processor (computer, machine, or human) formulates problems effectively.

Many countries have adopted a new approach and integrated the concept of CT into their education programs. The United States of America (USA) has integrated CT into its high-school computer science course programs, while England started to provide CT education to students aged 5-16 (Voogt et al., 2015). Many other countries (France, Finland, Italy, Portugal, Malta, Croatia, and Scotland) have proposed a comprehensive approach to digital learning and teaching. Turkey offers primary school students CT within the scope of the Information Technologies and Software (ITS) course (Ministry of National Education of Turkey, 2018a). High school students in Turkey receive CT within the scope of the Computer Science course (Ministry of National Education of Turkey, 2018b). The curriculum of the ITS course sets target skills. The curricula of both courses focus on CT under the heading of "Computational Thinking" (Ministry of National Education of Turkey, 2018b). Since then, researchers and educators have been working on incorporating CT into the curricula and how to teach and evaluate it.

Although there has been a growing body of research on CT in recent years, it is still an understudied area. What is more, when those studies were holistically evaluated, only a few concentrates on developing and evaluating CT skills, whereas none of them propose an instructional design. Some of those studies employ experimental designs (Goel & Kathuria, 2010; Hung, 2012; Ismail et al., 2010; Kazakoff & Bers, 2012; Kyungbin & Jonassen, 2011; Kose et al., 2013; Wang & Chen, 2010), while some others are case or action studies (Esteves et al., 2011; Fessakis et al., 2013; Kahn et al., 2011; Kordaki, 2010; Lin & Liu, 2012; Moreno, 2012; Wang et al., 2012). In addition, there is no mixed design study that focuses on teaching CT skills. Moreover, the samples of those studies consist of K-12 and undergraduate students, but none focus on middle school students.

CT is a high-level thinking skill. Students with CT skills are likely to adapt to technological life more easily and become better prepared for work life. Students can become lifelong learners and develop other high-level thinking skills without being affected by rapidly transforming tools and materials. The "Information Technologies and Software" course consists only of a Teacher Guide. It has no other sources, such as a textbook or an activity book. That is why the instructional design proposed by this study will be a resource for teachers and students. The present study aimed to help students develop CT skills and also help teachers improve the course content and the methods and techniques. The sample was divided into three groups: Experimental 1 (n=24), Experimental 2 (n=28), and control (n=30). Experimental 2 used computers, while Experimental 1 did not. The goal was not to help participants develop computer skills but it was to help them make sense of concepts and develop CT skills to enable them to learn on their own and it was more than just teaching them how to code and program. The aim is also encouraging them to adopt different perspectives to solve problems on different platforms and to put CT skills into practice in academic life. "Problem-Solving and Programming" unit within the scope of the ITS course was chosen and designed a teaching technique and then implemented and evaluated it. Experimental 1 performed unplugged coding (UC) programming activities. Experimental 2 performed block-based programming (BBP) activities. The control group was lectured based on the ITS course curriculum with no interventions. Data were collected using the CT Self-efficacy Scale (CTSES), the International Informatics and Computational Thinking Activity Task Test (IICTATT), and a Computational Thinking Performance Test (CTPT) developed by the researcher. Therefore, the main research question was, "How do BBP and UC teaching activities affect fifth graders' CT skills, self-efficacy, and academic performance."

In general, the rationale of the study can be listed as follows: firstly, considering that there is not enough research on CT in Turkey. Especially since there are limited research carried out with mixed methods. This research is important in order to fill this deficiency. Secondly to enable students to gain CT skills and thus enable them to adapt to technological life more easily and become lifelong learning individuals. Thirdly, to design and implement an instructional design for the development and evaluation of ICT skills. Considering the fact that the CT skills has recently began to enter curricula all over the world. It could be asserted that there is a need for CT studies at this level. The focal point of this study involves the effect of block-based programming and unplugged coding activities on CT skills, self-efficacy and academic performance in order to remove this deficiency in the literature. Taking all this into account, the study sought answers to the following sub questions:

1. Is there a significant difference in posttest CTSES scores between the groups?
    - Is there a significant difference in posttest CTSES “reasoning” subscale scores between the groups?
    - Is there a significant difference in posttest CTSES “abstraction” subscale scores between the groups?
    - Is there a significant difference in posttest CTSES “decomposition” subscale scores between the groups?
    - Is there a significant difference in posttest CTSES “generalization” subscale scores between the groups?
    - Is there a significant difference in posttest IICTATT scores between the groups?
2. Is there a significant difference in posttest CTPT scores between the groups?
3. What do participants think about the BBP and UC teaching activities?

METHOD

Research Model

The study adopted an embedded mixed design consisting of two stages: quantitative and qualitative (Creswell, 2012). In embedded mixed designs, the quantitative stage is the dominant research methodology, whereas the qualitative stage is conducted only to support quantitative data. Qualitative and quantitative data were collected before, during, and after the interventions (BBP and UC teaching activities). In the first stage, a pilot study was performed before the interventions to collect the qualitative data. The quantitative data were collected to determine participants’ CT skills before the interventions. Afterwards, participants performed the BBP and UC teaching activities, and quantitative data were collected to determine their experiences and the progress of the interventions. Qualitative data were collected again to assess participants’ post-interventions CT skills. The quantitative stage employed a pretest-posttest randomized control group design, which is an experimental design. The qualitative stage employed a case study, involving the in-depth analysis of a situation or phenomenon (Creswell, 2012). **Table 1** shows the symbolic representation of the research design.

Table 1. Symbolic representation of the research design

Group		Pretest	İnterventions	Posttest
Experimental 1	M	O1	X1	O4, G1
Experimental 2	M	O2	X2	O5, G2
Control	M	O3	X3	O6

M: Random Assignment
O1, O2, O3: Demographic Characteristics Questionnaire, CTSES, IICTATT, CTPT
X1: Teaching UC-based problem-solving and programming unit
X2: Teaching BBP-based problem-solving and programming unit
X3: Teaching the current curriculum with no interventions
O4, O5, O6: CTSES, IICTATT, CTPT
G1, G2: Participants’ views

The groups were randomly assigned. The CTSES, IICTATT, and CTPT were used as pretests. Experimental 1 participated in UC activities. Experimental 2 participated in BBP activities. The control group was lectured based on the ITS course curriculum with no interventions (**Table 1**).

Designing a Teaching Technique

The learning environment was based on ADDIE (analysis, design, development, implementation, and evaluation) (Driscoll, 2002; Kaminski, 2007; McGriff, 2000). The "analysis" stage involved a pilot study and a needs analysis to assess the current situation. In the design and development stages, The ITS course was planned according to the general purposes, learning outcomes, and content of the Ministry of National Education of Turkey curriculum. UC- and BBP-based lesson plans were drawn up for Experimental Groups 1 and 2, respectively. The control group covered the current lesson plan with no interventions. In the implementation stage, the lesson plans were implemented for 28 hours of class to observe the process and to detect potential problems. The ITS instructor delivered the lessons to the control group. In the evaluation stage, participants' performance and self-efficacy in CT skills were assessed using the CTSES, IICTATT, and CTPT before (pretest) and after (posttest) the interventions. Some participants were interviewed after the interventions to augment the quantitative data with qualitative data.

Study Group

The sample consisted of 82 fifth graders of three public middle schools in the academic year of 2020-2021. After the groups were balanced in terms of variables for the quantitative stage, participants were recruited using simple random sampling, which is used to draw a sample from a population based on the principle of randomness (Büyüköztürk et al., 2012). The groups were equalized using CTPT, CTSES, and IICTATT/BEBRAS as pretests. It would not be feasible to interview all participants. Therefore, the sample of the qualitative stage consisted of seven participants from Experimental Group 1 and eight from Experimental Group 2. The interviewees were recruited using cluster sampling. Students in each classroom were clustered into low, moderate, and high GPA groups, from which the interviewees were chosen randomly.

Data Collection Tools

The data were collected using both qualitative and quantitative data collection tools. **Table 2** shows the data collection tools.

Table 2. Data collection tools

Instrument	Data Collection Period
Demographic Characteristics Questionnaire	Before Interventions
CTSES	Pretest– Posttest
IICTATT	Pretest– Posttest
CTPT	Pretest– Posttest
Interview Questionnaire	After Interventions

Descriptive variables (gender, parents' education, computer ownership, computer usage levels, and knowledge about programming and CT) were determined using a demographic characteristics questionnaire. Computational thinking self-efficacy was determined using the CTSES developed by Kukul and Karataş (2019). The CTSES consists of four subscales (reasoning, abstraction, decomposition, and generalization) and 18 items scored on a five-point Likert-type scale (1 = "Completely Disagree", 2 = "Disagree", 3 = "Neutral", 4 = "Agree", 5 = "Completely Agree"). The Computational Thinking Activity Task Test (IICTATT) consists of tasks in the informatics activity held in 46 countries, including Turkey, under "Bilge Kunduz International Informatics and Computational Thinking Activity." The test consists of 15 items rated as easy (Items 1-5), moderate (Items 6-10), and hard (Items 11-15). The number of items was reduced to nine based on a pilot study. The Computational Thinking Performance Test (CTPT) was developed to assess participants' performance at the "Problem-Solving and Programming" unit. A table of specifications was drawn up. The test consisted of 20 multiple choice questions regarding "Problem-solving Concepts and Approaches" and "Programming" learning outcomes within the scope of the unit of "Problem-Solving and Programming." The number of items was reduced to 17 based on a pilot study. Semi-structured interviews were conducted after the posttests. The researcher developed a semi-structured interview questionnaire. She consulted experts for the intelligibility and relevance of the interview questions and revised the form based on their feedback. It would not be feasible to interview all participants. Therefore, the sample of the qualitative stage consisted of a total of 15 participants (ten women and five men). Seven of them were drawn from Experimental Group 1 and eight from Experimental Group 2. The interviewees were recruited using cluster sampling. Students in each classroom were clustered into low, moderate, and high GPA groups, from which the interviewees were chosen randomly. The total duration of the interviews was 133.8 minutes.

Data Analysis

The quantitative data were tested for normality and then analyzed using the Statistical Package for Social Sciences (SPSS). The qualitative data were analyzed inductively. Categories, codes, and quotes were assessed using the Code Matrix Browser (CMB) and the Code-Subcode-Segments Model. The inductive analysis allows the researcher to explore relationships, themes, and categories in data (Patton, 2002). In inductive analysis, data are not grouped into specified categories, but findings emerge from data (Maykut & Morehouse, 1997). The qualitative data were analyzed using MAXQDA Analytics Pro 2018.

Validity and Reliability

The total 18-item CTSES had a Cronbach’s alpha (internal consistency) of 0.77. The 15-item IICTATT had a Kuder Richardson-20 (KR-20) reliability coefficient of .56, which was lower than expected due to the low number of items (Başol, 2013). The researcher developed the 20-item CTPT to assess participants’ performance at the unit of “Problem-Solving and Programming.” The test had a KR-20 reliability coefficient of .69. A KR-20 reliability coefficient of higher than .50 indicates reliability for a 10-20 item scale (Alpar, 2014). The researcher used triangulation to determine the validity and reliability of the semi-structured interviews. For credibility (internal validity), the researcher attended ITS classes for two weeks and four hours of class to get to know the teachers and students before the pilot study. For transferability (external validity), she made direct quotations from the interviews. She assigned the codes of P1.X and P2.X to the interviewees from Experimental Groups 1 and 2, respectively. She also explained the lesson plans, including the intervention process and activities. She clearly stated the research questions and problem status for consistency (internal reliability). Throughout the research, she checked the codes and themes with her dissertation advisor and reviewed them when she detected inconsistencies. The researcher filed and kept the raw data for confirmability (external reliability). Another expert also coded a randomly selected portion of the raw data. Intercoder reliability was calculated using the MAXQDA intercoder agreement and Cohen’s Kappa (κ) statistics. Cohen’s Kappa was found to be .81, indicating excellent agreement (Landis & Koch, 1977; Viera & Garrett, 2005).

RESULTS

The findings of the study are listed in accordance with the sub questions of the research.

Findings Regarding the First Sub questions

In line with the first sub questions, the Computational Thinking Self-efficacy Scale Scores were presented. Assumptions were tested before analysis. Between-group differences were determined using single-factor analysis of covariance (ANCOVA). **Table 3** shows the results.

Table 3. ANCOVA results regarding between-group differences in CTSES scores

Source of Variance	Sum of squares	sd	Mean square	F	p
Pretest(regression)	203.033	1	203.033	1.624	.206
Group	116.874	2	58.437	.467	.628
Error	9753.542	78	125.045		
Total	327039.000	82			
Corrected Total	10216.939	81			

*= p <0.05

There was no significant difference in corrected CTSES scores between the groups [F(2-78)=.467, p>.05] (**Table 3**). A paired sample t-test was used to determine any significant differences in CTSES subscale scores between the groups. **Tables 4** (Experimental 1), 5 (Experimental 2), and 6 (control) show the paired sample t-test results regarding CTSES subscale scores across groups.

Table 4. Experimental Group 1: Paired sample t-test results for CTSES and subscale scores

CTSES	Test	N	X̄	S	sd	t	p
CTSES	Pretest	24	46.75	24.42	23	-3.37	.003*
	Posttest	24	64.79	7.93			
Reasoning	Pretest	24	14.12	7.95	23	-2.66	.014*
	Posttest	24	18.91	2.61			
Abstraction	Pretest	24	13.29	7.27	23	-3.54	.002*
	Posttest	24	18.91	2.84			
Decomposition	Pretest	24	9.45	4.84	23	-2.76	.011*
	Posttest	24	12.37	2.35			
Generalization	Pretest	24	9.87	5.20	23	-3.43	.002*
	Posttest	24	14.58	3.03			

*= p <0.05

Experimental Group 1 had a higher total posttest CTSES score than pretest score [t(23)=-3.37, p<.05]. The group also had higher posttest CTSES “reasoning” [t(23)=-2.66, p<.05], “abstraction” [t(23)=-3.54, p<.05], “decomposition” [t(23)=-2.76, p<.05], and “generalization” [t(23)=-3.43, p<.05] subscale scores than pretest scores (**Table 4**). Nevertheless, the effect size was small in all scales.

There was no significant difference between pretest and posttest CTSES scores in Experimental Group 2 [t(27)=-1.36, p>.05]. Their posttest CTSES subscale scores were not significantly higher than their pretest scores (p>.05) (**Table 5**).

Table 5. Experimental Group 2: Paired sample t-test results for CTSES and subscale scores

CTSES	Test	N	X̄	S	sd	t	p
CTSES	Pretest	28	54.46	20.46	27	-1.36	.183
	Posttest	28	61.75	12.42			
Reasoning	Pretest	28	14.85	6.12	27	-1.56	.128
	Posttest	28	17.42	3.95			
Abstraction	Pretest	28	16.21	6.40	27	-1.29	.208
	Posttest	28	18.25	4.17			
Decomposition	Pretest	28	11.17	4.12	27	-1.14	.261
	Posttest	28	12.46	3.07			
Generalization	Pretest	28	12.21	4.88	27	-1.04	.306
	Posttest	28	13.60	3.67			

*= p <0.05

There was no significant difference between pretest and posttest CTSES scores in the control group [t(29)=.91, p>.05]. Their posttest CTSES subscale scores were not significantly higher than their pretest scores (p>.05) (**Table 6**).

Table 6. Control group: Paired sample t-test results for CTSES and subscale scores

CTSES	Test	N	X̄	S	sd	t	p
CTSES	Pretest	30	62.63	9.54	29	.91	.368
	Posttest	30	60.43	12.23			
Reasoning	Pretest	30	17.40	4.24	29	-.05	.954
	Posttest	30	17.46	4.64			
Abstraction	Pretest	30	17.96	3.67	29	1.84	.075
	Posttest	30	18.43	4.27			
Decomposition	Pretest	30	12.90	2.60	29	.36	.718
	Posttest	30	12.93	3.27			
Generalization	Pretest	30	14.36	2.82	29	.57	.572
	Posttest	30	15.90	3.36			

*= p <0.05

Findings Regarding the Second Sub questions

In line with the second sub questions, the International Informatics and Computational Thinking Activity Task Test Scores were presented. A paired sample t-test was used to determine significant differences between pretest and posttest IICTATT scores. **Table 7** shows the results.

Table 7. Paired sample t-test results for IICTATT scores

IICTATT	Test	N	X̄	S	sd	t	p
Experimental 1	Pretest	24	5.37	1.61	23	-1.735	.096
	Posttest	24	6.00	1.76			
Experimental 2	Pretest	28	5.96	1.73	27	2.488	.019*
	Posttest	28	6.92	1.77			
Control	Pretest	30	5.30	1.46	29	.280	.781
	Posttest	30	5.40	1.47			

*= p <0.05

Only Experimental Group 2 had a higher total posttest IICTATT score than pretest score [t(27)=2.488, p<.05]. Between-group differences were determined using ANCOVA. **Table 8** shows the results.

Table 8. ANCOVA results for between-group Differences in IICTATT Scores

Source of Variance	Sum of squares	sd	Mean square	F	p	η2
Pretest (regression)	11.417	1	11.417	4.561	.036*	.055
Group	18.956	2	9.478	3.787	.027*	.089
Error	195.240	78	2.503			
Total	2562.000	82				
Corrected Total	222.439	81				

*= p <0.05

There was a significant difference in corrected IICTATT scores between the groups [F(2-78)=3.787, p<.05] (**Table 8**). The effect size of the interventions on the IICTATT total scores was measured using partial eta square based on Cohen's (1992) criteria ($Ƞ^2$<.06 = low effect size, .14>$Ƞ^2$>.06 = moderate effect size, and Ƞ ^{2}>.14 = high effect size). The results showed that the BBP activities had a moderate effect ($η^2$=.089) on CT skills in Experimental Group 2.

Findings Regarding the Third Sub Questions

In line with the third sub questions, the Computational Thinking Performance Test Scores were presented. A paired sample t-test was used to determine significant differences between pretest and posttest CTPT scores. **Table 9** shows the results.

Table 9. Paired sample t-test results for CTPT scores

CTPT	Test	N	X̄	S	sd	t	p
Experimental 1	Pretest	24	7.54	2.44	23	-6.417	.000*
	Posttest	24	11.29	2.54			
Experimental 2	Pretest	28	7.46	2.41	27	-3.860	.001*
	Posttest	28	12.10	2.60			
Control	Pretest	30	10.20	2.82	29	-2.264	.031*
	Posttest	30	10.83	2.32			

*= p <0.05

There was a significant difference between pretest and posttest CTPT scores in all groups [Experimental 1; t(23)=-6.417, p<.05], [Experimental 2; t(27)=-3.860, p<.05], [Control; t(29)=-2.264, p<.05]. Between-group differences were determined using ANCOVA. **Table 10** shows the results.

Table 10. ANCOVA Results for Between-Group Differences in CTPT Scores

Source of Variance	Sum of squares	sd	Mean square	F	p	η2
Pretest (regression)	.148	1	.148	.024	.878	.000
Group	37.249	2	18.624	2.979	.057*	.071
Error	487.656	78	6.252			
Total	10609.000	82				
Corrected Total	532.402	81				

*= p <0.05

There was a significant difference in corrected CTPT scores between the groups [F(2-78)=2.979, p<.05] (**Table 10**). The effect size of the interventions on the CTPT total scores was measured using partial eta square. The results showed that the BBP activities had a moderate effect (η^2=.071) on CT skills in Experimental Group 2.

Findings Regarding the Fourth Sub Questions

In line with the fourth sub questions, the Participants' Views of the Interventions were presented. Participants' views of the UC and BBP activities were grouped under the categories of "definition of the concept of computational thinking," "experience of programming," "contribution of BBP and UC activities," "attitudes towards the ITS course," "challenges of BBP and UC activities," "future plans," and "interdisciplinary ideas."

None of the interviewees had heard about "computational thinking" before. When they were asked to define the concept in their own words, they mostly made references to thinking (f = 5), coding (f = 3), mathematical thinking (f = 3), programming (f = 1), problem-solving (f = 1), artificial intelligence (f = 1), computational thinking (f = 1), and algorithm (f = 1). The following are some direct quotations:

> *"It's probably about algorithms, sequences, computational thinking, mathematical thinking, and coding, like the things we learned about; like decomposition, patterns, and things like that" P1.1.*

> *"Computational thinking makes me think about calculations; I mean, that's what I think it is all about." P1.5.*

Most interviewees (f=11) had no experience whatsoever in programming or coding. They stated that they did programming or coding for the first time during doctoral thesis courses. Five interviewees noted that they had done coding before at school. They remarked that they learned programming from games (f = 2) or extracurricular courses (f = 1). The following are some direct quotations:

> *We learned coding at school. We coded our names, body weight and height, and class P2.2.*

> *"Yes, we did coding at school, in fifth grade. The instructor had talked about a software program; I can't think of its name now. We can move that cat from one computer to another" P1.6.*

Interviewees stated that the interventions taught them about algorithms (f=6), programming (f=5), coding (f = 4), and authentic learning (f = 4). They noted that they learned new concepts (f=4) and how to sequence (f = 3) and use computers (f = 3). They also expressed that the interventions encouraged them to adopt different approaches (f=2) and taught them about loops (f = 1). The following are some direct quotations:

> *"Coding…I mean, I learned that what we have in cabinets and air conditioners is actually computer stuff. Like you gave the example of traffic lights, how they switch in a loop every second." P2.4.*

> *"There was a robot in the other class. I had no idea about them. I mean, those robots that can operate on people… you gave instructions to a classmate of ours, we said 'fetch a brush,' it fetched a toilet brush and then a toothbrush." P1.6.*

Interviewees already had positive attitudes towards the ITS course, and the interventions helped them develop more positive attitudes towards it (f=11). Two interviewees had negative attitudes towards the ITS course before the interventions, but they stated that they developed positive attitudes after the interventions. Two interviewees had neither positive nor negative attitudes towards the ITS course, but they noted that they developed positive attitudes after the interventions. The following are some direct quotations:

> *"I used to get bored of it, but now I think it's fun. I was like lukewarm about it, but now it's one of my favorite courses." P1.3.*

> *"I already liked the course; now I like it more." P1.2.*

Seven interviewees stated that they faced no challenges during the interventions. However, five interviewees noted that they learned slowly. Two interviewees had difficulty coding, while one interviewee had Internet connection problems. The following are some direct quotations:

> *"In last class, we were playing "Cops and Robbers," and I had a hard time giving instructions. I couldn't do it in the last sentence, but I had no other problems." P1.3.*

> *"When I clicked on the flag, like when I was doing with the thing that went 'one, two, three…' I had a little bit of a problem when I was programming. I mean, some activities were a bit too hard for me. Another thing was, there were too many boxes to put on top of each other. It would've been better if there were fewer boxes." P2.5.*

As for future plans, almost half the interviewees (f=7) stated that they would like to pursue careers in programming and coding. Six interviewees were interested in improving themselves in programming. They also wanted to teach what they learned to other people (f=3), take further courses (f=3), and participate in competitions (f=1). The following are some direct quotations:

> *"Maybe, I can teach what I've learned to other people, or maybe I can design a game from scratch." P2.8.*

> *"I can do what you do. I can teach these things to my cousin; I mean, I can teach them to students and my parents and relatives." P1.3.*

As for interdisciplinary ideas, almost all interviewees (f=12) stated that they used programming in math class. Six interviewees used it in Turkish class, while four used it in science class. The following are some direct quotations:

"I can use it in science class. For example, I think we can definitely use algorithms in science class, or I can give examples for loops; I mean, like animals; for example, a lioness gives birth, and then that cub grows up and, if it's a female, it gives birth to her own cub. It's like a loop that lasts forever. Let me give an example for people, like, for example, my sister gave birth to a boy, and now he has a daughter, I mean, it's like a male-female loop." P2.1.

"I can use sequencing if I get confused with math problems. Also, in Turkish class, I can use it in my homework assignments, like introduction, development, and conclusion." P2.5.

DISCUSSION

There was no statistically significant difference in corrected CTSES scores between the groups. Kukul (2018) also evaluated the impact of different interventions on students' CT skills, self-efficacy, and programming performance and reported that the interventions did not affect CTSES scores. Davidson et al. (2010) also did not find any significant impact of programming class on students' self-efficacy in CT and programming.

Posttest CTSES were compared both total and subscale scores. The results showed that the UC activities helped Experimental Group 1 develop self-efficacy in reasoning, abstraction, decomposition, and generalization. However, there was no difference in the other groups. This is probably because Experimental Group 1 already had high levels of self-efficacy in CT before the intervention. Özel (2019) looked into the effect of different programming teaching methods on intragroup computational thinking self-efficacy and Özel (2019) found that only BBP-based teaching had a positive effect on computational thinking self-efficacy. However, some studies have reported no effect of the BBP teaching technique on computational thinking self-efficacy (Kalelioğlu & Gülbahar, 2014; Lockwood & Mooney, 2017; Uslu et al., 2018). High-level thinking skills, such as CT, are skills that students learn through different and rich activities and use throughout their lives (Güneş, 2012).

Only Experimental 2 participants had a higher total posttest IICTATT score than pretest score. This result is consistent with the literature (Jegede, 2009; Mazman & Altun, 2013; Wiedenbeck, 2005). Sakamoto et al. (2013) concluded that BBP-based teaching promoted CT skills and programming performance. Oluk and Korkmaz (2016) found that Turkish middle school students who learned how to use Scratch (a block-based visual programming language) developed CT skills and had high programming performance. Research, in general, shows that students who learn coding and programming from Scratch have more CT skills (Fadjo, 2012; Shin & Park, 2014; Yünkül et al., 2017; Zhang & Nouri,2019). On the other hand, Kırçalı (2019) found that students who learned coding and programming through UC activities had higher posttest IICTATT scores than pretest scores. Apostolellis et al. (2014) also reported the same result.

There were statistically significant differences between pretest and posttest CTPT scores in all groups. Experimental 2 had the highest CTPT score of the three groups, suggesting that the BBP activities improved their programming performance in the ITS course. Calder (2010) and Malan and Leitner (2007) also concluded that BBP helped students understand the basics of programming, motivated them, and boosted their academic performance and confidence. Research shows that BBP helps improve learning and achievement levels (Hughes-Roberts et al., 2019; Meerbaum-Salant, 2013).

None of the interviewees had heard about the concept of "computational thinking" before. When they were asked to define the concept, they associated it with using information to think, coding, mathematical thinking, algorithm, computational thinking, problem-solving, programming, and artificial intelligence. Most interviewees performed coding for the first time during the research. Some others stated that they had a chance to write codes at school or during extracurricular courses they attended. Two interviewees had experience with coding through games. Ramalingam et al. (2004) argue that the more experience students have with programming, the higher their programming self-efficacy and performance.

The interventions taught participants about algorithms and how to design programs and write codes. The interventions also made them more comfortable using computers, encouraged them to take their friends' points of view from different angles, and helped them put what they learned into practice, resulting in authentic learning. According to Özcan et al. (2017), BBP helps students understand the logic of programming and algorithms. Çatlak et al. (2015) state that BBP arouses curiosity, teaches the basics of algorithms and programming, and promotes creative thinking and motivation. Participants who already liked the ITS course liked it more after the interventions. Participants who had either neutral or negative attitudes towards the course stated that they liked it better after the interventions. None of the participants expressed negative attitudes towards the course after the interventions. Teaching programming improves students' attitudes towards CT (Chen et al., 2019; Robertson, 2013). Most participants experienced no difficulty during the interventions. However, some noted that they learned slowly or had a hard time writing codes. Studies state "characteristics of the learning environment" and "usability" as limitations, which make it harder for students to concentrate on class (Crues et al., 2018; Kurhila & Vihavainen, 2015). Half the participants wanted to pursue careers in informatics. Some participants were interested in improving their programming skills, while others wanted to take further training and take part in competitions. Educators should choose and apply the right activities to help students develop CT skills. This is especially important for students who want to pursue careers in informatics (Missiroli et al., 2017). As for interdisciplinary ideas, participants stated that they would like to put their programming skills into practice in math, Turkish, and science lessons. According to Yadav et al. (2014), all teachers should know how to incorporate CT into their disciplines to help their students acquire CT skills. Lockwood and Mooney (2017) argue that CT should be integrated into courses rather than being a separate course. Jona et al. (2014) also advocates that teachers incorporate CT into their disciplines rather than teach it in an isolated fashion.

CONCLUSION

In short, it was concluded that CT self-efficacy did not differ according to the groups but B^3 programming activities contributed positively to students' reasoning, abstraction, decomposition, and generalization self-efficacy. It was concluded that the block-based programming activities, had a moderate effect on CT skills. Also, in the context of the group a significant difference was found only in the block-based programming activities which had a positive effect on increasing the CT skills. There were no significant differences between the students' CT academic performance scores according to the groups. However, within the context of the group, a significant difference was found in all three groups. According to the results of the interviews with the students, all of the students stated that they had not heard the concept of CT before. It was concluded that most of the students experienced programming for the first time thanks to the research. Students stated that they improved themselves and learned new concepts in subjects such as algorithm, programming and coding. Most of them already had a positive attitude and with this study, this attitude increased even more positively and almost half of them did not experience any difficulties, and it was concluded that a certain part of them had problems arising from slow learning. Almost half of them stated that they want to acquire a profession in this field. Also, they wanted to transfer their programming skills to Mathematics, Turkish and Science courses, respectively.

Among the limitations of this study, it is significant to mention that the research is limited to data collected from secondary school fifth grade students. Also, the data on students' CT skills in the research process is limited to the analysis of the quantitative and qualitative data obtained during the application process and the application process is limited to a total of 28 course hours.

Recommendations are divided into two both for practitioners and for further research. First of all, curricula should provide examples that facilitate authentic learning. Teachers should create environments and employ the proper methods to promote learning by doing and allow students to put their knowledge into practice. Teachers and students should be provided with career planning activities at regular intervals. Teachers should adopt interdisciplinary approaches that appeal to learning outcomes. All educational institutions (from preschool to higher education) should offer CT activities to ensure lifelong learning. For the further research, action research should conduct and in-service training studies to address students' problems. Researchers should use programming languages other than Scratch to determine their impact on students' computational skills and performance. Researchers should develop more scales on CT dimensions and performance and skill tests.

Author notes: This paper was derived from the doctoral thesis of the first author. A part of this study was presented as a virtual presentation at the International Congress of Science Culture and Education, October 21-24, 2021.

Author contributions: All authors were involved in concept, design, collection of data, interpretation, writing, and critically revising the article. All authors approve final version of the article.

Funding: The authors received no financial support for the research and/or authorship of this article.

Acknowledgements: The study was supported by the Scientific Research Projects Coordinator (BAP) of Cukurova University Rectorate under project No: SDK-2019-11753.

Declaration of interest: Authors declare no competing interest.

Data availability: Data generated or analysed during this study are available from the authors on request.

REFERENCES

Alpar, R. (2014). *Applied statistics and validity-reliability with examples from sports, health and education sciences* (3rd ed.). Detay Publishing.

Apostolellis, P., Stewart, M., Frisina, C., & Kafura, D. (2014, June). *RaBit escApe: A Board game for computational thinking* [Paper presentation]. Interaction Design and Children Conference, Denmark. https://doi.org/10.1145/2593968.2610489

Başol, G. (2013). *Measurement and evaluation in education.* Pegem Academy.

Büyüköztürk, Ş., Çakmak, E. K., Akgün, Ö. E., Karadeniz, Ş., & Demirel, F. (2012). *Scientific research methods*. Pegem Academy

Calder, N. (2010). Using Scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom, 15*(4), 9-14.

Çatlak, Ş., Tekdal, M., & Baz F.Ç. (2015). The state of teaching programming with Scratch software: A document review study. *Journal of Instructional Technologies & Teacher Education*, *4*(3), 13-25.

Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2019). The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages. *Computer Science Education*, 29(1), 23-48. https://doi.org/10.1080/08993408.2018.1547564

Cohen, J. (1992). A coefficient of agreement for nominal scales. *Educational and Psychologica Measurement, 20*(1), 37-46 https://doi.org/10.1177/001316446002000104

Creswell, J. W. (2012). *Educational research planning, conducting, and evaluating quantitative and qualitative research* (4th ed.). Pearson.

Crues, R. W., Henricks, G. M., Perry, M., Bhat, S., Anderson, C. J., Shaik, N., & Angrave, L. (2018). How do gender, learning goals, and forum participation predict persistence in a computer science MOOC? ACM *Transactions on Computing Education*, *18*(4), 1-14. https://doi.org/10.1145/3152892

Cuny, J., Snyder, L., & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientists*. https://doi.org/10.1016/j.edurev.2017.09.003

Davidson, K., Larzon, L., & Ljunggren, K. (2010). *Self-efficacy in programming among STS students*. Technical Reports from Computer Science Education course of Upssala University. http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports/Self-Efficacy.pdf

Dr. Scratch. (2014). *Dr. Scratch: Analyze your Scratch project here*. http://drscratch.org/

Driscoll, M. (2002). *Web-based training*. John Wiley & Sons, Inc.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and earning of computer programming through the use of the Second Life virtual world. *British Journal of Educational Technology*, *42*(4), 624-637. https://doi.org/10.1111/j.1467-8535.2010.01056.x

ET2020. (2016). *Education and training 2020*. http://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:C:2009:119:0002:0010:EN:PDF

Fadjo, C. L. (2012). *Developing computational thinking through grounded embodied cognition* (Unpublished Doctoral dissertation, Columbia Üniversitesi, Newyork, USA).

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

Goel, S., & Kathuria, V. (2010). A novel approach for collaborative pair programming. *Journal of Information Technology Education*, *9*, 183-196. https://doi.org/10.28945/1290

Güneş, F. (2012). Developing students' thinking skills. *Turkology Studies, 32*(32), 127-146.

Hughes-Roberts, T., Brown, D., Standen, P., Desideri, L., Negrini, M., Rouame, A., Malavasi, M., Wager, G., & Hasson, C. (2019). Examining engagement and achievement in learners with individual needs through robotic-based teaching sessions. *British Journal of Educational Technology*, *50*(5), 2736-2750. https://doi.org/10.1111/bjet.12722

Hung, Y.-C. (2012). The effect of teaching methods and learning style on learning program design in web-based education systems. *Journal of Educational Computing Research*, *47*(4), 409-427. https://doi.org/10.2190/EC.47.4.d

Institute for the Future (2011). *"Future world skills 2020", Institute for the Future, Palo Alto, CA.*

International Society for Technology in Education. (2016). *National education technology standards for students*. ISTE. www.iste.org/standards/nets-for-students

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). The effects of mind mapping with cooperative learning on programing performance, problem solving skill and metacognitive knowledge among computer science students. *Journal of Educational Computing Research,* *42*(1), 35-61. https://doi.org/10.2190/EC.42.1.b

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), 1-7.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). *Embedding computational thinking in science, technology, engineering, and math (CT-STEM)*. Future directions in computer science education summit meeting, Orlando, FL. https://doi.org/10.1007/s10956-015-9581-5

Kahn, K., Sendova, E., Sacristán, A. I., & Noss, R. (2011). Young students exploring cardinality by constructing infinite processes. *Technology, Knowledge and Learning, 16*(1), 3-34. https://doi.org/10.1007/s10758-011-9175-0

Kalelioğlu, 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.

Kaminski, J. (2007). *Use ADDIE to design online courses*. Nursing-informatics.com. https://nursing-informatics.com/ADDIE.pdf

Karaahmetoğlu, K. (2019). *The effect of project-based arduino educational robot applications on students' computer thinking skills and perceptions of basic stem skill levels* (Master thesis), Amasya University Institute of Science and Technology, Amasya. https://doi.org/10.17275/per.19.8.6.2

Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing. *Journal of Educational Multimedia and Hypermedia, 21*(4), 371-391.

Kırçalı, A. Ç. (2019). *Valuation of computerized and non-computerized tools used in algorithm teaching at K12 level in terms of various variables (*Unpublished master thesis). Marmara University, Istanbul.

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. *Computers & Education*, *54*(1), 69-87. https://doi.org/10.1016/j.compedu.2009.07.012

Kose, U., Koc, D., & Yucesoy, S. A. (2013). Design and development of a sample ''computer programming'' course tool via story-based e-learning approach. *Educational Sciences in Theory and Practice, 13*(2), 1235-1250.

Kukul, V. (2018). *The effect of different structured processes in programming teaching on students' computational thinking skills, self-efficacy and programming success* (Unpublished doctoral dissertation). Gazi University, Ankara, Turkey.

Kurhila, J., & Vihavainen, A. (2015). A purposeful MOOC to alleviate insufficient CS education in Finnish schools. *ACM Transactions on Computing Education*, *15*(2), 1-18. https://doi.org/10.1145/2716314

Kyungbin, K., & Jonassen, D. H. (2011). The influence of reflective self-explanations on problem-solving performance. *Journal of Educational Computing Research*, *44*(3), 247-263. https://doi.org/10.2190/EC.44.3.a

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*, 159-174. https://doi.org/10.2307/2529310

Lin, J. M. C., & Liu, S. F. (2012). An investigation into parent-child collaboration in learning computer programming. *Educational Technology & Society*, *15*(1), 162-173.

Lockwood, J., & Mooney, A. (2017). Computational thinking in education: Where does it fit? A systematic literary review. https://arxiv.org/abs/1703.07659

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM Bulletin, 39*(1), 223-227. https://doi.org/10.1145/1227310.1227388

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, ITiCSE-WGR 2014, 1–29. ACM, New York. https://doi.org/10.1145/2713609.2713610

Maykut, P., & Morehouse, R. (1997). Beginning qualitative research: A philosophic and practical guide. Falmer Press.

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

McGriff, S. J. (2000). *Instructional system design (ISD): Using the ADDIE model.* https://www.lib.purdue.edu/sites/default/files/directory/butler38/ADDIE.pdf

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010). Learning computer science concepts with scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)* ((pp. 69-76). New York. USA. https://doi.org/10.1145/1839594.1839607

Ministry of National Education of Turkey. (2018a). *Information technologies and software course curriculum*.

Ministry of National Education of Turkey. (2018b). *Computer science course curriculum set1 - set2*. http://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=335

Missiroli, M., Russo, D., & Ciancarini, P. (2017). Cooperative thinking, or: computational thinking meets agile. *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)* (pp. 187-191). GA. https://doi.org/10.1109/CSEET.2017.37

Moreno-Leon, J., & Robles, G. (2012). Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In *Scratch Conference* (pp. 12-15). https://doi.org/10.1145/2818314.2818338

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' Scratch skills with their computational thinking skills in terms of different variables. *International Journal of Modern Education and Computer Science*, *8*(11), 1-7. https://doi.org/10.5815/ijmecs.2016.11.01

Özcan, S. N., Ergün, K., Özge, K. & Neriman, E. 2017). High school students' views on the use of Scratch program in computer programming education. *International Journal of Scientific Research*, 182-188.

Özel, O. (2019). *The effect of programming methods on secondary school students' computational thinking skills, self-efficacy perception and programming success* (Doctoral dissertation), Marmara University, Turkey.

Patton, M. Q. (2002). *Qualitative evaluation and research methods* (3rd ed.). Sage Publications.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 171-175). https://doi.org/10.1145/1007996.1008042

Riley, D. D., & Hunt, K. A. (2014). *Computational thinking for the modern problem solver*. CRC Press. https://doi.org/10.1201/b16688

Robertson, J. (2013). The influence of a game-making project on male and female learners' attitudes to computing. *Computer Science Education,* *23*(1), 58-83. https://doi.org/10.1080/08993408.2013.774155

Sakamoto, K., Takano, K., Washizaki, H., & Fukazawa, Y. (2013). Learning system for computational thinking using appealing user interface with icon-based programming language on smartphones. In *Proceedings of the 21st International Conference on Computers in Education, ICCE*.

Shin, S., & Park, P. (2014). A study on the effect affecting problem solving ability of primary students through the Scratch programming. *Advanced Science and Technology Letters, 59*, 117-120. https://doi.org/10.14257/astl.2014.59.27

Uslu, N. A. (2018). The effect of visual programming activities on the computational thinking skills of secondary school students. *Aegean Journal of Educational Technologies*, *2*(1), 19-31.

Viera, A. J., & Garrett, J. M. (2005). Understanding interobserver agreement: The kappa statistic. *Family Medicine*, *37*(5), 360-363.

Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). K 6 computational thinking curriculum framework-implications for teacher knowledge. *Educational Technology & Society*, *19*(3), 47-57.

Wang, L. C., & Chen, M. P. (2010). The effects of game strategy and Preference matching on flow experience and programming performance in game-based learning. *Innovations in Education and Teaching International, 47*(1), 39-52. https://doi.org/10.1080/14703290903525838

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education*, 59(2), 412-422. https://doi.org/10.1016/j.compedu.2012.01.007

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In *Proceedings of the first international workshop on Computing education research* (pp. 13-24). https://doi.org/10.1145/1089786.1089788

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

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, *14*(1), 1-16. https://doi.org/10.1145/2576872

Yünkül, E., Durak, G., Çankaya, S., & Abidin, Z. (2017). The effects of Scratch software on students' computational thinking skills. *Necatibey Education Faculty Electronic Journal of Science and Mathematics Education, 11*(2), 502-517. https://doi.org/10.1007/s10758-018-9391-y

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K9. *Computers & Education*, *141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Correspondence: Nihan Arslan Namli, Department of Computer Technologies, Dortyol Vocational School of Higher Education, Iskenderun Technical University, Turkey. E-mail: nihan.arslannamli@iste.edu.tr