International Education Studies; Vol. 10, No. 8; 2017
ISSN 1913-9020 E-ISSN 1913-9039
Published by Canadian Center of Science and Education

An Analysis of Mathematics Education Students' Skills in the Process of Programming and Their Practices of Integrating It into Their Teaching

Semirhan Gökçe[1], Arzu Aydoğan Yenmez[2] & İlknur Özpınar[2]

[1] Department of Computer Education and Instructional Technology, Ömer Halisdemir University, Niğde, Turkey

[2] Department of Mathematics and Science Education, Ömer Halisdemir University, Niğde, Turkey

Correspondence: Semirhan Gökçe, Department of Computer Education and Instructional Technology, Ömer Halisdemir University, Niğde, Turkey. Tel: 90-388-225-4402. E-mail: semirhan@gmail.com

Received: February 22, 2017 Accepted: March 23, 2017 Online Published: July 29, 2017

doi:10.5539/ies.v10n8p60 URL: https://doi.org/10.5539/ies.v10n8p60

Abstract

Recent developments in technology have changed the learner's profile and the learning outcomes. Today, with the emergence of higher-order thinking skills and computer literacy skills, teaching through traditional methods is likely to fail to achieve the learning outcomes. That is why; teachers and teacher candidates are expected to have computer literacy skills. Programming is the main focus of this study since it is an important part of computer literacy. The study aims to analyze mathematics education students' skills in the process of programming and their practices of integrating it into their teaching. The participants of the study are 42 third grade students of an Elementary Mathematics Education Program of a state university in Turkey. Within the study in which theory and practice was carried out simultaneously, the participants were taught the basics of programming and the algorithms with C programming language. The teacher candidates put the theoretical knowledge into practice using the visual programming application by MIT App Inventor at the computer laboratory. In addition, they used the MIT App Inventor visual programming environment to develop programs they will use in teaching mathematics in groups. Given the component of teaching of programming during this process, it is considered that the process of teaching in question will be effective in planning the teaching process of future studies. The reason is that not only it analyses the development of the variables used in this study but also because it takes into consideration the opinions of teacher candidates.

Keywords: computer literacy, programming, skills, mathematics education, teacher candidate

1. Introduction

With the progress of technology, dramatic changes have occurred in every sphere of life, which have changed the ways in which people access knowledge and communicate with one another. Our age, in which the increase in the number and the variety of technological tools is directly proportional to the increase in human population, we are witnessing an increase in the production of knowledge, which has resulted in our age being called the "information age". As the technology reshapes our lives rapidly, the world of education has also been influenced by these changes, with the use of technology in education becoming more and more common. We started having technological tools within the classrooms a few years ago but now they have become teaching methods. In addition, technological advances in numerous fields from health to education, and from agriculture to industry, have changed the student profile and the learning outcomes. In a developing and changing world, it is necessary to adapt to the developing technology, and thus the society needs individuals who keep themselves up-to-date with technological developments and solve problems with the help of technology. Individuals are expected to acquire basic computer skills and put them into practice in daily life. With the higher order thinking skills and the computer literacy becoming more and more important, educating students in traditional methods will not meet the learning outcomes (O'Flaherty & Philips, 2015; Roehl, Reddy, & Shannon, 2013; Vaughan, 2014).

Without any doubt, one of the most important factors improving the quality of education is the teachers. Teachers need to use technology in a more efficient way to enhance instruction rather than only running it (Algozzine, Bateman, Flowers, Gretes, Hughes, & Lambert, 1999; Krueger, Hansen, & Smaldino, 2000). Therefore, teachers and teacher candidates are expected to possess adequate knowledge and skills in computer literacy (Akkoyunlu

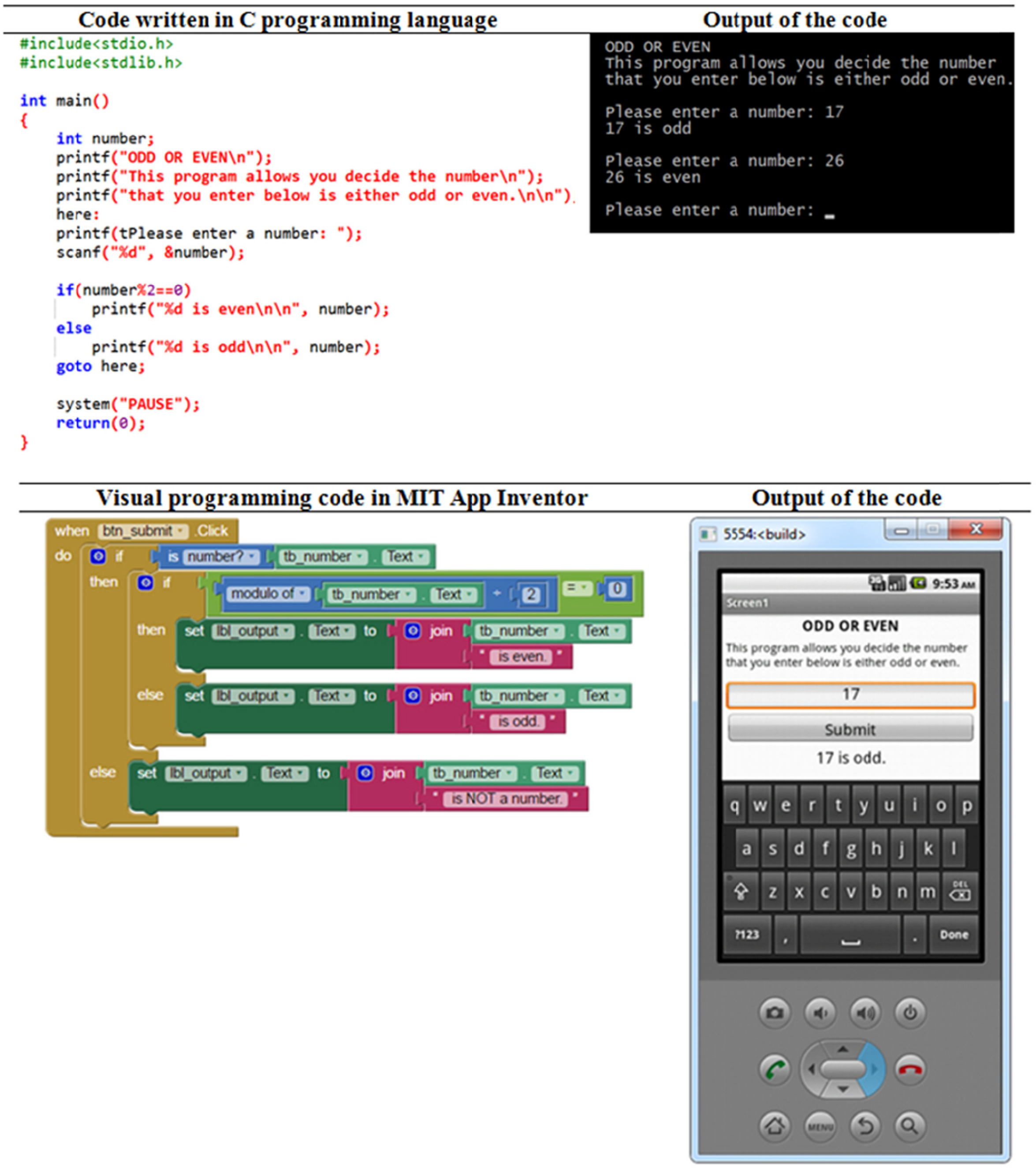
& Kurbanoğlu, 2003). Computer literacy is divided into four components (Kay, 1990), which are basic computer skills, computer awareness (Anderson & Klassen, 1981, Battista & Steele, 1984, Johnson, Anderson, Hansen & Klassen, 1980), software use (Ganske and Hamamoto, 1984; Hasset, 1984; Levin, 1983; Meierhenry, 1982; Pickert & Hunter, 1983) and programming skills (Cheng, Plake, & Stevens, 1985; Gabriel, 1985a, 1985b; Haigh, 1985; Luehrmann, 1981). Today, since technology supported teaching is one of the indispensable part of education, the teachers of the future should have more computer literacy skills than programming (Kılınç & Salman, 2006). So, more attention should be paid to courses in educational software and programming in teacher training programs as these concepts will become even more important in the future (Kılınç & Salman, 2006).

Programming is considered to be a cognitive skill as it involves such processes as mental descriptions, understanding of the program, making changes on a program already written, debugging, structuring conceptual knowledge and performing single operations (such as loops, conditionals) (Helminen & Malmi, 2010). While programming, a learner creates a mental model by re-expressing the purpose, the data and the function of the program as she/he understands it while solving a problem, and uses this model to make predictions and formulate the differences (Helminen & Malmi, 2010). In programming, the coder creates an external representation of problem solving processes, and so programming provides a person with an opportunity to apply a lot of deep thinking (metacognitive awareness) about their own thinking (Resnick, et al.,, 2009). Moreover, acquiring skills associated with programming helps students develop higher order thinking skills including reasoning, critical thinking and creative thinking (McMahon, 2009). Critical thinking, a skill developed via programming, includes the cognitive skills related to interpretation, analysis, evaluation, inference, explanation and self-regulation. These six cognitive skills are at the core of critical thinking (Facione, 1990). One of the skills employed in the learning of a programming language is problem solving (Gundurao, Manjunath, & Nachappa, 2010). One needs to develop good problem solving skills in order to learn how to write a successful computer program (Gundurao, Manjunath, & Nachappa, 2010). Gomes and Mendes (2007) also state that it is necessary for learners to develop good problem solving skills in order to learn to program. Another skill needed in problem solving is metacognition (Guss & Wiley, 2007). Metacognitive awareness helps one to have control or self-regulation on thinking, learning process and outcomes (Hartman, 1998). Metacognition in problem solving refers to processes and knowledge used for approaches towards successful problem solving (McCormick, 2003). In conclusion, for programming, one can make use of metacognition and metacognitive awareness used particularly during regulation and evaluation of solutions and outcomes. Students have various difficulties during the programming process, which demands many skills. As students pay too much attention to design rather than codes, to one single function or module rather than the system as a whole while writing a program (Berge, Borge, Fjuk, Kaasboll, & Samuelsen, 2003; Bucci, Long, & Weide, 2001), they experience low levels of abstraction, and hence get lost in too many details and fail to see the big picture. Learners' disposition to think holistically or analytically while problem solving will shape the process of programming. Holistic thinkers do not focus on the parts of an object but take the object as a whole, and make decisions taking into consideration the effect of the interrelation of the parts on the whole (Dewey, 2007; Hammouri, 2003). On the other hand, analytical thinkers always deal with the individual parts of an object in the first place and arrive at decisions after analysing how the parts interact to keep the system functioning (Dewey, 2007). Since programming is known to be useful for learners to improve their skills (Sleeman, Putnam, Baxter, & Kuspa, 1984), many countries have integrated introductory programming courses into their primary and elementary school curricula (Tucker, Deek, Jones, McCowan, Stephenson, & Verno, 2003). At the same time, it is also worth noting that programming courses are becoming widespread throughout the world because not only they are compulsory for professional competence but also they are an integral component of computer literacy. This study has focused on the skills that programming helps learners to acquire and on the fact that it is an important component of computer literacy. The main purpose of this study is to analyse the mathematics education students' skills in learning to program, which are critical thinking, problem solving, holistic and analytical thinking in problem solving, and metacognitive awareness, and their practices of integrating it into their teaching. The method of the study is included in the following section.

2. Methodology

2.1 Participants and Learning Environment

The one group pre-test post-test design was used in this study. The participants of this study are 42 third grade mathematics education students at a state university. They took a 14-week elective course in which they were taught the algorithms and the basics of programming via the C programming language, and put what they learnt into practice using MIT App Inventor environment during their laboratory hours. Also, they developed the programs they might use in their mathematics teaching using MIT App Inventor environment in groups. Figure 1 given below contains the screen shots to exemplify the learning environment where the theory and practice go hand in hand.



Code written in C programming language

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int number;
    printf("ODD OR EVEN\n");
    printf("This program allows you decide the number\n");
    printf("that you enter below is either odd or even.\n\n").
    here:
    printf(tPlease enter a number: ");
    scanf("%d", &number);

    if(number%2==0)
        printf("%d is even\n\n", number);
    else
        printf("%d is odd\n\n", number);
    goto here;

    system("PAUSE");
    return(0);
}
```

Output of the code

```
ODD OR EVEN
This program allows you decide the number
that you enter below is either odd or even.

Please enter a number: 17
17 is odd

Please enter a number: 26
26 is even

Please enter a number: _
```

Visual programming code in MIT App Inventor

```
when btn_submit .Click
do  if  is number? tb_number . Text
    then  if  modulo of tb_number . Text ÷ 2  = 0
          then  set lbl_output . Text to  join  tb_number . Text
                                                " is even. "
          else  set lbl_output . Text to  join  tb_number . Text
                                                " is odd. "
    else  set lbl_output . Text to  join  tb_number . Text
                                          " is NOT a number. "
```

Output of the code

5554:<build>

9:53 AM

Screen1

ODD OR EVEN

This program allows you decide the number that you enter below is either odd or even.

17

Submit

17 is odd.

Figure 1. A scenario dealt with using the C programming language and an application developed using MIT App Inventor

2.2 Data Collection Methods and Data Analysis

The research analyzed the effect of learning process on analytical and holistic thinking for problem solving, critical thinking, problem solving skills and metacognitive awareness, and on the effectiveness of the practices the teacher candidates developed. The participants made self-evaluation about their program developing process and were asked to respond to a written questionnaire to reflect the learning process in addition to a semi-structured interview.

This research employed the "Scale of Holistic and Analytical Thinking Styles While Problem Solving" developed by Umay and Arıol (2011) to determine the participants' holistic and analytical thinking styles. The "Scale of Holistic and Analytical Thinking Styles While Problem Solving" contains items each of which has two separate scenarios suitable for either holistic or analytical thinking, which can be adopted by the participants. There is an option described as "I don't have an opinion" for those who might not recognize how they think. Each item of the scale gives participants a score for their responses: 1 for "analytical thinking style", 2 for "I don't have an opinion", and 3 for "the holistic thinking style". Out of the five items in the scale, the lowest possible score is 5, and the highest possible score is 15. A score closer to 5 refers to analytical thinking disposition while a score closer to 15 refers to holistic thinking disposition. The scale has a reliability coefficient of .72.

The researchers used the California Critical Thinking Disposition Inventory, which was developed as a result of the Delphi Research Report in 1990 and adapted to Turkish by Kökdemir (2003). Cronbach's alpha internal consistency coefficient of the scale was calculated to be .77. The inventory is a 6-point Likert scale consisting of 51 items. The items use a 6-point scale from 1 to 6 in which 1 refers to "I do not agree at all", and 6 refers to "I totally agree". The Critical Thinking Inventory has six factors, which are analyticity, open-mindedness, curiosity, confidence, truth seeking and systematicity (Kökdemir, 2003). To find the participants' overall scores for the critical thinking disposition, the scores given to students based on their responses to the items listed in each factor were summed up, and the results were divided by the total number of items, and multiplied by 10, so that the minimum value of each sub-scale is 10 and maximum value is 60 and the range for the total scale is from 60 to 360. The participants scoring less than 240 are deemed to have lower critical thinking disposition while those scoring more than 300 are deemed to have higher critical thinking disposition (Kökdemir, 2003).

For the problem solving component, the researchers used the Problem Solving Inventory developed by Heppner and Peterson (1982), and adapted to Turkish by Şahin, Şahin and Heppner (1993). The internal consistency coefficient of the scale was calculated to be .81. The Problem Solving Inventory is a 6-point scale consisting of 35 items. The items on the scale range from 1 to 6, with respective responses being "strongly disagree" and "strongly agree". In the calculating the participants' total scores from the scale, their responses to items 9, 22 and 29 are left out from the calculation as per the protocol, and items 1, 2, 3, 4, 11, 13, 14, 15, 17, 21, 25, 26, 30 and 34 were reverse-scored. Following these processes, the total score is calculated by summing up the scores given based on the responses to individual items. The lowest possible score is 32, and the highest possible score is 192. The lower the score, the poorer problem solving skills a participant has, while the higher the score, the better problem solving skills a participant has.

For metacognitive awareness, the researchers employed the Metacognitive Awareness Inventory, developed by Schraw and Dennison (1994), and adapted to Turkish by Akın, Abacı, and Çetin (2007) after having administered it with undergraduate students. The reliability coefficient of the scale was calculated to be .87. The Metacognitive Awareness Inventory is a 5-point Likert scale consisting of 52 items. The items can be answered within the range of "never" and "always" and get scores from 1 to 5. The scale has eight factors, which are declarative knowledge, procedural knowledge, planning, monitoring, evaluation, debugging and information management (Akın et al., 2007). The scale does not contain any negative items that need reverse scoring. Each item is calculated with the score based on the response, and scores for the responses to the items are summed up. 52 is the lowest possible score while 260 is the highest possible score. Lower scores mean poorer metacognitive awareness whereas higher scores demonstrate better metacognitive awareness (Akın et al., 2007).

Three experts, two in mathematics education and one in computer education and instructional technology, assessed the effectiveness of the programs developed. Büyüköztürk (2008) emphasized the researches containing 15 or more sample size in each subgroup indicating that the use of parametric tests does not lead to a significant deviation in the significance level. However, parametric tests can only be used when the data follow a normal distribution. For this reason, in the first place, the data were tested for normal distribution in order to figure out whether parametric tests can be used while analysing the quantitative data. To this end, participants' critical thinking disposition, problem solving skills and metacognitive awareness levels were tested for normal distribution.

Qualitative data analysis of the research was performed by using content analysis method. Responses to the questions were recorded in the course of the interviews, and the data obtained from the interviews were itemised and checked before the analysis. The coding method was used to cluster together the similar data around certain themes, and to create consistent and coherent subdivisions. The three experts individually coded the raw data from the research to achieve the reliability of the process, and the results showed that 83% of inter-rater dependability was reached.

The outputs of study include the programs developed by the teacher candidates for the learning outcomes defined in the 5th-8th grade (11-14 years old students) national mathematics curriculum, the teacher candidates' self-evaluation statements on the programs and their evaluation of the teaching process. The self-evaluation statements include the pros and cons of the process, its contributions to their own learning and teaching competencies, their opinion on whether it will help them develop an application using a programming language as an in-class or out-of-class activities, and possible advantages and disadvantages for students in mathematics education.

3. Findings

The findings of this study, which investigates the mathematics education students' skills while learning to program–critical thinking, problem solving, holistic and analytical thinking styles while problem solving and metacognitive awareness–and the practices of integrating it into their teaching, are presented with the comparison of pre-test and post-test scores, which is followed by their evaluation of the programs they developed and qualitative findings.

3.1 Comparison of Problem Solving Inventory Scores from Pre-Test and Post-Test

A paired-sampled t-test was performed to determine whether there is a statistically significant difference between the participants' problem solving skills scores from pre-test and post-test. The results of the t-test are presented in Table 1.

Table 1. T-test results for the average scores of the pre-test and post-test in Problem Solving Scale

Problem Solving Scale	N	$\overline{X}$	S	sd	T	p
Pre-test	42	138.29	17.84	41	1.28	.209
Post-test	42	141.01	16.77			

As can be seen in Table 1, the participants' average score from the pre-test of the problem solving inventory is 138.29, and it became 141.01 in the post-test. According to the inventory, higher scores meant better problem solving skills; and thus it can be said that the participants had higher levels of problem solving skills both before and after they learned how to program. It has been found that there is not a statistically significant difference in the participants' problem solving skills before and after the learning process, so learning to program does not have a significant effect on problem solving skills [$t_{(41)=}$ 1.28, p>.05].

3.2 Comparison of the Scale of Holistic and Analytical Thinking Styles While Problem Solving Scores from Pre-Test and Post-Test

The participants' average pre-test scores (8.81) suggest that they have more disposition to think analytically; while the average score (11.93) they got from the post-test indicates that the group has more disposition to think holistically. A Wilcoxon signed-rank test was carried out to determine whether the observed difference is statistically significant or not, which results are presented in Table 2. The results of the analysis suggest that there is a statistically significant difference between the scores the participants got from the Scale of Holistic and Analytical Thinking Styles While Problem Solving before and they went through the learning process (z= 4.99, p<.05). Considering the mean rank and totals of the difference scores, it can be said that this difference observed is in favour of the positive ranks, that is to say, in favour of the post-test. It has been shown that the process of learning to program has a statistically significant effect on the holistic and analytical thinking styles while problem solving in favour of holistic thinking disposition.

Table 2. Wilcoxon Signed-Rank Test results of the scores from the Scale of Holistic and Analytical Thinking Styles during problem solving

Post-test - Pre-test	N	Mean Rank	Total Rank	Z	p
Negative Rank	4	9.88	39.50	4.99*	.000
Positive Rank	36	21.68	780.50		
Equal	2	-	-		

**Based on negative ranks.*

3.3 Comparison of California Critical Thinking Disposition Scores from Pre-Test and Post-Test

A paired-samples t-test was used to determine if there was a statistically significant difference between the participants' critical thinking scores from pre-test and post-test. Results of the t-test are presented in Table 3.

Table 3. T-test results of pre-test and post-test scores in Critical Thinking Scale

Critical Thinking Scale	N	$\overline{X}$	S	sd	T	p
Pre-test	42	241.01	28.86	41	7.01	.000
Post-test	42	277.27	36.38			

As can be seen in Table 3, the participants' average score from critical thinking disposition scale before they went through the learning process was 241.01, and the score changed to 277.27 after the learning took place. This suggests that students had average levels of critical thinking disposition both before and after the learning process. The teacher candidates' critical thinking disposition scores changed in a significant way after the learning took place, which means that the learning process has a significant effect on the critical thinking disposition [$t_{(41)=}$ 7.01, p<.05].

3.4 Comparison of Metacognitive Awareness Inventory Scores from Pre-Test and Post-Test

A paired-samples t-test was used to determine whether there is a statistically significant difference between the participants' scores on metacognitive awareness levels from pre-test and those from post-test. Results of the t-test are presented in Table 4.

Table 4. T-test results of the pre-test and post-test scores from Metacognitive Awareness Inventory

Metacognitive Awareness Inventory	N	$\overline{X}$	S	sd	T	p
Pre-test	42	178.31	22.91	41	5.11	.000
Post-test	42	195.40	26.21			

As can be seen in Table 4, the average score of metacognitive awareness inventory before the learning process was 178.31, and this score increased to 195.40 after the learning process. These values indicated that the participants had an average level of metacognitive awareness before learning to program while they started having a high level of metacognitive awareness after learning to program. It has been found that there is a significant difference between the teacher candidates' levels of metacognitive awareness before and after they learned to program; that is to say, the process of learning to program had a significant effect on the metacognitive awareness level [$t_{(41)=}$ 5.11, p<.05].

3.5 Evaluation of the Programs Developed

The outputs of the study contains the programs developed by the teacher candidates in line with the learning outcomes defined in the national mathematics curriculum of the 5th-8th grades (11-14 years of age) whom they will be teaching in the future. The programs developed were evaluated according to their convenience to the criteria established within the themes of Educational Content, Design, and Programming. The evaluation criteria were prepared by three experts, two of whom work in mathematics education and one of whom works in computer education and instructional technology. Content validity of the criteria was tested by six experts. The evaluation criteria were established based on a literature review (İpek, 2001; Şahin & Yıldırım, 1999; Uşun, 2000; Venezky & Osin, 1991; Yalın, 2001). Randomly selected three programs were scored by three experts based on the evaluation criteria and the inter-rater dependability was found to be 85%. The evaluation criteria and their themes are presented in Table 5.

Table 5. The evaluation criteria and the themes

THEME	CRITERION The developed program...	Totally unsuitable (0 point)	Partly unsuitable (1 point)	Partly suitable (2 points)	Very suitable (3 points)
EDUCATIONAL CONTENT	is prepared in line with the learning outcomes defined in the mathematics curriculum.				
	is suitable for students' development level.				
	contributes to meaningful learning for students.				
	is organized from simple to complex.				
	causes difficulty with mathematical sense-making for students.				
	provides motivational feedback to the students.				
EDUCATIONAL CONTENT & DESIGN	uses a comprehensible language.				
	contains motivational elements (audios, visuals, videos, animations and simulations etc.) for different learning needs of students.				
	has interesting screen designs supporting the teaching process.				
	highlights important points.				
DESIGN	contains informative instructions.				
	has clear, explicit and comprehensive guiding menus.				
	has visually consistent screens.				
	has no barriers (such as font type, size and color) on readability.				
DESIGN & PROGRAMMING	can function in different platforms (smartphone, tablet etc.).				
	can easily switch between displays.				
	provides an effective help page for users.				
	has buttons for switching between displays (forward, back and home).				
	has an interactive platform.				
PROGRAMMING	contains proper coding structures (variables, conditional statements, loops and arrays etc.).				
	contains the shortest and the simplest coding structures.				
	in general works without any compilation problem.				

Teacher candidates worked in groups of four (two of the groups contained five members) and developed 10 programs in total. The average scores given to each program by the three experts based on the evaluation themes are presented in Table 6.

Table 6. Average of the scores given by the three experts

Themes	Programs Developed									
	1	2	3	4	5	6	7	8	9	10
Educational Content	15.33	12.00	12.33	10.00	16.67	15.67	13.33	10.67	11.33	10.67
Educational Content & Design	9.67	6.67	9.33	8.67	10.00	9.67	7.67	8.33	10.67	9.00
Design	8.33	6.00	10.67	6.33	10.33	10.00	7.00	6.67	8.33	8.67
Design & Programming	10.67	9.00	13.00	8.00	14.00	13.67	8.33	7.67	11.67	10.33
Programming	6.00	2.67*	5.67	2.00*	8.33	5.33	2.67*	3.33*	7.00	6.00
Total average score	50.00	36.34	51.00	35.00	59.33	54.34	39.00	36.67	49.00	44.67
Percentage	76%	55%	77%	53%	90%	82%	59%	56%	74%	68%

The efficiency of the programs was criticized based on whether they got 50% of the total scores for each theme. Four of the programs (scores containing asterisk *) developed by the participants have problems in the programming theme. In such cases, the groups were able to address and resolve the problems according to the feedback provided by the experts.

3.6 Comments on the Program Development Process

The **t**eacher **c**andidates' (TC) comments relating to the evaluation of the teaching process included the pros and cons of the process, its contributions to their own learning and teaching competencies, their opinion on whether it will help them develop an application using a programming language as an in-class or out-of-class activity, and possible advantages and disadvantages for students in mathematics education.

Commenting on the pros and cons of the process, 31 teacher candidates remarked that they found it favourable to have the theory and practice simultaneously, as they thought that they could learn best by experience. Below are comments from teacher candidates from the interviews to support the statement in the previous sentence.

> *"[...] It's not so easy to grasp the programming logic, but we were able to learn it in a logical way for we went through the theory and practice at the same time [...] I think if students are made to take programming course, they should definitely be taught the theory and practice at the same time." (TC13)*
>
> *"[...] When we first saw the codes in the C programming language, it was really hard to make sense of anything, but as we practised it with many trials and errors, it all became more tangible [...]" (TC42)*

Some of the teacher candidates (TC5, TC11, TC22, TC25, TC33 and TC41) pointed that language was one of the barriers that caused difficulty for them as the commands in the C programming language as well as the MIT App Inventor environment were in English, with their English being not very good. However, they further remarked that this was a problem in the beginning, and this barrier vanished in the course of time as they became more knowledgeable and learned. Below is one such remark.

> *"[...] I had difficulty in the beginning with everything being in English. My classmates who were good at English were also better at remembering the codes, and using the menus and buttons of the MIT-Inventor. But later, I started getting used it as well, and in the end, it actually didn't matter to know English or not [...]" (TC5)*

The teacher candidates also stated that the drag and drop interface feature of the MIT App Inventor visual programming environment made programming more comprehensive as well as making it easier to concentrate on programming (TC1, TC4, TC10, TC14, TC20, TC27, TC31 and TC36). In addition, they found it to be an advantage to be able to see what effects were caused as a result of the changes they made on the MIT App Inventor. This is evident in the comments below from the teacher candidates' interviews.

> *"[...] First I was afraid that I was not going to be able to remember and write all these codes. But then I calmed down when we got to the laboratory right after the theoretical course as I learnt that the codes were already there on the MIT App Inventor environment, and all we needed to do was just to combine them logically [...]" (TC10)*
>
> *"[...] I think the best thing about it was that we could see the effects of the changes we made on the codes on the MIT App Inventor though the "Emulator" feature. Thanks to it, we can first check and then figure out how we can do [...]" (TC27)*

Among all the teacher candidates, 18 of them counted a number of advantages of working in groups including giving feedback to each other, restore the motivation of those group members who became demotivated by their first programming experiences, and helping those group members who did not believe that they would accomplish programming due to being bad with computers to keep on and achieve. Below are three of such remarks.

> *"[...] Developing program together was actually very instructive. It is because of the fact that we all tried different things and made different errors. For this reason, we were able to give each other really constructive and useful feedback, such as "This code should definitely work", or "This code has these and those problems" [...]" (TC8)*

> *"[...] It was easy for me to get demotivated by my earlier failed attempts at coding a program. However, as we started working as a group, everybody in the group felt like we could do anything, so long as we discuss things. So such a good atmosphere helped me avoid demotivation [...]" (TC32)*

> *"[...] I'm not a student who's got the best scores from the Information Technologies (IT) courses. So I thought I was never going to be able to make program, which was actually evident in my first attempts [...] but I never felt the same when we working as a group. I could keep on feeling better when other group members confirmed that they had similar difficulties with me" (TC35)*

On the other hand, some of the teacher candidates said that they did not have any difficulties during the process as they were familiar with the algorithmic thinking (TC3, TC6, TC12, TC13, TC21, TC26, TC30, TC34 and TC37). They think that designing algorithms is analogous to proving mathematical theorems, in which they use the variables step by step to design, and then verify the algorithms. One such comment is presented below.

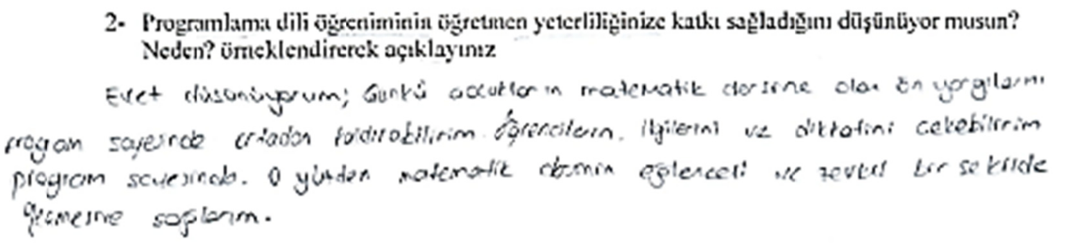
> *"[...] Well, I'm familiar with assigning variables to a loop. Indeed, we somehow use the same logic (that of programming) in our (mathematical) proofs. The only difference is that we need to have some more considerations to make the computer understand it [...] Shortly, we instruct the codes to behave in a certain way for all possible values a variable can have ("do this if…", "do that if…"). We need to make sure that there is no variable left out, and then we look at how it functions as a whole. On the other hand, we follow a deductive process to design. It's like verification of theorems in mathematical proofs starting with hypotheses." (TC21)*

While most of the participants stated that they were able to integrate technology into mathematics teaching, they added that they now understand what the computer does while it compiles the codes of the program and what is going on in the memory, so they felt themselves more competent and knowledgeable. They believed that it has helped to develop new competencies not only for their teaching but also for their learning. They felt that today it is important to know programming and what runs in the infrastructures of applications as it provides them with a new perspective that has a positive effect on their logical thinking, creativeness, and problem solving methods. Moreover, in terms of teaching competencies, the programs they developed provided them with effective teaching full of fun as well as a new evaluation and assessment instrument. This is obvious from the remarks of the participants in the written interviews (see Figure 2) and semi-structured interviews.

> *"[...] Now I see the applications around through a totally different perspective: I can't help making such comments about an application as "I guess this application would be better with these and those additions". Also, I think I've become more competent in computers [...] I definitely think that it improves my creativeness. Now I can do things I never thought I could do. The algorithms we design… They also improve my logical thinking. It seems like I can solve any problems" (TC15)*

> *"[...] Thanks to programming, now I know how I can make a subject matter more fun, conceptual and easier. I am more knowledgeable now about how to develop a new instrument or application [...] Now, I can develop an application that helps students for self-evaluation. " (TC17)*

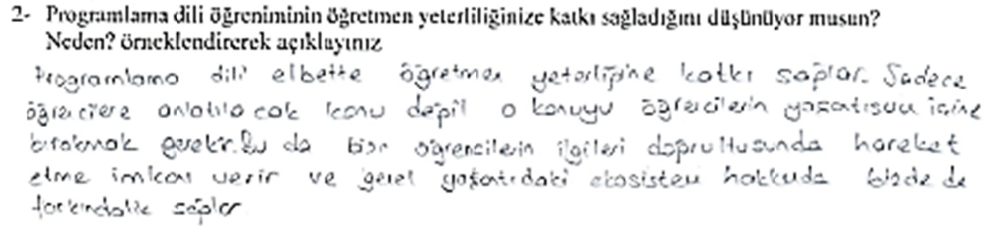
> *"[...] With the application we developed, it's possible to have a long-lasting teaching practice with fun, which can also provide instant feedback for students at home." (TC38)*



2- Programlama dili öğreniminin öğretmen yeterliliğinize katkı sağladığını düşünüyor musun? Neden? örneklendirerek açıklayınız

Evet düşünüyorum; Çünkü çocukların matematik dersine olan ön yargılarını program sayesinde ortadan kaldırabilirim. Öğrencilerin. İlgilerini ve dikkatini çekebilirim program sayesinde. O yüzden matematik dersimin eğlenceli ve zevkli bir şekilde geçmesine sağlarım.

"Yes, I think it contributes to my teaching competence, by this way with the programs I can eliminate the prejudices of students to mathematics. I can attract the attention and interest of the students with the programs. I will make my lessons enjoyable and funny with the programs." (TC9)



2- Programlama dili öğreniminin öğretmen yeterliliğinize katkı sağladığını düşünüyor musun? Neden? örneklendirerek açıklayınız

Programlama dili elbette öğretmen yeterliğine katkı sağlar. Sadece öğrencilere anlatılacak konu değil o konuyu öğrencilerin yaşantısının içine bırakmak gerekir. Bu da bize öğrencilerin ilgileri doğrultusunda hareket etme imkanı verir ve genel yaşantıdaki ekosistem hakkında bizde de farkındalık sağlar

"Of course, learning how to program has positive contributions to my teaching competencies. I can leave the mathematics into students' life with the programs rather than only teaching as a concept. This give us a chance to act with the interests of students and it allows us awareness of students' lives." (TC24)

Figure 2. Responses by two participants (TC9 and TC24) to an item in the written interview about the contribution of learning a programming language to their teaching competencies

When the teacher candidates were asked whether they would give an opportunity for their students to develop an application though programming languages as an in-class or out-of-class learning activity, 12 teacher candidates thought that their students might have difficulty in such a case considering their students' (11-14 years old) profile. On the other hand, when the teacher candidates were asked to present their opinion about application development by using programming languages, six participants commented on possible pros and cons without specifying whether they will provide their students with that opportunity or not. On the other hand, 24 teacher candidates stated that now children grow up with technology in a digital age, so they will not have any big problems learning to program. They also believed that students can be taught it through a simpler structure, or they can work with interested students who will be involved in voluntary work in programming. According to the participants, among possible pros and cons of using programming languages to develop an application for students were in general helping students improve their competencies and higher order thinking skills, building up their confidence about technological competence; however, if it was not taught them through a simpler structure, this might lead to a loss of motivation for technology. This is obvious from the participants' responses in the semi-structured interviews presented below.

> *"[…] This way, students can acquire many skills that will help them with many things in life. They can learn to think more logically and systematically. They will feel they are more proficient in computers, so they will be more confident. But if they think this work is too difficult, then they may feel they are incompetent." (TC2)*
>
> *"[…] It was really difficult even for us, and when I think about their younger age, I think it will be even more difficult for them. I don't think it is appropriate for their age […]" (TC40)*

4. Discussion and Conclusion

The teaching process is highly important to help the teachers of the future to graduate as computer-literate individuals. Considering the component of teaching programming in this process, acquiring programming-related skills is known to help students acquire higher order thinking skills (McMahon, 2009). This study investigated the effect of the process of learning to program on teacher candidates' problem solving skills, analytical and holistic thinking disposition while problem solving, critical thinking disposition and metacognitive awareness. The results of the study showed that learning to program did not have a significant effect on teacher candidates' problem solving skills. There was no significant difference between the participants' pre-test and post-test scores of the Problem Solving Scale as they had good problem solving skills, and this can be explained with their field of study. It was also found that there was a significant difference in participants' scores from Analytical and Holistic Thinking Disposition Scale before and after the learning took place, which means that learning to program had a significant effect on their holistic thinking disposition. It is known that learners miss the big picture, and get lost in too many details during program coding, for they are inclined to concentrate on the code itself rather than the design and on one single function rather than the system as a whole (Berge, Borge, Fjuk, Kaasboll, & Samuelsen, 2003; Bucci, Long, & Weide, 2001). Based on the finding indicated by this study, the researchers consider that the group's thinking disposition grew into a holistic one thanks to the teacher candidates' attempts at paying more attention to the effect of the interrelations of the modules on the whole by focusing on the system as a whole instead of loops, conditions, and functions during their learning process, although they had the analytical thinking disposition at the beginning. In addition, when it comes to critical thinking disposition and metacognitive awareness, the participants' critical thinking disposition and metacognitive awareness scores improved significantly after the learning took place, which means that learning how to program has a significant effect on teacher candidates' critical thinking disposition and metacognitive awareness. The presence of a significant difference in critical thinking disposition and metacognitive awareness at the same time can be explained by the fact that the two are interrelated concepts. Indeed, the definition of critical thinking given by Paul (1993) is almost the same with that of metacognition. Bensley (2011) states that a number of studies have pointed out to the positive relationship between metacognition and critical thinking. The significant difference found can be explained by the skills developed through programming mentioned in the literature (McMahon, 2009; Guss & Wiley, 2007). Also, the literature contains many other studies pointing out to similar changes in these variables (Kwon, Yoon, & Lee, 2011; Unuakhalu, 2008).

Among the outputs of the practice process are the programs developed by the teacher candidates in line with the learning outcomes defined in the 5th-8th grades curriculum. These programs were evaluated in accordance with the criteria created and grouped under the themes called Educational Content, Design and Programming. The findings suggest that the participants were able to develop effective programs according to these criteria. In consequence, it would be right to make the deduction that the participants were able to integrate effectively the knowledge and skills they acquired during learning to program into their teaching practices.

It is possible to evaluate this 'process of learning to program' in general taking into consideration the skills acquired, programs developed and positive effects on thinking. The research made use of theory and practice at the same time during the teaching of programming. The participants found it favourable to learn theory and practice at the same time, as they think they can learn best by experience. The literature also supports this kind of learning process, with findings suggesting that the learning takes place with more success using the theory and practice at the same time following the provision of the basics to the students (Crews and Murphy, 2004; Ziegler and Crews, 1999 as cited in Hu, 2004). In addition, there are studies emphasising that use of visuals in software ─ especially making use of visual tools in subjects associated with conditional statements and loops ─ and interactive methods have significant effect on student success and motivation (Arabacıoğlu, Bülbül, & Filiz, 2007; Cooper, Dann, & Pausch, 2003; Gültekin, 2006; Hu, 2004; Kelleher, Pausch, & Kiesler, 2007; Malan & Leitner, 2007; Peppler & Kafai, 2007; Ramadhan, 2000). There also are other studies demonstrating that using graphics and animations helps students pay more attention to classes (Bishop-Clark, Courte, & Howard, 2007; Brusilovsky & Spring, 2004; Lin & Zhang, 2003). Therefore, the MIT Inventor application helped students to regulate the program samples they were provided with and to create their own designs in an interactive way, which made the learning process more effective. The participants told that one of the difficulties they experienced during this process was the language barrier because of the fact that the MIT App Inventor platform was in English though their English not being very good. Nevertheless, they further stated that this was a problem only in the beginning, and this barrier disappeared over time, as they got more familiar with the application. The literature contains studies that found a significant positive correlation between academic achievement in English and academic achievement in computer programming (Leeper & Silver,1982; Nowaczyk, 1983), as well as studies that found no correlation between proficiency in English or any other foreign language and success in programming (Byrnes & Lyons, 2001, as cited in Jones & Burnettt, 2008). The researchers of this study are of the opinion that although the teacher candidates saw the language as a barrier in the beginning, they overcame this barrier in the course of the process. Additionally, the participants found the drag and drop feature of the MIT App Inventor environment to be useful in understanding and concentrating on programming. They also thought that the feature of the MIT App Inventor that enabled them to see the effects of the changes they made was a useful one. Similarly, Cliburn (2008) states that the drag and drop interface helps students to focus on programming concepts instead of debugging, as it prevents syntactic errors. According to Naser (2008), students have difficulty in identifying relationships between algorithms and making predictions as to how the effects change when there are changes in the parameters of the algorithm. When considered from this point of view, MIT App Inventor is considered to be advantageous.

The majority of teacher candidates stated that group work provided them with some opportunities including giving feedback to one another, helping group members who were demotivated following their earlier failed attempts at programming to restore their motivation, and helping group members who thought they would fail in their attempts as they were not proficient computer users to keep on the process. The literature counts these opportunities as factors affecting learners' success in programming. Nelson and Rice (2000) emphasizes in their research the importance of receiving feedback about the codes written, while Leeper and Silver (1982) and Petersen and Howe (1979) present their research findings demonstrating that learners' previous computer experiences affect their success in programming. Bennedsen and Caspersen (2008), on the other hand, states that learners feel guilty and incompetent if they fail at their first programming attempts, and thus lose confidence and motivation. However, the participants of this study were able to overcome these negative factors listed in the literature by benefitting from the opportunities provided for them by the group work. The methods and techniques used in this research, including having the theory and practice simultaneously, collaborative learning, designing algorithms, use of an additional software, creating animations and making use of accessible codes, are also credited by Lin and Zhang (2003) for their positive effect on learner motivation. It is noteworthy to mention the effect of the participants' academic background in addition to the basic components of the learning environment. The participants are students at the department of elementary mathematics education, and thus have already learnt many of the skills needed for successful programming on their undergraduate courses. Indeed, there are many studies pointing out to the positive correlation between results in mathematics and successful programming (Byrne & Lyons, 2001; Erdoğan, 2005; Fletcher, 1984; Pea & Kurland, 1983; Soloway, Lochhead, & Clement, 1982, as cited in Reed & Burton, 1988; Webb, 1985). The participants also recognised this relationship, as it is obvious from the interviews. They said that designing an algorithm is analogous to mathematical proofs in which one uses variables systematically, as in the case of programming, to arrive at the proof after the verification process. Literature tells us of the difficulties experienced by students on introductory courses to programming due to lack of readiness (Cooper et al., 2000; Samurçay, 1989, as cited in Pane & Mayers, 1996). From this perspective, it is clear that the participants had an academic advantage in their process of learning to program thanks to their possession of the state of readiness for algorithmic thinking.

In evaluating the teaching process, the participants were asked to respond to items asking them to comment on the pros and cons of the process, on its possible contributions to their learning and teaching competencies, on whether they would, in the future, provide their own students with the opportunity to learning to program to develop applications, and on possible advantages and disadvantages this process could produce in the teaching of mathematics. While most of the participants stated that they were able to integrate technology into mathematics teaching, they added that they now understand what the computer does while it runs the codes of the program and what is going on in the memory, so they felt more competent and knowledgeable. Considering this through a computer literacy perspective, it can be said that this process is a step toward competencies in being principled, thoughtful and making accurate judgments as they think they are more competent and knowledgeable (Akkoyunlu & Kurbanoğlu, 2003). The teacher candidates expressed a strong opinion about the positive contributions of this process not only to their teaching competencies but also to their own learning. They thought the knowledge of programming and of what runs in the applications' background helps them gain a new perspective and improve their logical thinking, creativeness and problem solving skills. Their comment related to teaching competencies suggests that they think programming helps an effective learning full of fun to take place and helps students to create a new measurement instrument. This finding supports the profound effect of computer literacy on students' own learning and teaching practices. When it comes to the item which asks the teacher candidates whether they would provide their students, when they start teaching, with the opportunity to use a programming language to develop an application, 12 teacher candidates thought that their students might have real difficulty in such a case considering their target student profile (11-14 years of age) would not have basic knowledge in programming languages. Six participants commented on possible pros and cons without specifying whether they will provide their students with that opportunity or not. On the other hand, 24 teacher candidates stated that now children grow up with technology in a digital age, so they will not have any big problems learning to program, and stated that students can be taught it through a simpler structure, or they can work with interested students who will be involved in voluntary work in programming. According to the participants, among possible pros and cons of using programming languages to develop an application for students were in general helping students improve their competencies and higher order thinking skills, building up their confidence about technological competence; however, if it was not taught them through a simpler structure, this might lead to a loss of motivation for technology. Such comments by the participants, which point out to potential problems together with solutions to these problems, suggested that they developed a high level of awareness.

In conclusion, this study focuses on mathematics education students' skills development and integration of those skills into their teaching practices during pre-service teacher education. The participant also confirmed their new awareness with their opinions. Integrating programming courses into curricula is becoming a widespread phenomenon in the world not only because programming is a prerequisite for professional competence, but also because is an important part of computer literacy. It is highly important for the teacher candidates to be a computer-literate before their graduation from the university. Therefore, it is recommended that programming courses should not be confined only to the students of the Computer Education and Instructional Technology Departments; they should be taken by all teacher candidates instead. Moreover, it is a very important process for the teacher candidates to graduate as computer-literate individuals. The participants of this research, students of mathematics education, they already had good problem solving skills even in the beginning of the process. However, it is possible to design the learning to program process with various activities aimed at improving problem solving skills for different student groups. Whether the same results will come out from research with similar or different groups requires testing with further research. It is believed that this study presents recommendations for future studies in terms of designing the teaching process, as it observes the progress of skills development and takes into consideration the participants' opinions.

References

Akın, A., Abacı, R., & Çetin, B. (2007). The validity and reliability of the Turkish version of the metacognitive awareness inventory. *Educational Sciences: Theory & Practice, 7*(2), 671-678.

Akkoyunlu, B., & Kurbanoğlu, S. (2003). Öğretmen Adaylarının Bilgi Okuryazarlığı ve Bilgisayar Öz-Yeterlik Algıları Üzerine Bir Çalışma. A Study on Teacher Candidates' Perception of Information Literacy and Self-Adequacy in Information Technology *Hacettepe University Faculty of Education Journal, 24*, 1-10.

Algozzine, B., Bateman, L. R., Flowers, C. P., Gretes, J. A., Hughes, C. D., & Lambert, R. (1999). Developing technology competencies in a college of education. *Current Issues in Education* [On-line], *2*(3). Retrieved from http://cie.ed.asu.edu/volume2/number3/

Anderson, R. E., & Klassen, D. L. (1981). A conceptual framework for developing computer literacy instruction. *AEDS Journal, 14*, 128-143.

Arabacıoğlu, C., Bülbül, H., & Filiz, A. (2007). *Bilgisayar programlama öğretiminde yeni bir yaklaşım*. A new approach towards teaching computer programming. 2007 Conference on Academic Information Technology, Dumlupınar University, Kütahya. Retrieved from http://ab.org.tr/ab07/bildiri/99.doc on 13 June 2016

Battista, M. T., & Steele, K. J. (1984). The effect of computer-assisted and computer programming instruction on the computer literacy of high ability fifth grade students. *School Science and Mathematics, 84*(8), 649-658. https://doi.org/10.1111/j.1949-8594.1984.tb09580.x

Bennedsen, J., & Caspersen, M.E. (2008). Teaching of Programming. In J. Bennedsen, M. E. Caspersen, & M. Kölling (Eds.), *Reflections on the Teaching of Programming Methods and Implementations* (pp. 6-16). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77934-6_2

Bensley, D. A. (2011). Rules for reasoning revisited: Toward a scientific conception of critical thinking. In C. P. Horvath, & J. M. Forte (Eds.), *Education in a Competitive and Globalizing World: Critical Thinking* (pp. 1-36). New York, NY, USA: Nova.

Berge, O., Borge, R. E., Fjuk, A., Kaasboll, J., & Samuelsen, T (2003, November 24-26). *Learning Object Oriented Programming*. Paper presented at the Norsk Informatik konferanse (Norwegian Informatics Conference), Norvey, Oslo.

Bishop-Clark, C., Courte, J., & Howard, E. V. (2007). A Quantitative and Qualitative Investigation of Using Alice Programming to Improve Confidence, Enjoyment and Achievement among Non-Majors. *Journal of Educational Computing Research, 37*(2) 193-207. https://doi.org/10.2190/J8W3-74U6-Q064-12J5

Brusilovsky, P., & Spring, M. (2004). Adaptive, Engaging, and Explanatory Visualization in a C Programming Course. *Proceedings of EDMEDIA' 2004 - World Conference on Educational Multimedia, Hypermedia and Telecommunications* (s. 1264-1271). Lugano, Switzerland.

Bucci, P., Long, T. J., & Weide, B. W. (2001, February 21-25). *Do we really teach abstraction?* Paper presented in 32rd SIGCSE Technical Symposium on Computer Science Education (pp. 26-30). USA, North Carolina. https://doi.org/10.1145/364447.364531

Büyüköztürk, Ş. (2008). *Sosyal Bilimler için Veri Analizi El Kitabı. A Handbook of Data Analysis for Social Studies* (9th ed.) Ankara: Pegem Akademi.

Byrne, P., & Lyons, G. (2001). The Effect of Student Attributes on Success in Programming. In Proceedings of the Sixth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005) (Canterbury, UK, June 25-27, 2001). ACM Press, New York, NY, 2001, 49 - 52. https://doi.org/10.1145/377435.377467

Cheng, T. T., Plake, B., & Stevens, D.J. (1985). A validation study of the computer literacy examination: Cognitive aspect. *AEDS Journal, 18*(3), 139-151. https://doi.org/10.1080/00011037.1985.11008395

Cliburn, D. C. (2008). *Students Opinions of Alice in CS1*. Paper presented in 38th ASEE/IEEE Frontiers in Education Conference, October 22-25. NY, Saratoga Springs. https://doi.org/10.1109/FIE.2008.4720254

Cooper, S., Dann, W., & Pausch, R. (2003). Using animated 3D graphics to prepare novices for CS1. *Computer Science Education, 13*(1), 3-30. Retrieved from http://www.sju.edu/~scooper/alice/scooper_csej.pdf

Crews, T., & Murphy, C. (2004). *Programming right from the start with Visual Basic.NET*. Prentice Hall.

Dewey, R. A. (2007). *Psychology: An introduction*. Scarborough: Wadsworth, 2004.

Erdoğan, B. (2005). Programlama başarısı ile akademik başarı, genel yetenek, bilgisayara karşı tutum, cinsiyet ve lise türü arasındaki ilişkilerin incelenmesi. *A review on the relationship of success in programming and academic success, general ability, attitude towards computers, gender and high school type* (Master's thesis, Marmara University Graduate School of Educational Sciences).

Facione, P. A. (1990). *A Statement of Expert Consensus for Purpose of Educational Assessment and Instructions*. The Delphi Report. East Lansing, National Center for Research on Teacher Training. EBSCOST ERIC Document No: ED315423.

Fletcher, S. H. (1984). *Cognitive Abitities & Computer Programming*. EDRS(ED259700).

Gabriel, R. M. (1985a). Assessing computer literacy: A validated instrument and empirical results. *AEDS Journal, 18*(3), 153-171. https://doi.org/10.1080/00011037.1985.11008396

Gabriel, R. M. (1985b). Computer literacy assessment and validation: Empirical relationships at both student and school levels. *Journal of Educational Computing Research, 1*(4), 415-425. https://doi.org/10.2190/CQ0A-0YMT-J10G-5WBV

Ganske, L., & Hamamoto, P. (1984). Response to crisis: A developer's look at the importance of needs assessment to teacher educators in the design of computer literacy training programs. *Educational Computer Technology Journal, 32*(2), 101-113.

Gomes, A., & Mendes, A. J. (2007) Learning to program – difficulties and solutions. *Proceedings of the 2007 international convergence on Engineering education*, September 3-7, Coimbra, Portugal

Gültekin, K. (2006). *Çokluortamın programlama başarısı üzerindeki etkisi/Effects of multimedia on success*. (Master's thesis, Hacettepe University Graduate School of Science).

Gundurao, H. K., Manjunath, N. S., & Nachappa, M. N. (2010). *Computer Technology and Computer Programming*. Mumbai, IND: Global Media.

Guss, C. D., & Wiley, B. (2007). Metacognition of problem-solving strategies in Brazil, India, and the United States. *Journal of Cognition and Culture, 7*(1), 1. https://doi.org/10.1163/156853707X171793

Haigh, R. W. (1985). Planning for computer literacy. *Journal of Higher Education, 56*(2), 161-171. https://doi.org/10.2307/1981664

Hammouri, H. A. M. (2003). An investigation of undergraduates' transformational problem solving strategies: Cognitive / metacognitive processes as predictors of holistic/analytic strategies. *Assessment & Evaluation in Higher Education, 28*(6), 571-586. https://doi.org/10.1080/0260293032000130225

Hartman, H. J. (1998). Metacognition in teaching and learning: An introduction. *Instructional Science, 26*(1), 1-3. https://doi.org/10.1023/A:1003023628307

Hasset, J. (1984). Computers in the classroom. *Psychology Today*, 22-28.

Helminen, J., & Malmi, L. (2010). *Jype-a program visualization and programming exercise tool for Python.* In Proceedings of the 5th international symposium on Software visualization (pp. 153-162). ACM. https://doi.org/10.1145/1879211.1879234

Heppner, P. P., & Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology, 29*(1), 66. https://doi.org/10.1037/0022-0167.29.1.66

Hu, M. (2004). Teaching Novices Programming with Core Language and Dynamic Visualization. *Papers from the Proceedings of the 17th NACCQ (National Advisory Committee on Computing Qualifications)* (s. 94-103). Retrieved from http://www.naccq.ac.nz/conference05/proceedings_04/hu.pdf on 13 May 2016.

İpek, İ. (2001). *Bilgisayarla Öğretim Tasarım, Geliştirme ve Yöntemler/Teaching with Computers: Design, Development and Methods*. Ankara: Tıp Teknik Yayınevi.

Johnson, D. C., Anderson, R. E., Hansen, T. P., & Klassen, D. L. (1980). Computer literacy What is it? *Mathematics Teacher*, 91-96.

Jones, S., & Burnett, G. (2008). Spatial ability and learning to program. An *Interdisciplinary Journal on Humans in ICT Environments, 4*(1), 47-61. https://doi.org/10.17011/ht/urn.200804151352

Kay, R. (1990). The relation between locus of control and computer literacy. *Journal of Research on Computing in Education, 22*(2), 464. https://doi.org/10.1080/08886504.1990.10781935

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Table Of Contents*. San Jose, California, USA. https://doi.org/10.1145/1240624.1240844

Kılınç, A., & Salman, S. (2006). Fen ve Matematik Alanları Öğretmen Adaylarında Bilgisayar Okuryazarlığı/ Computer Literacy of Science and Mathematics Teacher Candidates *Mersin University Faculty of Education Journal, 2*(2), 150-166.

Kökdemir, D. (2003). Decision Making and Problem Solving under Uncertainty (Unpublished doctorate thesis). Ankara University, Institute of Social Sciences.

Krueger, K., Hansen, L., & Smaldino, S. (2000). Preservice teacher technology competencies: A model for preparing teachers of tomorrow to use technology. *TechTrends, 44*(3), 47-50. https://doi.org/10.1007/BF02778227

Kwon, D., Yoon, I., & Lee, W. (2011). Design of programming learning process using hybrid programming environment for computing education. *KSII Transactions on Internet and Information Systems (TIIS), 5*(10), 1799-1813. https://doi.org/10.3837/tiis.2011.10.007

Leeper, R. R., & Silver, J. L. (1982). *Predicting success in a first programming course*. 13th SIGCSE Technical Symposium on Computer Science Education (s. 147-150). Indianapolis, Indiana, United States. https://doi.org/10.1145/800066.801357

Levin, D. (1983). Everyone wants 'computer literacy' so maybe we should know what it means. *The American School Board Journal*, 25-28.

Lin, C., & Zhang, M. (2003, April). The use of computer animation in teaching discrete structures course. *MICS 2003 Proceedings the 36th Annual Midwest Instruction and Computing Symposium.* Retrieved from http://www.micsymposium.org/apache2-default/mics_2003/Lin.PDF on 24 June 2016

Luehrmann, A. (1981). Computer literacy: What should it be? *Mathematics Teacher*, 682-686.

Malan, D. J., & Leitner, H. H. (2007). *Scratch for budding computer scientists*. 38th ACM Technical Symposium on Computer Science Education. Covington, Kentucky. https://doi.org/10.1145/1227310.1227388

McCormick, C. B. (2003). Metacognition and learning. In W. M. Reynolds, & G. E. Miller (Eds.), *Handbook of psychology: Educational psychology* (pp. 79-102). Hoboken: Wiley. https://doi.org/10.1002/0471264385.wei0705

McMahon, G. (2009). Critical Thinking and ICT Integration in a Western Australian Secondary School. *Journal of Educational Technology & Society, 12*(4).

Meierhenry, W. C. (1982). Microcomputers and adult education. *Training and Developmental Journal*, 58-66.

Naser, S. S. (2008). Developing Visualization Tool for Teaching AI Searching Algorithms. *Information Technology Journal, 7*(2), 350-355. https://doi.org/10.3923/itj.2008.350.355

Nelson, M., & Rice, D. (2000). *Introduction to algorithms and problem solving*. 30th ASEE/IEEE Frontiers in Education Conference, IEEE. Retrieved from http://fie-conference.org/fie2000/papers/1068.pdf on 3 May 2016

Nowaczyk, R. H. (1983). *Cognitive skills needed in computer programming*. The Annual Meeting of the Southeastern Psychological Association, Atlanta, Georgia.

O'Flaherty, J., & Philips, C. (2015). The use of flipped classrooms in higher education: A scoping review. *Internet and Higher Education, 25*, 85-95. https://doi.org/10.1016/j.iheduc.2015.02.002

Pane, J., & Myers, B. (1996). *Usability issues in the design of novice programming systems*. School of Computer Science Technical Reports, Carnegie Mellon University, CMU-CS-96-132.

Paul, R. W. (1993). The logic of creative and critical thinking. *American Behavioral Scientist, 37*(1), 21-40. https://doi.org/10.1177/0002764293037001004

Pea, R. D., & Kurland, D. M. (1983). *On The Cognitive Prerequisites of Learning Computer Programming (Technical Report No.18).* New York: Bank Street College of Education, Center for Children and Technology.

Peppler, K. A., & Kafai, Y. B. (2007). *What video game making can teach us about learning and literacy: Alternative pathways into participatory cultures*. Paper to be presented at the Digital International Games Research Association meeting in Tokyo, Japan. Retrieved from http://scratch.mit.edu/files/DiGRA07_games_kafai.pdf on 28 May, 2016

Petersen, C. G., & Howe, T. G. (1979). Predicting academic success in Introduction to Computers. *Association for Educational Data Systems Journal, 12*(4), 182-191. https://doi.org/10.1080/00011037.1979.11008252

Pickert, S. M., & Hunter, B. (1983). Redefining "literacy." *Momentum, 14*(3), 7-9.

Ramadhan, H. A. (2000). Programming by discovery. *Journal of Computer Assistes Learning, 16*, 83-93. https://doi.org/10.1046/j.1365-2729.2000.00118.x

Reed, W. M., & Burton J. K. (1988). *Educational computing and problem solving*. Haworth Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67. https://doi.org/10.1145/1592761.1592779

Roehl, A., Reddy, S., & Shannon, G. (2013). The flipped classroom: an opportunity to engage millennial students through active learning strategies. *Journal of Family and Consumer Sciences, 2*(105), 44-49. https://doi.org/10.14307/JFCS105.2.12

Şahin, N., Şahin, N. H., & Heppner, P. P. (1993). Psychometric properties of the problem solving inventory in a group of Turkish university students. *Cognitive Therapy and Research, 17*(4), 379-396. https://doi.org/10.1007/BF01177661

Şahin, T., & Yıldırım, S. (1999). *Öğretim Teknolojileri ve Materyal Geliştirme/Educational Technologies and Materials Development*. Ankara: Anı Yayıncılık.

Schraw, G., & Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary educational psychology, 19*(4), 460-475. https://doi.org/10.1006/ceps.1994.1033

Sleeman, D., Putnam, R. T., Baxter, J. A., & Kuspa, L. K. (1984). *Pascal and High-School Students: A Study of Misconceptions*. Technology Panel Study of Stanford and the Schools (ERIC Document Reproduction Service No. ED258552).

Tucker, A. Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003). *A Model Curriculum for k12 Computer Science: Final Report of the Association for Computing Machinery (ACM)*. New York. Retrieved from http://www.isp.org.pl/podstawa/podstawa_files/K12_Computer_Science.pdf

Umay, A., & Arıol, Ş. (2011). Baskın olarak bütüncül şekilde düşünenler ile baskın olarak analitik stilde düşünenlerin problem çözme davranışlarının karşılaştırılması/Comparing holistic thinkers and analytical thinkers in terms of problem solving skills, *Pamukkale University Faculty of Education Journal, 30*(11), 27-37.

Unuakhalu, M. F. (2008). Enhancing problem-solving capabilities using object-oriented programming language. *Journal of Educational Technology Systems, 37*(2), 121-137. https://doi.org/10.2190/ET.37.2.b

Uşun, S. (2000). *Özel Öğretim Teknolojileri ve Materyal Geliştirme/Special Education Technologies and Materials Development*. Ankara: Pegem A Yayıncılık.

Vaughan, M. (2014). Flipping the learning: An investigation into the use of the flipped classroom model in an introductory teaching course. *Education Research and Perspectives, 41*, 25-41.

Venezky, R., & Osin, L. (1991). *The Intelligent Design of Computer Assisted Instruction*. Longman Publishing Group, New York & London.

Webb, N. M. (1985). Cognitive requirements of learning computer programming in group and individual settings. *AEDS Journal, 18*(3), 183-193. https://doi.org/10.1080/00011037.1985.11008398

Yalın, H. İ. (2001). *Öğretim Teknolojileri ve Materyal Geliştirme/Educational Technologies and Materials Development*. Ankara: Nobel Yayınları.