International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 1
ISSN: 2088-5334

An Analysis of Pre-service Teachers' Learning Process in Programming Learning

Seong-Won Kim[#], YoungJun Lee[*]

[#]*Global Institute For Talented EDucation (GIFTED), Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, Republic of Korea*
E-mail: sos284809@gmail.com

[*]*Dept. of Computer Education, Korea National University of Education,250 Taeseongtabyeon-ro, Grangnae-myeon, Heungdeok-gu, Cheongju, 28173, Republic of Korea*
E-mail: yjlee@knue.ac.kr

***Abstract*— As the importance of computing technology increases, computer science education is being actively implemented around the world. Because computer science education is being introduced into the curriculum, research on how to effectively teach programming (which is the core of automation) is actively underway. Although the importance of block-based programming languages has increased, most studies have focused on text-based programming languages. As interest in programming increases, block-based programming languages will be taught to a variety of audiences. Therefore, this study analyzed Code.org, which provides a development environment for block-based programming; this study then investigated the programming learning process of pre-service teachers, who used Code.org. Sixteen pre-service teachers participated in the study, and their learning processes were uncovered by analyzing their programming results. This suggests that pre-service teachers can learn sequential and necessary repetition without difficulty. However, the pre-service teachers failed to use the repetition block through abstraction. Besides, for While and Until, pre-service teachers did not understand the concept of repeating according to the condition. For Counter, pre-service teachers had difficulty repeating the use of variables. In the condition, pre-service teachers were not able to separate the command, which should be executed when the condition is True and when it is False. For Event, pre-service teachers had no problem utilizing the function, but they were not able to call the function with a parameter. Based on this, it was confirmed that a pre-service teacher can understand the principle of programming development in advance by understanding the abstraction, condition, and variable in the loop statement. In this study, there was a limit to practicing block-based programming language due to the platform's low scalability. Future research should solve these problems and diversify the research subjects.**

***Keywords*— programming; learning process; pre-service teacher; difficulties of programming; code.org; computer science.**

I. INTRODUCTION

Due to its development, computing technology is increasingly being used in various areas of everyday life, such as the industry and the economy. New disciplines and areas of computing technology are emerging, and humans' lives are rapidly changing as a result of computing technology. It has been predicted that future life will change into a form that has not been experienced. During the 2016 World Economic Forum, this change was called the Fourth Industrial Revolution [1]. In the Fourth Industrial Revolution, they said that lifeforms would change rapidly based on different technologies, such as artificial intelligence, the Internet of Things, and robots. Therefore, the importance of computing technology has increased, and the need to cultivate human resources in the computing field has also increased [2]. As a result, there have been attempts to promote computer science education around the world [3].

The United Kingdom has mandated "Computing" subjects in K-12 courses, and they are pursuing computer science education in cooperation with various companies, such as the BBC, Microsoft, and Samsung [4]. The United States has developed standards and frameworks for computer science education and is making enormous investments into training computer science talent [5]. In Korea, a 2015 revision curriculum requires students to take an informatics course in middle school; computer science education is referred to as software education [6]. Besides, various countries (such as Japan, France, and Finland) are trying to introduce computer science education into their curriculums [7].

The most important core competency in computer science education is computational thinking. Computational thinking is the key to abstraction and automation. Computational

thinking is composed of abstraction and automation, and programming is very important in automation [8]. Thus, research on text-based programming languages (such as C, Java, and Python) has been actively conducted [9]–[11]. Computer science education is necessary in K-12 education [12], but elementary and middle schools use block-based (rather than text-based) programming languages for education—according to the students' cognitive levels [13]. However, there is not much research on block-based programming languages [14]–[16]. In addition, as the importance of programming increases, research should be conducted on pre-service teachers learning and teaching block-based programming languages [17]–[20]. But there is a lack of research on the process of learning of block-based programming languages [18].

To fill this gap, this study examined pre-service teachers' learning process of a block-based programming language. For this research, learning courses and lessons (which were taken by pre-service teachers) at Code.org were analyzed. Code.org provides a learning environment for a block-based programming language. Each task analyzed the pre-service teachers' results in order to determine the programming learning process.

II. Materials and Method

A. *Code.org*

Although the importance of computer science is increasing, the number of students majoring in computer science has plummeted due to difficulties in programming. To solve this problem, computer science-related institutions, non-profit organizations, and governments have joined to develop educational programs that will help students become interested in computer science. As a result of these movements, visual programming languages have been developed (such as Scratch, Blockly, Alice, and Kodu); these will help students become interested in programming [19]. Scratch and Blockly are block-based programming languages that have been developed in a block-based programming language; this has been done to address grammatical errors, which often cause difficulties for students when learning programming [20]. As many literatures have verified the effectiveness of the block-based programming language, Code.org was developed by various institutions (such as Google, Microsoft, Facebook, and Twitter) in 2013. Code.org was developed to help students become interested in computer science and learn the principles of computer science. Code.org provides a course titled 'CS Fundamentals', which is suitable for kindergarten and middle school students. Code.org also has App Lab, Game Lab, Web Lab, and Physical Computing (CS Discoveries), which are for middle school students. 'CS Principles course' is for high school students and undergraduate students who are pursuing an education in computer science. The site provides students with an educational environment for learning computer science at a high level. It also provides educational materials and student management systems for teachers [21]. In short, it provides the total package for a K-12 computer science education.

Code.org is a basic programming environment; like Scratch and Blockly, Code.org provides a block-based programming language environment. The CS Fundamentals course is broken up as follows: Course 1 (age 4 ~ 6), Course 2 (age 6 and above), Course 3 (age 8 ~ 18), Course 4 (age 10 ~ 13), and the Accelerated CS Course (ages 10 ~ 13) (e.g. Fig. 1).

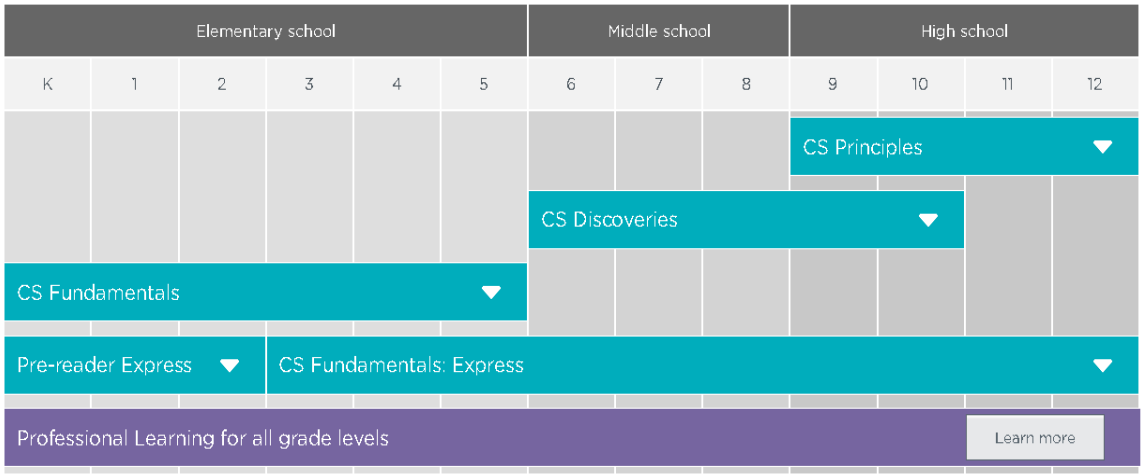


Fig. 1 Programming patterns of pro-service teachers' in Lesson 5 at The Farmer 3

There is also an Hour of Code Course, which allows the learner to experience computer science for an hour. The Hour of Code Course includes a variety of activities based on topics that students will be interested in, such as Angry Birds, Star Wars, Minecraft, and Frozen. Code.org's learning process is structured so that teachers can provide a specific course or so that students can learn the course on their own (e.g., Fig. 2).

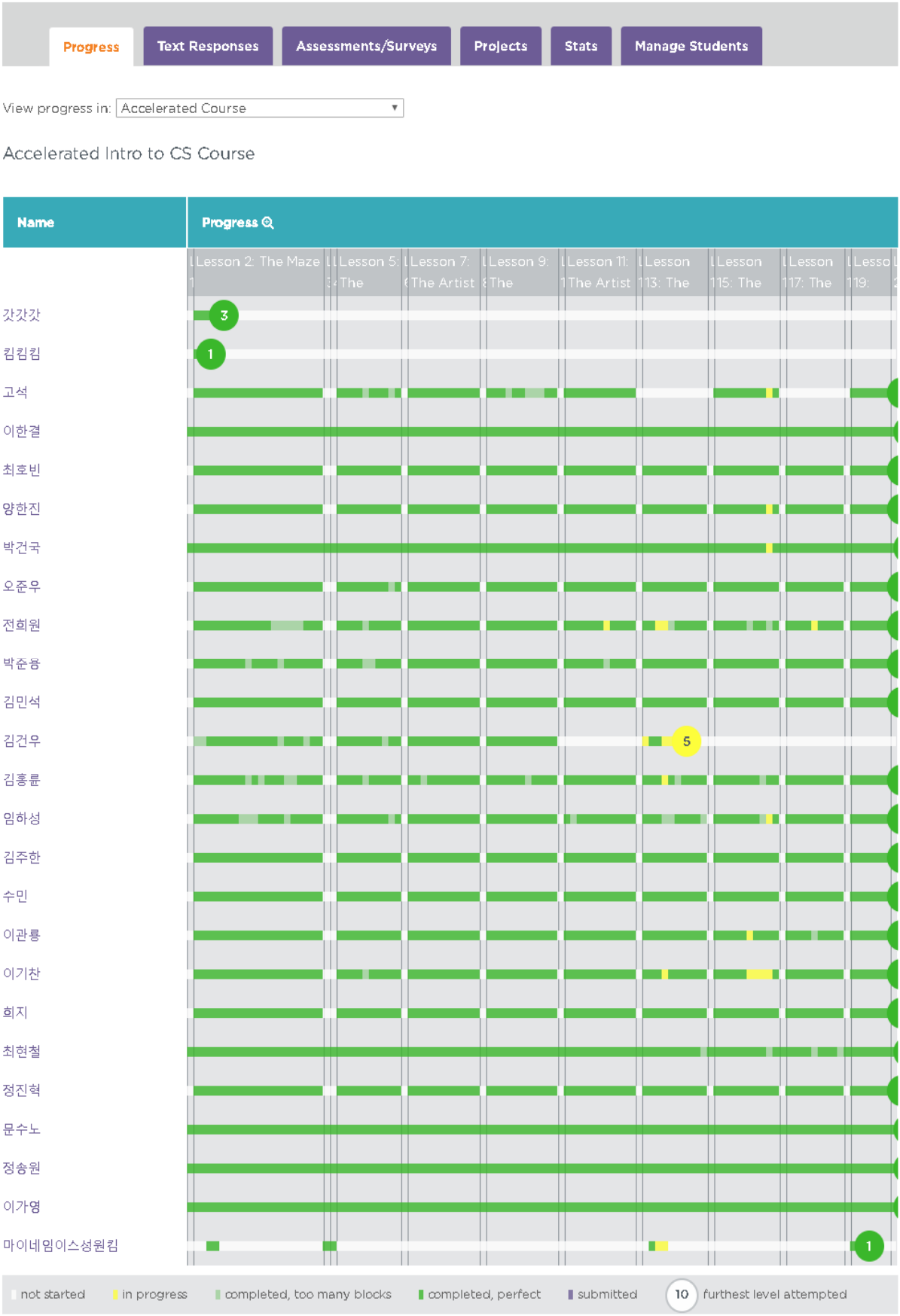


Fig. 2 Teacher pages for managing students at code.org

When a student selects the course, they want to study; the student can see their overall progress within the course. Besides, if a student clicks on the desired lesson, a video about the course will be provided. The video will introduce the lesson, and students will learn how to progress by watching the video. After watching the video, the student will proceed and start programming, and he or she will solve the problem by using the toolbox and workspace based on the instructions. After programming the solution to the problem, the student can check their results in the Play area and repeat the process of debugging. If the student has trouble in solving a problem, a video or hint is provided (to reduce the difficulty). Code.org is currently available in over 40 languages and is free of charge; thus, students from many different countries can learn computer science.

B. Research Procedure

This study investigated the programming learning process of pre-service teachers. Pre-service teachers were recruited to participate in the research. Lectures were conducted so that the teachers could learn a block-based programming language. The pre-service teachers who participated in the research performed programming tasks, and the researchers compared the results of the tasks performed by the pre-service teachers with the solutions. Based on the comparison, the results and implications of the pre-service teachers' programming learning process were derived.

C. Research Subjects

In this study, pre-service teachers at the Korea National University of Education (which is in Korea) were studied. To recruit research subjects, the researcher held liberal arts lectures at Korea National University of Education. Explanations of the lectures and research's content were announced in advance; a pre-service teacher gave the lectures. A total of 24 pre-service teachers were enrolled in the open lectures. Of these, 22 (who agreed to participate in the research and sincerely participated in the research process) were selected as the research subjects.

The Korea National University of Education, which is the subject of this research, is a university that specializes in training teachers. Therefore, all undergraduate students attending the Korea National University of Education can be considered pre-service teachers. As for the demographics of the study subjects, there were 17 males and 5 females. The majors of the pre-service teachers were as follows: one Korean education, seven mathematics education, two English education, two early childhood education, one Chinese education, three physical education, two computer education, three chemistry education, and one environment education. The pre-service teachers included nine students in their freshman year, 10 students in their sophomore year, and three students in their junior year. There was no senior among the pre-service teachers [22].

The programming experiences of the pre-service teachers were investigated in order to understand their programming learning process. Of the 22 pre-service teachers, six pre-service teachers had experience with programming. Three out of these six had experience with text-based programming languages (like C and Java) and block-based programming languages. This was since, among the participating pre-service teachers, pre-service teachers were majoring in computer education. The other three of the six had experience using Scratch. Because three had used Scratch, they were able to create simple programs through Scratch, but they had no formal education in programming or computer science.

D. Treatments

In this study, the treatment was conducted as a part of liberal arts lectures in the fall and winter semesters at Korea National University of Education. The lectures were held for a total of 15 weeks (from August 31, 2017 to December 7, 2017). The programming learning process took four to seven weeks of lectures. The treatment was carried out on Code.org and consisted of the information from the accelerated CS Course. The accelerated CS Course had been developed for teaching K-8 students; the course consisted of 20 lessons that can be taken in 20 hours. Of the 20 lessons in the accelerated CS Course, 11 were unplugged. An unplugged activity is an ongoing activity that does not require a computer and that helps the learner understand the principles of computer science. So, in this study, artifacts of the other nine lessons (excluding the unplugged activities) were analyzed. This was done in order to focus on the programming learning process of pre-service teachers. To use Code.org, the researcher opened a class by using a teachers' account, and the 22 pre-service teachers completed the accelerated CS Course together in class. In the first task, Code.org was explained to the pre-service teachers; then, pre-service teachers carried out the remaining task by themselves [23]–[26].

E. Analysis

The purpose of this study was to analyze the programming learning process of pre-service teachers. First, in the accelerated CS Course, the programming elements included in the task were analyzed for nine lessons (which excluded the unplugged activities). After analyzing the elements in the accelerated CS Course, the programming results were investigated; a pre-service teacher prepared the results for each task. Based on the results (except for those marked "completed, perfect"), researchers analyzed the artifacts of the pre-service teachers that were marked "completed, too many blocks" and "in progress." In the literature, the learning process of programming has been discussed in order to investigate programming difficulties [15], [16]. In this study, the researchers compared the solution to the questions with the artifacts of the students; this was done in order to examine their programming learning process.

III. Results and Discussion

A. Analysis of Programming Elements in Code.org

In the Accelerated Course, nine lessons out of the 20 were examined; these nine lessons did not include the 11 unplugged lessons (Introduction to Computer Science, Computational Thinking, Paper Programming, Algorithms, Functions, Conditionals, Song Writing, Abstraction, Relay Programming, The Internet, and Wrap-up). Each of the nine lessons included tasks. The nine lessons were composed of

themes (The Maze, The Farmer, and The Artist). This study was analyzed by themes (e.g., Table 1 in Appendix).

1) The Maze

In the first lesson, The Maze, the blocks were assembled to move an object in the task. Tasks that move to a specific destination, including move and angular rotation were composed. Task 6 used repetition to move and turn objects. In Task 10, the Until statement was added. Task 14 added an If statement to the Until statement. An If-else statement was introduced in Task 18, and a Nested if it was used in Task 20. When the statements were used together, the Maze would experience coordinate movements, and Move and Turn could be used to escape the maze. In addition, Repeat and Until, If, If-else, and Nested if it could be used sequentially for efficient coordinate movement. In this way, the maze was composed of activities so that the student could learn sequence, loop, and condition, which are the most essential parts of programming.

2) The Artist

The Artist was like The Maze, but it consisted of activities that required the learner to draw specific shapes. The Maze simply moved up, down, left, or right. In contrast, The Artist allowed the user to think and design the angle of the rotation according to the shape. Therefore, The Artist consisted of activities for drawing shapes, such as Move, Turn, Skill, and Repeat. Artist 2 consisted of advanced activities for drawing shapes. Building on drawing simple shapes, Tasks included not only using multiple If statements but also developing tasks that use Nested repeats to draw various shapes.

The Artist 3 added content related to the event. A function was added as the content of the event, and a variable and random number were added for the function. Besides, in Task 6, a loop using a counter appeared. The Artist 3 consisted of Move and Turn commands in the function in order to draw triangles or rectangles by merely using the functions.

In Artist 4, a task arrived in the Goal states, including iteration, in the function. Thus, it was confirmed that the use of the advanced function has appeared. Also, based on Task 5, the nested function has appeared. Examples of functions and variables were provided for efficient troubleshooting. In Task 6, problem-solving activities using a function with a parameter appeared. In this way, the activity of drawing the existing figure did not call several functions, but it passed the parameter in the function and made an efficient program call through the Nested function. In the last task, Task 9, figure drawing activities using the counter, function, Nested function, and function with a parameter was shown.

The Artist 5 was not an activity that the learner created, but rather an activity that confirmed the result of the completed program. In this way, The Artist consisted of activities in which the learner exercised the movement and rotation of the object in the activity. This was done by drawing a specific figure or drawing various shapes by using the loop and event. One thing to note is that The Artist did not include a condition because it contains tasks that draw shapes.

3) The Farmer

The Farmer consisted of a task like a type developed in The Maze. The Maze was about going to a specific location while avoiding obstacles, and The Artist was about drawing a shape out of a model. The Farmer, however, added a command to Fill or Dig the ground while moving to a specific location. The Farmer was also composed of the Repeat, Nested repeat, Until, and If tasks.

In the Farmer 2, functions were added to the existing activities, and commands that contained iterations were included in the function (in addition to merely executing sequential commands in the function). In this way, pre-service teachers practiced how to organize programming efficiently during the execution of the same command. Finally, Tasks included problem-solving by using the sequence, loop, condition, and event, including If-else.

The Farmer 3 consisted of debugging tasks, running the blocks that were already present, and working on fixing the code to solve problems. In addition to designing and developing code for problem-solving, the students could analyze, execute, and correct already completed code. The Farmer 3 was sequentially presented with the sequence, loop, condition, and event elements as the tasks progressed. In this way, The Farmer taught the loop, condition, and event in a necessary sequence, and activities were constructed so that the students could experience debugging.

B. Analysis of Pre-Service Teacher's Programming Learning Process

1) The maze

The pre-service teachers solved all the problems in The Maze. The problems were solved in the problem-solving process, and there were cases where many blocks were used. Thus, pre-service teachers experienced no difficulty in solving problems by assembling blocks in a block-based programming environment. Besides, it was confirmed that Repeat, Until, If, If-else and Nested could be utilized in the process of reaching a specific position through Move, Turn, and solving problems. The following are cases in which the problem could not be resolved efficiently. In Lessons 8, 9, and 10, the pre-service teachers showed a tendency to present instructions in sequence even if the problems could be solved by repetition. In examples that did not solve the problem efficiently, there is a task shown in Fig. 3 (Lesson 9 in The Maze), which required two repetitions and three rotations.

The pre-service teachers should have come up with the solution of repeating two advances and three rotations by abstraction. However, the patterns that did not solve the problem seemed to be patterns that simply used repetition twice to advance—or that could not repeat all the commands. Only some of the commands were repeated, and the remaining commands were solved sequentially (e.g. Fig. 3).

Based on this, it was determined that the pre-service teachers understood repetition but failed to abstract and form a block based on it to solve the problem. Therefore, the pre-service teachers' solutions showed that repetition was simply used in the initial stage of programming (rather than for implementing repetition).

The pre-service teachers also failed to solve problems efficiently when learning the If statement. An example is Lesson 14; the task was to avoid obstacles and move to flowers. The learner needed to implement a move that advances if there was no path to the left (or rotate to the left if there was a path). The pre-service teachers were instructed that, if they all had a path to the left, they would have to turn to the left.

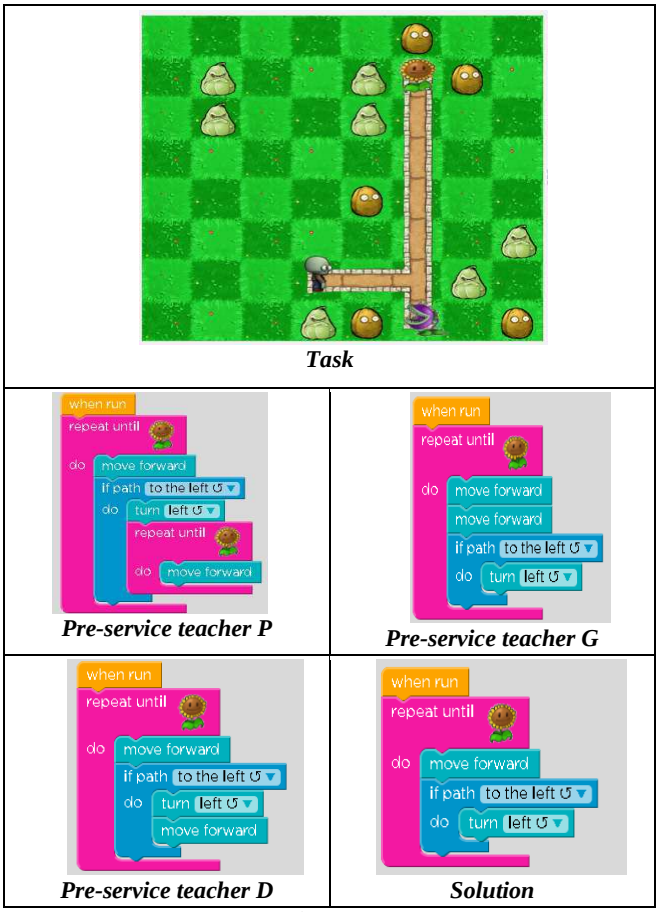


Fig. 3 Programming patterns of pro-service teachers in Lesson 9 at The Maze

However, pre-service teacher p put the Until statement in the If statement, and pre-service teachers g and p used an unnecessary move block (e.g. Fig. 4). These errors persisted in Lessons 15, 16, and 17. These results show that pre-service teachers can learn the concepts of sequence, loop, and condition in the block-based programming learning process. However, some pre-service teachers tended to solve problems inefficiently. While they understood the function of the blocks, they were not able to properly use Repeat and If to abstract in automation. “Reference [15]” suggested that pre-service teachers need a lot of practice and time to use the loop command [15] properly. “Reference [16]” pointed out that, when using a block-based programming language, there is a tendency to automate problem-solving without abstracting the problem; this results in problems with the sequential programming [16]. Therefore, as the pre-service teachers solved the problem in block-based programming learning, it was confirmed that the teaching was needed to solve the problem based on the abstract process.

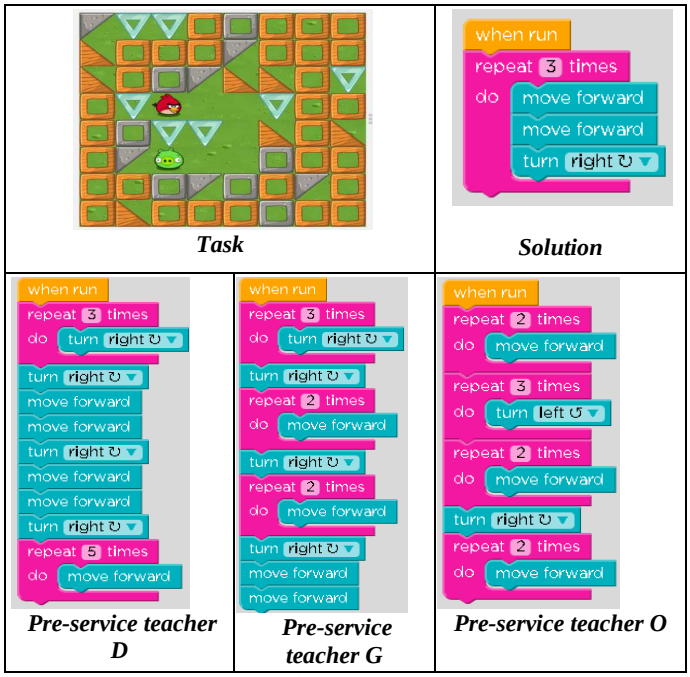


Fig. 4 Programming patterns of pro-service teachers in Lesson 14 at The Maze

2) The Artist

All the pre-service teachers solved the problem in The Artist 1. However, like The Maze, some pre-service teachers solved the problem inefficiently. By analyzing the pre-service teachers’ results, The Artist showed that the problem had been resolved inefficiently; unnecessary angular rotation blocks or various patterns were used to solve the problem (as shown in Fig. 5).

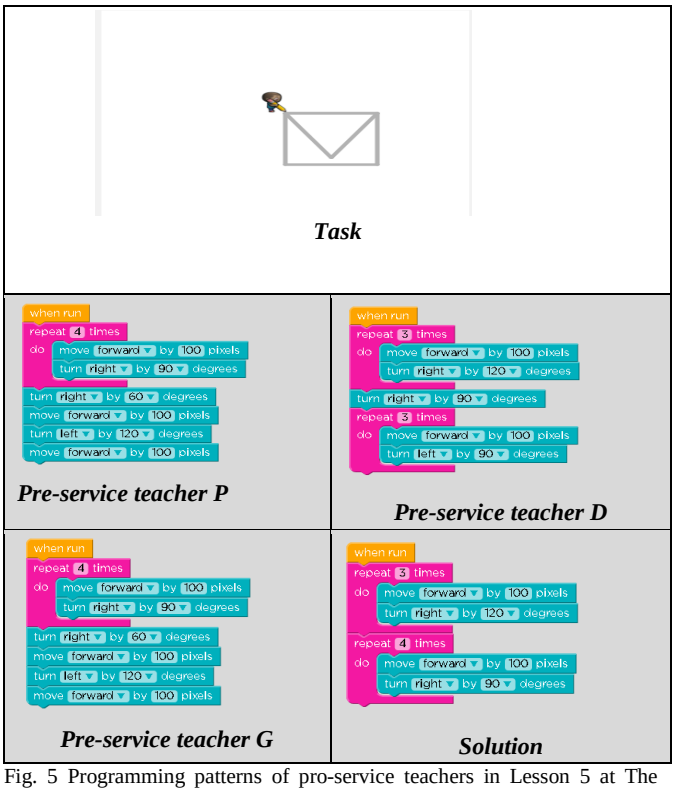
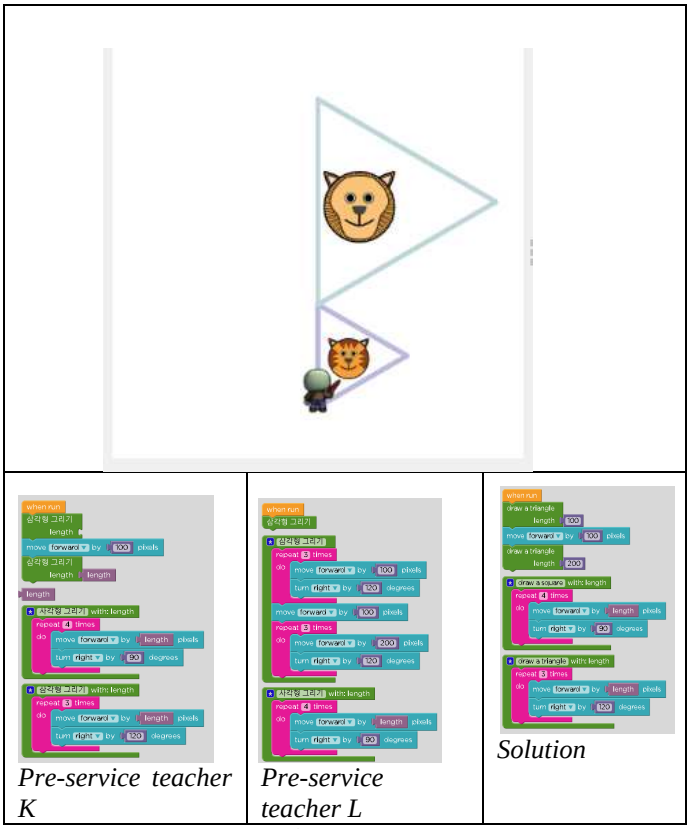


Fig. 5 Programming patterns of pro-service teachers in Lesson 5 at The Artist 1

The Artist 1's problem had already been experienced by the pre-teachers in The Maze, so it was confirmed that the Repeat command was used. Artist 2 also showed the same pattern. Nested repeat was present in The Artist 2, and all the pre-service teachers were able to solve this problem. Besides, The Artist 1 had 10 cases of an ineffective solution, while The Artist 2 did not. Therefore, the pre-service teacher had some difficulty in abstracting and automating the problem through the coordinates, but the result showed that it was easy to abstract and automate in the process of drawing a figure.

In the Artist 3, the concepts of variables and functions emerged. As a result, pre-service teachers were unable to solve problems for the first time. Some pre-service teacher did not command within the counter in Lesson 11 of The Artist 3. Pre-service teacher p, who solved the problem, utilized the function; however, when the first counter appeared, he failed to solve the problem and did not understand the concept of the counter at all. Thus, Repeat and Nested repeat, until were understood by pre-service teachers (based on their Code.org activities). However, it has been shown that repetition using variables (such as the counter) is hard to understand at first (e.g. Fig. 6).



Pre-service teacher K | *Pre-service teacher L* | *Solution*

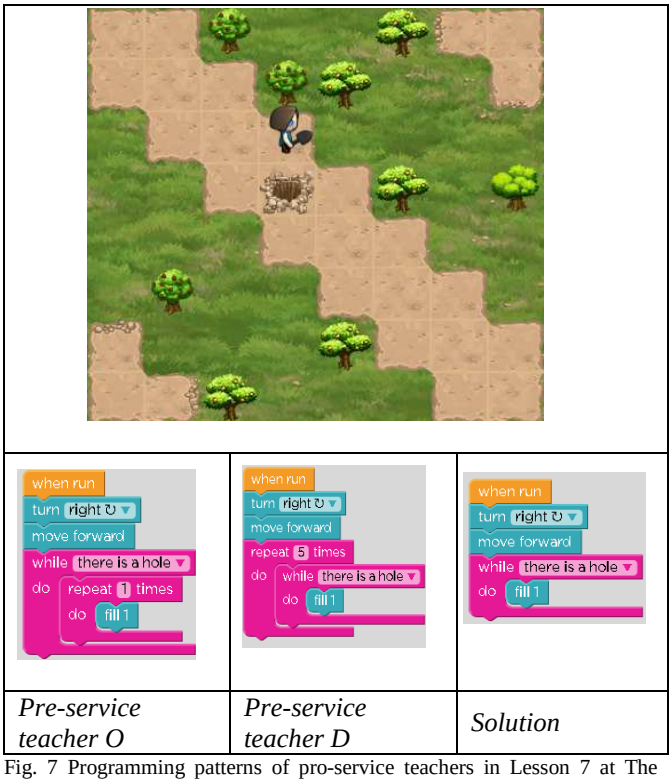
Fig. 6 Programming patterns of pro-service teachers in Lesson 11 at The Artist 3

In the Artist 4, many pre-service teachers had difficulty in learning programming. Many pre-service teachers failed to solve the problem with a function with a parameter (Lesson 6). All the pre-service teachers did not feel that it was difficult to implement the function, but they found it difficult to set the parameters. In Lesson 9, which required the problem to be solved by using the function with a parameter and counter, many pre-service teachers failed to solve the problem. Lesson 9 was about passing different parameters through a counter (when a function was called with a parameter) to draw the same shape in different sizes.

The pre-service teachers showed that the call to the counter failed. However, all of them used a certain number of iterations to draw shapes. The contents of the variable first appeared in Lesson 5 of The Artist 3. However, The Artist 3 and 4 did not explain the concept of the variable; instead, it replaced the existing calculation in the toolbox. Therefore, the pre-service teachers did not understand exactly what the variable was or how to use it in the counter or function with a parameter [15]. Therefore, the pre-service teacher did not use the function to pass the value in the problem-solving process, utilize the stored value, and change the stored value [16]. Finally, the Artist 5 was performed by all the pre-service teachers; this was because it was a direct execution process.

3) The Farmer

Like the Artist and The Maze, the pre-service teachers solved all the problems in The Farmer's first lesson. However, some pre-service teachers solved the problems inefficiently. An example is Lesson 7. Some pre-service teachers had been practicing problem-solving using the While statement in Lessons 5 and 6. While solving the advanced problem, the pre-service teacher solved the problem inefficiently. When implementing the While statement, the pre-service teacher added repeat inside or outside the While statement (e.g. Fig. 7).



Pre-service teacher O | *Pre-service teacher D* | *Solution*

Fig. 7 Programming patterns of pro-service teachers in Lesson 7 at The Farmer 1

Based on this, the pre-service teacher had not yet learned that the While statement is used to execute repetition. The pre-service teacher simply used repeat to repeat. Therefore, the While statement was recognized as a conditional

statement rather than an iteration. The pre-service teacher demonstrated a similar pattern in The Maze for Until. Therefore, it is easy to implement repetition (Repeat, Nested repeat), but it can be confirmed that it is difficult to repeat according to the condition (Until, While).

Some pre-service teachers did not solve the problem in The Farmer 2. Although there were pre-service teachers who failed to solve problems in various lessons, most pre-service teachers failed Lesson 4. The pre-service teachers solved the problem of using the repeat function in Lesson 2 and Lesson 3. In this case, the function was defined, and there was no change in direction. In Lesson 4, the pre-service teachers needed to look at existing functions but define functions using iterations. Some pre-service teachers had, in this way, been inefficient when defining iterations. Besides, some pre-service teachers failed to construct a repeat block by using the function, Move, and Turn blocks. This suggested that the pre-service teachers had difficulty in constructing repeat blocks [27].

Finally, in The Farmer 3, which involved experimenting with debugging, eight pre-service teachers did not solve the problem. Lesson 5 was the task of configuring the remove instruction to be executed according to the conditions specified by the instructions. The pre-service teacher understood that the remove command should be executed according to condition statements during debugging. However, it was unclear what command should be put in the loop (e.g., Fig. 8).

C. Discussion

"Reference [15]" stated that pre-service teachers should have at least four weeks (four times) of practice in repetitive blocks [15]. In this study, it was confirmed that pre-service teachers had difficulty solving the problem with the repetitive block—even after four weeks. This difficulty was due to the pre-service teachers' failure to abstract the problem [13].

"Reference [16]" suggested a reason why pre-service teachers have difficulty in learning programming. They said that there are difficulties in creating patterns and structuring overlaps in commands that should be written in the loop [16]. In this study, pre-service teachers failed to abstract commands that should have been repeated. However, in "Reference [16]", cases of not understanding the commands of the loop were not shown in the use of repeat and Nested repeat. However, for Until and While, it was not possible to execute the conditional command, and the counter could not use the variable properly [16]. In this way, it was confirmed that understanding the condition was limited to the use of the variable [18].

The course composition of Code.org itself may have caused this problem. Conditional statements were not included in The Artist but were included sometimes in The Maze and The Farmer. Only 12 out of 98 tasks used conditional statements. Therefore, there was the description and practice of conditional statements, but there was not much problem-solving using conditional statements. Thus, it may have been difficult to utilize Until or While due to a lack of understanding of the condition [29]. The problem with the variable was that the result is because the learner did not understand the variables in the process of solving the problem. The problem of the variable affected the counter and the function with a parameter. These problems appeared to be the most difficult in the problem-solving process [21].

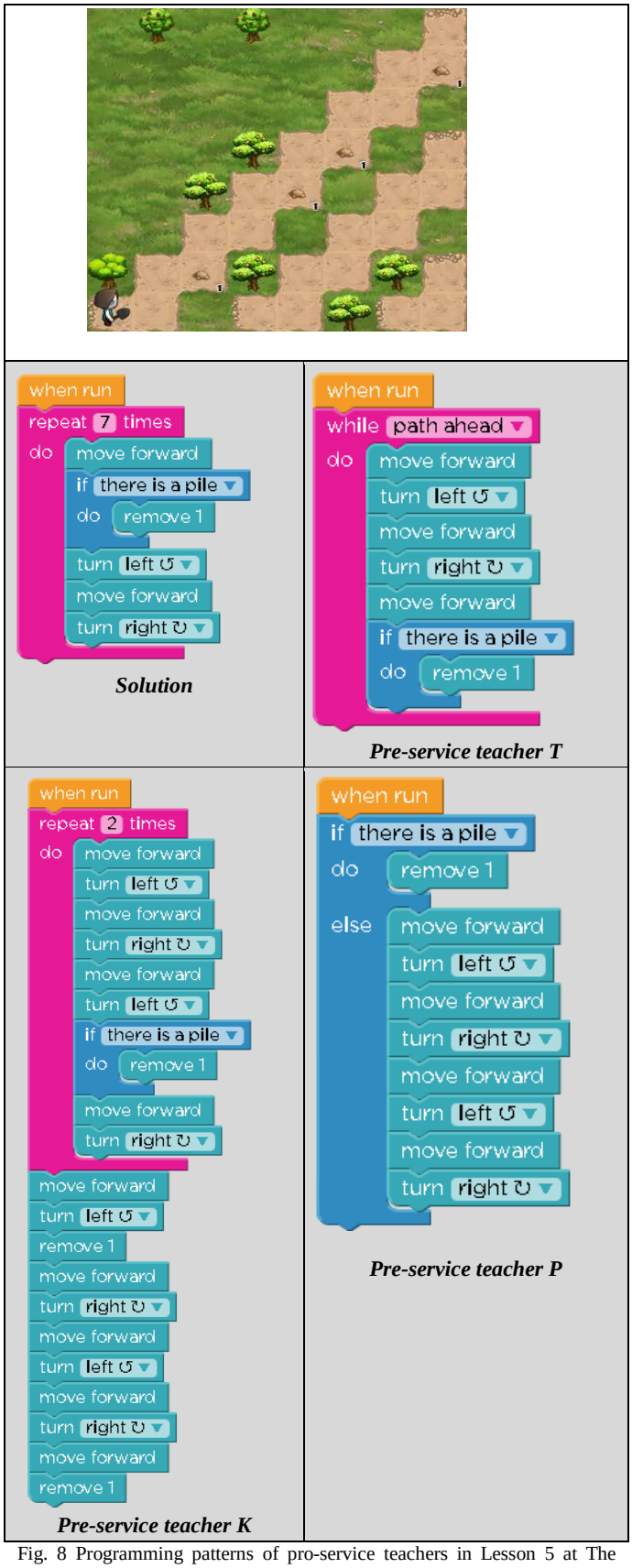


Fig. 8 Programming patterns of pro-service teachers in Lesson 5 at The Farmer 3

"Reference [14]" stated that novices and experts use the same problem-solving methods when learning a block-based programming language [14]. However, in this study, it was shown that novices (specifically pre-service teachers) implement automation when solving problems but do not perform abstraction properly. Therefore, experts and pre-service teachers have different ways of solving problems [28],[30].[31].

IV. Conclusion

This study tried to analyze the programming learning process of pre-service teachers by using Code.org, which is a block-based programming development environment. To do this, the accelerated CS Course at Code.org was used, and the artifacts of pre-service teachers were compared with the solutions.

The accelerated CS Course at Code.org was largely composed of three activities. The Maze was configured to learn sequence, loop, and condition while moving an object to specific coordinates. The Artist consisted of five steps, and the pre-service teachers learned sequence, loop, and event while drawing a specific shape. The Farmer was structured to teach sequence, loop, condition, and event; this was done so that the pre-service teachers could perform a specific skill. In this lesson, the contents of the loop were included in many lessons, but conditions and events were in relatively few lessons. There was a problem that involved using a variable in an event, but a description of the variable was not included.

The results showed that pre-service teachers generally performed Code.org tasks. Though, for sequence and loop, both Repeat and Nested repeat were programmed to reach the target state of interest. Some pre-service teachers could not solve them when using limited blocks. This result was that since the pre-service teachers did not abstract the problems suitably. The pre-service teachers failed in the process of decomposing the problems and recognizing the pattern and patterning; this was due to their tendency to solve the problems in an improvised manner. The pre-service teachers tried to solve the problems sequentially rather than by using repetition. Besides, the results confirmed that the Until, While and counter blocks were not appropriately used for problem-solving because the pre-service teachers lacked understanding about the blocks. Until and While, the pre-service teachers tended to use the loop repeatedly; this was because they did not understand it as a loop (as they lacked understanding of the condition). The pre-service teachers also had difficulty in using variables and were not able to solve problems using the counter and function with a parameter block.

In this study, the pre-service teachers were able to examine the learning process of block-based programming. The research was conducted on a platform called Code.org. Code.org is an effective tool for learning the principles of computer science and programming, but it has limits (in terms of programmers creating their programs). Therefore, studies should be conducted to find out whether the problems occur when learning highly extendable programming languages (such as Scratch and Blockly). In this study, the results of pre-service teachers' problem-solving were analyzed in order to understand pre-service teachers' programming learning process. Therefore, how the abstraction should be carried out and what the automation process should be in the actual problem-solving process were not examined. There was no correlation between the difficulty and the results in the programming learning process. Future studies should continue to research these areas. This study can be used as basic research to help develop a programming learning process and support system for pre-service teachers.

This study gathered its results by analyzing the programming learning process of pre-service teachers. Besides, this study provided implications for programming curriculum development and programming language platform development.

References

[1] K. Schwab, *The fourth industrial revolution*, Crown Business, 2017.

[2] D. Mitch, *The role of education and skill in the British industrial revolution*, In The British Industrial Revolution, pp. 241-279. Routledge, 2018.

[3] S. Sentance and A. Csizmadia, "Computing in the curriculum: Challenges and strategies from a teacher's perspective," *Education and Information Technologies*, vol. 22, no. 2, pp. 469-495, 2017.

[4] A. Manches and L. Plowman, "Computing education in children's early years: A call for debate," *British Journal of Educational Technology*, vol. 48, no. 1, pp. 191-201, 2017.

[5] G. Chapman, "Inspire, Innovate, Improve!: What does this mean for CS for All?," *In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* , pp. 1-1, ACM, Mar, 2017.

[6] S. W. Kim and Y. Lee, "Development of a Software Education Curriculum for Secondary Schools," *Journal of The Korean Society of Computer and Information*, Vol. 21, No. 8, pp. 127-141, 2016.

[7] P. J. Rich and C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking*, Springer, 2017.

[8] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33-35, 2006.

[9] I. Milne and G. Rowe, "Difficulties in learning and teaching programming—views of students and tutors," *Education and Information technologies*, vol. 7, no. 1, pp. 55-66, 2002.

[10] B. Özmen and A. Altun, "Undergraduate Students' Experiences in Programming: Difficulties and Obstacles," *Turkish Online Journal of Qualitative Inquiry*, vol. 5, no. 3, pp. 1-27, 2014.

[11] V. G. Renumol, S.Jayaprakash and D. Janakiram, "Classification of cognitive difficulties of students to learn computer programming," *Indian Institute of Technology*, India, 2009.

[12] S. W. Kim and Y. Lee, "The Effect of Robot Programming Education on Attitudes towards Robots," *Indian Journal of Science and Technology*, vol. 9, no. 24, pp. 1-11, 2016.

[13] S. W. Kim and Y. Lee, "Development and Application of Arduino-Based Education Program for High School Students'," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 18, 2017.

[14] S. Kim, S. Han and H. Kim, "Analysis of Programming Processes Through Novices` Thinking Aloud in Computational Literacy Education," *The Journal of Korean association of computer education*, vol. 14, no. 1, pp. 13-21, 2011

[15] J. Sung, S. Kim and H. Kim, "Analysis of Art and Humanity Major Learners` Features in Programming Class," *The Journal of Korean association of computer education*, vol. 18 no. 3, pp. 25-35, 2015.

[16] J. Choi and Y. Lee, "The analysis of learners' difficulties in programming learning," *The Journal of Korean association of computer education*, vol. 17, no. 5, pp. 89-98, 2014.

[17] S. W. Kim and Y. Lee, "Development of TPACK-P Education Program for Improving Technological Pedagogical Content Knowledge of Pre-service Teachers," *Journal of the Korea Society of Computer and Information*, vol. 22, no. 7, pp. 141-152, 2017.

[18] S. W. Kim and Y. Lee, "The Effects of Programming Education using App inventor on Problem-solving Ability and Self-efficacy, Perception," *Journal of the Korea Society of Computer and Information*, vol. 22, no. 1, pp. 123-134, 2017

[19] S. W. Kim and Y. Lee, "A Study of Educational Method Using App Inventor for Elementary Computing Education," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 18, 2017.
[20] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman and Y. Kafai, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60-67, 2009.
[21] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code. Org," *Computers in Human Behavior*, vol. 52, no. 200-210, 2015.
[22] S. A. Ariffin, "Mobile learning in the institution of higher learning for Malaysia students: Culture perspectives," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 1, no. 3, pp. 283-288, *2011*.
[23] Hooshyar & Lim, H. (2018). Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review. *ACM Computing Surveys (CSUR)*, 50(6), 90.
[24] S. N. Razali, F. Shahbodin, M. H. Ahmad and H. A. M. Noor, "Measuring validity and reliability of perception of online collaborative learning questionnaire using rasch model," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 6, pp. 966-974, 2016.
[25] Hooshyar & Lim, H. (2017). A systematic review of data-driven approaches in player modeling of educational games. *Artificial Intelligence Review*, 1-21.
[26] A. A. Patak, H. A. Naim, A. Ma'ruf and M. N. A. Ghafar, "Design and Validation of Online Learning Environment Questionnaire," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 3, pp. 334-338, 2016.
[27] Hooshyar & Lim, H. (2016). Applying an online game-based formative assessment in a flowchart-based intelligent tutoring system for improving problem-solving skills. *Computers & Educatio*n, 94, 18-36.
[28] J. Bennedsen and M. E. Caspersen, *Exposing the programming process. In Reflections on the Teaching of Programming*, pp. 6-16, Springer, Berlin, Heidelberg, 2008.
[29] Hooshyar & Lim, H. (2017). A procedural content generation-based framework for educational games: Toward a tailored data-driven game for developing early English reading skills. *Journal of Educational Computing Research*, 0735633117706909.
[30] M. M. Lehman, "Process models, process programs, programming support," *In Proceedings of the 9th international conference on Software Engineering*, pp. 14-16. IEEE Computer Society Press, Mar, 1987.
[31] Hooshyar & Lim, H. (2017). Development and Evaluation of a Game-Based Bayesian Intelligent Tutoring System for Teaching Programming. *Journal of Educational Computing Research*, 0735633117731872.

APPENDIX A. THE RESULTS OF PROGRAMMING ELEMENT IN CODE.ORG

TABLE I
ANALYSIS OF PROGRAMMING ELEMENTS OF ACCELERATED COURSE IN CODE.ORG

	Task	Sequence			Loop				Condition			Event				
		Move	Turn	Skill	Repeat	Nested repeat	Until & While	Counter	If	If-else	Nested if	Function	Nested function	Function with parameter	Variable	Random number
The Maze	1	O														
	2	O														
	3	O	O													
	4	O	O													
	5	O	O													
	6	O			O											
	7	O	O		O											
	8	O	O		O											
	9	O	O		O											
	10	O					O									
	11	O	O				O									
	12	O	O				O									
	13	O	O				O									
	14	O	O				O		O							
	15	O	O				O		O							
	16	O	O				O		O							
	17	O	O				O		O							
	18	O	O				O			O						
	19	O	O				O			O						
	20	O	O				O			O	O					
The Artist	1	O	O	O												
	2	O	O	O												
	3	O	O	O	O											
	4	O	O	O	O											
	5	O	O	O	O											
	6	O	O	O	O											
	7	O	O	O	O											
	8	O	O	O	O											
	9	O	O	O	O											
	10															
The Artist 2	1	O	O	O	O											
	2	O	O	O	O											
	3	O	O	O		O										
	4	O	O	O		O										
	5	O	O	O		O										
	6	O	O	O	O											
	7	O	O	O		O										
	8	O	O	O		O										
	9	O	O	O		O										
	10	O	O	O		O										
	11															
The Farmer	1	O		O												
	2	O		O												
	3	O		O	O	O										
	4	O	O	O	O	O										
	5	O	O	O	O	O	O									
	6	O		O			O									
	7	O	O	O			O									
	8	O		O	O											
	9	O	O	O			O									
	10	O		O			O		O							
	11	O		O			O		O							
The Artist 3	1	O	O	O	O											
	2	O	O	O								O				
	3	O	O	O	O							O				
	4	O	O	O	O							O				O
	5	O	O	O								O			O	
	6	O	O	O				O				O			O	
	7	O	O	O				O							O	
	8	O	O	O								O			O	
	9	O	O	O	O							O				
	10	O	O	O				O				O				
	11	O	O	O				O							O	

The Farmer 2	1															
	2	O		O	O							O				
	3	O		O	O							O				
	4	O	O	O	O							O				
	5	O		O	O							O				
	6	O		O	O							O				
	7	O		O	O							O				
	8	O		O	O							O				
	9	O		O	O							O				
	10	O		O	O					O		O				
The Artist 4	1	O	O	O	O							O				
	2	O	O	O	O							O				
	3	O	O	O	O							O				
	4	O	O	O	O							O				
	5	O	O	O	O							O	O			
	6	O	O	O	O									O	O	
	7	O	O	O	O							O	O	O	O	
	8	O	O	O	O							O	O	O	O	
	9	O	O	O	O			O				O	O	O	O	
	10															
The Farmer 3	1	O	O	O												
	2	O	O	O												
	3	O	O	O	O											
	4	O	O	O	O		O									
	5	O	O	O	O				O							
	6	O	O	O	O				O							
	7	O	O	O		O						O				
	8	O	O	O	O	O						O				
	9	O	O	O	O			O						O	O	
The Artist 5	1	O	O	O				O							O	
	2	O	O	O				O							O	
	3	O	O	O				O							O	
	4	O	O	O	O									O	O	
	5	O	O	O	O			O						O	O	
	6	O	O	O	O				O					O	O	