Informatics in Education, 2024, Vol. 23, No. 2, 385–408

DOI: 10.15388/infedu.2024.14

Productive Failure-based Programming Course to Develop Computational Thinking and Creative Problem-Solving Skills in a Korean Elementary School

Dagyeom LEE, Youngjun LEE

Department of Computer Education, Korea National University of Education
Cheongju, Republic of Korea
e-mail: gyeomdalee@gmail.com, yjlee@knue.ac.kr

Received: May 2023

Abstract. As our society has advanced in the era of digital transformation, education has been transformed from knowledge-centered to competency-centered to solve future problems in the light of unpredictable changes and events in our lives. Programming education provides the basic knowledge needed, and fosters higher-order thinking skills in the process of generating and converging ideas to solve problems. However, in Korean elementary schools, it is mostly based on a lecture-based instructional design and focuses on knowledge delivery, which has limited the educational effects of programming. However, productive failure (PF) focuses on learning concepts in authentic problems, and lets the students generate different solutions and discuss them in an acceptable environment, with the result that they fail to solve the problem. Therefore, this study developed a PF-based educational program and tested it on sixth-grade students in a Korean elementary school. The results showed that the computational thinking (CT) and creative problem-solving (CPS) skills of the experimental group were significantly greater than those of the control group, with a medium effect size for CT and a high effect size for CPS skills. To generalize the results and increase the applicability, follow-up studies should expand the subject of the study, develop specific teaching guidelines for teachers, and invent various learning problems appropriate to the students' level and different domains of learning.

Keywords: productive failure, programming education in elementary school, computational thinking, creative problem-solving skills.

1. Introduction

The technological advances of the Fourth Industrial Revolution have brought about tremendous changes in all aspects of our lives. Education is also undergoing a transformation from knowledge-centered, where the main focus is on how much knowledge is memo-

rized, to competency-based, where competencies such as creativity and problem-solving are developed (Taguma and Barrera, 2019). This is because advances in technology, such as artificial intelligence, are replacing low-level thinking, such as rote memorization. Computer science education is not only about acquiring the principles and knowledge of computer science and related technologies but also about experiencing the entire process of identifying and solving problems using the knowledge and skills students have learned. In particular, programming education focuses on exploring different ways to solve problems, finding the best solution, and expressing it clearly, which is effective in improving competencies such as computational thinking (CT), logical thinking, critical thinking, and creative problem-solving (CPS) skills (Oldridge, 2017; Shih, 2019; Silapachote and Srisuphab, 2017; Zeng, 2013). As these core competencies are critical to solving the challenges of society now and in the future (ICT, 2020; Taguma and Barrera, 2019), research on effective teaching of IT to improve competencies through programming education has received considerable attention (Papavlasopoulou, 2019).

The educational innovation of shifting from a knowledge-centered curriculum to a competency-centered curriculum has also been tried in the field of computing education around the world. In the United States, the Association for Computing Machinery (ACM), IEEE Computer Society, and the Association for Information Systems (AIS), the leading academic organizations in computer science, have developed a competency-based curriculum called Computing Curricula 2020 (CC2020). CC2020 defines competencies as the sum of knowledge, skills, and dispositions needed to solve authentic problems, and provides a systematic curriculum and content framework for developing these competencies (Han Sung, 2021). In 2015, the South Korean government also developed a revised, competency-based curriculum. This curriculum (Ministry of Education, 2015a) and other competency-based curricula all emphasize that the competencies should be transferable to the real world beyond the classroom. For this purpose, researchers have carefully selected content that can foster competencies and have designed systematic curricula. However, questions remain about whether the appropriate instructional design has been implemented according to the curricula. In South Korea, instructional materials such as textbooks are mainly based on direct instruction and teacher-centered demonstrations, a one-way mimetic method that limits the development of students' competencies (Youngsik, 2018). To reflect on the current instructional design and improve it, it is necessary to conduct a study to explore an alternative instructional design suitable for the competency-based curriculum.

Therefore, this study developed a productive failure (PF)-based programming educational course and tested it to effectively improve CT and CPS skills in a Korean elementary school. In many works, these competencies are identified as core competencies in informatics (Ritter and Standl, 2023) and a problem-solving process rather than programming skills, as these are required for future human resources (Pewkam and Chamrat, 2022). Thus, two research questions (below) guide this study.

RQ1: What impact does the PF-based programming course have on CT among students in a Korean elementary school?

RQ2: What impact does PF-based programming course have on CPS skills among students in a Korean elementary school?

This research is a semi-experimental study with a sample of 69 students from the sixth grade of an elementary school: the experimental group (35 students) and the control group (34 students). The development of CT and CPS skills was measured before and after the courses. We hypothesized that the experimental group would outperform while solving the problems and achieve better than the control group in CT and CPS skills.

This paper is organized as follows: Section 2 briefly demonstrates the trends of programming education in Korea and PF as an alternative instruction. Section 3 describes how to compose the procedures of the program course and details the methodology. Section 4 presented the results according to the research questions. Finally, the conclusions from this study and discussions are presented in Section 5.

2. Background

In this section, we present the current state of programming education in elementary schools in Korea, based on an analysis of textbooks and existing research. We propose PF as an instructional design to improve this education, as follows: the emergence of PF, the pedagogical principles of PF, and why PF is suitable for programming education in elementary schools among instructional designs that are based on teaching via problem solving.

2.1. *Programming Education Trends in Elementary Schools in South Korea*

In 2014, the South Korean government released the Strategy Report for Realizing a Software (SW) – Centered Society to promote SW education in elementary schools (Ministry of Education, 2015a), and programming education was officially introduced into the elementary school curriculum in the 2015 Revised Curriculum. It was organized in the "Practical Arts" curriculum for more than 17 hours per year for grades 5–6 (Ministry of Education, 2015b). The teaching and learning directions set out in the curriculum call for the use of educational tools that consider the developmental level of elementary school students and teach them to apply what they have learned in the real world.

However, the textbooks, which are general teaching materials for implementing the curriculum in class, present the tasks and instructional design in the class in a way that limits their ability to achieve the goal of a competency-based curriculum. Typical programming instructional approaches presented in textbooks are as follows. The instructor teaches programming knowledge and structures at the beginning of the class and then presents a simple programming module. At this point, the textbook already has a sample code, and the student solves the problem by copying the sample code after the instructor's demonstration. Finally, students are given an exercise to practice what they have learned, and they create a program that solves the problem. This instructional design consists mainly of direct instruction and demonstration by the teacher. The goal of the programming class implemented by this design does not focus on developing

competencies such as CT and problem-solving skills, but rather on gaining experience in using programming languages and the grammar of block programming languages (Youngsik, 2018).

This instructional design does little to improve competencies and enable the transfer of skills to the real world. First, because the instructor teaches programming knowledge and structures early in the course, students are not given the opportunity to discover and organize their knowledge on their own. In addition, the problems in the textbook are authentic and presented in a well-structured way. While this may help in solving the tasks in class, as they are clearly stated so that students can identify the key conditions of the task, simply applying what they have learned makes it difficult for students to acquire competencies such as CT and transfer skills. This is because most real-world problems are unstructured and require students to use abstract thinking to identify problems and discover key conditions. Finally, activities in which the teacher demonstrates a model answer and students copy it prevent the emergence of divergent thinking that generates multiple problem solutions. An alternative instructional design is needed to overcome these limitations.

Research on programming education for elementary school students has also identified the need for further research to develop an effective instructional design of programming education. Lee *et al.* (2022) conducted a study to analyze the current state of programming education research in elementary schools. They examined which competencies and instructional designs were related. They found only some 10 studies comparing instructional designs or strategies to improve competencies. By contrast, approximately 90 studies compared and analyzed the educational effects of different educational tools, such as physical computing and programming languages. Therefore, there is a need for research to explore instructional designs for effective programming education in elementary schools.

2.2. *Pedagogical Design and Theoretical Foundations: PF*

Teaching via problem solving is an instructional design based on constructivism that effectively guides the development of students' thinking skills by gradually allowing the students to form core concepts as they solve problems, including problem-based learning and project-based learning. By presenting students with authentic problems and letting them experience the entire process, this instructional design is more engaging than traditional lectures and is effective in developing problem-solving skills. However, teaching via problem solving requires a lot of class time, as pointed out by Kerrigan *et al.* (2021). Currently, only 16 hours are allocated to teach Informatics related content (Software, programming, physical computing) in the curriculum of all grades of elementary school in Korea. Among them, around 8 hours are allocated to teach programming and it is evidently not enough to teach programming Lee *et al.* (2022). PF deals with both the benefits of teaching via problem solving and the problem of insufficient time. This can reduce the amount of direct teacher instruction within the allocated time, freeing up time for student-led activities.

PF originated as an attempt to analyze the role of failure in learning, as opposed to the typical problem-solving instructional design. Kapur (2008) found that, among two groups of students with the same learning objectives, the group of students who reached an impasse in the problem-solving process and failed the task performed better than the group of students who solved the problem with direct instruction from the teacher but did not reach an impasse. A similar result was also found in a study by Schwartz and Martin (2004), who reported that students who learned concepts but failed to generate standard solutions through direct instruction in a classroom achieved statistically higher outcomes than students who succeeded in arriving at solutions. These findings contradict the traditional belief that the experience of successfully solving problems presented in class enhances learning, and has stimulated research into how failure can enhance learning.

Drawing on pedagogical theory, Kapur (2008) argued that failure in class can be productive for learning if it has the following factors, the first being impasse-driven learning (VanLehn *et al.*, 2003). An impasse is a state of being stuck in a problem-solving process. When students reach an impasse, they try to solve the problem in different ways to get through it, thereby experiencing and analyzing more of the structure of the problem. Piaget (1964) understood it as a process that can occur in the learning process. He saw learning as a process of assimilation or adaptation of a person's internal schema to fit the external environment. diSessa (2006) also argued that learners can experience an impasse, in which they realize that their knowledge differs from standard concepts or solutions by experiencing a mismatch between the external environment and their internal schema. He explained that learners continue to learn to overcome this impasse.

The second factor of PF is an ill-structured problem. Ill-structured problems contrast with well-structured problems, which are the types of examples and exercises usually presented in textbooks. Well-structured problems make it easy for students to identify the core concept in the question and what knowledge and skills are needed to solve the problem. They also have limited problem-solving space, so that students can perform within it. And they are less relevant to students' lives because the variables in the problem are manipulated to facilitate solutions. These manipulations are effective for solving unit tasks but have the limitation of making it difficult to transfer learning outcomes and skills to the real world. However, ill-structured problems present students with authentic problems that do not limit their problem-solving space and performance (Kapur, 2010). When solving ill-structured problems, students can see the relevance of problems to their lives and become engaged, knowing that they have to analyze and identify variables in problems. Marton (2006) explains the pedagogical benefits of ill-structured problems through the theory of the retrospective transfer effect. This effect consists of internal and external transfer effects. Students experience an internal transfer in that they learn more about problem-solving structures as they identify, analyze, and solve ill-structured problems. After that, they experience an external transferring-out effect in that they apply the problem-solving structures they used to solve ill-structured problems when solving authentic problems (Bransford and Schwartz, 1999). PF facilitates the internal and external transfer of learning by iteratively presenting and solving ill-structured problems. In addition, target concepts and problem-solving structures can be effectively learned by generating a lot of solutions when experiencing an impasse and challenge and by discussing ways to overcome them.

PF also focuses on having students generate a variety of problem solutions. In the classroom, the teacher does not take the lead in providing cognitive scaffolding from the outset, but rather plays a dispositional supportive role in creating an open atmosphere that allows for failure. However, there is no explicit teacher guidance. The comparison of the solutions generated at the end of the lesson allows for elaborating on the target concept and connecting it to prior knowledge.

The pedagogical benefits of PF have been validated by empirical studies (DeCaro and Rittle-Johnson, 2012; Kapur, 2014). First, PF activates prior knowledge and helps learners to identify gaps in the target concept. Second, learners are self-regulated and willing to continue learning throughout the PF lesson to fill the gaps. Third, it helps students to recognize the limits of their prior knowledge by allowing them to experience the process of generating solutions before teacher guidance. Fourth, the activity of comparing, contrasting, and discussing solutions with students and a teacher helps them to better identify important features of the target concept.

3. Methodology

3.1. *Research Design*

To achieve the objectives of this study, we defined the following research questions and hypotheses:

RQ1: What impact does the PF-based programming course have on CT among students in a Korean elementary school?

- **Hypothesis 1.0:** There is no evidence that the PF-based programming course can impact on students' CT in a Korean elementary school.
- **Hypothesis 1.1:** The PF-based programming course can impact on students' CT in a Korean elementary school.

RQ2: What impact does PF-based programming course have on CPS skills among students in a Korean elementary school?

- **Hypothesis 2.0:** There is no evidence that the PF-based programming course can impact on students' CPS skills in a Korean elementary school.
- **Hypothesis 2.1:** The PF-based programming course can impact on students' CPS skills in a Korean elementary school.

This study adopted a control group pre-test–post-test design on a quasi-experimental basis, as shown in Table 1. It consists of two groups of students, the experimental and the control group. The experimental group received the PF-based programming course, while the control group received an exemplary instructional course presented in a textbook that mainly used lectures and direct teaching methods. CT and CPS skills were measured before and after the program to see if there was a statistical difference between the two groups.

Table 1
Research design

Groups	Independent Variables	Dependent Variables
Experimental	PF-based PC	CT, CPS
Control	PC	CT, CPS

Some variables are part of this design:

- **Programming course (PC):** This independent variable represents the programming course in an elementary school;
- **Productive failure (PF):** We considered the robotics course proposed and applied it to the experimental group as an independent variable;
- **CT skills (CT):** Performance in the pencil-paper test that explores CT skills;
- **CPS skills (CPS):** Responses in a self-report evaluation survey that explores CPS skills.

3.2. *Profile of Participants*

In this study, we considered a sample of 69 students from the sixth grade in a city in South Korea, with a total of four classes selected by convenience sampling. Classes in elementary schools in Korea are organized according to the results of the overall academic achievement of the previous school year, so that the average is evenly distributed among the classes. Therefore, we randomly divided four classes into two. The experimental group consisted of 35 students (17 males, 18 females), and the control group consisted of 34 students (16 males, 18 females), as shown in Table 2.

3.3. *PF-based Programming Course*

The ADDIE model (analysis, design, development, implementation, evaluation) is a widely used curriculum design framework that provides a systematic, sequential approach(Schlegel, 1995). There are five phases in the ADDIE model. In the analysis stage, we extracted the achievement standards related to programming learning present-

Table 2
Profile of participants

Groups	Number of Classes	Gender		Number of students
		Male	Female	
Experimental	2	17	18	35
Control	2	16	18	34
Total	4	33	36	69

Table 3
Course development phases according to the ADDIE model

Stages	Contents
Analysis	• Exploring the programming-related standards in the 2015 Revised Curriculum • Analyzing instruction models • Analyzing research trends in programming education for elementary school students
Design	• Organizing learning objectives and sequences based on the achievement standards • Selecting an educational programming language (EPL) • Selecting tools to measure CT and CPS
Development	• Developing PF-based instructional materials and organizing content • Developing the process of the program • Expert review of the program
Implementation	• Applying the program • Examining CT and CPS tests
Evaluation	• Analyzing pre- and post-test results data for each group • Analyzing the effectiveness of the training program and identifying areas for improvement

ed in the 2015 Revised Curriculum as the objectives of the course. We also analyzed the characteristics and procedures of the PF model to apply it appropriately. In addition, we explored the existing research trends in programming education for elementary students to reflect the development stage of students in programming education. In the design phase, the objectives of the course were created based on the performance standards selected in the analysis phase. We selected teaching tools and competency measurement instruments. In the development stage, we developed the curriculum of the course based on the PF model. The curriculum was reviewed and revised by computer science education experts and teachers for content validity. Then, materials for teachers and students were developed for the application of the course in the field. In the implementation phase, the developed course was applied to the research subjects. We measured the competencies before and after the course and collected the data. Finally, in the evaluation stage, the data were analyzed to verify the effectiveness of the course and to reflect on it for improvement. This study was conducted according to the procedure outlined in Table 3, and the details of the analysis, design, and development stages are described in each chapter. The details of the implementation and evaluation phases are described in the results and conclusions.

3.3.1. *Analysis*

To design the course, we first analyzed the achievement standards related to programming education in the elementary school curriculum in South Korea. Programming-related achievement standards are shown in Table 4. Among them, we selected the achievement standard “Understand the structure of sequence, selection, and iteration in the process of creating a program to solve a problem” as the objective of the course because it contains programming-related knowledge and structures and improves the transfer of skills to the real world. Since only eight hours are allocated in the curriculum to achieve this objective, we decided that the course also should be eight periods long.

Table 4
Achievement standards of programming education in the 2015 Revised Curriculum

Criteria	Achievement Standards
Understanding Software (SW)	• Identifying examples of software applications and understanding their impact on our lives.
Procedural Problem Solving	• Thinking about and applying the sequence of problem solving by procedural thinking.
Elements and Structure of Programming	• Experiencing the basic programming process using programming tools. • Designing a simple program that inputs data, performs necessary processing, and outputs results. • Understanding the structure of sequence, selection, and repetition in creating programs to solve problems.

Second, we analyzed the PF model. The PF model consists of two phases: "Generation & Exploration" and "Consolidation & Knowledge Assembly". In the Generation & Exploration phase, ill-structured problems are appropriate to motivate students and activate their prior knowledge. Groups of three or four students have a discussion expressing and explaining the important features of the problem and finding the target concept in different ways. In the Consolidation & Knowledge Assembly phase, pupils compare the solutions they come up with, exploring their similarities and differences or their practicalities and the limitations of solutions. Finally, important features of the target concept are identified by comparing them with the teacher's standard solution. These are combined to form the target concept clearly (Kapur, 2010).

However, to apply the PF model to programming content, it needs to be revised with specific learning activities, considering the characteristics of programming education content. It must also reflect the developmental stage of elementary school students, as the PF model has mainly been studied for middle school students and above. Therefore,

Table 5
PF-based programming instructional model

Steps		Activity
PF	DDD	
Generation & Exploration	Discovery	• Presenting the problem scenario • Identifying the key element of the goals • Exploring factors that are important to solving the problem
	Design	• Identifying variables and behaviors needed to solve a problem • Generating multiple solutions • Describing and evaluating solutions through group discussion
	Development	• Implementing the best solution determined through discussion • Posing a "what if" problem (for those who succeed in solving the problem)
Consolidation & Knowledge Construction	Evaluation & Feedback	• Presentation and feedback on each group's solution • Finding key features by comparing to standard solutions • Structuring the learning contents

we combined the PF model with the discovery-design-development (DDD) teaching model, which has been empirically proven to be an effective instructional design in programming education for elementary school students. DDD is also theoretically based on the constructivism perspective. Students take the initiative in learning programming, and the teacher acts as a guide. The results of applying this model to elementary school students have shown that it is effective in developing defining competencies, such as learning motivation and confidence, as well as CT skills (Soojin, 2017).

Therefore, to develop a PF-based programming course in this study, we set up the model as shown in Table 5, taking into consideration the developmental stage of elementary school students and content characteristics.

3.3.2. *Design*

The Lesson plan is as follows: In PF-based program, the teacher presented ill-structured problems in the form of scenarios, which is very close to the student's life. Students were motivated by recognizing that the problems presented in the scenario were often in their school lives, and that solving this problem would make their school life more convenient. When making problem-solving program, we facilitate cooperative activities through intervening and counseling. The group of three to four students was necessarily deadlocked in the process of solving the problem. Although at what stage the deadlock occurred was different for each group, it was observed that all groups reached at least two deadlocks. We presented the following guidelines based on productive failure to the group who was in the deadlock of learning or asked for help: all members were asked to create at least one solution. Rather than choosing one of the generated solutions, we made them organize the similarities and differences of their solutions and synthesize them to derive a 'group solution'. It was observed that these guidelines increased the members' tendency to cooperate. This led to coding activity which was easy to do individually, leading to cooperative coding like paired programming. The control group's lesson plan was designed similarly, except that the productive failure teaching method was not applied. The difference between them is that first, the program of the control group did not present an ill-structured problem. Instead, the program was produced based on the topics presented in the textbook. The topics presented in the textbook were verified to take into account the student's level and interest, but they were structured and not very close to the students' real life as the problems presented in the PF-based program. Second, in group activities, we did not provide guidelines based on productive failure. Students naturally performed activities by only a few students presenting a solution or simply selected and coded one of the solutions.

The topics were selected to meet the course's objective. According to the design principles of the PF model, the topics should be authentic problems and presented in an ill-structured way. The course consisted of four topics, with two hours per topic. The first topic, "What to do during the morning in school in order," is an unplugged programming task that uses procedural thinking as prior knowledge to suggest the order of tasks to be performed during morning activities. This topic contains a problem that cannot be solved only by prior knowledge and requires more knowledge. Students should develop the most efficient sequence of tasks in a limited time. The second

topic, "Create a program to write a to-do list," is designed to implement sequential programming. The problem situations were designed to reflect their educational environment, in which students were given handheld devices. The students were motivated and actively engaged in class because they could create a program that could be used on their own portable devices. The third topic was "Create a program to teach disaster evacuation tips." As June is a time when natural disasters, such as typhoons and flooding, could often occur in South Korea, it was an appropriate time to present this as a problem situation. In the process of implementing a program on how to prepare for disasters, the students felt the lack of prior knowledge and the need for a new structure, such as a repetitive structure, and actively participated in learning to meet these needs. The last topic was "Create a program to inform about the availability of outdoor physical classes." Outdoor physical education is not possible when the temperature is too high or the concentration of fine dust is high. By creating a program that informs students about the availability of outdoor classes based on weather information, they can learn about the conditions. To reflect the real-world situation of the students, the program is based on the weather and fine dust information in the area where the school is located.

The programming language to be used in this course is a block-based programming language considering the developmental characteristics of elementary school students. Scratch 3.0 and Entry are block-based programming languages commonly used in Korean elementary schools. Scratch has been used the most in Korean programming education for elementary school students, but the usage rate of both languages has been similar since Entry was introduced in textbooks of the 2015 Revised Curriculum, and it is commonly used in the classroom (Lee *et al.*, 2022). We decided to use Entry in the experimental and control group, as all textbooks for the 2015 Revised Curriculum in South Korea selected Entry as their programming language (as shown in Fig. 1).

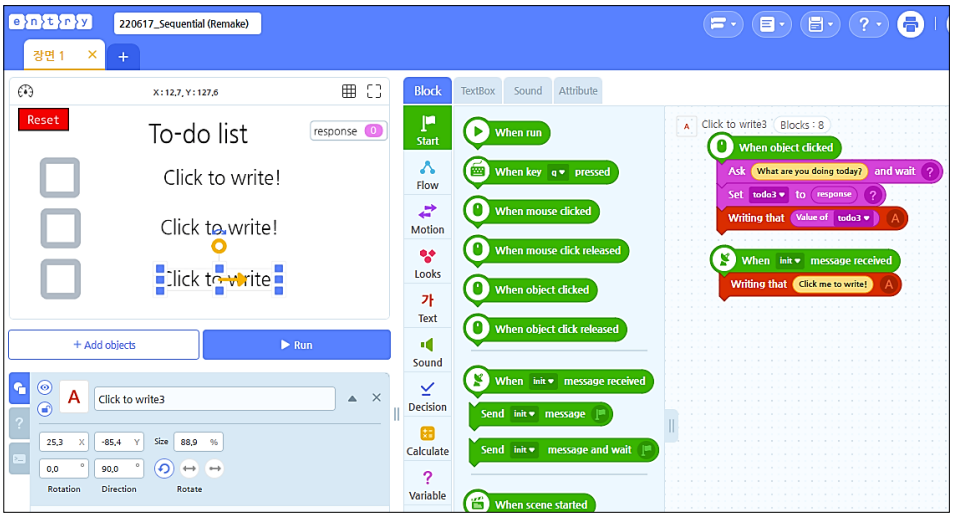

Fig. 1. Screen capture of the workspace in 'Entry'.

3.3.3. *Development*

The program developed in this study consisted of eight hours for both the experimental and the control groups. The contents of the activities in the two courses are compared and presented in Table 6. We presented ill-structured problems, standard solutions for the PF-based course [see Appendix A].

To confirm the validity of the course developed above, the content validity of the course was tested by computing education experts and teachers. The test questions of the contents were designed to judge the appropriateness of learning topics, learning objectives, content organization, the teaching plan, and the relevance and quantity of the course contents. The participants responded to four-point Likert scale questions and open-ended questions. The content validity ratio (CVR) was verified based on the percentage of respondents who answered the questions as required (3 and 4 responses based on a four-point Likert scale) among all respondents. Experts who participated in the test are a doctor of education majoring in elementary computing education, three doctoral students majoring in elementary computing education, three master's students, one master's student majoring in informatics education for the gifted, and one master's student majoring in computing education. The results of the expert review of the education program are shown in Table 7, and it was determined that content validity was ensured because it was over the minimum value of 0.78.

The open-ended responses to the program from the expert group and the modifications reflecting them are shown in Table 8.

The implementation and evaluation steps taken in this study are presented in the Results and Conclusions chapter.

Table 6

Activities of the courses by the groups

Periods	Phases	Experimental group	Phases	Control group
1	Best solution for problem solving	What to do during the morning in school	Sequential structure	Create a program to meet a figure athlete character
2				Create a program that responds to a figure's words
3	Sequential structure	Create a program to write a to-do list	Repetition structure	Create a program to identify the capital of your country
4				Create a program that makes a sound for correct or incorrect answers
5	Repetition structure	Create a program to teach disaster evacuation tips	Conditional structure	Create a program to make a flower with four petals by stamping petals at regular intervals
6				Create a program to make a flower with six petals by stamping at regular intervals
7	Conditional structure	Create a program to inform about the availability of outdoor classes	Best solution for problem solving	Create a robot vacuum cleaner that will change direction when it hits a wall
8				Create a robot vacuum cleaner that avoids obstacles on the floor

Table 7

Results of the expert review

Criteria	Contents	CVR
Learning topic	Appropriateness of learning topics	1
	Reflective of performance standards	1
Learning Objectives	Appropriateness of learning objectives	1
	Clarity of learning objectives	1
	Level of learning objectives	1
Construction	Appropriateness of content organization	1
Contents & Methods	Appropriateness of topic and content	1
	Variety of learning methods	1
	Fostering creativity	1
Relevance	Relevance to CT	1
	Connections between content	1
	Relevance to the authentic problems	1
Quantity	Appropriateness of content	1

Table 8

Results of the expert review (open question)

Feedback from the experts	Modification
Clear statement of the objective	Revised the learning objective statement to "I can create a program to solve a problem."
Strategy to check prior knowledge	Added the teaching strategy to remind students of their prior knowledge.

3.4. *Instruments*

To measure CT skills, we used the CT Test for Elementary School Students developed by Kim (2019). They analyzed the achievement standards related to CT skills in the 2015 Revised Curriculum in South Korea. She developed questions to assess CT skills, including abstraction and automation skills, as shown in Table 9. This test is a summative assessment consisting of short-answer and long-answer questions. The content validity index (CVI) of this test was 1.0, which is highly valid. In addition, she measured a difficulty coefficient to examine the appropriateness of the content and difficulty level for sixth-grade learners. Generally, a difficulty coefficient of 0.3 to 0.8 means that a percentage of the number of correct answers given by the subjects is considered an appropriate level of difficulty (Cangelosi, 1990). The difficulty coefficient of the test indicated that the test instrument was generally moderately difficult; out of 18 items, one item was difficult, 14 items were moderately difficult, and three items were easy.

To measure the CPS skills of elementary school students, we used a test developed by the research team at the Psychology Research Center of Seoul National University

Table 9
Contents of the CT skills test

Criteria	Standards	CT factors	Question numbers
Abstraction	• Determine what information is necessary to solve a problem and what information is unnecessary.	Analysis, decomposition, abstraction	1, 2, 3, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18
	• Analyze provided data to discover relationships or rules between data.	Analysis, decomposition, abstraction	
	• Represent problem-solving processes procedurally.	Abstraction, algorithm	
Automation	• Understand sequential, selection, iterative, and parallel structures.	Automation, parallelizing	3, 5, 6, 7, 8, 9, 13
	• Represent the problem-solving process in a form that a computer can understand.	Automation	
	• Interpret algorithms or instructions to predict outcomes.	Simulation	

Table 10
Results of the reliability of the CPS skills test

Factors	Question number	Cronbach's α	Number of questions
Understanding and mastery of knowledge and skills	1, 2, 3, 4, 5	0.770	5
Divergent thinking	6, 7, 8, 9, 10	0.749	5
Critical/logical thinking	11, 12, 13, 14, 15	0.849	5
Motivation	16, 17, 18, 19, 20	0.832	5
Total		0.899	20

(Cho, 2002). This test is designed to test students' CPS skills in school environments. The test has four sub-factors of CPS skills: Understanding and mastery of knowledge and skills, Divergent thinking, Critical and logical thinking, and Motivational factors. The test consists of 20 questions, five questions for each factor, and is designed to be answered on a five-point Likert scale. It has been used in several studies to test the CPS skills of elementary school students in Korea, and its reliability has been verified with a Cronbach's alpha of 0.899. The item structure and reliability of the CPS skills test are shown in Table 10.

4. Results

In order to answer the research questions defined, we analyzed the results obtained from the data of the experimental and the control groups. We present the data analysis procedures and results in relation to each of the research questions.

4.1. *Data Analysis Procedures*

The data collected in this study were analyzed using IBM SPSS Statistics 26, i-statistics 2.01, in the following ways. First, we analyzed the statistical assumptions of normality and homogeneity of variance using the Shapiro-Wilk and Levene hypothesis tests.

We compared the post-test results of the experimental and control groups after the programming course using an independent samples t-test to see if the difference in the means was statistically significant. A paired-samples t-test was conducted to determine the degree of change in CT and CPS within each group. Cohen's d, an effect size value, was calculated based on the group size, mean, and standard deviation to determine the effect size of PF-based programming course on improving CT and CPS in elementary school students.

4.2. *Statistical Assumption*

Although the class organization in Korean elementary schools is relatively equal in terms of the average results of academic achievement assessments, we analyzed the statistical assumption of normality and homoscedasticity using the Shapiro-Wilk and Levene hypothesis tests, respectively, considering a significance level of $\alpha = 0.05$ in the students' CT and CPS skills by group.

We measured the CT and CPS skills of the two groups to ensure that they met the normality. The Shapiro-Wilk test was performed on a small sample of fewer than 50 students in each group, and the results are shown in Table 11. Only the probability of significance of the experimental group's CPS skills was 0.009 ($p < 0.05$), which did not seem to meet normality. However, we consider it to meet the normality assumption because the result of the Kolmogorov-Smirnov test showed that normality was met (Yoo, 2021), and the kurtosis and skewness did not exceed 2 (Snedecor and Cochran, 1980).

We also measured the homoscedasticity of CT and CPS skills to ensure that there were no significant differences between the two groups. As a result, the CT and CPS

Table 11

Normality test results of the experimental and control groups

Factors	Groups	Kolmogorov-Smirnov			Shapiro-Wilk			Kurtosis	Skewness
		Statistic	df	P	Statistic	df	p		
CT (pre)	*Exp.*	*0.087*	*35*	*0.200*	*0.965*	*35*	*0.325*	*-0.375*	*0.690*
	Con.	0.144	34	0.071	0.954	34	0.161	-0.486	-0.517
CPS (pre)	Exp.	0.142	35	0.072	0.913	35	0.009*	-1.157	1.626
	Con.	0.119	33	0.200	0.960	34	0.256	-0.112	-1.053

*$p < 0.05$

Table 12
Homoscedasticity test results of the experimental and control groups

Factors	Groups	N	M	SD	Levene's Equal Variance Test		t	p
					F	p		
CT	Exp.	35	13.80	4.418	1.950	0.167	1.656	0.102
	Con.	34	11.85	5.321				
CPS	Exp.	35	3.254	0.603	0.740	0.393	0.081	0.935
	Con.	33	3.242	0.598				

skills of the two groups met the homoscedasticity of variance with the population, as shown in Table 12, and there was no significant difference in the mean of the skills between the two groups ($p > 0.05$).

4.3. *RQ1: What Impact does the PF-based Programming Course Have on CT among Students in a Korean Elementary School?*

We measured the CT skills of each group after applying the educational program, and analyzed the results of the independent t-test, shown in Table 13. The mean of the experimental group was higher than the control group, and the difference between the groups was statistically significant ($p < 0.05$). We also found that the experimental group scored higher than the control group on both abstraction and automation skills, and the difference between the two groups was statistically significant. Thus, we concluded that the CT skills of elementary school students who received the PF-based programming course improved significantly compared to the CT skills of elementary school students who received programming education based on traditional lectures and direct instruction.

To measure the change within the group, the pre-test and post-test results were analyzed using a paired samples t-test. The results showed that the experimental group

Table 13
Results of the independent sample t-test of CT skills

Skills	Group	N	M	SD	Levene's Test		t	p
					F	p		
Abstraction	Exp.	35	11.29	2.976	0.854	0.359	2.107	0.039*
	Con.	34	9.74	3.136				
Automation	Exp.	35	5.09	1.721	0.023	0.881	2.157	0.035*
	Con.	34	4.21	1.666				
CT	Exp.	35	16.37	4.473	0.291	0.591	2.263	0.027*
	Con.	34	13.94	4.445				

*$p < 0.05$

Table 14
Results of the paired sample t-test of CT skills in the experimental group

		Exp.	N	M	SD	t	p
CT		Pre-	35	13.800	4.418	4.47	0.000***
		Post-	35	16.371	4.473		
Subfactors	Abstraction	Pre-	35	9.629	3.011	4.11	0.000***
		Post-	35	11.286	2.976		
	Automation	Pre-	35	4.171	1.654	3.75	0.000***
		Post-	35	5.086	1.721		

$^{***}p < 0.001$

Table 15
The results of the paired sample t-test of CT skills in the control group

Competencies		Con.	N	M	SD	t	p
CT		Pre-	34	11.853	5.321	2.78	0.004**
		Post-	34	13.941	4.445		
Sub-skills	Abstraction	Pre-	34	8.265	3.941	2.66	0.006**
		Post-	34	9.735	3.136		
	Automation	Pre-	34	3.588	1.672	2.05	0.024*
		Post-	34	4.206	1.666		

$^{*}p < 0.05$ $^{**}p < 0.01$

improved their CT skills, including abstraction and automation skills, as shown in Table 14. The control group also improved their CT skills, as shown in Table 15, which means both programs had the effect of improving CT. This is consistent with existing research results showing that programming course improves CT. However, when comparing the results between the groups in the previous section, the experimental group was statistically significantly higher than the control group in terms of the degree of improvement in CT skills. This difference can be attributed to the instructional design between the two groups and confirms that the PF instructional design is more effective in programming education.

4.4. *RQ2: What Impact does PF-based Programming Course Have on CPS Skills Among Students in a Korean Elementary School?*

We analyzed the data obtained by measuring the CPS skills of the two groups using an independent samples t-test, and the results are shown in Table 16. We found that the mean CPS score of the experimental group was higher than that of the control group, and the difference was statistically significant ($p < 0.001$). We also found that the experimental group scored higher than the control group on all four sub-skills of CPS, and the difference in scores was statistically significant. Thus, we concluded that the CPS

Table 16
Results of the independent sample t-test of CT skills

Skills	Group	N	M	SD	Levene's Test		t	p
					F	p		
Understanding and mastery of knowledge and skills in a specific field	Exp.	35	3.583	0.728	0.034	0.854	2.309	0.024*
	Con.	33	3.164	0.769				
Divergent thinking	Exp.	35	3.514	0.685	0.783	0.379	2.494	0.015*
	Con.	33	3.085	0.735				
Critical/logical thinking	Exp.	35	3.983	0.608	0.538	0.466	2.407	0.019*
	Con.	33	3.624	0.628				
Motivation	Exp.	35	4.200	0.531	1.931	0.169	5.790	0.000***
	Con.	33	3.346	0.681				
CPS	Exp.	35	3.820	0.522	0.159	0.691	4.096	0.000***
	Con.	33	3.305	0.515				

*$p < 0.05$ ***$p < 0.001$

skills of elementary school students who received the PF-based programming course were significantly improved compared to those who received the programming course based on traditional lectures and direct instruction.

The pre-test and post-test results were analyzed using a paired samples t-test to investigate the changes within the group. The results of the experimental group are shown in Table 17. We found that the scores of both CPS and sub-skills improved, and the improvement was statistically significant. The results for the control group are shown in Table 18, which showed a slight increase in CPS compared to before the program, but not a statistically significant increase from before to after the course. When looking at the sub-skills, the divergent thinking score decreased, and only the critical/logical thinking factor showed a statistically significant increase. The pre- and post-test scores

Table 17
Results of the paired sample t-test of CPS skills in the control group

Competencies		Con.	N	M	SD	t	p
CPS		Pre-	33	3.242	0.598	0.83	0.207
		Post-	33	3.305	0.515		
Sub-skills	Understanding and mastery of knowledge and skills in a specific field	Pre-	33	3.097	0.773	0.68	0.251
		Post-	33	3.164	0.769		
	Divergent thinking	Pre-	33	3.115	0.773	-0.26	0.397
		Post-	33	3.085	0.735		
	Critical/logical thinking	Pre-	33	3.461	0.729	1.72	0.048*
		Post-	33	3.624	0.628		
	Motivation	Pre-	33	3.297	0.786	0.36	0.360
		Post-	33	3.345	0.681		

*$p < 0.05$ **$p < 0.01$

Table 18
Results of the paired sample t-test of CPS skills in the experimental group

Competencies		Exp.	N	M	SD	t	p
CPS		Pre-	35	3.254	0.603	5.36	0.000***
		Post-	33	3.820	0.522		
Sub-skills	Understanding and mastery of knowledge and skills in a specific field	Pre-	35	3.023	0.689	4.27	0.000***
		Post-	33	3.583	0.728		
	Divergent thinking	Pre-	35	3.371	0.805	5.96	0.000***
		Post-	33	4.200	0.531		
	Critical/logical thinking	Pre	35	3.163	0.780	2.77	0.004**
		Post	33	3.514	0.685		
	Motivation	Pre	35	3.480	0.731	3.85	0.000***
		Post	33	3.983	0.601		

***p < 0.01 ***p < 0.001*

changes of both groups suggest that the PF-based programming course was effective in improving CPS. However, the change in the control group showed a slight increase in CPS but a decrease in divergent thinking.

4.5. *Effect Size*

Cohen's d-values were calculated to present the effect of the PF-based program on the improvement of each competency. Cohen's d-value is a statistical number that measures a continuous variable in two independent groups and indicates the magnitude of the treatment effect in the experimental group through the difference in the mean, which can be explained by supplementing the *p*-value (Lee, 2016). Since both groups satisfied homoscedasticity of variance, the standardized mean difference was calculated to present the effect size as Cohen's d-value. As a result, the effect size of the PF-based programming course is a medium effect size, as shown in Table 19.

Table 19
Effect size of the PF-based programming course on CT skills

Factors		Group	N	Post-test score		d
				M	SD	
CT		Exp.	35	16.371	4.473	0.54
		Con.	34	13.941	4.445	
Sub-skills	Abstraction	Exp.	35	11.286	2.976	0.51
		Con.	34	9.735	3.136	
	Automation	Exp.	35	5.086	1.721	0.52
		Con.	34	4.206	1.666	

Table 20
Effect size of the PF-based programming course on CPS skills

Factors		Group	N	Post-test score		d
				M	SD	
CPS		Exp.	35	3.820	0.522	0.99
		Con.	33	3.305	0.515	
Sub-skills	Understanding and mastery of knowledge and skills in a specific field	Exp.	35	3.583	0.728	0.56
		Con.	33	3.164	0.769	
	Divergent thinking	Exp.	35	4.200	0.531	0.61
		Con.	33	3.085	0.735	
	Critical/logical thinking	Exp.	35	3.514	0.685	0.58
		Con.	33	3.624	0.628	
	Motivation	Exp.	35	3.983	0.601	1.40
		Con.	33	3.345	0.681	

The effect size of the PF-based programming course on improving CPS among elementary school students is shown in Table 20. As shown in the table, the effect size is high. Examining sub-skills, we found that the effect size is very high, especially for the "motivation" skill.

5. Discussions

5.1. *Conclusions*

In this study, we proposed the PF-based programming course to enhance students' competencies, to make them become full members of future society and to transfer skills to the real world. We systematically developed a programming course based on the idea of PF, following the ADDIE model steps, and ensured its content validity through expert review. We also engaged sixth-grade students in the course to examine its ability to improve their CT and CPS skills.

Considering research questions 1 (RQ1), we confirmed that students who received the PF-based programming course cultivated their CT skill better than those who received the traditional lecture and direct instruction-based course. Moreover, the PF instructional design showed a medium effect size for improving students' CT skills. Therefore, we dismissed Hypothesis 1.0 (There is no evidence that the PF-based programming course can impact on students' CT in a Korean elementary school.) and adopted Hypothesis 1.1 (The PF-based programming course can impact on students' CT in a Korean elementary school.).

Regarding research questions 2 (RQ2), we found that students who received the PF-based programming course improved their CPS skill better than those who received the traditional lecture and direct instruction-based course. Moreover, the PF instruc-

tional design showed a high effect size for improving students' CPS skills. Therefore, we dismissed Hypothesis 2.0 (There is no evidence that the PF-based programming course can impact on students' CPS skills in a Korean elementary school.) and adopted Hypothesis 2.1 (The PF-based programming course can impact on students' CPS skills in a Korean elementary school.).

In conclusion, PF-based programming instruction was effective in helping elementary students to develop CT and CPS skills through programming education. Students repeatedly experienced the process of generating, failing, improving, and retrying different solutions in an atmosphere that allowed them to fail at solving problems, and finally to solve problems successfully, rather than following the teacher's model solution.

5.2. *Discussions*

We analyzed why the productive failure was effective in improving computational thinking. First, ill-structured and authentic problems were more effective in developing abstract thinking, which requires students to understand and analyze the problem and extract the key elements needed to solve it. On the other hand, structured problems were less effective in developing abstraction thinking because they were presented in a way that made it easier for students to understand the problem and extract the key elements. Second, productive failure encourages students to select problem-solving methods and appropriate algorithms and represent them in different ways. It can be interpreted that these helped students improve their computational thinking. However, in the lecture-based course, students were expected to copy exemplary solutions, which limited the development of computational thinking. Third, failure is allowed in the process of implementing problem-solving methods in a programming language, which effectively enhances automation thinking.

We analyzed why productive failure was highly effective in improving creative problem-solving. First, scenario-type authentic problems were influential in motivating students to learn. Second, creating an atmosphere where students were allowed to fail in an activity that involved generating multiple solutions and expressing them in a programming language was effective in promoting divergent thinking and understanding and mastery of knowledge and skills in a specific field. Productive failure allows students to fail and experience multiple attempts rather than pushing them to succeed in solving a problem. As students' divergent thinking improved, they became actively involved in the process of modifying and improving their own or their group's solutions, ultimately leading to a more successful outcome. Fourth, in the consolidation steps, critical and logical thinking was promoted through activities that involved comparing their solutions to those of others. We observed that students also practiced comparing multiple solutions to find the similarities and the differences through discussion and determined which solution was more efficient in synthesizing all findings through discussion. These activities helped them discover what makes a problem solution effective, which was effective in improving their creative problem-solving skills (Bae, 2006).

We found that computational thinking skills improved significantly with a direct instructional programming course. This is consistent with previous research showing that block based programming languages are effective in improving computational thinking in elementary school students (Zhang and Nouri, 2019). However, the effect size showed that productive failure was more effective than creative problem-solving and did not show a significant increase in the direct instructional programming course. The reason for this is that the textbook presented structured problems that were not authentic, which did not trigger students' motivation. Also, the teacher taught the target concepts in a direct teaching method at the beginning of the class. The following activities also didn't provide the opportunities to explore various solutions. Therefore, it failed to improve students' divergent thinking.

In conclusion, we found that the productive failure-based approach to programming education in elementary school is effective in fostering computational thinking and creative problem solving. Students repeatedly experienced the process of generating, failing, improving, and retrying different solutions in an atmosphere that allowed them to fail at solving problems rather than just following the teacher's model answer to succeed in solving problems. In this learning process, students were able to use computational thinking skills naturally and exercise their creative problem-solving skills in finding efficient ways to solve problems.

5.3. *Recommendations*

Based on the results, we propose the following recommendations for future work. First, a follow-up study should be conducted to expand the topics and contents and generalize the effectiveness of the PF-based programming instruction proposed in this study. Second, specific teaching strategies need to be developed to encourage students in the process. PF relies on teacher strategies, such as creating a learning atmosphere that allows students to fail and provides emotional support. If students are frustrated by failure and give up on problem solving, learning will not progress. Therefore, detailed teacher prompts and feedback must be provided to keep students on track. In addition, when conducting group activities in programming instruction, student grouping can act as a variable that affects learning (Son and Sohn, 2014). Therefore, teachers should use specific strategies for group formation and group interactions such as discussion. Finally, there is a need to develop a variety of topics and problems and materials to implement PF-based programming instruction. The essence of a PF strategy is to present ill-structured and authentic problems, thereby motivating, engaging, and encouraging higher-order thinking. So, it should be developed for different levels of students, subjects, and areas of interest. To this end, PF-based programming instruction should be suggested as an effective way to teach programming in materials such as textbooks.

References

Bae, Y. (2006). *Robot Programming Education Model in Ubiquitous Environment for Enhancement of Creative Problem-solving Ability.* Korea National University of Education. Cheongju. http://www.riss.kr/link?id=T10372477

Bransford, J., Schwartz, D. (1999). Rethinking transfer: A simple proposal with multiple implications (Vol. 24). *Washington DC: American Educational Research Association*.

Cangelosi, J. (1990). *Designing Tests for Evaluating Student Achievement*. Longman Publishing Group.

Cho, S.H. (2002). *Brief Creative Problem Solving Test Development Study (II) (CR2002-43)*.

DeCaro, M.S., Rittle-Johnson, B. (2012). Exploring mathematics problems prepares children to learn from instruction. *Journal of Experimental Child Psychology*, 113(4), 552–568.

diSessa, A.A. (2006). A History of Conceptual Change Research: Threads and Fault Lines. In *The Cambridge Handbook of: The Learning Sciences.* (pp. 265–281). Cambridge University Press.

Han Sung, K. (2021). Exploring Implications for Revision of Informatics Curriculum Based on Computing Curricula 2020: Focusing on Articulation Analysis [Exploring Implications for Revision of Informatics Curriculum Based on Computing Curricula 2020: Focusing on Articulation Analysis]. *The Journal of Korean Association of Computer Education*, 24(2), 105–117.

ICT, K.M. o. S. a. (2020). *Innovate Korea 2045 : Challenges and Changes for the Future*. Sejong: Korea Ministry of Science and ICT Retrieved from https://www.kistep.re.kr/boardDownload.es?bid=0003&list_no=39882&seq=12345

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379–424.

Kapur, M. (2010). Productive failure in mathematical problem solving. *Instructional Science*, 38, 523–550.

Kapur, M. (2014). Productive failure in learning math. *Cognitive Science*, 38(5), 1008–1022.

Kerrigan, J., Weber, K., Chinn, C.A. (2021). Effective collaboration in the productive failure process. *The Journal of Mathematical Behavior*, 63, 100892.

Kim, Y. (2019). *Development of Achievement Criteria and an Assessment Tool to Measure Computational Thinking* (Publication Number the degree of Master of Education) Seoul National University of Education]. Seoul. http://www.riss.kr/link?id=T15342908

Lee, D., Yi, S., Lee, Y. (2022). A Study of Domestic Programming Education in Elementary School Based on Systematic Literature Review [A Study of Domestic Programming Education in Elementary School Based on Systematic Literature Review]. *The Journal of Korean Association of Computer Education*, 25(6), 35–50. DOI: 10.32431/kace.2022.25.6.003

Lee, D.K. (2016). Alternatives to P value: confidence interval and effect size. *Korean Journal of Anesthesiology*, 69(6), 555–562.

Marton, F. (2006). Sameness and difference in transfer. *The Journal of the Learning Sciences*, 15(4), 499–535.

Ministry of Education, K. (2015a). *The 2015 Revised Curriculum*. Sejong: Korea Ministry of Education.

Ministry of Education, K. (2015b). *The 2015 Revised Curriculum of Practical Art*. Sejong: Korea Ministry of Education.

Oldridge, M. (2017). Is it about coding? No. It's about computational thinking. In.

Papavlasopoulou, S.G., Michail N. Jaccheri, Letizia. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, 99, 415–427. https://doi.org/10.1016/j.chb.2019.01.008

Pewkam, W., Chamrat, S. (2022). Pre-Service Teacher Training Program of STEM-based activities in Computing Science to Develop Computational Thinking. *Informatics in Education*, 21(2), 311–329.

Piaget, J. (1964). Cognitive Development in Children: Development and Learning. *Journal of Research in Science Teaching*, 2, 176–186.

Ritter, F., Standl, B. (2023). Promoting Student Competencies in Informatics Education by Combining Semantic Waves and Algorithmic Thinking. *Informatics in Education*, 22(1), 141–160. https://doi.org/10.15388/infedu.2023.07

Schlegel, M. (1995). *A Handbook of Instructional and Training Program Design*.

Schwartz, D.L., Martin, T. (2004). Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. *Cognition and Instruction*, 22(2), 129–184.

Shih, W.-C. (2019). Integrating computational thinking into the process of learning artificial intelligence. *Proceedings of the 3rd International Conference on Education and Multimedia Technology*, New York.

Silapachote, P., Srisuphab, A. (2017). Engineering courses on computational thinking through solving problems in artificial intelligence.

Snedecor, G.W., Cochran, W.G. (1980). *Statistical method 7th Ed*. The Iowa state university press, Ames, Iowa, USA, 1980, 39–63.

Son, K.H., Sohn, W.S. (2014). The Development and Application to Computer Programming Education using Arduino. *The Journal of Education*, 34(3), 159–179.
https://scholar.kyobobook.co.kr/article/detail/4050025409434

Soojin, J. (2017). Design and effect of development-oriented model for developing computing thinking in SW education. *Journal of The Korean Association of Information Education*, 21(6), 619–627.

Taguma, M., Barrera, M. (2019). *OECD Future of Education and Skills 2030: Curriculum Analysis*.
https://www.oecd.org/education/2030-project/

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., Baggett, W.B. (2003). Why do only some events cause learning during human tutoring? *Cognition and Instruction*, 21(3), 209–249.

Yoo, S. (2021). *SPSS Statistical Analysis for Writing a Thesis* [SPSS statistical analysis for writing a thesis]. Hwangsogirlum Academy.

Youngsik, J. (2018). The Problems and Improvement of the SW Education Policy in Elementary School. *Proceedings of Journal of The Korean Association of Information Education*, 9(1), 91–97.

Zeng, D. (2013). From Computational Thinking to AI Thinking. *IEEE Intelligent Systems*, 28(06), 2–4.

Zhang, L., Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, 141, 103607.
https://doi.org/https://doi.org/10.1016/j.compedu.2019.103607

D. Lee is a Ph.D. student in the Department of Computer Education at the Korea National University of Education, as well as a teacher at the public elementary school in Korea. The field of her scientific activity to date covers the areas of computer science education, Artificial intelligence convergence education and Learning science.

Y. Lee is a Ph.D. in the Department of Computer Science at the University of Minnesota in the U.S. and is currently a professor of computing education at the Korea National University of Education. The field of his scientific activity to date covers intelligent systems, learning science, information education, and artificial intelligence education. He has served as the Korean representative of the TC3: Education Subcommittee, which conducts educational research from computer science to AI at the International Federation for Information Processing (IFIP).