International Journal of Learning Technologies and Learning Environments
International Institute of Applied Informatics
2025, Vol. 8, No. 1, IJLTLE806

Implementing Toggle Function for Card Operation-Based Programming Learning Support System and Evaluating its Effectiveness

Shimpei Matsumoto [*], Hiroshi Shigematsu [*], Taiki Okuhira [*]

Abstract

In many cases, programming classes in higher educational institutions have desired a system that supports students to concentrate on the essential learning task intended by the instructor. Then, focusing on thinking about the relations of meaningful parts, a card operation-based programming learning support system, COPS, was developed. This system targets to think only the relations of meaningful parts called chunks consisting of one or more statements and aims to control cognitive resources by limiting the patterns of learning activity. Through the practical use of an actual programming class, COPS was able to make learners concentrate on the intended learning task with reducing cognitive load. However, since COPS simplifies learning activities, there is a possibility that COPS does not have more learning effectiveness than the typical coding exercise. Therefore, to make COPS close to the learning activity of actual programming, we focus on the toggle function. The toggle function is a select box placed on the card, and it requires the user to learn to choose one statement from two or more statements. Thus, the toggle function would make the learning task of COPS closer to actual programming learning. This paper shows the detail of the design and implementation of the toggle function to COPS.

Keywords: programming, card operation, relation between meaningful parts, cognitive load, toggle.

1 Introduction

Programming requires various abilities and activities, such as mathematical skills, linguistic skills, logical thinking skills, knowledge of computer architecture, and the ability to design algorithms. Based on such characteristics, some previous research said that the burden of programming learning is enormous for novices unfamiliar with computers. In addition, allocating cognitive resources efficiently to address essential learning tasks is not easy for its novices because programming requires learners to have a variety of skills at once [1]. Originally, programming itself has a high intrinsic load. So, reducing the learner's external cognitive load when educating programming is said to be important [2].

[*] Hiroshima Institute of Technology, Hiroshima, Japan

Based on such a background in programming education, a card operation-based programming learning support system, COPS, was developed. The primary purpose of COPS is to reduce the influence of nonessential cognitive load [4] as much as possible for programming education, focusing on the thinking about the relations of meaningful parts called chunks consisting of one or more statements. The card operation implemented by COPS is a form of programming learning that deals with learning tasks that limit the patterns of learning activities to make students' cognitive resources concentrate on an essential learning activity intended by its tutor. COPS aims to acquire habits to be aware of the program structure and grasp the overall structure with the learning task focusing only on the thinking about some meaningful parts in the whole source code. This method of designing learning tasks by focusing on internal structures is called the open information structure oriented approach [5]. Thanks to the aforementioned features, COPS can limit the influence of nonessential cognitive load and make the programming an accessible learning experience for its novices.

Especially the previous research suggested that COPS was a more effective learning method for novices [3]. The previous research also revealed that COPS was an efficient learning method to shorten the learning time more than the typical coding-based learning while having the same learning effect as the typical coding-based learning [6]. On the other hand, since the learning activity of COPS limits only the card operation, the learning effectiveness seems only for the thinking of card permutation. Then, the learning tasks of COPS should be more complex, which will allow students to think deeply, similar to the actual algorithm design. This complication means giving elements other than card permutations to the learning task, and we believe that one of such elements is thinking data dependencies. Increasing the number of cards is one approach to increasing the complexity of the learning task. Increasing the number of dummy cards, which are not included in the correct answers, is another easy approach to increase the complexity of the learning task. However, an increase in cards directly leads to a decrease in visibility. The decrease in visibility will cause an increase in the workload to find the required card, which leads to an increase in nonessential load and a decrease in essential cognitive load. Finding the number of cards that maximizes the learning effect is best but not easy. Therefore, we should utilize a factor other than the number of cards to improve the complexity of the learning task. Among various methods to improve the complexity of learning tasks while keeping visibility, we focus on the toggle function. Using the toggle function of the programming learning mechanism [7] proposed by Parson may be possible to realize the complexity of learning tasks independent of the number of cards, which will lead to giving learning activities closer to the actual algorithm design.

This study aims to verify whether a mechanism of the toggle function is effective for COPS to suppress the nonessential cognitive load or not, which occurred along with the increase of cards. To accomplish this objective, there are three issues that we need to address: 1. Designing the toggle functionality to be available in COPS, 2. Implementing the toggle function on the actual COPS, 3.Clarifying the usefulness of the toggle function through comparative experiments.

2 Basic Concepts

2.1 Closed-ended Learning Task

In general programming courses, students are often presented with learning tasks that require a variety of skills at the same time. For example, there are many learning tasks that require many kinds of skills such as creativity, computational skills, algorithm construction skills, data structure design skills, at the same time. When there are many skills required, grasping which skills are insufficient is difficult for learners. As a result, learner's self-evaluation of their own skills becomes difficult, and besides, grasping learners' level of understanding is also difficult for instructors. This tendency is considered to be one of the factors that inhibit learners' self-learning. The more "open-ended" the learning task is, the more difficult it becomes to evaluate skills. Therefore, many researchers and educators have suggested that a system in which the goals of the material are subdivided so that the learners themselves can grasp the degree of their errors and learn through tasks in which the necessary skills are limited is effective in supporting programming learning. In addition, they say that restricting the learner's choice of activities to "simple operations" is also effective in analyzing comprehension. Restricting activity patterns facilitates the analysis of the learning process. Thus, a programming learning task in which the required skills are limited and the operations that learners can perform are restricted is considered to be a closed-end learning task [8].

COPS is one of the closed-ended tasks and is used to check the learner's level of understanding. Closed-ended means that there is always only one correct answer to a given problem. In a closed-ended problem, the learner does not create all the source code from scratch but rather creates a finished product using pre-prepared materials. Closed-ended problems include source code reading [9], description completion for a part of the entire source code [10], fill-in-the-blank programming problem [11], modification of templates [12], and coding puzzle [13][14].

2.2 Parson's Puzzle

Parson's Puzzle [7] is a card operation-based programming learning support system, and the function is close to COPS. Parson's Puzzle provides a learning task to assemble each part with one instruction like a puzzle. The previous research conducted a practical use of Parson's Puzzles for students at Otago Polytechnic who joined a programming class [7]. As a result, 80% of the students answered "Parson's Puzzle is easy to use" and gave a positive opinion on Parson's Puzzle. Parson's Puzzle aims to help you understand the concept of programming while having fun. Therefore, Parson's Puzzle is not intended to improve your programming skills. In Parson's Puzzle, each card corresponds to one statement. The learner selects the cards needed to fulfill the question's requirements by drag-and-drop. The learner can answer the question by placing the cards in the appropriate order in the answer box. With Parson's Puzzle, programming tutors can convey the design pattern to be learned because the structure of the answer is fixed. One of the primary purposes of the card operation-based programming learning method is to understand the design pattern intended by the tutor. Therefore, Parson's Puzzle and COPS goals are assumed as the same. However, there are several differences between Parson's Puzzle and COPS. One is the purpose. The goal of Parson's Puzzle and COPS is to convey the basic concept of programming. On the other hand, the goal of COPS is to acquire programming knowledge and, consequently,

improve programming skills. The structure of learning tasks is also slightly different from there. Concretely, The number of statements on each card is different. One card has only one statement in Parson's Puzzle, but one card may have one or more statements in COPS.

Parson's Puzzle might be useful as a mechanism to fill the gap between code description and visual programming language. Furthermore, since there is no need to type in source code directly, reducing the cognitive load on memorizing grammar and typing might be possible. However, there are no previous studies that have clarified these effects. There is also no previous research mentioning the cognitive load. Reports on the usefulness of Parson's Puzzle have been limited to the fact that the puzzle features can make programming entertaining. Although such a situation, we consider that Parson's Puzzle has a mechanism to support the allocation of cognitive resources to the learning task that the tutor aims by reducing the nonessential cognitive load because the function of Parson's Puzzle is close to COPS. In particular, Parson's Puzzle has some interesting features not found in COPS. For example, Parson's Puzzle has a mechanism that can change the data structure of the presented program called the toggle function. Therefore, we focus on the toggle function of Parson's Puzzle. The toggle function is considered to be useful for learning that thinks more deeply about the structure. Therefore, introducing the toggle function of Parson's Puzzles into COPS and clarifying the effectiveness is considered an important issue in improving the learning effectiveness of COPS.

2.3 Cognitive Load Theory

Cognitive load theory is to think about the cognitive load generated by learning to obtain new knowledge and skills more efficiently [4]. There are 3 kinds of cognitive load: intrinsic load, extraneous load, and germane load. Intrinsic load relates to learning task itself, extraneous load relates to the inappropriateness of teaching materials and teaching methods, and germane load relates to the solution process itself which contributes beneficially to learning [4]. Since programming has a high intrinsic load inherently and it is said that an extraneous load can be intentionally reduced by the instructor. The programming instructor needs to reduce an extraneous load of learners as much as possible. Therefore, various previous studies on programming referring cognitive load have been conducted.

3 Card Operation-Based Programming Learning Support System

3.1 Card Operation-Based Programming Learning Method

Card operation-based programming learning method is a type of programming exercise that provides learners with two or more cards written some part of the source code and gives learning tasks to assemble such cards in an appropriate order that runs a program properly. In particular, the goal of learning is to fulfill the question's requirements by rearranging the cards and giving them the proper order. The card operation-based programming learning method aims to concentrate cognitive resources on targeted learning activities while indirectly reducing some learning activities as extraneous load after dividing programming into several learning activities. When assuming a source code consisting of 1 line 1 statement, programming is regarded as a learning activity to think a line or two or more lines of statements. This learning requires mainly thinking about the meaning and processing of each program line. Previous research roughly divided learning activities of programming into

several types, and one of them is " learning activities to think about the relationship between statements", which is mainly to think about the relationship between some parts in a source code [3]. The card operation-based programming learning method is one effective way to facilitate "learning activities to think about the relationship between statements".

In typical programming learning centers on coding exercises, frequent grammatical errors due to typos are considered a problem. Finding typos and fixing bugs themselves are suitable activities and necessary skills. Therefore, such activities are necessary, but if the tutor intends for the learning to be otherwise, these will become only nonessential tasks. That is, the important learning activities relatively change because the knowledge and skills to be learned vary depending on the lesson's objectives. In view of these considerations, the card operation-based programming learning method focuses on the fact that the learning activity of considering the relationship between lines is not enough in typical programming lessons and intends to solve it.

There are four types of learning activities to consider the relationship between lines: thinking algorithm, which is a learning activity to design and understand efficient processing procedures; thinking design pattern, which is a learning activity to design and understand proper processing order; tracing, which is a learning activity to track changes of data correctly; logical debugging, which is a learning activity to understand and correct mistakes in the processing of multiple instructions. The card operation-based programming learning method deals with these four learning activities as essential cognitive loads and learning activities such as grammar and typing as indirectly important tasks and nonessential loads. The card operation-based programming learning method is not supposed to completely replace the typical coding-centered programming exercise but is a supplemental tool for improving general lessons. The card operation-based programming learning method is intended to be used in some typing-centered classes. However, to make card operation-based programming learning part of a lesson, some software to facilitate the learning activity of card operation-based programming learning is essential because of the time constraint of the lesson. This is why COPS has been developed [3].

3.2 Details of Implementation

Fig. 1 shows the appearance of COPS. COPS has some functions as a Learning Management System. All functions of COPS are available through the Internet as long as user-side device equips a web browser available the standard of HTML5. COPS runs on Ubuntu 15.04 as an operating system, and uses the following software applications; Apache 2.4.10, a web server program to provide web service; Ruby 2.1.2, a server-side scripting language to implement the function; Ruby on rails 4.2.4, a server-side web application framework; jQuery 1.11.3, a cross-platform JavaScript library to perform dynamic UI; UIkit 2.23.0, front-end framework for developing web interfaces; MySQL 5.6.28, a database management system to store all system data.

The learner of COPS can move cards by dragging and dropping mouse operation from the selection space at the right to the answer space at the left and assemble a program as an answer by moving cards needed to fulfill the requirements of each question (see Fig. 2). Like the actual source code, the learner can place an indent before the card as needed. In the actual source code, indentation does not affect the execution result. Therefore, in COPS, indentation does not affect the evaluation of correctness as in the actual source code. The correctness evaluation depends on the sequence of cards and the execution result. When the learner completes their arrangement of cards to the answer space, the learner can

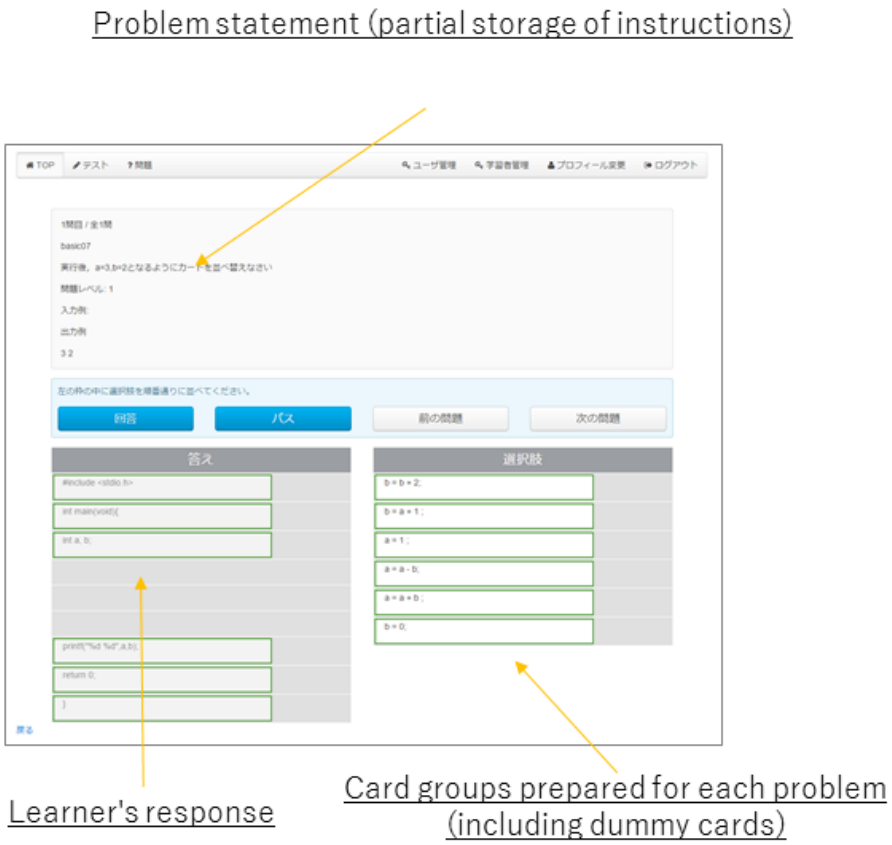

Figure 1: Appearance of card operation-based programming learning support system

automatically receive a true/false judgment by pressing the answer button. When the answer button is pushed, assembled cards are converted to the source code by the backend system. The backend system also makes a JSON format file including the source code, transfers the JSON format file to GCC, GNU Compiler Collection, through the web interface system for C language compilation originally developed by our research project. Finally, the frontend system of COPS shows the execution result on its webpage after COPS receives the JSON file, including the execution result. When the learner pushes the answer button, the learner gets a judgment result on whether the answer is correct or not. Along with the judgment result, COPS provides feedback messages corresponding to the learner's answer, including some hints or the intention of the question. Feedback includes execution results and pre-set tips by the tutor. The feedback message needs to be registered in advance when the tutor creates each question, but the blank setting (No feedback is given to the learner) is also possible. The feedback setting would depend on the tutor's policy.

In addition to execution results and the pre-set tips, COPS has added the ability to automatically evaluate card adequacy and provide feedback based on card placement. The appropriateness of the card placement is indicated by the color that seems most intuitive to the learner. After responding, COPS provides the appropriateness of the learner's response as feedback by representing cards with the color corresponding to its appropriateness. The

Figure 2: A screenshot when a learner assembles a program

evaluation of the card's suitability is based on the sequence of correct answers given when the question is created. When creating questions, the tutor sets the sequence of cards as the correct answers in advance, and the sequence determines the order of the cards. Depending on the content of the question, unordered settings are also possible for some parts. For example, if there are two assignment statements, a=1 and b=2, the order is not unique. If the order of the cards answered by the learner is the same as the correct answers, the color of all cards in the answer space turns blue. If the order of the cards answered by the learner is different from the correct answers, the color of the cards turns yellow. In other words, the condition for turning yellow is that the card is part of the correct answer but in the wrong place. If the card the learner answered is not part of the correct answer, the card's color turns red. In short, the dummy card chosen as the answer turns red.

4 Proposal

This research newly proposes an implementation way of toggle function to CPOS. The toggle function is one of the main mechanisms that Parson's Puzzle has and for which Parson's Puzzle is highly regarded. Therefore, incorporating the toggle function into COPS is expected to have a higher learning effect.

The toggle function is a mechanism that provides an area where the learner can exclusively select a statement in a card (see Fig. 3). The toggle function gives a question with a large number of selections without increasing the total number of cards. We will discuss the need for the toggle function in more detail from now on. Dummy cards are a mechanism to give selections that do not include the correct answer, which are the cards including wrong statements. Answers containing dummy cards cannot satisfy the question requirements because dummy cards have different control structures and data than the correct answers. In addition, dummy cards may have syntax errors. In any case, dummy cards are a distraction to the learner. However, the learner learns better than usual because of the dummy cards. Let the method using dummy cards be the usual method. The reason is that the difficulty of a problem depends on the number of cards in the choices. In other words, we can simply

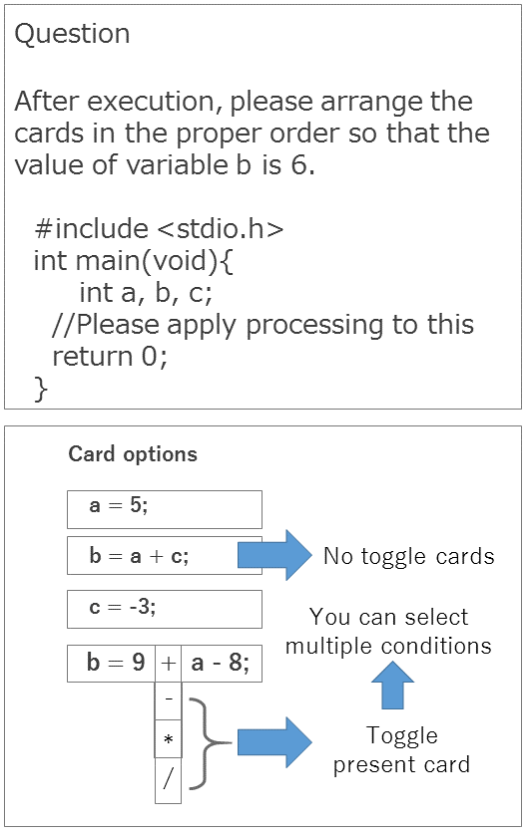

Figure 3: Concept of the toggle function

estimate that the more dummy cards there are, the more complex the question becomes and the more learning is realized. Of course, we can increase the number of cards included in the correct answer instead of dummy cards to improve the difficulty of the question. However, since we focus only on dummy cards, we will not include such a discussion. Now, what happens when the number of cards increases? It is like a nervous breakdown: the work to find the cards increases. In addition, the size of the computer display screen is limited. Thus, increasing the number of cards would lead to a significant increase in cognitive load. The work of searching for cards can be assumed as the extraneous cognitive load. Therefore, we need the toggle function to solve such a trade-off. The toggle would allow the learner to think deeply about the structure while maintaining visibility.

This paper implements the toggle function by giving the select box to a part of the statement in a card (see Fig. 4). Using the toggle function includes an arithmetic operator, a comparison operator in a branch statement, and a variable. The toggle function would make the learning with COPS not only closer to actual structure thinking but also more fun, like Parson's Puzzle.

The card operation method can be considered as a structured task [15] for assembling parts. In the structured task, in order to realize sufficient learning, more than a certain number of parts are required. On the other hand, when parts are increased, the ability to grasp these and to find suitable parts is assumed to be necessary. This is a considerable problem for learners with low verbal ability and cognitive processing ability. The toggle

Figure 4: implementation of the toggle function

function can compress and present information to solve the problem of increasing parts. So, we can assume that the toggle function is a mechanism that can suppress unnecessary cognitive load, generally thought of as a critical problem in structured tasks. COPS is an actual form of structured tasks. Showing the usefulness of the toggle function in COPS will lead to showing the usefulness of the toggle function for structured tasks, so the report of this study will be valuable.

5 Experiment

5.1 Experimental Condition

The experiment aims to clarify whether the toggle function could make the learner deeply think while suppressing the generation of nonessential cognitive load. The subjects were 1-4 grade 20 university students having enough programming language skills. All subjects have also learned C programming language. In this experiment, the learning goal was to acquire knowledge about binary search and bubble sorting.

Firstly, to measure the subject's comprehension level of the learning goal, subjects took a pre-test including two questions for up to 45 minutes. One question is about the binary search, and the other is bubble sort. Then, the subjects were divided into two groups; the one group learning with the toggle function was the experimental group, and the other group without the toggle function was the control group. Subjects in the control group were provided with dummy cards for all toggle choices. For example, suppose a learning task had a toggle with four choices. In the experimental group, a single card with one toggle will be provided. On the other hand, the control group will be provided four dummy cards because the toggle is unavailable. Each group included ten subjects. Based on the pre-test results, we determined the group of each subject. We chose a combination such that the difference between the means and variances of the pre-test would be minimal in the two groups. Naturally, there were no statistically significant differences in the means or

variances of the pre-tests.

We conducted an experiment to measure the learning effect. There was about a week interval between the experiment and the pre-test. The experiment had six questions: three questions were about bubble sort, and the others were about binary search. The experimental time was 60 minutes at maximum. Subjects worked on the task in their group's way within the experimental time. The subjects in the experimental group could use the toggles equal to 10 or more dummy cards. The toggles provided all dummy selections in the learning tasks, so there were no dummy cards in the experimental group. On the other hand, the subjects in the control group had to address given tasks without the toggles. It means that the experimental tasks had a large number of dummy cards. This experiment dealt with learning tasks consisting of cards, each of these had only one statement. In this experiment, the toggles gathered the differences between the arithmetic operations and the comparison operators.

Finally, we gave the post-test of 45 minutes at maximum to confirm the learning effectiveness. There was about a week between the experiment and the post-test. After the post-test, we analyzed the effectiveness of the toggle function based on the post-test results, the response to the questionnaire, and the response to cognitive load.

5.2 Experimental Results

The evaluation criteria for the pre-and post-tests are as follows. First, let me show the evaluation method for source code description questions. The subject's source codes were sorted into two groups according to whether input/output was possible or not. If the input/output could not clear the test case at all, the score was 1; if it could partially clear the test case, the score was 2. If an input/output could clear the test case, the score went to 3-5. if the source code included a waste description or not good writing, the score was 3; if there was a waste, but it was accepted, the score was 4; if the answer was perfect, the score was 5. In addition to these, pre-and post-tests asked to supplement statements to some parts in each source code, which is adopted in general programming ability judgment.

In addition to the test evaluation, we measured cognitive load and assessed the questionnaires. The evaluation method of the cognitive load was based on previous studies [3]. All items were on a 6-point interval scale. The maximum value (positive) was set at six and the minimum value (negative) at 1. In order to make the respondents aware of the interval scale, the subjects were asked to check one of the numbers placed at equal intervals.

Fig. 5 shows the results of the post-test. Error bars mean standard error. Fig5 is the average scores of post-test and Fig. 6 is the average time of post-test. In Fig. 5, there was significantly different by Welch's t-test ($p < .05$). There was no statistically significant difference in the test time, as shown in Fig. result2, but the results of the post-test scores were fewer in the experimental group. These results suggest that the toggle function may improve the learning effect.

The aim of this paper is to clarify whether toggle function can make learners think deeply while suppressing the generation of non-essential cognitive load. The non-essential cognitive load referred to here is, in particular, the task burden of reordering cards. In the card operation-based programming learning system, utilizing dummy cards, a mechanism for preparing statements not included in the correct answer is a way to realize deeper thinking about structure. However, preparing a large number of dummy cards means that the amount of cards that the learner can rearrange will be huge. The increase of dummy cards will cause a non-essential cognitive load relating to the workload of sorting the cards be-

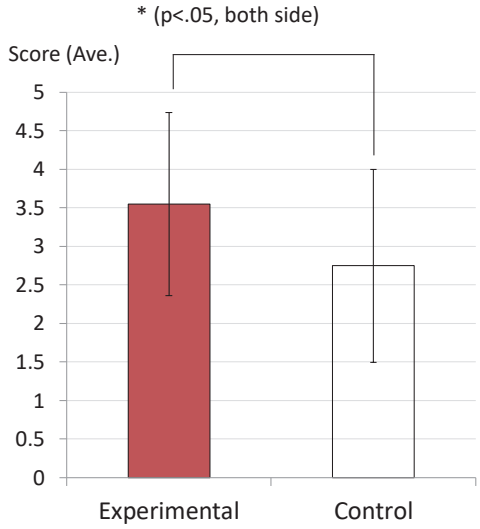

Figure 5: The average scores of the post-test

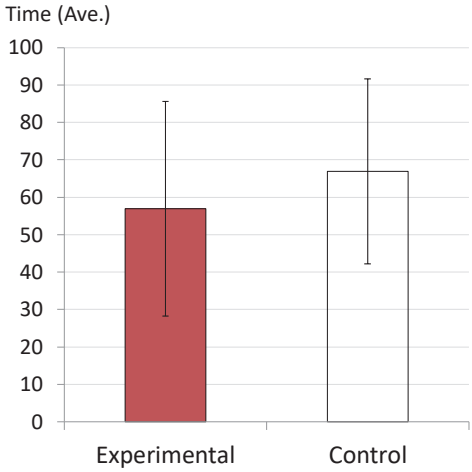

Figure 6: The average time of the post-test

cause the recognition performance is lowered. In this regard, we expect toggle functionality to be effective. Therefore, from the prepared questionnaire in this paper, as shown in Table 1, we clarify whether toggling of the dummy cards lead to the reduction of the cognitive load relating to the workload of utilizing cards. The contents of the questionnaire to clarify extraneous cognitive load of each learning element are shown in Table 1. The questions included 8 questions to evaluate the effectiveness of the toggle. Specifically, we prepared the question "how much extra load did COPS generate" for each learning element such as grammar, data structure, and algorithm. The details of each learning element required by this experiment are listed below.

1. "Grammar" refers to the activity of storing programming rules and syntax, etc., and creating a program according to it.
2. "Typing" refers to the activity that actually drives the program. Here, in conjunction with the card operation type learning support system, we asked how much load they felt about typing when they had a programming lesson including both the card operation-based learning and the coding-based lesson.

Table 1: Questions for measuring extraneous cognitive load

No	Content
ECL1	When you think of "C grammar and syntax", how much extra load did COPS generate for you to solve given questions?
ECL2	When you think of "data structure", how much extra load did COPS generate for you to solve given questions?
ECL3	When you think of "trace of variables", how much extra load did COPS generate for you to solve given questions?
ECL4	When you think of "algorithm", how much extra load did COPS generate for you to solve given questions?
ECL5	When you think of "design patterns", how much extra load did COPS generate for you to solve given questions?
ECL6	When you think of "debugging syntax and compiler errors", how much extra load did COPS generate for you to solve given questions?
ECL7	When you think of "debugging of logical errors (when syntax errors do not occur but processing results differ)", how much extra load did COPS generate for you to solve given questions?
ECL8	When you grasp the whole of each question, how much extra work load did COPS generate for you?

3. "Data structure" refers to activities that understand and use data structures such as variables, arrays, and tree structures.
4. "Tracing" refers to learning activities that track changes in data correctly
5. "Algorithms" refer to learning activities that design and understand efficient procedures
6. "Design pattern" refers to learning activities that design and understand proper processing order
7. "Grammar debugging" refers to debugging (detection and correction of errors) activities that are performed when a grammar error (error due to a programming language description error) occurs. In the questionnaire, subjects were asked how much load they felt about grammatical errors when they had a programming lesson including both the card operation-based learning and the coding-based lesson.
8. "Logical debugging" refers to learning activities that understand and correct errors in the processing of multiple instructions.

In addition to the items assessing extraneous cognitive load, we prepared items assessing intrinsic cognitive load and germane cognitive load. The number of questions was limited to three because they were not as important as the extraneous cognitive load, and we should consider the subjects' burden. Intrinsic cognitive load is referred to as ICL and germane cognitive load as GCL in the following. In this experiment, the professor intended to make learning enjoyable, so the question asking about enjoyment was also referred to as GCL. The three items are shown in Table 2. In Table 2, intrinsic cognitive load is referred to as ICL and germane cognitive load as GCL relatively. In this experiment, the teacher intended to make learning enjoyable, so the item asking about enjoyment was also referred to as gername cognitive load.

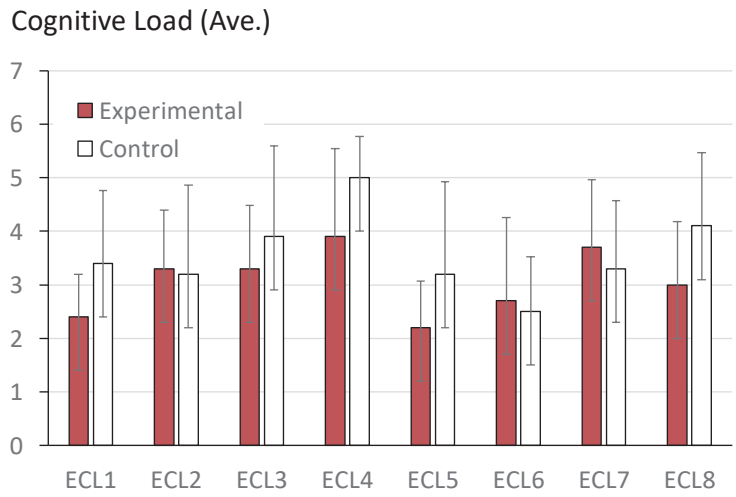

Figure 7: Extraneous cognitive load for learning each element

Table 2: Questions for measuring intrinsic and germane cognitive load

No	Content
ICL	How much of a burden was the learning task presented in this experiment for you?
GCL1	Did you enjoy the learning task presented in this experiment?
GCL2	Were you able to focus on the essential learning intended by the teacher?

The mean values of the experimental and control groups' extraneous cognitive load for each learning component are shown in Fig. 7. Welch's t-test showed no statistically significant difference between the two groups for all items. On the other hand, $p = 0.08$ for ECL1, $p = 0.09$ for ECL4, and $p = 0.09$ for ECL8 were obtained by t-test. Also, the experimental group tended to have lower overall loadings than the control group. This suggests that toggling is effective in reducing the extra load during learning. For ECL6 and ECL7, the values for the experimental group were higher than those for the control group, although not significantly so. These two questions are related to debugging. The toggle has the effect of hiding all instructions except the selected one. Conversely, if the toggle is not used, all instructions can be viewed at any time. If the commands are hidden, debugging becomes more difficult. Therefore, the load on the experimental group was higher than that of the control group. In ECL6 and ECL7, there is no significant difference between the two groups. In light of the above, toggling is a useful mechanism that has the effect of reducing the extra load on the learner.

Fig. 8 shows the means of intrinsic cognitive load and germane cognitive load for the experimental and control groups, respectively. Welch's t-test showed no statistically significant differences between the two groups for all questions. These results indicate that toggling does not increase germane cognitive load. However, this result does not negate the usefulness of the toggles. This result supports the idea that toggling can provide the same learning as in the previous section. Fig. 7 suggests that toggling may reduce the intrinsic

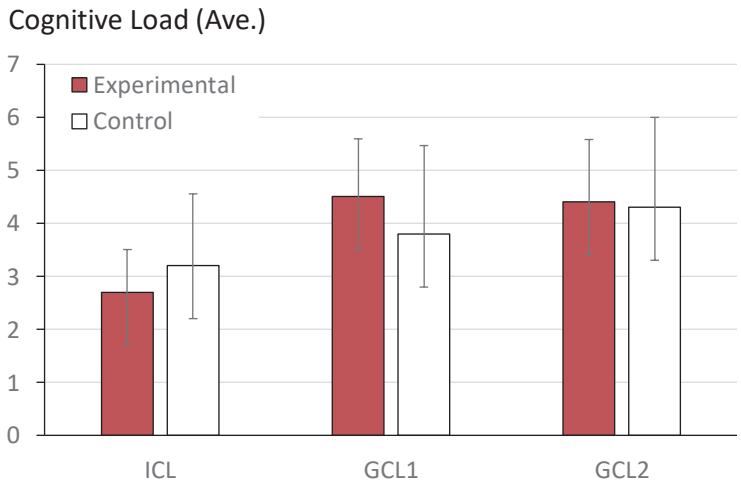

Figure 8: Intrinsic cognitive load and germane cognitive load

cognitive load. The present results confirm that the intrinsic cognitive load and germane cognitive load are the same as the usual learning. Furthermore, the learning effect of the toggles was revealed in Fig. 5. Therefore, toggling is likely to be useful. It is not surprising that there is no difference in the intrinsic cognitive load between the two groups.

6 Additional Experiment

6.1 Experimental Condition

The results of the previous section confirm the usefulness of the toggles. However, it cannot be said from these results alone that toggles themselves have the function of increasing the learning effect. In order to use toggles more effectively, investigating the range of application of toggles is necessary. Therefore, we conducted an additional experiment to clarify the range of application of toggles.

The previous section suggested that toggles are a better mechanism than dummy cards and that they can reduce extra burden by improving visibility. It was shown that toggles may improve the learning effect as a result of the reduction of extra load. On the other hand, the usefulness of the toggles themselves has not been discussed. In other words, the experiments in the previous section do not fully clarify whether the toggles themselves have a function to enhance the learning effect. We can clarify whether the toggles themselves have a learning effect by comparing two groups: one with the toggles added and the other without the toggles added. Therefore, we conducted experiments to clarify the range of usefulness of the toggles. The control group was given a learning task with several dummy cards. The experimental group was given a learning task with one or more four-choice toggles for some cards of the same learning task as the control group. The learning effects of the toggles themselves were investigated by comparing the learning effects of the two groups. The research question is “Do the toggles themselves have a function to enhance learning?”

The subjects were 20 1-4 grade university students who had learned C programming language even once. We first conducted pre-test up to 30 minutes and divided the subjects

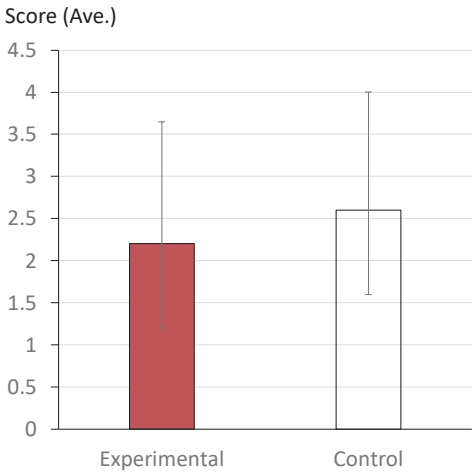

Figure 9: The average scores of the post-test in the additional experiment

into two groups so that their academic levels would be equal. After that, programming exercises were conducted up to 30 minutes with each learning method. The content of the learning task in the exercise was the same for both the experimental group and the control group. In the experimental group, one or more four-choice toggles were added to each learning task. In the experimental group, a toggle corresponding to 10 or more dummy cards was given, and in the control group, only three dummy cards were given. The correct cards used for the answer is the same number for both the experimental group and the control group. In this experiment, we used the arithmetic operations and comparison operators as toggles. Each question was configured as one card and one statement. After the exercise, finally, the post-test was conducted up to 30 minutes to confirm the learning effectiveness. After the post-test, we conducted a questionnaire on the cognitive load by 6-point interval scale. The scoring method of pre- and post-test was the same as in the previous section.

6.2 Experimental Results

The results of the post-test are shown in Fig. 9, where the error bar means standard error. From Fig. 9, there was no statistically significant difference, but the control group without the toggle was higher than that of the experimental group. In the post-test, the control group had more subjects than the experimental group in the number of people who performed well in the post-test. When subjects' responses were examined, subjects in the experimental group tended to respond more unintentionally than the control group. From this, excessive error options may reduce the learning effectiveness.

Fig. 10 shows the extraneous cognitive load for learning each element in the additional experiment. From Fig. 10, it was revealed that the significant difference in ECL8 was found in Welch's t-test, and the experimental group had a significantly higher learning load than the control group. ECL8 is an extra load that occurs when the entire problem is grasped. For all the other toggles, the load on the experimental group tended to be higher than that on the control group, although this was not statistically significant. This suggests that the addition of the toggles may have created an extra load. These results suggest that the toggles themselves have no function to enhance the learning effect, and are likely to be useful only

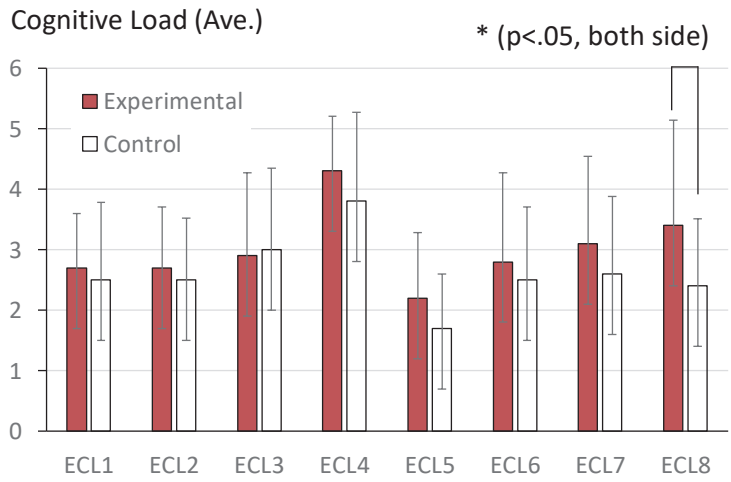

Figure 10: Extraneous cognitive load for learning each element in the additional experiment

as a method to reduce the number of dummy cards.

Fig11 shows intrinsic cognitive load and germane cognitive load in the additional experiment. ICL of the experimental group was higher than that of the control group, and the GCL was lower than that of the control group. This is an undesirable trend. Welch's t-test shows that the ICL of the experimental group is significantly higher than that of the control group. This is not surprising since the experimental group has more toggles than the control group. It is expected that the GCL would increase as the difficulty of the learning task increases, since there are more elements to be learned. However, this was not the case in this study. This suggests that the toggles themselves are not effective in increasing the learning effect. Furthermore, it is suggested that the difficulty level of the learning task should be appropriate for the learner. In this experiment, the experimental group had false choices corresponding to 10 dummy cards even using the toggle function. On the other hand, there were 3 dummy cards in the control group. On this point, we expected the toggle function to solve the negative effects of increasing the number of dummy cards, but the expected results were not shown. Therefore, even if the toggle function is introduced, work burden similar to that of the dummy card may be generated, and it may not have been possible to concentrate cognitive resources on essential learning. When the questionnaire results were examined, the control group obtained higher results in the item "Can you concentrate on learning?" than the experimental group although it was not a statistically significant difference. Including the questionnaire results, the toggle function is likely not versatileness for controlling the cognitive load.

The experimental results suggested that the control group without the toggle had a higher learning effect than the experimental group. Moreover, based on the results of the Fig. **??**, it became clear that the toggle function had a significantly higher learning load than the control group. Based on the above results, we need multiple dummy options to realize the appropriate structured task [15], and we should use toggles to better present these options to the learner. That is, preparing a dummy option without intention may inhibit appropriate learning.

These results suggested that the number of dummy options may be related to the work load. Moreover, it was suggested that the toggle function could reduce the cognitive load

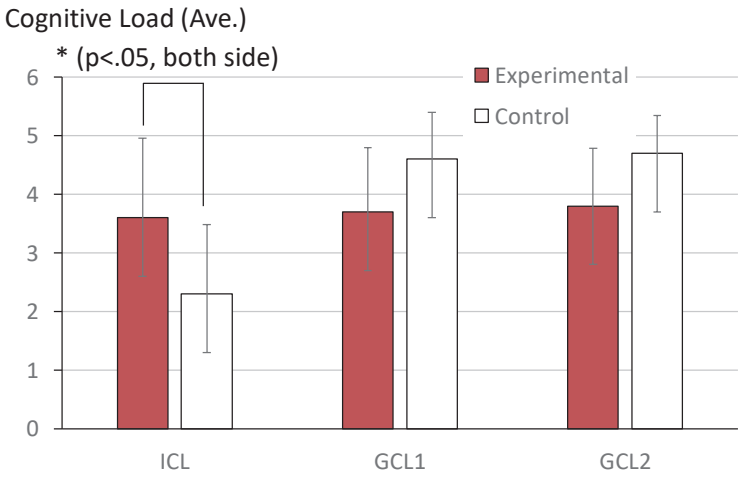

Figure 11: Intrinsic cognitive load and germane cognitive load in the additional experiment

of the workload and support the allocation of cognitive resources to the essential learning to think the structure. From the above, it was suggested that the toggle function could reduce the cognitive load to rearrange cards. However, it is not effective for all, and there is a possibility that there is no effect when preparing options other than learning intended by the instructor. We previously thought that the toggle function could solve the negative effect of increasing the number of false choices, but the effect of the toggle was not versatileness. In order to make the learner think more appropriately, it is thought that it is necessary to narrow down the false choices in advance to some extent in addition to just increasing the dummy statements. If we want learners to learn appropriate learning, it is considered effective to set a dummy card that follows the learning the instructor intends.

The usefulness of the toggle was not explained in terms of cognitive load in previous studies [17]. Therefore, this is the most significant contribution of this paper.

7 Conclusion

This paper conducted comparative experiments to verify whether the toggle function leads to suppressing nonessential cognitive load for COPS. In order to accomplish the purpose, firstly, we newly designed the toggle functionality to be available in COPS. Next, we implemented the toggle function on the actual COPS. Finally, we clarified the usefulness of the toggle function through comparative experiments. As the result of the comparison experiment, though the experimental group using the toggle function did not show a statistically significant difference in all learning tasks, the post-test average score of the experimental group was higher than the control group. Considering the result of the questionnaire survey clarifying that the toggle function reduces the cognitive load of the workload of sorting cards, the learning effectiveness of the toggle function might suggest. The questionnaire survey also suggested that reducing the cognitive load of the workload would help to reduce the cognitive load of the whole learning system. Of course, we don't expect the toggle function to be good at everything. What we want to clarify is the condition of the learning task for which the toggle function is effective. This study revealed some of the conditions under which the toggle function effectively reduces the cognitive load induced by dummy

cards. These results of this paper are only some of the conditions under which the toggle function is effective in the learning task. However, further research using a similar approach will clarify the characteristics of the toggle function. The result of this paper would be the first step in this direction, so the academic value of this paper should be accepted as sufficient.

Acknowledgment

This work was partly supported by Japan Society for the Promotion of Science, KAKENHI Grant-in-Aid for Scientific Research(C), No. 22K02815 and No. 23K02697.

References

[1] Lisack S. K, Helping Students Succeed in a First Programming Course: A Way to Correct Background Deficiencies. International Association for Computer Information Systems Conference,(1998),(in Mexico).

[2] T S. Garner, A Tool to Support the Use of Part-Complete Solutions in the Learning of Programming, Proceeding de conference, pp.222-228(2001).

[3] Matsumoto.S,Hayashi.Y,Hirashima.Y, Development of a Card Operation-Based Programming Learning System Focusing on Thinking between the Relationsof Parts, IEEJ Transactions on Electronics, Information and Systems,Vol.138 No.8 pp.999-1010(2018)

[4] J. Sweller, J. Merrienboer, F. Paas, Cognitivearchitecture and instructional design, Educa- tional psychology review, Vol.10, No.3, pp.251-296（1998）

[5] T. Hirashima, Y. Hayashi, S. Yamamoto: “Triplet Structure Model of Arithmetical Word Problems for Learning by Problem-Posing”, Int. Conf. on Human Interface and the Management of Information, pp.42- 50 (2014).

[6] Murakami.R, Morinaga.S, Matsumoto.S, Hayashi.Y, Hirashima.Y,Learning Effect of Card Operation-Based Programming Learning System,(in Japanese).

[7] D. Parsons, P. Haden, Parson’s programming puzzles: a fun and effective learning tool forrst programming courses, Proceedings of the 8th Australasian Conference on Computing Education, Vol.52, pp.157-163 (2006)

[8] Pehkonen E., Use of Open-Ended Problems in Mathematics Classroom. Research Report 176, 1997.

[9] Okimoto K., Matsumoto S., Yamagishi S., Kashima T., Developing a source code reading tutorial system and analyzing its learning log data with multiple classification analysis, Journal of Artificial Life and Robotics, Vol.22, Issue 2, pp 227-237 (2017).

[10] Kashihara A., Soga M., Toyoda J., A Support for Program Understanding with Fill-in-Blank Problems, Transactions of Japanese Society for Information and Systems in Education 15(3), pp.129-138 (1998), In Japanese.

[11] Funabiki, N., Zaw, K. K., Ishihara, N., Kao, W. C. (2017). A Graph-based Blank Element Selection Algorithm for Fill-in-Blank Problems in Java Programming Learning Assistant System. IAENG International Journal of Computer Science, 44(2), IJCS_44_2_14.

[12] Iwamoto T., Matsumoto S., Hayashi Y., Hirashima T., Examining Presentation Method of Question's Requirement for Game Development-Based Programming Learning Support System, Proc. of AROB 24th, GS5-3, pp.130-133 (2019).

[13] Yamaguchi, T., Oba, M. (2020). Measurable Interactive Application to Find Out User Recognition and Strategy when Problem Solving. J. of Software, 15(1), 12-22.

[14] YAMAGUCHI, T., NIIMI, A., OBA, M. (2021). Let's Measure and Analyze Thinking Process: Temporal Co-Occurrence Analysis for Learning Process. Journal of Japan Society for Fuzzy Theory and Intelligent Informatics, 33(4), 117-125.

[15] Funaoi.H, Ishisa.K, Fukuda.H, Yamasaki. K, Hirashima.T, Comparison of Two Concept-Mapping Methods : How They Effect on Learners' Memorization?, Japan Society for Educational Technology, Vol.35, pp.125-134 (2011), (in Japanese)

[16] Morinaga.S,Matsumoto.S,Hayashi.Y,Hirashima.Y, Effects of Toggle Function of Card Operation-Based Programming Learning System for suppressing adverse effects caused by an increase in error options, The 20th IEEE Hiroshima Section Student Symposium, A2-12, pp.132-135 (2018), (in Japanese)

[17] H. Shigematsu, T. Maeta, T. Okuhira, S. Matsumoto, Evaluating the Effectiveness of Toggle Function for Card Operation-Based Programming Learning Support System, Proc. of 2022 11th International Congress on Advanced Applied Informatics, pp.244-249 (2022).