International Journal of Learning Technologies and Learning Environments
International Institute of Applied Informatics
2018, Vol.1, No.1, P.41 - 60

Application of Programming Education Support Tool pgtracer for Homework Assignment

Tetsuro Kakeshita*, Miyuki Murata †

Abstract

We have developed a programming education support tool pgtracer implemented as a Moodle plug-in. Pgtracer provides fill-in-the-blank programming questions to the students and collects student log to analyze student's learning process and understanding level. In this paper, we apply pgtracer at an actual programming course to provide homework assignments to the students. We develop fill-in-the-blank questions based on the course syllabus at each week. Student activities on pgtracer are analyzed using various functions provided by pgtracer. The analysis results are utilized to analyze understanding level of each student and to develop questions for the succeeding weeks. We also provide the analysis result to the teacher about the activities and achievement of the students for better collaboration between lecture and homework. We received positive feedbacks from both of the teacher interview and student survey about the usefulness of pgtracer as programming education support tool.

Keywords: Learning analytics, Computer programing education, Homework assignment, E-learning, Education support tool, Moodle, Fill-in-the-blank question

1 Introduction

Computer programming is essential at national institute of technology and university majored in science and engineering. Recently, Japanese government announced a plan to start computer programming education at elementary school in 2020 in order to develop citizens who can fully utilize computer and various IT services.

However we often find students with low programming skill at an actual class. Many of them do not understand basic programming concepts such as loop, function and pointer. Although individual support for such students is quite important, there is a financial limitation so that the number of teachers, support staffs and teaching assistants is not enough at many educational institutions.

* Saga University, Saga, Japan
† Kumamoto National College of Technology, Yatsushiro, Japan

We proposed a programming education support tool pgtracer [1][2] to support teachers and students to overcome these difficulties[‡]. Pgtracer is developed as a Moodle plug-in so that the student can learn computer programming at any time and place as long as the internet connection and personal computer is provided. Pgtracer utilizes fill-in-the-blank questions composed of program and trace table (Figure 1). The green texts written in Japanese are the comments so that the readers can skip them. A trace table represents execution order of the steps and the value of each variable at each step. This means that students need not develop a computer program from the scratch. When a student fills the blanks and submits the answer, pgtracer automatically executes the filled program and compares the execution result with the right answer. At the same time, pgtracer collects student log of filling the blanks so that teacher can analyze the collected log to check the activity and achievement of the individual student as well as those of the entire class [3]. This is a typical application of the learning analytics.

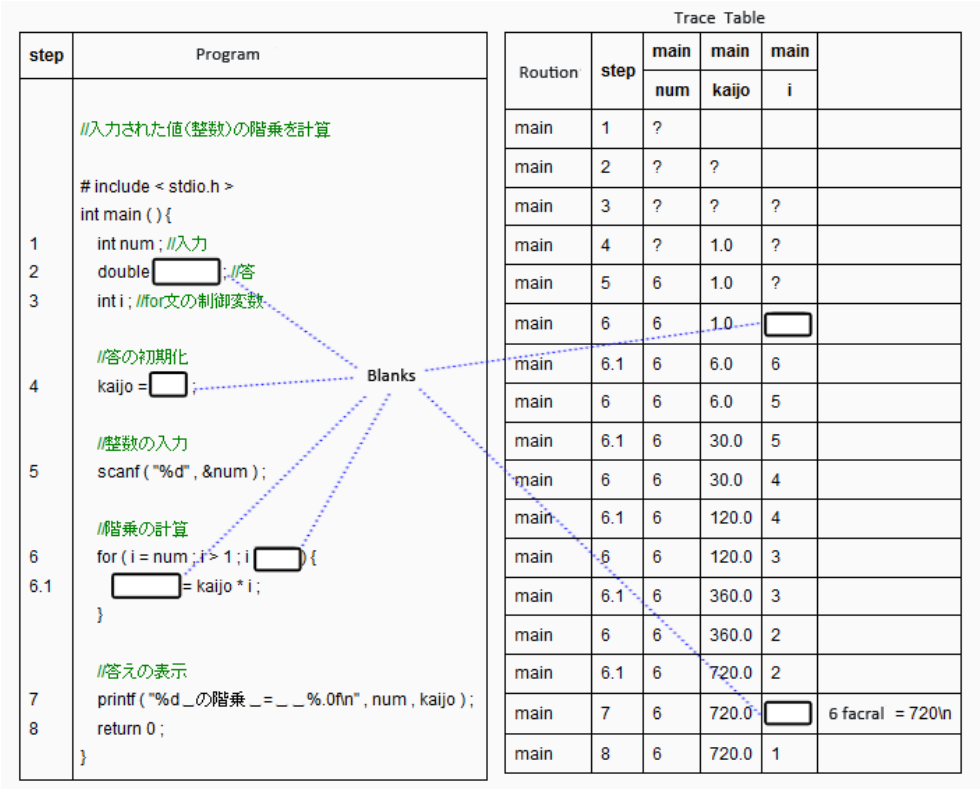


Figure 1: A Fill-in-the-Blank Question of pgtracer

In this paper, we utilize pgtracer at an actual programming course at Kumamoto National Institute of Technology. While our previous papers [1-3] focused on functionality of pgtracer, this paper focuses on our experience to utilize pgtracer. We develop three fill-in-the-blank questions each week based on the course syllabus. The questions are assigned to the students as homework. Such homework or learning after the class is mandatory at higher education in Japan so that we utilize the homework to improve and evaluate programming skill of the students.

We also monitor the student activity using pgtracer and provide feedbacks to the teacher. Student activity is analyzed using the number of students to solve each question. Student

[‡] We invented the tool name so that the name "pgtracer" is not an abbreviation.

achievement is analyzed using distributions of score and required time of each question and blank. The analysis result of the student log is utilized to clarify understanding level of each student and to improve the questions for the succeeding weeks. We also interviewed the teacher and selected students having various levels of achievement. A survey questionnaire is conducted to collect comments and opinions from the students.

Nishida et al. proposed a programming education environment PEN for novices [4]. A student must describe a program from the scratch so that the learning time tends to be longer compared with our approach. Funabiki et al. proposed a blank element selection algorithm and developed JPLAS to provide fill-in-the-blank questions of Java programs [5]. However JPLAS does not support trace table and does not provide log analysis functions.

Itado et al. proposed a method to evaluate student's ability from exercise sentence sorting or corresponding coding [6]. This method uses exercise scores. However required time and answer process are not considered. Although Malliarakis et al. proposed a framework that guides incorporation of learning analytics mechanisms in computer programming education [7], the detail of the collected data is not presented. Helminen et al. developed a web-based Python programming environment which collects and analyzes student's programming process [8]. This environment also does not provide a function to estimate student's achievement level. Maeda et al. analyze programming process by collecting keyboard input [9]. Although this approach is useful to understand behavior of each student, it is difficult to analyze achievement of the entire class.

This paper is organized as follows. The next section introduces the concept of fill-in-the-blank question, programming education process and the analysis functions provided by pgtracer. Preparation policy of the fill-in-the-blank questions is explained in Section 3. In Section 4, we report the student behavior to the homework assignment. In Section 5, we analyze correlation between mid-term examination and pgtracer scores. Difficulty level analysis of the questions and blanks are discussed in Section 6. We conducted a survey questionnaire and student interview as well as interview to the teacher. The results are reported and discussed in Section 7.

2 Programming Education Support Tool pgtracer

2.1 Fill-in-the-Blank Question

A fill-in-the-blank question is composed a pair of a C++ program and a trace table representing execution order of steps and the value of each variable at each step (Figure 1). Various types of blanks can be defined within a fill-in-the-blank question as explained below [1].

- A blank at a value of a variable within the trace table can be used to check student's recognition of change of the value of the variable.
- A blank at a step number, a variable name or a routine name within the trace table can be used to check student's recognition of execution order of statements or the name of the corresponding variable or routine.
- A blank at a single token within the program, such as variable name, operator, reserved word, etc. can be used to check student's ability of elementary programming.

- A blank at a sequence of tokens within the program such as expression, statement, compound statement, routine can be used to check student's ability of more advanced programming.

There is a case where more than one right answer exists for a blank within a program. Pgtracer evaluates all of such answers to be correct as long as the execution result of the program and the original trace table are consistent.

2.2 Programming Education Process using pgtracer

Figure 2 represents the programming education process using pgtracer. It also represents the learning process of computer programming using pgtracer. A fill-in-the-blank question is composed of a program, a trace table, a mask for the program and a mask for the trace table. They are described using XML. We separate a program and a mask for the program so that multiple masks with different difficulty levels can be defined for a single program. Trace table and the corresponding masks are separated for the same reason. Question DB contains valid combinations of the XML files.

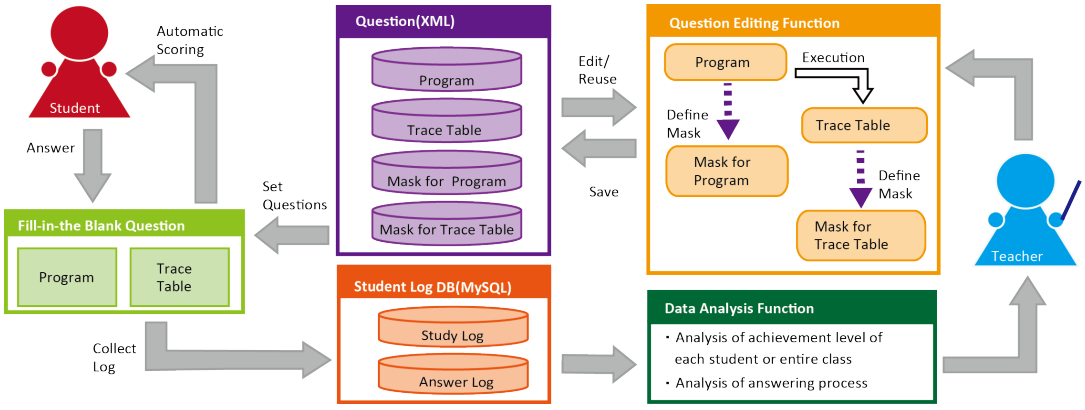


Figure 2: Programming Education Process utilizing pgtracer

When a student answers to a question, the system automatically evaluates the answer and feedbacks the score to the student. The student then can view the right answer. At the same time, pgtracer collects the log data of the answering process and the score.

The collected data is utilized to analyze the achievement level and the learning process of each student and the entire class. Pgtracer provides various analysis functions for the collected data as explained in Section 2.3. The teacher uses the analysis functions to improve the educational contents including fill-in-the-blank questions and the instruction to each student.

Pgtracer also provides functions to create and edit XML files representing a program, a trace table, a mask for program and a mask for trace table. Pgtracer automatically converts a program to the corresponding XML file. Then a teacher provides input data file to execute the program. Then pgtracer generates the XML file representing the corresponding trace table using the input data file. Next the teacher can create and edit program mask and trace table mask using pgtracer. The teacher can specify masks and hidden portion of the program and the trace table. Pgtracer then generates XML files representing the masks for program and trace table.

Thus a teacher can create and edit XML files without XML knowledge.

The teacher can define various options of the created questions. The option includes question mode (self-learning mode and examination mode), show/hide of the correct answer after automatic scoring, show/hide of the analysis function to the students, coloring of the corresponding step when a student selects a blank within a trace table.

2.3 Student Log Analysis Function

Pgtracer provides seven functions in order to analyze student log from various viewpoints [3]. These functions are categorized into three types. Five analysis functions are introduced in this section.

2.3.1 Analysis Function of Student

The student utilization function (Figure 3) illustrates the overall behavior of the students in terms of the average score and required time to answer the questions. This function is useful to understand the overall achievement and workload of the students.

The analysis function of a student (Figure 4) summarizes the achievement of the selected student. Note that the actual student name is excluded from the figure because of the privacy reason. This function is useful to understand the overall achievement of each student.

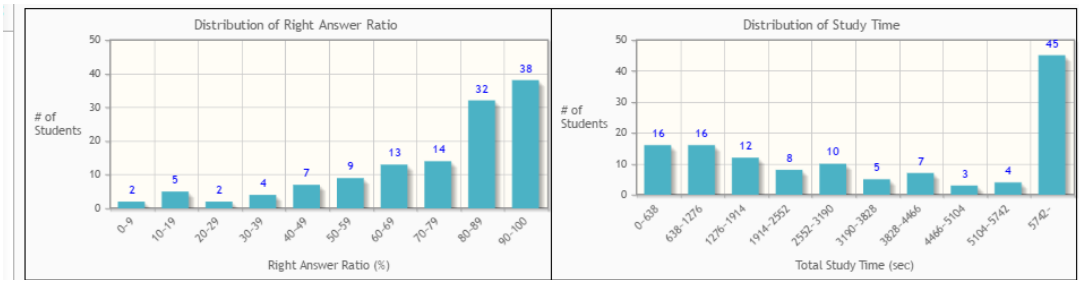


Figure 3: Student Utilization

# of Attempted Questions	Total # of Attempts	Total Study Time	Right Answer Ratio (%)
21	32	2:3:1	93.71

Copy Excel CSV Print Search:

Theme	Title	Level	First Attempt			Attempt with Max Score			Attempt Number	
			Score (%)	Time to Start Learning	Required Time	Score (%)	Time to Start Learning	Required Time		
1 (01)	問題(1)-3	1	81	12:21:50	0:8:22	98	20:09:27	0:8:8	2	Show Detail
実験 1 (01)	問題(1)-1	1	73	2016/10/12 08:31:10	0:5:58	100	2016/10/12 12:17:04	0:2:5	3	Show Detail
実験 1 (01)	問題(1)-2	1	25	2016/10/12 12:09:22	0:7:10	91	2016/11/25 19:57:32	0:3:29	3	Show Detail
実験 1 (02)	問題(2)-3	1	90	2016/11/25 20:33:52	0:2:48	90	2016/11/25 20:33:52	0:2:48	1	Show Detail

Showing 1 to 125 of 125 entries

Figure 4: Analysis Function of a Student

2.3.2 Analysis Function of Each Question and Blank

The analysis function of a question (Figure 5) illustrates the averages and distributions of the student score and required time. The averages and distributions are separately calculated for the

case of the first attempt and the case of the maximum score. The average and distribution for the first attempt correspond to the actual ability of the students, while those for the maximum score can be used to recognize students' effort since many of the students repeat the trial until they earned the 100% score.

The analysis function of a blank within a question (Figure 6) represents the right answer ratio and average required time for each blank of a question. Then it is possible to recognize difficulty level of the blanks. Each of the blanks is assigned a unique number as illustrated.

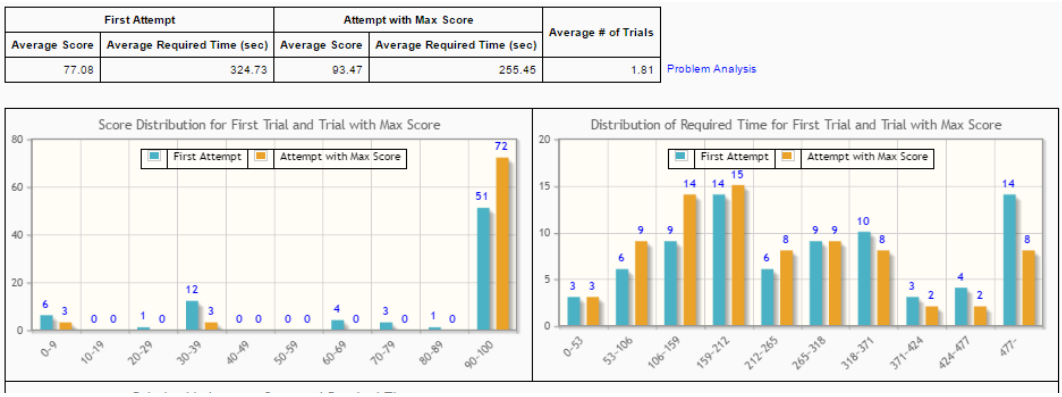


Figure 5: Analysis Function of a Question

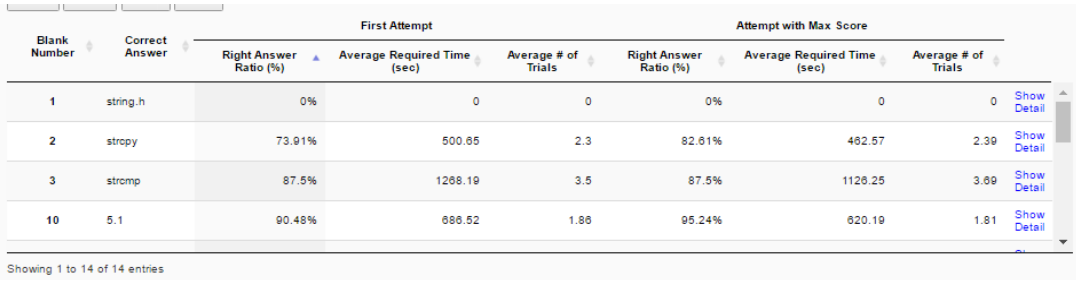
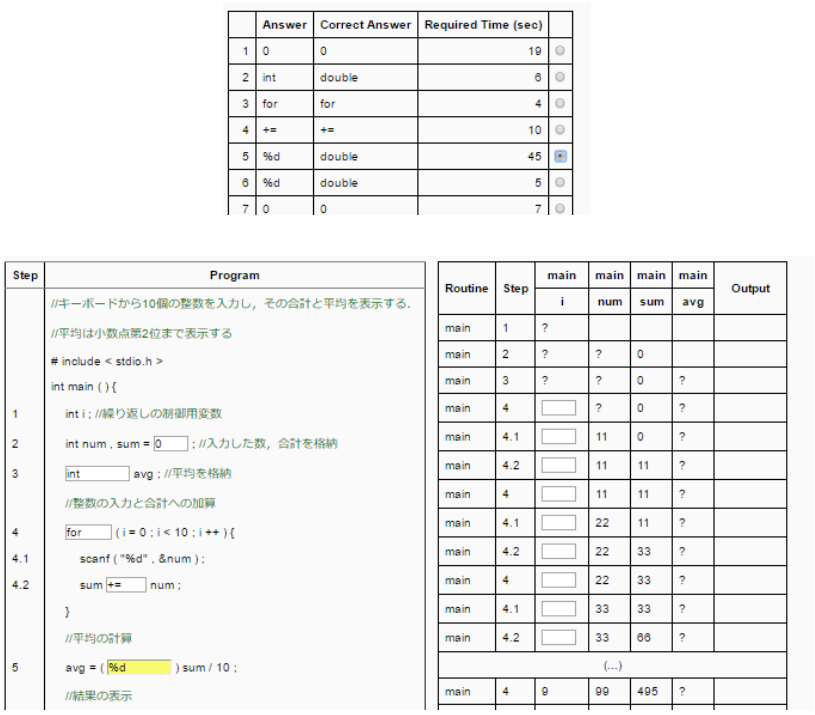


Figure 6: Analysis Function of a Blank within a Question

2.3.3 *Process Analysis Function*

The analysis function of an answer process (Figure 7) represents the intermediate state of the selected student answer. The upper table represents the sequence of filled blanks of the selected answer associated with the required time to fill each blank. When a teacher selects a blank, pgtracer illustrates the state of the student answer when the student filled the selected blank at the lower table representing the question. The yellow blank is the one recently updated.



	Answer	Correct Answer	Required Time (sec)	
1	0	0	19	
2	int	double	6	
3	for	for	4	
4	+=	+=	10	
5	%d	double	45	
6	%d	double	5	
7	0	0	7	

Step	Program
	//キーボードから10個の整数を入力し，その合計と平均を表示する．
	//平均は小数点第2位まで表示する
	# include < stdio.h >
	int main ( ){
1	int i ; //繰り返しの制御用変数
2	int num , sum = 0 ; //入力した数，合計を格納
3	int avg ; //平均を格納
	//整数の入力と合計への加算
4	for (i = 0 ; i < 10 ; i ++){
4.1	scanf ("%d" , &num) ;
4.2	sum += num ;
	}
	//平均の計算
5	avg = (%d) sum / 10 ;
	//結果の表示

Routine	Step	main i	main num	main sum	main avg	Output
main	1	?				
main	2	?	?	0		
main	3	?	?	0	?	
main	4		?	0	?	
main	4.1		11	0	?	
main	4.2		11	11	?	
main	4		11	11	?	
main	4.1		22	11	?	
main	4.2		22	33	?	
main	4		22	33	?	
main	4.1		33	33	?	
main	4.2		33	66	?	
			(...)			
main	4	9	99	495	?	

Figure 7: Analysis Function of an Answer Process

3 Preparation of Fill-in-the-Blank Questions

3.1 Course Outline

The experiment was performed for the 218 students at the second academic year of Kumamoto National Institute of Technology from October 2016 to February 2017. The students are majored in one of the following areas: mechanical engineering, electronics, civil engineering, architecture, bio technology and chemistry. They are learning computer programming using C language at the common course named “Fundamental of Computer Science”. The course is composed of 30 weeks of 90 minutes classes each week. The course is provided by two teachers, one is giving lecture and exercise and the other, one of the authors, is supporting the course by utilizing pgtracer for the department of biological and chemical systems engineering.

Table 1 represents course outline of “Fundamental of Computer Science”. Our experiment was performed during the second semester. We introduced pgtracer to the students at the end of June 2016 and registered the students to Moodle at the beginning of July 2016. We provided 9 questions including tutorial question and announced the student to utilize pgtracer.

Table 1: Lecture Plan

First Semester		Second Semester	
Week	Contents	Week	Contents
1	Fundamentals of Computer	1	One Dimensional Array
2	Representation of Numeric Values	2	
3	Flowchart	3	Two-Dimensional Array
4	Constant, Variable, Assignment	4	
5	I/O (printf, scanf)	5	Pointer

First Semester		Second Semester	
6	Type and Operator	6	
7	Conditional Branch	7	Function
8	Mid-Term Examination	8	Mid-Term Examination
9	Conditional Branch	9	Function
10	for Statement	10	
11	while Statement	11	Variable Scope
12	do while statement	12	File
13	break, continue, switch	13	Struct
14	Exercise	14	
15	Examination	15	Examination

3.2 Preparation Policy of Fill-in-the-Blank Questions

We prepared the fill-in-the-blank questions for the experiment according to the following policy.

- We provide three questions for each of the weeks 1-10 in the second semester.
- Each question is usually based on the teaching contents of the corresponding week. We utilize the same or similar program taught at the corresponding week for the question. However we sometimes include questions of the previous weeks to check the student achievement.
- The question is presented in the self-learning mode. Students can know whether their answers are correct or not just after their filling of each blank, since pgtracer instantly evaluates each blank just after a blank is filled in the self-learning mode.
- Although pgtracer supports program mask for an entire statement, we restrict program mask for a token or a part of an expression. This is because that the students are programming beginners.
- Trace table mask are defined for a set of consecutive cells of the same column. This is because that the value of a variable at the previous or next step provides hints to the students.
- There are the cases that the same topic is taught for two weeks. Then more complex questions are assigned for the latter week which partially contains algorithm components.

Table 2: List of Questions

Week	Question	Description
1	(1)-1	Calculate and print sum, difference, multiplication, division and reminder of two integers
	(1)-2	Show absolute value of the input integer and whether it is even or odd.
	(1)-3	Calculate and print 1-th power to 5-th power to the input integer.
2	(2)-1	Calculate and print sum and mean value of the 10 input integers.
	(2)-2	Store 10 integers to an array and print them in the reverse order.
	(2)-3	Show the string stored in a character array by converting upper case letter to lower case and vice versa
3	(3)-1	Show the number of negative integers until 5 positive integers are

Week	Question	Description
		found.
	(3)-2	Concatenate and print two strings
	(3)-3	Show the point having the largest distance from the origin among three.
4	(4)-1	Calculate and store 2 to the 0-th power to the 8-th power and show the selected value.
	(4)-2	Show the length of the strings until "end" is detected.
	(4)-3	Calculate and print sum of the edge length of four triangles stored in a two-dimensional array.
5	(5)-1	Same as (3)-3 but we delete comments explaining variables
	(5)-2	Assignment and print variables through pointer.
	(5)-3*	Add two float variables using pointer.
6	(6)-1	Same as (4)-3 but we delete comments explaining variables
	(6)-2	Calculate and show area of a circle from the diameter using pointer.
	(6)-3	Print array elements using pointer.

We found during the experiment that the students tend to quit the exercise when the width of the trace table is too wide to be displayed on the computer screen. Considering this, we improve the questions such that the program and trace table can be displayed within the screen by adjusting the number of array element, dividing a long comment into two or more lines and by replacing long names of a variable or a function with shorter ones.

The questions are developed by the two authors. One creates the questions as a supporting teacher of the target course and recognizes the actual teaching contents and progress of the course. The other author reviews the created questions based on the validity of the place of the blanks, difficulty level and consistency between comments and source code.

We developed 30 questions for 10 weeks. Table 2 represents the description of the questions for weeks 1-6. The same option is used for the questions of the same week. The common options throughout the all questions are self-learning mode and use coloring of the step corresponding to the current blank in the trace table. We basically show the correct answer after the students fill a blank. But we hide the correct answer to the students for the questions on weeks 4 and 9 for the comparison purpose.

The total number of blanks within a question is between 6 and 14. The average number of blanks is 10. The average time to create three questions for a particular week is about 2 hours. We spent 1 hour to write the target programs and 1 hour to register and edit the program and trace table masks using pgtracer.

4 Analysis of Student Behavior

Figures 8 to 10 represent student behavior at each week. We can observe from Figure 8 that the difficulty levels of the questions are appropriate. Figure 9 represents the sum of the required time to solve the three questions at each week. We can observe from the figure that student workload is not too heavy.

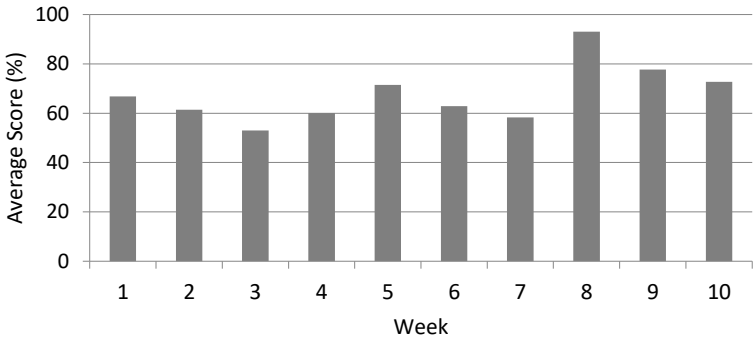


Figure 8: Average Score at Each Week

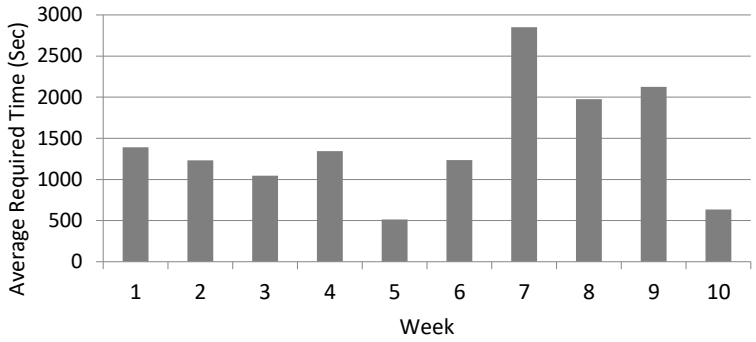


Figure 9: Average Required Time at Each Week

However the number of students solved the questions is decreasing as illustrated in Figure 10. This is mainly because that we are not teaching the course so that the homework score does not affect the final evaluation of the course. Thus we asked for the course teacher to use some of the homework questions at mid-term examination and to announce that to the students. Then about 50% of the students solved the questions. However the number of students drops at week 7, since the mid-term examination is scheduled at week 8 so that many students focused on the exam preparation than the homework. We expect that the situation will change by utilizing the homework score for the final evaluation.

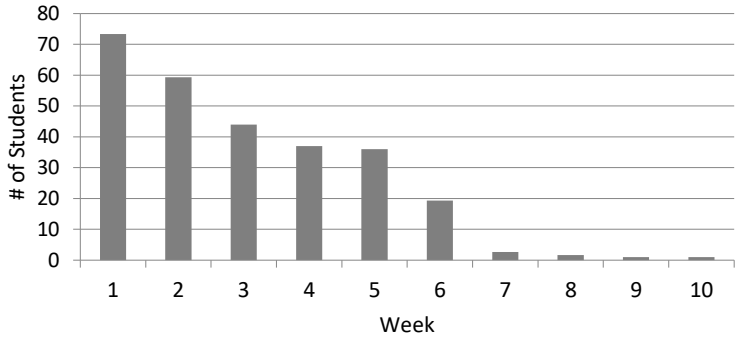


Figure 10: Number of Students Solved the Homework at Each Week

Since the number of students solved the homework is quite small after week 7, we shall analyze the student log between weeks 1 and 6 in the succeeding part of this paper.

Figure 11 represents the score distribution of question (2)-3. The highest peak is between 90 and 100%. The second peak is between 0 and 10%. This tendency is common among most of

the questions. The highest peak indicates that the question is not difficult.

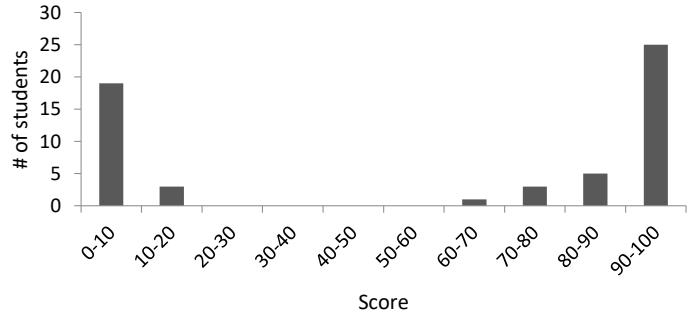


Figure 11: Score Distribution of Question (2)-3

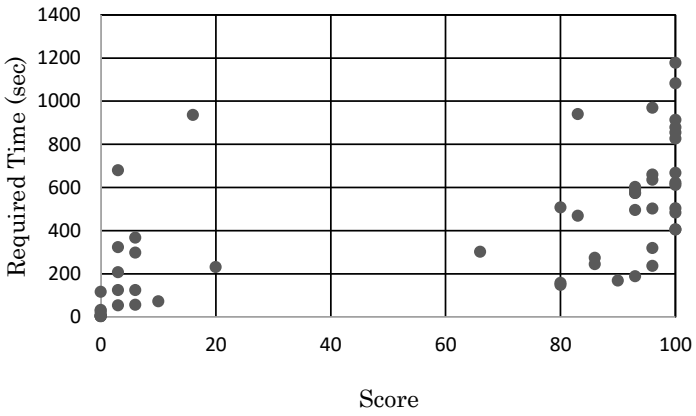


Figure 12: Relationship between Score and Required Time of Question (2)-3

Figure 12 represents the relationship between score and required time of the same question as Figure 11. The required time of the answers with low score tend to be short. This implies that the students only view the question during preparation of the examination and did not actually solve the question. Although some of the students with low score took more than 10 minutes, we observe the student log using the process analysis function and find that those students actually quit to fill the blanks at an early stage of their solving the question.

We found two patterns of the required time distribution. One is the distribution pattern with twin peaks at the shortest and longest time intervals (Figure 13). Questions (2)-3, (3)-1, (3)-2 and (3)-3 belong to this pattern. The average score of these questions is between 43.75 and 63.25. The average required time exceeds 330 seconds (5.5 min.). The other is the pattern with a single peak at the middle of the time intervals (Figure 14). Questions (2)-2, (5)-2 and (5)-3 belong to this pattern. The average score of these questions is between 69.88 and 79.81. The average required time is between 136 and 306 seconds so that these questions are easier than the questions belonging to the first pattern. We observe that the students tend to give up solving the questions if they feel that the question is difficult or complicated.

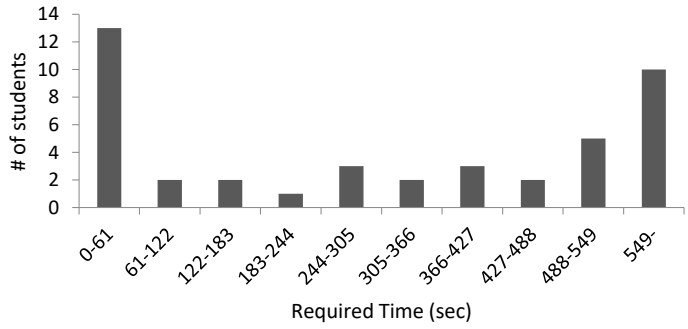


Figure 13: Distribution Pattern with Twin Peaks of the Required Time

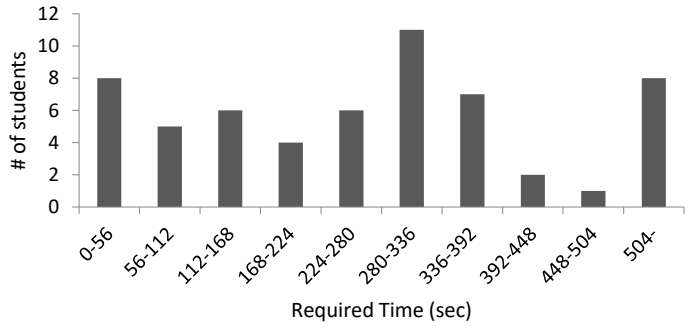


Figure 14: Distribution Pattern with Single Peak of the Required Time

5 Correlation Analysis between Mid-Term Examination and pgtracer Exercise Score

We shall analyze the relationship between the mid-term examination and the exercise using pgtracer in this section.

Figure 15 represents the relationship between the mid-term examination and the pgtracer scores. The correlation coefficient between these two values is 0.39, which means that there is a weak correlation between them.

Mid-term examination score is high but the pgtracer score is low for the two students within the solid circle. Average required time to solve a question is 58 seconds for one student, which is far less than the average required time. The other student answers to the 5 questions within 2 minutes among the 11 questions he solved. These facts indicate that the learning attitude of these two students is not positive.

On the other hand, the average required time is more than 6 minutes for the two students within the dashed circle. These two students rewrite the same blanks many times while solving the pgtracer exercise. The fact indicates that the student is making trial and error while solving the questions. We can say that the programming skill of these students is not high because skilled students do not require trial and error. However we can say that the learning attitude of these students is quite positive.

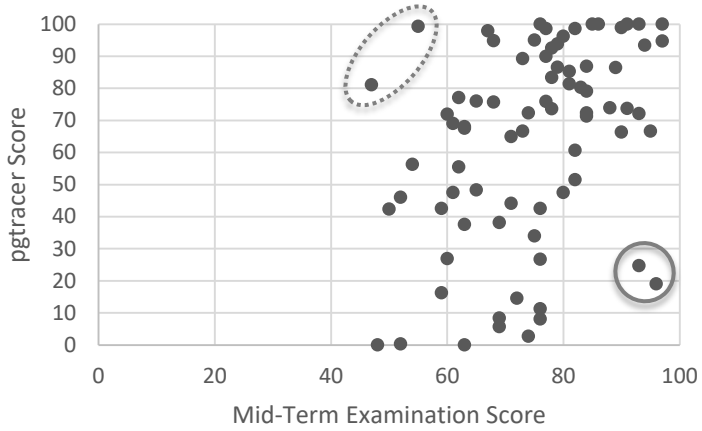


Figure 15: Relationship between Mid-Term Examination Score and pgtracer Score

Table 3 represents the difference between the examination scores of weeks 8 and 15 classified by the number of solved questions of the exercise between weeks 1 and 6. We can observe that the examination score of the week 15 is improved for the students who solved more questions during the experiment. This is an evidence that pgtracer is useful to improve programming skill of the students.

Table 3: Relationship between # of Solved Questions and Examination Score

# of Solved Questions	Average Difference of Exam Score (Final Exam – Mid-Term Exam)
18	+2.47
1 ～ 9	+0.89
0	−0.18

6 Difficulty Level Analysis of Question and Blank

In this section, we shall analyze the difficulty level of the questions and blanks. The difficulty level of a question or a blank is estimated using the right answer ratio and the required time to fill the blank(s).

We shall first analyze three questions whose average required time is more than 5 minutes: Questions (3)-1, (3)-2 and (3)-3.

The blanks defined in a do-while statement require the longest time in question (3)-1. Frequency of using a do-while statement is not high so that understanding level of the students tends to be low. We defined two blanks for “do” and “while” within the program so that the students must keep consistency of the two blanks. 20% of the students’ answers are incorrect. Typical mistakes are confusion with for or while statements.

The blanks to ask correct names of the library functions related to <string.h> require the longest time in question (3)-2. Typical mistakes are the misspelling of the function names. Actually the average score of question (3)-2 is the lowest among the 18 questions in Table 2.

The blanks to ask correct indices of the array require the longest time in question (3)-3. The

right answer ratios are around 85%. Students need to understand the position of each element within the array in order to correctly fill the blanks.

Table 4: The Blanks which the Right Answer Ratio is less than 60%

Question (1)-3	Right Answer Ratio	43.2%
	Place of Blank	ans = ans * n within a for statement
	Typical Wrong Answers	No Answer (13 answers), n^i (4 answers), *i, *n, n*, etc.
Question (2)-3	Right Answer Ratio	53.3%
	Place of Blank	s[i]='\0' within a while statement
	Typical Wrong Answers	No Answer (8 answers), \0, 10, EOF, etc.
Question (4)-1	Right Answer Ratio	41.7%
	Place of Blank	Column to show output result within trace table
	Typical Wrong Answers	No Answer (6 answers), 024, 16, 64, 256, etc.

We shall next analyze the blanks whose right answer ratio is less than 60% illustrated in Table 4. The right answer ratio is defined by the number of right answers divided by the total number of answers filled to a blank. The "blank" answers are excluded from the calculation.

The extracted blank in question (1)-3 is a typical one using the for statement. Typical wrong answers are the incorrect use of the power operator by confusing the operator defined in Microsoft Excel. Students are also required to understand the iterative algorithm to calculate the power of the input value. Such combination of algorithm and operator is a typical place where inexperienced students are confused.

The blanks to ask the termination condition of a string require the longest time in question (2)-3. 46.7% of the answers are incorrect. Students could not write anything for 50% of such incorrect answers. Although question (2)-3 is a typical one for string manipulation using while statement, we found the students who did not precisely memorize termination symbol of a string literal.

The blanks to ask an output value within the trace table require the longest time in question (4)-1. Students are required to understand trace table to correctly fill the blank. However we find some students who do not understand the trace table. This is essentially because typical students are not familiar with trace table. At the same time, we also find that the students with high programming skill quickly understand trace table and utilize the information shown in the table as a hint to solve the question. This implies a potential correlation between the understanding level of the trace table and the programming skill. This is also indicated by Egi [10].

The explained topics that the achievement of the students is low can be regarded as candidates of future educational improvement. Pgtracer is thus useful to quantitatively analyze student achievement as explained in this section.

7 Overall Evaluation

7.1 Analysis of Questionnaire to the Students

We conducted a questionnaire to the students after the experiment. Table 5 represents the questions to the students. The number of answers was 118 (92.2% of the enrolled students).

Table 5: Questions to the Students

(1)	Average time to answer the questions per week
(2)	Average time to work on other exercise per week
(3)	Did you solve more than 9 questions from question (1)-1 to (6)-3 ?
(4)	If the answer to question (3) is yes, how was the difficulty level of the questions?
(5)	If the answer to question (3) is yes, was the exercise useful to learn computer programming?
(6)	If the answer to question (3) is no, select reason from the provided list.
(7)	If you select "others" in question (6), please specify the reason.
(8)	Provide comment and/or suggestion to improve pgtracer if you have some.

Consider questions (1) and (2). The students take 2 to 3 subjects other than the current subject (Fundamental of Computer Science) which assign homework to the students. Figure 16 represents the relationship between the spent time for the current subject and other subjects per week. The numbers within the bubbles represents the number of students corresponding to the answer represented by the x-y coordinate of the chart.

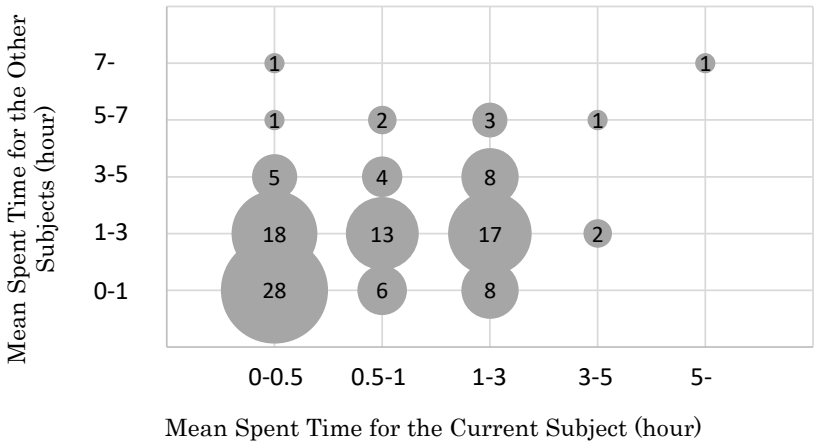


Figure 16: Relationship between the Spent Time per Week for the Current and Other Subjects

We can observe that 53 students (44.9%) spend less than 30 minutes per week for the programming exercise of the current subject. However there are 4 students (3.39%) who require more than 3 hours. Required time to solve the homework greatly differ depending on the students.

For question (3), 32.2% of the students solve more than 9 questions. Figure 17 illustrates the reason of not solving more than 9 questions collected from the remaining 67.8% of the students.

The most popular reason is that the students did not have enough time. Typical students are taking other subjects as well as other school activity such as club activity and student council activity. Also considering the reason that pgtracer score do not affect evaluation, we need to provide some kind of incentives to motivate students to work on pgtracer exercise.

Considering the reason from 15 students that the student did not understand how to use pgtracer, more explanation and instruction are required at the initial stage to introduce pgtracer. We re-

ceived the comments in question (8) that they did not understand the concept of trace table. Although we explained and demonstrated how to understand trace table, most of the students are not familiar with the concept of trace table within the conventional computer programming education. Some students did not understand that the trace table represents the execution order of the steps within a program and the value of the variables at each step.

On the other hand, 10 students answered that they already understand computer programming. More difficult problems should also be provided to motivate such type of students having sufficient programming fundamentals.

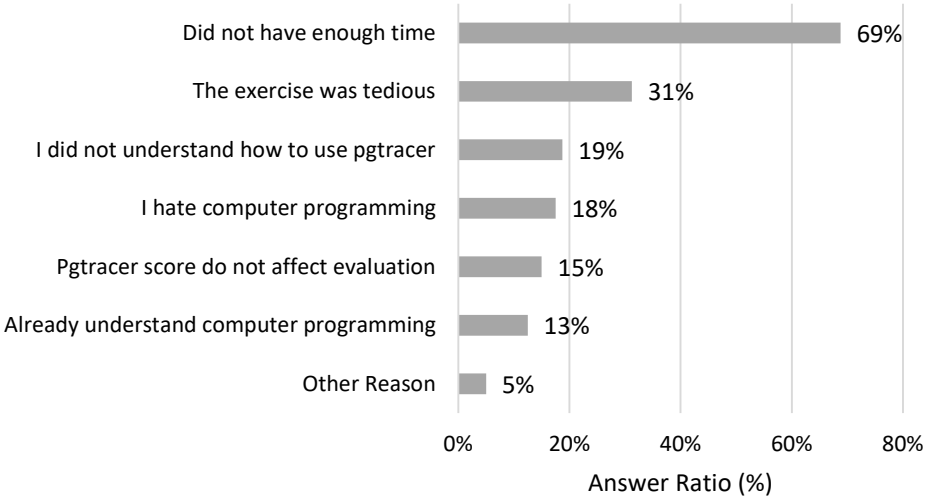


Figure 17: Reasons of not Solving More than 9 Questions

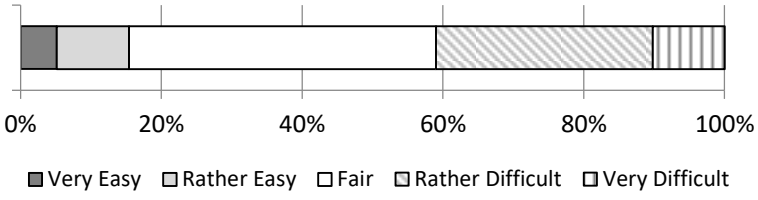


Figure 18: Difficulty Level of the Questions

Next we shall consider questions (4) and (5). As illustrated in Figure 18, 60% of the students answered that the pgtracer questions are easy or fair. This means that the provided questions were not too difficult and are at the appropriate level as a self-learning exercise.

We also find that 75% of the students answered that pgtracer is useful to learn computer programming as illustrated in Figure 19.

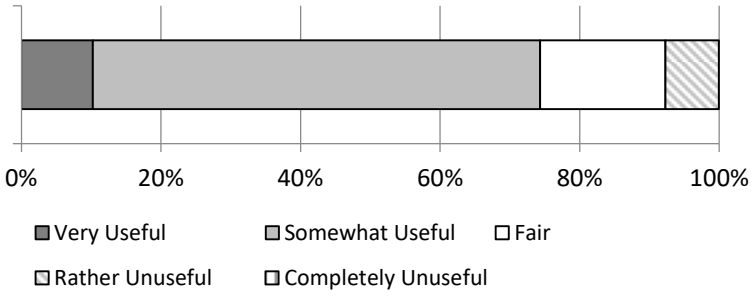


Figure 19: Is pgtracer useful to Learn Computer Programming?

We received some student comments collected from question (8).

The current pgtracer cannot show the entire columns of the trace table when the trace table is too wide. Although the entire columns can be displayed by shrinking the contents, the shrunk contents can be unreadable for the students. As a result, we sometimes find unfilled blanks which are not displayed on a web browser. We have notified this to the students but some of the students missed the notice. It is important to keep the width of the trace table small such that the entire columns of the trace table are displayed without shrinking the contents. This requires a careful selection of the displayed/hidden columns and the order of the displayed columns.

Some students provide comment that pgtracer should be utilized within the lecture. More collaboration is expected between the lecture and homework to facilitate self-learning.

7.2 Student Interview

We interviewed the students who solved more questions than average. They are expected to have high motivation so that we can efficiently obtain meaningful feedback. They said that fill-in-the-blank question is easy to solve than creating a program from the scratch. The self-learning mode quickly provides correctness judgement after filling a single blank. Such function is evaluated favorably so that we can conclude that the self-learning mode is an effective means to facilitate students' motivation. The students said that they can clearly understand basic grammar and behavior of the program. Such positive effect can be expected to the programming education using pgtracer.

Some students respond that they understand how to develop correct algorithm by reading comments associated to the program described according to our programming guidelines. Reading a good program is an effective means to learn computer programming. Pgtracer can facilitate careful reading of such programs by providing fill-in-the-blank question of the trace table.

We also interviewed the students with low achievement level. Some students said that they solved the problems through discussion with their friends. Although pgtracer score does not directly represents skill of the student in such case, we can expect to facilitate group learning or LTD (learning through discussion) using pgtracer.

We have sent e-mails to the students each time we provided new problems. Students said that

such messages are useful for them to continue motivation of learning.

7.3 Teacher Interview

We also interviewed the primary teacher of the target class and obtained the following comments. These comments also demonstrate effectiveness of programming education utilizing pgtracer.

- Since pgtracer allows defining various types of blanks depending on the understanding level of the students, I can create wide range of problems which can cover both programming beginners and experienced students.
- Since pgtracer shows skeleton of the program so that it is suitable to teach good programming style.
- Pgtracer is suitable to teach execution flow of a program.
- Difficulty level of the exercise is appropriate since basic problem and relatively difficult problem are both provided.
- Most of the students using pgtracer achieved the highest level score at the final examination. Some of these students achieved 10% more score compared with the first semester.
- It is recommended to improve pgtracer to visualize behavior of the program. Then the understanding level of the students will be improved.

8 Conclusion and Future Work

In this paper, we conducted a self-learning experiment using pgtracer at an actual programming course. We observed certain skill up of the students who actively solved the homework assignment. Both of the students and the primary teacher of the course respond that pgtracer is useful for programming education. Pgtracer questions can be stored to the DB for future reuse. Main obstacle to use pgtracer is the unfamiliarity to the trace table from the viewpoint of the students. Since the number of students solved our questions are not large enough, we did not perform statistical analysis of the collected data in this paper.

We also analyze student behavior and understanding level using the analysis functions provided by pgtracer. We detect various topics that the understanding of the students is not enough by analyzing the blanks with a low right answer ratio or a long required time. We find the tendency that the students repeatedly fill the same blank until they obtain a satisfactory score even when their understanding level is not enough.

The strings filled to such blanks represent the range of students' thought. This means that we can obtain valuable information about the topics that the students do not understand well or the students understand incorrectly by analyzing the candidate answers they used. Such information is useful to improve programming education. Students can also improve their programming skill by practicing their weak points that are detected through the analysis.

As a future work, we are planning to analyze intermediate student answers in order to improve pgtracer. We are also planning to investigate effective means to teach trace table to the students.

Appropriate incentive mechanism will also be investigated through PDCA cycle illustrated in Figure 2.

Acknowledgement

This research is supported by JSPS KAKENHI under grant numbers 16K01022 and 17K01036.

References

[1] T. Kakeshita, R. Yanagita, K. Ohta, "Development and Evaluation of Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Question", *Journal of Information Processing: Computer and Education*, Vol. 2, No. 2, pp. 20-36, Oct. 2016. (in Japanese)

[2] T. Kakeshita and K. Ohta, "Student Feedback Function for Web-based Programming Education Support Tool pgtracer", 5th Int'l Conf. on Learning Technologies and Learning Environment (LTLE 2016), Kumamoto, Japan , pp. 322-327, July 2016.

[3] T. Kakeshita and K. Ohta, "Student Log Analysis Functions for Web-based Programming Education Support Tool pgtracer", 17th Int'l Conf. on Information Integration and Web-based Applications & Services (iiWAS 2015), Brussels, Belgium, pp. 120–128, 2015.

[4] T. Nishida, et al., "Implementation and Evaluation of PEN: The Programming Environment for Novices", *Journal of Information Processing*, Vol. 48, No. 8, pp. 2736-2747, 2007. (in Japanese)

[5] N. Funabiki, et al., "Analysis of Fill-in-the-Blank Problem Solutions and Extensions of Blank Element Selection Algorithm for Java Programming Learning Assistant System", Proc. World Congress on Engineering and Computer Science (WCECS 2016), San Francisco, USA, pp. 237-242, 2016.

[6] Y. Itado, F. Harada and H. Simakawa, "Judgement of Learner Ability from Exercise Sentence Sorting and Corresponding Coding", Forum on Information Technology (FIT 2013) K-035, pp. 635–638, 2013. (in Japanese)

[7] C. Malliarakis, M. Satratzemi and S. Xinogalos, "Integrating Learning Analytics in an Educational MMORPG for Computer Programming", IEEE 14th Int'l Conf. of Advanced Learning Technologies, pp. 233–237, 2014.

[8] J. Helminen, P. Ihantola and V. Karavirta, , "Recording and Analyzing in-Browser Programming Sessions", 13th Koli Calling Int'l Conf. on Computing Education Research, pp. 13-22, 2013.

[9] K. Maeda and Y. Nakano, "An Analysis of the Programming Process", *Journal of Japan Society for Educational Technology*, Vol. 19, No. 3, pp.171-180, 1995. (in Japanese)

[10] T. Egi and A. Takeuchi, "Development and Evaluation of Debugging Support System of Guide Tracing for Beginners", *Journal of Japan Society for Educational Technology*, Vol. 32,

No. 4, pp. 369-381, 2009. (in Japanese)