Analysis of Student Activity and Its Effect Utilizing Programming Education Support Tool pgtracer

Miyuki Murata
Faculty of Liberal Studies
National Institute of Technology,
Kumamoto College
Yatsushiro, Japan
m-murata@kumamoto-nct.ac.jp

Naoko Kato
Faculty of Liberal Studies
National Institute of Technology,
Ariake College
Omuta, Japan
naoko@ariake-nct.ac.jp

Tetsuro Kakeshita
Graduate School of Information
Science
Saga University
Saga, Japan
kake@is.saga-u.ac.jp

***Abstract*—In this paper, we analyze learning activity of students and its effect when we utilize the programming education support tool pgtracer to provide homework for their programming course. We assigned homework utilizing pgtracer to the students and incorporated its learning achievement to their evaluation of the course. As a result, this method improved the learning activity of the students. Their understanding of the trace table was related to understanding of the programming. Their answering processes to reach to a correct answer varied according to their programming skill. As for their understanding of the program, those who continuously did their homework understood a program better than those who only prepared for the exam in a short term. Therefore, we can conclude that providing homework utilizing pgtracer is effective for students' centered learning of programming.**

Keywords—Learning Analytics (LA), computer programing education; e-learning; Moodle; fill-in-the-blank question

I. INTRODUCTION

Computer programming is essential at national institute of technology and university majored in science and engineering. Recently Japanese government decided to start university entrance examination containing computer programming from 2025. The importance of programming education is increasing.

However, we often find students with low programing skill in an actual programming class. It is useful for a student to develop as many programs as possible in order to acquire practical programming skill. However the students cannot practice enough in an actual class, because there is a limitation of teaching staff and time. In order to compensate the lack of them, students need learning outside of the class. It is a usual case that the students tend not to learn themselves only by entrusting their willingness. The teacher cannot recognize the learning activity of the students outside of the class. When the teacher gives the student homework and some reports, their burden will become heavy to mark and to confirm submission status of each student.

We are developing a programming education support tool pgtracer working on the web [1-3]. Pgtracer works as a Moodle plug-in so that pgtracer provides an learning environment to a student at any time and place. Pgtracer provides fill-in-the-blank programming questions to the students. The teacher can define various types of blanks within the program and corresponding trace table. Furthermore the automatic scoring function of pgtracer reduces the teacher's burden to evaluate student answers. Pgtracer automatically collects learning activity data of the students and provides data analysis functions. The data analysis functions will also support teachers to recognize learning activity and achievement of each student and the entire class.

There are many support tools for programming education. [4] provides an environment using learning history and provides various analysis function for a teacher and a student. but it cannot trace the values of variables. [5] provide a fill-in-the blank question written in Java. But it does not provide log analysis functions and does not support trace table. [6] is a framework that guides incorporation of learning analytics mechanisms in computer programming education. However the detail of the collected data is not presented. [7] is a web-based Python programming environment which collects and analyzes learner's programming process. This environment also does not provide a function to estimate student's achievement level. There also are researches about learning analytics for programming education. [8, 9] help individual students by analyzing compile errors and feeding the result back to the students. On the other hand, pgtracer can also detect execution errors of the program.

We provide various programming homework using pgtracer to the students at a lecture called "Fundamental of Computer Science" since 2016 [10]. The lecture is provided for the second year students of the Kumamoto Institute of Technology. However the learning activity of the student was not enough in 2016 since the learning achievement was not reflected to the student's score. We also found many students who did not understand trace table.

In this paper, we address the above issues by incorporating the learning achievement outside of the class using pgtracer to each student evaluation. In order to encourage student's learning activities, we carry out quizzes which contain fill-in-the-blank questions assigned as past homework, and reflect them on the student's evaluation of the class. To promote

978-1-5386-7447-5/18/$31.00 ©2018 IEEE
DOI 10.1109/IIAI-AAI.2018.00057

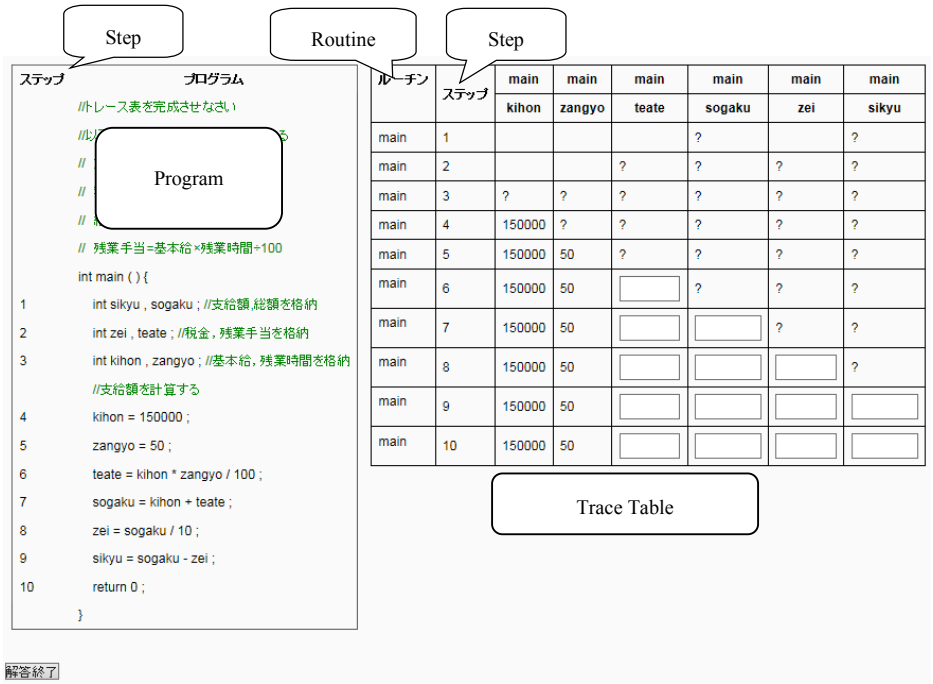

Fig. 1 Fill-in-the-Blank Question (2)-2 of pgtracer.

understanding of the trace table, we explain trace table in more detail and use the same trace tables to explain value changes of the variables in the lecture. We assign homework utilizing pgtracer every week. The homework corresponds to the lecture contents defined in the syllabus. We monitor the learning activity of the students using the data analysis function of pgtracer, and find the effectiveness of our strategy. As a result, we find that students' learning activity was improved by incorporating the learning achievement utilizing pgtracer. Students' understanding of the trace table was related to understanding of the programming.

This paper is organized as follows. The next section introduces fill-in-the-blank questions provided by pgtracer. We next explain the course planning in Section 3. In Section 4, we analyze the student's understanding of the program and trace table, learning activity and the relationship between student's intrinsic motivation for programming utilizing the collected data by pgtracer. The result are provided and discussed in the last section.

II. PGTRACER AND FILL-IN-THE-BLANK QUESTION

A fill-in-the-blank question of pgtracer is composed of a C++ program and a trace table as shown in Fig.1. The trace table represents execution order of steps with the routine name, values of each variable and output at each step. A teacher can define various types of blanks within the program and trace table [1]. Some examples of the blanks are a sequence of tokens or a statement within a program; step number and variable value within a trace table. A student fills the blanks so that the program and the trace table become consistent. The trace table is important for program comprehension and can help students when they get stuck during programming. Thus we expect that a trace table is an effective means for programming education especially for beginners.

When a student fills a blank, pgtracer automatically collects student log. A log record contains student id, question number, place of the blank, student answer, the evaluation result and the time at which the blank is filled. Pgtracer provides various types of data analysis functions to analyze the collected log [2]. A student can check their own learning activity, average and distribution of the entire class using the analysis function. A teacher can also check each student and question such as right answer ratio, required time and an answer process by utilizing the data analysis functions. Then the teacher can provide various feedback to the students.

The teacher can also define various options to the created questions. The options include question mode (self-learning mode or 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. By setting self-learning mode option to the question, students can know whether their answers are correct just after their filling of each blank.

III. EXPERIMENT PLAN

The purpose of this experiment is to clarify the learning behavior of the students when we evaluate students using the pgtracer exercise results, and to detect topics that the students understand insufficiently.

A. Course Planning

The experiment was performed for 130 students at the second academic year course named "Fundamental of

TABLE 1
LECTURE PLAN

Second Semester			
Week	Contents	# of Questions	Note
1	How to Execute Programming	-	User Registration of pgtracer
2	Constant, Variable, Assignment	2	Teach How to Use pgtracer
3	Output (printf), Operator	2	1st Questionnaire
4	Flowchart	3	
5	Input (scanf, getchar)	3	1st Quiz
6	Conditional Branch	3	Report1
7	Mid-Term Exam	-	
8	Conditional Branch	3	2nd Questionnaire
9	switch statement	2	
10	Nested Conditional Branch	3	
11	for Statement	3	2nd Quiz
12	while and do while statement	3	Report2
13	break, continue	3	
14	Exercise	2	
15	Examination	-	
16	Summary	-	3rd Questionnaire

Programming I" of Kumamoto Institute of Technology from October 2017 to February 2018. The students are majored in mechanical engineering, electronics, civil engineering, architecture, bio technology or chemistry so that they are not majored in computer science. It is the first course for the students to learn computer programming. One of the authors is giving lecture and exercise of the course.

Table 1 represents the lecture plan of the course and the number of questions included in the homework. The table also contains the schedule of reports, quizzes and questionnaires. With the experience of last year in mind, we reflected the score of quizzes and examinations which contain pgtracer questions on the evaluation of the student's achievement. Specifically, the ratio of evaluation contains 60% average of examination, 20% report and 20% quizzes.

We carried out two examinations using e-learning system introduced in our campus. These examinations contain two fill-in-the-blank questions assigned as homework. We also assigned two reports to the students. The students submit a program, an execution result and a flowchart. At the same time, the students must take an oral examination. A quiz was carried out using pgtracer. The quiz contains three fill-in-the-blank questions assigned as past homework.

We explain how to register to pgtracer and how to use pgtracer at the first two weeks to support the students who do not understand the concept of trace table. Furthermore, we use a similar trace table when we explained the transition of a variable value in order to decrease the gap between explanation in the lecture and a fill-in-the-blank question provided by pgtracer.

TABLE 2
INFORMATION OF QUESTION(2)-2

Description	Calculate and print a salary allowance.
Educational Objective	Students can trace assignment statements.
Difficulty Level	Easy
Place of Blank (# of blanks)	Within a trace table (5)

TABLE 3
QUESTIONS TO THE STUDENTS

(1)	Did you understand how to use pgtracer?
(2)	Did you understand a trace table?
(3)	How was the difficulty level of the questions?
(4)	How many blanks per a question?
(5)	Did you have intrinsic motivation to learn programming?
(6)	Average time to answer the question per week
(7)	Average time to work on other exercise per week
(8)	Was the exercise using pgtracer useful to learn computer programming?
(9)	Did you refer the programming style (indent and comment) of pgtracer to create your program?

B. Homework Preparation Policy

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

- Define the blanks at a value of variable, a part of statement and execution step to confirm that the students understand the lecture.
- Clarify the educational objective of each question.
- Try to decrease student's workload by setting the following guidelines for the question size to facilitate continuous use of pgtracer.
  - Two to three questions per lecture.
  - Three to five blanks per question.
- The question is represented in the self-learning mode. This is because, through our experience last year, we found that evaluation of student answer just after the student fills a blank increase the student's intrinsic motivation [10].
- We show the correct answer after the students submit the answer.

Fig. 1 illustrates Question (2)-2, and Table 2 represents the information of Question (2)-2. The number of blanks within a trace table is counted by the number of blanks having different values of the previous row.

The deadline of homework is the next lecture. There is no penalty when a student did not finish the homework. We monitored learning activity of students and difficulty of the questions.

C. Questionnaire to the Students

We performed a questionnaire three times to confirm the student's intrinsic motivation for programming and

TABLE 4
BLANKS WHICH RIGHT ANSWER RATIO AT FIRST TRIAL LESS THAN 80%

Question	Place of Blank Right answer is in [ ]	Right Answer Ratio (%)	Average number of trials	Typical Wrong Answers. The number in () means the number of answers. If it is blank that mean one answer.
(14)-1	If(j< [num-i]){ within a nested for statement	44.7%	5.8	No Answer (23), num (5), 5 (5), 4 (3), i (3), 3, num+i, num/10, sum-1, =num
(4)-3	[int hour;] within a variable definition statement	74.4%	4.3	No Answer (14), int hour (4), 1, 1 時間 30 分*1, 90, hour, hour=?, int, int<hour>, printf, sum hour
(11)-3	pow = [pow * 2]; within a for satement	76.3%	5.0	No Answer (16), 1024 (3), 10, 2*n, 32, 4, j++, n*2, n^2, pow*j
(12)-3	}while([cnt_pos < 5]); within a do-while statement	76.7%	4.2	No Answer (5), cnt_pos==5 (3), cnt_pos<6 (2), "%d".cnt_neg, break, cnt_pos, cnt_pos>5, cnt_pos>=5, cnt_pos<=5, num<0, num<5, num<8, pos==5
(14)-1	printf([“\n”]); with in a nested for statement	76.8%	3.5	No Answer (4), "" (3), \n (3), ", "", "%d", ":", %, *, 改行*2

*1 means 1 hour and 30 minutes, *2 means newline

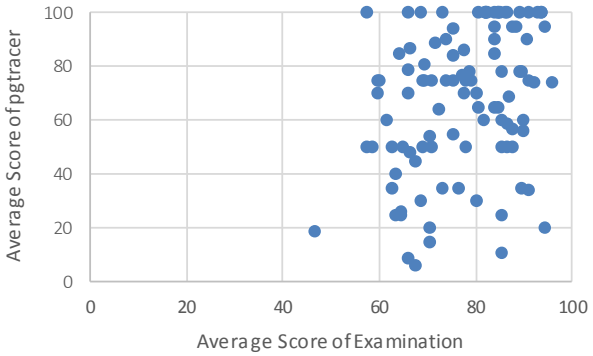

Fig.2 Relationship between the Averages of pgtracer Score and Examination Score

understanding level for a trace table. Table 3 represents the questions to the students. We asked the questions (7) to (9) only in the third questionnaire.

IV. ANALYSIS OF STUDENTS ACTIVITY

A. *Understanding of Computer Programming*

By analyzing the student's log, we find the following distributions. The answer ratio of each question is in the range of 80.0-99.2% so that most of the students solve the questions. The average of the right answer ratio at the first trial distributes within 63.4-92.2%. The median of the required time distributes in the range of 63.5-436.5 seconds. The average scores of most questions exceed 80%. This indicates that the questions were easy for the students.

The questions with average less than 70% are the questions (3)-2, (4)-3, (12)-3, and (14)-2. These questions have blanks within the program. Question (10)-1 contains blanks in the trace table corresponding to a for-statement. Question (11)-1 includes a nested for-statement and corresponding blanks in the trace table. We find that the students need longer time to fill the blanks within the trace table of the iterative statement.

Fig. 2 illustrates relationship between the average of the examinations score and the average score of the four questions whose score averages are less than 70%. Here we consider the students who completely answer the four questions

The correlation coefficient between these two is 0.32, thus there is a weak correlation between them. The correlation coefficient between the average of the examination score and the average of required time of the four questions is 0.07, so that there is no correlation between them. By observing the free description in the third questionnaire, we found some students who learn from his friend to solve the questions. Then a student with low programming skill can obtain a high score. We found many empty answers which the date of the first trial was after the deadline. They just view the questions to prepare the examination.

On the other hand, we find that the correlation coefficient between the average of the overall required time and the average of examinations is -0.13. The students who have high programming achievement answered the questions more quickly. However, as the question becomes difficult, the students who obtained a high score require longer time. Actually the questions with positive correlation coefficient are (6)-1, (10)-3 and (14)-1 except the four questions explained before. The average score of these questions are less than 80% except for the case of (6)-1.

Table 4 represents the blanks whose right answer ratios are less than 80%. All the blanks are within an iterative statement except for Question (4)-3. The right answer ratio of a blank in Question (14)-1 is the lowest. The corresponding blank is defined in a nested for-statement. From these facts, we can conclude that filling the blank within an iterative statement is difficult for the students.

By observing the answer processes of the Question (14)-1 in detail, we found characteristic answer processes below.

- Student 1 has high ability of computer programming, because the average of his examination is 94.5%. After he repeated filling the blank using variable num and i, he finally filled the right answer.
- Student 2 finally filled the right answer, but he filled some constants in the blank at first. He noticed to use two variables, num and i, after he considered long time at the 28th step. The average of his examinations is 80.5%.

- Student 3 filled some constants in the blank at first, then he noticed to use variable num. However, he did not notice to use variable i, thus he could not fill the right answer. The average of his examinations is 72.5%.
- Student 4 has low ability of programming, because the average of his examinations is 66.0%. He repeated filling completely wrong answers in the blank, because he could not understand the program used in the question.

By the above consideration, we found that there is the difference among student's answer processes to lead the right answer and the difference is caused by their achievements of programming.

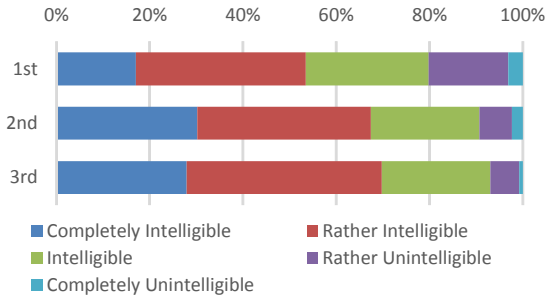

Fig. 3 Understanding of Trace Table

B. *Understanding of a Trace Table*

Fig. 3 illustrates the answer of question (1). In the first questionnaire, 79.8% of the students answered that they understand the concept of trace table. The percentage increases to 93.0% in the third questionnaire. We explained the trace table concept in the lecture, and used a similar table for tracing the change of a variable value. We consider that these helped students to understand the concept of trace table.

Table 5 represents cross tabulation of the answers of the question (1) and the average of the examination for each answer. The correlation coefficient for the mid-term examination is 0.20 and that for the final examination is 0.27. The readers can observe a weak correlation between the understanding of trace table and the examination score. A higher average score at the mid-term examination for the students who answer "Completely unintelligible" is the only exception. This is because a student answer "Completely unintelligible" and scored 92% at mid-term examination. The student did not resolve the fill-in-the-questions at all. The average score for this case becomes 67.0% except the student.

TABLE 5
UNSERSTANDING OF TRACE TABLE AND THE EXAMINATION SCORE

	Mid-Term Examination		Final Examination	
	#of students	Average score	#of students	Average score
Completely Intelligible	39	80.3	36	75.6
Rather Intelligible	48	78.3	54	76.0
Intelligible	30	79.1	30	68.5
Rather Unintelligible	9	67.0	8	64.5
Completely Unintelligible	3	76.0	1	51.0

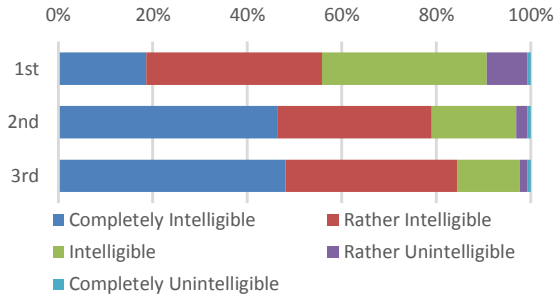

Fig. 4 Understanding How to Use pgtracer

C. *Student Activity*

Fig. 4 illustrates the answer of question (2). More than 90.0% of the students answered that they understand how to use pgtracer. We explained the purpose of pgtracer and the user registration at the first two lectures for 30 and 20 minutes respectively. We improved the explanation considering our experience of the previous year.

More than 80% of the students finished the homework before the deadline until 9th week. Student intrinsic motivation is kept high until the mid-term examination. We also provided time to work on the homework during a lecture.

On the other hand, the ratio of the students who finished homework in 10th week to 12th week within the deadline decrease less than 60%. We presume three reasons for this. The student got the winter vacation between 10th and 11th weeks. We gave an assignment to the students at 6th week, so some students gave higher priority to the assignment than the homework. We announced the student's activity and urged to do homework at 8th week, but we did not do this after that.

Fig. 5 illustrates the total number of answered questions for each week. The date represented in the horizontal axis are the checked date of the student activity. Fig. 5 also illustrates the date performed the examination, quizzes and questionnaires. Here, three date mean there are three target classes. Observing Fig. 5, we find that the total number of answers increase in the

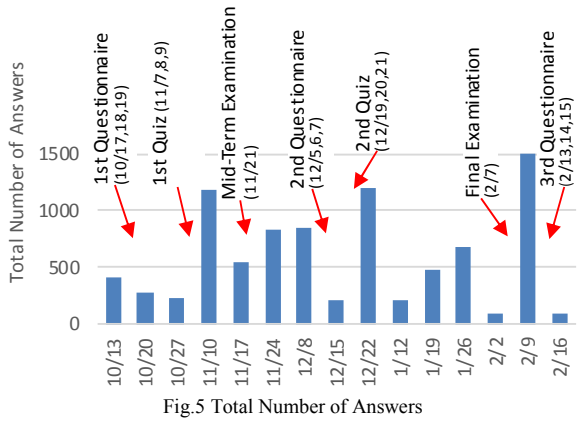

Fig.5 Total Number of Answers

TABLE 6
INTERISTIC MOTIVATION TO PROGRAMMING AT EACH EXAMINATION

Answer of Question(5)	Mid-Term examination			Final Examination		
	2nd Questionnaire (# of students)	Average of Examination Score	Average of Answers Rate within the Deadline	3rd Questionnaire (# of students)	Average of Examination Score	Average of Answers Rate within the Deadline
Very Strong	36	79.9	84.0%	28	76.2	73.0%
Rather Strong	63	79.8	92.1%	60	75.3	77.1%
Neutral	25	73.5	82.8%	28	70.9	58.0%
Rather Weak	3	76.0	89.7%	7	60.1	53.1%
Very Weak	2	64.0	42.3%	6	65.2	67.2%

periods which contain two quizzes and two examinations. We find that the students prepared for the quizzes and examinations. We also observe that the total number of the answers is small before the quizzes and the examinations. We consider that the students intended to solve the homework just before the quizzes and the examinations not just after the lecture.

All of the questions have more than 80% of the answer ratio, and the average answer ratio is 92.3%. The answer ratio was much improved compared to our experiment in 2016. The reason is clear because the provided questions did not affect evaluation of the students.

Only 9 students (6.9%) stopped answering within 10 seconds at more than six questions. Thus we can conclude that most of the students answered the questions seriously.

Fig. 6 illustrates the result of question (3). In all of the questionnaires, more than 50% of the students answered "Reasonable". Observing the number of the blanks per question, more than 70% of the students also answered "Reasonable". 81.3% of the students answered more than 80% of the questions. 83.0% of the students finished homework for one lecture within 30 minutes. From the above reasons, we can conclude that the difficulty level and the number of blanks are reasonable to the student.

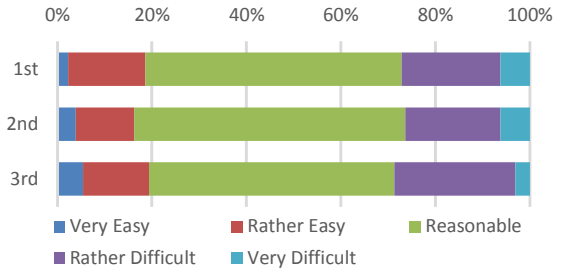

Fig.6 Difficulty Level of the Questions

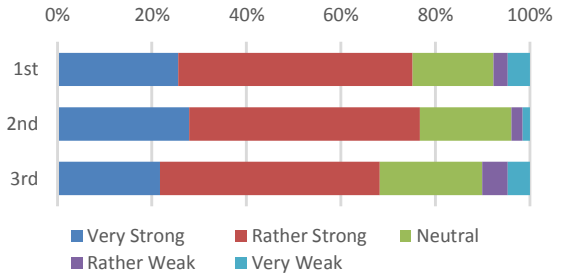

Fig.7 Interest and Intrinsic Motivation of the Students

Next, we examine that the number of students who finished homework within the deadline decreased after the mid-term examination. We consider that this is caused by the difference of scores between the mid-term and the final examinations.

The overall average between the two examination scores decreases 5.3%. The average decrease about the students (Group 1) who finished homework within the deadline after the mid-term examination is 4.3%, while the average decrease about other students (Group 2) is 7.2%. The students in Group 2 had answered the questions to prepare for the examinations and quizzes. We can thus conclude that continuous activity improves achievement level of computer programming than intensive activity.

D. Intrinsic motivation of the student

Fig. 7 represents the answers to question (5). The 75.2% of the students answer "Very strong" or "Rather Strong" in the first questionnaire. In the second questionnaire, the ratio slightly increased to 76.7%, and deceased to 68.2% in the final questionnaire. We can presume that the students were interested in computer programming. But the percentage decreases as the lectures becomes difficult for the students.

Table 6 represents cross tabulation between answer of question (5) and the statistics at the two examinations. We observe a weak correlation coefficient between the student's answer and each examination score.

TABLE 7
THE NUMBER OF STUDENT ANSWERED LESS THAN "RATHER WEAK" IN 3RD QUESTIONNAIRE AND ANSWERING ACTIVITY OF THEM

Answer of question (5) in the 3rd Questionnaire	Very Weak		Rather week	
Questionnaire	First	Second	First	Second
Very Strong	0	1	0	0
Rather Strong	1	2	2	2
Neutral	2	2	4	4
Rather Weak	0	0	1	1
Very Weak	3	1	0	0
Answer rate within the deadline	67.2%		53.1%	
Average of answered questions	68.0		65.4	

Considering about mid-term examination, the average score of the students who answer "Very Strong" is the highest (79.9%). The lowest one is the average score of the students who answer "Very Weak" (64.0%). The difference of these scores is 25.9 points. Observing the answer rate within the deadline, the rate of the students who answer "Very Weak" is the lowest. We can thus consider that the intrinsic motivation to the computer programming influence on the examination score.

However, the average score of the students who answer "Rather Weak", is the lowest in case of the final examination. We examine the students who answer "Rather Weak" or "Very Weak" in the third questionnaire to analyze the reason of the difference. Table 7 represents the result of the first and the second questionnaires about them.

The half of them answer "Very Weak" in the first questionnaire too. Although their intrinsic motivation increased temporary, we consider that their intrinsic motivation was low through the second semester. On the other hand, the students who answer "Very Weak", except for one student, answered more than "Neutral" in the first questionnaire. We can presume that their intrinsic motivation decreased more quickly during the semester.

Observing the average number of questions solved before the deadline, the number of students who answer "Very Weak" is larger than the number of students who answer "Rather Weak". Therefore, the students who answer "Very Weak" answered seriously, but they have a sense of difficulty. Therefore, the students who answer "Very Weak" in the final questionnaire have recognized that they are not good at computer programming from an early phase, but they did homework seriously. On the other hand, the students who answer "Rather Weak" in final questionnaire gradually stopped to work on the homework because of the declining intrinsic motivation. Thus, we can presume that their scores of the final examination fell. Therefore, we find that the decline of examination score is larger the student who gradually lose his intrinsic motivation than the student whose intrinsic motivation is low from an early phase in the semester.

V. Conclusion and Future Work

In this paper, we prepared questions for self-learning utilizing pgtracer and observed the learning activity of the students and its effect. As a result, we found the followings.

The students who continuously worked on their homework understood computer programming better than those who prepared for the exam in short term. Iterative statements were difficult for the students. In addition, the understanding of the program becomes better for the students understanding trace table better. Furthermore, the student's answering process to lead a correct answer varied according to each programming skill of them.

By examining the learning activity of the student, it was revealed that pgtracer could support self-learning of the student. However, we need plans to keep student intrinsic motivation. In addition, it is important to detect the students losing interest and willingness from the middle of the semester since their drop tend to be bigger than the students with low intrinsic motivation from the beginning.

As a future work, we are planning to analyze the different of student's processes to lead to a correct answer depending on their understanding level. We shall also provide feedback considering understanding level of each student. Such feedback will be useful for self-learning. Then, it will be necessary to provide retention function so that a student can learn continuously. We also have a plan to extend pgtracer to automatically generate questions of a desired difficulty level by setting places of blanks and show/hide of the comment. In the current lecture plan, we utilize pgtracer in one direction such as a homework assignment reflecting the content of the lecture. We are planning to design interactive lecture activity, such as explaining problems with a low answer rate in classes or assigning the same problems again.

Acknowledgment

We appreciate for the cooperation of the students who joined the evaluation experiment using pgtracer. 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, K. Ohta, "Student log analysis functions for web-based programming education support tool pgtracer", 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS2015), pp. 120-128, 2015.

[3] M. Murata, T. Kakeshita, "Analysis method of student achievement level utilizing web-based programming education support tool pgtracer", 5th International Conference on Learning Technologies and Learning Environment (LTLE 2016), Kumamoto, Japan , pp. 316-321, July 2016.

[4] Ohashi, et al., "A Programming Learning Base System for Support Tools Utilizing Learning History", IEICE technical report: 133(316), pp.15-20, 2013. (n Japanese)

[5] N. Funabiki, et al., "Analysis of fill-in-the-blank problem solutions and extensions of blank elent selection algorithm for Java programming learning assistant system", World Congress on Engineering and Computer Science (WCECS 2016), San Francisco, USA, pp. 237-242, Oct. 2016.

[6] C. Malliarakis, M. Satratzemi, S. Xinogalos, "Intergrating learning analytics in an educational MMORPG for computer programming", IEEE 14th Int. Conf. of Advanced Learning Technologies, pp. 233–237, 2014.

[7] J. Helminen, P. Ihantola and V. Karavirta, "Recording and analyzing in-browser programming sessions", 13th Koli Calling Int. Conf. on Computing Education Research, pp. 13-22, 2013.

[8] X. Fu, et al., "Error Log Analysis in C Programming Language Courses", The 23rd International Conference on Computers in Education (ICCE2015), pp.641-650, 2015.

[9] X. Fu, et al., "Error Log Analysis for Improving Educational Materials in C Programming Language Courses", The 2nd ICCE workshop on Learning Analytics (LA2015), pp.412-417, 2015.

[10] T. Kakeshita, M. Murata, "Application of Programming Education Support Tool pgtracer for Homework Assignment", *International Journal of Learning Technologies and Learning Environments*, 2018 (in Press).