International Journal of Learning Technologies and Learning Environments
International Institute of Applied Informatics
2022, Vol. 5, No. 1, IJLTLE565

Analysis of Student's Learning Log Data in Fill-in-the-Blank Programming Questions

Tetsuro Kakeshita [*], Miyuki Murata [†],
Naoko Kato [‡], Youhei Nakayama [§]

Abstract

We have developed a programming education support tool pgtracer which provides fill-in-the-blank questions containing a C++ program and a trace table. In this paper, we analyze the study log and the answer log collected by pgtracer. We analyze student activities and incorrect answers to find the tendency and frequent mistakes of the students. We next classify the type of incorrect answers in the log data for 18 fill-in-the-blank questions with 127 blanks. We then identify the patterns of frequently observed errors using association analysis. Furthermore, we analyze the answering process to fill the blanks of the students and find that the right answer ratio affects the answering process. We expect that these analysis techniques and the results help to improve programming education through feedback to the class and the teacher.

Keywords: Computer programming education, e-learning, fill-in-the-blank question, Learning Analytics (LA), Moodle

1 Introduction

Computer programming is essential at national institute of technology and university majored in science and engineering. However, we often find students with low programming skills in an actual programming class. It is useful for a student to develop as many programs as possible to acquire practical programming skills. However, the students cannot practice enough in an actual class due to a limitation of teaching staff and time.

To cope with this problem, we are developing a programming education support tool pgtracer [1]. Pgtracer is developed as a Moodle plug-in so that a student can learn computer programming at any time and place as long as a personal computer and internet connection are provided. Pgtracer automatically evaluates student answer as soon as a student submit their answers. The automatic scoring function of the pgtracer reduces the teacher's workload and can provide a quick

[*] Computing Division, Saga University, Saga, Japan
[†] Faculty of Liberal Studies, National Institute of Technology, Kumamoto College, Kumamoto, Japan
[‡] Faculty of Liberal Studies, National Institute of Technology, Ariake College, Fukuoka, Japan
[§] Graduate School of Information Science, Saga University, Saga, Japan

response to the students. Pgtracer also provides data analysis functions from various viewpoints [2]. The functions also support teachers to recognize the learning activity and achievement of each student and the entire class.

We provided various programming homework using pgtracer to the student at our programming courses [3, 4]. The courses are provided for the first and second-year students of the Institute of Technology, Kumamoto College. We have found the following facts through our experience. (1) The students who continuously worked on their homework understood computer programming better than those who prepared for the examination in a short time. (2) The understanding of the program becomes better for the students understanding the trace table better. (3) Student's programming skill affects their answering process to lead a correct answer.

In this paper, we analyze student activities and incorrect answers to find the tendency and frequent mistakes of the students using statistical analysis language R. We classify the types of incorrect answers based on possible reasons for the incorrect answers for 18 fill-in-the-blank questions with 127 blanks. We next identify frequently observed patterns of the errors using association analysis. Furthermore, we analyze the answering processes to fill the blanks of the students and find that the right answer ratio affects the answering process.

Various research contributions are willing to support computer programming education. We shall introduce representative related researches in Section 2 and explain the difference from our research. Section 3 introduces pgtracer functions. We next explain the outline of the experiment in Section 4. In Section 5, we shall analyze the problems using the collected data. In Section 6, we shall classify incorrect answers and identify frequently observed patterns of the errors using association analysis. In Section 7, we shall analyze the answering process. The result and future work are discussed in the last section.

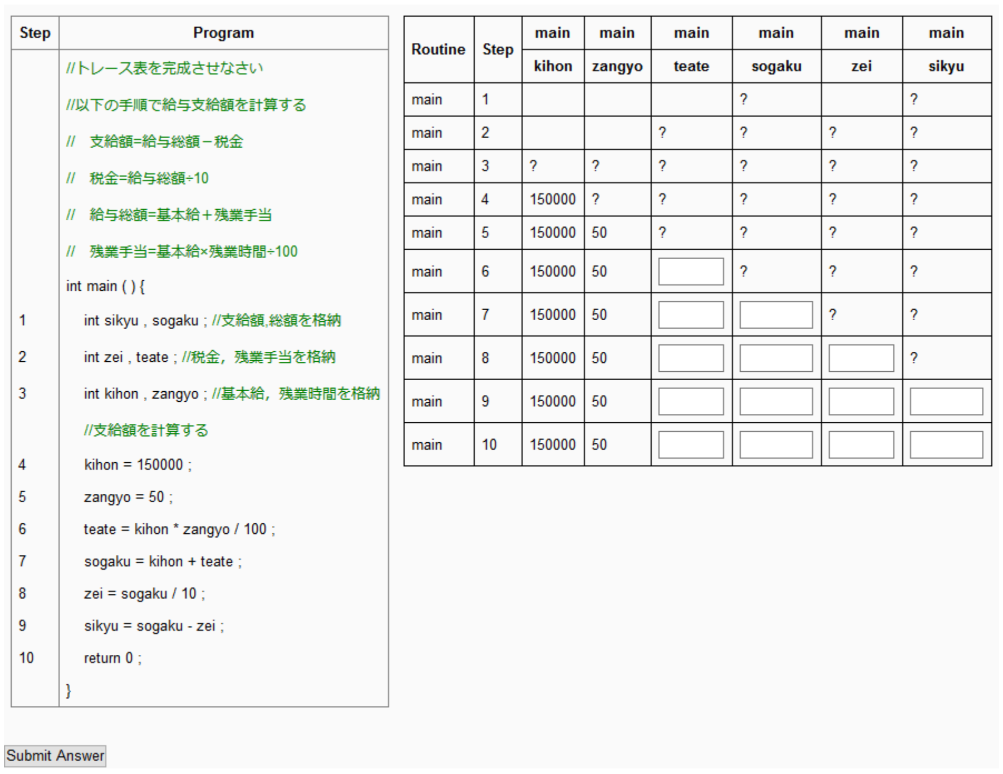
2 Related Works

Fu et al. analyze the error log of the programming and access log of the educational contents [5]. To support students studying utilizing on-line learning, Hering et al. and Gotthardt et al. analyze learning activities using SAS Business Analytics, and constructs a learning environment that provides valid educational content such as text and video [6, 7]. Malliarakis proposes a framework to guide the introduction of a learning analytics mechanism using game-based learning analytics [8]. Ishiwada extracts frequent edit patterns of the students utilizing sequential pattern mining [9]. It automatically estimates the learning progress by analyzing the result and the learning activity. Igaki proposes the individual guidance support system called C3PV [10]. It stores the coding process and visualizes the process.

Pgtracer automatically collects learning log data such as student answer, score, required time to fill each blank, right answer ratio and required time to solve each question. Pgtracer also provides the data analysis functions to understand the learning activity of a student and an entire class, and to detect weak points of the students. The blanks can be defined at various program elements (e.g. single token, a sequence of tokens, expression, and statement) and trace table (e.g. a variable value, step number, and variable name). Thus the fill-in-the-blank questions provided by pgtracer have more flexibility and can express a wider level of difficulty than other existing programming education support tools.

3 Programming Education Support Tool Pgtracer Utilizing Fill-in-the-Blank Questions

Programming education support tool pgtracer provides a fill-in-the-blank question composed by program and trace table (Figure 1). A trace table represents the value of the variable and output at each step. A fill-in-the-blank question is defined by four XML files, each of which represents a program, a trace table, a mask for the program or a mask for the trace table. It is possible to define various types of blanks within a program or a trace table by utilizing a mask file.



Step	Program
	//トレース表を完成させなさい
	//以下の手順で給与支給額を計算する
	// 支給額=給与総額－税金
	// 税金=給与総額÷10
	// 給与総額=基本給＋残業手当
	// 残業手当=基本給×残業時間÷100
	int main ( ) {
1	int sikyu , sogaku ; //支給額,総額を格納
2	int zei , teate ; //税金，残業手当を格納
3	int kihon , zangyo ; //基本給，残業時間を格納
	//支給額を計算する
4	kihon = 150000 ;
5	zangyo = 50 ;
6	teate = kihon * zangyo / 100 ;
7	sogaku = kihon + teate ;
8	zei = sogaku / 10 ;
9	sikyu = sogaku - zei ;
10	return 0 ;
	}

Routine	Step	main	main	main	main	main	main
		kihon	zangyo	teate	sogaku	zei	sikyu
main	1				?		?
main	2			?	?	?	?
main	3	?	?	?	?	?	?
main	4	150000	?	?	?	?	?
main	5	150000	50	?	?	?	?
main	6	150000	50		?	?	?
main	7	150000	50			?	?
main	8	150000	50				?
main	9	150000	50				
main	10	150000	50				

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

Pgtracer provides three major functions to create a fill-in-the-blank question, functions to provide and evaluate a question, functions to collect and analyze a log.

1) Functions to Create a Fill-in-the-Blank Question [1]

A fill-in-the-blank question is composed of four XML files, representing either of a program, a trace table, masks for the program and the trace table. Pgtracer provides the functions which automatically create these XML files. A token, consecutive sequence of tokens, expression, and statement is the candidate of blanks within a program. Variable value, step number, and variable name are the candidate of blanks within a trace table. We have clarified that the difficulty level of a question can be controlled by the place of a blank [1, 2]. The questions composed of the four types of XML files are stored in the question database.

2) Functions for Presenting a Fill-in-the-Blank Question and Automatic Evaluation of Student Answer [1]

This function presents the list of fill-in-the-blank questions stored in the question database and presents the selected question by the student. The definition of the question contains various options. 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, the coloring of the corresponding step when a student selects a blank within a trace table. In the case of self-learning mode, the pgtracer evaluates the answer by comparing it with the correct answer just after filling of each blank. Here, if there are multiple correct answers, pgtracer cannot evaluate the student answer immediately. After the answer is submitted to pgtracer, pgtracer compiles and executes the filled program, and then compares the result with the correct trace table. Therefore pgtracer can evaluate the answer correctly even if a question has multiple right answers.

3) Functions to Collect Answer and Analyze Log [2]

Pgtracer collects a student answer, correct answer, required time just after filling to each blank. Pgtracer then provides the learning history function, the analysis function of each student, the detailed analysis function of each problem, the detailed analysis function of each blank, the detailed analysis function of the learning process of the select student. These analysis functions are visualizing and/or aggregating the collected log data so that advanced analysis methods proposed in Sections 5 to 7 are not included.

Utilizing these functions, the students can check their learning activity and the average and distribution of whole users. And a teacher can recognize the right answer ratio and required time each student or each question using a table or a graph. The teacher can utilize the analysis result to improve the teaching method and fill-in-the-blank questions for effective learning.

4 Experiment Plan

The experiment was performed for three classes in the first academic year of the Institute of Technology, Kumamoto College in 2017 and 2018 academic years. The students are learning computer programming at a common course named “Fundamentals of Programming I”. And they do not major in computer science. The number of students is 130 in 2017 and 123 in 2018 respectively. Although we utilize the analysis results of both academic years, student profiles are the same so that the entire story of this paper is kept consistent.

We assigned two or three fill-in-the-blank questions to the students as homework each week. These questions are selected from the lecture contents of the corresponding week. The deadline for homework is the next lecture. Although there is no penalty when a student did not finish the homework, we utilize the questions at the mid-term and final examinations so that many of the students solved the questions.

We carried out two quizzes using pgtracer in the fifth and twelfth lectures at two classes (88 students). The quizzes contain three questions assigned as homework. The quizzes are executed in the examination mode and do not show the correct answer after the students submit their answers. We allowed the students to repeatedly solve each question.

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

- Define blanks at a value of a variable, a part of a statement and execution step to confirm that the students understand the lecture.
- Clarify the educational objective of each question.
- Try to control student's workload by preparing 2-3 questions per lecture and 3-5 blanks per question to facilitate continued use of pgtracer.

We prepared 34 questions through the 13 lectures. All of the questions set the same option. The questions are set in the self-learning mode. This is because, through our experience in the past, we found that evaluation of student answer just after the student fills a blank increase the student's intrinsic motivation. The analysis function is shown, and the coloring of the corresponding step when a student selects a blank within a trace table. The collect answer shows after the student submitted. For example, Table 1 represents information of question (2)-2 (Figure 1) in the second week of the 2018 academic year.

Table 1: Information of Question (2)-2

Description	Calculate and print a salary allowance.
Educational Objective	Can trace assignment statements. Aware of the order of assignment.
Difficulty Level	Easy
Place of Blank (# of blanks)	Within a trace table (4) Note: Only the blanks whose values are changed from the previous column are counted.

5 Analysis of the Problems

Table 2 represents the answer ratio, the average score and the required time at the first attempt. The place of blanks represents the type of the fill-in-the-blank question where P and T respectively represents blanks within program and trace table. The average score is calculated excluding unanswered questions. Most of the questions have an average score of higher than 70%. Thus the questions are relatively easy for the students. We analyze the questions whose average score is less than 70%. Questions (3)-2, (4)-3, (13)-3, (15)-1 and (15)-2 are provided as a homework, and Questions (4)-1, (4)-3, (10)-2 are provided as quizzes. All of these questions contain blanks within programs. Most of the questions, except (4)-3, are considered to be difficult for the student.

There is a negative correlation of -0.59 between the average score and the average of the required time. Thus we consider that the student who marks a higher score can answer in a short time.

Table 2: Profiles of the Questions

	Question	Place of Blanks	Answer Ratio (%)	Average Score (%)	Required Time (sec)
Homework (Carried out using pgtracer)	(2)-1	T	94.7	95.0	190.7
	(2)-2	T	93.9	93.0	245.3
	(3)-1	T	92.4	89.7	297.8
	(3)-2	P	90.2	42.2	509.2
	(4)-1	T	89.4	80.1	295.9
	(4)-2	P	88.6	87.2	236.0

	Question	Place of Blanks	Answer Ratio (%)	Average Score (%)	Required Time (sec)
	(4)-3	P	87.1	51.0	327.9
	(5)-1	T	87.1	77.4	227.0
	(5)-2	P	84.8	70.5	364.4
	(5)-3	P	84.8	70.0	338.9
	(6)-1	T	84.1	94.6	123.3
	(6)-2	P	85.6	88.1	139.5
	(6)-3	P, T	84.8	86.1	110.8
	(7)-1	T, P	85.6	95.8	134.8
	(7)-2	P	85.6	76.3	289.0
	(9)-1	P	90.9	90.7	184.9
	(9)-2	T	88.6	89.6	288.8
	(9)-3	T, P	88.6	81.7	230.1
	(10)-1	T	87.1	93.2	111.8
	(10)-2	P	85.6	74.7	167.6
	(11)-1	T	84.8	73.1	877.5
	(11)-2	P	81.8	83.2	211.1
	(11)-3	P	81.1	73.5	297.8
	(12)-1	T	81.8	81.6	389.2
	(12)-2	P	81.1	80.6	255.7
	(12)-3	P	80.3	78.3	304.2
	(13)-1	T	79.5	97.0	146.7
	(13)-2	P	78.0	70.4	306.8
	(13)-3	P	78.8	64.2	325.8
	(14)-1	T	76.5	91.7	209.1
	(14)-2	T	74.2	97.5	133.5
	(14)-3	P	73.5	78.6	313.3
	(15)-1	P	64.4	69.7	281.4
	(15)-2	P, T	62.9	64.0	410.0
Quiz (Carried out using online test using WebClass)	(3)-1	T	100.0	79.7	135.1
	(4)-1	T	98.9	50.6	206.0
	(4)-3	P	93.2	25.6	159.9
	(9)-1	P	100.0	87.1	118.5
	(10)-2	P	100.0	54.8	110.3
	(11)-2	P	100.0	73.9	94.1

2018 Academic Year

6 Analysis of Incorrect Answer

6.1 Preliminary Analysis

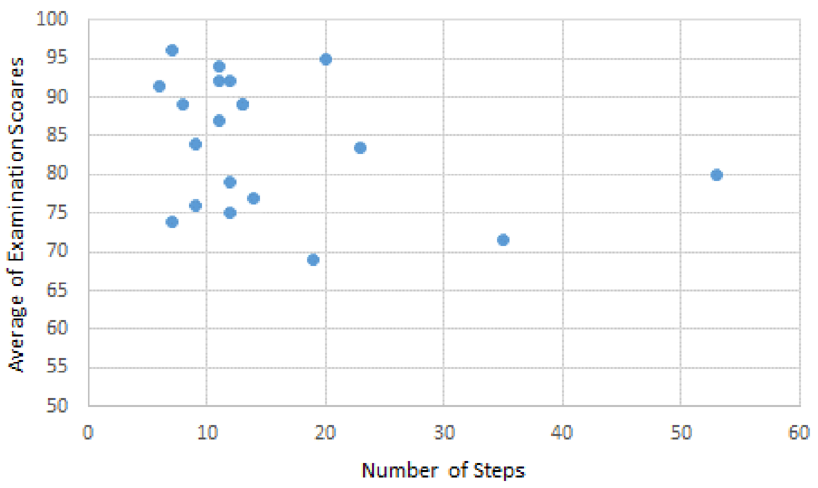
Table 3 represents the blanks whose right answer ratio is less than 70%. The blanks contain in the question are provided as homework. All blanks are defined within a program. No. 1, 4, 6, 7, 8 are defined within an iterative statement. Blank No. 1 with the lowest score is defined within a nested for-statement. Therefore, we found that a blank contained within the iterative statement is difficult for the students.

For the cases of No. 2 and 3, we consider that the students could not describe printf and getchar standard function calls. As for No. 5, it was the first time to fill in the variable definition part, so that students might be confused about what to answer.

Table 3: Blanks whose Right Answer Ratio at First Trial is less than 70%

No.	Place of Blank and Right answer	Right Answer Ratio (%)	Average # of Trials	Typical Wrong Answers.
1	if(j< num - i){ within a for statement	37.3	5.3	No answer (25), num (5), i(2), 0, 4, 6, i++, num+i
2	printf("Class %c\n");	54.6	7.7	No answer (8), "%c" (2), "Class%c" (2), Class%c , class E, "class %s", etc.
3	ch=getchar();	60.0	4.0	No answer (6), 65 (6), "Kuma-moto", getcher, str, int, %d, etc.
4	}while(cnt_pos < 5); within a do-while statement	60.0	4.7	cnt_pos>5 (6), No answer (5), cnt_pos==5 (2), cnt_pos<6, "%d int, cnt_neg==5, cnt_pos, cut_pos>5, etc.
5	int hour; within a variable definition statement	65.2	3.6	No answer (12), hour (5), 1(4), 1.5 9/hour, double hour; int hour, scanf("%d","hour"), int, inthoure;, inthour, scanf("%lf",&mail)
6	cntPos ++; within a if statement	65.7	3.7	No answer (8), cntPos=num/nu m (2), and=ans+i, cntNeg=4, cnt Pos+1, cntPos+=cntPos, cntPos +i, cntPos= etc..
7	printf("\n"); with in a nested for statement	66.1	3.5	No answer (8), "" (3), "*" (2), "%d", i, "|n", *, &num, ans=5
8	pow = pow * 2; within a for statement	68.5	4.0	No answer (11), n^2 (3), 2, 2*n^, 22, 32, 4, i*j, j*i, n*j, pow*j

2018 Academic Year



2018 Academic Year

Figure 2: Distribution between the Average Exam Score and Number of Steps during Answering Process (Score = 100%)

We analyze blank No. 1 whose right answer ratio is the lowest in Table 3 in detail. Figure 2 represents the distribution between the number of steps needed to answer and the average of the examinations for the students who marked a 100% score at question (15)-2. Here, the blank in question (15)-2 are defined within a program, and the number of blanks in the question is five. The number of steps needed to answer the question represents the number of blanks that the students filled during the answering process of the question. Pgtracer counts the number by incrementing a counter when a student selects a blank after filling a blank. Since the number increases if a student rewrites a blank after filling the blank, the number represents the amount of trials and errors of the student. As shown in Figure 2, most students answered with less than 20 steps, but the correlation coefficient is -0.31 between the number of steps required and the average of the examinations, which has a weak negative correlation.

Table 4 represents incorrect student answers entered during the answering process. We can observe from this table that the students reach the correct answer after trials and errors of various incorrect answers.

Table 4: # of Students vs. Incorrect Answer during Answering at Blank No. in Table 3

Incorrect Answer	# of Students	# of Student Trials
Expression using i and num (i+num, i*num, etc.)	2	4
Expression using num (num, num-1, etc.)	14	32
Expression using i (i, i++, i-j, etc.)	8	16
Other expression (j, *5, 5-j etc.)	3	4
Constant （0, 2, 4, 5 etc.）	12	42
int	1	1
Invalid string (j]\)	1	1

2018 Academic Year

We consider the answering process for the two answers that had a large number of steps, such as 53 and 35 in Figure 2. The students input the expression using variables num and i first, but thereafter, with a constant interval from 0 to 10 at short intervals. After that, the students thought for a while and entered an expression using variables num and i, and finally got to the correct answer. The difference in the number of steps in the two answers depends on the number of times the students enter the constant.

From the above, we can observe that the students use num or i, the students try to input constant before they notice that num and i are used in combination. Also, after noticing that both num and i are used, it can be seen that the correct answer (num-i) can be easily derived.

6.2 Classification of the Answers

Our research group considers that analyzing the wrong answer can measure the search domain of the student, that is, the degree of understanding of the program. Under this assumption, we classify the wrong answers into the following five cases.

(A) No answer

(B) An error in a blank within a program which is caused by student not understanding the instruction, i.e. the student does not understand what to describe. Such instruction is commonly described in the comments at the top of the program.

(C) An error in a blank within a program which is caused by student not understanding how to describe the requested code using C. The student correctly understands the instruction in this case.

(D) An error in a blank within a trace table. The student does not correctly understand the trace table in this case.

(E) Correct answer including unintended one.

We consider the case (A), no answer means that the student does not understand the instruction so that the student has no idea to lead to the correct answer. It also means that the student does not answer the question because he/she just wanted to look at the question.

In the program of the fill-in-the-blanks problem, we have explained the idea of the algorithm or the concrete algorithm using comment within the program. The case (B) is the case where it was judged based on the comment that the specific process to be performed by drilling the program could not be reproduced. It can be further classified according to whether the comment instruction is specific or not.

On the other hand, the case (C) is a case where it was judged that although concrete processing was understood based on the comment, it was not understood how to express it in C language. The case (C) is further classified according to the type of error, and the distribution and the reason why the solution is reached are examined. The case (D) is defined to classify incorrect answers within a trace table.

Although most of the answers in case (E) are correct answers, there are few unintended answers. Typical examples of the latter cases are the answers using literal values, such as 10, instead of using constant such as N.

We further classify the 9659 answers as represented in Table 5 by analyzing the answers collected from the students in the 2017 academic year. The readers can observe the number of cases whose values include small numbers, such as 72.5, in several cases in Table 5. This is because the number of cases is counted so that each answer is counted only once even if the cases where multiple errors are contained within an answer. We estimated the weight of each error so that the sum of the weights always becomes equal to 1.

The readers can also observe level 3 categories such as C1.1 in Table 5. These categories are the subcases of the corresponding level 2 category such as C1. The number of cases of the level 3 categories is, however, excluded from the cases of the corresponding level 2 category. Thus the number of cases of C1 category does not include the number of cases of C1.1 and C1.2.

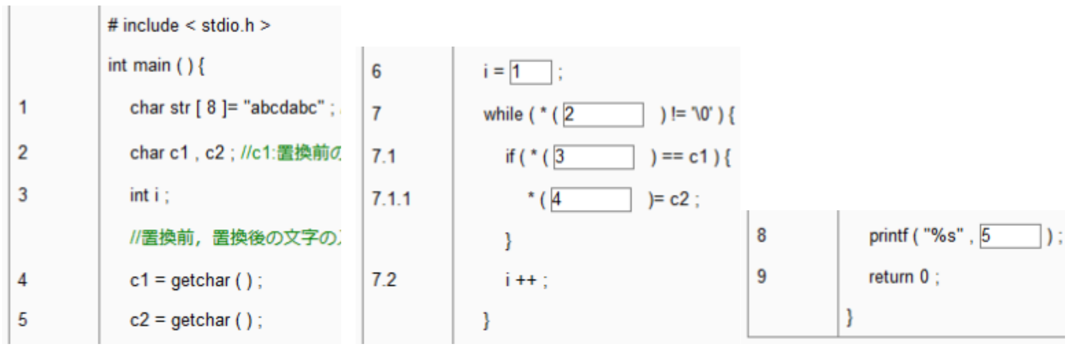
Table 5: Detailed Classification of the Answers

Category		# of Cases	Description
Level 1	Level 2		
A		389	Blank Answer (Students do not understand instruction, or they do not intend to answer)

Category		# of Cases	Description
Level 1	Level 2		
B			Students do not understand instruction provided using comments within a program.
	B1	25	The comments contain specific instructions.
	B2	89	The comments are rather abstract so that students need to understand specific instructions.
C			Although students understand the instruction, they failed to write the code correctly.
	C1	72.5	Students do not understand the notion of pointer or how to describe pointer operation.
	C1.1	30.5	Students do not distinguish between address and pointer.
	C1.2	13.5	Students do not understand how to access a particular data item using a pointer.
	C2	24.5	Incorrect use of an array (For example, students use array at the place where a pointer is suitable)
	C3	56	Incorrect function definition and function call
	C4	9	Incorrect use of formal and/or actual parameter
	C5	1	Errors related to simple or nested for loop
	C6	27	Errors related to the scope of a variable
	C7	14	Errors related to variable definition
	C8	43	Misspelling, incorrect calculation and/or misunderstanding
	C9	4	Students do not distinguish expressions with/without side effect
D			An error in a blank within a trace table. The student does not correctly understand the trace table in this case.
	D1	13	Students do not understand the value update caused by the increment/decrement operation.
	D2	73.5	Students do not understand the notion of pointer or the result of a pointer operation.
	D2.1	13	Students do not correctly understand the address referenced by a pointer.
	D3	83	Students do not correctly understand the behavior of function calls.
	D4	20	Students do not correctly understand the specification of a routine.
	D5	25	Students do not correctly understand what to fill at each blank in a trace table.
	D6	12	Errors related to parameters of a function
E			Correct answer including unintended one
	E1	8605	Correct answer
	E2	16	Unintended correct answer

2017 Academic Year

Figure 3 represents a problem to demonstrate answer classification. The numbers in the left column are the step numbers for the trace table. This program replaces a character c1 to another one c2 within string str. Students should note that the array name str can be used as an address of the string. Such a note was described as a comment of the program to provide instruction to the students. However, the corresponding comment is omitted in Figure 3 since the comment is written in Japanese.



```
        # include < stdio.h >
        int main ( ) {
1           char str [ 8 ]= "abcdabc" ;
2           char c1 , c2 ; //c1:置換前の
3           int i ;
            //置換前，置換後の文字の
4           c1 = getchar ( ) ;
5           c2 = getchar ( ) ;

6           i = [1] ;
7           while ( * ( [2] ) != '\0' ) {
7.1            if ( * ( [3] ) == c1 ) {
7.1.1              * ( [4] )= c2 ;
               }
7.2            i ++ ;
            }

8           printf ( "%s" , [5] ) ;
9           return 0 ;
        }
```

Figure 3: Problem to Demonstrate Answer Classification.

Table 6 represents the student answers for blank No. 2 of the above problem. The correct answer is "str+i". However, only 32% of the answers are correct, which is the lowest among the blanks in this problem. All the answers of the blank No. 2 are classified based on the criteria listed in Table 5. The classification is represented in the category column of Table 6.

Table 6: Classification of Answers of Blank No. 2 of the Program in Figure 2.

Incorrect Answer	Reason of the Answers	# of Students	Category
"" (Blank)	Students did not understand instruction. Or they did not intend to answer the question.	40	A
char str	Multiple definitions of the variables with the same name. The student did not understand instruction.	1	B2
getchar	Incorrect function call. The student did not understand instruction.	1	B2
Str	Misspelling. Students did not understand how to access a particular data item using a pointer.	7	C1.2, C8
str++	Students did not understand how to access a particular data item using a pointer.	1	C1.2
str+1	Students did not understand how to access a particular data item using a pointer.	1	C1.2
str+i	Correct answer	29	E1
str[8]	Incorrect use of an array (violation of array subscript). The student did not notice "*" just before the blank.	1	C2
str[i-1]	Incorrect use of an array. The student did not notice "*" just before the blank.	1	C2
str[i]	Incorrect use of an array. The student did not notice "*" just before the blank.	6	C2

2017 Academic Year

6.3 Association Analysis of the Incorrect Answers

We next analyze the relationship among categories of incorrect answers based on the classification defined in Table 5. Such a relationship can be extracted through the association analysis.

The association analysis is commonly utilized at POS services etc. to extract useful and non-trivial combination of the products which customers like. The Apriori algorithm [11] is often used for the association analysis. An association rule {X}⇒{Y} represents that if a customer buys product X then he also purchases product Y. The Apriori algorithm calculates support, confidence and lift values to evaluate each association rule. The support value supp(X∪Y) of an association rule is the ratio of X and Y among the entire data set and indicates how frequently X and Y appear in the data set. The confidence conf(X⇒Y) is defined by supp(X∪Y) / supp(X) and indicates how often the association rule is true. The lift value lift(X⇒Y) is defined by conf(X⇒Y) / supp(Y). Higher lift value implies that product Y is likely to be purchased in the case that product X is purchased.

We first prepared the data set for each trial of each student and a fill-in-the-blank question. Since a student may solve a question multiple times, all of the trials are treated independently within the data set. As a result, the data set contains 2086 records. Each record contains the answer classifications as items. We then evaluate the extracted association rules based on support and confidence. Table 7 represents the list of the extracted rules having the support values higher than 0.003 (occurs more than 5 times) and the confidence values higher than 0.5. The association rules are sorted in the descending order of confidence.

Table 7: Extracted Association Rules.

No.	Association Rule	Support	Confidence	# of Occur-rences
1	{A,C5}⇒{C7}	0.0077	100%	16
2	{B2,C2,C9}⇒{C1}	0.0053	100%	11
3	{C1,C2,C9}⇒{B2}	0.0053	100%	11
4	{B2,C1,C9}⇒{C2}	0.0053	100%	11
5	{C2,C9}⇒{B2}	0.0053	92%	11
6	{C2,C9}⇒{C1}	0.0053	92%	11
7	{C1,C9}⇒{C2}	0.0053	92%	11
8	{ C1,C9}⇒{B2}	0.0053	92%	11
9	{B2,C1,C2}⇒{C9}	0.0053	92%	11
10	{B2,C9}⇒{C2}	0.0053	85%	11
11	{B2,C9}⇒{C1}	0.0053	85%	11
12	{B2,C2}⇒{C1}	0.0053	85%	11
13	{}⇒{A}	0.793	79%	165
14	{B2,C1}⇒{C2}	0.0058	75%	12
15	{B2,C2}⇒{C9}	0.0052	73%	11
16	{C5}⇒{C7}	0.0144	73%	30
17	{C1,C2}⇒{B2}	0.0058	71%	12
18	{B2,C1}⇒{C9}	0.0053	69%	11

2017 Academic Year

The association rules No. 1 and No. 16 represent that the errors of for statement and variable definition tend to occur simultaneously. This implies that students tend to define a loop variable within a for-statement to cause errors. The other rules are combinations of B2, C1, C2, and C9.

This implies that students without a deep understanding of pointer and array notion tend to misunderstand abstract instructions described in the comment. Such students tend to make careless mistakes.

Such association analysis is useful in various situations. It becomes possible to report frequently observed combinations of incorrect answers to the teachers. Then a teacher can improve his teaching instruction at the class. It also becomes possible to improve the exercise questions to provide more variations of similar questions so that students with various skill levels can learn computer programming according to their programming skills.

7 Analysis of Answering Process

7.1 Required Time to Fill the First Blank

The required time to fill the first blank is calculated from the time at which a student started to answer a question. The required time to fill the first blank can be considered to contain the time not only to answer the current blank but also to understand the entire program. We observe almost no correlation between the required time to fill the first blank and the examination score.

Table 8 shows the average of the required time to fill the first blank for three groups categorized based on the average score of the examinations. We can observe that the group with the highest average score required a longer time than the group with the middle score. This implies that the students in the highest score group understand the entire program before the students start filling the blanks. On the other hand, the students in the middle score group spent less time to understand the entire program. We can guess from these facts that the students with higher achievement tend to understand the entire program at the initial phase of their solving a problem. We also find the lowest score group required the longest time. We consider that the students in the lowest score group needed a long time to understand the entire program and intention of the question.

Table 8: Category of Incorrect Answers of Blank No. 1 in Table 3

Average Exam Score	Required Time to Fill the First Blank	# of Students
High (More than 80)	58.6 sec	70
Middle (between 80 and 60)	49.6 sec	52
Low (Less than 60)	120.7 sec	6

2018 Academic Year

Table 9: Place of Blanks and Required Time to Fill the First Blank

Place of Blank	Average Required Time to Fill the First Blank	# of Students
Program	44.6 sec	22
Trace table	47.5 sec	15
Program and Trace Table	50.5 sec	3

2018 Academic Year

Table 9 represents the average required time to fill the first blank of each place of blanks. To fill the blank within a trace table, the student needs to understand the entire program. Thus we can guess that the student a required longer time to fill the blank within a trace table than the blank within a program. Furthermore, when the blanks are defined within both a program and a trace table, it takes the longest time. This implies that the difficulty level of a fill-in-the-blank question is the highest in this case since the students need to understand the relationship between the program and trace table under the condition that both of them are incomplete.

7.2 Analysis of Answering Process

We shall analyze the answering process of individual students in this section. If a student fills a blank during the answering process multiple times, we utilize the first-filled blanks. In the case that the blanks are defined only within a program and only within a trace table, we found that the student started filling the first blank at the top of the program or the trace table. On the other hand, in the case that the blanks are defined within both of a program and a trace table, the first-filled blank was classified into either a program or a trace table, as shown in Table 10. Table 11 represents the place of the first answers and the average exam scores. We could not recognize the difference depending on the student preference.

Table 10: Place of the First-filling Blank (Question has Blanks within Program and Trace Table)

Question	Topic	Program	Trace Table
(6)-3	if statement	66	24
(7)-1	else-if statement	24	74
(9)-3	if statement	40	56

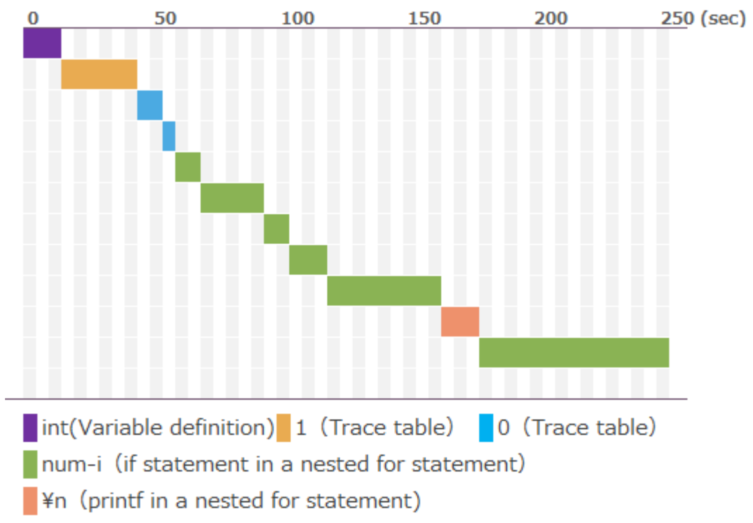
2018 Academic Year

Table 11: Place of the First-Filled Blank and Average Examination Score

Preferred Place of the First-Filled Blank	Average Exam Score (%)	# of Students
Prefer program	77.8	50
Prefer trace table	81.8	48
Same preference	77.1	27

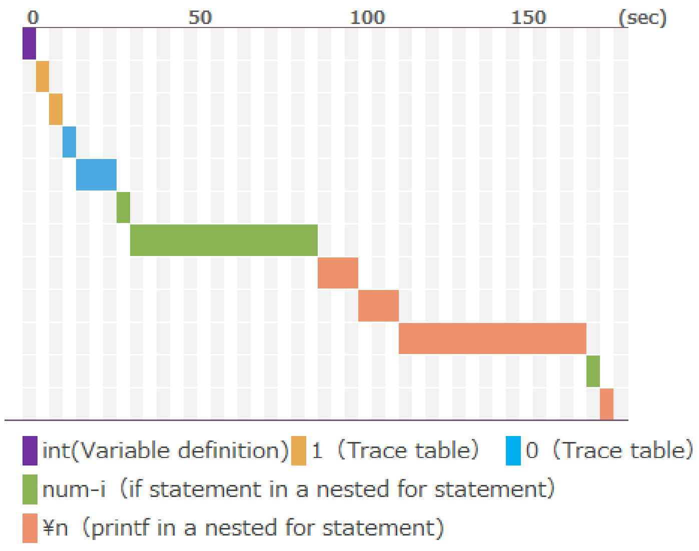
2018 Academic Year

Figures 4 and 5 show the student's answering process in question (15)-2. The horizontal axis represents the elapsed time and each color represents the filling of a blank in (15)-2. The question contains the explanation of the blanks, i.e. correct answer so that the two students achieve a 100% score of the question. Student 1 in Figure 4 has an average score of 95.0% for examination, while Student 2 in Figure 5 has a 75.0% score so that Student 1 has higher programming achievement. The order of both answers is the same, but we can observe that the time taken for the first blank answer is longer for student 1. It is considered that student 1 understood the problem before starting the first filling and then started to answer.



2018 Academic Year

Figure 4: Answering Process (Score=100%, Average Exam Score = 95.0%)



2018 Academic Year

Figure 5: Answering Process (Score=100%, Average Exam Score = 75.0%)

8 Conclusion and Future Work

In this paper, we assigned homework using pgtracer in a programming subject that was started in the first year of the Institute of Technology, Kumamoto College and analyzed the logs collected by pgtracer. We classified the student answers based on the cause of the incorrect answers. As a result, we found frequently observed combinations of errors. We also found that more difficult for the students to fill in the program than to fill in the trace table for the students, that the accuracy rate of the blanks in the program related to the iterative process is low, the students fill the blanks in the order from the top. It was found that the student having the high achievement of programming took more time to fill the first blank. We expect that these analysis techniques and the analysis results are useful to improve programming education through feedback to the class and the teacher.

Future work includes feedback to the teachers to complete the PDCA cycle of programming education using pgtracer. Then it will become possible to create more fill-in-the-blank questions for the students with various achievement levels.

Acknowledgement

This research is supported by JSPS Kakenhi #20K03265 and #20K03232.

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), Brussels, Belgium, pp. 120-128, Dec. 2015.

[3] T. Kakeshita, M. Murata, “Application of Programming Education Support Tool pgtracer for Homework Assignment”, International Journal of Learning Technologies and Learning Environments, Vol. 1, No. 1, pp. 40-61, 2018.

[4] 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.

[5] X. Fu, A. Shimada, A. Ogata, Y. Taniguchi, D. Suehiro “Real-time learning analytics for C programming language courses”, ACM International Conference Proceeding Series, pp. 280-288, 2017.

[6] W. Hering, H. Huppertz, B. J. Kramer, et al. “On benefits of interactive online learning in higher distance education: Case study in the context of programming education”, eLmL - International Conference on Mobile, Hybrid, and On-line Learning 2014, pp. 57-62, 2014.

[7] K. Gotthardt, B. J. Kramer, J. Magenheim, J. Neugebauer “On benefits of interactive online learning in higher distance education: Repeating a learning analytics project in the context of programming education”, International Journal on Advances in Life Sciences Vol, 6, Issue 3-4, pp. 350-363. 2014.

[8] C. Malliarakis, M. Satratzimi, S. Xinogalos, “Integrating learning analytics in an educational MMORPG for computer programming”, Proceedings - IEEE 14th International Conference on Advanced Learning Technologies, ICALT 2014, pp. 233-237. 2014.

[9] K. Ishiwada, Y. Morimoto, S. Nakamura, et al., "Development of a Method for Analyzing Source Code Editing Processes to Estimate Students' Learning Situations", IEICE Technical Report 116(438), pp.75-80. 2017(in Japanese).

[10] H. Igaki, S. Sato, A. Inoue, et al., "Programming Process Visualization for Supporting Students in Programming Exercise", Transactions of Information Processing Society of Japan, 54(1), 330-339, 2013. (in Japanese).

[11] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules", in Proc. 20th VLDB Conference, pp. 487-499, 1994.