A Programming Education Support Tool pgtracer utilizing Fill-in-the-Blank Questions

Overview and Student Functions

Tetsuro Kakeshita
Kosuke Ohta
Ryo Yanagita
Graduate School of Information Science
Saga University, Saga 840-8502, Japan

Abstract— **This paper presents overview and student functions of a programming education support tool pgtracer utilizing fill-in-the-blank questions. Pgtracer runs under Moodle and provides fill-in-the-blank questions composed of a C++ program and a trace table to the students. The tool can provide questions having various difficulty levels from the same program. This can be realized by changing the position of the blanks of the program and trace table. When a student fills the blanks, the tool automatically evaluates the answer. The tool automatically collects answers, time and evaluation result as student log. A teacher can analyze understanding level of the students by analyzing the log.**

Keywords-computer programming; programming education; fill-in-the-blank question; collection and analysis of student log; online education; education support tool

I. Introduction

Programming education is important at university and high school particularly majored in science and engineering. However there are obstacles in programming education since student's ability is declining due to the increase of the students at higher education. The lack of support staff for programming education is also observed frequently. It is usually difficult for a student to evaluate his own achievement level and to choose appropriate problems to improve his programming ability.

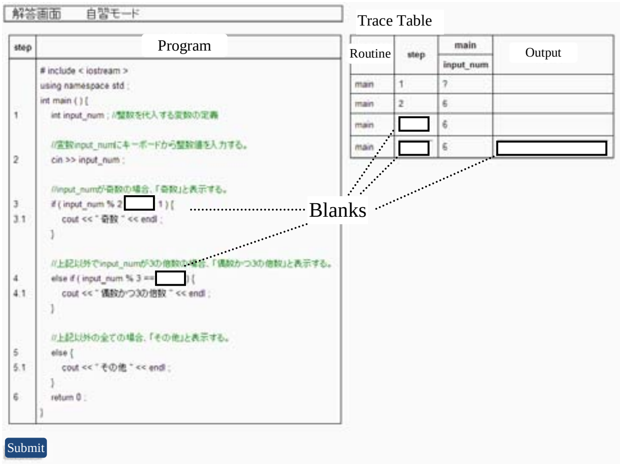
In this paper, we propose a programming education support tool pgtracer utilizing fill-in-the-blank questions in order to quantitatively evaluate student's ability of program comprehension and program development. Pgtracer provides editing functions for questions and themes as well as data analysis functions for teachers. It also provides problem selection function, automatic scoring function, collection functions of study logs as functions for student. We focus on the overview and student functions of pgtracer in this paper.

Pgtracer generates fill-in-the blank questions of program and trace table, and collects student log of the answering process. Then it becomes possible for a teacher to understand the achievement level and learning process of each student and those of the entire class. The teacher can thus utilize the information to realize effective PDCA (plan, do, check, action) cycle for computer programming education.

We conducted an operation test of pgtracer at a programming course that 62 students are enrolled. We prove the feasibility of the analysis of achievement level and learning process of the students through the operation test. We also prove that pgtracer is useful as a self-learning tool through questionnaire result. We collected various comments to improve the usability of pgtracer from students. These comments are utilized to improve pgtracer so that we can expect the increase of the pgtracer users.

II. Fill-in-the-Blank Questions for Programming Education

A fill-in-the-blank question generated by pgtracer is illustrated in Figure I. A question is composed of a pair of a program and a trace table representing execution order of the steps with the routine name and the values of each variable and output at each step. A student fills the blanks such that the program and the trace table become consistent. Trace table is important for program comprehension and can help student when they get stuck during programming [1]. Consequently it is important to visualize execution process of a program for a novice programmer [2]. Thus we can expect that trace table is an effective tool for programming education especially for beginners.



```
step
     # include < iostream >
     using namespace std ;
     int main ( ) {
1       int input_num ; //整数を代入する変数の定義

        //変数input_numにキーボードから整数値を入力する。
2       cin >> input_num ;

        //input_numが奇数の場合、「奇数」と表示する。
3       if ( input_num % 2 □ 1 ) {
3.1        cout << " 奇数 " << endl ;
        }

        //上記以外でinput_numが3の倍数の場合、「偶数かつ3の倍数」と表示する。
4       else if ( input_num % 3 == □ ) {
4.1        cout << " 偶数かつ3の倍数 " << endl ;
        }

        //上記以外の全ての場合、「その他」と表示する。
5       else {
5.1        cout << " その他 " << endl ;
        }
6       return 0 ;
     }
```

Routine	step	main	Output
		input_num	
main	1	?	
main	2	6	
main	□	6	
main	□	6	□

FIGURE I. Fill-in-the-Blank Question of pgtracer

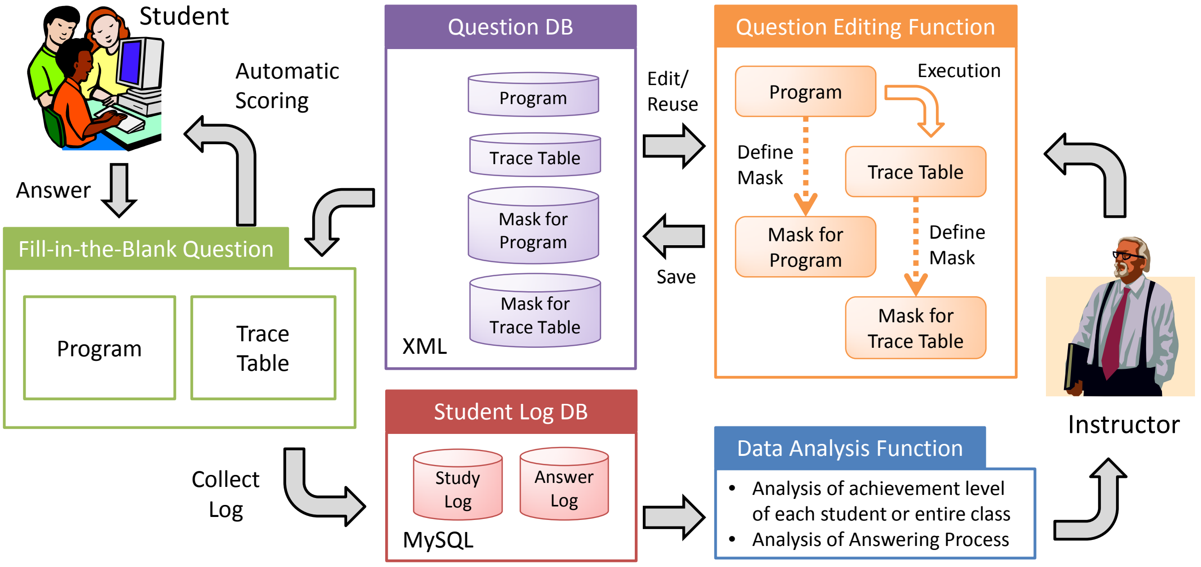


FIGURE II. PROGRAMMING EDUCATION PROCESS UTILIZING PGTRACER

Pgtracer provides various types of blanks within program and trace table so that we can check both program comprehension through blanks within trace table and program composition through blanks within program. Therefore we can evaluate a wide range of students with various levels of achievement by providing the following types of fill-in-the-blank questions.

- 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 corresponding variable.
- 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 complex programming.

There is a case where more than one right answer exists for a fill-in-the-blank question. Pgtracer evaluates all of such answers as correct as long as the execution result of the program and the trace table are consistent with the answer.

III. RELATED WORK

There are various types of research contribution for programming education.

Programming education environment PEN focuses on the programming from the scratch [3]. It provides functions for stepwise execution and monitoring of variables. It is usually the case that the learning process is more efficient by utilizing pgtracer since a student only has to fill the blanks.

Taguchi proposed a method to analyze achievement of a student through collaborative filtering [4]. There are other works to analyze programming ability of a student [5]. Pgtracer focuses on the collection of student's learning data so that the techniques proposed by these works can be utilized to analyze programming ability of the students and to automatically select appropriate questions depending on the achievement level of the student.

IV. PROGRAMMING EDUCATION UTILIZING PGTRACER

Pgtracer is developed as a plug-in of well-known lecture management system Moodle. This approach enables to provide a one-stop service for programming education utilizing other educational contents. A student can learn online for 24 hours and 365 days.

Figure II represents a process of programming education utilizing pgtracer. A fill-in-the-blank question is composed of a program, a trace table, a mask for program and a mask for 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 program. Trace table and the corresponding masks are separated for the same reason. Question DB contains valid combination of the XML files.

When a student answers to a question, the system automatically evaluates the answer and feedbacks the score to the student. The student can then 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. The instructor 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 program, trace table, mask for program and mask for trace table. XML files representing program and trace table are generated by pgtracer. Pgtracer also provides functions to edit the XML files representing masks. The generated files and configurations are stored at the question DB.

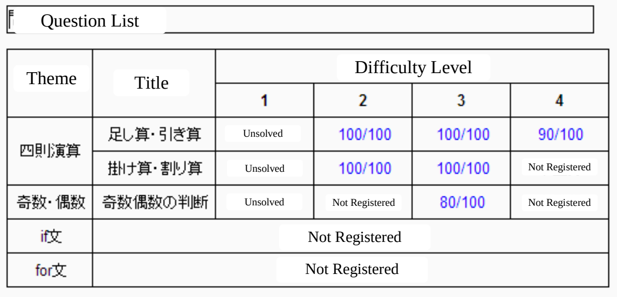
V. Student Functions of Pgtracer

A. *Question List and Achievement Summary*

When a student starts pgtracer, pgtracer first shows the question list (Figure III). The question list represents the questions categorized by theme, title and difficulty level. The difficulty level of a question is defined depending on the place of the blanks within the question. A student selects a question from the list or an instructor directs the student to solve specified questions.

The question list also represents the achievement summary of the student. When the student solves a question, score of the answer is calculated immediately and the maximum score for the question is shown with the highest possible score. "Not registered" represents that no question is registered by the instructor. "Unsolved" represents that the student has not solved the question.

Each of the score and "unsolved" corresponds to the links to the question. A student can take the question by clicking the link.



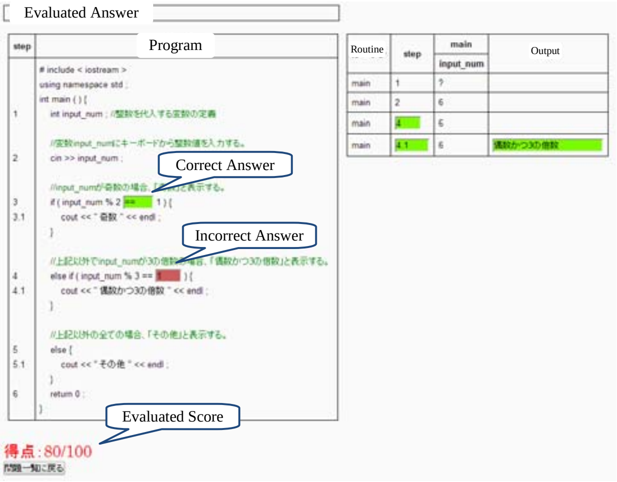
Theme	Title	Difficulty Level			
		1	2	3	4
四則演算	足し算･引き算	Unsolved	100/100	100/100	90/100
	掛け算･割り算	Unsolved	100/100	100/100	Not Registered
奇数･偶数	奇数偶数の判断	Unsolved	Not Registered	80/100	Not Registered
if文	Not Registered				
for文	Not Registered				

FIGURE III. Question List and Achievement Summary

B. *Automatic Scoring and Feedback*

When a student clicks a link on the question list, he can take the corresponding question (Figure I). The student can choose arbitrary blanks to fill before completing the question. When he clicks the "Submit" button at the end of the question, pgtracer automatically evaluate the answer by executing the completed program and compares the execution process with the completed trace table.

The total score is illustrated at the bottom of the score menu with red font (Figure IV). Each of the blanks is colored in green (correct answer) or red (incorrect answer). When the student points at a red blank, pgtracer shows the right answer to provide a feedback to the student (Figure V).



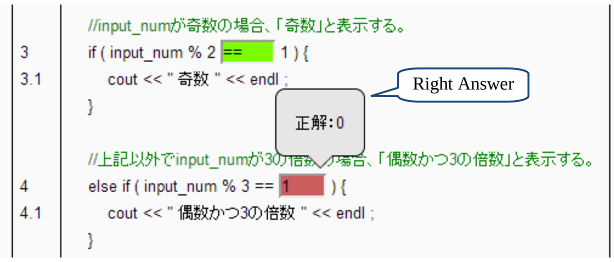
```
step  Program
      # include < iostream >
      using namespace std ;
      int main ( ) {
1       int input_num ; //整数を代入する変数の定義

        //変数input_numにキーボードから整数値を入力する。
2       cin >> input_num ;

        //input_numが奇数の場合、「奇数」と表示する。
3       if ( input_num % 2 == 1 ) {
3.1       cout << " 奇数 " << endl ;
        }

        //上記以外でinput_numが3の倍数の場合、「偶数かつ3の倍数」と表示する。
4       else if ( input_num % 3 == 1 ) {
4.1       cout << " 偶数かつ3の倍数 " << endl ;
        }

        //上記以外の全ての場合、「その他」と表示する。
5       else {
5.1       cout << " その他 " << endl ;
        }
6       return 0 ;
      }
```

Routine	step	main	Output
		input_num	
main	1	?	
main	2	6	
main	4	6	
main	4.1	6	偶数かつ3の倍数

得点：80/100

問題一覧に戻る

FIGURE IV. Automatic Scoring of a Question



```
3       //input_numが奇数の場合、「奇数」と表示する。
        if ( input_num % 2 == 1 ) {
3.1       cout << " 奇数 " << endl ;
        }
                                    正解：0
        //上記以外でinput_numが3の倍数の場合、「偶数かつ3の倍数」と表示する。
4       else if ( input_num % 3 == 1 ) {
4.1       cout << " 偶数かつ3の倍数 " << endl ;
        }
```

FIGURE V. Feedback Function for Incorrect Answer

Pgtracer provides two modes for automatic scoring and feedback. They are examination mode and self-learning mode. The instructor can choose a desired mode for each question. In case of the examination mode, the automatic scoring is executed only when a student clicks the "Submit" button as explained above. On the other hand, pgtracer checks the answer when the student fills a single blank in case of the self-learning mode.

Automatic scoring for the self-learning mode is executed by filling out the blanks with the right answer. Furthermore, the student can choose whether to show the right answer for the red blanks. When the student chooses not to show the right answer, pgtracer only shows the scoring result by the color of the blank.

C. *Assist Function to Fill out Trace Table*

It is possible to define many blanks for a question. In such case, a student has to do a tedious task to fill out the blanks by a same value especially for the trace table. Pgtracer provides assist function to automatically fill out blanks of trace table to prevent such situation. The assist function is useful to keep motivation of the students.

The assist function works as follows.

- When a student fills a blank corresponding to a value of a variable, pgtracer automatically fills the same value to the succeeding columns of the same row.
- If there is a column which the student enters a value himself among the succeeding columns explained

above, pgtracer does not fill the column and the succeeding columns.

Figure VI illustrates an example of the assist function. At first the student enters “10” to the green column at the left table. Then pgtracer automatically fills the consecutive columns of the same row. Next the student enters “15” to the red column at the center table. Pgtracer then complements the consecutive columns. Finally the student enters “30” to the second column at the right table. Pgtracer fills the value to the two consecutive columns but does not fill the remaining column since they are set by the student himself.

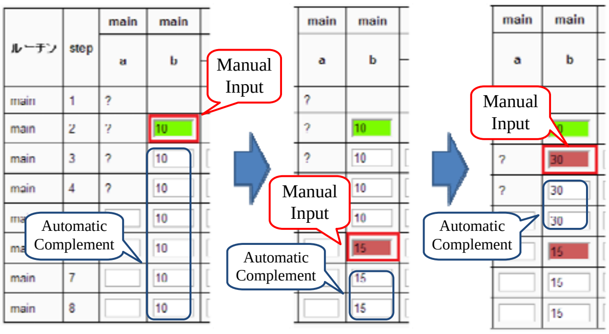


FIGURE VI. ASSIST FUNCTION TO FILL OUT TRACE TABLE

VI. EVALUATION

We conducted an experimental evaluation of pgtracer with 62 students taking a programming course at Saga University. We prepared 84 questions with 21 titles and four different levels. Approximately half of the questions are defined as the examination mode and the remaining questions are defined as the self-learning mode.

A. *Workload of Students*

Figure VII represents the distribution of the workload of the students. Although some students work more than 5 hours, most of the student works between 1 and 4 hours. An average student takes 55.1 questions during the experiment so that the efficiency of learning through pgtracer is expected to be higher than the programming exercise from the scratch.

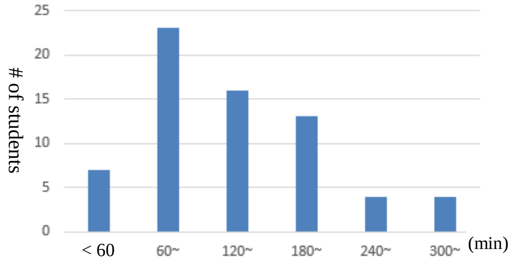


FIGURE VII. DISTRIBUTION OF STUDENT WORKLOAD

B. *Evaluation of of Elapsed Time for Each Blank Type*

We next analyze the elapsed time to fill a blank for each blank type. TABLE I represents the result.

The readers can observe that the time to fill a blank of a program is longer than the time to fill a blank of a trace table. This corresponds to the difference of difficulty between program comprehension and program composition.

TABLE I. MEAN TIME TO FILL A BLANK FOR EACH CATEGORY

Category			**Answer Time (sec)**
Trace Table	Value of a Variable	Same	5.34
		Not Same	9.14
	Step Number	Consecutive	5.04
		Non-consecutive	7.41
	Variable Name		11.55
Program	Single Token		9.75
	Expression		19.45
	Statement		22.24

Within the trace table, required time for a value of a variable and the time for a step number are almost the same. However the time to fill a variable name is longer than the other case. The time to fill a variable value becomes longer when the value is modified at the step. Similarly the time to fill a step number becomes longer when the step number is not consecutive due to iteration or conditional branch.

It is natural that the time becomes longer depending on the blank of single token, expression and statement within a program.

C. *Satisfaction of the Students*

We collected feedback from the students. More than 90% of the students answered that pgtracer is useful to confirm grammar of the programming language and to evaluate their programming ability. 48 % of the students answered that the usability of pgtracer is good or excellent and 43% answered for average level.

VII. CONCLUSION

We developed and evaluated the programming education support tool pgtracer. We focused on the general principle and student functions in this paper. The teacher functions including question editing function and data analysis function will be presented in our next paper [6].

REFERENCES

[1] T. Egi, A. Takeuchi, “Development and evaluation of debugging support system of guide tracing for befinners”, Journal of Japan Society for Educational Technology, Vol. 32, No. 4, 369-381, 2009. (in Japanese)

[2] N. Huang, et al., “A programming educational system based on program visualization”, Technical report of IEICE ET99-1, 1999. (in Japanese)

[3] 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)

[4] H. Taguchi, et al., “Programming training of students according to individual understanding and attitude”, Journal of Information Processing, Vol. 48, No. 2, pp. 958-968, 2007. (in Japanese)

[5] H. Nakajima, et al., “A QA cycle for education of programming: A mechanism for automatically putting questions corresponding to learner’s skill”, Journal of IEICE, Vol. J88-D-I, Vol. 2, pp. 439-450, 2005. (in Japanese)

[6] T. Kakeshita, et al., “A programming education support tool pgtracer utilizing fill-in-the-blank uestions: teacher functions”, Proc. 2nd International Conf. on Education Reform and Modern Management (ERMM 2015), Atlantis Press, April 2015, 4 pages. (in Press)