Automatic Generation of C Source Code for Novice Programming Education

Shimpei Matsumoto[1(✉)], Koki Okimoto[2], Tomoko Kashima[3], and Shuichi Yamagishi[1]

[1] Faculty of Applied Information Science, Hiroshima Institute of Technology, 2-1-1 Miyake, Saeki-ku, Hiroshima 731-5193, Japan
s.matsumoto.gk@cc.it-hiroshima.ac.jp

[2] Graduate School of Science and Technology, Hiroshima Institute of Technology, 2-1-1 Miyake, Saeki-ku, Hiroshima 731-5193, Japan

[3] Faculty of Engineering, Kinki University, 1 Takaya Umenobe, Higashi-hiroshima City, Hiroshima 739-2116, Japan

Abstract. To efficiently support novice programming learners feeling programming difficult, clarifying the cause of preventing programming understanding, and developing a new teaching method appropriate for their understanding degree would be necessary. The objective of this paper is to develop a learning support system with reading source codes. This paper also aims to evaluate the effectiveness of the developed system from the subjective viewpoint of learners. The developed system can automatically generate a source code of C programming language in which there is no particular meaning because the source codes as learning materials are generated randomly. The developed system was utilized in a programming lecture for novice programming learners. This paper obtained student responses from a questionnaire, after the students had completed one semester of the instruction in programming, and analyzed the data. From the analysis result, it turned out that different evaluation patterns existed depending on the learner's basic programming skill.

1 Introduction

With the rapid development of information technology, today's society has been requiring a lot of software development engineers. Among software development, programming has been considered as an essential skill. However, usual programming instructional methods cannot afford to neglect an issue: polarization. The polarization consists of two group; the one is a group of learners who are readily acceptable to learn to program, and the other is a group of learners who does not receive any concept of programming. This problem would be uniformly continued to be repeated even though the programming education has promoted and various learning materials of programming have been actively developed. About this problem, Konecki mentioned that programming education comes with many reoccurring problems and difficulties that its novice learners experience, and offered suggestions on these reasons [1]. Our previous study [2,3] also noted

M. Kurosu (Ed.): HCI 2016, Part I, LNCS 9731, pp. 65–76, 2016.
DOI: 10.1007/978-3-319-39510-4_7

a similar tendency. An effective instructional method to support a learner not good at programming has not constructed enough as long as the authors survey. This reason would be in the point that programming requires various skill such as logical thinking, language, imagination, ingenuity, and mathematical ability. On the other hand, we are not still sure what and how much skill is necessary for programming. Additionally, there is no method to indicate learner's detail of the degree of each skill, so a learner is also not sure what kind of skill should be trained to overcome the difficulty of programming. Therefore, a system and a method, which can effectively collect data of learning leading to the definition of understanding degree in each skill field while supporting daily programming education, would be effective for a learner not good at programming.

The aim of this paper is to develop a source code reading based learning support system to enrich a programming lecture, which is available for self-study and mini-examination. The developed system of this paper can automatically generate a source code of C programming language used as a learning material. Since the source code is generated randomly, there is no particular meaning in the source code. The type of question with the source code is to answer the proper value of a variable after the execution of source code. With the developed system, a learner would learn the basis of the processing flow, memory retention, calculation ability, and basic knowledge such as the assignment operator and the increment/decrement which are the difficult concept for learners not good at programming. The developed system aims to equip minimum knowledge, but necessary for programming without describing a source code, and to rebuild his/her confidence to make a program. The authors expect that the writing skill of source code would be eventually improved because the psychological resistance will be reduced from his/her confidence for programming. Additionally, experiencing much source code reading not depending on a particular context will contribute to writing a readable code [4,5]. The developed system would be also useful as a method to grasp what and how much a skill is insufficient for improving programming ability because it can collect each learner's response according to the characteristic of a question from daily education.

This paper also aims to evaluate the effectiveness of the developed system from the subjective viewpoint of learners. The developed system was utilized in a programming lecture for novice programming learners majoring Informatics and tried to support the instruction of source code reading. This paper obtained student responses from a questionnaire, after the students had completed one semester of the instruction in programming, and analyzed the data. From the analysis result, it turned out that different evaluation patterns existed depending on the learner's programming skill.

2 Developed System

2.1 Source Code Reading

The developed system in this paper is for training the skill of reading source code of C language. Writing a program is particularly difficult for learners being not

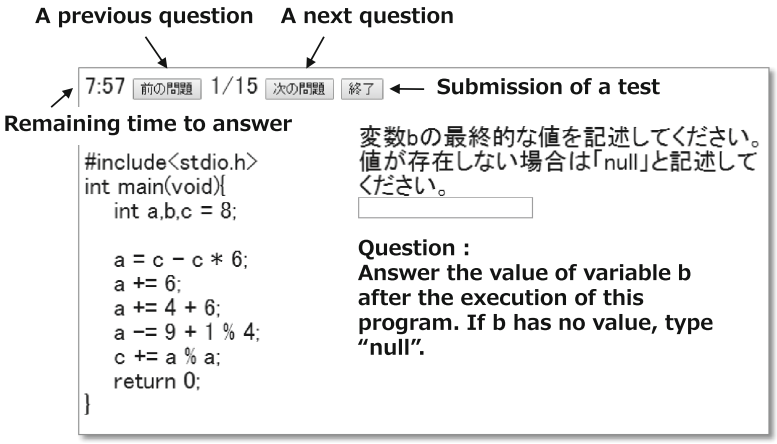

Fig. 1. A screenshot of the developed system

good at programming, so this system tries to reduce the psychological resistance to programming.

The system can automatically generate source codes as a learning material and give a question that requires students to answer the proper value of a variable after the execution of a source code. The source code is randomly generated, i.e., it consists of the series of meaningless procedure, so it does not have a meaning to process. This paper considers that a source code without a certain meaning would be effective to equip the basic knowledge of programming language specification. The total lines of the generated source are several dozen. Such source code is not common because a source code with several dozen lines is too short to process a certain meaning work, but this paper considers that reading source code described in a short sentence is important to get used to a large-scale practical program as a first step. In addition to these, recently since many software development sites have actively adopted programmer-centered software development approach with utilizing open source software, the need for reading technique of source code not depending on comment is now becoming greater and greater. Additionally, most of the time to program is said to be reading [4]. These backgrounds mentioned above are the main reasons that the authors focused on the source code reading blended programming education.

2.2 Specification of Implementation

The developed system is a web application. All functions of the developed system are available through the Internet as long as a client device equips a web browser available the standard of HTML5. The developed system runs on an operating system with Linux kernel 3.16, and uses the following software: Apache 2.4.7, a web server program, to provide web service, PHP 5.5.9, a server-side scripting language, to implement the function, jQuery 1.7.2, a cross-platform JavaScript

library, to perform dynamic UI, and MySQL 5.6.16, a database server program, to manage all system data.

Figure 1 shows an example of a screenshot of the developed system, and the source code shown in Fig. 1 is a question for a learner. The question type requires a learner to answer the value of a variable after the execution of a presented source code, and the variable is randomly chosen. Some question styles are available in the developed system, such as a free descriptive question, and a multiple choice question. The details are as follows.

Free Descriptive Question. This type of question is addressed in this paper. All learners should put an integer number into a specified textbox by using a keyboard. The developed system picks up a variable randomly from a presented source code, and the variable is used as a question. There is another type of free descriptive question which requires answering all variable values in a presented source code, but this capability is not mentioned in this paper.

Multiple Choice Question. This type of question requires a learner to select an option whose all variable values after execution are proper. All variable values in an option are different each time except the correct option because they are randomly generated.

Alignment Question. This type of question is multiple choice, and the question requires a learner to select an option with the proper order of statements. The distracter's order of statements is randomly generated. As all variable values after execution are shown, each learner can think the order of statement by using the values. The difficulty level of this question is not influenced much by the combination of the distracters, and this point is different from the above two types of question.

Statement Supplement Question. A source code lacking a statement, and all variable values of this source code after execution are given to a question. This type of question is multiple choice, and the question requires a learner to select an option with a proper statement for the lacked place. The lacked place is randomly given, and the distracters are also randomly generated.

As shown above, this paper addresses only the trial result of free descriptive question. All source codes used as questions consist of only sequential processing, do not include branch and repetition to give learners a firm foundation in programming. Besides, all source codes consist of only integer variables. In the developed system, a question and its difficulty level differ for each learner even though the source code is same. Therefore, a sufficient number of questions statistically guaranteed should provide when executing a test.

2.3 Automatic Generation of a Question

The developed system has the capability to generate a source code automatically as a question. As the statements of a source code are randomly ordered, there is no meaning in the source. The administrator of this system can give following the conditions to obtain a source code by the configuration form:

Table 1. Conditions to generate a source code

Term	S1	S2	S3	S4	S5	S7	S11	S12	S13
1	3	4	3	0.3	+,-,		*		
2	3	4	2	0.3	+,-		*		
3	3	4	3	0.3	+,-,*,/,%	+,-	*		
4	3	4	3	0.4	+,-,*,/,%	+,-	*		
5	3	4	2	0.4	+,-,*,/,%	+,-,*,/,%	*	*	
6	3	4	2	0.6	+,-,*,/,%	+,-,*,/,%	*	*	

- S1: The number of variables: 1–9 variables are available, where variable names are alphabetical sequence, i.e., a, b, c, ...,i or array a[0], a[1], ..., a[9].
- S2: The number of statements: 1–10 statements are available, where a closing curly bracket does not include the number.
- S3: The maximum number of variables calculated: 1–10 variables are available.
- S4: Appearance ratio of variables in a calculation: 0–100% is available.
- S5: Use/disuse of operators: use/disuse of addition, subtraction, multiplication, division, and remainder operators are selectable. A source code consists of only simple assignment statement when all operators are disused.
- S6: Use/disuse of the selection (if) and the repetition (for, while) statement.
- S7: Use/disuse of compound assignment operators: use/disuse of addition, subtraction, multiplication, division, and remainder operators are selectable.
- S8: Constraints for conditional expression: 4 settings are available for conditional expression of "if" or "while": only numerical values (both right and left sides), variables and numerical values (Appearance ratio of variables is stochastically determined), variables and numerical values with a constraint (a variable is certainly in the left side), only variables (both right and left sides).
- S9: Complex condition: use/disuse of complex condition, and the number of conditions (1–9).
- S10: Nest: use/disuse of nest, the number of nests (1–9), and the appearance ratio of nest (0–100%).
- S11: Use/disuse of increment/decrement.
- S12: Use/disuse of initialization.
- S13: Appearance ratio of a redundant statement (0–100%): e.g. "a = 1; a = 2;".

3 Experiment and Result

This paper tried to support a programming lecture by using the developed system. The lecture is for freshman students majoring in Informatics, most of them are a beginner of programming, and used C language. This programming lecture continued one semester, and it consisted of 15 lectures. The developed system was utilized with examination style in 6 lectures among the 15 lectures. The

```
#include<stdio.h>
int main(void){
    int a,b,c = 1;

    c = c - 7;
    a = c - 2 + 5;
    b = 7 + 5;
    c = 9 - a - 9;
    return 0;
}
```

(a) A question in the 1st test

```
#include<stdio.h>
int main(void){
    int a,b;

    b = 5 - 10;
    a = 2 + b;
    b = b - a - b;
    return 0;
}
```

(b) A question in the 2nd test

```
#include<stdio.h>
int main(void){
    int a,b,c = 6;

    a = 3;
    c = 9 % c;
    b = c / a;
    c = a / 6;
    return 0;
}
```

(c) A question in the 3rd test

```
#include<stdio.h>
int main(void){
    int a = 2,b,c;

    a -= a;
    b = a;
    b += 9;
    c = a / 9 + b;
    return 0;
}
```

(d) A question in the 4th test

```
#include<stdio.h>
int main(void){
    int a,b,c = 2;

    b = 8;
    a = ++c;
    c += 7 * 8;
    b += 2;
    return 0;
}
```

(e) A question in the 5th test

```
#include<stdio.h>
int main(void){
    int a = 6,b,c;

    c = a++ * a;
    a -= c-- - ++c;
    b = a;
    a -= 5;
    return 0;
}
```

(f) A question in the 6th test

Fig. 2. Examples of a question in each term

author provided 15 questions for each test, and each testing time is 10 min. The amount of knowledge in each test was related to the contents of the lecture according to the progress of this lecture. Concretely, in the early stage of this lecture, a test includes only simple calculation with addition and subtraction, on the other hand, in the closing stage, svarious operators and techniques are given to generate a source code in the late stage. The detail of the settings to generate a source code is shown as Table 1, and examples of a question in each test are shown in Fig. 2. In this paper, as the source codes only including simple assignment operators were given for students, the settings for generation code S6, S8, S9, S10 are not mentioned. After each test, questions included in a test are available to practice as much as a learner wants.

This paper obtained 108 student responses from the questionnaire, after the students had completed one semester of the instruction in programming, and analyzed the data. The objective of the analysis to check the effectiveness of reading source code, so each question does not ask students for the effectiveness of the developed system itself. Namely, as the developed system is only to facilitate the automation of scoring a test, what this paper wants to do is to clarify the degree of contribution of reading source code for programming learning, and the content of source codes generated by a method in this paper. This paper gave 15 questions asking how much reading source code contributed to the programming learning and obtained student responses with 6 grade Likert scale. Questions given to a student are as follows.

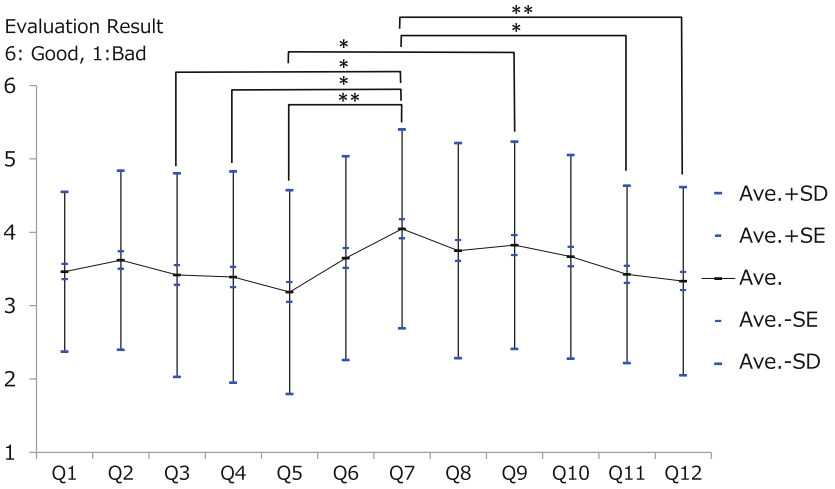

Fig. 3. Evaluation result of all students

- Q1: The degree of contribution for understanding the fundamental concept of programming.
- Q2: The degree of contribution for understanding the mechanism of assignment operator and the sequence of operation in the assignment operator.
- Q3: The degree of contribution for understanding/giving attention to the advanced knowledge such as compound assignment operator and increment/decrement.
- Q4: The degree of contribution to getting used to programming.
- Q5: The degree of contribution to reading another man's program and for fixing a bug.
- Q6: The degree of contribution to getting used to a calculation with a remainder operator.
- Q7: Will you be able to read the remainder operator "%" properly if you continue to train reading source code?
- Q8: Will you be able to read the increment/decrement "++, - -" properly if you continue to train reading source code?
- Q9: Will you be able to read the compound assignment operators "+=, -=" properly if you continue to train reading source code?
- Q10: Will you be able to read the compound assignment operators "*=, /=, %=" properly if you continue to train reading source code?
- Q11: Will you be able to deal with the difference in the sequence of operation depending on compilers?
- Q12: The degree of contribution of reading randomly and automatically generated source code for acquiring knowledge of programming.

Questions Q1–6 are to reveal the contribution of reading source code for programming learning, Q7–10 are to reveal the expectation of reading source

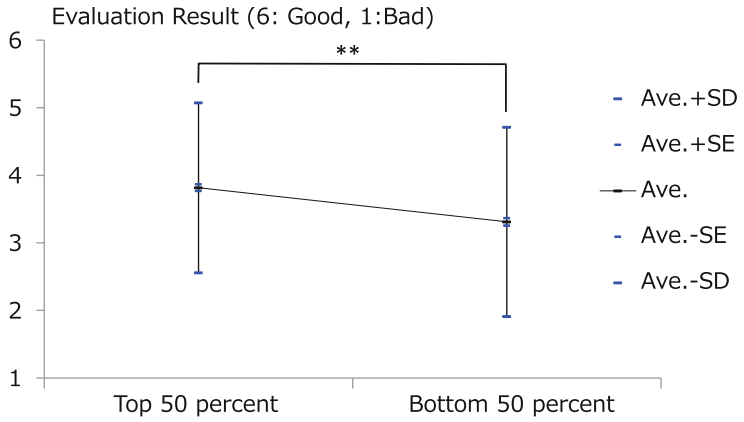

Fig. 4. Gap between top and bottom 50 %

code for improving programming skill, and Q11 and Q12 are to reveal the quality of source code.

From the normality testing with skewness and kurtosis for each item, questions except Q1 indicated normality ($p < 0.05$). Therefore, this paper assumed each learner's response as interval scale and applied parametric testing method. Additionally, from the Bartlett's and Levene's tests, the null hypothesis of equal variances was rejected ($p < 0.01$), and the result suggested that the result of each item's response includes its own feature. The summary of evaluation results is shown in Fig. 3, where * indicates a significant difference $p < 0.05$, and ** indicates $p < 0.01$ respectively. From the high evaluation results of Q7 and Q9, the reading source code blended programming learning would be strongly expected for improving the reading of remainder operator and compound assignment operators +=, -=. On the other hand, the results of Q3, Q4, Q5, Q11, and Q12 were relatively low, and especially the evaluation result of Q5 was worst. From these findings, most students would feel that reading the automatically generated source codes contributed to acquiring the fundamental programming knowledge, but the effectivenesses for getting used to programming, reading another man's program and fixing a bug were less. As these reasons, the source codes in this experiment were strikingly different from a practical source code and were not the contents for fixing a bug. Based on the assumption, the evaluation for fixing a bug will improve if a source code with a bug is used as a question. Similarly, the evaluation of Q4 will improve if a practical source code is used as a question. The results of Q3, Q11, and Q12 are considered to be strongly related, i.e., the factor in Q12 affected the result to Q3 and Q11. Reading randomly generated source codes forced to touch a confusing thing about programming. Concretely, the factor in Q12 would give a strong impact to the evaluation of Q11 because the execution result of a complex statement including both a compound assignment operator and a increment/decrement differs according to compiler. For example, the execution result of the source code (f) shown in Fig. 2 depends on the kind

Table 2. Comparison of each item's average value between top and bottom and its significant difference

Students	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Top	3.593	3.833	3.556	3.556	3.389	3.852	4.537	3.944	4.370	4.056	3.778	3.333
Bottom	3.333	3.407	3.278	3.222	2.981	3.444	3.556	3.556	3.278	3.278	3.074	3.333
Sig. Dif		*					**		**	**	**	

of compiler, and also its version. This point would be an extremely troublesome mechanism for students not good at programming. Similarly, the consideration that Q12 would give a high impact to Q3 was based on the assumption that source codes including extremely complex statements as shown in Fig. 2 would cultivate a sense of fear for programming.

All students in this experiment have taken a course in the algorithm at last semester. In this course, students learned a fundamental concept of algorithms, such as sequence, selection, and repetition, which are also the essential knowledge to the program. It is reasonable to assume that the score of the algorithm course would be strongly related to the programming skill because the correlation of score between algorithm and programming was high in the previous data. Based on this reason, students were divided into 2 groups based on the score of the algorithm, the top 50 % of 54 students, and the bottom 50 % of 54 students. Two-way ANOVA analyzed the difference of the averages of all items for each group. As shown in Fig. 4, there was a significant difference $p < 0.01$ between 2 groups. Since the evaluation result of the bottom 50 % students, whose basic skill of programming are not probably enough, was lower than the top 50 % one, the result would suggest that reading the source codes generated by the method in this paper failed to contribute fully to reduce a resistance for programming. However only in this result, we cannot say for sure that a student whose programming skill is insufficient is certainly a student feeling not good at programming. Therefore, shortly we will obtain a subjective evaluation of the developed system from the interview of a student not good at programming. This paper examines the reason of the tendency in Fig. 4 as follows. A student with enough programming skill can realize one's sufficient/insufficient points in programming from the scoring result of reading source codes because he/she had enough skill to make a self-assessment. The developed system gave an opportunity to reflect, so he/she would give the learning of reading source codes a good evaluation. On the other hand, a student whose fundamental programming skill is not enough cannot do them. Therefore, positive correlation between the basic programming skill and the evaluation result of reading source code is considered to be quite natural. Based on this consideration, to make the developed system more acceptable to every student, we ought to give feedback with an easy-to-understand comment just after the scoring to support one's reflection about reading.

Table 2 shows each item's averages of the top and the bottom 50 % students with significant differences. The result of the top and the bottom was same in

Q12, and it is the noted point because the whole trend is that the top's evaluation was over the bottom's evaluation, but Q12 was its exception. The results must be either the high expectation from the bottom or the low expectation from the top about reading the source codes generated by the method in this paper. Considering a balance between Q12 and the other questions, the former assumption is natural. Due to this reason, the significant differences would be generated by the high evaluation of the top 50 % students. So, we might be possible to devise a way to improve the satisfaction of the bottom because there is enough possibility to improve the evaluation of the questions with a significant difference. Namely, this paper considers that we can reduce the gap to the level with no significant difference, and a method to reduce the gap is to give feedback mentioned above.

4 Related Works

Previous studies on programming are roughly classified into three objectives [6,7]: analysis of student's learning history data and discovery of student's characteristic, proposal of a programming teaching method, and development of ICT-based learning support system. So far, researches on programming education have often focused on writing source code, but some of them have addressed reading source code [8]. Reading source code has also been considered to be an important ability for obtaining a deeper understanding of programming, but good reading strategies have not been known so much.

Earlier programming education studies on reading source code provide a lot of valuable knowledge about a characteristic of programming learners, teaching method, and programming learning support system. Firstly, this paper shows the remarkable efforts of software development. A framework on reading source code was proposed [9] where the process of reading code consists of two steps: reading comprehension and meaning deduction. Reading comprehension was defined as the inverse of coding, and meaning deduction was also defined as the inverse of algorithm design. Arai et al. developed a learning support system for programming beginners that facilitates the process of learning by reading source code, and confirmed that the system was effective [10]. Arai's work trained the skill to convert source code into an equivallent abstract processing flow, and the skill was positioned as the reading comprehension. There is also an interesting learning support system for code reading [11,12]. They assumed that learners will understand programs and algorithms by recalling an image consisting of three fields: the program code, objects processed by the program, and a sequence of concrete operations for the objects. Their proposed system visualized the three fields and their relationships to support understanding the relationships and correspondence among their components. Additionally, they discussed how code reading using their proposed system allows learners to cultivate a superior understanding of programming. Tang developed a distributed, social code review tool designed for the specific constraints and goals of a programming course named "Caesar" [13]. Caesar is capable of scaling to a large and diverse reviewer population,

provides automated tools for increasing reviewer efficiency, and implements a social web interface for reviewing that encourages discussion and participation.

Busjahn et al. took a further look into the role of reading source code in programming learning from the result of interviews with programming instructors using the miracle question, on the role of code reading and comprehension [14]. They claimed that a possible means to foster programming learning is to teach reading directly, including reading strategies, and besides, reading should probably be made more explicit as learning goal in itself.

Lopez analyzed student responses to an examination, after the students had completed one semester of instruction in programming [15]. He showed that the performance of students on code tracing tasks correlated with their performance on code writing tasks, and a correlation was also found between performance on "explain in plain English" tasks and code writing.

5 Conclusion

This paper developed a learning support system for reading a source code and showed the detail of the developed system. This paper implemented a function of automatic C source code generation for a learning material. This paper also aimed to evaluate the effectiveness of the developed system from the subjective viewpoint of learners and tried to support a programming course by using the developed system. Mini-tests were given several times in the programming lecture by using the function of automatic generation of C source codes. This paper obtained 108 student responses from the questionnaire, after the students had completed one semester of the instruction, and analyzed the data. From the analysis result, firstly it turned out that most students would feel that reading the automatically generated source codes contributed to acquiring the fundamental programming knowledge, but the effectivenesses for getting used to programming, reading another man's program, and fixing a bug were less. From the evaluation result, we can understand that reading randomly generated source codes forced to touch a confusing thing about programming, and it would cultivate a sense of fear for programming. Secondly, students were divided into 2 groups based on the basic programming skill, and Two-way ANOVA analyzed these differences. The analysis result showed that reading the source codes generated by the method in this paper failed to contribute fully to reduce a resistance for programming. Based on this result, it was found that we ought to give feedback with an easy-to-understand comment just after the scoring to support one's reflection about reading to improve the student's satisfaction not good at programming. But, we confirmed that it might be possible to devise a way to improve the satisfaction of the bottom group of students because there would be enough possibility to improve the evaluation from the data of responses.

Acknowledgments. This work was partly supported by Japan Society for the Promotion of Science, KAKENHI Grant-in-Aid for Young Scientists (B), No. 13304922, Grant-in-Aid for Scientific Research(C), No. 26350296.

References

1. Konecki, M., Petrlic, M.: Main problems of programming novices and the right course of action. In: Proceeding of the 25th Central European Conference on Information and Intelligent Systems, pp. 116–123 (2014)
2. Kashima, T., Matsumoto, S., Yamagishi, S.: Knowledge acquisition with eye-tracking to teach programming appropriate for learner's programming skill. In: Proceeding of The Third Asian Conference on Information Systems, pp. 287–292 (2014)
3. Okimoto, K., Matsumoto, S., Yamagishi, S., Kashima, T.: A source code reading based learning support system for novice programming education. In: Proceedings of the 21nd International Symposium on Artificial Life and Robotics, PS3, pp. 765–768 (2016)
4. Boswell, D., Foucher, T.: The Art of Readable Code. (Theory in Practice), O'Reilly Media (2011)
5. Spinellis, D.: Reading, writing and code. ACM Queue **1**(7), 84–89 (2003)
6. Robins, A., Rountree, J., Rountree, N.: Learning and teaching programming: a review and discussion. Comput. Sci. Educ. **13**(2), 137–172 (2003)
7. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., et al.: A survey of literature on the teaching of introductory programming. ACM SIGCSE Bulletin **39**(4), 204–223 (2007)
8. Lopez, M., Sutton, K., Clear, T.: Surely we must learn to read before we learn to write! In: Proceedings of the Eleventh Australasian Conference on Computing Education, vol. 95, pp. 165–170 (2009)
9. Kanamori, H., Tomoto, T., Akakura, T.: Development of a computer programming learning support system based on reading computer program. In: Yamamoto, S. (ed.) HCI 2013, Part III. LNCS, vol. 8018, pp. 63–69. Springer, Heidelberg (2013)
10. Arai, T., Kanamori, H., Tomoto, T., Kometani, Y., Akakura, T.: Development of a learning support system for source code reading comprehension. In: Yamamoto, S. (ed.) HCI 2014, Part II. LNCS, vol. 8522, pp. 12–19. Springer, Heidelberg (2014)
11. Kogure, S., Okamoto, M., Yamashita, K., Noguchi, Y., Konishi, T., Itoh, Y.: Evaluation of an algorithm and programming learning support features to program and algorithm learning support environment. In: Proceedings of the 21st International Conference of Computers in Education, pp. 418–424 (2013)
12. Yamashita, K., Nagao, T., Kogure, S., Noguchi, Y., Konishi, T., Itoh, Y.: Code-reading support environment visualizing three fields and educational practice to understand nested loops. Res. Pract. Technol. Enhanced Learn. (RPTEL) **11**(1), 1–22 (2016). doi:10.1186/s41039-016-0027-3
13. Tang, M.: Caesar: a social code review tool for programming education, Doctoral dissertation, Massachusetts Institute of Technology (2011)
14. Busjahn, T., Schulte, C.: The use of code reading in teaching programming. In: Proceedings of the 13th Koli Calling International Conference on Computing Education Research, pp. 3–11 (2013)
15. Lopez, M.: Relationships between reading, tracing and writing skills in introductory programming. In: Proceedings of the Fourth international Workshop on Computing Education Research, pp. 101–112 (2008)