Development of a Computer Programming Learning Support System Based on Reading Computer Program

Haruki Kanamori[1,*], Takahito Tomoto[2], and Takako Akakura[2]

[1] Graduate School of Engineering, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan
kanamori-haruki@ms.kagu.tus.ac.jp
[2] Faculty of Engineering, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan
{tomoto,akakura}@ms.kagu.tus.ac.jp

Abstract. In this paper, we describe the development of a support system that facilitates the process of learning computer programming through the reading of computer program. Reading code consists of two steps: reading comprehension and meaning deduction. In this study, we developed a tool that supports the deduction of a program's meaning. The tool is equipped with an error visualization function that illustrates a learner's mistakes and makes them aware of their errors. We conducted experiments using the learning support tool and confirmed that the system is effective.

Keywords: programming learning, flowchart, error-based simulation.

1 Introduction

This paper describes the development of a support system that facilitates the process of learning computer programming through the reading of computer program. In this study, we define reading source code as working backward from the code to determine the original requirement that led to the program. The process of reading code consists of two steps: reading comprehension and meaning deduction (see Fig. 1).

Information technology has spread throughout society, but there is a shortage of information engineers, and it is to train them in great numbers. There is extensive research on learning computer programming through the construction of computer programs [1]. However, gaining a deep understanding of programming requires learners to read source code as well [2].

Programming experts are highly skilled at reading code since this skill is essential in debugging programs and inferring their purpose [3]. Reading code is also important to gain a deeper understanding of programming. Furthermore, posing problems is often useful in understanding the scope of a computer program [4]. Accordingly, we

[*] Corresponding author.

developed a support system that facilitates the process of learning programming through reading code.

2 The Process of Programming

In previous research, the process of programming has been considered to consist of two steps: algorithm design and coding. Algorithm design is the step in which structures, such as flow diagrams, are used to construct the abstract process based on the program's requirements. This processing flow is independent of the programming language. In contrast, coding is the step in which the abstract flow is converted into source code, which necessarily depends on the programming language. In learning programming, learners are often given problems as requirements, and write the appropriate source code by first considering the abstract processing flow.

We consider reading code to be an important skill that adds to the process of programming. In this study, we propose that the process of reading code consists of two steps: reading comprehension and meaning deduction (see Fig. 1). Reading comprehension is the inverse step of coding, and meaning deduction is the inverse step of algorithm design. In reading comprehension, learners are required to convert source code into an equivalent abstract processing flow. In meaning deduction, learners are required to deduce a requirement from the abstract processing flow.

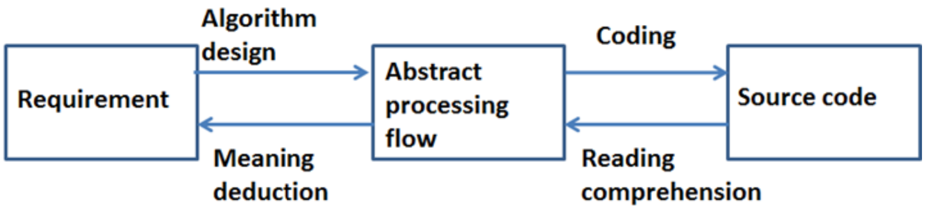

Fig. 1. The process of programming

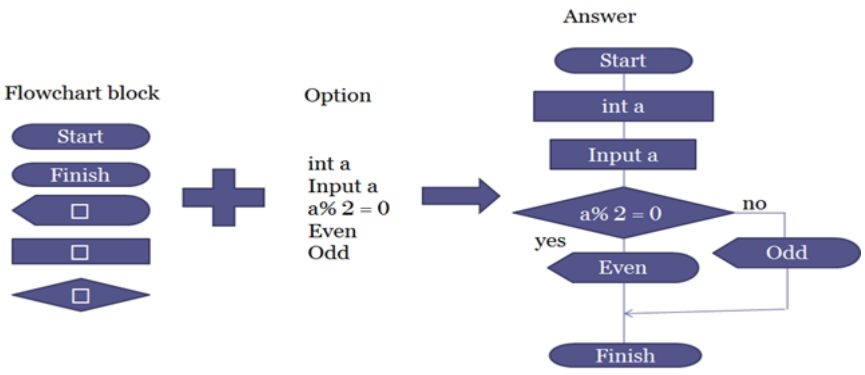

Fig. 2. How to write a flowchart

3 Learning Using a Flowchart

At the reading comprehension step, learners construct flowcharts from given pieces of source code. A flowchart has the advantage of making a problem (requirement) more likely to be discovered by representing it visually. Figure 2 shows the process of constructing a flowchart. A learner chooses a series of flowchart blocks and populates each block with one of several given options. Next, they connect the flowchart blocks with lines. By reducing the degrees of freedom of the answer, it is easier to convey the intent of the program.

4 Deducing Process Requirements

At the meaning deduction step, learners deduce process requirements from flowcharts by choosing statements and concepts from a number of options. By reducing the degrees of freedom of the answer, it is easier to convey the intent of the program.

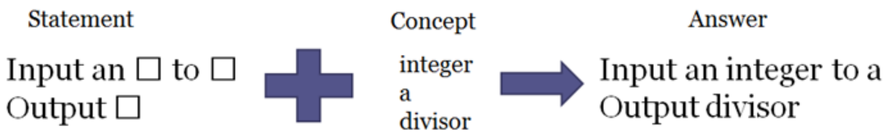

Fig. 3. How to write requirements

5 Error Visualization

Error visualization is the process of illustrating error [5]. Feedback is capable of teaching the correct answer and pointing out errors, but the learner stops thinking if they are simply shown the correct answer. If a learner is only shown their mistakes, they are not able to understand how and why they erred. In contrast, illustrating the error can make the learner aware of their errors. On this basis, we developed a learning support system that includes an error visualization function.

6 Preliminary Experiment

We conducted two experiments with two different objectives: the objective of Experiment 1 was to examine the reading skill level of learners; and the objective of Experiment 2 was to examine the influence of reducing the degrees of freedom of an answer.

6.1 Experiment 1

In Experiment 1, we spent 10 min explaining the principles of writing a flowchart to 62 second-year university students attending a programming course. The students were asked to solve four reading comprehension problems in 20 min, four algorithm design problems in another 20 min, and four coding problems in a final 20 min. Problems were given in a free-response format, and the maximum score for each problem was 2 points.

Table 1 shows the results of Experiment 1. The average score was 1.20 for the reading comprehension exercise, 1.21 for the algorithm design exercise, and 1.69 for the coding exercise. From these results, we can conclude that reading comprehension and algorithm design were difficult. Although algorithm design is often considered to be more difficult than coding, reading comprehension was found to be as difficult as algorithm design.

Table 1. Experiment 1 Results

	Average score	Standard deviation
Algorithm design	1.21	0.52
Coding	1.69	0.39
Reading comprehension	1.20	0.38

6.2 Experiment 2

In Experiment 2, we took 10 min to explain the principles of writing a flowchart to 12 fourth-year university students. After the explanation, the students were asked to solve six reading comprehension problems in 30 minutes followed by six meaning deduction problems in 15 minutes. How to answer is proposed in sections 3 and 4. The maximum score for each problem was 2 points.

Table 2 shows the results of Experiment 2. The average score was 1.21 for the reading comprehension exercise and 0.64 for the meaning deduction exercise. From these results, we can conclude that the effect of reducing the degrees of freedom of the answer was small, and that meaning deduction was a difficult task. From Experiment 1 and Experiment 2 we confirmed the need to develop a support system that facilitates the process of learning programming through reading code.

Table 2. Experiment 2 Results

	Average score	Standard deviation
Reading comprehension	1.21	0.55
Meaning deduction	0.64	0.21

7 Learning Support System

7.1 Learning Screen

Figure 4 shows the learning screen of the learning support system. The learner uses concept and statement buttons to construct a problem statement. First, a student presses a statement button, which brings the statement with blank to the answer column. Next, a student presses a concept buttons and select blank, which inserts the concept into the blank. When the learner has completed an answer, he or she presses the answer button. If the answer is correct, a message of "Correct answer" is displayed; if the answer is incorrect, the system shows the feedback screen.

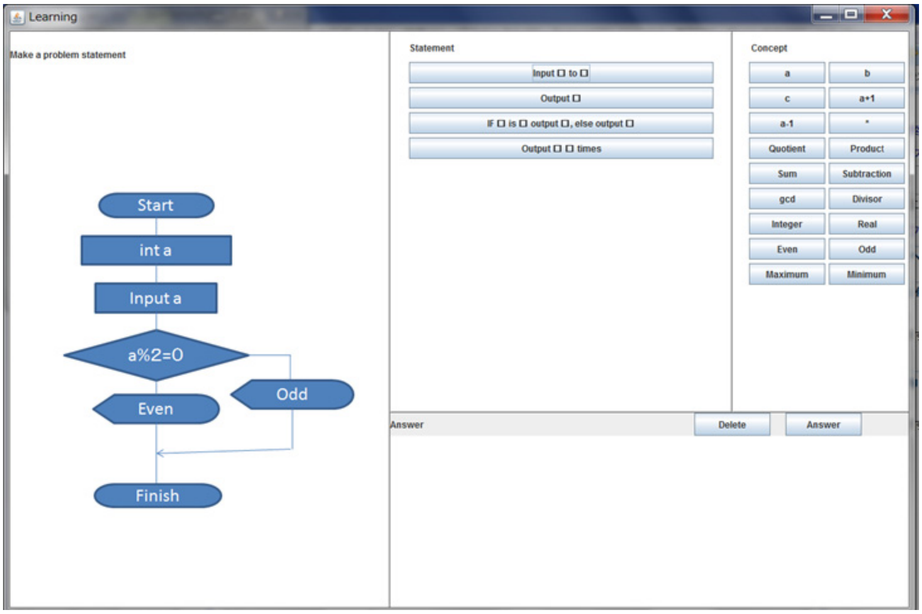

Fig. 4. Learning screen

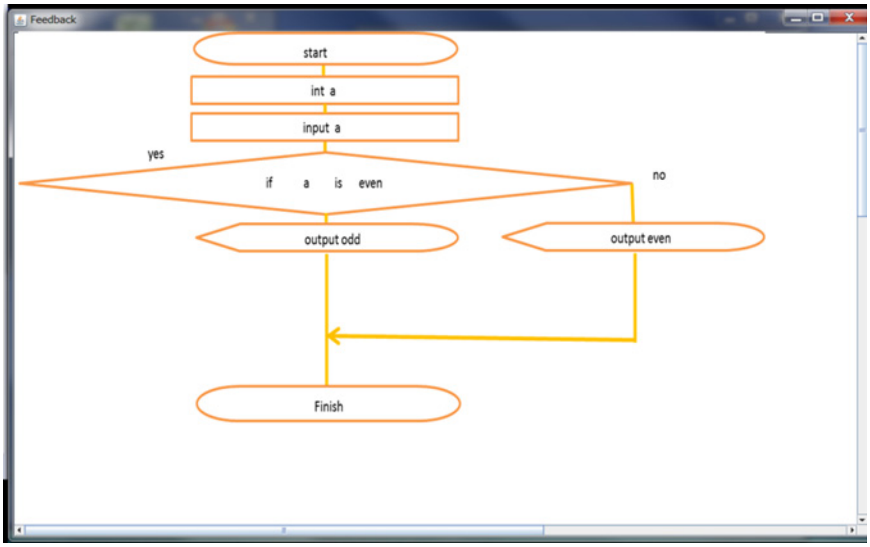

Fig. 5. Feedback screen

7.2 Feedback Screen

Figure 5 shows the feedback screen. If a learner incorrectly deduces a requirement, the system generates an incorrect flowchart based on the incorrect data, and the learner looks for their mistakes by comparing the incorrect flowchart to the correct flowchart.

8 Assessment Experiment

To ascertain the usefulness of the learning support system, we conducted an assessment experiment. In the assessment experiment, we first administered a pre-test for all participants (12 fourth-year university students). In the pre-test, after explaining the principles of writing a flowchart for 10 min, the participants were asked to solve six meaning deduction problems in 15 min. The maximum score for each problem was 2 points. Next, the participants were divided into three groups: an experimental group (4 students), control group 1 (4 students), and control group 2 (4 students). We spent 5 min explaining to the experimental group how to use the system, followed by a period of 30 min in which the group learned meaning deduction using our system. Next, the participants were asked to solve 10 meaning deduction problems in 30 min. In control group 1, the participants were asked to solve five algorithm design problems in 15 minutes, followed by studying algorithm design problems by viewing the correct answer. Finally, the participants were asked to solve 10 meaning deduction problems in 30 minutes. In control group 2, the participants were first asked to solve five meaning deduction problems in 15 min, followed by studying meaning deduction problems by viewing the correct answer. Finally, the participants were asked to solve 10 meaning deduction problems in 30 min. The maximum score for each problem was 2 points.

Table 3 shows the results of assessment experiment. For control group 1, the average post-test score was 1.33 and the average pre-test score was 0.50. This result shows that supporting meaning deduction learning is beneficial. The average post-test score was 1.53 for control group 2 and 1.50 for the experimental group. However, the difference between the average pre-test score and the average post-test score was 1.00 for the experimental group and 0.90 for control group 2. From this result, we confirmed that our system is effective.

Table 3. Assesment Experiment Results

	Pre-test	Post-test	Difference (post-test minus pre-test)
Experimental group	0.50	1.50	1.00
Control group 1	0.79	1.33	0.53
Control group 2	0.63	1.53	0.90

9 Conclusions and Future Work

In this study, we developed a learning support system to provide guidance in meaning deduction, and evaluated the effectiveness of our system. From the results of the assesment experiment, we confirmed that it is necessary to support meaning deduction learning, and that our system is effective. However, the assessment experiment did not include enough participants, and it is necessary to increase the number of participants in future experiments. Additionally, we did not develop a learning support system for guidance in reading comprehension, but believe it is necessary to develop one in the future.

References

1. Matsuda, N., Kashihara, A., Fukukawa, K., Toyoda, J.: An instructional system for constructing algorithms in recursive programming. In: Proc. of the Sixth International Conference on Human-Computer Interaction, Tokyo, Japan, pp. 889–894 (1995)
2. Corbi, T.A.: Program understanding challenge for the 1990s. IBM Syst. J. 28(2), 294–306 (1989)
3. Uchida, S., Kudo, H., Monden, A.: An experiment and an Analysis of debugging process with periodic interviews. In: Proceedings of Software Symposium 1998, Japanese, pp. 53–58 (1998)
4. Lyn, D.: Children's Problem Posing within Formal and Informal Contexts. Journal of Research in Mathematics Education 29(1), 83–106 (1998)
5. Hirashima, T.: Error-based simulation for error-visualization and its management. Int. J. of Artificial Intelligence in Education 9(1-2), 17–31 (1998)