Asian Journal of Research in Education and Social Sciences
e-ISSN: 2682-8502 | Vol. 3, No. 1, 10-24, 2021
http://myjms.mohe.gov.my/index.php/ajress

A Trial of Learning Programming Using a Six-step Method

Tatsuhiro Tamaki[1], Atsushi Onishi[2], Yasuo Uchida[3*]

[1] Department of Media Information Engineering, National Institute of Technology, Okinawa College, Nago, Japan
[2] Department of Integrated Science and Technology, National Institute of Technology, Tsuyama College, Tsuyama, Japan
[3] Department of Business Administration, National Institute of Technology, Ube College, Ube, Japan

*Corresponding Author: uchida@ube-k.ac.jp

Accepted: 15 February 2021 | Published: 1 March 2021

Abstract: *CS (Computer Science) unplugged is an educational method, highly effective for learning, in which students learn about information science through experiences that do not involve the use of computers. A wide range of research is being conducted in this area. However, much of the previous research has not gone beyond experiencing CS unplugged, or has been limited in terms of connections to full-fledged programming education. Thus, we have incorporated a method called CS plugged as a new approach for complementing CS unplugged, and proposed a method for learning programming comprising six steps. In CS unplugged, the basic idea is to not use computers, but our view is that education techniques using computers are essential as a bridge to full-fledged programming. Our aim is to establish a learning method made up of the following six steps: (1) Step 1: CS unplugged, (2) Step 2: CS plugged, (3) Step 3: Diagramming of processing for an activity (visualization), (4) Step 4: Natural language description of processing for the activity (abstraction, element extraction), (5) Step 5: Confirming operation of the algorithm using a trace table to which original expressions have been added (verification), and (6) Step 6: Writing full-fledged program code (abstraction, coding). This research used the six-step method in a C language programming class for young adults at a level corresponding to second-year high school students. This paper provides an overview of the research.*

Keywords: six-step method, learning programming, CS unplugged, CS plugged

1. Introduction

CS unplugged (Bell et al., 1999) is garnering attention as an educational technique promoting fundamental understanding of information science, and active research is under way in this area. CS unplugged is a method for teaching information science without using a computer, advocated by Tim Bell et al. of the University of Canterbury in New Zealand.

However, much of the previous research has not gone beyond experiencing CS unplugged, or has been limited in terms of connections to full-fledged programming education.

In the classes we teach on a day-to-day basis, a considerable percentage of students feel they are not up to programming. Based on findings such as past student questionnaires, there are thought to be three main obstacles (Tamaki et al., 2016). One is the process of devising algorithms, and another is abstraction in mapping to a programming language. In addition, there is acquiring an image of the operation of a program when these two are integrated. The primary methods for dealing with these issues are: CS unplugged for devising algorithms,

natural language description of processing for abstraction, and CS plugged for acquiring an image.

We propose the CS plugged method as a new approach to complement CS unplugged. We use "CS plugged" as an independently-defined term to mean carrying out the activity of CS unplugged using a computer. That is, the basic idea of CS unplugged is to "not use a computer," but if the aim is to progress to full-fledged programming, our view is that education techniques using computers are essential to serve as a bridge.

We have incorporated a method called CS plugged as a new approach for complementing CS unplugged, and proposed a method for learning programming comprising six steps. Our aim is to establish an educational method for progressing from CS unplugged to full-fledged programming.

2. Background

A great deal of research is being carried out on CS unplugged. In the beginning, an instruction book on how to use CS unplugged with children at the elementary school level was written as part of research by Tim Bell (the originator of CS unplugged) and his collaborators.

Subsequent research can be classified into a number of approaches. These include: (a) Analysis and research on the educational method itself of CS unplugged (including generalization of unplugged), (b) Research on implementing CS unplugged and realizing its educational effects, (c) Research to devise new activities for CS unplugged, (d) Research on CS plugged tools, (e) Research combining CS unplugged with other educational methods, and (f) Research on progressing from CS unplugged to full-fledged programming (the subject of this research).

From another perspective, students can be divided by age group into elementary school and below, and junior/senior high school and above. For progressing to programming after CS unplugged, visual languages such as Scratch are used with the younger group, and full-fledged languages such as Java or Python are used with the older group.

The latest research trends in Japan include: research linked with Scratch, research on implementation and evaluation of instruction programs, and research on evaluation criteria for systematic programming education in primary and secondary education. Many efforts are being made in these areas.

In terms of the latest research trends overseas, the group led by Tim Bell is conducting research from the standpoint of design and narrative, research looking at computational thinking, and research on curriculum design. Many studies are being carried out by other researchers, such as research focusing on algorithms, research looking at coding, and research analyzing the relationship with computational thinking.

However, most research inside and outside Japan is focused on implementation and evaluation of CS unplugged, and its applications. Therefore, our intention is to pioneer a new research field which aims to go beyond understanding of information science, and serve as a bridge to learning full-fledged programming.

3. Literature Review

It is said that programming is difficult for beginners. So, first, I will give an overview of the discussion about barriers for beginners to learn programming.

Hofuku (2013) raises the question, "Why is programming difficult?" In high school classes, he points out "repetition" as one of the places where beginners can easily stumble. Then, it was clarified that the repetition without variables is easy to understand, but the repetition with variables is difficult to understand why beginners find it difficult. Based on these results, the chiefs developed a tool to support the step-by-step understanding of programming beginners. We also showed that it is effective in analyzing the leap of teaching materials that can contribute to the stumbling of beginners. It can be said that this confirms that when a beginner learns a new concept, the idea of small steps that gradually incorporates new things is effective.

Komatsu (2015) discussed the problems and countermeasures of programming education in programming classes at universities. We sort out and analyze the problems of conventional programming education and propose a new programming teaching method. It shows that it is a programming teaching method that fuels the learner's positive emotions. In other words, we are trying to make people interested by using the game as a subject, and to make them understand immediately at the line level by explaining while inputting the program code in real time. According to the proposed method, it is stated that the problem of dropouts that frequently occurs in programming education is unlikely to occur. However, as a limitation of this teaching method, it is assumed that the number of participants is up to about 20.

Matsubara (1986) made an analysis and proposal from a cognitive science perspective. First, as a problem, he points out that although programming has the peculiar property of objectively observing the human thinking process, an effective learning method or teaching method for programming has not been established. Then, it is cognitive science to consider variables, arrays, and control structures, which are important concepts of programming, and to use the thinking framework that students have already built in their minds and casually use in their daily lives. I conclude that it is the easiest way to teach.

Next, I will give an overview of research on CS unplugged.

CS unplugged is computer science unplugged, which is a teaching method for teaching information science without using a computer, which was advocated by Tim Bell et al. Of the University of Canterbury in New Zealand. In CS Unplugged, students learn computer science topics through group work and hands-on activities using teaching tools such as worksheets. CS unplugged learning items (topics) are called activities, and various subjects such as binary numbers, image representation, and text compression are prepared.

Idosaka et al. (2008a) report on practical efforts for elementary school students. This is a practice in which CS unplugged lessons were given to elementary school students at local government events. As a result of selecting and improving some teaching materials so that elementary school students can understand them, we have confirmed that these contents can be fully understood and effective even by elementary school students. Since it is for elementary school students, we have succeeded in expanding children's interests and improving their communication skills by giving lessons mainly on games.

Idosaka et al. (2008b) proposed the design of lessons using CS unplugged in "Technology / Home" in junior high school. There, although it is a short time, we will pick up five activities selected from the viewpoint of junior high school students, those with strong elements to learn the mechanism and principle of computers, and those with learning expertise that are easy for junior high school students to learn. I practiced it. He also reported that he enjoyed and actively worked on the learning content even though he felt it was difficult, and that he became more interested in computers.

Hofuku et al. (2008) report on lesson practice in the high school subject "Information B". Information B requires an understanding of how information is represented and the processing mechanism of a computer. Therefore, we took up binary numbers / character codes, image representation, text compression, etc. and put them into practice. As a result, it was shown that the degree of comprehension and self-evaluation were improved compared to the lessons using only textbooks up to the previous year.

Wada (2009) worked on an algorithm class that used both CS unplugged and board writing lectures at the university. We propose a lesson plan that leads to a more accurate and deep understanding by combining lectures on the blackboard with the aim of making it suitable for university lessons.

As mentioned above, there are many lesson practices that incorporate CS unplugged, but there are not many efforts that lead to programming introductory education.

Among them, the work of Manabe et al. (2013) is noteworthy. There, we focused on the form of "group learning using a balance", which is handled in the original CS unplugged, based on the learning of sorting algorithms that have not been sufficiently verified until now, and it takes less time than the balance. I am trying to use a "simulated balance on the screen" that can be practiced at. In addition, the learning effect of incorporating "individual learning", which allows students to concentrate more on practical training, is evaluated, including classroom lessons that do not use CS unplugged. As a result, it was clarified that in the practice of trying and fixing the learned algorithm by oneself, the degree of understanding of individual learning is higher than that of group learning, especially when using a simulated balance on the screen. did.

Finally, we give an overview of the discussion of computational thinking and programming thinking.

Ota et al. (2016) conducted a survey on information education curriculum including programming education in other countries. As a result, each country aims to develop abilities such as abstraction, problem analysis, algorithms, data utilization, evaluation, and collaborative work, centering on the concept of computational thinking that includes programming education as information education. It was clarified that the learning content was defined. In addition, programming education has similar contents, in the lower grades of elementary school, instructions are given using robots and puzzles, in the upper grades of elementary school, a program including branching and repetition is created using visual language, and in junior high school and high school, a text language is used. It states that it is a flow to develop a program including multiple data types and modules using.

Computational thinking is also translated as computational thinking, and Wing (2006) brought the term to the forefront. Wing states that "thinking like a computer scientist means more than

being able to program a computer. It requires multiple levels of abstract thinking," but it is clear in the essay. No definition has been made. Hayashi (2018) is conducting a survey on the concept of Computational Thinking (CT).

The term "programming thinking" used in Japan was discussed in the Ministry of Education, Culture, Sports, Science and Technology (2016), and "a series of activities that I intended. What kind of combination of movements is necessary to realize it, how to combine the symbols corresponding to each movement, and how to improve the combination of symbols, the more intended activity. It is defined as "the ability to logically think about whether or not to approach."

4. Objectives

The objective of this research is to establish a universal educational method for effective programming education at the introductory stage. CS unplugged is thought to be effective for information science education, and related research includes Manabe et al. (2012) on using teaching aids for learning algorithms in CS unplugged, and Feaster et al. (2011) on implementation for high school students. In the latter case, success has been reported, although the results are limited. However, at present almost no research is being done on progression from CS unplugged to full-fledged programming languages. Therefore our objective is to develop new techniques for progressing from CS unplugged to full-fledged programming. We aim to establish an instructional method comprising the following six steps: (1) Step 1: CS unplugged, (2) Step 2: CS plugged, (3) Step 3: Diagramming of processing for an activity (visualization), (4) Step 4: Natural language description of processing for the activity (abstraction, element extraction), (5) Step 5: Confirming operation of the algorithm using a trace table to which original expressions have been added (verification), and (6) Step 6: Writing full-fledged program code (abstraction, coding).

5. Methodology

We implemented the six-step method in class, incorporating two newly created unplugged activities, and afterward administered an anonymous questionnaire.

- Objective: Conduct a programming class using the six-step method, and obtain questionnaire evaluations of its effectiveness.
- Subjects: Second-year students in the Media Information Engineering Department of the National Institute of Technology, Okinawa College, in the 2019 academic year (valid responses received from 37 students)
- Programming learning experience of subjects: Second-year students learn C language programming in a 90-minute weekly class starting in April. They have no experience learning programming prior to that. They had learned topics such as conditional decisions, iteration, arrays, and pointers by the date of class implementation.
- Implementation date and time: Jan. 24, 2020, 14:40–16:10 (90 mins.)
- Overview of implementation: 1) Explanation of problem by the teacher, 2) Implementation of Step 1, 3) Implementation of Step 2, 4) Implementation of Step 3, 5) Implementation of Step 4, 6) Implementation of Step 5, 7) Implementation of Step 6, 8) Administration of questionnaire

Unplugged Activity

We devised the following new unplugged activity.

(1) Theme

Creation of contact network

(2) Overview

Students develop understanding of the concept of pointers by comparing with a contact network. They also consider a model (data structure) for realizing a contact network.

(3) Relationship with subject learning

- [Pointers] Relationship between pointers and arrays.

(4) Skills

- Diagram the model (contact network).
- Discover relationships and regularities between data items.

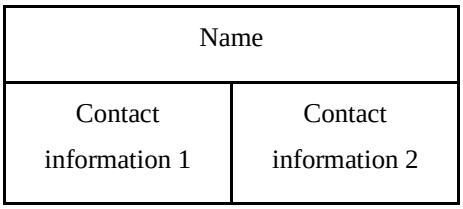
(5) Instructional materials

- Contact network diagramming worksheet (Figure 1)
- Conceptual diagram of array (Figure 2)

(6) Basic principles

The teacher T wants to create a contact network for the class.

- Number of students in the class is assumed to be 26.
- Student names are indicated with letters of the alphabet A–Z. The slash "/" is used as a symbol indicating the teacher.
- It is assumed that contact starts from the teacher and each person contacts two students.
- The elements of the contact network are diagrammed as follows.



In this notation, the contact information (usually a phone number, etc.) is assumed to be a pointer (address) in memory where the name of each person is stored. For example, the contact information for student A is expressed by prefixing the name with &, like this: &A.

Therefore, the teacher has the information for "&A" and "&B" as contact information for contacting student A and student B. Student A has "&C" an "&D" as contact information for the next two students. For students who have no further contact information, the symbol "\0" is designated as data indicating that there is no contact information for the next person.

(Example with data indicated)

Step 1: Programming Unplugged

In Step 1, each student was asked to work alone on the activity described in the previous section: creating a contact network.

Due to the importance of free idea generation by each person, students were asked to first carry out the following task:

- Each student produces a simple diagram of their image of a contact network (drawing a rough sketch adequate for explaining to the neighboring student).

Next, they were each asked to do the following task to compare their idea with the idea conceived by the teacher. It was also explained that the content of this worksheet is one example of creating a network.

- Students are asked to fill in the blanks in the contact network diagramming worksheet in Figure 1.

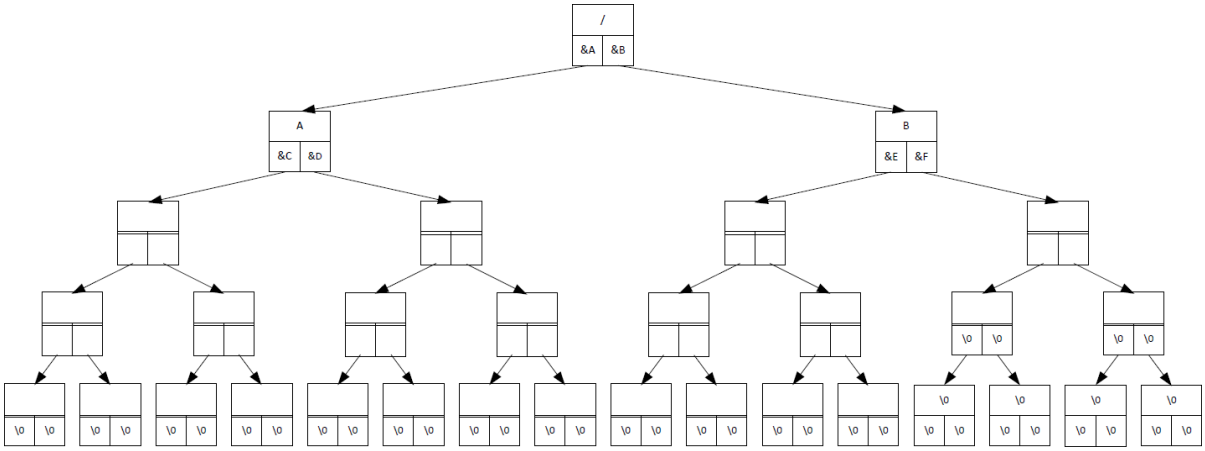


Figure 1: Contact Network Diagramming Worksheet (Problem Version)

Furthermore, each student was asked to devise a data structure as preparation to serve as a bridge to programming.

- Each student produces a simple diagram of their image of how to realize the data in the contact network as an array (drawing a rough sketch adequate for explaining to the neighboring student).

At the end of this step, an explanation was provided by presenting Figure 2. The purpose was to enable comparison between the idea of each student and the idea created as an example by the teacher.

Step 2: Programming Plugged

In the C language, the locations where data is stored are clearly indicated as addresses, and it was decided to have students acquire an image by executing the program in Listing 1 in order to understand the relationships between data items. It was also explained that this conceptual diagram is one example of creating an array.

- Students fill in the blank spaces (addresses) in the conceptual diagram of an array in Figure 2. This is done by executing a program (List 1) written in the C language.

Index	Address	Name	Contact information 1	Contact information 2
0		/	&A	&B
1		A	&C	&D
2		B	&E	&F
3		C	&G	&H
4		D	&I	&J
5		E	&K	&L
6		F	&M	&N
7		G	&O	&P
8		H	&Q	&R
9		I	&S	&T
10		J	&U	&V
11		K	&W	&X
12		L	&Y	&Z
13		M	¥0	¥0

Figure 2: Conceptual Diagram of Array

List 1

```
#include <stdio.h>

int main(void)
{
	int i = 0;
	char name[] = "/ABCDEFGHIJKLMNOPQRSTUVWXYZ";
	int n = 27;

	for (i = 0; i < n; i++) {
		printf("%p : %c\n", &name[i], name[i]);
	}

	return(0);
}
```

Step 3: Diagramming of Processing for the Activity (Visualization)

The diagram of processing (visualization) was explained (Figure 3) to clarify how processing should proceed in order to solve the problem.

Step 4: Natural Language Description of Processing for the Activity (Abstraction, Element Extraction)

This is the stage where students are asked to provide a natural language description (abstraction, element extraction) of the processing in order to carry out algorithm design. Therefore, students were first asked to find regularities in processing (Figure 4).

However, in this implementation, the teacher only explained the gist due to time constraints.

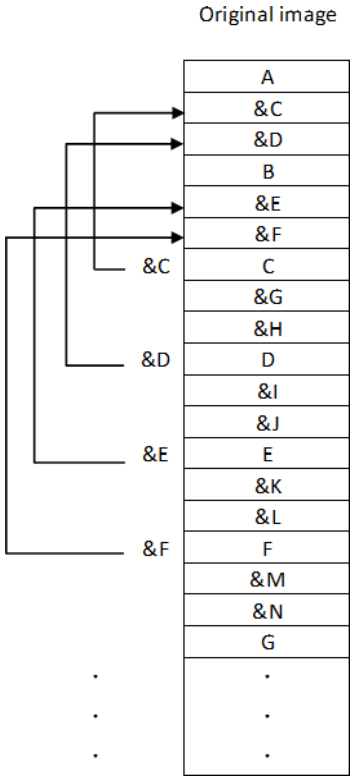


Figure 3: Diagramming of Processing

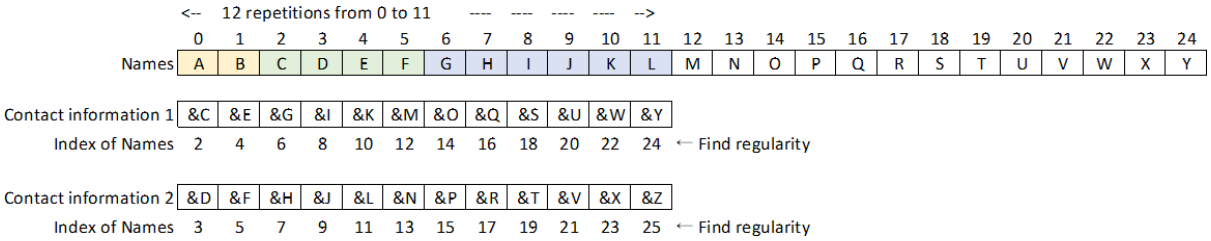


Figure 4: Discovery of Processing Regularities

Description in Natural Language (Japanese)

- Create a character array for "Names" with 27 elements: / and A–Z.
- Create pointer array for "Contact information 1" having 27 elements storing the characters "\0".
- Create pointer array for "Contact information 2"' having 27 elements storing the characters "\0".
- Repeatedly perform the following processing for the 13 people / and A–L which have the following contacts information.
- Substitute the respective addresses to be contact information in the respective arrays for Contact information 1 and Contact information 2 corresponding to the diagram above.
- Display the student names (/ and A–L), Contact information 1, and Contact information 2 in a list as follows.

```
/ --> A , B
A --> C , D
B --> E , F
C --> G , H
D --> I , J
E --> K , L
F --> M , N
G --> O , P
H --> Q , R
I --> S , T
J --> U , V
K --> W , X
L --> Y , Z
```

Step 5: Confirming Operation of the Algorithm Using a Trace Table (Verification)

In this stage, students use a trace table to confirm, with pencil and paper, whether the devised algorithm operates properly. However, in this implementation, the teacher only explained the gist due to time constraints (Figure 5).

Number of people
12

Names[0]	Names[1]	Names[2]	Names[3]	Names[4]	Names[5]	…	
A	B	C	D	E	F	…	

Repeat count (Index)	Name to put in Contact information 1 (Array index)	Contact information 1[0]	Contact information 1[1]	Contact information 1[2]	Contact information 1[3]	…	Name to put in Contact information 2 (Array index)	Contact information 2[0]	Contact information 2[1]	Contact information 2[2]	Contact information 2[3]	…
0	2	&Names[2]					3	&Names[3]				
1	4		&Names[4]				5		&Names[5]			
2	6			&Names[6]			7			&Names[7]		
3	8				&Names[8]		9				&Names[9]	
.	.					.	.					.
.	.					.	.					.
.	.					.	.					.

Figure 5: Trace Table

Step 6: Writing Full-fledged Program Code (Abstraction, Coding)

Program code is written as the final stage (List 2). However, in this implementation, a fill-in-the-blanks format was adopted due to time constraints.

List 2

```
#include <stdio.h>

int main(void)
{
        int i = 0;
        char name[] = "/ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        char *next1[] = {'\0', '\0', '\0', '\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0',
'\0','\0', '\0','\0', '\0'};
        char *next2[] = {'\0', '\0', '\0', '\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0', '\0','\0',
'\0','\0', '\0','\0', '\0'};

        int max = 13; // Numbers we counted : /～L
```

```
        i = 0;
        while (i < max)
        {
                next1[i] = &name[2 * i + 1];
                next2[i] = &name[2 * (i + 1)];
                i++;
        }

        for (i = 0; i < max; i++) {
                printf("%c --> ", name[i]);
                printf("%c , ", *next1[i]);
                printf("%c\n", *next2[i]);
        }

        return(0);
}
```

6. Result and Discussion

Questions on the Questionnaire

The questionnaire asked the following six questions:

Q1. Do you think the learning method in "Step 1: Programming unplugged" (task of diagramming problem content) is useful for learning programming?
Q2. Do you think the learning method in "Step 2: Programming plugged" (task of confirming by executing a program that serves as a hint) is useful for learning programming?
Q3. Do you think the learning method in "Step 3: Diagramming of processing for an activity" (task of visualizing processing) is useful for learning programming?
Q4. Do you think the learning method in "Step 4: Natural language description of processing for the activity" (tasks such as abstraction and element extraction) is useful for learning programming?
Q5. Do you think the learning method in "Step 5: Trace table" (task of confirming (verifying) operation of the algorithm) is useful for learning programming?
Q6. Overall, do you think the learning activity using the "six-step method" is useful for learning programming?

Subjects responded to each question with one of four possible answers: I think so, I think so somewhat, I don't really think so, and I don't think so. In addition, a free comment space was provided for each item, with the instructions "If possible, please describe the reason for your choice."

Questionnaire Results

Table 1 shows the results of the multiple-choice questionnaire.
The results of Q1 to Q6, were divided, respectively, into positive responses and negative responses, and a population proportion test was carried out. The results showed that $P < .01$ for all questions, and responses were found to be generally favorable.

Table 1: Questionnaire Results

Response	Number of responses	Percentage of responses
Q1 I think so	22	60%
I think so somewhat	12	32%
I don't really think so	2	5%
I don't think so	1	3%
Q2 I think so	20	54%
I think so somewhat	15	41%
I don't really think so	2	5%
I don't think so	0	0%
Q3 I think so	22	59%
I think so somewhat	14	38%
I don't really think so	1	3%
I don't think so	0	0%
Q4 I think so	16	46%
I think so somewhat	16	46%
I don't really think so	2	5%
I don't think so	1	3%
Q5 I think so	20	57%
I think so somewhat	13	37%
I don't really think so	1	3%
I don't think so	1	3%
Q6 I think so	23	65%
I think so somewhat	9	26%
I don't really think so	2	6%
I don't think so	1	3%

The free comment responses are indicated below. In parentheses are the number of responses including those with the same meaning.

- Q1 response results

(Favorable responses)

・Diagramming made it easier to think about (12)

・I felt it was easier to write a program (6)

・It looks like it will reduce failures (errors, etc.)

(Unfavorable responses)

・It was too difficult, and I didn't understand it well

- Q2 response results

(Favorable responses)

・I think it's good because applications can be done if the basics are understood

・The structure of programming is easier to understand when given a hint (9)

(Unfavorable responses)

・This is fine if one is only considering an algorithm, but I think it's unsuitable for learning how to put everything together

・Failure may sometimes lead to learning

・Considering only the content in this case, I thought the explanation was inadequate to impart understanding to a person who did not understand very well in the first place

- Q3 response results

(Favorable responses)

・The mechanism is better understood by seeing the processing (9)

・ I felt it was easier to write a program

(Unfavorable responses)

None

- Q4 response results

(Favorable responses)

・I think it's easier to understand when expressed in words that I myself often use (4)

(Unfavorable responses)

・I don't even understand the natural language description.

・It's hard to get an image

- Q5 response results

(Favorable responses)

・It's easier to understand the mechanism of the algorithm (3)

・I thought verification is good

(Unfavorable responses)

・In this class, this step wasn't done, and it was difficult to determine its effectiveness

・I couldn't understand the algorithm

- Q6 response results

(Favorable responses)

・This will broaden my methods when writing other programs

・As a result of the entire exercise, I was able to understand the mechanism of the program (5)

・It was good because I was able to go through the process of forming an image, and then writing code

(Unfavorable responses)

・It was too difficult, and I didn't understand it well

・I understand what we're doing, but it's difficult to see what part to incorporate into the program and how to do it

・The usual approach is better

Discussion of Questionnaire Results

Regarding Q1, 92% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. In the free comment too, 12 students indicated that it "deepens understanding." For the most part, it seems that use of CS unplugged was seen as effective at the problem understanding stage.

Regarding Q2, 95% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. Nine students indicated that "the structure of programming is easier to understand when given a hint," and this showed that the intent of CS plugged was understood. Among the unfavorable responses was the suggestive response that "failure may sometimes lead to learning," and this showed the need to improve procedures, including debugging.

Regarding Q3, 97% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. Nine students responded that "the mechanism is better understood by seeing the processing," and this seems to be regarded as an indispensable step for understanding processing.

Regarding Q4, 92% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. Four students responded that "I think it's easier to understand when expressed in words that I myself often use," and it seems they felt that approaches such as abstraction and element extraction through natural language description are effective.

Regarding Q5, 94% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. Three students responded that "it's easier to understand the mechanism of the algorithm," and it appears that the necessity of the task of confirming operation of the algorithm (verification) was evaluated highly.

Regarding Q6, 91% of the students (combining "I think so" and "I think so somewhat") indicated a favorable response. Five students responded that "as a result of the entire exercise, I was able to understand the mechanism of the program." Overall, they felt the learning activity using the "six-step method" was useful for learning programming, and their view of the method was generally positive.

Discussion of Implementation Results

The sorting problem requires processing with the proper linkage of a data structure (array) and algorithm, and this can be regarded as a barrier for programming beginners.

In this implementation, learners indicated a generally positive evaluation of the use of CS plugged employing a simple sorting teaching aid. The CS plugged tool employed only a slow display using timer processing during execution, and incorporating interactive features such as step display may help improve understanding. In this case, the format of the trace table was provided beforehand, but there is a need to perform this step from the beginning (i.e., devising the format). It is thought that description of processing in natural language (Japanese) was too vague in terms of what was required, and there is a need to consider indicating the general framework of the content to be described.

In this case, Java source code was not written, and all that was done was show a program example. Therefore, it is difficult to determine whether steps extending this far link up with the next step of writing source code.

7. Conclusion

At a college of technology, a class was implemented using the six-step method for an introductory course of programming education for second year students (who are at the level of the second year of high school). The questionnaire results revealed a generally positive evaluation of the use of these techniques at the introductory stage. However, in this implementation, there were constraints on class time, and thus for Step 4: Natural language description of processing for the activity, and Step 5: Table trace frequently used when learning programming, it was impossible to spend adequate time on some points, and the teacher only suggested the explanation and results. Also, source code was written using a fill-in-the-blanks format.

As issues for the future, there will be a need to formulate learning scenarios which guarantee close linkage from Step 1 to Step 5. We will also need to improve methods up to writing source code and debugging, which are the final steps of programming.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 19K03104.

References

Bell, T., Witten, I.H., & Fellows, M. (1999). Computer Science Unplugged: Off-Line Activities and Games for All Ages (Original Book). http://csunplugged.org

Feaster, Y., Segars, L., Wahba, S. K., & Hallstrom, J. O. (2011). Teaching CS unplugged in the high school (with limited success). In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, 248–252.

Hayashi, K. (2018). A Review of the State of Computational Thinking Discourse, Research report of JET Conferences, 18(2), 165-172.

Hofuku, Y., Idosaka, Y., Kanemune, S., & Kuno, Y. (2008). Using CS Unplugged in High school Information-B Classes, Proceedings of the Summer Symposium, 2008, 201-206.

Hofuku, Y. (2013). "Peta-gogy" for Future : Why is Programming Difficult?, IPSJ Magazine, 54, 252-255.

Idosaka, Y., Aoki, H., Kanemune, S., & Kuno, Y. (2008a). A Practical Approach for Elementary Schoolchildren with ”Computer Science Unplugged”, Proceedings of the Summer Symposium, 2008, 25-31.

Idosaka, Y., Kanemune, S., & Kuno, Y. (2008b). Evaluation of ”Computer Science Unplugged” in junior high school, IPSJ SIG Technical Report, 2008-CE-93(7), 163-170.

Komatsu, K. (2015). Problems and Measures of Programming Education, Bunkyo Gakuin University Research Institute Management Review, 25(1), 83-104.

Manabe, H., Kanemune, S., & Namiki, M. (2012). Development and Effect of Digital Materials for CS Unplugged, IPSJ SIG Technical Report, 2012-CE-113(14), 1-9.

Manabe, H., Kanemune, S., & Namiki, M. (2013). Effects of Teaching Tools in CSU Algorithm Education, IPSJ Journal, 54(1), 14-23.

Matsubara, Y. (1986). Cognitive Scientific Study of Programming (1), Information and communication studies, 7, 96-104.

Ministry of Education, Culture, Sports, Science and Technology. (2016). About the Way of Programming Education in the Elementary School Stage, http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/074/siryo/__icsFiles/afieldfile/2016/07/07/1373891_5_1_1.pdf

Ota, G., Morimoto, Y., & Kato, H. (2016). The Comparative Survey of Computer Science and Programming Education for Primary and Secondary Schools in the UK, Australia and USA, Japan Journal of Educational Technology, 40(3), 197-208.

Selby, C. & Woollard, J. (2013). Computational thinking: the developing definition, https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf

Tamaki, T., Tanabe, M., Onishi, A., Sakamoto, M., & Uchida, Y. (2016). From Natural Language to Programming Language: A Stepwise Educational Method for Algorithms, Advances in Education Research, 90, 9-14.

Wada, B. T. (2009). An university's class on algorithms using both Computer Science Unplugged and chalkboard lecture, IPSJ SIG Technical Report, 2009-CE-100(5), 1-7.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.