Enhancing programming logic thinking using analogy mapping

R A Sukamto* and R Megasari
Department of Computer Science Education, Universitas Pendidikan Indonesia, Bandung, Indonesia.

*Corresponding author’s e-mail: rosa.ariani@upi.edu

Abstract. Programming logic thinking is the most important competence for computer science students. However, programming is one of the difficult subject in computer science program. This paper reports our work about enhancing students’ programming logic thinking using Analogy Mapping for basic programming subject. Analogy Mapping is a computer application which converts source code into analogies images. This research used time series evaluation and the result showed that Analogy Mapping can enhance students’ programming logic thinking.

1. Introduction
Computer science major students should have the skills to make computer program. Making computer program needs logic thinking to understand the algorithm sequence in order to solve the given problem. Therefore, logic thinking needs to be enhanced in computer science students. There were many researches about how to enhance students' logic thinking in learning programming, for instance, enhancing logical thinking using cooperative learning among Computer Science students [1], learning programming at the computational thinking level using digital game-play [2], algorithmic and logical thinking development based on programming skills [3], enhance logical thinking using RAPTOR (a visual programming development) [4], teaching programming using Scratch [5]. Therefore, this paper proposed our works using Analogy Mapping as media which converts source code in C programming language into analogies images for enhancing students' logical thinking.

1.1. Programming Logic Thinking
Computational thinking is a particular approach to solving problems, designing systems and understanding human behaviour that represents the fundamental concepts to computing. Computational thinking is a kind of analytical thinking in many fields [6]. Logic in computational thinking is fundamental to understand about how a computer works because everything in a computer is controlled by logic therefore logic is used to reason about program behavior. Reasoning is the process of determining the conclusions while kind of reasoning performed by humans is also known as intelligent reasoning [7]. Logical reasoning helps humans to explain why something happens.

Logic programming is a particular approach to represent programming. In programming, a logic is a kind of language and it has syntax and semantics. Computations are logic inferences while programs are logic formulas of certain form. Inference rules describe correct ways to determine the conclusions [7].
Therefore, the computer science students need to have programming logic thinking for learning programming as the vital subject in their major.

1.2. Analogy Mapping
Analogy Mapping is a computer program which developed using compiler principle for reading source code in C language. This computer program uses CodeMirror library for showing the source code and giving different color for every type of token. This library uses the first process of compiler principle such as

- lexical analysis, which gets every token from source code,
- syntactic analysis, which gives a label for every token,
- semantic analysis, which identifies token as tree and coloring every token.

Analogy Mapping converts sequence of token to be analogies images. This process uses state machine principle [8]. The Analogy Mapping interface is shown in figure 1

![Analogy Mapping Interface](image)

**Figure 1.** Analogies of sequence instruction in program code (tampilkan analogi = show the analogies, halamaniku = my yard, ketika isi halamanmu tak seperti yang diharapkan, periksa lagi kode programnya = if your yard picture not as your expectation, please check your code) [8]

1.3. Minimalism Learning
John Millar Carroll is a scientist who made many researches about computer and interface design documentation. His biggest research is about minimalism documentation which cited by many researches in computer science. He developed minimalism model for system approach [9].

Minimalism model is usually used for people who prefer problem solving than tutorial. John M. Carrol adopted the Nurnberg Funnel that represent minimalism as nine principles and implement the principles into three innovative research projects. The nine principles are as follows [9]:

1. Getting Started Fast (minimalism introduction)
2. Training on Real Track (problem solving practices)
3. Reading in Any Order (Modularity) (reading sequence modules)
4. Exploiting Prior Knowledge
5. Coordinating System and Training (synchronizing between system and learning process)
6. Supporting Error Recognition and Recovery (introducing and error finding)
7. Using the Situation (creating conducive condition for students)
8. Developing Optimal Training Designs (developing optimal learning design)
9. Reasoning and Improvising.

2. Methods
Analogy Mapping is being one menu of an online judge system calling CSPC System. It is used by students for practice, exercise, and evaluation of the basic programming subject and it is a web-based application supported by DOM Judge library [10]. Therefore, students can access the system easily anytime.

The learning process was done using minimalism model approach during time series evaluation milestone. Every evaluation milestone was done as follows:
1. the students did the pre-test using CSPC System,
2. the students got the minimalism explanation about the lesson concept and examples using Analogy Mapping as learning media in the class, this part fulfill the getting started fast principle,
3. the students were given a time duration for practicing to solve the problem, making source code using CSPC System and Analogy Mapping, and get online response to questions, these processes fulfil the rest eight principles of minimalism model,
4. the students did the posttest.

Every pre-test and posttest problems was a programming problem solving focused on logic and there are four lesson topics for evaluation. Every lesson topic is shown in table 1.

**Table 1. Lesson topics**

| Evaluation | Lesson Topics |
|------------|---------------|
| I          | • variable and data types, operator, and input output mechanism concept  
|            | • comparison (if), nested comparison and switch |
| II         | • looping (for)  
|            | • basic type array |
| III        | • array of string |
| IV         | • procedure  
|            | • function |

3. Result and Discussion

3.1. Evaluation Result
There are 48 first year students and 5 students from other year level in basic programming class. They consist of 13 female students (24.5%) and 40 male students (75.5%). There were 23 students learning basic programming before (43.4%). The most of those students learned basic programming focused on syntax than logic. The basic programming subject in this research was focused on basic programming logic and C languages was a tool for helping the students how making the implementation as a computer program.

The pretest and posttest results in every evaluation are shown in table 2 and 3.

**Table 2. Pre-test results**

| Evaluation | Pretest averages | Pretest maximum values | Amount of students who can do the pretest |
|------------|------------------|------------------------|------------------------------------------|
| I          | 5                | 80                     | 6                                        |
| II         | 4.433962         | 95                     | 6                                        |
| III        | 0.754716         | 40                     | 1                                        |
| IV         | 0                | 0                      | 0                                        |
| Averages   | 2.0377356        | 43                     | 2.6                                      |
Table 3. Post-test results

| Evaluation | Pretest averages | Pretest maximum values | Amount of students who can do the posttest |
|------------|------------------|------------------------|------------------------------------------|
| I          | 79.58491         | 100                    | 53                                       |
| II         | 71.69811         | 100                    | 52                                       |
| III        | 71.03774         | 100                    | 53                                       |
| IV         | 62.64151         | 100                    | 53                                       |
| Averages   | 71.24057         | 100                    | 52.75                                    |

N-Gain values as time series are shown in table 4.

Table 4. N-Gain results

| Evaluation | Pretest Averages | Posttest Averages | N-Gain        |
|------------|------------------|-------------------|---------------|
| I          | 5                | 79.58491          | 0.785104      |
| II         | 4.433962         | 71.69811          | 0.70385       |
| III        | 0.754716         | 71.03774          | 0.708175      |
| IV         | 0                | 62.64151          | 0.626415      |
| Averages   | 2.0377356        | 71.24057          | 0.705886      |

N-Gain or normalized gain was first introduced by Hake as a rough measure of the effectiveness of a course in promoting conceptual understanding. The overall N-Gain [11] is as follows,

\[ \text{N-Gain} = \frac{(71.24057 - 2.0377356)}{(100 - 2.0377356)} = \frac{69.2028344}{97.9622644} = 0.7064234 \]

Therefore, the N-Gain category is high which means that the learning process has high effectiveness.

3.2. Discussion

There were only several students who can do their pretest because the pretest problem focused on programming logic of the lesson topics using C programming language and most of the students did not learn about it before. In the other hand, almost all of the students tried to do their posttest after they learn about programming logic thinking in every lesson topic. The fourth evaluation has the lowest N-Gain because the lesson topic was more complex than the other.

4. Conclusion

Based on the research result, the N-Gain reaches 0.7, therefore, it shows that the learning process has high effectiveness for learning basic programming.

5. References

[1] Othman M, Hussain F M and Nikman K 2017 Enhancing logical thinking among computer science students through cooperative learning Gading Journal for the Social Sciences, 14 01.
[2] Kazimoglu C, Kiernan M, Bacon L and MacKinnon L 2012 Learning programming at the computational thinking level via digital game-play Procedia Computer Science 9 522-531.
[3] Milková E and Hulková A 2013 Algorithmic and Logical Thinking Development: base of programming skills WSEAS transactions on computers 12 2 41-51.
[4] Nikunja Swain P E 2013 Raptor-A vehicle to enhance logical thinking. Journal of Environmental Hazards 7 4 353-359.
[5] KALELIOĞLU F and Gülbahar Y 2014 The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective Informatics in Education 13 1.
[6] Wing J M 2008 Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences* **366** 1881 3717-3725.

[7] Flach P A 1994 *Simply Logical intelligent reasoning by example* (United Kingdom: University of Bristol).

[8] Sukamto R A, Prabawa H W and Kurniawati S 2017 Analogy Mapping Development for Learning Programming *Journal of Physics: Conference Series* **812** 1 012109

[9] Farkas D K and Williams T R 1990 John Carroll's the Nurnberg funnel and minimalist documentation. *IEEE Transactions on Professional Communication* **33** 4 182-187.

[10] NN 2012 DOMJudge http://domjudge.sourceforge.net

[11] Hake R R 1998 Interactive-Engagement vs. Traditional Methods: A Six-Thousand-Student Survey of Mechanics Test Data for Introductory Physics Courses Physics Education Research Supplement to AJP.