Analogy Mapping Development for Learning Programming

R A Sukamto, H W Prabawa and S Kurniawati

Universitas Pendidikan Indonesia, Jl. Setiabudhi No. 229, Bandung 40154, Indonesia
E-mail: rosa.ariani@upi.edu

Abstract. Programming skill is an important skill for computer science students, whereas nowadays, there many computer science students are lack of skills and information technology knowledges in Indonesia. This is contrary with the implementation of the ASEAN Economic Community (AEC) since the end of 2015 which is the qualified worker needed. This study provided an effort for nailing programming skills by mapping program code to visual analogies as learning media. The developed media was based on state machine and compiler principle and was implemented in C programming language. The state of every basic condition in programming were successful determined as analogy visualization.

1. Introduction

The media developed in this study maps many lines of computer program code to image as an analogy. The analogies used to provide assistance to students in representing the concept of programming that requires a process of abstraction into concrete form. The researcher already made observation for many years how teaching students using analogy as visualization of computer program code. The ideas using the analogies for teaching basic programming have been used for many years by first author. Therefore, the media was made for implementing the analogies ideas. There are many researchers using visual simulation for learning such as research which use simulator to make a program in programmable controller [1], reseach about using state machine for checking complex programmable logic devices (CLPD) design [2], and research about a ModelSim tool for designing electronic circuits in a machine using the state machine principle [3], animated interactive analogies for understanding programming variables [4]. Thus the visualization method is one of the solution to help novice learners understand the concept.

The media used state machine and compiler principles. The state machine is a sequence of analysis condition as processes or ideas [5]. The state machine principle was used for visualizing sequence of the program code process as sequence of image of analogy. The state was also considered using the compiler process concept. CodeMirror was used for tokenizing the program code as a token which CodeMirror is a library for javascript [6]. This media in purpose is one effort to implement basic programming analogy in order to make the students more easy understanding programming than before.

2. Why using analogies?

The analogies can be understood as a comparison of the similarity of the two concepts. The familiar concept is called the analog concept and the unfamiliar one the target concept. Both the analog and the target have features (also called attributes) which shown in Figure 1. If the analog and the target share similar features, an analogy can be drawn between them. A systematic comparison, verbally or visually, between the features of the analog and the target is called a mapping [7][8].

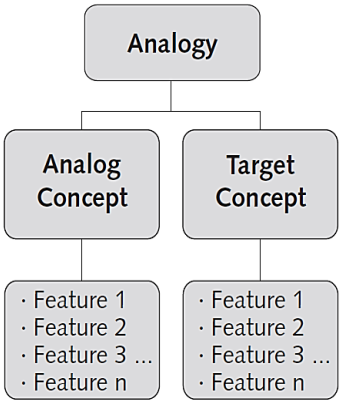


Figure 1. A conceptual representation of an analogy [8]

History proves that the analogy has contributed not a simple scientific discoveries. not by evidence, but rather as an inspiration. Analogy has also given its role in providing an explanation for a particular discoveries [8]. An analogy used in the classroom, textbooks and web-based learning should be designed to be activated in the elaboration, the cognitive processes that establish a relationship between what is already known and what will be known [8] including in programming [9][10][11]. Therefore analogy can be used as an effective tool in learning, teachers need to have enough knowledge regarding the pedagogical aspects. In its most basic form, at least the teachers have the ability to [12]:

- defines correspondence between analogy chosen by students as well as mapping targets: which concepts are obtained directly by teachers and which concepts developed by the students as a result of generalization analogy
- understand that analogy can not provide all of the aspects of learning that is required by students. It can be interpreted that the more the analogy is used to represent a concept, it would be better
- understand that not all students can understand the linkage between the concepts taught by analogy given.

Studies have shown that when used effectively, the analogy is a valuable pedagogical tool for teachers and media training for students in understanding the concepts.

3. Analogy mapping development

The student's learning style can categorized as follows:

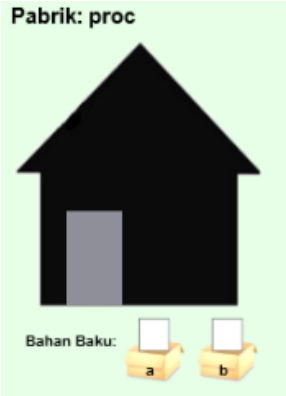
- visual learners which understand better by seeing some image or imagination image in mind,
- auditory learners which understand better by hearing,
- kinesthetic learners which understand better by seeing real things that touched or moved [13].

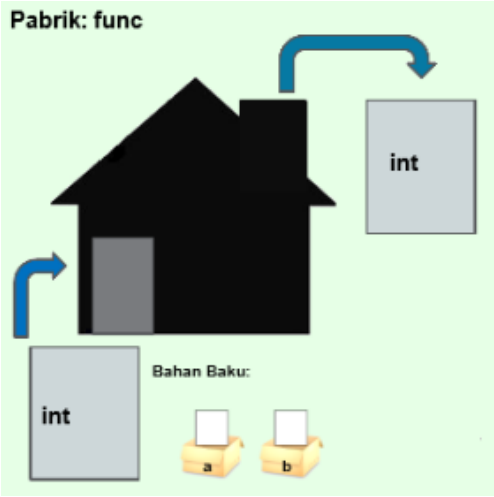
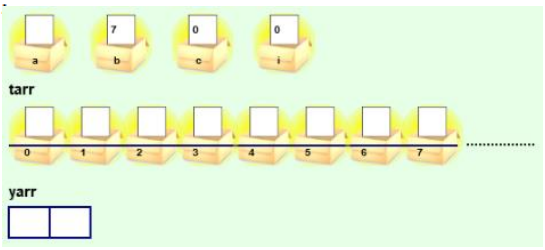
According to this, there are two parts using images or things surrounding as visualizing something. A media can help students for recording easier in their mind the things by using images. Thus, the students can conceive any text easier by using image if the text is converted as an image as an analogy. This research was developed a media for mapping every line of program code as an analogy which based on the concept can help students to understand programming. The analogies used in this research can be seen in the Table 1. Analogy Mapping is written using Indonesia language.

Table 1. Analogy mapping screen shoots examples.

Program Code	Analogy Image	Description

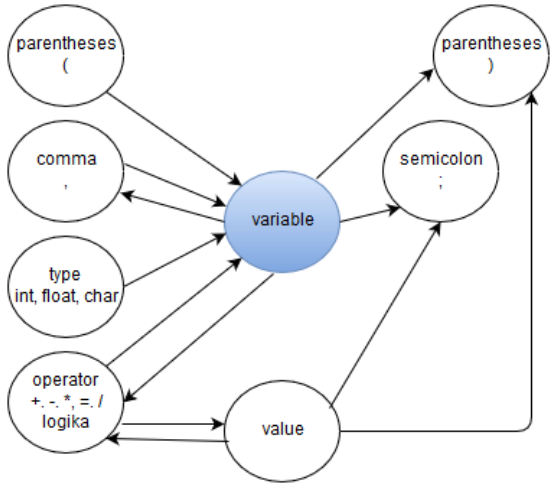
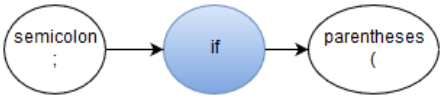
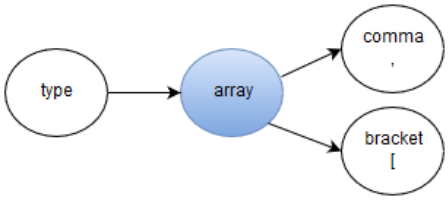
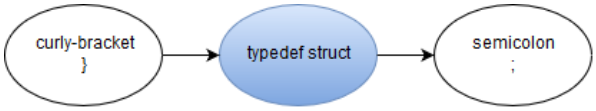
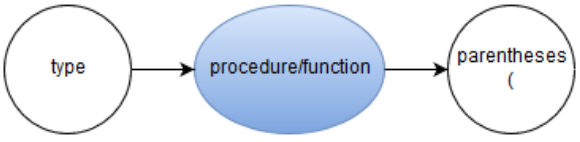
`typedef struct{`
`  int a;`
`  int b;`
`  int a1;`
`  int b1;`
`  int a2;`
`  int b2;`
`  int a3;`
`  int b3;`
`}bungkusan;`
		a structured type declaration, a bud is a symbol for early declaring (*jenis* = type, *bungkusan* = type name)

`void proc(int a, int b){`
 	
`}`
		a procedure declaration (*pabrik* = factory, proc = factory name, *bahan baku* = raw material)

`int func(int a, int b){` `}`		a function declaration (*pabrik* = factory, func = factory name, *bahan baku* = raw material)
`int a, b = 7, c, i, tarr[9], yarr[1][2];`		variables and arrays declaration in one line

Every analogy was developed by using the state machine principle manually. State machine principle more popular with finite automata which can model many state conditions in software or hardware. There are many state machine model examples used in this research as shown in the Table 2. The result analogies of sequence instruction in program code are shown in the Figure 2.

Table 2. State machine examples.

Description	State Machine
A variable declaration A variable usage	
If condition	
Looping for	
A one dimension array	
A structured type declaration	
A procedur declaration A function declaration	

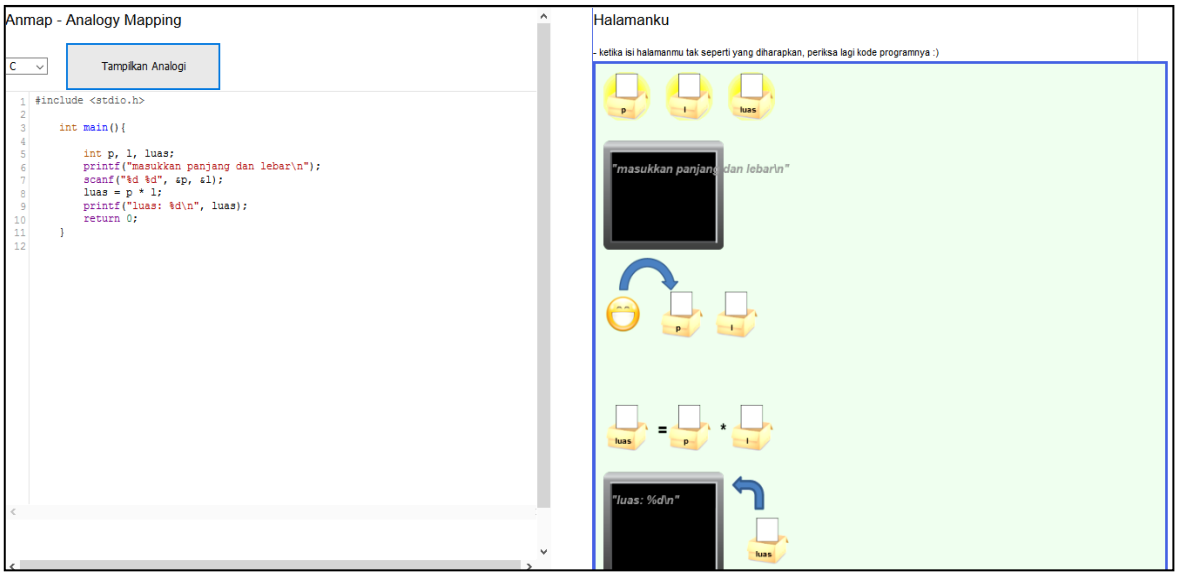


Figure 2. Analogies of sequence instruction in program code (tampilkan *analogi* = show the analogies, *halamanku* = my yard, *ketika isi halamanmu tak seperti yang diharapkan, periksa lagi kode programnya* = if your yard picture not as your expectation, please check your code).

Analogy Mapping was tested by 10 students that registered as first algorithm and programming subject students. They made many source codes to solve several problems and those source codes were used as input of Analogy Mapping. The result of the students tested was shown in Table 3.

Table 3. Analogy mapping testing result

Students	Source Codes	Lines	Correct Analogies	Wrong Analogies
Student 1	7	527	521	6
Student 2	9	263	257	6
Student 3	8	455	449	6
Student 4	8	498	495	3
Student 5	6	116	99	17
Student 6	5	115	113	2
Student 7	4	105	103	2
Student 8	3	47	44	3
Student 9	4	357	357	0
Student 10	3	79	77	2

4. Discussion

There are still appeared wrong analogies because there were several state undefined yet. The researcher will use the media to teach the students in order to get data as evaluation of the advantages of the media using the minimalist method in visual analogy learning model.

5. Conclusion

According to the result, all basic programming concept can be converted into an image as analogy using state machine and compiler principles, but the tested result showed that 47 analogies were wrong analogies (1.835938 %) and 2513 analogies were correct analogies (98.16406%). The media was supported by CodeMirror library as part of the process which converts program code into token and its label.

6. References

[1] Bathelt, Jens 2004 *Programming a Programmable Logic Controller* (Swiss: Federal Instituts of Technology Zurich)
[2] Simeonov, Simeonov I, Kukenska V S and Ilarionov R T 2005 *Realization of The Final State Machine with CPLD Device* (Bulgaria: Electronics)
[3] Altera Corporation 2011 Using *ModelSim to Simulate Logic Circuits for Altera FPGA Devices*. (San Jose: ALTERA Corporation-University Program)
[4] Doukakis D, Grigoriadou M and Tsaganou G 2007 *Understanding The Programming Variable Concept with Animated Interactive Analogies*
[5] Hopcroft, John E, Motwani R and Ullman J D 2001 *Introduction to Automata Theory, Languages, and Computation: Second Edition* (Boston: Addison-Wesley)
[6] _____, https://codemirror.net/(accessed 20th June 2016).
[7] Glynn S M 1994 Teaching science with analogies : a strategy for teachers and textbook author. 1994 *Reading Research Report No.15* (National Reading Research Project of University of Georgia and University of Mariland)
[8] Glynn, S M 2008 *Making Science Concept Meaningful to Students : Teaching with Analogies*.
[9] Forisek M and Steinova M 2012 *Proc. of the 43rd ACM technical symposium on Computer Science Education (Raleigh)* pp 15-20
[10] Gorgina, Moape T, Ojo S O and Lall M 2015 *Proc. of the EDSIG Conference on Information Systems and Computing Education (Wilmington)* pp 1-9
[11] Levy, Omer, Markovitch S 2012 *Proc. of the Twenty-Sixth AAAI Conference on Artificial Intelligence (Ontario)* pp 991-7
[12] Harrison, Allan G, Treagust and David F 2005 Metaphor and Analogy in Science Education. (Berlin: Springer Science and Business Media)
[13] Gardner Howard 2010 *Multiple Intelligences: New Horizons in Theory and Practice* (New York: Basic Books)