International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 12 (2019), pp. 3082-3088
© International Research Publication House. http://www.irphouse.com

A Study on The Effects of Block-Based Computing Language on Algorithmic Problem Solving

Jungin Kwon

Assistant Professor, Department of General Studies, Sangmyung University, Seoul, Korea.

ORCID: 0000-0001-7120-0964

Abstract

With the recent rapid development of the knowledge information age, SW-oriented creative and convergent talent is emerging as a keyword of future society. In recent years, SW education centered on majors has been spreading to non-majors and computing language and algorithm education for non-majors have become necessary. However, in the university where SW education centered on existing major was implemented, the curriculum, evaluation, and education system of SW education centering on the non-major is not yet structured. In this study, the curriculum was designed and systematically organized in the SW education of the non-majors, and the curriculum was created and used in the actual class. The main problem of the SW education of the non-major is that they have a preconceived notion that the SW education is difficult. Based on the previous study, the training should be preceded by a procedural listing of the non-major idea. We designed and taught curriculum based on block-based computing language. After that, we conducted a pre-post evaluation of motivation for learning by non-major. In addition, the algorithm training focused on sequential experience in problem solving and conducted pre-post evaluation. The results show that block-based computing languages have an effect on procedural listing of problem solving, comprehension of computing concepts, and expression of algorithms, but do not affect the choice of efficient algorithms. Based on this research, we hope to develop various education methods of block-based computing language for curriculum and educational motivation for SW education of non-major students.

Keywords: Computational Thinking, Block-based Computing Language, SW Education, Problem Solving, Algorithm Ability

I. INTRODUCTION

Recently, the importance of SW education and Computational Thinking is emerging due to the rapid development of knowledge information age. Wing(2006) suggested a new paradigm of information society education, arguing that all people should learn and study Computational Thinking as well as learning how to read and write[1].

This kind of SW education frenzy is spreading to all academic and industrial fields, and SW-oriented creativity and fostering human resources are emerging as key keywords of future social talents. Therefore, SW education based on Computational Thinking is emphasizing the importance of SW education for non-major students[2], as it aims to expand education opportunities not only for majors in a specific academic field but also for non-majors.

However, most of the non-major students have not received systematic SW education at all through the curriculum. In addition, most of these students' lack awareness of the need for SW education or are significantly less interested in SW education. For non-major students with these characteristics, the interest in SW education should be given priority over SW education. Therefore, SW education centered on the non-major students should provide various education opportunities to understand and integrate SW in each academic field in order to stimulate interest and motivation of learning. It should be composed of education that can understand and utilize SW from the viewpoint of non-major students[2].

In this study, we developed a SW course based on the concept and definition of Computational Thinking in order to understand the principles of computer science for non-major students and apply it to real life. The developed software course was conducted in the 2016, the first and the second semester of SW education for non-major students. Then we investigated the effect of block-based computing language on the list of algorithmic problem solving of non-major students.

II. RELATED WORK

II.I. Computational Thinking

The Computational Thinking was first described by Seymour Papert in an attempt to create a geometric thinking in 1996, and was more widespread by Wing[1].

Wing(2006, 2008) is a computer scientist's view of how to solve problems involving recursive thinking, abstract thinking, positive thinking, procedural thinking, logical thinking, concurrent thinking, analytical thinking and strategic thinking about the use of computer thinking[1],[3]. Therefore, he defined Computational Thinking as an analytical and procedural thinking process that must be learned to solve problems. Phillips(2007) and Perkovic(2010) saw Computational Thinking as more than just a skill for learners, but they saw thinking processes that improved people's intelligence[4],[5]. Thus Computational Thinking is not only a tool that can enhance the problem solving methods that we are already aware of and are currently teaching but also a measure of nurturing knowledge that enables one to view or approach complex or unstructured problems that could not be solved in the past. Therefore, Computational Thinking is defined as a process of analytical and procedural thinking to creatively solve problems in various fields.

Computer Science Teachers Association(CSTA)and International Society for Technology in Education(ISTE) classify the key concepts of information scientific thinking and abilities, based on David Barr, John Harrison, & Leslie Conery's(2011) study, into Data Collection, Data Analysis, Data Representation, Problem Decomposition, Abstraction, Algorithms & Procedures, Automation, Simulation and Parallelization, and suggest Computational Thinking as a standard computer science education process of K-12[6],[7],[9],[10]. The nine key concepts of Computational Thinking are shown in Table 1.

Table 1 Computational Thinking Main Concepts [6],[7],[8],[9],[10]

Concept	Definition
Data Collection	Problem understanding, analysis and collect data based on analysis to solve the problem
Data Analysis	Carefully sorting and analyzing the data collected and data provided in the problem
Data Representation	Express data in problem using graphs, charts, words and images
Problem Decomposition	Dividing and analyzing the problem to solve the problem
Abstraction	Defining the main concepts to reduce the complexity of the problem
Algorithm & Procedures	Expressing the steps required to solve the problem until now
Automation	Creating an algorithm of the solution procedure for a computing machine to carry it out
Simulation	Creating an experimental model to solve the problem
Parallelization	Coming up with a common objective to solve a problem

II.II. Non-Major Students for SW Education

As SW-oriented society spreads, each university is strengthening the training of SW talent with problem solving ability. It also includes SW basic education for non-major students, and encourages the development of SW curriculum design and support programs centering on non-major students. However, it is somewhat unreasonable to think that the curriculum or the curriculum provided so far is the development of the curriculum or the curriculum for the non-major students.

SW education for non-major students should develop an algorithmic tendency to solve the problem in a given field of study rather than focus on program development or coding. In other words, the SW education of non-major students should focus on improving the problem-solving ability of SW to solve the problems that can occur in each field of study. Most of the changing from problem to solving situation is presented.

Third, we focus on SW education that reflects characteristics non-major students are having difficulty in implementing the problems given to them in SW, and they are afraid of the SW itself. The reason for non-major students feel difficult is because most of the curriculum and education forms are conducted and evaluated in the same way as those of the majors. Even if the interests and interests of SW subjects are high, the introduction of the curriculum of the same method as that of the major and the introduction of the part of the curriculum given to the major can reduce the interest and interest of the SW educator. In addition, there is no educational contents in the SW course based on Computational Thinking, which is recently recognized as important. Computational Thinking based SW education for non-specialists focuses on algorithmic representation of problem solving rather than past coding education or SW development education.

Therefore, this study aims to design the development of curriculum for SW education based on Computational Thinking and to measure the effectiveness of algorithm solving by non-major students.

II.III. Design of a SW Course using Block-based Computing Language

The block-based programming language provides the commands necessary for programming in block form. It is also possible for a novice learner to easily experience the programming concept by programming a combination of necessary commands by selecting blocks[11],[12].

Wang(2015) reported that students who were majoring in engineering were more likely to achieve high levels of achievement when they were taught block-based languages such as App Inventors developed by MIT as the computing language of the first programming classes[13]. Kolling & Rosenberg(1996) general beginners use a "line by line" approach rather than approaching from the big structure aspect of development in programming courses, and spend less time in designing and testing code. And that they tend to write programs with only simple error correction[14]. O'Neillas(2012) stated that students can develop computer-based thinking skills and problem-solving skills using the Scratch language and find immediate solutions to problems[15].

Based on the results of the study that block-based computing language can reduce the fear of inducing excitement and programming of non-major students, we designed SW courses using block-based computing language for non-major students.

The goal of SW courses using block-based computing language for non-major students is as follows.

First, it aims to understand the principles of computer science and recognize the importance of SW by real life and academic field. It gives the importance and motivation of the SW education through the past cases which were able to present the practical problems that can be encountered through the real life experience to the non-major students and solve it with the principle of computer science.

Second, we develop a course that focuses on the algorithmic sequence of the problem solving ability enhancement based on the Computational Thinking. Sequence of solutions to problems that may arise in real life or each discipline is listed in sequence, and a method of sequencing the process of

of each major so that non-major students can solve problems in the major field through SW. We provide motivation for learning by presenting problem situation in each field related to major of students.

Fourth, it aims to improve the sequential problem solving ability based on Computational Thinking. Consider the improvement of problem solving ability based on Computational Thinking by presenting the nine key elements of problem solving process proposed by CSTA (Computer Science Teachers Association) and ISTE (International Society for Technology in Education) do.

Fifth, programming is applied to select algorithm considering efficiency among various methods of problem solving. It needs to understand that there are various solutions for problem solving in the same situation, and to find a way to choose a more efficient algorithm.

III. METHOD

III.I. Curriculum Development

After the design of SW education based on Computational Thinking for non-major students, we developed two subjects such as ‘Computational Thinking and SW Coding' and 'Problem Solving and Algorithm'. The period of these courses development is from October 10, 2015 to July 1, 2016, and the overall schedule for the curriculum development is shown in Table 2.

Table 2. Course Development Schedule

Time	Schedule
2015. 10.	Course development decision
2015. 11.	First-rate data preparation and expert opinion hunting
2015. 12.	Conducting expert reviews after preparing secondary data
2016. 1.	Use it as teaching materials for preparatory college
2016. 2.	After the revision, experts are evaluated.

III.I.I. Curriculum Development

'Computational Thinking and SW coding' is a course to raise the awareness of the principle of computer science, trend of the latest IT technology and importance of information society for non-major students who are new to SW. 'Computational Thinking and SW coding' is designed to learn the basic concepts of software through the theoretical education to understand the importance of computer science and SW education, and the principles of computer operation in the basic concept of SW education for non-major students. In the middle part, it includes the problems solving procedure and application process based on the concept of Computational Thinking.

In addition, the practical training is aimed at procedural listing of problem solving through block-based computing language Entry. A block-based computing language Entry is a block-based visual programming language similar to Scratch. It is a software education platform developed by the Institute of Entry Education. It is composed of easy interface shown in Figures 1.

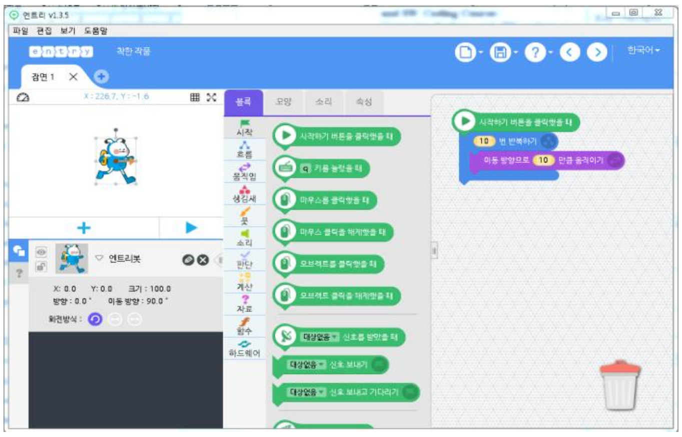

Fig. 1. Menu of Entry[12*]*

Entry can create animations, stories, games, etc. in a similar way to learners' scratches, and they can easily make corrections, exchanges, and extensions through the sharing of produced programs. Therefore, it is a computing language that somewhat supersedes the problems of excessive grammar learning and error correction in the existing programming language learning of non-major students. Entry can be more intuitive to Computational Thinking than traditional programming languages and focus on solving problems of a given task. The strength is that it provides much more objects than scratches, so that users can easily utilize necessary pictures or characters. The course design is shown in Table 3.

Table 3. Design of Computational Thinking and SW Coding Course

Key Concepts	Theoretical education	Practical training
Understanding the Principles of Computer Science	Computer science and SW education	Unplugged activity
	Principle of computer operation and data representation	Introduction and usage of Entry
	The development of the latest IT technology	Entry basic example
Concept and Application of Computational Thinking	Concept of Computational Thinking	Sequence / Event / Signal Example
	Solving problems based on Computational Thinking	Entry Application Example

	Understanding SW principles	Example of variables and lists
Procedures for Problem Solving		
	Variables and lists	Operator example
	Understanding Logic Operations	Selection example
	Understanding the control statements	Repeat logic example
	Understanding iterative logic	Repeat logic example
	Differentiating the differences between loops	Function Example
	Understanding the concept of functions	Sort and Search
	Understanding navigation and alignment	Advanced Example

III.I. II. Development of 'Problem Solving and Algorithm' Course

The course of 'Problems Solving and Algorithm' is aimed at improving efficiency in sequential listing if 'Computational Thinking and SW coding' are aimed at increasing interest in SW education of non-major learners and sequential list of problem solving. 'Problem Solving and Algorithm' are designed to help students choose the efficiency of the process of solving problems based on their athlete knowledge, experience, and thinking to solve the problems presented in the story rather than simply coding the program to solve the problem Education.

To solve the disadvantages of block-based computing language, problem solving and algorithmic training were conducted using python, a text-based language. The course design is shown in Table 4.

Table 4. Design of Problem Solving and Algorithm Course

Key Concepts	Theoretical education	Practical training
Procedures for Problem Solving	Necessity and process of problem solving	Concept of Physical Computing
	Step-by-step procedure for problem solving	Use of physical computer
	Concept and purpose of data structure	Python data entry
Various Problem Solving Methods	Solution Techniques for Puzzle Problems	Python Data Handling
	Techniques for solving logical problems	Program flow control
	Solution Techniques for Information Science Problems	Working with graphics
Choosing Efficient Problem Solving Methods	Data sorting technique	Program flow control
	Data search technique	Writing functions
	Brute Force Algorithm	Sorting technique
	Divide and Conquer Algorithm	Object-Oriented Program Concepts

III.II. Subjects and Characteristics of Research

This study was conducted to investigate the satisfaction of the subjects of 546 students in the 'Computational Thinking and SW Coding' and 'Problem Solving and Algorithm' subjects of the non-major students. The specific characteristics of the respondents are shown in Table 5.

Table 5. Characteristics of Study Subjects (n=546)

Item	Division	Number (persons)	Ratio (%)
Sex	Male	245	44.9
	Female	301	55.1
Major	Department of Humanities & Social	364	26.1
	Department of Social & Science	182	8.7

The questionnaire survey was conducted on a total of 20 questionnaires on a scale of 6 points. The contents of questionnaires were self-assessed on the satisfaction of the teaching plan, the satisfaction of contents composition, and the improvement of Computational Thinking after learning. The aggregation of the questionnaire was very similar, yes, slightly negative, positive, not at all, not so, slightly negative. The results of the questionnaire are shown in Table 6.

Table 6. Survey Results for Positive Responses (n=546)

Survey contents	Computational Thinking & SW Coding (%)	Problem Solving & Algorithm (%)
Understanding the Principles of Computer Science	61.4	71.8
Concept and Application of Computational Thinking	85.0	78.3
Procedures for Problem Solving	63.3	76.7
Various Problem Solving Methods	71.7	82.9
Choosing Efficient Problem Solving	60.0	81.0

The results of the survey showed that most of the students were satisfied with the developed textbooks and that they understood the importance of the information society through the textbooks of this class and recognized the importance of problem solving ability based on Computational Thinking.

III. III. Experimental Design

The experimental design of the study verified the effectiveness of the block-based computing language on the algorithmic expression through the learner's dictionary and post-test.

Q_1 X Q_2
Q_1 : Pre-test X : Conduct Block-based Computing Language Education Q_2 : Post-test

In order to measure the effectiveness of the algorithmic expression on the problem solving, a total of 40 questionnaires were used and 4 factors were measured using the 22nd questionnaire. As a result of the factor analysis, the factor of the algorithm expression was composed of procedural listing of problem solving, understanding of computing concept, expression of algorithm, and factors of selection of efficient algorithm.
According to Nunnally(1978), Van de Van & Ferry(1980) reported that the reliability of the measurement tool was not problematic when the exploratory research field was 0.6 or more. Therefore, in this study, Cronbach alpha coefficient was used to measure the internal consistency reliability of each factor. The measurement reference value was set to 0.6 or more[16],[17]. The results of the analysis of reliability are shown in Table7.

Table 7. Reliability Verification Results

Concept Variable	Question	Pre-test Cronbach α	Post-test Cronbach α
Procedural Listing	Q_2	0.794	0.808
	Q_3		
	Q_5		
	Q_6		
	Q_9		
Understanding Computing Concepts	Q_11	0.649	0.716
	Q_14		
	Q_15		
	Q_17		
	Q_18		
	Q_19		
Representation of Algorithms	Q_21	0.863	0.961
	Q_23		
	Q_25		
	Q_26		
	Q_29		
	Q_30		
Choosing Efficient Algorithms	Q_31	0.890	0.881
	Q_37		
	Q_38		
	Q_39		
	Q_40		

IV. Results

In order to investigate the effect of block-based computing language on algorithmic problem solving, pre-post t-test was performed on four factors.
Table 8 shows the results of verifying the preliminary results of procedural listing. The mean value increased from 3.84 to 3.97 and the significance probability was also smaller than 0.05, which was statistically significant.
Block-based computing language has been shown to affect procedural ordering.

Table 8. Procedural Listing

	M	n	SD	t	df	p
Pre-test	3.84	546	0.34	6.789	545	.000
Post-test	3.97	546	0.50			

$p<0.05$

Table 9 shows that the mean value increased from 3.15 to 4.01 and the significance probability was also less than 0.05 as a result of preliminary examination of the reflection of the Computational Thinking concept, so there was a statistically significant difference.
Block-based computing language has an effect on understanding of Computational Thinking concept.

Table 9. Understanding Computational Thinking Concepts

	M	n	SD	t	df	p
Pre-test	3.15	546	0.56	2.876	545	.001
Post-test	4.01	546	0.23			

p<0.05

Table 10 shows the result of verifying the pre & post-test of the algorithmic expression. The mean value increased from 2.79 to 3.01, and the significance was also less than 0.05, which was statistically significant.
In other words, it is shown that block-based computing language affects algorithmic representation.

Table 10. Algorithmic Representation

	M	n	SD	t	df	p
Pre-test	2.79	546	0.43	3.862	545	.012
Post-test	3.01	546	0.33			

p<0.05

Table 11 shows the result of the preliminary examination of the selection of efficient algorithms. The mean value increased from 3.57 to 3.69, but there was no statistically significant difference because the significance probability was greater than 0.05.
In other words, the block-based computing language does not affect the selection of efficient algorithms.

Table 11. Selection of Efficient Algorithm

	M	n	SD	t	df	p
Pre-test	3.57	546	0.32	1.796	545	.055
Post-test	3.69	546	0.97			

p<0.05

As a result of verifying that the block-based computing language has effects on procedural listing of problem solving, understanding of computer concept, expression of algorithm, and selection of efficient algorithm, which are factors of algorithmic problem solving, The procedural listing of the solution, the understanding of the computer concept, and the expression of the algorithm, but does not affect the choice of the efficient algorithm.
Thus, it was concluded that block-based computing language is effective as a language of basic education that can attract interest and interest in SW education of non- major.

V. CONCLUSION

Recently, the rapid development of the future knowledge-based society has been focused on the improvement of problem solving ability based on the SW. As the necessity of SW education and training course is claimed, attention is focused on software education and supply and demand of SW talent is urgent. However, at the universities where SW education centered on existing majors focused on, the curriculum, evaluation, and education system of SW education centering on the non-specialized ones have not yet been systematized.
In this study, the curriculum was designed for systematization of the SW education of non-major and opened the curriculum, and then the lesson was written and used in the actual class. The main problem of SW education of non-major is that they have a preconceived notion that SW education is vaguely difficult and based on the previous study, As a result of the training after designing, it was concluded that block-based language is effective in procedural listing of problem solving, comprehension of Computational Thinking concept, and expression of algorithm. However, the effectiveness of the algorithm was not influenced by the education using the block-based language. It is more effective to motivate learning by using block-based language and to construct curriculum using Computing Language when SW education is conducted to non-major.
Based on this research, we hope to develop various education methods of block-based language for curriculum and educational motivation for SW education of untrained person. This not only allowed us to recognize the necessity of SW education in each major field, but it also can provide an opportunity to develop various talent fields and SW skills. I hope that the SW education model will be presented in order to suggest a new direction to the SW education of the non-major.

REFRENCES

[1] J. M. Wing, Computational Thinking. Communications of the ACM, Vol. 49, No. 3, 2006, pp. 33-35.

[2] Jungin Kwon & Jaehyoun Kim, A Study of SW Curriculum based on Computational Thinking for non-majors. Korean Society for Internet Information, Vol. 17, No. 1, 2016, pp. 247-248.

[3] J. M. Wing, Computational Thinking and Thinking about Computing. Philosophical Transactions of the Royal Society, Vol. 366, 2008, pp. 3717-3725.

[4] P. Phillips, Computational Thinking: A Problem-Solving Tool for Every Classroom. in NECC,

http://www.cs.cmu.edu/~CompThink/resources/ct_pat_phillips.pdf, 2007.

[5] L. PerKovic, A. Settel, S. Hwang and J. Jones, A Framework for Computational Thinking across the curriculum, Proc. of International Conference on ITiCSE, 2010.

[6] Computer Science Teachers Association & International Society for Technology in Education, Computational Thinking Teacher Resources, http://csta.acm.org/Curriculum/ sub/CompThinking.html, 2011.

[7] D. Bar, J. Harrison and L. Conery, Computational thinking: A Digital Age Skill for Everyone. Learning & Learning with Technology, Vol. 38, NO. 6, 2011, pp. 20-23.

[8] Jungin Kwon, A Study on the Effectiveness of Computational Thinking based Teaching and Learning on Students Creative Problem Solving Skills, Ph.D. Dissertation, Sungkyunkwan University, 2014.

[9] http://www.iste.org/docs/learning-and-leading-docs/march-2011-computational-Thinking-ll386.pdf

[10] http://csta.acm.org/Curriculum/sub/CurrFiles /472.11CTTeacherResources_2ed-SP-vF.pdf

[11] Scratch: https://scratch.mit.edu

[12] Entry: https://playentry.org

[13] Wang, K.. Enhancing the teaching of CS 1 by programming mobile apps in MIT App. age, Vol. 2, No. 1, 2015, pp. 26.671.1-26.671.9.

[14] Kolling, M. & Rosenberg, J. Blue- ALanguage for Teaching Object-Oriented Programming, Proc. of the 27th SIGCSE Technical Symposium on Computer Science Education, 1996, pp. 190-194.

[15] Ornelas Marques, F., Marques, M.T. No Problem? No Research, little Learning ... Big Problem!. Systemic, Cybernetics and Informatics, Vol.10, No. 3, 2012, pp. 60-62.

[16] Nunnally, J. C., Psychometric Theory, New York, NY: McGraw-Hill, 1978.

[17] Van de Ven, A. H. & Ferry, D. L., Measuring and Assessing Organizations, John Wiley & Sons, Inc. 1980.