Design of Programming Learning Process using Hybrid Programming Environment for Computing Education

DaiYoung Kwon[1], IlKyu Yoon[1] and WonGyu Lee[2]
[1] Computer Science Education, Graduate School, Korea University
Anam-dong, Seoul, Korea.
[e-mail: {daiyoung.kwon, ilkyu.yoon}@inc.korea.ac.kr]
[2] Department of Computer Education, Korea University
Anam-dong, Seoul, Korea.
[e-mail: lee@inc.korea.ac.kr]
*Corresponding author: WonGyu Lee

Received March 31, 2011; revised July 31, 2011; accepted August 28, 2011; published October 31, 2011

Abstract

Many researches indicate that programming learning could help improve problem solving skills through algorithmic thinking. But in general, programming learning has been focused on programming language features and it also gave a heavy cognitive load to learners. Therefore, this paper proposes a programming activity process to improve novice programming learners' algorithmic thinking efficiently. An experiment was performed to measure the effectiveness of the proposed programming activity process. After the experiment, the learners' perception on programming was shown to be changed, to effective activity in improving problem solving.

Keywords: Programming learning, algorithmic thinking, teaching and learning strategy

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This work was funded by the National Research Foundation of Korea under Grant 2010-0015981.

DOI: 10.3837/tiis.2011.10.007

1. Introduction

Computing technology is converged in various areas of the current information-oriented society, and is the foundation of developing new technologies. Furthermore, computing technology is influencing life, thinking and the behaviors of people who are living in an information-oriented society, with its effects still growing [1]. Especially, the importance of thinking abilities such as computational thinking, algorithmic thinking, abstraction and automation from computing education is emphasized for the active use of information technology [2][3]. Accordingly, various curriculums including computing education have been developed and applied worldwide [2][3][4][5][6]. Especially, the report in ACM(A Model Curriculum for K-12 Computer Science) emphasizes an algorithmic thinking for the improvement in problem-solving ability, based on computing technology and its active use [2]. Also, in this report, programming activity is suggested as an educational content for improvement of algorithmic thinking. Results from various research studies show that the programming learning can actually help improve algorithmic thinking, by designing and implementing problem solving procedures and verifying the results [7][8][9].

In the process of general programming learning, the step of learning programming language features such as commands and grammar is performed first, and then the step of learning how to design algorithms for problem solving is carried out later. In the step of learning programming language features, students typically demonstrate their knowledge of the language features by comprehending an already coded program [10]. This step causes great cognitive burdens to novice learners and therefore, they spend lots of time on learning programming language features [11]. Knowledge of the language features such as program instructions or syntax is necessary for using the programming language, but not sufficient enough to improving algorithmic thinking. However, introductory programming courses often emphasize the learning the programming language features than focusing on how to design algorithms, despite core activity of algorithmic thinking being accomplished in the process of implementing algorithms. Therefore, a programming activity for learning algorithmic thinking must be pointed towards designing algorithms.

Recently, various educational programming languages have been developed to ease the difficulties of learning programming language features [12][13][14]. Several precedent studies showed that educational programming language reduced the learners' cognitive load and encouraged learning algorithmic thinking [9][15].

Even though educational programming language is used, programming activities which follow the process of general programming learning would be still recognized as the activities for understanding the programming language features. And knowledge of the language features is necessary for improving algorithmic thinking, but not sufficient enough to show improvement on its own [10]. Consequently, for efficient learning of algorithmic thinking, not only using educational programming languages, but also engaging in programming activities that focus on algorithmic thinking should be carried out. However, the majority of studies on programming activities were carried out in a learning strategies perspective, aimed at effectively solving problems that occur during programming activities such as Fair Programming [16], rather than trying to improve algorithmic thinking.

Therefore, this paper proposes a programming activity process for novice learners to improve algorithmic thinking efficiently in introductory courses. The proposed programming activity process is designed to minimize the step of learning programming language features and skills. The difference from general programming activity processes is that it does not have an overall programming language learning step, but directly starts with problem solving algorithm designs, thus only learning the minimal amount of programming language methods needed to implement algorithms during the programming process.

To prove the validation of the proposed programming activity process, an experiment was performed to measure the learners' perception of programming. The experiment results showed that learners' perception of programming had been changed positively.

This paper is organized as follows. Section 2 presents difficulties of the general programming learning and related works of educational programming environments. Section 3 presents the programming activity process for learning algorithmic thinking in detail. Experiment and analyses on experimental results are explained in Section 4 and 5. Section 6 concludes this paper.

2. Related Work

2.1 Programming Difficulties and Learning Strategies

Programming is a very complex subject that requires effort and a special approach in the way it is learned and taught. The discussion on the nature and types of problems encountered in learning programming is based on areas of difficulty as identified [17], but with some additional concerns are identified through reseach such as the issue of resources. These areas are: (1) the cognitive requirements of programming; (2) syntax and semantics; (3) orientation; (4) the auxiliary skills needed for programming; and (5) resource constraints [18].

The first area of difficulty in programming is the cognitive requirements of programming. In general, programming requires a wide range of knowledge, such as the detail of syntax, the semantics specific to the programming language used, and mental models of how to slove the problem. The second area of difficulty in teaching programming is the notation for representation of a program. Notation refers to the symbols of a programming language and the syntactic rules for combining them into a program [19]. The third area of difficulty in teaching programming is the orientation. Orientation refers to the difficulties students have in recognizing and identifying what the term programming actually means, what processes programming actually entails, and what it is useful for. The fourth area of difficulty in teaching programming is the demand for additional or auxiliary skills. Auxiliary skills necessary for programming include proficiency in dealing with the development environment, such as the operating system interface of the computer, and also other technical skills such as editing, compiling, and debugging of a program [20]. The last area of difficulty in teaching programming is the lack of quality instructional resources and the lack of teachers with a sufficient background in teaching programming [21].

The most renown study on learning strategies is Pair Programming. Pair Programming is a programming method in which two people pair up on one computer and each take on a role. The first person, who inputs using the keyboard or mouse, is called the Driver, and the second person, who also looks at the algorithms next to the Driver and establishes the overall strategy is called the Navigator [22]. This programming learning strategy is proven to be very effective in the educational field as well as recognized to decrease errors and increase productivity in the industrial field [23][24][25]. Most of the other studies, were carried out an easy approach to programming activites, with strategies aimed at effectively using tools such as Unplugged [26], and reasearches with learning strategies aimed at increasing algorithmic thinking were barely done.

2.2 Educational Programming Environments

Algorithmic Thinking can be effectively improved through programming activities which use educational programming environments. Accordingly, educational programming environments are organized for the beginners to easily engage in programming activities with interest. For this, educational programming environment provides intuitive commands and simple interfaces. The following studies are examples of the education programming environment developed recently for algorithmic thinking

Scratch [13] is an educational programming language that was made by the Lifelong Kindergarten team from the MIT-media lab and the YasminKafai team from UCLA. It is consisted of objects named ‘Sprite’. The Paint Editor allows to users to create their own sprites or customs and background image from their local drive. The various set of command blocks, which is called script, makes sounds or reacts to other sprites. The command is divided into 8 categories such as motion, looks, sound, pen, control, sensing, operators and variables. Novice programmers learn programming with interest, by using various multimedia that creates games and animations with offered characters, backgrounds and the sound effect. Moreover, scratch is an educational programming language that helps students to get more effective learning through the sharing of their ideas with others on all types of devices in computing environments [27]. Programming activities with scratch can help the learners to understand the flow of a program and learn algorithms [28]. Also, students can have interests, and it offers an opportunity for them to easily approach programming. They can also share their works online and can help each other by exchanging their opinions [29].

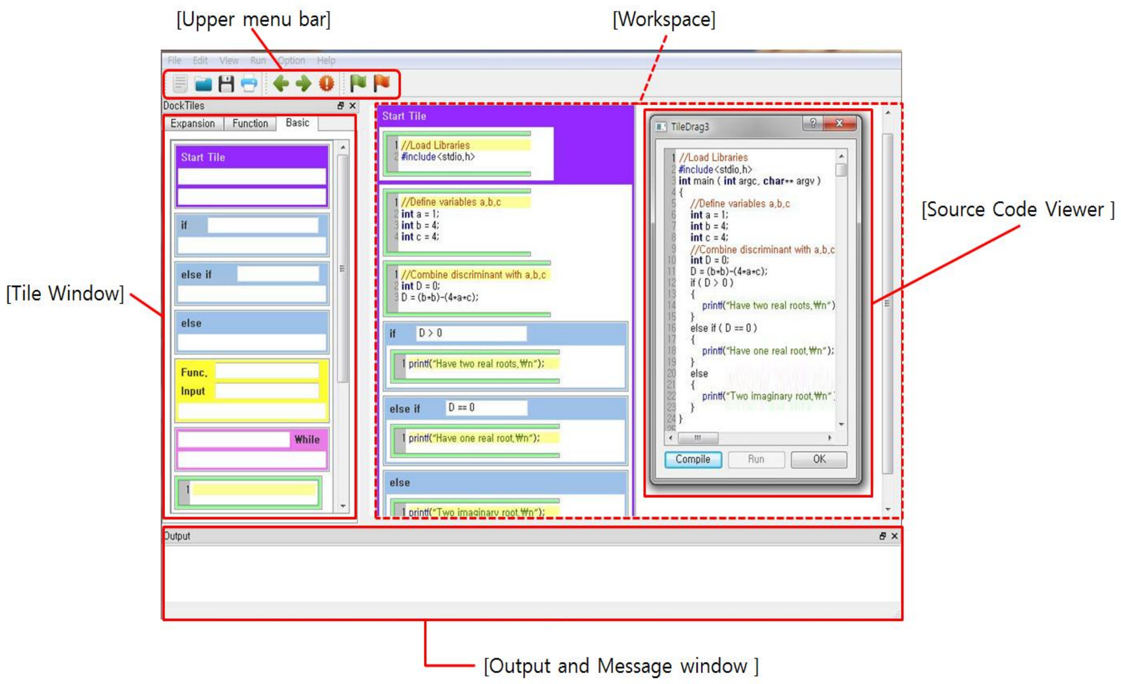
Squeak Etoys [14] is an educational programming language made by Alan Kay. Squeak Etoys is object oriented programming language. It helps students create objects by drawing pictures with various multimedia data such as photographs, sound, and so on. It is a kind of educational programming tool that drags and drops each command to each object. Its media-rich tile scripting is visual programming and typing error free. Squeak Etoys has 13 categories commands, which are color, geometry, pen use, test, motion and etc. Novice programmer can create his/her own story, game or puzzle by using the ‘Book’ function and become nterested in it. Also, Squeak Etoys is an educational programming tool used in various subject field like Mathematics, Science, Art and etc., not only by students but also by teachers. This provides an opportunity to interestingly express the complex system of a real-world, from it's interaction with the objects.

Dolittle is a language based on object-oriented way of thinking, developed by Kanemune in 2000 [12]. It is different from the other subject-oriented languages and was developed for programming education for K-12 students. It is a subject-oriented language that doesn’t need any high-abstraction concepts like ‘inheritance’ or ‘class’. This means that when it creates a new subject, it programs by coping prototype subject instead of using class. Without understanding of definition of class or hierarchy, novice programmers can concentrate on the problem solving processes.

Computer Science Unplugged was developed by Tim Bell and his colleagues at the University of Canterbury in New Zealand [26]. From elementary school children to college students, it is purposed for understanding computer science without using computer. It is composed of 20 Unplugged activities, and the subjects are about data processing, information theory, algorithm, programming, automata, information security, code, human computer interface, etc.

2.3 Hybrid Programming Environment

Hybrid programming environment [30] is the educational programming environment that concurrently supports both a tile scripting interface such as scratch [13] and a textual programming interface. As providing both control statement tiles, which can present sequence, repeatation, branch and text tiles, in which text can be entered unrestrictedly, hybrid programming environment can offer two types of programming interfaces. Therefore, as shown in **Fig. 1**, programming based on tile scripting can be performed by operating the tiles which include programming codes. Also, textual programming can be performed by writing programming codes in the text tile directly in the hybrid programming environment.



```
//Load Libraries
#include<stdio,h>
int main ( int argc, char** argv )
{
    //Define variables a,b,c
    int a = 1;
    int b = 4;
    int c = 4;
    //Combine discriminant with a,b,c
    int D = 0;
    D = (b*b)-(4*a*c);
    if ( D > 0 )
    {
        printf("Have two real roots,₩n")
    }
    else if ( D == 0 )
    {
        printf("Have one real root,₩n");
    }
    else
    {
        printf("Two imaginary root,₩n"
    }
}
```

Fig. 1. User interface of hybrid programming environment

Through the characteristic of hybrid programming environment, which supports two types of programming interface, learners can use tile scripting interface by using general textual programming languages such as ‘C language’. And it can support various programming languages.

Programming activities that utilize the hybrid programming environment, first configures the algorithm-based programming frame in the workspace by using the tile in the tile window, then goes on to create programming codes based on the functions that are provided in the programming language in each tile. In the revision process, the algorithm-related parts are usually carried out based on the tile scripting interface, while the programming code areas are done through textual interface. These merits of hybrid programming environment can offer a flexible programming environment which can support various types of programming activities, from activities for novices to activities for experts.

3. Programming Activity Process for Learning Algorithmic Thinking

3.1 Comparison with general programming learning process

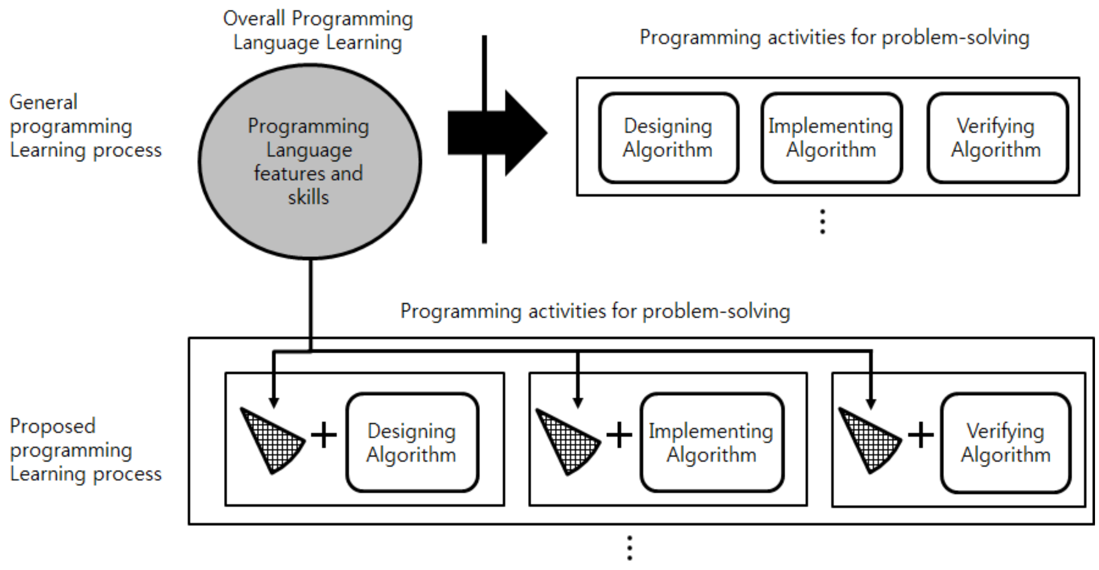

Overall Programming Language Learning
Programming activities for problem-solving
General programming Learning process
Programming Language features and skills
Designing Algorithm
Implementing Algorithm
Verifying Algorithm
Programming activities for problem-solving
Proposed programming Learning process
Designing Algorithm
Implementing Algorithm
Verifying Algorithm

Fig. 2. Comparison with general programming learning process

As seen in **Fig. 2**, the programming learning process proposed in the this paper skips the programming language learning and goes strarts with problem-solving activities. First, the students create an algorithm to solve the given problem. During this process, the students learn limited programming language features that are required in the algorithm expression, such as control statements and annotations, and then create the algorithm utilizing the features. Second, the program coding process which implements the algorithm is carried out. In this process, the students learn about the programming lanague functions and skills that are needed to implement the created algorithm, and also create the programming code. The last is the process of verifying the created program. This step, similar to previous steps, the students learn programming language functions and skills that are used in the verification, and the verification work is carried out.

Compared to the general programming learning process, the proposed programming learning process does not prceed with the programming language learning process separately, but only learn the necessary information for the algorithm design, implementation, and verification processes. Therefore, the programming language learning section is minimized, and in the problem solving algorithm-generation perspective, the programming langauge learning is dependantly performed, so programming activity which is focused on algorithmic mindsets proceed.

Also, by deciding on the level of the programming language learning depending on the level of the algorithms, and by offering the appropriate programming activity problem level, the programming language learning can also be carried out in an effective, spiral form of learning.

3.2 Details of the Programming Learning Activity Process with Hybrid Programming Environment

Fig. 3 shows the details of the proposed programming activity process.

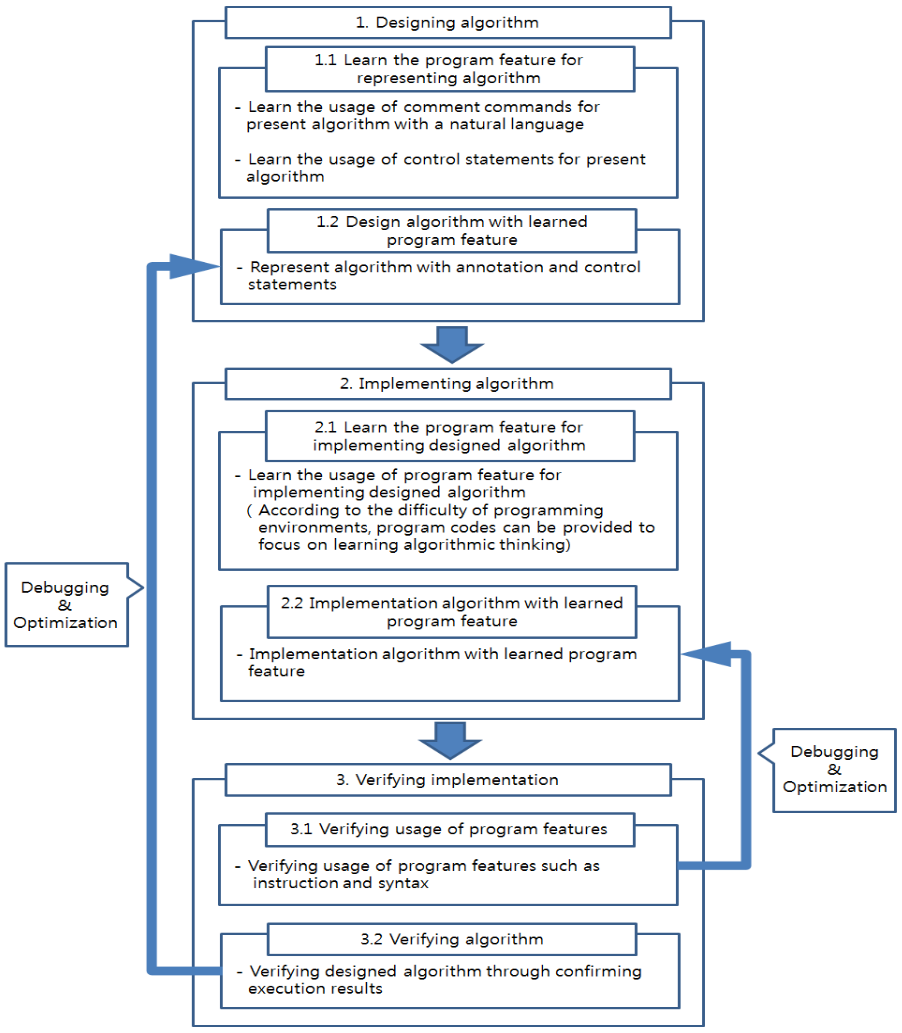
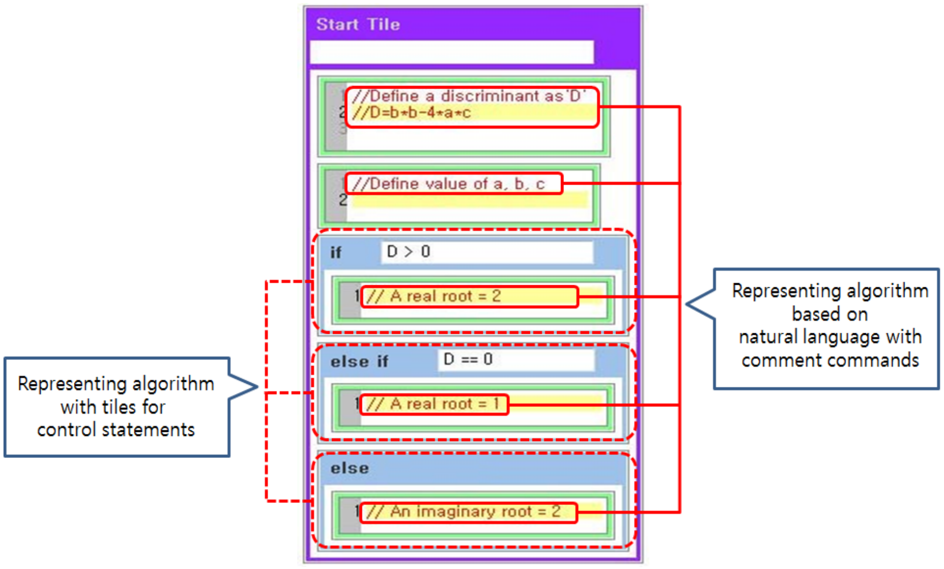

1. Designing algorithm
1.1 Learn the program feature for representing algorithm
- Learn the usage of comment commands for present algorithm with a natural language
- Learn the usage of control statements for present algorithm
1.2 Design algorithm with learned program feature
- Represent algorithm with annotation and control statements
2. Implementing algorithm
2.1 Learn the program feature for implementing designed algorithm
- Learn the usage of program feature for implementing designed algorithm (According to the difficulty of programming environments, program codes can be provided to focus on learning algorithmic thinking)
2.2 Implementation algorithm with learned program feature
- Implementation algorithm with learned program feature
3. Verifying implementation
3.1 Verifying usage of program features
- Verifying usage of program features such as instruction and syntax
3.2 Verifying algorithm
- Verifying designed algorithm through confirming execution results
Debugging & Optimization
Debugging & Optimization

Fig. 3. Programming activity process for learning algorithmic thinking

As shown in **Fig. 3**, the proposed programming activity process consists of 3 steps (Designing algorithm, implementing algorithm and verifying implementation). And the each 3 steps have 2 sub steps.

The first is designing the algorithm, which consists of 2 sub steps(learning programming features and designing algorithms based on the learning features). In the sub step for the learning programming features to represent an algorithm, commands for comments and control statements such as sequence, repeatation and branch are learned to represent problem solving procedures. Then the sub step for designing algorithms proceeds. In this sub step, an algorithm is designed based on natural languages with learned programming features in the previous sub step as shown in **Fig. 4**.

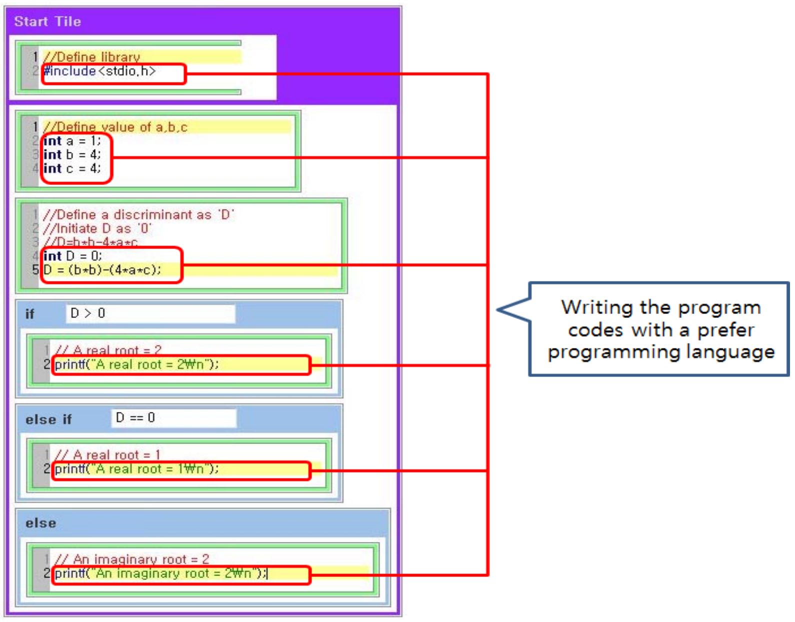


```
Start Tile

//Define a discriminant as 'D'
//D=b*b-4*a*c

//Define value of a, b, c

if      D > 0
    // A real root = 2
else if D == 0
    // A real root = 1
else
    // An imaginary root = 2
```

Representing algorithm with tiles for control statements
Representing algorithm based on natural language with comment commands

Fig. 4. Example of representing algorithm using hybrid programming environment

The second step is implementing the algorithm, and also has 2 sub steps (learn the programming features for implementing designed algorithm and implementing the algorithm with learned programming feature). In the first sub step, the minimum program features to implement the algorithm, such as variable types and input function, are learned. Then the sub step for implementing the algorithm proceeds. In this sub step, an algorithm is implemented with learned programming features in previous sub step as shown in **Fig. 5.** But, due to the difficulty of programming features required in implementation, already coded programs can be provided to focus on learning algorithmic thinking.



```
Start Tile

//Define library
#include<stdio.h>

//Define value of a,b,c
int a = 1;
int b = 4;
int c = 4;

//Define a discriminant as 'D'
//Initiate D as '0'
//D=b*b-4*a*c
int D = 0;
D = (b*b)-(4*a*c);

if      D > 0
    // A real root = 2
    printf("A real root = 2₩n");
else if D == 0
    // A real root = 1
    printf("A real root = 1₩n");
else
    // An imaginary root = 2
    printf("An imaginary root = 2₩n");
```

Writing the program codes with a prefer programming language

Fig. 5. Example of implementing an algorithm using hybrid programming environment

The third step is verifying implementation, and also has 2 sub steps(verifying usage of program features and verifying algorithm). In the first sub step, the usage of program features is verified through inspecting the error from execution of the implemented program from the previous step. In the second sub step, the algorithm is verified through confirming the execution results from the implemented program. If problems occur in this step, users are able to return to previous steps and modify the algorithm or programming codes for correct problem solving procedures.

The proposed programming activity process that is focused on designing algorithm and its representation is the difference from the general programming learning process. In the proposed programming activity, the steps of learning program language features are minimized or excluded. Hence, in the process of programming activities, learners could improve their algorithmic thinking effectively, by reducing the cognitive load of learning program language features.

4. Method

4.1 Experiment Procedure

4.1.1 Development of Questionnaire

Neuroscientific measurement methods are known to be the most scientific way to measure thinking skills, since it is very difficult to measure directly [31]. However, for such measurements neuroscientific tasks(stimulus) and measurement methods are required, as well as long periods of learning [32]. Because of such limits in measuring thinking abilities, the current study wants to indirectly measure the feasibility of the proposed programming activity process, by measuring the changes in the students perception towards programming activity, with the recognition variable basis being the measuring of the educational effects [33][34].

In general, programming activity is a difficult and challenging subject area which places a heavy cognitive load on its learners [18]. Also, programming has been recognized as a specialized skill for computer engineers or computer scientists. This is because general programming activity focused on learning the programming language features rather than learning how to solve problems, such as algorithmic thinking [10]. In this paper, the proposed programming activity process focused on learning algorithmic thinking. Therefore, learners can recognize proposed programming activity as an educational activity for improving problem-solving skills rather than as a specialized activity for experts. In order to validate the proposed programming activity process, we developed a questionnaire to measure changes of perception on programming.

In this paper, the perception of programming was divided into two specific categories: 'perception on the role of programming' and 'perception on the purpose, necessity and interest of programming '. The questionnaire on the perception on the role of programming consisted of 4 questions, and there were 8 questions that inquired on the purpose, necessity and interest of programming.

4.1.2 Subjects of Experiment

The subjects for the experiment were 89 pre-service elementary school teachers who enrolled in a class named 'Theory of elementary school computer education' at the C university of education. There were 29 males and 60 females, and none of the participants had prior experience in programming.

4.1.3 Procedure of the Programming Course

The hybrid programming environment based programming course were composed in 3 weeks(6 sessions). The specific procedures of the programming course based on the proposed programming learning process using hybrid programming environment is as follows.

Table 1. Procedure of the programming course

Class	Major Activities
1	Pre-Questionnaire Introduction of Hybrid Scripting Programming Language
2	A Discriminant Algorithm(1)
3	A Discriminant Algorithm(2)
4	A Sorting Algorithm(1)
5	A Sorting Algorithm(2)
6	An Add Algorithm Post-Questionnaire

4.2 Method of Analysis

In this study, we conducted two dependent sample t-test for analyzing change of learners' perception on programming. Also, we conducted chi-square test for analyzing changes of learners' perception on the most appropriate role of programming. We used SPSS 12.0 for analyzing measurement data.

5. Experiments Results & Analyses

5.1 Analysis of Role of Programming

Table 2 is the pre-post analysis results for the 4 roles of programming. Each queston is measured on a 5 point Likert scale, and the participants answered all questions. The question "It is an effective educational tool for enhancing problem solving skills?" showed an increase, going from 3.34 to 3.74, and was statistically significant. This shows that the proposed programming activity process was recognized as improving the students' problem solving skills.

Table 2. Analysis of the learners' perception on the role of programming

Question	Pre M(SD)	Post M(SD)	t
Everyone needs to learn the basic programming skills because it is widely used in other areas of study.	3.12(.90)	3.13(.83)	.094
It is a high school level vocational education for those who want to become professional programmers.	3.79(.87)	3.79(.92)	.000

It is an academic field for researchers with the purpose of conducting research.	3.51(.97)	3.53(.83)	.160
It is an effective educational tool for enhancing problem solving skills.	3.34(.89)	3.74(.94)	2.993**

** p < .01

Table 3 Shows the results for the questionnaire that required the participants to pick one programming role that they thought was the suitable. The pre-post results showed a noticeable difference, and was statistically significant (p<.05).

Table 3. Results of the chi-square test

Question	Result of Questionnaire		x^2	p
	Pre Frequency(%)	**Post** Frequency(%)		
Everyone needs to learn the basic programming skills because it is widely used in other areas of study.	14(15.6)	15(16.7)	9.856	.020(*)
It is a high school level vocational education for those who want to become professional programmers.	52(57.8)	35(38.9)		
It is an academic field for researchers with the purpose of conducting research.	5(5.6)	3(3.3)		
It is an effective educational tool for enhancing problem solving skills.	18(20)	36(40)		
Sum of Frequency	89	89		

p<.05(*)

As shown in Table 3, the choice "It is a high school level vocational education for those who want to become professional programmers" was selected by 52 participants at first, and then decreased to 35 after, and the choice "It is an effective educational tool for enhancing problem solving skills" increased, going from 18 before to 26 after. This shows the that the proposed activity process changed the participants' recognition of programming as an activity that is not only needed for professional programmers, but as an activity that improve problem solving skills.

5.2 Analysis of Purpose, Necessity and Interest of the Programming activity

Table 4. shows analysis of learners' perception on purpose, necessity and interest of programming.

Table 4. Analysis of learners' perception on purpose, necessity and interest of programming

Fact	Question	Pre M(SD)	Post M(SD)	t
Purpose of Programming	The purpose of programming is to develop good softwares.	3.84(.74)	3.73(.77)	1.010
	Programming education can be helpful in understanding the main computer science principles and structures.	3.82(.76)	3.96(.79)	1.216
	Programming can enhance general problem solving skills.	3.48(.80)	3.93(.85)	3.488**
Necessity of Programming	Programming can be easily approached by lower grades in elementary schools.	2.39(1.01)	2.73(.99)	2.598*
	There is a need to adopt programming education in the regular school curriculum.	2.91(.79)	3.13(.86)	1.918
	Programming education is needed in order to live in an information-oriented society.	3.57(.81)	3.52(.84)	.387
Interest of Programming	I am able to create programs using the programming tools.	1.95(.90)	2.47(1.03)	3.506**
	I am interested in programming and I would like to learn more about it.	3.00(1.20)	3.48(1.03)	2.747**

** p < .01

In case of the purpose of programming, all 3 questions showed an increase in points from before to after. Especially, the answer "Programming can enhance general problem solving skills" showed a statistically significant (p<.01) increase, going from 3.48 (pre) to 3.96 (post). This shows that the proposed programming activity process is recognized as increasing the students' problem solving ability. For the need of programming, the question "Programming can be easily approached by lower grades in elementary school" showed a statistically significant (0<.05) difference. Thus, when applying the proposed programming activity process, programming activity for problem solving will be possible even for lower elemntary scshool grades. Lastly, for the interest in programming, both questions showed a statistically significant (p<.01) difference, which can be analzyed as the proposed programming activity being recognized as very interesting.

Through chapters 5.1 and 5.2, we proved the feasibility of the proposed programming activity process. Therefore, to become a programming learning model that effectively increases algorithmic thinking, programming language learning, such as the proposed programming activity process, needs to be subsidiarily made in algorithmic thinking learning.

6. Conclusion

In this paper, we proposed the programming activity process for learning algorithmic thinking. The proposed programming learning process was designed to focus on learning algorithmic thinking, rather than the programming language features and skills. Through the experiment for investigating learners' perceptions on programming, the validity of the proposed programming activity process was proven. Furthermore, developments of introductory programming courses based on the proposed programming activity process are required to learn effective algorithmic thinking.

References

[1] C.E. Wills, D. Deremer, R.A. McCauley, L. Null, "Studying the Use of Peer Learning in the Introductory Computer Science Curriculum," *Computer Science Education*, vol. 9, pp. 71-88, 1999. Article (CrossRef Link)

[2] A. Tucker, F. Deek, J. Jones, D. McCowan, C. Stephenson, A. Verno, "A Model Curriculum for K-12 Computer Science," Final Report of the ACM K-12 Task Force Curriculum Committee, 2003.

[3] R. Shackelford, A. McGettrick, R. Sloan, H. Topi, G. Davies, R. Kamali, J. Cross, J. Impagliazzo, R. LeBlanc, B. Lunt, "Computing Curricula 2005: The Overview Report," in *Proc. of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2005.

[4] Ministry of Education, "The Ontario Curriculum Grades 10 to 12 - Computer Studies(revised)," *http://www.edu.gov.on.ca*, 2008.

[5] Department of Computer Science and Engineering, "CS Curriculum for K-12 Schools," *http://www.cse.iitb.ac.in/*, 2010.

[6] Advanced Placement Central, "Computer Science A Course Description," *http://apcentral.collegeboard.com*, 2010.

[7] B. Adelson, "Problem Solving and the Development of Abstract Categories in Programming Languages," *Memory and Cognition*, vol. 9, pp. 422-433, 1983. Article (CrossRef Link)

[8] R.D. Pea, D.M. Kurland, "On the Cognitive Effect of Leaning Computer Programming," *New Ideas Psychology*, vol. 2, no. 2, pp.137-168, 1984.

[9] S. Cooper, W. Dann, R. Pausch, "Developing Algorithmic Thinking with Alice", in *Proc. of Information Sys. Educators Conf.*, pp.506-539, 2000.

[10] M.C. Linn, "The Cognitive Consequences of Programming Instruction in Classroom," *Educational Researcher*, vol. 14, no. 5, pp. 14-29, 1985.

[11] S. Garner, "Cognitive Load Reduction in Problem Solving Domains," in *Proc. of ICCE2001*, 2001.

[12] S. Kanemune, T. Nakatani, R. Mitarai, S. Fukui, Y. Kuno, "Dolittle-Experiences in Teaching Programming at K-12 Schools," in *Proc. of The Second International Conference on Creating, Connecting and Collaborating through Computing*, 2004. Article (CrossRef Link)

[13] M.l Resnick, J. Maloney, A. Monroyhenrández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, Y. Kafai, "Scratch : Programming for All," *ACM Communications*, vol. 52, no. 11, pp. 60-67, Nov. 2009. Article (CrossRef Link)

[14] A. Kay, "Squeak Etoys, Children, and Learning," http://www.squeakland.org/resources/articles.

[15] M.C. Carlisle, "Raptor: A Visual Programming Environment for Teaching Object-oriented Programming," *Journal of Computing Sciences in Colleges archive*, vol. 24, no. 4, April, 2009.

[16] J.E. Hannay, T. Dyba, E. Arisholm, D.I.K. Sjøberg, "The Effectiveness of Pair Programming: A Meta-Analysis", *Information and Software Technology*, vol.51, no.7, pp.1110-1122, 2009. Article (CrossRef Link)

[17] B.D Boulay, "Some Difficulties of Learning to Program," *Journal of Educational Computing Research*, vol. 2, no. 1, 1986.

[18] I.T. Chan Mow, "Issues and Difficulties in Teaching Novice Computer Programming," *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*, pp. 199-204,

2008.
[19] D.G. Moursound, "Increasing your expertise as a problem solver: Some roles of computers," Eugene, OR: ISTE, 2002.
[20] M. Pedroni, "Teaching Introductory Programming with the Inverted Curriculum Approach," *Diploma thesis*, 2003.
[21] A. Gomes, A.J. Mendes, "Learning to Program - Difficulties and Solutions," in *Proc. of ICEE 2007*, 2007.
[22] K. Beck, C. Andres, "Extreme Programming Explained: Embrace Change", Addison-Wesley, 2003.
[23] I.D. Steiner, "Group Process and Productivity", Academic Press, New York, London, 1972.
[24] J.M. Levine, R.L. Moreland, "Progress in small group research", *Annual review of psychology*, vol. 41, pp.585–634, 1990.
[25] N.L. Kerr, R.S. Tindale, "Group Performance and Decision Making", *Annual review of psychology*, vol. 55, pp.623–655, 2004.
[26] T. Bell, I. Witten, M. Fellows, "Computer Science Unplugged," http://www.csunplugged.org, 1998.
[27] K. Peppler, Y.B. Kafai, "From SuperGoo to Scratch: Exploring Creative Digital Media Production in Informal Learning," *Learning, Media, and Technology*, vol. 32, no. 2, pp. 149-166, 2007. Article (CorssRef Link)
[28] O. Meerbaum-Salant, M. Armoni, M, Ben-Ari., "Learning Computer Science Concepts with Scratch," in *proc. of the 6th international workshop on Computing education research*, pp. 69-76, 2010. Article (CrossRef Link)
[29] Y. Kafai, K. Peppler, G. Chiu, "High Tech Programmers in Low-income Communities: Creating a Computer Culture in a Community Technology Center," *Communities and Technologies*, pp. 545-564, 2007. Article (CrossRef Link)
[30] D. Kwon, J. Kim, Y. Kim, W. Lee, "Developing a Hybrid Programming Interface for Educational Programming Languages in Computing Education," in *Proc. of KSII The 2nd International Conference on Internet (ICONI),* Dec. 2009.
[31] C.A. Paynter, K. Kotovsky, L.M. Reder, "Problem-solving Without Awareness: An ERP Investigation," *Neuropsychologia*, vol. 48, no. 10, pp. 3137-3144, 2010.
[32] C. Zimmerman, "The Development of Scientific Thinking Skills in Elementary and Middle School," *Developmental Review*, vol. 27, no. 2, pp. 172-223, 2007.
[33] F. Norales, "Post-secondary Student's Attitudes toward Computers," *Journal of Computer Information Systems*, vol. 15, no. 7, pp. 15-20, 1987.
[34] L. Shashaani, "Gender-based Differences in Attitudes Toward Computers", *Computers Education*, vol. 120, no. 2, pp. 169-181, 1993. Article (CrossRef Link)

DaiYoung Kwon is a Ph.D. student in the Department of Computer Science Education at Korea University. He received his B.S and M.S degrees in Computer Science Education form Korea University, Seoul, in 2000 and 2006. His research interests include algorithmic thinking learning, educational programming languages and cognitive experiments for evaluating thinking abilities.

IlKyu Yoon is a Ph.D. student in the Department of Computer Science Education at Korea University. He received his B.S in Computer Science Education from KongJu National University, KongJu in 2008 and M.S degrees in Computer Science Education form Korea University, Seoul, in 2010. His research interests include algorithmic thinking learning, educational programming languages and logical thinking.

WonGyu Lee is a Professor at Korea University, Dept of Computer Science Education in Seoul, Korea. His M.S and Ph.D. degrees in Computer Science from Tsukuba University. From 1993 to 1996 he was a senior researcher at Korea Arts and Culture Service. He was the president at Korea Association of Computer Education in 2002. Currently, he is the chief director of Creative Informatics & Computing Institute at Korea University. His research interests include computer science education, information retrieval and Database.