Brain, Digital, & Learning
2017, Vol. 7, No. 2, 67-73.

Understanding Computational Thinking as a Mode of Thinking in K-12 Education

Hye-Jeong Kim*
Chung-Ang University, Korea

<ABSTRACT>

Computational thinking is a thinking skill used to understand and solve problems through automation and abstraction using computing tools. As a systematic mode of problem solving in a radically changing global society, computational thinking is a rising and core competency. The recent focus on computational thinking as a core competency for all students has led to outline a national curriculum including software as a required subject in Korea. In this study, a conceptual approach to computational thinking in K-12 education is explored. Also, an integrated approach between computing and school curriculum using real-world problems by competent teachers is emphasized for the inclusion of computational thinking in K-12 education.

Key words : Computational thinking, Computing, Programming, Digital literacy

Introduction

Rapid changes in the context of the Fourth Industrial Revolution have led to the reconsideration of the competencies of human capital. In 2016, the world was shocked when AlphaGo, the artificial intelligence program developed by Google DeepMind, beat Korean Go Champion Sedol Lee. People were shocked because they did not imagine that current computing power could address a highly complex and creative problem such as Go, which is a difficult game for a computer to master. After defeating Lee, social responses to this historical result in Korea were observed in various fields, such as the Korean government's $863 million investment in artificial-intelligence research and educational support by the federal government and local governments, including increasing the number of research schools in coding and increasing the number of new students being trained in computing by universities (i.e., the number of new students managed by the Ministry of Education in Korea) (Zastrow, 2016).

Notably, the interest in AlphaGo's historical

* Corresponding Author : Hye-Jeong Kim, hjkim0305@gmail.com
Received May 24, 2017; Accepted June 24, 2017; Published June 30, 2017

victory focused on discussion of future competencies necessary for survival for students working with artificial intelligence and how to enhance these competencies in the current education system. On this topic, computational thinking began being discussed as a core competence of the future competencies. Although somewhat later to start than other countries (e.g., UK, Finland, and US), the requirement of coding and relevant abilities to enhance computational thinking in Korean school curriculum is slated to begin in 2018, and teacher training and curriculum development are underway. In the midst of these changes, computational thinking has been considered as key to preparedness for future success, similar to traditional literacy such as writing, reading, and arithmetic.

Within the circumstances of the school setting, this paper discusses the meaning and characteristics of computational thinking in a school setting and educational approaches, including instructional strategies.

Computational Thinking

Janet M. Wing, who coined the term computational thinking in 2006, defined computational thinking as a thought process for solving problems, designing systems and understanding human behavior that draws on concepts fundamental to computing (Wing, 2006). She insisted that in the 21st century, computational thinking should be regarded as a fundamental skill that all students need to develop, similar to reading, writing, and arithmetic. Furthermore, the NRC in the US (2010) insisted that computational thinking is a cognitive skill that is of use to everybody. Computational thinking is an analytical thinking approach to solving problems.

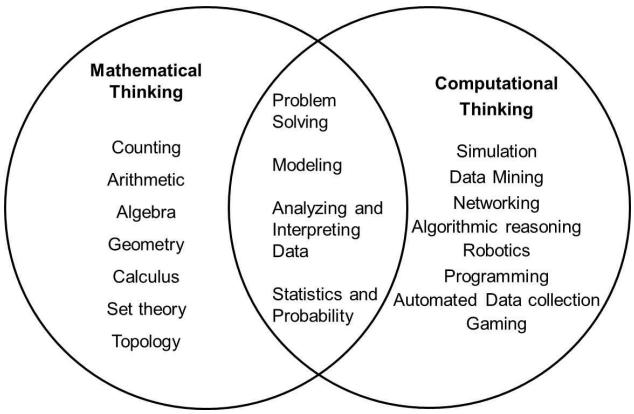
What does "computing" mean with reference to the phrase "computational thinking"? Wing (Wing, 2008; Wing, 2010) proposed that "computing is the automation of abstractions" and characterized abstraction as a core concept in computational thinking and as the most important and high-level thought process. Simply, abstraction is to model reality to gain knowledge and to solve problems about the phenomenal world. For example, representing real-world phenomena using mathematics and physics formulae to model them is an abstraction in the context of mathematics or physics. Thus, computational thinking is a thought process that can be employed to understand and solve problems using the abstraction characteristic of computing (Wing, 2006). People who engage in computational thinking can become more efficient problem solvers. Here, the method of concrete abstraction is computer programming using computer languages, including Python, C/C++, and Scratch. Computational thinking includes abstraction in addition to algorithmic thinking automation, decomposition, debugging, and generalization (Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016). Thus, computational thinking is a cognitive thinking process moving from problem analysis to problem solution rather than simply a programming process. Sometimes, computational thinking includes logical thinking, algorithmic thinking, or parallel thinking and involves various thought processes (Wing, 2010).

Computational Thinking as Rigorous Thinking

Computational thinking is more than just the skill of computer programming (Grimson et al., 2017). To be educated in computational thinking,

people can understand how to model large-scale systems using abstraction and modularity, analyze complex datasets, and support a computational complement to physical experiments on real-world problems (Grimson et al., 2017). Fundamentally, people that have computational thinking skills use a computer programming language as a framework within which to explore computational concepts (Grimson et al., 2017).

Most of all, computational thinking is similar to mathematical or logical thinking (Grimson et al., 2017; Sneider, Stephenson, Schafer, & Flick, 2014) and problem solving (V. Barr & Stephenson, 2011), see Figure 1.



[Figure 1] Venn diagram of mathematical and computational thinking (Sneider et al., 2014)

According to Barr, Harrison, and Conery (2011, 21p), computational thinking as problem solving includes the following:

- Formulating problems in a manner that enables us to use a computer and other tools to help solve them
- Logically organizing and analyzing data
- Representing data through abstractions, such as models and simulations
- Automating solutions through algorithmic thinking in a series of ordered steps
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem-solving process to a wide variety of problems

Mathematics

$A = \pi r^2$, π =3.1415

Radius = 4

Solution

$A = \pi \cdot r^2$

$= \pi \cdot 4^2$

$= 50.264$

Radius = 5

Solution

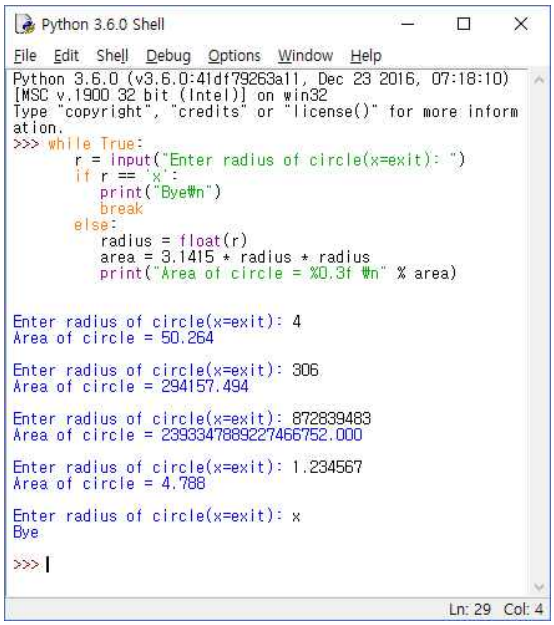
$A = \pi \cdot r^2$

$= \pi \cdot 5^2$

$= 78.540$

…

Programming in Python



[Figure 2] Calculation of the area of a circle via mathematics and Python programming

What, then, makes computational thinking rigorous thinking? Computational thinking combines structured reasoning with creative exploration of paths to reach a result. Grimson et al. (2017) emphasized the differences between computational thinking and mathematical thinking as managing complexity, limiting resources, and effectiveness in modeling physical and social systems. Most of all, computational thinking emphasizes how-to knowledge (imperative) instead of what-is-true knowledge (declarative). For example, the formula for the exact area of a circle is different from algorithmic methods to compute the area of a pentagon using coordinates; see Figure 2. When the area of a circle is calculated in mathematics, the equation for the area of a circle is adopted and calculated. The equation is A (area) equals r squared. Here, pi is a constant that is equal to 3.1415. Here, the radius of the circle is four. r2 equals four times four, or 16. The area of the circle, A, is 3.1415 times 16. Therefore, the area of the circle is 50.264. In this example, if it is necessary to continually recalculate the area with different values for the radius, people have to recalculate after entering new values. It will take time to obtain the result each time the area must be recalculated.

However, the approach of computational thinking fundamentally emphasizes abstraction and automation. Typically, the tool of computational thinking is a programming language such as Python. In Python, to calculate the area of a circle, it is expected to calculate the equation with iteration of the process for different values of the radius and the results. Thus, the code includes a while statement to automate repetitive calculations. In the flow of execution for a while statement, True is used to evaluate the condition for executing the body or exiting the statement. If the condition is true, the body is executed, and the program then goes back to the top of the statement to reevaluate the condition. Until x is entered, the loop in the while statement will repeat forever to calculate the area of the circle; this is called an infinite loop. The value of the radius can be entered at the prompt, and the results can be obtained without limits of trials and numbers (here, floating-point real values).

In computational thinking, the expression of abstracted thoughts or tasks can vary by context. For example, to understand the concept of the area of a circle, children can draw a turtle program based on the mathematical concepts of logic. People can print a circle-shaped badge using a 3D printer and a modeling program.

Computational Thinking in K-12 Education

The pervasiveness of computational thinking is in line with many aspects of 21st century competencies, such as creativity and critical thinking (Ananiadou & Claro, 2009; Binkley et al., 2012; Mishra & Yadav, 2013). Educators recognize that computational thinking through programming is important for K-12 students in the era of the 4th Industrial Revolution (Lye & Koh, 2014).

When Logo programming, in which students moves a turtle on the screen by entering commands, was introduced, it was considered a tool for developing thinking skills and was integrated with mathematics (Feurzeig, Papert, & Lawler, 2011). In recent years, the idea of Logo as a thinking tool revived people's interest and spread widely through the term "computational thinking" and many new programming languages, such as Scratch and Python. Computing, as a core concept in computational thinking, and computer programming, as a tool to represent computing, can be employed in K-12 education.

K-12 educators can teach computational thinking using two approaches, including independent subjects and subject-integrated approaches. Historically, the traditional approach to computational thinking is to educate students in independent computing subjects. Lye and Koh (2014) asserted that recent interest in programming for K-12 settings should consider the method to be related with educational outcomes that it can potentially foster. The characteristics of computational thinking elaborated by many researchers emphasized it as a problem solving method using computer programming, with strategies such as algorithms, abstraction and debugging (Bundy, 2007; Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011).

Recently, the pervasiveness of computational thinking as a key skill for K - 12 students emphasizes the importance of experiencing modes of thinking and their applications in the real world (Barr & Stephenson, 2011; Voogt, Fisser, Good, Mishra, & Yadav, 2015). Students in K-12 schools should be exposed to real-world situations in which they are required to solve problems using programming. Students who experience real-world contexts will develop more motivated attitudes toward computational experiences as a mode of creative problem solving. Finally, computational thinking should be considered as a core competency instead of a skill. Thus, teachers need to create or adopt various educational environments to engage students in computational thinking activities. Barr and Stephenson (Barr & Stephenson, 2011) emphasized that competency in computational thinking ideas and tools at the K-12 level are important because students will enter a workforce heavily influenced by computing. They also highlight "algorithmic problem solving practices and applications of computing across disciplines, and help integrate the application of computational methods and tools across diverse areas of learning". Thus, computational thinking in K-12 education has been developed by integrating computing into the curriculum and by including it as an independent subject such as computing or computer programming to create educational opportunities in domain-specific contexts across different disciplines (Israel, Pearson, Tapia, Wherfel, & Reese, 2015; Mishra & Yadav, 2013). For example, a framework developed by Sengupta, Kinnebrew, Basu, Biswas, & Clark (2013) for aligning scientific inquiry with concepts in computational thinking can serve as an example of how to integrate computational thinking in K-12 education. In Korea, the Ministry of Education and the Ministry of Science, ICT and Future Planning highlighted the importance of exposing students to computational thinking in schools and have outlined a national curriculum integrating computing education, including required subjects such as Software, to help students understand and apply these essential skills. The revised national curriculum includes 17 hours of computing education at the elementary school level and 34 hours at the middle school level beginning in 2019 and 2018, respectively.

Conclusions

In summary, computational thinking as rigorous thinking emphasizes the importance of integrating computational thinking curriculum and education based on its special characteristics. Computational thinking is a thinking skill used to understand and solve problems through automation and abstraction using computing tools. In the era of the 4th Industrial Revolution, modern life is strongly influenced by computing, making it imperative that educators recognize that

computational thinking through programming is important for K-12 students and the need to integrate computational thinking in school. Computing is no longer a subject in K-12 education to teach computer science principles. Students need to be educated through an integrated approach between computing and school curriculum using real-world problems. Thus, students should be engaged in integrated computational thinking during their learning activities and should be instructed by competent teachers. Embedding computational thinking into the foundational concepts of big ideas from many subjects can be accomplished via instructional design, learning activities, and integrated lessons.

References

Ananiadou, K., & Claro, M. (2009). 21st Century Skills and Competences for New Millennium Learners in OECD Countries (EDU Working paper no. 41).

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning and Leading with Technology*, 38(6), 20 - 23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads,* 2(1), 48.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining Twenty-First Century Skills. *In Assessment and Teaching of 21st Century Skills* (pp. 17 - 66).

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing Computational Thinking in Compulsory Education. https://doi.org/10.2791/792158

Bundy, A. (2007). Computational Thinking is Pervasive. *Journal of Scientific and Practical Computing*, 1(2), 67 - 69.

Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5), 487 - 501.

Grimson, E., Chakrabarty, D., Cuthbert, M. S., Hosoi, P., Mueller, C., Orlin, J., & Voorhis, T. Van. (2017). *A computational thinking requirement for MIT undergraduates*.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, 82, 263 - 279.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51 - 61.

Mishra, P., & Yadav, A. (2013). Of Art and Algorithm: Rethinking Technology & Creativity in the 21st Century. *TechTrends*, 57(3), 10 - 14.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2), 351 - 380.

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science Framework and NGSS: Computational thinking in the science classroom. *Science Scope*, 38(3), 10 - 15.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715 - 728.

Wing, J. (2006). *Computational Thinking*, 49(3), 33

- 35.

Wing, J. (2008). Computational thinking and thinking about computing. *IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM*, (July), 3717 - 3725.

Wing, J. M. (2010). Computational Thinking: What and Why? Thelink - The Magaizne of the Varnegie Mellon University School of Computer Science, (March 2006), 1 - 6.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). *Introducing computational thinking in education courses. Educational Studies*, (2), 465 - 470.

Zastrow, M. (2016). South Korea trumpets $860-million AI fund after AlphaGo “shock.” Nature. Retrieved from http://www.nature.com/news/south-korea-trumpets-860-million-ai-fund-after-alphago-shock-1.19595.