Computational Thinking in Solving Engineering Problems – A Conceptual Model

Choon Hui, Neo[1], Jee Khai, Wong[2*], Voon Chiet, Chai[3], Yaw Long, Chua[2], Yeh Huann, Hoh[1]
[1]Faculty of Engineering and Technology, Tunku Abdul Rahman University College, Setapak, Kuala Lumpur, MALAYSIA.
[2]Institute of Informatics and Computing in Energy (IICE), Universiti Tenaga Nasional, MALAYSIA.
[3]School of Energy, Geoscience, Infrastructure and Society (EGIS), Heriot-Watt University, MALAYSIA.
*Corresponding author: wongjk@uniten.edu.my

Received: 25 May 2021; **Accepted:** 11 September 2021; **Published:** 14 September 2021

To cite this article (APA): Neo, C. H., Wong, J. K., Chai, V. C., Chua, Y. L., & Hoh, Y. H. (2021). Computational Thinking in Solving Engineering Problems – A Conceptual Model. *Asian Journal of Assessment in Teaching and Learning*, *11*(2), 24-31. https://doi.org/10.37134/ajatel.vol11.2.3.2021

To link to this article: https://doi.org/10.37134/ajatel.vol11.2.3.2021

Abstract

Programming is an excellent approach to cultivating computational thinking (CT) skills lacking among current engineering undergraduate students. Although highly useful in teaching programming skills, physical, tangible programming tools available in the market are limited to users aged 12 and below, a gap that impedes the effort to cultivate problem-solving skills and computational thinking among engineering students. As a result, many students who join engineering programmes are without solid computer programming skills. This paper proposes a method to tackle the said gap by applying physical programming education blocks. The programming blocks have various logical functions and input-output capabilities that allow decision-making, looping, and function calling. Users can build their logical thinking skills in the form of cause-and-effect analysis using the play method. Through this approach, students can enhance their programming skills, which improves their computational thinking ability and complex problem-solving skills. It is hoped that such an approach could help them in transiting from tangible programming to text-based programming.

Keywords: Computational Thinking, Education, Programming, Teaching and Learning

INTRODUCTION

Computational thinking (CT) focuses on the mindset and the way the mind thinks rather than solely on computing (Li, 2020). There have been multiple studies and discussions on the importance of computational thinking in many areas. Denning (2017) explained how computational thinking could benefit society, diSessa (2018) discussed its applications in mathematics education, and Grover and Pea (2013) suggested that it has the potential to impact younger students as young as five to six years old in the K-12 education. In addition, according to the United States National Research Council (2010), computational thinking should be acquired by everyone and not limited to only programmers. Thus, given the importance of computational thinking, skills in computational thinking should be instilled among students. As computational thinking is related to the way the mind thinks, it is closely related to programming.

Programming can ease tasks for humans, from simple manual tasks to more demanding or near impractical tasks such as analysing big data (Hooda, 2017). Today, programming has become increasingly vital in various fields such as information technology, data analysis, computer science, and

engineering. In 2015, there were approximately seven million jobs for occupations that required coding skills (Dishman, 2016), and programming jobs were increasing at a rate of 12 per cent faster than the market average (Oracle Academy, 2015). Since then, many have argued that programming should be included as part of the national curriculum, on par with subjects such as the native language, science, and mathematics (McFadde, 2019). Yet, many undergraduate students experience hardship when trying to understand the basic concepts of programming such as logic and algorithm (Mutiawani and Junita, 2014). Thus, the lack of programming literacy further reduces the students' interest to learn other relevant subjects (Tan et al., 2009).

At present, Malaysia is facing a shortage of students enrolling in Science, Technology, Engineering and Mathematics (STEM) programmes at the university level (Kaur, 2019). Despite many studies emphasising the importance of programming skills, it has not been included in the curriculum of the national education system. The latest list of subjects offered by the Ministry of Education of Malaysia for the Sijil Pelajaran Malaysia (SPM) 2021 indicated that such a gap exists (Lembaga Peperiksaan, 2021). To add to the said problem of low students number in STEM that causes a shortage of engineers for the nation, the lack of programming skills among the current engineering students will hinder their abilities to solve complex problems logically.

Efforts to improve students' programming skills require suitable educational tools, which are currently limited. Therefore, this study proposes a conceptual design that utilises physical block-based programming architecture with various logical functions and input/output capabilities to instil computational thinking (CT) and problem-solving skills. It is anticipated that the application of the physical block-based programming approach would be able to bridge the gap between secondary school and college programming syllabuses, as programming is less likely to be taught in secondary schools.

LITERATURE REVIEW

Definition of Computational Thinking

Wing (2006) first defined computational thinking (CT) as a set of thinking skills, habits, or approaches needed to solve complex problems using an information-processing agent, and that it is widely applicable in the current information society. In other words, computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by a computer (Wing, 2011). It covers more than programming, including a range of mental tools reflecting basic principles and concepts of computer science, for example: abstracting and decomposing problems, identifying recurring patterns, and generalising solutions. Therefore, computational thinking is a vital skill that everyone should acquire (Lockwood and Mooney, 2017). Orr (2009) stated that programming is a way to foster students' problem-solving skills and computational thinking.

To better understand computational thinking and its connection to programming, it is essential to refer to the four cornerstones concept of computational thinking, namely Decomposition, Pattern recognition, Abstraction, and Algorithm, as shown in Figure 1. Decomposition refers to the thinking of problems, systems, or processes in terms of their inner components. In short, problems can be solved easier by understanding, solving, and evaluating them separately. In other words, breaking down a problem into smaller parts makes them more manageable. Pattern recognition refers to the search for similarities within the problems. Abstraction refers to modelling the problems by capturing their essential properties, grasping their common features, and ignoring their differences. In short, it focuses only on the crucial information and disregards irrelevant details. Algorithm refers to the development of a systematic solution or the rules to solve the problem.

The four cornerstones in Figure 1 are typical steps applied in programming a computer to solve problems. Computational thinking enables a user to conceptualize the instructions for a computer for a given task, while programming is used to instruct the computer to execute the task. Thus, programming is closely related to computational thinking. However, despite its significance, multiple studies have reported that many engineering students lack proficiency in computational thinking, as they are not traditionally exposed to computing in the context of authentic learning experiences that are related to the real-world applications within their field of studies (Magana et al., 2016).

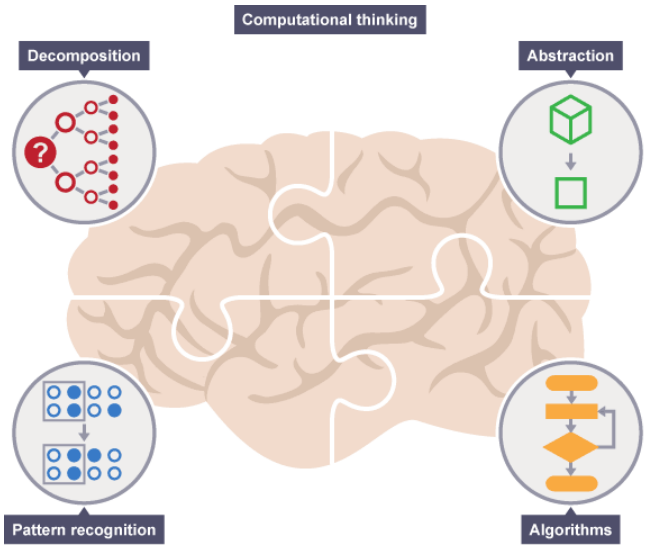

Figure 1. The four cornerstones of computational thinking (BBC, 2018).

Tangible Programming

According to Jean Piaget's constructivist theory, the idea of human's operational stages is initiated from age seven to eleven in the concrete operational stage. At this stage, a user can think logically in terms of objects but will experience difficulties in replacing them with symbols. In other words, humans are capable of solving problems in a logical fashion but lack the capacity to think hypothetically or abstractly. The following formal operational stage enables the replacement of objects with symbols, generalizing and manipulating abstract concepts through proportional reasoning and the derivation of cause-and-effect relationships. Although the theory suggests that the phases transition should occur at the age of 12, a well-established study by Williams and Cavallo (1995) discovered that most college students studying physics courses, including engineering, are still in the concrete operational phase. The study stated that college students lacked the ability to understand abstract concepts that they had never encountered in their previous experience, which led to the detainment of their logical thinking in the concrete operational phase. The manipulation of physical objects by play method offers a constructive and effective way of assisting engineering students in grasping abstract concepts involving programming and computational thinking ability.

Concept and Theory

By definition, a programming language is a set of grammatical rules for instructing a device such as a computer or a mobile phone to perform specific tasks. Whenever programming for beginners is mentioned, two critical topics must be considered: programming syntax and semantics. Syntax refers to the structure or form of expressions, statements, and program units, while semantics refers to the meaning of these elements. In other words, correct syntax enables the program to be compiled successfully, while correct semantics enables the program to work as intended.

As the majority of the engineering undergraduates are unfamiliar with the computer programming syntax, learning programming for these students is easier using visualisation and physical concepts than using text. Recent studies have shown that syntax is indomitable to novice students (Stefik and Siebert, 2013). Besides, there is a large variety of programming languages such as Python, C, Java, and Ruby with different programming syntax, making it difficult for students with less experience to learn to program.

On the other hand, tangible programming blocks form a physical program language on their own. Tangible programming blocks focus on the semantics part of programming to emphasise the meaning of the expressions and the statements they represent instead of the syntax. For engineering diploma students who lack adequate computational thinking and problem-solving skills, semantics

should be prioritised over syntax. Such priority would train them to think logically and structurally without emphasising the correct words and expressions used. Once the abstract concepts are grasped, and the students can correctly understand the logic to build a program, they are ready to learn the syntaxes of different programming languages.

A study conducted in Japan by Saito, Washizaki, and Fukazawa (2016) from Waseda University pointed out that visual input programming induces a more significant change in the students' attitude towards programming. Learners may develop preconceptions that text is more representative of programming and is more difficult (Saito et al., 2016). Physical block-based programming will never be as effective as text-based programming as the users cannot directly alter the code, resulting in fewer manipulations and freedom in learning programming. However, the advantage of physical block-based programming is its ability to help novices advance in their transition period from tangible programming to text-based programming.

METHODOLOGY

Software

The main development software selected for this project is Arduino as it has an easy syntax, and it is an open-source platform that supports many components such as TFT LCD touch screens, keypads, and other necessary hardware. The other supporting software includes Fritzing for circuit design and diagram drawing, Draw.io for program flow charts, and Protel for printed circuit board design after prototyping.

Hardware

The required hardware includes Arduino development boards such as Arduino Mega 2560 and Arduino Uno R3, 2.8-inch TFT LCD touch screens and shields, LCD keypads and shields, 7805 5V voltage regulators, capacitors, resistors, LiPo batteries, connector wires and various sensors such as infrared sensors, mini microphone modules, and light sensors. The Arduino development boards are selected to support the TFT LCD touch screen and transmit data. The LCD keypads and shields are selected for blocks that do not need complex settings, such as the arithmetic blocks that consist of only four operator types.

Transforming Abstract Concepts into Physical Concepts

The programming stage has five basic elements: Input, Conditions, Arithmetic, Loop, and Output. Operations of each element will be translated into physical blocks, as shown in Figure 2. Users will need to go through three different examples to understand the program structure before attempting to solve various simplified real-life problems using the available blocks. The difficulty level of the question can be elevated further to stimulate computational thinking and problem-solving skills.

Translation into Hardware

A brain board will be created using an Arduino Mega 2560 with a TFT LCD touch screen and shield. The brain board will prompt the user to select examples to study and display questions to solve. Referring to Figure 2, the input blocks (red) can be in the form of sensor blocks or blocks with LCD and keypad shields. The input blocks can input signals (sensors HIGH or LOW), numbers (LCD and keypad shield), or words (TFT LCD touch screen). The condition blocks (blue) will have attached TFT LCD touch screens, which will handle the if/else statements and the output transmission. Although the programming conditions usually revolve around the if/else and switch statements, in this project, all condition blocks are configured with only the if-then-else statements to enable easier understanding by the target students. The general logic of the condition blocks is as follows:

$$If\ (parameter\ determined\ by\ user == 1), \qquad then\ (output\ 1\ or\ 0\ decided\ by\ user)$$

$$else\ (action\ determined\ by\ user, can\ be\ null)$$

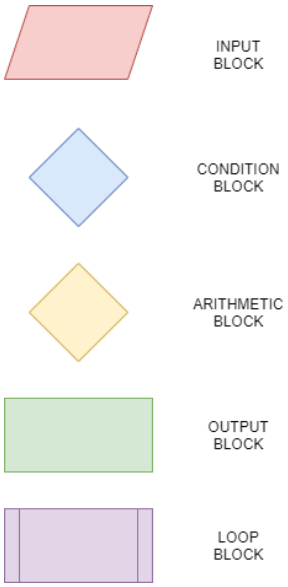

Figure 2. Types of blocks corresponding to each basic programming element.

A condition block will usually be followed by an input block, for example, a push-button input block that produces digital signals HIGH or LOW, which will then give an output signal to an output block with a value of 1 or 0 as decided by the user. The outputs 1 or 0 can also be used in conjunction with other blocks to trigger other conditions. For example, in connecting an arithmetic block with a conditional block, the user can program the arithmetic block so that an addition will be performed if 1 is received. Otherwise, a subtraction will take place. The output blocks (green) can be in the form of actuator blocks (servomotors, buzzers LEDs) or display output blocks (TFT LCD touch screens to display numbers or texts). An output block can be directly connected to an input block or connected to an input block cascaded with a conditional block.

The arithmetic blocks (yellow) allow the user to choose from four different arithmetic operators (+, -, × and ÷). Appearance-wise, the arithmetic blocks are blocks with an LCD and keypad shield. The loop blocks (purple) are pairs of blocks that can perform a loop action on a segment or the entire program n times or indefinitely, where n is a positive integer determined by the user. They are either placed right in front of a brain board and the display output block or made to enclose other blocks. For example, in looping a mathematical process, an arithmetic block and two input blocks will be placed in between the loop blocks.

Users can connect the various blocks, and the final structure of the combination will resemble a computer program structure that takes after the program structure of an Arduino program. Arduino is selected as the model for the program structure for the following reasons:

a. Arduino is easier to learn as its “programming language” is a simplified version of C/C++ with some domain-specific libraries.
b. This project focuses on the interaction of the user and the physical blocks, not the text program itself. In this aspect, Arduino simplifies the interfacing of the various sensors and actuators.
c. Arduino is a comprehensive open-source platform that is compatible with many types of sensors, actuators, development boards, and various components such as keypad shields and TFT LCD touch screens available in the market.

For the finished product, the blocks will be enclosed in junction boxes. For simplicity, ready-made PVC boxes will be purchased instead of constructed using 3D modelling and printing. PVC boxes are selected to house the components as they are strong enough to protect the circuit board and the components inside and are easily modified and customized.

RESEARCH PLANNING AND DISTRIBUTION OF TASKS

There are two major stages in the proposed project. The first stage involves prototyping and initial testing of the blocks. Upon completion of the prototyping and initial testing, in the second stage, the blocks will be sent for applications by selected engineering diploma students in Universiti Tenaga Nasional (UNITEN). The students' feedbacks will be collected and applied to improve the blocks.

The first stage involves four consecutive phases, as described in Figure 3. The first phase is research on computational thinking. The second phase involves determining a suitable programming approach that can foster computational thinking among engineering diploma students. Diploma level students need physical objects to help them grasp abstract programming concepts. However, at present, there are limited physical programming architectures that are suitable for diploma level students. Hence, the third phase involves research on tangible programming blocks. Finally, in the fourth phase, the first stage is concluded by developing and building a physical programming block set. Other project-related tasks such as report writing and components purchasing are uniformly distributed throughout the project.

A physical programming block architecture will be created using Arduino development boards as the bases. Due to its simplicity and its compatibility with many components, the main software programming platform for the blocks will be Arduino IDE. There are five types of blocks: input, output, arithmetic, condition, and loop. After the prototype is completed and test results are obtained, soldering will be required as the components on printed circuit boards are lightweight and neatly packed. The circuit boards will then be attached and enclosed by PVC junction boxes.

The second stage of the research will involve the direct application of the physical programming blocks by selected engineering diploma students. Before the application, a training module with relevant appropriate content, instructions, and hands-on exercises will be developed accordingly. The developed module content will be evaluated and validated by a panel of experts. The students will undergo training on the correct application of the programming blocks that will require them to complete the exercises provided. To determine the module reliability, a feedback form will be developed to collect the students' evaluation of the programming blocks. Based on the feedback collected, improvements to the programming blocks will be planned and performed.

CONCLUSION

Reviews on the available literature show that current engineering students lack computational thinking and that there are limited suitable physical programming educational aids available for them in the market. As a potential solution, this research proposes a physical block-based programming educational tool for undergraduate students. The physical blocks will consist of decision-making, looping, and input/output functions. The Arduino Software IDE will be the main programming platform for this project due to its open-source nature, capability to support various components, and the vast number of third-party libraries. For the hardware, Arduino development boards will be used as they are compatible with many components, have multiple input/output pins compared to other microcontrollers, and are relatively inexpensive compared to other microcontroller boards such as BeagleBone Black or Raspberry Pi. Using the play method, the developed blocks will enable users to build their logical thinking and problem-solving skills through cause-and-effect analysis.

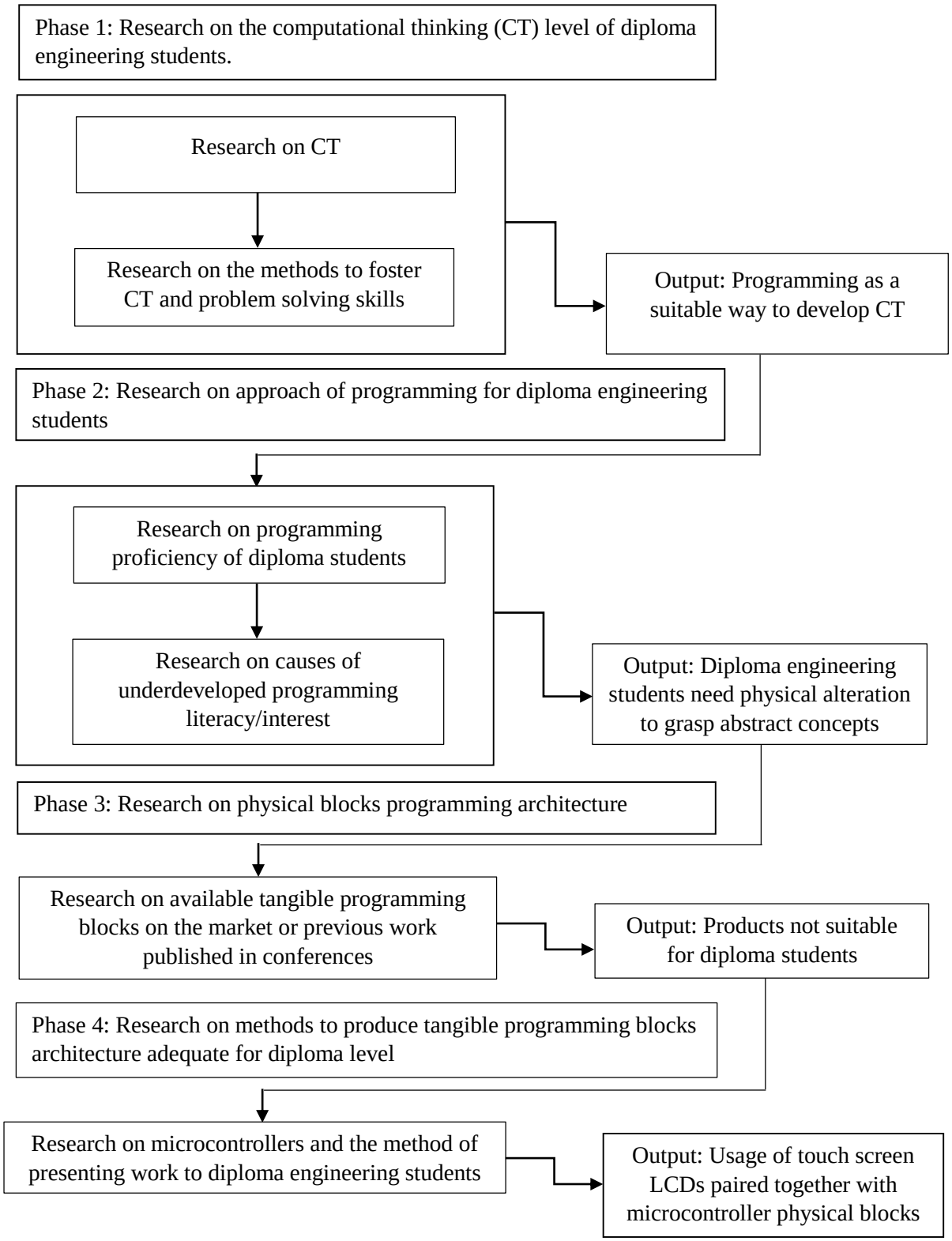

Figure 3. Research phase and planning.

REFERENCES

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33–39.

diSessa, A. A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning*, 20(1), 3–31.

Dishman L. (2016). Why Coding Is Still The Most Important Job Skill Of The Future. Can be accessed at https://www.fastcompany.com/3060883/why-coding-is-the-job-skill-of-the-future-for-everyone

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43.

Hooda S. (2017). #3 Reasons Why Everyone Should Learn Programming. Can be accessed at https://www.entrepreneur.com/article/289248

Kaur B. (2019). Shortage of STEM students at universities. Can be accessed at https://www.nst.com.my/news/nation/2019/09/518914/shortage-stem-students-universities

Lembaga Peperiksaan. (2021). Format Pentaksiran Sijil Pelajaran Malaysia SPM Mulai Tahun2021.accessed July 2021.http://lp.moe.gov.my/index.php/peperiksaan-pentaksiran/peringkat-menengah-atas/sijil-pelajaran-Malaysia-spm

Li, Y., Schoenfeld, A. H., diSessa, A. A., Grasser, A. C., Benson, L. C., English, L. D., & Duschl, R. A.(2020a). Computational thinking is more about thinking than computing. *Journal for STEM Education Research*, 3(1), 1–18.

Lockwood, J., & Mooney, A. (2017). Computational Thinking in Education: Where does it fit? A systematic literary review. https://arxiv.org/abs/1703.07659

Magana, A.J., Falk, M.L., Vieira, C., & Reese Jr, M.J. (2016). A case study of undergraduate engineering students' computational literacy and self-beliefs about computing in the context of authentic practices. *Computers in Human Behavior*, 61, 427-442.

McFadden C. (2019). Code Literacy: Why Coding Became Important. Can be accessed at https://interestingengineering.com/code-literacy-why-coding-became-important

Mutiawani, V. & Juwita. 2014, November. Developing e-learning application specifically designed forlearning introductory programming. In *2014 International Conference on Information Technology Systems and Innovation (ICITSI)* (pp. 126-129).

National Research Council. (2010). Committee for the Workshops on Computational Thinking: Report of a workshop on the scope and nature of computational thinking. Washington, D.C: National Academies Press.

Orr, G. (2009). Computational thinking through programming and omithmic art. In *SIGGRAPH 2009: Talks* (pp. 1-1).

Saito, D., Washizaki, H., and Fukazawa, Y. (2016). Comparison of text-based and visual-based programming input methods for first-time learners. *Journal of Information Technology Education: Research*, 16(1), 209-226.

Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education (TOCE)*, 13(4), 1-40.

Tan, P.H., Ting, C.Y., & Ling, S.W. (2009). Learning difficulties in programming courses: undergraduates' perspective and perception. In *2009 International Conference on Computer Technology and Development* (Vol. 1, pp. 42-46). IEEE.

Williams, K.A., and Cavallo, A.M. (1995). Reasoning Ability, Meaningful Learning, and Students' Understanding of Physics Concepts: Relating Students' Reasoning Ability and Learning Styles To heir Physics Misconceptions. *Journal of College Science Teaching*, 311-314.

Wing, J.M. (2011). Research notebook: Computational thinking – What and why. *The link magazine*, *6*.

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.