Journal of Social Sciences of Mus Alparslan University

anemon

Derginin ana sayfası: http://dergipark.gov.tr/anemon

Araştırma Makalesi ● Research Article

An Overview of Computational Thinking

Bilgi İşlemsel Düşünme Üzerine Genel Bir Bakış

Erhan Yokuş[*], Recep Kahramanoğlu[**]

Öz: Bilgisayarlar ve bilişim teknolojileri giderek artan bir hızda hayatımızın vazgeçilmez öğeleri haline gelmektedirler. Teknolojik donanımlar ve yazılımlar mühendislik, iş dünyası, sağlık sektörü gibi pek çok alanın yanında eğitim alanında da bilişim teknolojileri oldukça yaygın olarak kullanılmaktadır. Bu teknolojilerin verimli şekilde kullanılması için öğrenenlerin bilgisayarların çalışma mantığını anlamaları önem arz etmektedir. Bunu sağlamak amacıyla geliştirilen yöntemlerden biri de bilgi işlemsel düşünmedir. Bilgi işlemsel düşünme görece yeni bir kavram olmasına rağmen hızla gelişen bir kavramdır. Bilgi işlemsel düşünme bilgisayar biliminin temel kavramlarını kullanarak problem çözmeyi, sistem tasarlamayı ve insan davranışını anlamlandırmayı içeren bir düşünme şekli olarak tanımlanabilir. Bu çalışmanın amacı alanyazından kaynaklarla bilgi işlemsel düşünmenin çeşitli boyutlarını ele almaktır. Bu kapsamda öncelikle bilgi işlemsel düşünmenin tanımları üzerinde durulmuş, daha sonrasında bilgi işlemsel düşünmenin tarihi ve gelişim süreci hakkında bilgi verilmiştir. Bilgi işlemsel düşünmeyi oluşturan boyutlar ve eleştirel düşünme, yartıcı düşünme, algoritmik düşünme gibi bilgi işlemsel düşünme ile ilgili yeterliklere yer verilmiştir. Son olarak Google Education, Bebras, Code.org gibi bilgi işlemsel düşünme üzerine uygulamalar hakkında bilgi verilmiştir.

Anahtar Kelimeler: Bilgi işlemsel düşünme, Bilişim teknolojileri, Eleştirel düşünme, Yaratıcı düşünme

Abstract: Computers and information technologies are increasingly becoming indispensable elements of our lives. Technological hardware and softwares are widely used in the field of education as well as in many fields such as engineering, business world, and health sector. In order to use these technologies efficiently, it is important for learners to understand the working logic of computers. One of the methods developed to achieve this is computational thinking. Although computational thinking is a relatively new concept, it is a rapidly developing concept. Computational thinking can be defined as a way of thinking that involves solving problems, designing systems, and making sense of human behavior using the basic concepts of computer science. The aim of this study is to discuss various dimensions of computational thinking with sources from the literature. In this context, firstly, the definitions of computational thinking were emphasized, and then information about the history and development process of computational thinking was given. The dimensions of computational thinking and the competencies related to computational thinking such as critical thinking, creative thinking, and algorithmic

[*] Arş. Gör., Gaziantep Üniversitesi, Gaziantep Eğitim Fakültesi, Eğitim Bilimleri Bölümü
ORCID: 0000-0001-9909-4550, eyokus@gmail.com (Sorumlu yazar)
[**] Doç. Dr., Gaziantep Üniversitesi, Nizip Eğitim Fakültesi, Eğitim Bilimleri Bölümü
ORCID: 0000-0001-6670-8165, recepkahramanoglu@gmail.com
Cite as/ Atıf: Yokuş, E. & Kahramanoğlu, R. (2022). An overview of computational thinking. *Anemon Muş Alparslan Üniversitesi Sosyal Bilimler Dergisi*, *10*(1), 157-173. http://dx.doi.org/10.18506/anemon.1033403
Received/Geliş: 06 December/Aralık 2021
Accepted/Kabul: 29 December/Aralık 2021
Published/Yayın: 30 April/Nisan 2022

e-ISSN: 2149-4622.

thinking are included. Finally, information is given about applications on computational thinking such as Google Education, Bebras, and Code.org.

Keywords: Computational thinking, Information technologies, Critical thinking, Creative thinking

Introduction

Although computational thinking is a relatively new concept, it is a rapidly developing concept. Although it has different definitions, it does not have a clear definition. Wing (2006, p.33) defines computational thinking as "computational thinking involves solving problems, designing systems, and making sense of human behavior using the basic concepts of computer science". In this article Wing wanted to ensure computational thinking, reformulation of large and difficult problems as easier and more solvable, recursive thinking, analysis with abstraction, separation of focal points, being able to describe system behaviour by using invariants, using heuristic reasoning, thinking like a computer scientist and describing the thinking of multiple abstractions together (Wing, 2006).

Wing (2006) emphasized that computational thinking is necessary not only for students at all levels, but also for computer scientists. In this context, he suggested opening a "Thinking Like a Computer Scientist" course for university students. Today, with the increasing interest in computer science and the introduction of informatics courses in education levels, there is a positive increase in this trend. The aim of Wing is to draw attention to computational thinking both in education and in every part of society.

Working together in 2010, Cuny, Snyder, and Wing redefined computational thinking. According to this definition, computational thinking; It is expressed as “the thinking process that involves formulating problems and their solutions in order to present solutions in a way that can be fulfilled effectively by a computing” (Wing, 2011). In this definition, computational thinking is expressed as a process. This process includes not only solutions but also problems and formulation of these problems. This formulation process has two components. The first of these is that the solutions are in a structure that can be processed by the information processing units, that is, they are established in an algorithm system, while the other is the production of a solution that is not only the solution that produces the desired result, but also an effective solution (Çetin & Toluk-Uçar, 2020).

Contrary to Wing's definition, Denning (2009) argues that computational thinking does not have the effect required for other disciplines. In this context, Denning defends the argument that Wing's newly put forward information has actually been used in other fields of science for decades. Denning, who thought that computers had sufficient processing power according to the needs of the period in the 1980s, focused on how computers would be used in the scientific world in the future. It is claimed that these years are favorable for the initial movement for computational sciences (Çetin & Toluk-Uçar, 2020).

Another thought of Denning (2009) thinks that the reduction of computational thinking to computer science only fits this definition into a narrow field. For this reason, the idea that many scientists actually carry out information processes naturally is adopted. The role of computing here is important in understanding and controlling the process. Artificial (machine-generated) information or natural (DNA) computing is more fundamental than it is in Wing's definition. Computational thinking therefore falls short of describing computer science. Computer science has the feature of being a branch of science that examines both natural and artificial information processes. Denning, who claims that there are seven principles of computing, lists these principles as information processing, communication, coordination, automation, memory, design and evaluation. Also, computational thinking is more practical than a principle. Computer scientists are expected to have a good command of programming, systems thinking, performance monitoring and forecasting, and design approach, apart from their computing skills. As a result, although computer science is one of the basic practices of computer science, it does not only belong to computer science and it is not correct to describe computer science using it (Çetin & Toluk-Uçar, 2020).

Barr and Stephenson (2011) made another definition by comparing Wing's definition with Papert's understanding of computational thinking. According to Barr, Wing's definition has a one-sided content and requires an approach that includes problem solving. It focuses on the problem that the incoming problem should be solved by the computer, regardless of the field. Unlike Wing's definition, Papert's definition is bidirectional. It aims to focus on a new problem solving strategy that sees computing as a different problem solving approach. Computers help to see problems from different aspects. Therefore, computers act as assistants with information-processing characteristics during the information-building phase.

Computational Thinking in the History of Education

The term computational thinking was first used by Seymour Papert (Harel & Papert, 1991). While discussing how the computer can be used in geometry problems, Papert (1996) explained the relationship between the problem and its solution, and also pioneered this issue by showing that it can be used in the structuring of knowledge in computational thinking. Constructivism (Papert, 1993) and Action Process Object Schema (APOS) theory (Oktaç & Çetin, 2016) are the first to come to mind in the history of education when commemorating computational thinking. Logo programming language was developed by Papert et al. In the 1960s (Feurzeig & Papert, 2011). LOGO has been designed as a program in which students can easily learn in a short time and perform mathematical and logical operations. Programming languages are suitable for use with different perspectives as a technology in this context (Jonassen, 2000). In another study, Papert states that this program addresses structuralism, starting from a personalized learning and thinking approach (Resnick, et al., 1988). In terms of these results, computational thinking reveals the impression that structuralism is at the forefront in the context of constructivism.

According to Piaget (1964), while creating knowledge, the individual transforms the objects by changing them and facilitates learning and understanding with this transformation. In this process, the individual decides how to construct the object (Piaget, 1964). Through these programs, students can learn by concretizing abstract concepts. Here, the learning tool can consist of the codes contained in the program. With coding, mathematical and logical thinking styles can be developed. Although the logo language and the constructivist structure in its background were seen as a tool with a transformative potential in education, it lost its effect in British and American schools (Agalianos et al., 2001).

Basic Components of Computational Thinking

There is no single definition of computational thinking, nor is there a consensus on the basic components determined as standard. When the literature is examined, it is possible to mention that there are certain common concepts put forward by different authors (ISTE, 2011; Wing, 2006; Kalelioğlu et al., 2016). In order to convey the components in the literature as much as possible, sub-titles of abstraction, algorithmic thinking, evaluation, problem solving, separating the problem into components, pattern identification and generalization will be given (Çetin & Toluk-Uçar, 2020).

Understanding the Problem

In order to solve a problem, first of all, it is necessary to know and understand the problem, it means being able to define it. Although mathematical problems are the first to come to mind when the word problem is mentioned, it is known that there are basic skills related to problem solving in all disciplines. In order to fully express the problem solving process, the concept of the problem must first be understood correctly. The problem is defined by Dewey as “what confuses people, challenges them, and creates uncertainty about what is believed to be true” (Baykul, 2014). When considered in general, it is stated that the problem creates discomfort, contains uncertainty, solutions cannot be clearly seen, and mental or physical work is needed to eliminate these uncertainties. However, the main point here is that in order for a problem to be truly considered a problem, it must also be a problem for another person. Therefore, there is a need for classification of problems. The most common classification is to make a distinction between routine and non-routine problems.

Routine problems: These are problems that can be solved with previously learned rules, algorithms or formulas. These problems become practical for new learners, continue according to certain rules, internalize the relevant rule more quickly according to new problems and minimize the contribution of the situations used over time.

Non-routine problems: Defined as problems with known methods, techniques or formulas that have invisible solutions, require more thought than routine problems, and also allow the use of strategy on a case-by-case basis. These types of problems require high-level skills such as transfer and analysis, which enable students to develop their creative aspects, to think, to establish relationships, to search for patterns. Therefore, it is expected to be preferred more than routine problems (Altun, 2014).

It is put forward with the aim of finding a solution to the routine or non-routine problems. Before resorting to the path to solving problems, that problem must be diagnosed. Regardless of the discipline, it is essential to determine whether the problem is a problem or not. In this first step, which is expressed as understanding the problem, the content of the problem is revealed in all its aspects. Before proceeding to the solution, data about the problem is collected, analyzed and a preliminary presentation is made. The first step in problem solving is to identify and understand the problem. At this stage, the steps of identification and analysis, data collection, analysis and presentation of data are followed.

Decomposition

It is not always possible to see a big problem as it is. Generally, breaking problems down into smaller parts makes it easier and simpler to see the problem. In computer language, this subdivision is called a module. The modules of each problem are handled separately. Solutions are developed and presented for each module. These solutions are tested on modules. Then, each component is brought together and the necessary solutions are obtained for the solution of the big problem. This approach can be used in the development and modulation of complex software such as SPSS. An algorithm written in a way that does not exceed ten lines can be used in the development and use of such complex software. The main point here is that the problem can be separated into its separate components, the solution of each part must be solved independently of the other, and then these components must be able to be brought together (Liskov & Guttag, 2000).

Separating the problem into its components is often the result of a divide-and-conquer strategy. This is not just a case of coding. A divide-and-conquer strategy can be used in many areas. For example, in the definition of computational thinking, this strategy can be used to create a divide-and-conquer strategy and a computational thinking concept. To give a similar example through qualitative data analysis, it is very difficult to grasp the data as it is in this analysis technique. However, by making content analysis, the data is categorized according to its components, and then when these components are brought together, qualitative data can be made sense of (Çetin & Toluk-Uçar, 2020).

Abstraction

Although abstraction is different from breaking a problem down into its components, it is interrelated (Liskov & Guttag, 2000). The ability to abstract can be used efficiently to break down problems into their components. Abstraction basically makes a problem simpler by ignoring certain details. Let's take a play for example. The details of the writing of the play are abstracted, and the sections, sub-titles, content, subject, and how many acts of the play are roughly determined. These operations are done without writing the dialogs. Although the main problem here, the writing of the play, remains the same, the solution is simplified as the components are now ready. Abstraction helps computer scientists break down problems into their components. Also, unlike the considerations of Likov and Guttag, abstraction has other interpretations for computer science. From whatever angle one looks at abstraction, it is clear that it is among the basic concepts of computer science (Çetin & Toluk-Uçar, 2020).

Three different meanings of abstraction are emphasized. Its first meaning refers to the neglect of certain details. From this point of view, while ignoring certain features of the focused problem, two

different things are made the same by highlighting some distinctive features. For example, when a child's awareness of the physical characteristics of a cat is considered, and when it is considered that cats have paws, a tail, four legs and meowing when they see a different cat, the characteristics of the cat such as running speed, color and size are ignored. What is done here is to reveal a structure by emphasizing certain features and ignoring some of its features. The second meaning of abstraction relates to the situation. The state of being abstract is relative, not absolute. While a concept may have an abstract meaning for one person, it may have a concrete meaning for another. The third meaning related to abstraction is to be considered in terms of properties. According to this meaning, attention is paid to the properties of the structure rather than objects and operations. For example, the infinity of natural numbers is an appropriate use of abstraction in terms of properties in terms of natural numbers. Similarly, the commutative property of addition operations with natural numbers or the smallest element of natural numbers are examples of meaning in terms of properties (Çetin & Toluk-Uçar, 2020).

When looking at abstraction in terms of computer science, it is mentioned that the abstraction takes place in different layers (Wing, 2006). If a computer scientist were to use a function from the C library, he would ignore the details of its code and use the function. Information about the use of this function, such as what parameters it takes, what its name and function is, whether it returns a value, etc., attracts the attention of computer scientists. The programmer, who is the author of this library function, is not concerned with the algorithm necessary for the function to perform its task, but with the hardware and how the computer will operate this function. Although there is a function of interest here in both cases, the abstraction in different layers for the program writer and the user, that is, the meaning of the abstraction.

Pattern Recognition

The Turkish Language Association defines the pattern as "the development of objects or events following each other in a certain order" (Turkish Language Society, 2020). A pattern is defined as the way a certain data is expressed to recognize similarities, differences or a rule. Similarities or differences help solve problems through modelling. Through these models, it helps to solve complex problems more efficiently (Ministry of National Education, 2020). In the field of computational thinking, the term pattern has a meaning as model creation, modelling, pattern/model extraction.

In the LEGO guide, it is mentioned that the dimension mentioned as pattern recognition and accordingly generalization will provide convenience for preparing algorithms (LEGO Education, 2018). Traffic lights are a very beautiful event that can exemplify this pattern. It is assumed that a certain series of traffic lights continue in an infinite loop and is an example of a pattern in this way (Üzümcü, 2019).

It also appears as repetitive uses expressed as computer or machine learning in the field of artificial intelligence. Here it is used as an expression of automation (Wing, 2008). Robot programming, game design or other programming activities can be given as examples. In addition to these, automation is widely used in daily life such as chatting with people from different cultures, reading barcodes, vehicle recognition systems or contactless payment while using internet-based tools. Collecting information through sensors and analyzing them are examples of automation that computers can do (Ministry of National Education, 2020).

Algorithms

The way designed for solving a certain problem or for those who want to achieve a goal is expressed as an algorithm. Commonly used in mathematics and computer science. In this sense, it is a finite set of operations that end in a certain place from a defined starting point for doing a job. It is often used as a computer programming term. There is an algorithm on the basis of all programming languages. At the same time, the algorithm is the step-by-step process of revealing the behaviors, which are the roadmap for solving a problem, the commands or statements that will do the basic work, step by step, and ordering is very important here (Wikipedia, 2020).

In addition to the algorithm, the concept of algorithmic thinking is also included in the studies. Algorithmic thinking is defined as the basic skills that can be used to solve problems related to computer science and applications (Syslo & Kwiatkowska, 2015). Although the algorithm is only used in computer programming, algorithmic thinking has a feature that can be developed independently of programming (Otaran, 2017).

The algorithm is shown among the dimensions of computational thinking. Therefore, its relationship with programming is obvious (Israel et al., 2015). According to some opinions; The relationship between computational thinking and programming is controversial (National Research Council, 2010).

Testing

Performing the necessary tests in order to control what is done after the steps of computational thinking refers to the testing phase. Checking the steps followed in the path taken from the first dimension in computer science is the stage of testing the debugging steps before the evaluation.

Evaluation and Debugging

Computer science is an applied science compared to other disciplines. The knowledge it produces is both a part of daily life and has the opportunity to be applied in other fields. For this reason, it should be taken into account that algorithms produced by computer science have practical limitations as well as institutional constraints. It is not enough to produce an algorithm for the sole purpose of solving a problem. Because the algorithms used in solving this problem have the possibility of using more system resources than necessary. It takes too much time or may not be practical to use. For example, let's say that an algorithm is produced to crack the password of a system. If this algorithm only works for a short time like a year, then lags behind the system, that is, if it becomes outdated, if it lags behind today's technologies, it is considered inefficient in terms of practicality. In this context, evaluation is the process of examining whether the given or produced algorithm is efficient by looking at its adequacy in the solution used (Çetin & Toluk-Uçar, 2020).

According to Schneider and Gersting (2016), it is stated that the efficiency of an algorithm should not only fulfill its job, but also be aesthetically attentive in using system resources economically and easily. It may not be possible to find all of these features in the same algorithm. Sometimes an elegantly designed algorithm is difficult to understand. An algorithm that is difficult to understand may not be practical for its users. On the other hand, an algorithm has been written that is easy to understand but does not have a stylish aesthetic. What should be known is that in an algorithm, agreement, aesthetics and practicality should be known as holistic features that must be together. In this context, the most important feature of the algorithm is efficiency (Sedgewick & Wayne, 2011).

Users do not want to use a program that slows down the system when the computer is started, or that takes more time than necessary. Therefore, efficiency is based on how much memory the algorithm uses in addition to its existing inputs. If the algorithm uses only a few memory areas other than its own inputs, the efficiency of this algorithm can be mentioned. However, if this algorithm wants to occupy as much memory as the input space, then it is difficult to say that it is efficient in terms of space (Çetin & Toluk-Uçar, 2020).

Algorithms are also evaluated in terms of time. In this evaluation, the runtime is tested by running the algorithm on the computer. However, it is difficult to say that this approach is an accurate assessment. At this stage, the algorithm is run on a computer and within certain inputs. Therefore, evaluation and generalization cannot be made for all inputs or the entire computer (Sedgewick & Wayne, 2011).

Competencies Related to Computational Thinking

In computational thinking, it is stated that mental processes should be handled in terms of a certain processing dimension while trying to make sense of experiences that are different from normal thinking

processes. The common purpose of thinking skills; solving problems, making decisions, asking questions and seeking answers, and expressing ideas in the light of plans or in an organized way (Taylor, 1959).

One of the requirements of the 2000s we live in is related to the skills that individuals are expected to have. In this context, there is a number of lists in the literature. These lists include: "21. century thinking skills, digital citizenship and the International Society for Technology in Education (ISTE) lists of standards for students can be given as examples. The standards developed by ISTE for students as a result of the research conducted in 2016 are listed below (ISTE, 2016):

- Empowered Learner,
- Digital Citizen,
- Knowledge Constructor,
- Innovative Designer,
- Computational Thinker,
- Creative Communicator.

As can be seen from the list given above, the skill of "computational thinking" in the 21st century emerges as a quality that everyone should have. The main reason for this is that being a computational thinker is a universal ability as much as a literacy and mathematical skill (Wing, 2006; Wing, 2008).

Creative Thinking

Creativity is defined as a complex construct. It is commonly stated that it is the product of a broad intelligence which is language, music, mathematical, spatial, mental and bodily, both interpersonally and personally products (Gardner, 1985). At the core of creative thinking is the ability to think new things or think in new ways. It is also defined as "thinking outside the box". Discovering the connections between ideas by activating is an important step in the creativity cycle (Lau, 2011). Creativity is a naturally more developed ability in some people, which can develop with practice. People with creative thinking ability fulfill given tasks, solve problems and have the ability to design new ways to deal with difficulties (Lau, 2011). Offering a new way to work or presenting a solution with an unconventional perspective can help to work more efficiently.

Creative thinking has different subtypes. These are listed as creativity in expression, exploratory creativity and renewable creativity. It is the type of creativity in expression that is exemplified in drawings and games that appear spontaneously in children. The actions of scientists and artists demonstrate productive creativity. Innovative creativity is defined as problem solving ability or creation to improve an existing technology. Renewable creativity is more of a product of creative thinking that creates a new paradigm. It is also defined as new ideas emerging through consensus (Baker et al., 2001).

Algorithmic Thinking

An algorithm is defined as "a bundle of clear and effectively performable operations that, when executed, produce results within a finite time frame" (Schneider & Gersting, 2016). The historical process of the word algorithm is based on Al-Harizmi (Knuth, 1985). Al-Harizmi, a mathematician who lived in Central Asia in the 9th century, defines the algorithm as the methods produced for the problem solving process (Sedgewick & Wayne, 2011). An information processing unit undertakes the operation of this method. This does not specifically mean algorithm information processing unit. If the information processing unit is considered as a computer, the algorithm can run in different programming languages, but it can also run different computers. Here, the determinant of the solution of the problem is the method itself rather than the written program.

It is accepted that algorithms have an important place in computer science (Schneider & Gersting, 2016). In particular, automation systems have made work easier in many areas, enabling both time and

cost savings. For example, it provides great convenience for various exams held throughout the country to be concluded in a short time by an information processing unit and with a robust algorithm running in the background, and the controls can be made quickly. In addition, it should not be forgotten that data automation occupies an important place in human life while transactions are carried out without error rate thanks to the debugging system.

Algorithmic thinking means different things in different disciplines. According to Knuth (1985), algorithmic thinking means computational thinking. In mathematics education, it is used to mean operational thinking as opposed to creative thinking (Jonsson et al., 2014). In the current sense, algorithmic thinking is expressed in the literature as consisting of the solution steps produced by the computing unit in the solution of certain problems. It is a way of making sense and a skill that consists of rules and the steps necessary for the operation of these rules (Csizmadia, et al., 2015). In another definition, it is explained as "different skills used to construct a new algorithm and understand an existing algorithm" (Futschek, 2006). These skills are listed as follows:

- Ability to analyze the given problem,
- Ability to express the problem clearly,
- Ability to identify the basic operations required for the given problem,
- Ability to build the right algorithm through basic operations,
- Ability to handle the problem in terms of all possible special and general situations,
- The ability to increase the efficiency of an algorithm.

Futschek (2006) is of the opinion that one should not be dependent on a programming language for the development of algorithmic thinking. He complains that students do not have enough time to focus on programming features and design algorithms. For this reason, it is of the opinion that it would be more appropriate for students to develop their algorithmic thinking by using pseudo-code and dealing with algorithms suitable for their level (Futschek, 2006).

The computer-free computer science movement (Henderson, 2008) proposes the idea of developing algorithmic thinking without using computers. It is the Bilge Kunduz activity, which is an activity developed on computational thinking skills without using a computer (https://www.bebras.org/). These events are held in Turkey.

Collaborative Work

Individuals who have a good command of computer technology can also go a long way in terms of learning. The transfer of almost everything to digital environments necessitates both digital transformation and computational learning. In this context, development can be achieved by adding new dimensions to the content of learning. Computational thinking skill is important as an individual learning as well as a group skill. Because it is known that individuals who can work as a team contribute to each other and to the field. There are also working methods that can come together with correct and effective thinking and produce ideas, understand the problem and follow the solution path. The definition of cooperative learning is that group members encourage each other to understand the material to be learned and work more, and share information by discussing the issues among them. From this point of view, in terms of computational thinking competencies, it is possible to make a union of ideas and labor in the process of creating an algorithm and at every step until the solution of the problem is finalized. The purpose of these joint works, also called collaborative work, is to ensure success within the group as well as the individual responsibility of each individual, and to provide the expected contribution to the solution of the problem. This way of working will both increase interaction and contribute to success in the computational and algorithmic thinking process.

Critical Thinking

Critical thinking is a way of thinking that consists of mental processes such as reasoning, analysis, and evaluation. Sometimes the terms "discussion logic" or "informal logic" are used instead of the concept of critical thinking (Allen, 2004). It is based on critical thinking, questioning and skepticism. It also includes the processes of thinking on concrete or abstract concepts to arrive at clear judgments that are consistent with common sense and scientific evidence. In this respect, it complements creative thinking, which is known as another way of thinking (Wikipedia, 2020).

Critical thinking is a process consisting of five stages. It starts with the step of determining the subject to be discussed first. The next step is the collection of data, information and evidence on the subject. Arguments and viewpoints that differ are considered and examined. The assumptions, data, information and evidence on which the examined arguments are based are reviewed. The findings obtained in the second step are subjected to an evaluation in the third step. Unnecessary or invalid ones are weeded out. For the rest, similarities and differences between them are identified, and connections and relationships are defined. The fourth step is to put the remaining data, information and evidence into a logical framework. In the fifth and final step, the conclusions reached within the logical framework are presented. Finally, a judgment is made in the context of the validity and robustness of the results (Moore & Parker, 2008).

Critical thinking forms a whole from all senses, written and verbal expressions, data obtained by observation, experiment or reasoning technique. In order to be a critical thinker, it is necessary to have some critical thinking skills. These skills are sorting, connecting, analyzing, criticizing, matching, forming hypotheses, comparing, clustering, establishing cause-effect relationships, predicting, identifying patterns, distinguishing exceptions, planning, synthesizing, classifying, sequencing, deduction and induction, data collection and preparation for the analysis process (Şahinel, 2002).

Problem Solving

Problem solving is a term used to express “the situations in which reaching from one point to another is the goal and the optimum way of doing it must be chosen” (Wikipedia, 2020). Problem solving approaches, on the other hand, started to develop with cognitive psychology in the 1970s and were previously only associated with perception and memory use skills (Quesada et al., 2005). The most influential, striking and important work in this field was done by Newell and Simon (1972). In this study, it is aimed to explain the general processes of problem solving. In the problem solving process, the knowledge role of the individual was minimized and it was tried to focus on the problem types. In the process from the 1990s to the present, especially as a result of the increasing effects of interdisciplinary studies on social life, problem solving has moved away from its conceptual meaning and has turned into a new concept that includes cognitive activity (Çetin & Toluk-Uçar, 2020).

According to the view of Jonassen (2000); in information processing theory, it is not enough to look at the prerequisites for problem solving only as learning, and each problem is different from each other. Studies also support the view that problem solving strategies are insufficient in solving non-routine problems (Mayer, 1998). In addition, Jonassen stated that the efforts to develop a theory to solve a single problem, regardless of the field advocated by the information processing approaches of the 1970s, were not sufficient (Smith, 1991). He claimed that it is not appropriate to offer a uniform solution because the problems are not equivalent in content, form and structure. On the other hand, according to the theory of problem solving through schemas, he argues that the possibility of problem solving may be valid for certain types of problems. Mayer (1992), on the other hand, states that the problem solver is significantly affected by this scheme, depending on the type of problem, in order to perceive the problem. The schema created by the problem solver is based on previous problem solving experiences. Having a schema for problem solving helps a person to go to the solution by applying this schema to the relevant parts of the problem. Researches show that expert problem solvers have a higher ability to remember the schemas and reach the solution in a shorter time by recognizing the problem type, while novice solvers tend to use general problem solving strategies suggested by information processing theory due to their insufficient schema experience in this context (Sweller, 1988; Mayer, 1992).

In order to be successful in problem solving, it is necessary to have different knowledge and skills. These skills are; familiarity with problem types, content knowledge, metacognitive skills, heuristic knowledge, certain attitudes and beliefs. These attitudes and beliefs should be directed towards the field, problem solving, and the problem solver himself. Being familiar with the problem type facilitates solving routine problems, but this skill alone cannot be applied to a different type of the same problem. Since routine problems are more familiar types of problems, transfer solution is easier, while non-routine problems are less likely to find solutions. The problem solver's content knowledge contributes more to the problem solver than familiarity (Jonassen, 2000). The level of knowledge of the person is effective in understanding the problem and producing a solution. However, what leads to real success is how rich the content knowledge and conceptual skills of the problem solver are (Çetin & Toluk-Uçar, 2020).

Another important concept in problem solving is the heuristic approach. This approach is defined as "the body of knowledge and practical rules gained from previous experiences that can be used to make different choices and evaluations" (Polya, 1957). The heuristic approach, also known as the exploratory approach in the etymological sense, aims to aid discovery. Heuristic strategies consist of practical rules to help a person solve problems, and suggestions for starting and advancing problem solving. In this respect, Polya's (1957) heuristic approach is important. According to this approach, there are main strategies and sub-strategies related to them. This strategic method proceeds by briefly examining specific examples, predicting a pattern, and making a generalization from it. This thought is abstract thought. This abstract way of thinking – conscious or unconscious heuristic thinking – applies to all problem-solving processes. Polya identifies this approach with four key components: understanding the problem, creating a plan, implementing the plan, and checking for the final solution.

Another important factor in problem solving is metacognition. Metacognition is defined by Flavell (1979) as one's learning style, an individual's ability to perceive the difficulty of a task, how they use information to reach a goal, their comprehension skills, and their awareness of how their learning progresses. In other words, it is expressed as the ability to manage one's own thought process. Individuals with good problem-solving skills try to clearly state their goals while solving problems. They can understand the relationship between concepts and the components of the problem, follow their own understandings and make evaluations by choosing the paths that will lead to the goal. It is seen that people with advanced metacognitive representation code problems more easily and they overcome problems by strategically evaluating the mental processes that represent the problems (Schoenfeld, 1987). According to this view, it has been observed that experts spend more time in the analysis and validation stages, while novices spend more time in applying familiar rules without being sure that they have chosen the right solution.

Another important point to be examined in relation to problem solving is the problem itself. According to Polya's (1957) approach, problem solving is much more than simple calculation. A problem is a challenge, an obstacle, or a thinking process that is equivalent to research. In order for a situation to be defined as a problem in the problem solving process, the solution must not be known beforehand. In this direction, Jonassen (2000) defines the problem as "something unknown" and problem solving as "finding the unknown" and emphasizes that the solution is not known beforehand.

Applications Related to Computational Thinking

When the concepts and approaches in the field of computational thinking are examined, computer science is seen as one of the most important building blocks of the 21st century. Various approaches and arrangements are being made on computer technologies that provide great convenience in the field of education, save time and space, and enable faster interaction. There are some applications that are at the beginning of these. The nature of these activities is generally organized according to the purpose of game-based learning and learning by discovery. In computer science teaching, teaching can be done in a more fun and different way in the light of the basic philosophy of computer science. The most familiar counterpart of this teaching method includes many educational methods developed as "Computerless Computer Science (B3)" teaching and known as Computer Science Unplugged in English. The aim of

these activities is to teach the concept of computer science and computational thinking, and to create content that can be constructed by combining different materials in indoor and outdoor activities, practices (Çetin & Toluk-Uçar, 2020). The fact that this approach strives to motivate students with a pedagogical educational approach in which computer science can be explained verbally and to teach computer science from a young age makes significant contributions to the field.

When computational thinking education and applications are examined, it is necessary to automate programming and teaching by using block-based, text-based or physical programming tools. Prior to this, the content and importance of the programming concept should be taught for preliminary preparation. Then, computer support programming can be taught. However, first and foremost, skills such as problem solving, discovery of ways to a solution, separation into sub-problems, abstraction, explaining the solution step by step, following the given instructions, testing the solution and debugging the errors should be taught.

Code.org

Code.org; founded in 2013, blockchain-based public and non-profit; is an international organization whose aim is to increase participation in computer science, whose target audience is young women and underrepresented group students. It leads the way in its activities with the campaigns that unite 10% of the students from around the world every year with the organizations it organizes around the world. It has big supporters such as Microsoft, Google Education, Facebook, Amazon. The main goal of Code.org is for every student to have access to the opportunity to learn computer science, regardless of school. The user base of Code.org courses consists of tens of millions of students around the world and more than one million teachers who support them (https://code.org, 2020).

Code.org has video lectures and working practices arranged for different age groups on its website. Having so many options in terms of language makes it easier for its users to reach its target audience and for its users. Within the scope of teaching activities, there are topics such as computational thinking, abstraction, algorithm, debugging, functions, events, loops and variables.

The Barefoot Project

It is a project implemented in 2014, developed by teachers and adapted for computer science curricula (https://www.barefootcomputing.org, 2020). This project empowers a variety of free helpful online guides for primary school teachers in the UK, presentations that will make the computer curriculum more engaging. Alongside the teachers, the project aims to inspire students' development by helping them think and learn in a digital environment.

Google Education

Google Education is content developed by the Google company that offers professional support through educators. Its aim is to bring learning to everyone and to support education. It demonstrates its digital contribution to education through products, programs and free courses. It allows educators and students to carry out their studies with joint learning and innovation processes. It offers opportunities such as extensive content, digital tools, study materials, coding lessons at various levels from primary to higher education for all levels of education. It has a rich and free content of coding and computer science education, especially as resources and tools created by Google to develop creativity, encourage hands-on education and provide students with digital skills. It allows students to produce their own ideas. It supports projects that aim to close the global inequality of opportunity gap in education. Technology transforms teaching and learning processes and supports this process for free. It allows children to learn subjects at their own learning pace. It also helps them grow up as individuals who can solve problems and demonstrate their productivity in collaborative work (Google Education, 2020)

The courses consist of five parts. These sections start with computational thinking and continue with algorithm steps. While pattern finding and diagnosis continues with algorithm development, it ends as the final project with the step of applying computational thinking.

BBC Bitesize

BBC Bitesize is an education support website developed by the BBC in 1998 for students studying in the UK. He has detailed and extensive training on computational thinking in the field of computers. It is a free content designed specifically to support students in reviewing, repeating, and preparing homework. Bitesize provides support to students aged 5-16 in a variety of school subjects. It also considers the well-being of both children and young people and supports their career choices. In addition, Bitesize has hundreds of published course content to help UK students with homeschooling (www.bbc.com, 2020).

Bitesize's guides are written by teachers and subject matter experts. Content is mapped and published to follow the UK curriculum. Secondary Bitesize follows the exam board specifications of major UK exam boards including GCSE in England, Wales and Northern Ireland and National 4, National 5 and Highers in Scotland. With this aspect, it enables students to prepare for the right exams (www.bbc.com, 2020). “How do I use Bitesize?” on the site for help for students and parents. There are road maps with option. Bitesize can also be used by parents to help children learn. It can also be used independently by students to assist with homework and exam revision. Every student, whether in primary or secondary school, always has access at home or at school.

LEGO Education

LEGO Education is an educational approach based on hands-on training and learning that engages students as active participants in their own learning processes. It was founded by the LEGO group and has been in service for over 80 years. It is an application that has developed itself with studies that serve teachers and students in the context of education for about 40 years. It contributes to making learning fun and effective learning process (www.robokids.com.tr, 2020).

The history of lego is older than other applications. In this direction, Lego has been developing various applications and studies on children's learning for many years. The working logic of Lego is to enable children to make inventions through designs and have fun while doing this work. Not only with designs in the field of science and technology, but also with computer-aided trainings, it has developed its field of application. It provides learning and thinking through assembling parts, as well as inclusive training on learning software.

Lego defines coding as a part of computational thinking. For the WeDo 2.0 robotics set, they include computational programs in their curriculum among the project topics. Within the scope of this curriculum, eight different computational thinking lesson plans were prepared. They use field approaches of computational thinking in all STEM courses and other course areas (LEGO Education, 2018). Coding is seen as a tool used to develop computational thinking in the context of STEM. Computational thinking for lifelong application development is included as an 8-step solution path defined within the scope of STEM.

BEBRAS/Wise Beaver

Wise Beaver is an activity that means "Bebras" in the Lithuanian language, that is, the equivalent of the word "kunduz" and is used in Turkish with this meaning. The beaver was chosen as a symbol of activity because it is a creature that makes a lot of effort to reach its goal in life. The term “Wise” was added in front of it to better express the characteristics of beavers, and it was defined as the Wise Beaver.

Wise Beaver is an activity created to teach computer science concepts, computational thinking and to increase the awareness of learners. The main purpose is to direct students' attention to informatics, to think algorithmically, to comprehend logical processes, and to support computer science subjects and teaching (Dagiene & Stupuriene, 2016). The Wise Beaver project organizes an online event in many countries in order to reveal the situation of students regarding computer science concepts in the same period. It organized its first international event in 2004 in Lithuania. After this date, the effect of the event spread to other countries. In 2020, a large-scale event was organized with the participation of more

than 500,000 students. Many European countries have participated in the events since the beginning and these participations still continue. Turkey is among the countries that plan to participate in the event (bilgekunduz.org, 2020).

There are short questions called "Wise Beaver Quests" in the activity. These tasks are prepared with different difficulty levels for different age groups. The peculiarity of these questions is that they can be answered by people who have no prior knowledge of informatics. Learners should review what they know, make calculations, make decisions, establish cause-effect relationships, think analytically and have skills such as problem solving. When the questions are examined more deeply, they include processing processes such as algorithm development, debugging, pattern diagnosis, parallel processing, conditional and logical comparison. The aim here is to learn and question the concepts of computational thinking skills in the context of computer science.

In the Wise Beaver activity, explanations about computational thinking skills were given along with the question subfields. These sub-fields, which start with digital literacy, continue with programming, problem solving and data processing steps. What to do at each step and the results are different. As there is a different skill required at each step, its contribution is also different. They aim to increase students' awareness at all levels and to contribute to their computational thinking skills by increasing their interest in informatics.

Scratch

It is a free project that was launched in 2003 at MIT University's Media Lab, offering support in approximately fifty languages, including Turkish. Scratch is designed for the 8-16 age group. Scratch, the most well-known of the block-based visual programming tools, is inspired by Logo. The application area and target audience of this programming are different from each other. However, despite these differences, it has a wide range of uses. It provides service through its own web page. This website (scratch.mit.edu, 2020) is of great interest. It has a potential of around 20 million users. There are nearly 25 million projects shared over their websites. This program can be used by downloading it to personal computers. It also has an online usage feature. It has the largest user base among block-based programming applications (scratch.mit.edu, 2020).

The main features of block-based programs are that they have a block code structure. This block code structure provides easy learning. It allows for fewer errors and easy debugging. By offering multimedia support, it is more preferred with its design-oriented and online sharing opportunities. The general features of block-based programming tools are that they are simpler and more understandable than text-based programming. While the visual structure of the interface facilitates perception, it also provides fast learning. The tools appeal to user groups of all ages. It is possible to build simple applications and coding, as well as build structures with high design features. Debugging provides logical convenience in these tools, eliminating the problem of writing code. In addition, the fact that it has add-on features such as picture and sound as multimedia support also contributes visually. Not only for students, but also for educators, it has the feature of being both a great application opportunity and a tool to use as a material. The possibility of sharing online shows that users are open to communication and interaction.

While programming with Scratch, its contribution to computational thinking is to support the structure of visual thinking and creating context. It is mentioned that computational thinking has three dimensions for Scratch. These dimensions consist of concepts, practices, and perspectives. The concept dimension forms the basis of programming. Concepts include arrays, loops, events, operators, and data. In the Applications dimension, there are incremental and repetitive events, testing and debugging, recomparison, and finally abstraction and modulation phases. In the perspective dimension of computational thinking, Scratch includes the steps of expressing, establishing relationships and questioning (Üzümcü, 2019). It is possible to state that it is a programming structure that has a rich content in terms of these dimensions.

Conclusion

The act of thinking is a phenomenon that is unique to humans and is realized voluntarily or involuntarily at every moment of human life. Human beings are expected to take decisions in order to survive, and to perform the act of thinking while making these decisions. Each individual has the ability to think differently. For this reason, many studies have been done on thinking and the act of thinking has been tried to be defined. While studies on thinking are mostly in the field of interest of psychologists, the information-processing approach and its applications are also rooted in these studies. The skills of evolving from the importance of thinking skills to the computational thinking process are examined in this study because it is not possible to transition to the computational thinking process without comprehending the act of thinking. How to acquire these skills is the subject of educational sciences.

Computational thinking is not just the subject of computer science. It is claimed that it is a skill that should be known, learned and developed by everyone (Wing, 2006). Wing believes that computational thinking is as basic a phenomenon for every child as math, arithmetic and literacy. This idea broadly includes academics, educators, politicians, and the computer and communication technologies industry, which is in the information world. This interaction mass made it necessary to think about the definition of computational thinking and what its domains are. How computational learning will be done, how it will take place in education and why it is necessary are other factors that are examined. The relationship of computational thinking with other disciplines is also handled in this context and its scope is determined (Çetin & Toluk-Uçar, 2020).

The act of calculating at the root of the word "compute", which means to calculate, has passed into our language as "counting information". In this case, there are various definitions for making sense of the term "computational thinking", which is the semantic equivalent of the expression computational thinking. “Computer thinking” or “computational thinking” is the most used definitional equivalent (Yecan et al., 2017). Rather than defining, it is essential that the concept is full and that what is meant to be conveyed correctly.

References

Agalianos, A., Noss, R., & Whitty, G. (2001). Logo in mainstream schools: The struggle over the soul of an educational innovation. *British Journal of Sociology of Education*, *22*(4), 479-500.

Ahorani, D. (2000). Cotiga, Ergo, Sum! cognitive processes of students dealing with data structures. *ACM SIGCSE Bulletin*, *32*(1), *26*(30).

Allen, M. (2004). *Smart thinking: Skills for critical understanding and writing* (2. Edition). Oxford University Press.

Altun, M. (2014). *Ortaokullarda (5, 6, 7 ve 8. sınıflarda) matematik öğretimi*. Alfa Aktüel.

Baker, M., Rudd, R., & Pomeroy, C. (2001). Relationships between critical and creative thinking. *Journal of Southern Agricultural Education Research*, *51*(1), 173-188.

Barefootcomputing (2020, November 29). *About Barefoot*. https://www.barefootcomputing.org/about-barefoot

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?. *Acm Inroads*, *2*(1), 48-54.

Baykul, Y. (2014). *Ortaokulda matematik öğretimi (5-8. sınıflar).* Pegem Akademi Yayınları.

BBC. (2020, November 30). *BBC Bitesize.* https://www.bbc.co.uk/bitesize/articles/z6x992p

Beth, E. W., & Piaget, J. (1966). *Mathematical epistemology and psychology*. Dordercht: Reidel.

Bilgekunduz (2020, December 01). *Uluslararası Enformatik ve Bilgi İşlemsel Düşünme Etkinliği*. https://bilgekunduz.org/

Code.org (2020, November 29). *About Us*. https://code.org/international/about

Cortina, T. J. (2015). Reaching a broader population of students through unplugged activities. *Communications of the ACM*, *58*(3), *25*(27).

Csizmadia, A., Curzon, P., Dorling , M., Humphreys, S., Ng., T., Selby, C., & Woollard, J. (2015). *Computational thinking: A guide for teachers*. http://community.computingatscholl.org.uk/resources/2324

Çetin, İ. (2015). Students' understanding of loops in computer programming: An APOS theory perspective. *Canadian Journal of Science, Mathematics and Technology Education*, *15*(2), 1098-1111.

Çetin, İ., & Toluk-Uçar, Z. (2020). Bilgi işlemsel düşünme tanımı ve kapsamı. In Gülbahar, Y. (Ed.), *Bilgi işlemsel düşünmeden programlamaya* (pp. 41-78). Pegem Akademi.

Dagiene, V., & Futschek, G. (2008, July). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In *International conference on informatics in secondary schools-evolution and perspectives* (pp. 19-30). Springer, Berlin, Heidelberg.

Dagiene, V., & Stupuriene, G. (2016). Bebras: A sustainable community building model for the concept based learning of informatic and computational thinking. *Informatics in Education*, *15*(1), 25.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications Of The Acm*, *52*(6), 28-30.

Google Education (2020, November 29). *Google Education* https://edu.google.com/intl/ALL_tr/products/workspace-for-education/

Feurzeig, W., & Papert, S. A. (2011). Programming- languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, *19*(1), 217-228.

Flavell, J. H. (1979). Metacognition and comprehention monitoring: A new era of cognitive development inquiry. *American Psychologist*, *34*, 906-911.

Futschek, G. (2006, November). Algorithmic thinking: in secondary scholls- Evolution and Perspectives. *Springer Berling Heidelberg*, 159-168.

Gardner, H. (1985). *Frames of minds: Thinking skills: Critical tkinking and problem solving*. Cambridge University Press.

Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Henderson, P. (2008). Computer science unplugged. *Journal of Computing Sciences in Colleges*, *23*(3), 168-168.

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 & Education*, *82*, 263-279.

ISTE. (2011). *Operational Definition for Computational Thinking*. https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf.

ISTE. (2016). *ISTE standards for students*. https://www.iste.org/standards/standards/for-students-2016

Jonassen, D. H. (2000). *Computers as midtools for schools: Engaging critical thinking*. Prentice Hall.

Jonsson, B., Norqvist, M., Liljekvist, Y., & Lintner, J. (2014). Learning mathematics through algorithmic and creative reasoning. *The Journal Mathematical Behavior, 36*, 20-32.

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a sistematic research review. *Baltic Journal of Modern Computing, 4*(3), 583.

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *The American Mathematical Monthly*, *92*(3), 170-181.

Lau, J. Y. (2011). *An introduction to critical and creativity: Think more, think better*. New Jersey: John Wiley & Sons.

LEGO Education. (2018). *WeDo 2.0 Computational thinking teachers guide*. LEGO Education.

Liskov, B., & Guttag, J. (2000). *Program Development in JAVA: Abstraction, Specification and Onject Orient Design.*

Mayer, R. E. (1992). *Thinking, problem solving, cognition* (2. Ed.). Freeman.

Mayer, R. E. (1998). Cognitive, metacognitive and motivational aspects of problem solving. *Instructional Science, 26*, 49-63.

Ministry of National Education. (2020, November 21). *Öğretmen Kitaplığı*. https://ogretmen.meb.gov.tr/kitap/bilgiislemsel1/

Moore, B. N., & Parker, R. (2008). *Critical thinking* (9. Edition). McGraw-Hill.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. The National Academies Press.

Newell, A., & Simon , H. A. (1972). *Human problem solving*. Englewood Cliffs: NJ: Printice-Hall.

Oktaç, A., & Çetin , İ. (2016). *APOS teorisi ve matematiksel kavramların öğrenimi*. Pegem Akademi

Otaran, A. (2017). *Design, control and evaluation of educational devices with series elastic actuation*. (Unpublished master thesis). Sabancı University.

Papert, S. (1993). *Mindstorms: Children, computers and powerful ideas*. Basic Book.

Papert, S. (1996). An exploration in the space of mathematics educations. *Int. J. Comput. Math. Learn.*, *1*(1), 95-123.

Piaget, J. (1964). Cognitive development in children: Development and learning. *Journal of Research in Science Teaching, 2*(3), 176-186.

Polya, G. (1957). *How to Solve it: A New Aspects of Mathematical Methods*. Prentice University Press.

Quesada, J., Kintsch, W., & Gomez, E. (2005). Complex problem-solving: A field in search of a definition. *Theoretical Issues in Ergonomics Science, 6*(1), 5-33.

Resnick, M., Ocko, S., & Papert, S. (1988). LEGO, Logo, and design. *Children's Environments Quarterly*, 14-18.

Robokids. (2020, December 01). *LOGO education nedir*. https://www.robokids.com.tr/lego-education-nedir-

Schneider, G. M., & Gersting, J. (2016). *Invitation to computer science*. Nelson Education.

Schoenfeld, A. (1987). What's all the fuss about metacognition. In Scohoenfeld, A. (Ed.), *Cognitive Science and Mathematics Education*. Lawrence Erlbaum.

Scratch. (2020, December 02). *Programlamaya giriş*. https://scratch.mit.edu/studios/1611338/

Sedgewick, R., & Wayne, K. (2011). *Algorithms*. Addison-Wesley.

Smith, M. (1991). Toward a unified theory of problem solving. In M. Smith, M.(Ed.), *A view from biology*. Hillsdale.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*, 257-285.

Syslo, M. M., & Kwiatkowska, A. B. (2015, September). Introducing a new computer science curriculum for all school levels in Poland. In: *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, 141-154.

Şahinel, S. (2002). *Eleştirel düşünme*. Pagem Akademi Yayıncılık.

Taylor, I. A. (1959). The nature of the creative process. In P. Smith (Ed.), *Creativity: An examination of the creative process* (pp. 51-82). Hastings House Publishers

Turkish Language Society. (2020, November 27). *"örüntü" kelimesinin araştırması*. https://sozluk.gov.tr/

Üzümcü, Ö. (2019). *Bilgi işlemsel düşünme becerisine yönelik program tasarımının geliştirilmesi ve etkinliğinin değerlendirilmesi*. (Unpublished doctoral dissertation). Gaziantep University.

Wikipedia. (2020, November 27). *Algoritma.* https://tr.wikipedia.org/wiki/Algoritma

Wikipedia. (2020, November 27). *Eleştirel Düşünme*. https://tr.wikipedia.org/wiki/Ele%C5%9Ftirel_d%C3%BC%C5%9F%C3%BCnme

Wikipedia. (2020, November 28). *Problem Çözme*. https://tr.wikipedia.org/wiki/Bili%C5%9Fsel_bilim#Problem_%C3%87%C3%B6zme

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

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Wing, J. M. (2011). Research notebook: Computational thinking- what and why? . *The Link Magazine*, *6*, 20-23.

Yecan, E., Özçınar, H., & Tanyeri, T. (2017). Bilişim teknolojileri öğretmenlerinin görsel programlama öğretimi deneyimleri. *İlköğretim Online*, *16*(1), 377-393.

Disclosure Statements

1. Contribution rate statement of researchers: First author 50%, Second author 50%.

2. No potential conflict of interest was reported by the authors.