EFFECTIVENESS OF KIBO EDUCATIONAL ROBOTICS-BASED ACTIVITIES ON PRESCHOOLERS' COMPUTATIONAL THINKING SKILLS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF SOCIAL SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HASRET KÖKLÜ YAYLACI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF ELEMENTARY AND EARLY CHILDHOOD EDUCATION

FEBRUARY 2024

Approval of the thesis:

EFFECTIVENESS OF KIBO EDUCATIONAL ROBOTICS-BASED ACTIVITIES ON PRESCHOOLERS' COMPUTATIONAL THINKING SKILLS

submitted by **HASRET KÖKLÜ YAYLACI** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Elementary and Early Childhood Education, Early Childhood Education, the Graduate School of Social Sciences of Middle East Technical University** by,

Prof. Dr. Sadettin KİRAZCI
Dean
Graduate School of Social Sciences _____

Prof. Dr. Feyza ERDEN
Head of Department
Department of Elementary and Early Childhood Education _____

Prof. Dr. Refika OLGAN
Supervisor
Department of Elementary and Early Childhood Education _____

Examining Committee Members:

Prof. Dr. Erdinç ÇAKIROĞLU (Head of the Examining Committee)
TED University
Department of Mathematics and Science Education _____

Prof. Dr. Refika OLGAN (Supervisor)
Middle East Technical University
Department of Elementary and Early Childhood Education _____

Prof. Dr. Feyza ERDEN
Middle East Technical University
Department of Elementary and Early Childhood Education _____

Assoc. Prof. Dr. Göknur KAPLAN
Middle East Technical University
Department of Computer Education and Instructional Technology _____

Assoc. Prof. Dr. Dilek ALTUN
Ankara Yıldırım Beyazıt University
Department of Child Development _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Hasret Köklü Yaylacı

Signature :

ABSTRACT

EFFECTIVENESS OF KIBO EDUCATIONAL ROBOTICS-BASED ACTIVITIES ON PRESCHOOLERS' COMPUTATIONAL THINKING SKILLS

KÖKLÜ YAYLACI, Hasret

Ph.D., Department of Elementary and Early Childhood Education, Early Childhood Education

Supervisor: Prof. Dr. Refika OLGAN

February 2024, 243 pages

This study mainly investigated the effect of the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum on preschool children's computational thinking (CT) skills. This study also examined the effects of gender and age in month on children's CT skills. For the first purpose, data on CT skills were collected through two assessment tools, TechCheck-K and TACTIC-KIBO. TechCheck-K is an unplugged assessment tool that does not require coding/programming experience and was used for pretest and posttest administrations to compare CT skills in the control and experimental groups. The TACTIC-KIBO is a plugged-in assessment tool that requires preschool children's coding/programming experience with KIBO robotics and was used to examine the experimental group children's programming-based CT skills after implementation by categorizing them as proto-programmers, early programmers, and programmers. The results showed that the CAL-KIBO kindergarten curriculum significantly contributed to the development of CT skills in the experimental group and mostly developed powerful ideas were hardware/software, debugging, and control structures (with a ceiling effect). The TACTIC-KIBO results showed that the majority of children in the experimental

group reached the programmer level. Moreover, while the challenging powerful idea was design process, the challenging task for preschool children was programming by repetition loops. Finally, gender and age in month were not significant predictors of CT skills. However, the score gap between two age-in-month groups decreased in the experimental group after implementation, while the score gap in age-in-month groups remained in the control group. This demonstrated the developmental appropriateness of the CAL-KIBO kindergarten curriculum when used with preschool children.

Keywords: Computational Thinking Skills in Early Childhood, Educational Robotics Based Coding and Programming, TechCheck-K, TACTIC-KIBO, CAL-KIBO

ÖZ

KIBO EĞİTSEL ROBOTİK TEMELLİ ETKİNLİKLERİN OKUL ÖNCESİ ÇOCUKLARIN BİLGİ İŞLEMSEL DÜŞÜNME BECERİLERİ ÜZERİNDEKİ ETKİSİ

KÖKLÜ YAYLACI, Hasret
Doktora, Temel Eğitim, Okul Öncesi Eğitimi Bölümü
Tez Yöneticisi: Prof. Dr. Refika OLGAN

Şubat 2024, 243 sayfa

Bu çalışma temel olarak Okuryazarlık Olarak Kodlama-KIBO (CAL-KIBO) anaokulu müfredatının okul öncesi çocukların bilgi işlemsel düşünme (BİD) beceri düzeyleri üzerindeki etkisini araştırmıştır. Bu çalışmada ayrıca çocukların cinsiyet ve ay cinsinden yaşların BİD becerileri üzerindeki etkileri de incelenmiştir. İlk amaç doğrultusunda, BİD becerilerine ilişkin veriler TechCheck-K ve TACTIC-KIBO olmak üzere iki ölçme aracı aracılığıyla elde edilmiştir. TechCheck-K kodlama ve programlama deneyimi gerektirmeyen bir ölçme aracıdır ve kontrol ve deney gruplarında BİD becerileri düzeylerinin gelişimini karşılaştırmak amacıyla ön-test ve son-test uygulamalarında kullanılmıştır. TACTIC-KIBO, okul öncesi çocuklarının KIBO eğitsel robotlarıyla kodlama ve programlama deneyimini gerektiren bir değerlendirme aracıdır ve deney grubu çocuklarını uygulama sonrasında proto-programcılar, erken programcılar ve programcılar olarak kategorize ederek, programlamaya dayalı BİD becerilerini incelemek için kullanılmıştır. Bulgular, CAL-KIBO anaokulu müfredatının deney grubundaki çocukların BİD becerilerinin gelişimine anlamlı bir fark yarattığını ve en çok geliştirilen güçlü fikirlerin donanım/yazılım, hata ayıklama ve kontrol yapıları (tavan etkisiyle) olduğunu ortaya

koymuştur. TACTIC-KIBO bulguları ise deney grubundaki çocukların çoğunluğunun programcı seviyesine ulaştığını göstermektedir. Ayrıca, TACTIC-KIBO uygulaması sırasında çocukların cevap verirken en çok zorlandığı güçlü fikir tasarım süreci iken, zorlayıcı görev ise tekrar döngüleriyle programlama olmuştur. Son olarak, cinsiyet ve ay cinsinden yaş değişkenleri okul öncesi çocuklarının BİD becerileri düzeylerinin anlamlı yordayıcıları olmadığını ortaya koymaktadır. Öte yandan, deney grubunda ay cinsinden yaş grupları arasındaki puan farkı uygulama sonrasında azalırken, kontrol grubunda değişiklik göstermemektedir. Bu durum, CAL-KIBO anaokulu müfredatının okul öncesi çocuklarıyla kullanıldığında gelişimsel açıdan uygun olduğunu göstermiştir.

Anahtar Kelimeler: Erken Çocuklukta Bilgi İşlemsel Düşünme Becerileri, Eğitsel Robotik Tabanlı Kodlama ve Programlama, TechCheck-K, TACTIC-KIBO, CAL-KIBO

To my son Burak Kaan and beautiful family

ACKNOWLEDGMENTS

First of all, I would like to thank my dear supervisor Prof. Dr. Refika OLGAN. Throughout my undergraduate and graduate education, you were a role model and light to me. I have learned a lot from her about how to improve myself as an academic and researcher. Thank you very much for all your support and feedback, for all you have contributed to me. I would also like to express my sincere appreciation to dear Prof. Dr. Erdinç ÇAKIROĞLU, for his constructive feedback, contributions, and kindness throughout my thesis. It was a big chance for me to benefit from your support. I would also like to thank dear Prof. Dr. Feyza Erden who helped me improve myself with the courses I took during my undergraduate and graduate education and provided valuable contributions and feedback throughout my thesis process. I want to extend my special thanks to Assoc. Prof. Dr. Göknur KAPLAN and Assoc. Prof. Dr. Dilek ALTUN for their constructive feedback and valuable suggestions in my defense.

Moreover, I would like to thank Assist. Prof. Dr. Sungur GÜREL, Dr. Halil İbrahim SARI, and Dr. Sinan BEKMEZCİ for their help in my data analysis process, suggestions, and feedback throughout the thesis. Your help and suggestions during the analysis process contributed a lot to the progress of my thesis. I would also like to thank my dear office mate, Res. Ass. Ayça ARI, for the positive energy she gave me and the understanding and kindness she showed when I was working like crazy in the office. I would like to extend my thanks to my lovely friend Asst. Prof. Dr. Seher Avcu for all her kindness, encouragement, and support throughout my thesis process.

I am wholeheartedly thankful to my beloved family for their love, patience, encouragement and understanding throughout the process, to my dear mum, my dear dad, my dear brother, and my lovely husband. They are always there for me when I need their support. Last but not least, my dear lovely son *Burak Kaan*, it's time to spend more time with you, your smiling face has always made me stand up and feel stronger. I'm glad to have you in my life, you are very precious to me. I am so lucky

one to have such a beautiful family. I am also grateful to myself for completing this hard and significant milestone in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANCOVA	Analysis of Covariance
CT	Computational Thinking
CAL-KIBO	Coding as Literacy-KIBO
PTD	Positive Technological Development
PYD	Positive Youth Development
SPSS	Statistical Package for the Social Sciences

CHAPTER

INTRODUCTION

This chapter provides the background of the study, explains its purpose, significance, and defines some of the important terms it investigated.

1.1. Background of the Study

The education of the current generation heavily relies on the use of technology and digital tools, and it has been proposed that 21st century skills should be taken into account when setting educational goals and policies (Foot, 2014). Some of these skills are reported to be collaboration, problem-solving, critical thinking, creativity (Kylonen, 2012), and communication skills (Partnership for 21st Century Learning, 2015). Specifically, a detailed taxonomy proposed by the Assessing and Teaching of 21st Century Skills Organization (ATC21S) defines the pillars of 21st century skills as *ways of thinking* (creativity and innovation; critical thinking, problem-solving, decision-making; learning to learn and metacognition), *ways of working* (communication, collaboration, and teamwork), *tools for working* (information literacy, information technology, and communication skills) and *living in the world* (life and work, personal and social responsibility) (Binkley et al., 2010). The taxonomy emphasizes that although information and communication technologies are the primary means by which 21st century skills are acquired, teamwork is the foundation for working methods and thinking skills—which are primarily categorized under the general heading of Computational Thinking—that are universally acquired, such as reading, writing, and math (Wing, 2006).

It is mostly believed that computational thinking is one of the most important 21st century skills (Cuny et al., 2010; Davies et al., 2011) due to the fact that it encompasses critical thinking, analytical thinking, algorithmic thinking, and creative

thinking, which are components of 21st century skills (Barr & Stephenson, 2011; Grover & Pea, 2013; Voogt et al., 2015). Moreover, computational thinking skills and 21st century skills are closely linked, and these are necessary to learn in the current digital era as one of the core approaches to thinking and living in the modern world (Nouri et al., 2020; Tabesh, 2017). In fact, computational thinking is a set of abilities and refers to the capacity to deal with issues and solve problems in the real world (Buitrago-Florez et al., 2017) by "designing systems and understanding human behavior by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Individuals with the ability to think computationally are considered to have an advantage over other individuals in society. This is due to the belief that people with computational abilities can come up with original ideas and workable solutions for challenging circumstances (Wing, 2014). Furthermore, computational thinking skills are essential for programmers and individuals interested in pursuing a career in information and communication technologies (Saxena et al., 2020). The study of computational thinking skills has drawn a lot of attention recently for a variety of reasons, including the desire to comprehend the mechanisms, skills, and learning processes that go into their construction (Bers et al., 2022).

The term computational thinking was first introduced by Papert (1980, 1996) in his book "Mindstorms: Children, Computers and Powerful Ideas". Papert claimed that the focus of computational thinking is not to think like machines or computers but to process the mind in a computer-like way to achieve predetermined goals and debug. Additionally, Papert addressed the relationship between programming and computational thinking abilities, holding that procedural thinking could be developed through programming in all academic fields (Nouri et al., 2020). As Papert worked with Piaget for a long time, he was influenced by his educational philosophies and understanding of learning and teaching especially in young children (Relkin, 2018). According to Papert's constructionist learning theory, children learn more effectively when adults give them concrete activities that let them freely explore, find answers, and create solutions for real-world problems (1980). Furthermore, he emphasized the importance of "constructing a meaningful product" (Tabesh, 2017). Specifically, like learning by doing, constructing a meaningful product enables self-expression and enhances learning. Therefore, he focused on designing concrete, tangible materials

that allow children to express themselves and construct knowledge in their way (Strawhacker & Bers, 2018).

Following Papert, computational thinking skills have been reinterpreted by Wing as "the thought processes involved in formulating a problem and expressing its solution(s) in a way that a computer - human or machine - can effectively execute" (2014, p. 5). It encompasses mental skills like system design, human behavior identification and comprehension, and problem solving (Wing, 2006). Furthermore, computational thinking is conceptualizing at several levels of abstraction and is a fundamental skill (like reading and writing) that can be reasonably learned. It does not, however, primarily focus on computer programming. According to Wing (2014), people think and solve their problems through computational thinking strategies, and most importantly, it is not just about making people think like a computer, but equipping people with computing devices that help them to engage with the tasks and apply powerful ideas of computational thinking (Wing, 2006). Additionally, computational thinking encompasses both mathematical and engineering thinking. More specifically, it is based on mathematics and engineering through which people freely construct and design systems that relate to real-world problems and discover things beyond the world (Wing, 2006). Concepts, an intellectual framework for thinking, and not just artifacts but also hardware and software are all part of computational thinking (Wing, 2014). Finally, computational thinking is accessible to all, can be taught anywhere, and is a simple way to incorporate into teaching methods. It also helps people get used to using computational thinking techniques, which can grow with practice (Wing, 2006).

Several theories or frameworks describe the components, concepts, or pillars of computational thinking skills that could be embedded in classroom practices (Bers, 2018; Brennan & Resnick, 2012; Selby & Woollard, 2013). One of the most widely accepted approaches to computational thinking was provided by Brennan and Resnick (2012), who described seven concepts that are highly useful for transferring both programming and non-programming contexts. Accordingly, these concepts are *sequences, loops, parallelism, events, conditionals, operators, and data*. The sequences refer to "a series of discrete steps or instructions that can be executed by

the computer" (Brennan & Resnick, 2012, p. 3), and the series of codes are instructions in a program that determines the desired action or a specific activity. Loops refer to the repetition of actions in a program. The events are defined as "one thing causing another to happen" such as pressing a button to move a codable robot (Brennan & Resnick, 2012, p. 4). Parallelism is described as "sequences of instructions that occur simultaneously" (Brennan & Resnick, 2012, p. 4), such as programming with music playing continuously in the background. Conditionals involve making decisions about the instructions that will occur under certain conditions. To illustrate, when programming with Scratch, Scratch Junior (ScratchJr), and KIBO, if block is used to determine the conditions of a program. Operators allow mathematical expressions such as addition, subtraction, division, sine and exponents, and string operations through operator blocks. Finally, data involves "storing, retrieving, and updating values" contained as variables and lists (Brennan & Resnick, 2012, p. 6). To exemplify, programming with Scratch, a child can use a series of variable blocks to add up a score in an interactive game he has created.

Selby and Woollard (2013) offered another computational thinking framework that outlines the necessary skills for computational thinking. The researchers reported that the key computational terms that have been most defined in the previous literature are *thought process, decomposition, and abstraction.* Accordingly, computational thinking skills may be considered as a thought process rather than a thought outcome and can also involve abstraction - focusing on the key dimension of a problem rather than getting bogged down in detail - and decomposition - breaking down a large task into manageable parts. They, then, proposed a new framework for computational thinking skills that included *algorithmic thinking* -planning solutions step by step-, *evaluation* -comparing solutions to find the best-, and *generalization* - applying similar solutions to different problems that have the same patterns-, which are the most used concepts in problem-solving contexts (Tsai et al., 2023). Other frameworks include abstraction, modularity, generalization, decomposition, and algorithms (Atmatzidou & Demetriadis, 2016), data collection, data analysis, data representation, decomposition, abstraction, algorithms, automation, parallelism, and simulation (Barr & Stephenson, 2011); and data practices, modeling and simulation,

computational problem solving, systems thinking (Weintrop et al., 2016). These models have been developed to explain how students' computational thinking skills develop while integrating computational practices in secondary, K–9, K–12, and university students.

Although the integration of computational thinking into educational practices has received more attention in primary, secondary, and tertiary education, it has been identified over the past decade as an important skill to be fostered through developmentally appropriate practices from the early years (Batı, 2022; Bers et al., 2022; Saxena et al., 2020; Su & Yang, 2023). To achieve this, the early childhood period is critical for implementing educational practices that cultivate computational thinking skills (Flannery & Bers, 2013). In fact, Buitrago-Florez et al. (2017) proposed that preschoolers should be encouraged to acquire computational thinking skills from an early age in order to stimulate cognitive development, despite the fact that these skills are challenging to learn. This is so that they can progress to logical thinking through logical reasoning, as preschoolers are in the preoperational stage and the second substage, the intuitive thinking sub-stage (Piaget, 1951). To illustrate, young children can explain a situation with rational rather than magical beliefs. They can recognize cause-and-effect relationships, make predictions, and follow a step-by-step approach to achieve the goal. This is also due to a semiotic function that enables children to understand the reasons and results of their actions and why some actions are successful while others are not (Piaget, 1954; Saxena et al., 2020). To illustrate, during the design process of computational thinking, children experience an iterative learning process that allows them to follow their problem-solving strategies step by step - and construct their knowledge through trial and error and debugging techniques that find and fix errors, and such engagements can build their computational thinking skills (Bers, 2018). Children's language abilities also develop during this time, and symbolic thinking becomes more prevalent while logical thinking is less prevalent (Sigelman & Rider, 2012). However, with the aid of practical and active computational learning experiences (Su & Yang, 2023), typically developing children could move from concrete operations to abstract operations (Piaget, 1952, 1954). For instance, they can apply problem-solving skills by employing decomposition that is they can break down complex problems into

smaller steps. Additionally, the fundamental abilities of computational thinking, such as sequencing and algorithmic thinking, enable preschoolers to solve problems (Angeli, 2022). Throughout this process, developmentally appropriate practices will support the scaffolding of preschoolers' computational thinking processes and their transition to abstract thinking processes. (Batı, 2022). Most importantly, according to Piaget's (1954) theory of cognitive development, pattern recognition, and algorithm design skills can be easily developed in preschool children, which are the core concepts of computational thinking and lead to the development of their computational thinking skills (Saxena et al., 2020). Therefore, we may conclude that preschool children are at an appropriate age to independently develop learning and thinking processes that contribute to computational thinking skills as they experience them.

In terms of the advantages of developing computational thinking abilities early on, it can be claimed that young children can start cognitive development through problem-solving activities and real-life scenarios (Buitrago-Florez et al., 2017). Through developmentally appropriate practices and learning tools, children can acquire many other skills while employing computational thinking skills. To exemplify, while engaging with digital and robotic devices, preschool children not only improve their computational skills but also have playful experiences and develop their fine and gross motor skills (Bers et al., 2014). Moreover, both cognitive skills (Su & Yang, 2023) including positive executive functioning skills (Arfe et al., 2019; Yang et al., 2022), metacognitive skills (Highfield, 2010), pattern recognition, sequencing, and algorithm design skills (Saxena et al., 2020), modularity, algorithms, representation (Relkin et al., 2021), and recognition, sorting, and algorithm design (Saxena et al., 2020); and non-cognitive skills involving collaborative work and communication skills in digital platforms can be developed while engaging in computational activities (Bers et al., 2019; Critten et al., 2021). For example, a study conducted by Critten et al. (2021) found that children under 36 months of age improve their communication and collaboration skills (e.g., group cohesion and teamwork) through programming and coding-based guided play activities. Preschoolers would indeed learn how to collaborate with others, listen to others' viewpoints, and share ideas during the computational learning process (Bers,

2018). In conclusion, computational thinking in the early years provides a strong foundation for preschool children's intellectual development, problem-solving skills, creativity, and reasoning. It also equips children with social-emotional skills that will serve them well throughout their lives and enable them to navigate and thrive in a technology-driven world.

In another study, Weintrop et al. (2015) stated that the study of computational thinking skills can be broken down into well-defined, teachable, and measurable skills. Furthermore, a developmentally appropriate framework for explaining the constructs of computational thinking skills would be studied (Bers, 2018). However, the frameworks or approaches to concepts and core skills of computational thinking are mostly appropriate for older children (Atmatzidou & Demetriadis, 2016; Brennan & Resnick, 2012; Selby & Woollard, 2013; Weintrop et al., 2016). As a result, Bers (2018) created a theory of computational thinking skills for young children that is developmentally appropriate and reported that children can acquire seven powerful ideas. Indeed, the *powerful ideas* term was firstly interpreted and explained by Papert (1980), and Bers (2018) defined the seven powerful ideas of computational thinking as hardware/software, representation, control structures, algorithms, modularity, debugging, and design process. In response to the call by Weintrop and his colleagues (2015), the researcher noted that these powerful ideas are developmentally appropriate pillars of computational thinking skills for young children and include teachable and measurable concepts (Bers, 2018). Thus, it was decided in this study to investigate and evaluate preschoolers' computational thinking abilities while focusing on seven powerful ideas.

Preschoolers' computational thinking abilities would be developed through the use of appropriate teaching and learning practices that incorporate developmentally appropriate frameworks (Batı, 2022). The developmentally appropriate computational practices include unplugged and plugged-in activities including game creation, simulations, Scratch programming, and educational robotics (Martins et al., 2023). For instance, unplugged practices are carried out without the use of computers, tablets, smartphones, etc., and are implemented by using paper and pencil activities, game cards, mazes, puzzles, crayons, and lots of materials around. They

are affordable and offer tangible examples of abstract computational thinking ideas (Batı, 2022). There are also plugged-in activities and learning tools that improve young children's computational thinking skills. For example, while programming ScratchJr, children interfere with a visual coding platform to create codes through a series of blocks and program the cat on the screen as they wish, and this process contributes to their self-expression and problem-solving skills. (Bers, 2018; Pugnali et al., 2017). In addition to screen-based coding and programming platforms, educational robotics also allows children to engage with tangible devices while focusing on their programming skills and cultivating the core concepts of computational thinking skills (Martins et al, 2023). Overall, children's computational thinking skills are developed through both plugged-in and unplugged activities and learning tools (Batı, 2022), but combining the two is thought to double children's computational learning (Saxena et al, 2020).

Research on developmentally appropriate computational practices emphasizes how coding and programming activities help children develop their computational thinking abilities because programming is an instrumental skill that helps with computational thinking (Arfe et al., 2019; Bers, 2008; Bers, et al., 2014). While programming through coding, several thinking procedures and skills are employed including sequencing, algorithms, debugging, and design process, problem representation, problem-solving, exploration, and iteration of multiple solutions which are skills that build computational skills (Bers, 2018; Papert, 1980). Therefore, it is recommended that coding be used for programming tasks to foster children's computational thinking abilities, and that young children be given the opportunity to learn with both digital and tangible manipulatives (Bers, 2018; Futschek & Moschitz, 2011; Strawhacker & Bers, 2018).

In this context, educational robotics has been identified as the most appropriate learning tool to promote computational experiences in early learning settings due to its tangible, concrete, fun, hands-on nature, and ease of integration into a variety of learning activities (Pugnali et al., 2017). Most importantly, children can apply powerful ideas of computational thinking while coding and programming educational robotics (Bers, 2018), learning abstract computational concepts while

improving their computational thinking skills (Kazakoff et al., 2013). By coding and programing through educational robotics, children are also active problem solvers, decision makers of their tasks or projects, collaborators of their peers, parents or teachers, and learners of many concepts embedded in computational activities (Bers, et al., 2014). In addition, codable robots also develop children's problem-solving skills, metacognitive and mathematical skills, sequencing skills, and algorithmic thinking skills (Futschek & Moschitz, 2011; Highfield, 2010; Papert, 1980). Codable robotics stimulates many children's higher-order thinking abilities through active learning by embodying abstract thought. Because of these benefits, educational robotics was chosen in the current study to foster preschoolers' computational thinking abilities through coding and programming exercises, which was found to be a critical component in the development of preschoolers' computational thinking abilities. Finally, there are a variety of robotic platforms that facilitate the implementation of programming tasks through coding (Highfield, 2010). Specifically, KIBO, Bee-bot, Roomer Too, Pro-bot, and KIWI robotics are the most preferred robotic platforms by researchers (Gonzales & Munoz-Repiso, 2017; Sullivan & Bers 2016a; Sullivan et al.; 2013). Because KIBO robotics is suitable for both research and kindergarten-aged children, it was chosen for use in the current study.

1.2. Background Factors of Computational Thinking Skills

Certain behaviors that are typical of each cognitive developmental stage are expected of young children (Piaget, 1954). However, in addition to the dominant behaviors or skills that are characteristic of each cognitive stage, Piaget noted the variability in children's performance on structurally similar tasks (Chapman, 1988) and explained this variability by his foundational understanding that "thought originates in action". In other words, the content of each piece of knowledge for a particular activity is developed independently by the child's actions (Müller et al., 2015, p. 144). In light of the computational thinking, it is possible to say that preschoolers' development of computational thinking and behavior may differ, that is could be heterogenous, despite the fact that they perform comparable tasks in the classroom. This could be due to the fact that children's computational thinking processes and behaviors also

grow independently depending on their different daily engagements. These daily engagements may include computational experiences after school and in the home environment, which could shape different backgrounds in children (Tsai et al., 2021). For example, children who engage in coding or programming with mobile devices at home may be better able to develop their computational thinking skills. Self-directed coding and programming at home through screen-based technologies could also have a positive impact on computational thinking skills (Rum & Ismail, 2017; Tsai et al., 2021). However, too much screen time is detrimental for children, and passive screen activities like watching TV, movies, or games on tablets and smartphones do not encourage them to use technology creatively or actively and can delay their cognitive, social, and emotional development (American Academy of Pediatrics [AAP], 2016). Hence, the duration and the quality of screen time are important issues to be controlled by educators and parents to prevent developmental delays in the early years. In fact, exploring the background issues that occur outside of school environment is important to explain the external factors that have an impact on the development of computational thinking skills in the early years and to find the effective teaching and learning strategies that can be implemented outside of school to double computational learning.

In addition, there are other background issues, such as age and gender, which independently influence children's success in computational tasks. Related literature reveals that gender does not have an impact on young children's computational thinking skills (Batı, 2022; Papadakis et al., 2016; Sullivan & Bers, 2013; Yang et al., 2023). Rather than age, motivation, gender stereotypes, and the gender of the teacher are reported to make differences between boys and girls and affect their success on computational tasks (Doube & Lang, 2012; Master et al., 2021; Master et al., 2023; Sullivan & Bers, 2018; Sullivan & Bers, 2019). For instance, male teachers are reported to have a positive impact on boys' computational thinking skills compared to girls (Sullivan & Bers, 2018; Sullivan & Bers, 2019). Boys, however, are said to think they are better at and more interested in coding-related tasks than girls as they get older and enter the first grade (Master et al., 2021). Nevertheless, with the help of developmentally appropriate computational practices, boys and girls could be equally successful in coding and programming tasks (Bati, 2022). Similar to

gender, age has also been reported to have no effect on children's success in computational tasks (Papadakis et al., 2016; Relkin et al., 2021), but as children get older, they are reported to complete more complex tasks, including five and more instructions (Sullivan & Bers, 2016a). This could be due to sufficient memory or working memory, which is the ability to remember many instructions and complete a task (Batı, 2022). Therefore, exploring computational practices considering young children's memory capacities is essential so that the children can easily engage, remember, and implement the desired instructions in the activities. In this regard, investigating the underlying causes of computational thinking abilities in preschoolers is an important topic to investigate.

1.3. Significance of the Study

Computational thinking skills encompass many skills including sequencing, algorithmic thinking, analytical thinking, abstract and creative thinking that employ many problem-solving strategies and design process procedures (Bers, 2018; Selby & Woollard, 2013; Wing, 2008). The definition of computational thinking skills also imply that these skills are multi-dimensional, cannot be investigated as a single skill as there are many underlying traits of computational thinking (Relkin et al., 2020). This feature of computational thinking would make it hard to explore the developmental processes, approaches and techniques that construct it. Nevertheless, there has been an increasing interest in exploring the developmentally appropriate computational practices, core developmental steps and components and contributors of computational thinking for the elementary school students (Noh & Lee, 2020; Tsai et al., 2021; Wei et al., 2021), and high school students (Atmatzidou & Demetriadis, 2016; Guggemos, 2021; Saritepeci, 2020; Zakaria & Iksan, 2020). Nonetheless, little is known about how various abilities and fundamental procedures influence preschoolers' computational thinking abilities (Su & Yang, 2023; Wing, 2008). In order to address concerns like "how and when should people learn this kind of thinking, and how and when should we teach it?" and to comprehend the solid foundation for developing these abilities, research is required (Wing, 2008, p. 3720) in early childhood environments. Investigating these issues could improve the computational thinking abilities of young children and offer early childhood

educators efficient computational practices. To begin with, though, it is important to outline the basic steps involved in computational thinking that young preschoolers would be taught.

Therefore, this study identifies the fundamental processes of computational thinking as the components of computational thinking skills, namely the seven powerful ideas (core contents and skills) of computational thinking (Bers, 2018). These powerful ideas are knowledge of *hardware* and *software*, knowledge of *representational* symbols that lead to specific actions, ability to recognize *control structures* in each task, ability to recognize and set *algorithms* to accomplish a given or desired task, ability to *debug* and apply *modularity* while solving a problem, and ability to follow a *design process* to iterate the entire computational thinking process. The powerful ideas are evidence-based components of computational thinking and developmentally appropriate content and skills that can be explored in preschool children (Bers et al., 2023). Through the integration of powerful ideas into their daily computational activities, children are able to develop new cognitive processes, make effective use of knowledge pertinent to their objectives, and make connections between their prior knowledge and newly constructed knowledge (Alimisis & Kynigos, 2009; Bers, 2018). By engaging in computational activities that incorporate the powerful ideas, the children could develop many other skills, including problem-solving, critical thinking, creativity, and innovation (Flannery et al., 2013; Portelance et al., 2015; Sullivan & Bers, 2012). Bers (2018) stated that although the powerful ideas can be integrated into the early childhood practices; the development of children's understanding of powerful ideas differs depending on their cultural background and future research is needed to explain the developmental process of powerful ideas in diverse contexts. The review of the literature revealed limited research examining the development of computational thinking through seven powerful ideas (Relkin, 2018; Relkin et al., 2021) and no research on the seven powerful ideas of computational thinking in preschool children in the national literature. It was noted that national researchers chose the dimensions they aimed to address in their studies in order to apply theories of computational thinking with preschoolers (Değirmenci, 2022; Bozkurt-Polat, 2023). Therefore, this research has a potential to validate powerful ideas of computational thinking in Turkish preschool

setting and can answer the question of whether powerful ideas can be effectively incorporated in Turkish educational settings. Moreover, by investigating the development of seven powerful ideas in Turkish preschool children, this research may also shed light on how and to what extent the powerful ideas of computational thinking are developed in Turkish preschool children and the results may contribute to both international and national fields for future research.

The plugged-in activities combined with unplugged activities are reported to be used to promote computational thinking in young children, to start with the effective teaching methods and learning tools (del Olmo Munoz et al., 2020; Saxena et al., 2020). Del Olmo Munoz et al. (2020), for example, used both plugged-in and unplugged activities and discovered that while both experimental groups' computational thinking significantly changed, but the group that started with unplugged learning experiences had a greater development in their computational thinking abilities. Another study investigated the contribution of tablets and educational robotics and reported that both learning tools can facilitate computational thinking in preschool children (Yang et al., 2023). The use of plugged-in learning materials is important because the symbols and tangible parts allow for sorting, patterning, observing, and correcting errors in a pattern or algorithm and are reported to contribute to computational thinking skills (Saxena et al., 2020) and the children become excited and motivated when introduced to new technologies (Monteiro et al., 2022) such as educational robotics. Educational robotics differ from computers and tablets because they do not have screen and increase teamwork, communication, collaboration, and socialization in children (Bers et al., 2019; Critten et al., 2021). Since combining plugged-in and unplugged activities is thought to double computational learning, both plugged-in and unplugged activities were implemented in the current research using KIBO educational robotics as plugged-in tools (Saxena et al., 2020). Indeed, no research using KIBO robotics was found; instead, researchers validated their practicality for Turkish preschool settings primarily with the Bee Bot robotics kit or the Code.org platform (Zurnacı & Turan, 2022). As a result, it can be concluded from an analysis of the national studies conducted thus far that this study is the first to investigate the application of KIBO robotics in Turkish preschool environments.

Research on effective learning strategies and techniques for teaching computational thinking is also needed to prepare professional programs that support early childhood teachers on how to integrate educational robotics with developmentally appropriate computational practices (Rusk et al., 2008). However, before determining the effective teaching practices, it is essential to investigate how educational robotics and computational thinking can be effectively integrated into existing curricula across different subject areas (Hsu & Ching, 2019). By shedding light on this issue, teachers would be professionally prepared to teach computational skills through educational robotics, incorporating the most practical with coding and programming activities. In addition, they would develop essential knowledge, skills, positive attitudes, or high self-efficacy beliefs that play a critical role in their instructional practices in their teaching profession (Rusk et al., 2008). It is suggested that educational robotics could be incorporated into preschool settings through three approaches or functions to cultivate computational thinking skills (Munoz-Repiso & Caballero-Gonzalez, 2019). First, educational robotics can be used as a learning tool, and second, it can be applied as a learning object (Goodgame, 2018). In this way, computational thinking processes can be experienced through coding the instructions and programming (Munoz-Repiso & Caballero-Gonzalez, 2019). In the third approach, educational robotics are used in learning environments as a didactic resource (Bruni & Nisdeo, 2017). This means that computational learning can be achieved through inquiry, where children can construct their knowledge through trial and error (Munoz-Repiso & Caballero-Gonzalez, 2019). In the current study, the application of these three strategies and their integration in the classroom were present to investigate computational thinking abilities through the use of educational robotics.

As aforementioned, educational robotics facilitates discovering the computational concepts and thinking skills by programming which was attributed as a significant contributor of computational thinking skills (Yang et al., 2022) and investigated in this study. Learning programming in the early years reported to have a positive impact on math and science achievement (Metin, 2020), socio-emotional interactions and language development (Bers, 2008), creativity (Canbeldek & Işıkoğlu, 2023; Clements, 1991), problem solving (Fessakis et al., 2013) and known as contributor to computational thinking skills (Arfe et al., 2020). There is some research on teaching

and learning programming in a school context, but much of the research is conducted at university level (Heintz et al., 2016; Nouri et al., 2020). Therefore, more investigation is required to determine how teaching programming to young children can help them develop their computational thinking abilities. Having explored the related literature, it was found that Munoz-Repiso and Caballero-Gonzalez (2019) implemented TangibleK program that include programming activities through robotics which significantly improved preschool children's computational thinking skills. Relkin et al. (2021) revealed that programming with robotics significantly contributed some powerful ideas of computational thinking. Moreover, Angeli and Valadines (2020) reported the benefits of educational robotics to develop computational thinking and spatial relationship skills. There were some other studies which focused on programming experience with educational robotics and its contribution to computational thinking skills (Elkin et al., 2014; Relkin et al., 2021). It was observed that there is little national research that investigates the impact of coding and programming activities on computational thinking in early years, despite reports that the number of studies has increased recently (Zurnacı & Turan, 2022). In fact, except for Bozkurt Polat (2023) and Değirmenci (2022), the national studies mostly investigated the unplugged coding or programming skills (Metin, 2020), the effect of unplugged programming on analytical thinking, creativity, and problem solving (Akyol-Altun, 2018; Çakır et al., 2021), creative thinking (Önal & Ardıç, 2020), and mathematical reasoning skills (Canbeldek & Işıkoğlu, 2023; Somuncu & Aslan, 2021). Hence, it is believed that the current study would shed light on the relationship between programming experience with educational robotics and computational thinking skills.

Apart from the aforementioned, Su and Yang (2023) conducted a review study wherein they concluded that measures of computational thinking processes must be valid and reliable. This is because several research was found to lack in reporting the validity and reliability of the computational assessments as required (Su & Yang, 2023). A range of measurement instruments was discovered through a review of the national and international literature in order to create an evaluation tool for preschoolers' computational thinking abilities (Relkin & Bers, 2021; Değirmenci, 2022; Bozkurt Polat, 2023). For instance, while Değirmenci (2022) designed the

Computational Thinking Skills Rubric (BIDR) to which consists of “abstraction, generalization, algorithm, decomposition, modularity, symbolization, debugging and evaluation” dimensions, Bozkurt Polat (2023) addressed the dimensions of “decomposition, algorithmic thinking, abstraction, and generalization”. There are also assessment tools measuring the computational thinking skills of children through interview protocols or project-based coding assessments (Bers et al., 2014; Relkin & Bers, 2019; Wang et al., 2014). These tests are difficult to administer to younger children due to their short attention spans, or they require learning a programming language beforehand. Therefore, it is essential to have a valid and reliable assessment tool that measures preschool children’s computational thinking skills without programming or coding experience and have evidence-based domains of computational thinking.

In this context, the TechCheck-K instrument was found to be appropriate to use with preschool children and translated into Turkish language for the research purposes. It was observed that translation and adaptation studies of the TechCheck-K measurement tool were previously published by other national researchers (Çetin et al., 2022; Yılmaz, 2022). Çetin et al. (2022) collected data from 236 students who were the first-grade children, newly began to elementary school and reported the mean score of children as 10.96. According to discrimination and difficulty indices, Çetin et al. (2022) reported that the TechCheck-K instrument is an easy tool for kindergarten children.

In addition, Yılmaz (2022) conducted a validation and adaptation study of TechCheck-K with 130 kindergarten children. The researcher changed some items in the TechCheck-K, and except for item 4, TechCheck-K instrument was found to be appropriate with 14 items with two factors, namely “critical thinking” and “mathematical thinking”. Different from the previous research (Çetin et al., 2022; Yılmaz, 2022), item response theory was executed in the current study which is considered to provide more information about each item in an instrument than classical test theories especially while developing or adapting instruments by larger samples (Baker & Kim, 2017). Hence, the current research has a potential to extend and enrich the previous results reported about the TechCheck-K through a larger sample of kindergarten children through item response theory.

Because programming skills contribute to computational thinking skills (Arfe et al., 2020) and these skills were investigated in this study in the context of computational thinking skills, it was also necessary to measure preschoolers' programming performance while explaining their computational thinking processes. Given that KIBO educational robotics was used in this study, a context-appropriate tool was also required to gauge children's computational thinking abilities related to programming. To measure children's programming-based computational thinking skills across robotic platforms, there isn't a universal assessment tool. Therefore, the TACTIC-KIBO assessment tool (Relkin & Bers, 2019) was translated into Turkish. The TACTIC-KIBO instrument intertwines the computational thinking skills with programming to measure children's programming-based computational thinking skills. After applying a series of activities via KIBO robotics, this instrument allows researcher to categorize children's programming related computational thinking skills in terms of *proto-programmer, early-programmer, programmer, and fluent programmer levels.*

Although TACTIC-KIBO tool involves four levels of programming related computational thinking skills, only, the first three levels were tested in this study owing to the children's developmental reasons. The fourth level -fluent programmer- involves "if" blocks, which are conditionals investigated through if-then conditional statements while giving instructions to the children. It is reported that conditionals that involve if statements challenge kindergarten children while answering the assessment questions in the fourth level of TACTIC KIBO (Barrouillet & Lucas, 1999; Janveau-Brennan & Markovits, 1999; Relkin, 2018; Rich et al., 2017). These are usually the more difficult sequencing tasks, and it is advised to teach them last once the children have mastered the simpler ones. This is due to the belief that these ideas develop more gradually between the ages of 6 and 12 and are more abstract for young children to understand (Barrouillet & Lecas, 1999). Therefore, the fluent programmer level was not investigated in the current study. In this context, the results of this study may also contribute to both national and international literature about the practicality of TACTIC-KIBO assessment tool while examining the first three levels of programming related computational thinking skills in preschool children.

In addition to those above, the factors that may influence the development and integration of computational thinking is not a new topic and has been explored in previous research through randomized controlled trials and systematic reviews. While some studies mainly focused on the predictive role of age (Atmatzidou & Demetriadis, 2016; Batı, 2022; Bers et al., 2020; Catana-Kuleli, 2018; Korucu et al., 2017; Yildirim & Uluyol, 2022) and gender (Atmatzidou & Demetriadis, 2016; Relkin, et al. 2021; Relkin & Bers, 2021; Sullivan & Bers, 2012; Tsai et al., 2021; Yang et al., 2023); there were also studies that aimed to examine the contributors to computational thinking as background variables (Alsancak-Sırakaya, 2020; Tsai et al., 2021; Yildirim & Uluyol, 2022). Therefore, some background factors were examined in this study in order to identify the factors that may support young children's computational thinking abilities as well as to explain the factors to take into account when planning computational activities. In fact, research indicates that the only ways to support computational processes (Tikva & Tambouris, 2021) and lessen the impact of current demographic and cognitive disparities on children's computational thinking abilities (Bati, 2022). Therefore, by investigating the impact of a coding and programming-based curriculum through educational robotics on the computational thinking skills, potential factors that may play a significant role in acquiring these skills could be explained. In fact, this may also enable how early childhood practitioners to be better supported and professionally trained to design and implement developmentally appropriate activities in their classrooms by considering children's background factors (Rusk et al., 2008).

To begin with the predictive role of age in computational thinking skills of preschool children, in their study Yildirim and Uluyol (2022) investigated whether there is a difference in computational thinking skills of primary school children regarding the grade level and reported that there is a significant difference between first grade children and second, third and fourth grade students in favor. Similarly, Korucu et al. (2017) and Catana-Kuleli (2018) found that grade level made a significant difference in computational thinking skills. Atmatzidou & Demetriadis (2016) noted that children's age makes a difference in particular dimensions of computational thinking, such as modularity and decomposition, algorithms and generalization, and abstraction. In addition, the results of Bers et al. (2022) and Relkin et al. (2021)

indicated a significant difference between first-grade and second-grade children in terms of computational thinking skills. Relkin, et al. also (2021) reported that programming activities that were implemented with first-grade and second-grade children made a significant contribution to the first-grade children but did not contribute to the second-grade children. Considering these studies, to the best of our knowledge, there is limited research on the effect of educational robotics-based programming implementations on the computational thinking skills of preschool children with different ages in months (Atmatzidou & Demetriadis, 2016; Değirmenci, 2022; Relkin & Bers, 2021). Therefore, this was investigated in the present study to contribute to the relevant literature.

Gender differences can also be taken into account when creating efficient teaching and learning strategies for computational thinking, in addition to age. Considering that computational thinking skills can be acquired in early childhood without disregarding equality, gender factor should be particularly sensitive in preschool settings when designing activities, and the activities would address both the interests of both boys and girls (Metz, 2007). Since negative stereotypes make it harder for girls to pursue STEM fields later in life, introducing educational robotics and programming experiences would stimulate girls' interest in these fields (Master et al., 2023). In fact, there is a gender stereotype threat that males appear more confident than females in STEM fields (Sullivan & Bers, 2012, 2019). Examining the studies on whether gender makes a difference in computational thinking skills, the results are mixed. For instance, Tsai et al., (2021) indicated that male students were better at decomposition than the female students. This means that males are better than females at breaking down larger tasks into smaller, more manageable parts to solve problems. Another study revealed that there was no difference between boys and girls after receiving a story-inspired programming experience (Yang et al., 2023). In another study conducted with preschool children, Relkin and Bers (2021) reported a non-significant difference between boys and girls in computational thinking skills. However, according to Relkin et al. (2021), the computational thinking skills of the second-grade children changed significantly after the programming activities with robotics, and gender was a significant factor, while did not make a significant difference in the computational thinking skills of the first-

grade children. These conflicting results could be due to the fact that the children's stereotypes about STEM disciplines change as they get older and such stereotypes would be identified to be addressed through developmentally appropriate practices from the early years (Doube & Lang, 2012; Master et al., 2021; Master et al., 2023; Sullivan & Bers, 2018; Sullivan & Bers, 2019). However, more research is needed to explain the possible effects of gender on the computational thinking skills of Turkish preschool children, as the results may differ in terms of cultural aspects (Bers, 2018). In addition, the results of this study may also indicate whether the implemented curriculum meets the interests of both boys and girls by treating each child equally. This is because, early childhood programs tailored to the interests of boys and girls are thought to have no effect on the computational thinking abilities of the participants (Bati, 2022).

In addition to above mentioned factors, there are other background issues that involve daily experiences, and were found to create differences in computational thinking skills (Alsancak-Sırakaya, 2020; Soleimani et al.; 2016; Tsai et al., 2021; Yang et al. 2022; Yildirim & Uluyol, 2022). In fact, predicting those factors would enable early childhood teachers to design and develop efficient classroom activities but also the parents to contribute to the computational thinking skills of the preschool children after school (Tsai et al., 2021). In the current study, therefore, some background factors were attributed as children's daily experiences occurring at home including *screen time, digital play experiences on mobile phones or tablets,* and *play experiences with tangible materials*. In reviewing the related literature, some research was found to investigate the contribution of background factors to the computational thinking skills. For example, Tsai et al. (2021) found that self-directed learning and after-school learning had a positive impact on the computational thinking skills of junior high school students. Specifically, the students' self-directed programming learning was found to have a positive impact on abstraction, decomposition, algorithmic thinking, evaluation, and generalization. Similarly, Yildirim and Uluyol (2022) explored the predictive role of daily computer use in computational thinking skills and found a significant difference in primary school children. In fact, as the daily computer time increased, the mean scores of the children also increased. According to other studies, students' computational thinking

skills are predicted by their ownership of mobile devices, attitudes toward science, and attitudes toward mathematics (Alsancak-Sırakaya, 2020). Including self-directed learning into programming courses also improves students' programming performance and perceptions of self-directed programming (Peng et al., 2017), which may have contributed to the students' computational thinking skills. Considering younger children Lee et al. (2022) highlighted the importance of puzzles, blocks, sorting and patterning activities, daily activities including patterns of daily routine sequences which are unplugged daily activities. The researchers noted that these experiences are certain types of learning experiences to practice computational thinking skills and form a computational background in children in terms of decomposition, abstraction, pattern recognition and algorithm design. Moreover, in their experimental study, Yang et al. (2022) compared the effects of block play and robot programming in early childhood education and stated that block play can promote multiple concepts and skills for computational thinking. Also, Soleimani et al. (2016) reported that block play can develop some mathematical skills including numeracy, sequencing, categorizing geometric shapes understanding space and physical properties of objects which contribute to computational thinking skills (Soleimani et al., 2016). Nevertheless, no research examining the aforementioned background factors in younger children and their predictive roles on computational thinking skills was discovered when searching the relevant literature. Thus, another goal of this study is to investigate this significant relationship.

Overall, using KIBO educational robotics to implement coding and programming activities, this research examines preschoolers' development of computational thinking skills in light of pertinent literature. Additionally, the preschoolers' computational thinking abilities were investigated in relation to their background variables. Consequently, the current study was carried out to address the following research questions considering the literature:

1. Is the TechCheck-K assessment tool valid and reliable to evaluate the computational thinking skills of preschool children?

2. Do computational thinking skills of preschool children develop with the educational robotics-based coding implementation (via the TechCheck-K)?

2a. What are the general patterns of experimental group and control group preschool children's pretest computational thinking scores?

2b. Is there a significant difference between the experimental group and the control group of preschool children in terms of their computational thinking skills before the educational robotics-based coding implementation?

2c. What are the general patterns of experimental group and control group preschool children's posttest computational thinking scores?

2d. Does educational robotics-based coding implementation produce greater computational thinking scores in the experimental group children than in the control group children after adjusting for pretest differences in computational thinking scores?

3. What are the programming related computational thinking levels of preschool children in the experimental group after the educational robotics-based coding implementation (via the TACTIC-KIBO)?

4. Do background factors (age in month and gender) of preschool children in the experimental and control group make a difference in their posttest computational thinking scores after adjusting for pretest differences in computational thinking scores?

1.4. Definition of Important Terms

Computational thinking skills: "The thought processes involved in formulating a problem and expressing its solution(s) in a way that a computer - human or machine - can effectively execute" (Wing, 2014, p. 5).

Coding: Coding involves writing source code and translating it into a language to make a machine or application work and using specific commands or programming languages program a computer or application (Hunsaker & West, 2019).

Programming: Broader process of designing, writing, testing, and maintaining computer programs by coding process (Smith & Johnson, 2020).

Unplugged computer science activities: The activities that do not involve computers and screen use and develop many cognitive skills by physical materials and games to enable understanding of fundamental processes of computer science (Chen et al., 2023).

Plugged-in computer science activities: The activities that provide programming and coding experience through use of an interface or any technological device enabling to perform various tasks based on programming instructions (Bati, 2022).

KIBO educational robotics: KIBO robotics are screen-free learning tools that were developed by DevTech research group (2018) and involve wooden programming blocks with sensors and modules (i.e., lightbulb module, sound, and distance sensors).

TechCheck-K: An unplugged measurement of computational thinking skills which intertwines computational tasks with seven powerful ideas (Relkin & Bers, 2021).

TACTIC-KIBO: A plugged-in measurement of computational thinking skills which intertwines coding and programming activities with seven powerful ideas and requires coding/programming experience with KIBO educational robotic platforms.

Constructionism: An educational theory developed by Papert and described as engaging with tangible materials or meaningful projects which involve use of technology and social interaction (1991).

The Seven Powerful Ideas of Computational Thinking: Powerful ideas are core contents and skills of computational thinking skills that can be developed by computer science activities involving unplugged or plugged-in learning tools (Bers, 2018).

Hardware: It is a part of the information operating system interprets the software and turn the information from the software into processes (Bers, 2021).

Software: It is a part of the information operating system installed to control and direct the hardware (Bers, 2021).

Control Structures: Recognizing the order in which a set of commands is started and executed (e.g., the cause and effects in programs, conditionals, repetitive actions, loops etc.).

Representation: Understanding that programming languages use symbols to represent specific actions (Bers, 2021).

Debugging: Detecting and fixing the problems/errors in a program, code, hardware or software through systematic analysis and evaluation (Bers, 2021).

Modularity: Breaking a complex task into manageable and smaller tasks by writing the instructions or grouping the instructions to a category (Bers, 2021).

Algorithms: Logically organizing the instructions of a program/task by sequencing/ordering the steps (Bers, 2021).

Design Process: It is an open-ended and iterative process which includes identifying a problem or goal, determining the solution ways, testing, revising, and retesting the solution ways if necessary (Bers, 2021).

CHAPTER 2

LITERATURE REVIEW

Information about the studies that are connected to the current study is provided in this section. The first step is to define computational thinking and how it is taught in early childhood education. The study's theoretical underpinnings are then discussed. The prior research on the integration of computational thinking skills through plugged-in and unplugged activities in preschool and upper grade classrooms is then highlighted.

2.1. The Computational Thinking Skills

Jeanette Wing provided the first definition of the term computational thinking (2008) as "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science" (p. 33). Moreover, computational thinking is defined as a type of analytical thinking and believed to involve mathematical thinking (Wing, 2008). According to this definition, when thinking computationally, problem solving, engineering thinking, design and evaluation, and scientific thinking procedures are employed. Wing (2008) essentially highlights the importance of thinking at multiple levels of abstraction as a fundamental component of computational thinking, which can be applied to and developed within the physical and mathematical sciences. More specifically, according to Wing (2008), abstraction is used for algorithm and programming language. In both cases, people use abstraction to represent their ideas and purposes. In other words, defining the abstraction process is significant when identifying the details, we need and ignoring irrelevant information to actualize our purpose in a practical way. Wing (2006) identifies this process as a basis of computational thinking and adds that abstraction works with decomposition. This is

because, a person should first be able to separate the real concerns of a problem so that s/he can break it into large and complex tasks into smaller and manageable units that are problem solution ways.

In a recent study, Melro et al. (2023) reported the learning opportunities through coding and computational thinking with three dimensions namely *functional, collaborative, critical* and *creative*. The functional dimension refers to the operational or required skills of computational thinking. This dimension is related to the problem solving with computers and benefits from mathematical and engineering thinking while problem solving (Shute et al., 2017). The collaborative dimension is associated with the computational participation and collaborative work of people. It also emphasizes the social interaction among people when coding. For example, when coding, people interact with each other, use, and develop their communication and interpersonal skills (Melro et al., 2023). It is also noted that when engaging with collaborative dimension of computational thinking, peer learning, sharing artifacts that were created during collaborative learning processes are used and in cognitive perspective people approach computational thinking in the lens of algorithmic thinking and problem solving (Kafai, 2016). In addition, to achieve the collaborative dimension, it is necessary to work within a team, negotiate with team members, and create a group solidarity to find collective solutions for a common problem (Barr & Stephenson, 2011).

According to Melro et al. (2023) the last dimension is creative and critical of computational thinking, which are reported to be dynamic and are described as different ways of solving problems and innovative aspects of problem solving. According to Brennan and Resnick (2012), it also includes creating a product through artistic activities and expressing oneself in creative ways. Melro et al. (2023), who reviewed the studies that considered computational thinking in the context of creativity, found that the level of creativity is related to the speed of solving computational problems (Israel-Fishelson et al., 2021) and that creativity is also about finding alternative, innovative solutions by approaching a problem from different perspectives (Korkmaz et al., 2017). When computational thinking is considered in the context of critical thinking, it is stated that questioning is

important, it helps to make informed judgements and helps to make decisions based on evidence-based information (Brennan & Resnick, 2012; Korkmaz et al., 2017). According to one of the studies addressing critical thinking in the context of computational thinking (Caeli & Bundsgaard, 2020), critical thinking suggests that technological tools should be used in a way that respects ethics and personal rights. For instance, racism or sexism are some of the ethical issues that the people take into consideration while engaging with computational thinking through critical pedagogy approach. Lastly, Melro et al. (2023) reported a three-dimensional mapping of computational thinking and noted that three dimensions of computational thinking are–intertwined and dynamic. They also concluded that functional dimension is individual/self-centered and involves mere coding with cognitive processes. The collaborative dimension is, on the other hand, community centered, and coding is employed *by other people*. Lastly, critical, and creative dimensions are society centered and coding is employed *for others.*

2.2. The Computational Thinking Skills in Early Childhood Education

Wing (2008) reported that in order "to ensure a common and solid basis of understanding and applying computational thinking for all, then this learning should best be done in the early years of childhood" (p. 3720). Furthermore, in order to effectively teach and foster computational thinking in young children, it is imperative to provide an explanation of learning methodologies, computational thinking's fundamental concepts, the sequence in which these elements are taught, and how these elements are integrated into day-to-day activities and events (Wing, 2008). Wing (2014) highlights that young children should learn computational thinking skills from the early years like writing, reading and arithmetic (Wing, 2014). For this purpose, numerous studies indicate that helping children learn problem solving, algorithmic thinking, analytical thinking, creative thinking, logical thinking, decomposition, pattern recognition, abstraction, and sequencing skills to contribute their computational thinking skills is beneficial (Brackmann et al., 2016; Brennan & Resnick, 2012; Su & Yang, 2023).

Although computational thinking skills are considered to be difficult to learn in early childhood, many abstract components can be easily acquired from the early years

with appropriate learning and teaching techniques (Martins et al., 2023). For instance, Wing (2006) indicates that we might improve young children's reading, writing, arithmetic, and analytical skills with computational thinking. Thus, it is possible to incorporate computational thinking abilities into regular academic tasks. In early childhood, computational thinking skills can also be earned through concrete experiences including hands-on, playful, kinesthetic, and child-centered learning. For instance, unplugged activities without any screen experience and programming, as well as plugged-in materials consisting of screen and manipulative parts contribute to computational thinking. Both methods develop computational skills, but it has been suggested that activities using both can be more beneficial (Saxena et al., 2020).

In addition, while introducing the ways of developing computational thinking, the term coding is widely used over programming (Tedre & Denning, 2021). This is because, programming is limited to technological tools, while coding can be performed through not only technological tools but also through paper-pencil activities, games, puzzles, and concrete tools that enable abstraction in the early years. For instance, Lee and Junoh (2019) suggest using the concepts and knowledge that young children can connect with their lives and concretize the abstract concepts. In this sense, daily routines can be implemented to improve computational thinking skills such as brushing teeth, making a cake or washing hands activities (Saxena et al., 2020). Although they seem so simple, these tasks develop sequencing, decomposition and modularity, and algorithm-related skills (Martins et al., 2023). This is because, after determining their objective or problem, children first define each step that will help them achieve it through decomposition, then they arrange those steps logically through pattern building and sequencing, and finally they create an algorithm that can be thought of as an unplugged coding process.

2.3. Theoretical Frameworks of the Study

2.3.1. Constructivism

This study is developed around the Constructivism theory of Jean Piaget (1952) which explains the four fundamental stages of cognitive development beginning

from the birth. Piaget's theories of cognitive development were taken into consideration when explaining the computational learning behaviors and development of preschoolers, and these theories accounted for the majority of the explanations for the developmental processes of the preschoolers.

First of all, Piaget noted that during the sensorimotor stage (birth to age 2), preoperational stage (ages 2 to 7), concrete operations stage (ages 7 to 11) and formal operations stage (roughly 11 years and beyond) cognitive and intellectual development occur. It is believed that these stages follow the same order in all children, but the given age ranges are average age ranges since the beginning or ending of stages can differ in children due to their different experiences (Sigelman & Rider, 2012). Put another way, each child develops cognitively differently based on their experiences from day to day; some develop cognitively more slowly than others, and some develop cognitively more quickly than others.

In the *sensorimotor stage*, the learning happens through senses and motor abilities and babies cannot use symbols to represent real objects or events (Sigelman & Rider, 2012). Nevertheless, intelligent actions increase as much as they engage with concrete materials and stimulus that they can see, touch, smell, or taste to discover the real world. In the *preoperational stage*, however, the children have a capacity to use representational knowledge, that is, they think symbolically (Saxena et al., 2020). However, they are not able to create representation of symbols or events since it is an abstract skill for them (Olteanu, 2015). In addition, they mainly improve their language skills to express their ideas and feelings in a better way. In this stage, the children may not yet logically think and reason for the events occurring around them, and because they are egocentric, they recognize the events around them through their own perceptions (Sigelman & Rider, 2008). Therefore, due to lack of tools for reasoning, the children in this stage are believed to unable to solve problems through logical thinking (Piaget 1952, 1954).

In the concrete operations stage, the children begin to think logically, tend to solve problems through systematic trial-and-error processes (Piaget, 1952, 1954). In addition, this stage's children can make concrete operations in their heads, such as

addition and subtraction (Sigelman & Rider, 2012). Moreover, the children in this stage are believed to make conclusions from their observations which involve cause and effect relationships and constructed around also others' perceptions (Sigelman & Rider, 2012). The concrete operations stage differs from the preoperational stage for two reasons: namely, decentration and reversibility. The decentration refers to the child's ability to focus on more than one aspect of a problem or an event while reversibility refers to the ability of recalling the steps of a process and moving between the steps in any order (Olteanu, 2015). For instance, a child in the concrete operations stage can remember the steps of an experiment, move back or forth between the multiple steps while he can have difficulty to remember all steps of an experiment and sequence of the experiment process or a solution path in the preoperational stage. Therefore, the children in the preoperational stage need adult support to remember the next or previous step of an experiment. This is also a result of their poor working memory, which is an executive function that involves the memory of numerous instructions, objects, words, or representations used in continuous thought processes (Cowan & Alloway, 2008). It is reported that working memory has time-related, space-related, and energy-related limits (Cowan & Alloway, 2008). It is also noted that a person's familiarity with the tested material and use of mnemonic strategies such as rehearsal, affects the person's working memory limits (Cowan & Alloway, 2008). For instance, older children are better than younger children in remembering multiple commands and are faster to accomplish a given task compared to younger ones (Bati, 2022).

According to Piaget, the ability to acquire a new cognitive content is linked to the children's stage of intellectual development (Müller et al., 2015). Indeed, each cognitive stage gives information about the child's readiness to acquire new knowledge or skill. When acquiring new knowledge, the children experience three major processes of cognitive learning which are assimilation, accommodation, and equilibration (Sigelman & Rider, 2012). For instance, when a child encounters with new knowledge, that child tries to incorporate it into the preexisting knowledge, which is called assimilation (Müller et al., 2015). During the assimilation, the child deals with new knowledge in his own terms, that is schema. During the accommodation, the child begins to understand that the new knowledge differs from

the existing knowledge, does not fit to the new experience, and changes the previous schema with the new one (Sigelman & Rider, 2012). When the new knowledge challenges the child, cognitive conflict could occur and can be reduced through equilibrium, by which mental stability is achieved (Müller et al., 2015).

In a constructivist classroom, the children experience child-centered activities rather than teacher-centered ones (Sigelman & Rider, 2012). During the child-centered activities, the children acquire the knowledge by themselves, and teachers are facilitators who support children's learning by asking explorable questions and learners who engage in learning by doing (Becker, 2002). When using instructional technology, constructivism requires to use interactive media, in which children actively explore the knowledge (Becker, 2002). Throughout technology-based constructivist learning experiences, the children engage in peer learning as they work with their collaborators when making small group projects. They learn to listen to others' opinions and experiences, share their knowledge with others, and receive feedback which may enable them to complete their own technological artifacts (Bers, 2021).

2.3.2. Constructionism

Seymour Papert who introduced the concept of *computational objects to think with*, wrote his book namely "Mindstorms: Children, Computers, and Powerful Ideas" and instead of giving a description for computational thinking, he explained what computational learning and thinking process includes, how computational processes can be developed and why it is necessary for today and future generations (1994). While explaining his theory and philosophies for children's learning, it seems that Papert especially focused on the computational cultures in which he believed that the children could overcome the cultural barriers to develop their thinking skills with space-age objects, such as computers. Indeed, Papert mainly emphasized the role of computers to create new relationships with knowledge by disregarding what can be learned in what age. He reported that computers and technological tools carry powerful ideas and are computational "objects to think with". Hence, he devoted himself to develop computational objects to challenge children for computational

thinking and learning and introduced “Logo Turtle” which is a cybernetic animal that can be controlled by the computer and requires mastery of numbers to be used by preschool children. For instance, the children should press *Forward* button and type *100* to move Turtle in a straight line 100 Turtle steps. While engaging with Turtle, the children can program the turtle on the screen, and discover new commands to move turtle in different formats. The child goes through a process that encompasses multiple learning processes, including making various shapes based on previously learned information, experimenting with different approaches, making mistakes and realizing those mistakes, coming up with a new plan, and trying until the goal is reached. According to Papert, the experiences gained in this process are powerful ideas that are easier to learn with technological tools and programming and constructs computational thinking. Therefore, in the lens of Papert, first, it can be stated that computational thinking is an open-ended process of learning that can be shaped by people with their self-engagements with technological tools and involves many steps while learning. According to Papert, children have opportunities for learning, thinking, and developing emotionally and cognitively when they interact with computational objects.

As a student of Piaget, Papert’s theoretical and philosophical ideas on how to think, process and learn knowledge were also based on his theories. As Piaget emphasized the need for children to learn by doing from the very beginning of their lives (1968), Papert emphasized the self-engagement of children to computationally develop their thinking skills. As Piaget underlined the concrete and manipulative learning materials and supported the constructivist learning, Papert contributed a new learning theory to the literature called “constructionism” (1991). Since creating and programming digital manipulatives requires active exploration, Papert (1993) needed a new term for learning. Moreover, similar to Piaget’s ideas about how children learn, constructionism learning theory focused on child-centered activities and self-construction of knowledge, but also on active exploration of digital manipulatives and tangible activities primarily accomplished with technological tools (Papert, 1993). According to Papert, it is important to make learning visible through concrete objects that learners discover in their own ways. Nevertheless, it’s also essential to employ facilitating teachers to create meaningful learning experiences about the

outside world (Papert, 1991). Therefore, this theory also emphasizes the need for facilitating teachers who give children the freedom to construct their own knowledge through challenging computational tasks and computational objects to think with.

2.3.3. Powerful Ideas of Computational Thinking

Table 2. 1. Seven Powerful Ideas of Computational Thinking (Bers, 2021; Relkin, 2018; Vizner, 2017)

The Powerful Ideas of Computational Thinking	The Description
Hardware/Software	Hardware is part of the information operating system that is set up to interpret the software and turn the information from the software into processes. Software, on the other hand, is a part of the information operating system installed to control and direct the hardware. While software provides instructions to the hardware, hardware is designed to send output data to the media. Hardware and software work together to perform the tasks of receiving, processing, and sending information.
Control Structures	The order in which a set of commands is started and executed. Iterative actions, conditional actions, loops, and events observed in a program include control structures.
Representation	It involves understanding that concepts or actions that can be represented by symbols.
Algorithms	It is a series of sequential steps followed to solve a problem or achieve a goal. It is basically related to sequencing skills.
Modularity	The fulfillment of a complex task by breaking it into small parts. Solving subproblems first to accomplish the main problem and writing/drawing/grouping the instructions to be followed to achieve the desired goal.
Debugging	It includes identifying and solving problems/errors that occur in a task/program. To exemplify, noticing that the Begin block is missing in a program or recognizing that the wait for clap block does not work since the sound sensor is not attached to the KIBO body are some debugging examples.
Design Process	Like the engineering design process, it is an iterative process that includes defining a problem or objective, determining the solution ways to be followed, testing, revising, and retesting them when necessary.

There are seven powerful ideas of computational thinking that are involved in the computational thinking and learning process (Bers, 2021). These powerful ideas are hardware/software, control structures, representation, algorithms, modularity,

debugging and design process (Bers, 2021) (see Table 2.1.). While algorithms are related to the ability to sequence the blocks in the right order and logical organization, modularity is related to understanding the commands of the task, breaking the large number of blocks into smaller parts, and completing the task (Dietz et al., 2019; Rijke et al., 2017). Control structures are related to understanding the functions of blocks and the cause and effects of programming tasks (Hassenfeld et al. 2020). Design process consists of iterative processes to develop programs or tangible materials with several steps (Bers, 2021). In the design process, the child first tries to sequence the blocks in the right order and then checks whether his design works. Should the solution fail to yield the desired outcomes, the child modifies its design until it becomes effective. Debugging involves the child's ability to define the problem in the task, search for solutions and manage one of the solutions, build the digital manipulative according to the solution, test, improve, and share the effect of the implemented solution on a particular problem (Misirli & Komis, 2023). For instance, the child should be able to identify the software or hardware error that resulted in the failure if they build and program the robot incorrectly while sequencing and scanning the blocks. In the sections that followed, each potent concept was also thoroughly explained in terms of how it was developed and how it helped to advance computational thinking.

2.3.3.1. Hardware/Software

Hardware refers to information operating system that may involve visible and hidden components that get the information from the software (Bers, 2021). For instance, the visible components of hardware involve keyboard, screen, printers, or the bodies of robotics while mainboards or graphics are hidden hardware parts. Software involves programming languages and is a part of the information operating system installed to control and direct the hardware (Bers, 2021; Vizner, 2017). Hardware and software execute together to send and receive data for a specific purpose. Young children should understand what hardware and software are and how programming is done with them in the context of early childhood education. In this way, an attempt should be made to teach these topics prior to teaching coding and programming. Then, other potent concepts could be effectively obtained.

2.3.3.2. Control structures

Control structures are also known as control flows which include the cause-and-effect relationships and sequential executions in the program (Zeng et al., 2023). In the programming context, the programmer knows and control the order of outputs that the robot or program will execute when engaging with the control structures. When programming an educational robotic in an early childhood setting, a child can explain the robotic's behaviors in the right order and understand the purpose and outcome of each action the robotic performs. Control structures also involve repeat functions, events, nested structures, loops, and conditionals (Bers, 2021). For instance, a child should use a repeat and end repeat blocks with number parameters when s/he is asked to move KIBO robotic further. In this process, the child uses loops and repeat functions when programming KIBO and knows the cause-and-effect of a program.

In order to understand control structures, the children should know patterns (Bers, 2021; Zeng et al., 2023). This is because pattern recognition and pattern construction are mainly used during control structures. Control structures also require understanding of conditions when programming with specific conditions. To illustrate, when a child programs KIBO using distance sensor, it is necessary to use distance parameters and repeat or if blocks. By doing this, particular conditions can be created that move KIBO in a complex execution.

2.3.3.3. Representation

Representation refers to the understanding that symbols represent specific concepts or actions and can be ordered to represent specific programs (Bers, 2021; Zeng et al., 2023). It includes models, diagrams, or symbols that are used to solve problems (Wing, 2006). When coding and programming, iconic representations are important to be understood by young children since they concretize abstract concepts when programming (Murcia & Taog, 2019). Acosta et al. (2023) reported that repetition patterns are one of the significant concepts when acquiring representation. In the early years, introducing representation also involves teaching basic concepts of

managing and constructing information. For example, children who learn to represent algorithms through block-based coding acquire core computational thinking skills such as algorithmic understanding and analytical thinking (Papadakis et al., 2016; Wing, 2006, 2008). In order to develop representation, the children can be involved in storytelling activities to represent the beginning, middle and end part of the story through creating algorithms. By doing this, the children learn to concretize the abstract concepts in their mind and express their ideas in representative models. The children can also engage in mathematic activities which involve sequencing, pattern recognition, and repetition patterns (Acosta et al., 2023). Lastly, the children can involve in programming activities which enable them to employ fundamental components of computational thinking throughout the process (Wing, 2006).

2.3.3.4. Algorithms

An algorithm is a problem-solving approach to define the steps of solution path (Kanaki & Kalogiannakis, 2022). It can be employed through algorithmic thinking which is a significant cognitive and procedural component of computational thinking as it includes creating series of steps to achieve a goal or solve a problem (Figueiredo et al., 2021; Wong & Jiang, 2018). It involves several skills such as sequencing, pattern recognition and construction, problem specification, algorithm design, analytical thinking, and abstraction (Bers, 2021; Futschek, 2006; Kanaki & Kalogiannakis, 2022; Wong & Jiang, 2018). Some research reports that using computer programming is not a requirement for using algorithmic thinking skills, but having these skills is because using computer programming is essential to coding, programming, problem decomposition, and methodical problem solving (Wong & Jiang, 2018). In addition, it is used in everyday life since it is a problem-solving approach and has a connection to mathematics (Figueiredo et al., 2021).

2.3.3.5. Modularity

Modularity is creating autonomous processes that involve frequently used commands for a specific function and solutions that can be used in different problems

(Atmatzidou & Demetriadis, 2016). It can be accepted as one of the problem-solving strategies as it entails task partitioning to manage and simplify a large task (Brusoni et al., 2007). In addition, modularity involves analyzing the problem in a modular way and dealing with complexity of a problem or goal through decomposing it into smaller parts to solve (Wimsatt, 2007). It is not only creating sub-tasks but also creating a large task including the smaller ones (Lavigne et al., 2020). Throughout the modularization process, the sub-problems can be either investigated in independent teams or individually, depending on the complexity of a task. A given problem would need to be solved in a more complex manner without pre-determined potential solutions in the absence of non-modular search modules (Brusoni et al., 2007). In fact, because of the numerous mistakes that could arise when testing the solution path, this also hinders someone from finishing the task more quickly. Thus, modularity is an important skill that is suggested to be acquired to successfully accomplish a complex task with a proper planning and prevent person from spending a lot of effort even for irreversible failures (Dietz et al., 2019).

While modularization, problem decomposition can be used to break a large task into manageable partitions (Brusoni et al., 2007). Wing (2008) reported that problem decomposition is a cornerstone of computational thinking and necessary to accomplish other foundational skills of computational thinking such as algorithmic thinking and abstraction. In addition, abstraction is necessary for understanding the relevant information towards the problem (Dietz et al., 2019). It is stated that while problem decomposition is one of the hardest computational concepts to acquire, abstraction is one of the most important skills to develop computational thinking skills (Rijke et al., 2018; Wing, 2006).

Lister (2011) reported that people begin to develop abstraction skills regardless of age and cognitive development stage. Dietz et al (2019) stated that without any intervention, by the time children reach the age of 4, they can evaluate the practicality of a solution path and can find the correct solution path when provided by two problem decomposition ways. This shows that young children begin to improve foundational decomposition skills of evaluating, reasoning, and understanding cause and effect relationships between variables beginning from the

early years (Cortesa et al., 2018). Decomposition and abstraction are related to the age of young children as previous research has shown that older children are faster and more successful at completing decomposition and abstraction tasks than younger children (Dietz et al., 2019; Rijke et al., 2018).

2.3.3.6. Debugging

The process of identifying bugs and errors in a program or algorithm, coming up with a workaround, and fixing errors to ensure proper operation is known as debugging (Misirli & Komis, 2023). During debugging, the program or the machine—including its hardware and software—may have errors. For example, in programming, an algorithm is set for a desired goal and coded using a programming language by the programmer. In this process, the programmer may make mistakes in the order of the codes or may forget to use essential hardware or software material that execute the program. Then, the programmer tries to understand the errors that may occurred when programming and determine some debugging strategies. Each strategy is set through assumptions, a control structure and evaluation mechanism (Metzger, 2004).

Debugging in early childhood involves an algorithm design through coding, detecting and fixing errors in an algorithm. The process also involves pattern recognition. It is a high-level skill for preschool children and hard to be acquired by them (Misirli & Komis, 2023). Nevertheless, when exposed to appropriate learning tools and experiences, young children can learn how to debug (Cuneo, 1986). Cuneo (1986) reported that young children can benefit from a variety of debugging strategies while using a tangible coding and programming material. The researcher indicated that even though the children do not have computer programming experience, they can be able to find missing commands in a program, fix the program through sequencing commands in a correct order. Flannery and Bers (2013) reported that when programming, young children can differ in terms of the debugging strategies they use and attributed this difference to their different levels of reasoning skills. This is because, older children were found to have better at debugging strategies compared to younger children.

2.3.3.7. Design Process

Design process is an open ended and iterative process in which several steps are followed to develop programs or tangible materials (Bers, 2021). The steps involve ask, imagine, plan, create, test, and improve, and share which is an infinite cycle (Bers, 2018). For example, the child can start the design process at any point and go back and forth between the steps as often as needed. If a child wishes to program KIBO by writing a program that includes the beginning, middle, and end of a specific story, he or she should consider the story's scenes and how KIBO can be used to represent them through coding. After imagining and planning the program, the child can create his/her own program by coding the beginning, middle and end scenes through programming blocks, then test and improve the program until it satisfies the need. As the child engage with design process, become familiar with the iterative process, develop their debugging skills through many iterations, and acquire perseverance (Bers, 2021). Lastly, the child shares a product which is meaningful for him, learn how to express himself in front of others and may also receive feedback from his peers which may help them to improve their product.

2.3.4. Positive Technological Development

Positive Technological Development (PTD) framework was developed by Bers (2012) to explain the contribution of coding to the personal growth of young children. The PTD involves six Cs of positive behaviors that are communication, collaboration, community building, choices of conduct, creativity, and content creation (Bers, 2021). It is believed that these 6Cs are implemented in classrooms when coding and using technology during the activities and develop positive behaviors in children (Bers, 2021). For instance, while communication, collaboration, and community building contribute to the intrapersonal domain, choices of conduct, creativity, and content creation develop interpersonal domain of personal growth (Bers, 2012) (see Figure 2.1).

Bers (2021) also reported that the 6Cs of positive behaviors are linked to the personal assets that are described under Positive Youth Development (PYD) as caring,

connection, contribution, competence, confidence, and character (Lerner et al., 2005). For instance, when *collaborating*, the children work together and have a common goal to accomplish, learn to *care* and assist others (Bers, 2021). When *communicating*, the children set *connections* with their peers by externalizing their thoughts and could have arguments with their friends which is believed to be more productive than their arguments with adults (Piaget, 1977). To illustrate, during the technology circles of Coding as Literacy-KIBO kindergarten curriculum, the children share their experiences when coding with KIBO. They may share the challenges they faced with, and the solutions they tried to fix their errors. When *community building*, the children are reminded for their *contribution* to the learning environment (Bers, 2021). For example, the children can share their small group projects in an exhibition or a demonstration day to celebrate their learning process with community members (Bers, 2021).

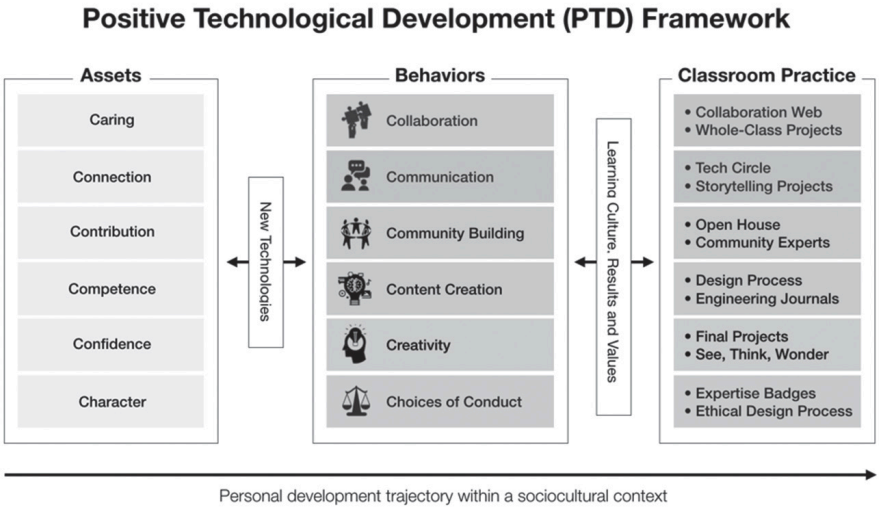

Figure 2. 1. PTD Behaviors, Related Assets, and Classroom Practices

Following that, during *content creation*, children design meaningful products for them and acquire a sense of *competence* (Bers, 2012). In actuality, children interact with computational behaviors and experience the design process while creating content (Bers, 2021). They might experience setbacks occasionally, but they will learn to control their emotions or create coping mechanisms to handle setbacks. The kids use their creativity throughout this phase, which boosts their self-esteem as well (Bers, 2021). For example, when a child tries to program KIBO for Hokey Pokey

dance, that child first create a content for the program which involves variety of programming blocks that were selected by the child and can be enrichened through their creativity. During this process, the child may face with errors, but when s/he can debug the errors, they develop self-confidence and become more self-motivated to further their programming engagements with KIBO. Lastly, Bers (2021) reported that when coding and programming, a child's *choices of conduct* construct or reflect his/her *character*. In particular, behavior decisions involve moral and ethical decisions made while utilizing their programming and coding abilities (Bers, 2021). For example, coding literacy has both positive and negative applications for society when used carelessly. Piaget (1965) emphasized that "moral development emerges from action" (Bers, 2021, p. 143). In this sense, the choices a child make when coding, develop their character and moral identity.

2.4. KIBO Educational Robotics

KIBO educational robotics was developed by DevTech research group with Marina Umaschi Bers' leadership and firstly named as KIWI in 2011. It was specifically designed for 4 to 7 years old children to create a tangible and developmentally appropriate playground in which children can discover powerful ideas (Bers, 2021). Easy integration of KIBO robotics into math, literacy, music, drama, and art activities is one of its most beneficial features (Bers, 2018). Due to its pluggable and detachable parts, KIBO can be used in a variety of ways by young children and early childhood educators (Relkin, 2018; Vizner, 2017). In fact, children learn programming as they interact with KIBO and observe the program they create by sequencing wooden blocks. They can also manipulate the blocks until they accomplish their intended goals. KIBO is a research-based, screen-free, hands-on robotics platform designed for teaching computer science and engineering in early childhood (Vizner, 2017).

Specifically, KIBO robotics platform involve many programming blocks, a variety of sensors and art platforms that can be used by children in accordance with their purposes. In addition, KIBO's body, motors, wheels, sensors, and light output is a hardware, while programming blocks are software (see Figure 2.1.). This is because,

the children benefit from the programming blocks as a programming language while the hardware parts are used to get information from the software and execute it. The programming blocks are made up wooden and has colorful labels with icons, texts and barcodes that represent specific actions for KIBO. It also has a scanner which is used to scan the barcodes on the programming blocks when programming KIBO.

Figure 2. 2. KIBO Robot, Programming Blocks, Sensors, and Parameters

When creating the algorithms for KIBO, it is first necessary to use green BEGIN block to start the program. In addition, the program should finish with the END block so that KIBO stops and finishes the program. The children can choose which of these two programming blocks to use in the program and arrange them in any order they like. Certain blocks are straightforward sound or motion blocks, while others are complex programming blocks that use if and repeat blocks.

When programming, the children can either set their own algorithms and program KIBO, or create a desired program given by the teacher. During this process, the children can use light output so that KIBO flashes light, use sound sensor to use wait for clap block, light, or distance sensor to create more complex programs involving if blocks and repeat blocks.

2.5. Related Studies

2.5.1. Studies Investigated Preschool Children's Computational Thinking Skills by Programming Activities through Educational Robotics

Angeli and Valanides (2020) investigated young children's computational thinking skills by implementing programming activities through Bee-Bot robotics. The researchers employed Type A and Type B scaffolding techniques when carrying out the activities. They also looked into how boys and girls differ in terms of computational learning. Both scaffolding approaches investigated the children's memory limitations in recalling the command sequences while taking gender differences into account. Type A was designed to observe the children's individualistic problem solving by arranging the command cards and creating an algorithm to move the Bee-Bot toward a desired goal. During Type A, the children experienced a kinesthetic, spatially oriented, and individualistic process while coding and programming. In Type B, however, each child was involved in the activity through the researcher's collaboration. The child stated the commands of the goal-oriented program, and the researcher recorded the program by drawing it for the child. The child then used the algorithm to test it with Bee-Bot. The comparative results showed that while the boys benefited more from the Type A technique, the girls benefited more from the Type B technique, that involved collaborative writing. The researchers concluded that the children mostly used their decomposition skills as a problem-solving strategy to cope with the complexity of the tasks.

Another study conducted with Bee-Bot and preschool children (Munoz-Repiso & Caballero-Gonzalez, 2019) conducted a quasi-experimental study and investigated the effect of robot-based coding activities on the debugging, sequencing, and action correspondence domains of computational thinking skills. While the experimental group was exposed to Bee-Bot coding, the control group received no treatment. The pretest results showed a significant difference between the groups in favor of the experimental group. That is, the children in the experimental group had statistically significantly higher scores than the control group. Although the groups were not equal to begin with, the researchers did not change the study groups. In addition, the groups were not normally distributed, so the researchers performed U of Mann-

Whitney and W of Wilcoxon tests. The results showed a significant difference between the groups' pretest and posttest scores in favor of the experimental group. Since the groups were not equivalent on the pretest scores, the researchers deepened the results in terms of the dimensions of computational thinking and reported a significant improvement in the sequencing, debugging, and correspondence dimensions for the experimental and control groups. In fact, the researchers reported that while there was a two-point difference between the experimental group children's pretest and posttest scores, the control group had less than a one-point difference in the dimensions of computational thinking. Lastly, according to the results of the Wilcoxon test, they concluded that there was a significant difference between the control and experimental groups after the treatment.

Misirli and Komis (2023) investigated computational thinking skills of preschool children (n = 526) aged 4-6 to explore the effect of programming a BeeBot robotics on debugging. The researchers followed design-based research through iterative practices of scenario-based teaching design. It was reported that the children's debugging performance is mostly related to their syntactic knowledge that is grammatical errors including the knowledge of which materials create a certain command and which commands result in a desired program. Debugging performance of children was also related to their semantic knowledge which is known as logical errors, control commands or spatial reasoning commands. Specifically, when the children make syntactical errors, they may write an incorrect program which may include wrong commands. When they do not create a syntactically correct ordered program, this also causes semantic errors since the robotic does not execute the purposed activity. Misirli and Komis (2023) also highlighted that the children in their study were not taught for the debugging steps, but they were able to follow the steps of debugging intuitively until they fulfil the program. Also, in contrast to previous research on using graphical interface for debugging skills, the children were able to manage debugging and created programs more than three commands due to tangible programming aspect of robotics.

In a national study, Bozkurt Polat (2023) conducted a mixed model study involving quantitative and qualitative research with 40 children, 20 children were in the

experimental group and 20 children were in the control group. To develop children's computational thinking skills and learning behaviors, the researcher designed a Tangible Robotic Coding Education Program that included 24 activities and was implemented three times a week using the BeeBot robotics kit. In addition, the researcher developed a 45-item instrument to measure the computational thinking skills of the preschool children by observing the children. The domains that were measured in the assessment tool was, abstraction, algorithmic thinking, decomposition, and generalization. The results showed that the Tangible Robotic Coding Education Program contributed to the computational thinking skills and learning behaviors of the preschool children, which was found to be permanent after the follow-up measure. The results also revealed that the program mostly contributed to the decomposition, algorithms, and abstraction domains of computational thinking.

In their study, Sullivan and Bers (2016a) implemented a KIWI robotics-based curriculum in eight weeks and sought to measure the progress of four- to seven-year-old children (pre-kindergarten through second grade) in building robots and programming tasks. The 8-weeks curriculum included seven activities that focused on learning the parts of robotics (sound, distance, and light sensors), constructing the parts of the KIWI robotic, programming it by sequencing and scanning the right blocks in the right order, and achieving tasks implemented with conditional statements. The researchers integrated the theme of "me and my community" into the activities and implemented the same curriculum with all the children. While implementing the curriculum, the researchers moved on to the next activity in the curriculum when the children were observed to be comfortable and ready to learn the more difficult tasks. The children's learning about the robots were controlled by robot parts task and programming abilities were measured by Solve-Its tasks. Due to the small sample of study, some assumptions could not be met, and the researchers conducted nonparametric test to control the effect of robotic-based activities to the children's learning about the robot parts. Their results showed a nonsignificant difference in children's programming skills in terms of child's grade. This means that all the children aged between four to seven years can learn KIWI robotics regardless of age. The children were found to successfully identify the motors and

light output (lantern) of the KIWI but have a difficulty while recognizing the sensors. The children were in fact perplexed by the lantern and light sensor because of their similar designs. The study's findings, which included an analysis of the children's programming development, also revealed that while older preschoolers had experience with conditional statement tasks and had learned every sensor in KIWI, younger kids could handle simpler tasks. Children in the second grade did, in fact, receive the highest programming scores. All children, including the kindergarten, the first and second grade children were successful at sequencing tasks, but challenged by Repeat Loop and Conditional statement tasks. While there were children who could not sequence the blocks in accordance with the story's sequence, there were also children who made a programming error that does not move KIWI. Most importantly, the pre-kindergarten children could success the Solve-It 1, which involves sequencing four instructions; and did not score as well on Solve-It 2 since it is harder than the first one due to five instructions. The Kruskal-Wallis test results revealed that there was a significant difference between pre-kindergarteners, kindergarteners, first and second grade children for the hard solve it task, as kindergarten, first and second grade children performed better than pre-kindergarten children on hard sequencing tasks. The researchers concluded that even pre-kindergarten children can master coding and programming tasks through educational robotics, but the number of instructions is important due to their limited working memory and capacity to remember the desired program. They also stated that while pre-kindergarten children were slow through the curriculum and tried to understand action correspondence between the blocks and KIWI's actions, the second graders were faster than other grades to learn the programming and robotic concepts and needed less instruction and help.

Moreover, Sullivan and Bers (2018) sought to measure the three to six-year-old children's sequencing skills in a mixed study. The researchers also searched how the KIBO robotics would affect social and personal behaviors of the preschool children. However, this part of the study was not handled in detail by the researchers. Instead of this, the researchers reported the results of their first purpose. Before the implication of curriculum that was named as "dances from around the world", early childhood teachers were trained by the researchers for KIBO robotic and its

functions. In addition, curriculum was examined with the teachers to understand how KIBO can be used in the activities the teachers will prepare. After that, early childhood teachers prepared activities for their students whose ages, prior knowledge and needs were different. More specifically, the teachers integrated KIBO robotics into their classroom to teach social and personal behaviors to their students. Subsequently, the teachers administered Solve-Its scale to identify sequencing skills of children before the implication. Following the intervention of the curriculum, the teachers assessed sequencing skills of children again through Solve-Its scale. The intervention results indicated that the tasks related to *if-conditional tasks* are the most complex activities for the participants in study. According to results, the children's mean scores for sequencing skills in each task decreased when the tasks get harder.

Acosta et al. (2023) carried out a mixed method longitudinal study with preschool children over three years to implement plugged-in activities through BeeBot robotics in the first year, iPad and online game in the second year and Cubetto robotics in the third year. The researchers mainly focused on the bridge between the algebraic thinking and computational thinking by instructing repetition patterns. Their results showed that the children's representation skills increased since they understood the logic in representation and made less errors when creating representations of patterns. In addition, with respect to the justification type, the 3- and 4-year-old children mostly use elaboration, while 5-year-old children mostly use validation. The researchers came to the conclusion that pattern recognition and decomposition play a crucial role in problem solving and serve as a link between computational and algebraic thinking. They also stressed how important it is to comprehend the patterns of repetition to succeed at representation.

Sullivan and Bers (2016b) conducted another study to investigate the programming performance of preschool children regarding their gender and grade levels. The researchers used KIWI robotics with 45 preschool children aged between 4-to-7 which included pre-kindergarteners, kindergarteners, first-graders, and second-graders. The researchers examined the children's success at robot parts task and Solve Its. The results regarding robot parts task showed that boys and girls did not significantly differ which means that they equally performed. However, first and second grade children performed better than kindergarteners. In addition, the Solve

Its revealed that boys performed better than girls at more complex programming tasks involving repeat loops with sensors.

Sullivan et al. (2013) aimed to measure the design, building, and coding skills of preschool children (n = 37) after they completed a one-week training. During the training, the researchers conducted math, art, and literacy activities with classroom teachers. Throughout the training, children also received an engineering design journal from the researchers, which contained a variety of literacy and artistic exercises. In the classroom, the children engaged in math, reading, and art activities as well as robotics. Robotics was incorporated into the classroom setting in this way. Data for the study was gathered via interviews with the children both before and after the training, as well as through observations made in the classroom. The interviews mainly explored the children's understanding of the robot and the engineer. Sullivan et al. (2013) reported that the children's knowledge and comprehension of engineer and robot increased after the training. In addition, the analysis of classroom observations showed that the children needed adult support while working with the robots. This was due to their inability to complete each of the design engineering stages prepared in the training.

In another study, Gonzalez and Munoz-Repiso (2017) explored the attitudes of preschool children towards Bee-bot robots which were introduced in classroom activities. When implementing activities with Bee-bot robots, children were first informed about the parts of the robot and its movements. This was followed by mathematical activities designed to teach many mathematical concepts such as spatial organization, number, and quantity. Finally, the children were administered a 3-point Likert-type attitude scale, which included three icons (happy face, neutral face, and sad face) referring to agree, neutral, and disagree. The results showed that the children enjoyed working with the Bee-bot robots and wanted to explore them in future activities.

2.5.2. Studies Investigated Preschool Children's Computational Thinking Skills by Screen-Based Programming Activities

Papadakis et al. (2016) conducted a small case study which involved 43 children and aimed to explore the effect of ScratchJr activities on the development of

computational thinking skills by teaching basic programming concepts in preschool education. The children were observed in terms of their abilities on understanding a single block and other blocks that create a complex project and transform the blocks in an integrated operational program. Their intervention was completed in 13 hours and constructivist learning approach was used so that all children actively engage in computational learning process. Because there was not developmentally appropriate assessment tool, the researchers examined the children's achievement in fundamental programming concepts and used the assessment tools provided by Portelance and Bers (2015) and Strawhacker et al. (2013). Their results revealed that there was no significant difference between the boys and girls in terms of computational and digital skills. In addition, the children's age was found to have no impact on understanding basic programming concepts. Their qualitative results showed that the children mostly use motion blocks, "move right" most frequently. In addition, they found that the preschool children made more errors as the number of characters increased on the screen and, they confused the left and right turn blocks. Papadakis et al. (2016) came to the conclusion that the ScratchJr program has a great deal of potential for enhancing children's sequencing skills because computational thinking skills are linked to analytical thinking and encompass a variety of skills like problem-solving, algorithm design, evaluation, and systematic analysis.

In addition, Del Olmo Munoz et al. (2020) carried out a quasi-experimental study to explore the effect of unplugged and plugged-in activities to the computational thinking skills of preschool children. 84 second-grade children participated in the study and Code.org courses were executed in two groups. In addition, the intervention was conducted in two phases. In the first phase, one group was engaged with plugged-in activities (control group) and another group engaged in unplugged activities (experimental group). In the second phase, both groups worked with the plugged-in activities. By doing this, experimental group engaged with both plugged-in and unplugged activities while the control group was only exposed to the plugged-in activities. There was a pretest, mid-test and posttest administrations that measured the children's motivation and computational thinking skills. While the pretest measured the computational thinking skills of both groups, the mid-test measured the motivation and computational thinking skills right after the first phase of the

intervention. Then, plugged in instructions were held and posttest administration occurred for the motivation and computational thinking skills of both groups. Their results revealed that while there was no significant between the groups regarding motivation, at the end of the treatment the motivation was found to have a slight decrease. In addition, unplugged instruction was more effective in developing the computational thinking skills of the children in easy and hard tasks at the mid-test and posttest. Their results also showed, although it was not significant, there was a greater improvement in computational thinking scores of the boys. Lastly, in the plugged-in group the girls were found to have less motivation compared to the boys while there were hardly differences in the boys and girls in the unplugged group.

Bers et al. (2023) conducted an experimental study with kindergarten, first-grade, and second-grade children by implementing Coding as Literacy-ScratchJr curriculum. The researchers examined the coding stages and computational thinking skills of experimental and control group children. While the coding stages of children were measured via Coding Stages Assessment (de Ruiter & Bers, 2021), the computational thinking skills were investigated via TechCheck. Their results revealed that Coding as Literacy-ScratchJr curriculum mostly contributed to the coding skills development in children compared to their computational thinking skills. The researchers concluded that a curriculum called Coding as Literacy-ScratchJr may be more effective in helping children learn how to code and program. The significance of abstraction skills in the development of computational thinking abilities was highlighted by the researchers, who also noted that although the purpose of ScratchJr was to help children conceptualize abstract ideas, the Coding as Literacy-ScratchJr curriculum could benefit from activities that enhance abstraction abilities.

Another study on the same topic was conducted by Portelance et al. (2015). They investigated the programming skills of preschool children (n = 62) through ScratchJr and a 6-week curriculum. The researchers mainly focused on the basic programming concepts and investigated the programming blocks that were mostly preferred by children when coding and programming with ScratchJr. The researchers examined the patterns in the use of programming blocks and checked differences across grades

which included kindergarten, first grade and second grade children. Their results revealed that mostly used blocks were motion blocks with “move right”, and a trigger block that was “start on green flag”. The least used programming block was “stop” block which is a control flow block. In addition, significant differences were observed between the grades. There were variations in how end blocks and trigger blocks were used. Post-hoc results showed that kindergarteners used trigger blocks more frequently than second graders. Additionally, it was discovered that second graders used fewer red end blocks than kindergarten and first graders. Lastly, the results revealed that while the first graders mostly used look blocks, second graders are significantly tended to use control flow blocks and sound blocks than the first graders and kindergarteners. The findings indicated that the age of the children and the software features that older children found more accessible than those of younger children were related to the way the children used different programming blocks.

2.5.3. Studies Investigated Preschool Children’s Computational Thinking Skills through Combining Plugged-in and Unplugged Activities

In their study, Yang et al. (2023) investigated the effects of plugged-in computational activities in a culturally responsive story-based environment and unplugged computational learning environment without storytelling. There were three research groups; one was exposed to story-inspired Matatalab robot programming (SIRP), story-inspired tablet programming through ScratchJr (SITP) and active control group who was exposed to the unplugged computational thinking education. The researchers also explored the effect of gender and socioeconomic status of the children to the SIRP and SITP interventions. Their findings revealed that there was a significant difference among SIRP and control group; SITP and control group, and SIRP and SITP groups. Pairwise comparisons also revealed that the children who experienced programming activities through robotics and story-based environment had the highest mean score. Respectively, the children who were exposed to the story inspired tablet programming through ScratchJr and the children who were exposed to the unplugged computational learning environment improved their mean scores respectively. Lastly, the researchers reported that there were no significant interactions between the groups in terms of gender and socioeconomic status of the

children. Yang et al. (2023) concluded that by adapting the coding as a literacy and coding as a playground approaches, they integrated more culturally responsive and effective approach to fill the gap between programming and computational thinking in Chinese classrooms.

A study conducted by Saxena et al. (2020) explored the development in computational thinking skills of children who are 3 to 6 ages and focused on pattern recognition, sequencing, and algorithm design. The activities were unplugged and plugged-in which were conducted through BeeBot robotics. The unplugged activities included recognizing the pattern on LEGO bricks and maintaining the pattern and sequencing stories that required to sequence the pictures of daily routines. The researchers stated that introducing pattern recognition and sequencing prepared the children for the BeeBot activities. The unplugged activities were also based on vocabulary building songs. For instance, the children behaved as robots and teacher or one of the children gave verbal commands to determine the movements of the children. The researchers found that all students successfully completed the pattern recognition activity and sequencing stories. However, four children out of 11 could not complete the algorithm design who were aged three to four and aged four to five.

Another study conducted by Yang et al. (2022) investigated the effect of robotics-based activities and block play on preschool children's computational thinking skills, sequencing ability, and self-regulation. The researchers used Matatalab coding set, and their activities mainly focused on the powerful ideas of correspondence, problem decomposition, algorithms, and pattern recognition while coding. While one group experienced robotics-based programming, other experimental group experienced structured block play and each experimental group was matched with control groups. While the children's computational thinking skills were measured by TechCheck, self-regulation skills were measured through Head-Toes-Knees-Shoulders (HTKS) and sequencing skills were measured through Picture Sequencing Task (PST). After the implementations, the treatment was found to contribute to the children's computational thinking, self-regulation and mostly sequencing skills which are strong predictors of early coding skills (Bers, 2021). In addition, there was a significant difference among the experimental groups as robotics-based

programming resulted in higher improvements in sequencing abilities but there was not a significant difference among experimental groups regarding computational thinking and self-regulation skills. Moreover, the children with lower self-regulation skills in the robot programming group were examined to have greater changes on their self-regulation skills after the treatment. This outcome was ascribed to the treatment's novelty, meaning that the children may have improved their lower baseline scores as a result of the positive aspects of differential treatment (Diamond & Lee, 2011; Flook et al., 2015).

In a national study, Değirmenci (2022) investigated the cognitive flexibility and computational thinking skills of preschool children through MTiny robotics-based and unplugged-activity based programs. The classroom teachers carried out the 15 activities in both programs over the course of eight weeks, integrating them with story books. Additionally, the researcher created a three-step rubric to assess preschoolers' computational thinking abilities. Three tasks measuring the abstraction, decomposition, modularity, representation, debugging, and evaluation domains were part of the rubric. The pilot study of the assessment tool with 208 children revealed that the children who did not take any coding implementation, had the lowest mean scores in modularity, generalization, and decomposition domains while the highest mean score was in abstraction. After the implementation of the programs, both interventions were found to significantly contribute to the cognitive flexibility and computational thinking skills of children. On the other hand, children exposed to activity-based coding practices scored higher on computational thinking than children exposed to educational robotics-based activities.

Another national study conducted by Canbeldek and Işıkoğlu (2023) focused on the effect of coding and robotics education on preschool children's creativity, self-regulation, problem solving, early mathematical reasoning and language skills. The researcher developed Coding and Robotics Education Program (PCP) comprising unplugged coding, robotic coding, and block coding activities. The program was constructionist, hands-on, play-based and based on concrete-to-abstract principles. The activities were also based on algorithm, sequencing, branching, loops, and decomposition. Lastly, the theme of the curriculum was environmental awareness

which was cultivated through math, science, arts, music, play and language activities. The curriculum involved 12 unplugged coding activities and implemented with grid-based coding games and focused on teaching debugging, sequencing, and algorithms. During the robotics coding activities, the researcher implemented 11 activities and used BeeBot, Doc and Matatalab in individually and small groups. Lastly, there were six block coding activities by ScratchJr application. With two experimental groups the activities were conducted twice a week and there were also two control groups to compare the effects of the intervention. Before the treatment, the children's skills were measured, and their pretest results revealed that all groups in the study to have similar developmental characteristics. After the implementation, independent t-test and paired samples t-test results showed a significant and positive impact on experimental group preschool children's measured skills. However, there was also significant difference in language and creativity scores of the control group. Therefore, the researcher also performed the General Linear Model analysis and found that the increase in creativity and language skills of the experimental group was significantly higher than the control group.

2.5.4. Studies Investigated K-12 Students' Computational Thinking Skills through Plugged-in and Unplugged Activities

Atmatzidou and Demetriadis (2016) investigated the effects of robotics-based activities on the computational thinking skills of junior high and high vocational students through The Lego Mindstorms NXT 2.0. The researchers implemented 11 lessons, 2 hour each, conducted once a week. It was aimed to explore the students' improvement in abstraction, generalization, algorithms, modularity, and decomposition domains of computational thinking. The researchers assessed the students' development in two stages. While the first assessment was administered after the fourth activity, the second assessment was accomplished after completing all the sessions. The researchers stated that independent of age and gender all students got the same level after completing the treatment, but significant differences occurred in some dimensions. In fact, they reported that there is a significant difference in favor between the modularity and decomposition dimensions among junior high school and high vocational students based on the results of paired t-tests.

However, analysis of covariance that assigned the first assessment as covariate revealed that statistically significant difference in only decomposition domain. Moreover, although a significant difference was observed among boys and girls in the first assessment -after the fourth activity-, after completing the treatment, there was no difference between boys and girls in terms of their computational thinking skills. Indeed, the first assessment results for junior high school children revealed that boys had higher mean scores than girls which was a significant difference. Nevertheless, the girls' scores catch up with boys' scores after completing all sessions. This result revealed that girls may need more time to develop their computational thinking skills and all students can reach at the same level with developmentally appropriate practices.

A study conducted by Kanaki and Kalogiannakis (2022) examined the algorithmic thinking skills of first and second grade students with respect to their age after implementing an environmental education course. The students' computational thinking skills were measured through administering *PhysGramming* which is a digital platform for four to eight years old children and involves puzzles categorized by the number of puzzle pieces. Furthermore, the algorithmic thinking levels of the students were classified as basic, medium, satisfactory, and excellent based on how well they performed on four-, six-, nine-, and twelve-piece puzzles. Their findings demonstrated that students' levels of algorithmic thinking increased as they grew older. In addition, attending a second grade was found to increase the possibility of being categorized in excellent and satisfactory levels while decreased the probability of being categorized as basic and medium level student.

Wong and Jiang (2018) investigated the effect of 12 lessons of computational thinking curriculum through coding and programming with Scratch on the algorithmic thinking and debugging skills of fifth grade children (n = 83). Participants had taken a 6-hour coding course in fourth grade and therefore had prior experience before the implementation. To measure algorithmic thinking and debugging skills, the researchers used a newly developed instrument consisting of eight multiple-choice questions. Their pre- and post-test results showed that the curriculum significantly contributed to the algorithmic thinking and debugging skills

of the fifth-grade students. However, there was no significant difference between pre-and posttest measurement regarding algorithm construction. The researchers attributed this result to the lack in correct use of computational concepts to create a correct algorithm. For instance, before creating an algorithm, the students should first be able to create the correct sequence of the commands, that is, they should first construct syntactically correct algorithms. In addition, they should be able to understand and know the loops and conditionals to create a repeating action. The researchers suggested to support children to develop these fundamental concepts independent from computer programming since previous research emphasized that abstract programming concepts are difficult for young children when they are involved in graphical interfaces.

In another study on the same subject, Rijke et al. (2018) investigated the abstraction and decomposition skills of six to 12 years old students (n = 200) through computational thinking tasks. The researchers also examined the differences among perceived difficulty, cognitive load, and flow in terms of their abstraction and decomposition. Their results revealed that age has a positive correlation with abstraction and decomposition development. Specifically, the researchers found that older children were better than younger children at the decomposition and abstraction tasks which are interrelated with algorithmic thinking skills. The researchers also noted that there was an interaction effect between gender and age which produced different results for girls and boys on the decomposition. Lastly, the researchers reported that there were no significant differences between young and older students in the perceived difficulty, cognitive load, and flow constructs. Therefore, it was concluded that young children can engage in abstraction and decomposition tasks.

In another study, Robledo-Castro et al. (2023) investigated the executive functions of primary school children through computational thinking intervention (n = 30). Implementing eight weeks of curriculum with the experimental group, the children were engaged with unplugged activities and plugged-in activities through coding and programming with computers while control group children did not receive any treatment. The children's executive functions and metacognitive processes were

measured by Neuropsychological Battery of Executive Functions in the pretest and posttest procedures. The researchers also used BANFE-2 scale which involved several subscales focusing on dorsolateral-working memory and dorsolateral-executive function. Their results revealed that after the computational thinking intervention, the children's development was mainly related to the children's prefrontal cortex based metacognitive processes. The results also revealed that the executive functions of children are associated with the anterior prefrontal cortex and dorsolateral cortex which gives information about a child's self-monitoring and metamemory issues; but not related to the orbitofrontal area which is related to the inhibition, rule tracking and risk-benefit processing.

CHAPTER 3

METHODOLOGY

Information regarding the study's methodological specifics is provided in this chapter. The study's design, goal, and methods for Phases 1 and 2, the population and sample in Phases 1 and 2, data collection tools, procedures for Phases 1 and 2, the findings of the pilot study in Phase 1, the Phase 2 intervention process, treatment validity, and study assumptions are all covered in this chapter.

3.1 Design of the Study

The research methodologies employed in this research differ depending on the study's goals. In this sense, Figure 3.1 provides the summary of the research methodologies, and each step was explained in the following paragraphs. While explaining the design of the study, the purpose of the study, the data collection tools, and the aim of each procedure was explained respectively.

This study employed quantitative research design which involved two parts: Phase 1 and Phase 2: Main study. In Phase 1, the main purpose was to translate data collection tools (the TechCheck-K and TACTIC-KIBO instruments) into Turkish and conduct the pilot study of the TechCheck-K instrument. The purpose was also translating the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum into Turkish to implement in the main study. After the translations and taking expert opinions for the data collection tools and curriculum, the psychometric issues of the TechCheck-K assessment tool were investigated (Relkin & Bers, 2021). Cross-sectional survey research was utilized to gather data from the participants at a specific point in time for this purpose (Fraenkel et al., 2011). The procedures required to conduct Phase 1 were explained and presented in Figure 3.2 in detail.

First, experimental research which is one of the quantitative research methodologies that allows researchers to examine cause-and-effect relationships (Fraenkel et al., 2011) was conducted in two phases in Phase 2. In the first phase, the quasi-experimental pretest-posttest control group design was conducted to know whether any change occurred in computational thinking skills of preschool children after the robotics-based coding implementation. This design allows researchers to control or reduce the threats to internal validity and assigns experiment and control groups non-randomly. As a result, the experimental and control groups were not assigned at random. Then the TechCheck-K assessment tool, which is unplugged, was used to administer the pretest and posttest.

Figure 3. 1. The Research Methodologies of the Current Research

Following that, in the second phase of Phase 2, a one-shot case study was carried out (Fraenkel et al., 2011) to examine the experimental group children's programming-based computational thinking skills by the TACTIC-KIBO assessment tool. This instrument is administered by programming tasks with KIBO educational robotics and the children's performance in programming related computational tasks is categorized to determine their computational thinking levels as *proto-programmers, early programmers,* and *programmers*.

In phase two, the parents' demographic data and the children's background characteristics in the control and experimental groups were also identified. To achieve this, comprehensive data on the parents and children was gathered, and certain child background factors were used to determine how much of an impact they had on the computational thinking abilities of the children in the control and experimental groups.

3.2. Phase 1

The first quantitative section of this research is called Phase 1. This section outlines the goals of the study, the methods used in the first phase, including participant selection, instrumentation, data collection, and analysis. Figure 3.2 provided a brief explanation of the procedures along with their goals.

3.2.1. The Purpose and Procedures Followed in Phase 1

This study aims to investigate the computational thinking skills of preschool children before and after implementing programming activities. For this purpose, necessary data collection tools were translated into Turkish language after taking permissions from the original authors of the instruments via e-mail. One of the data collection tools was the TACTIC-KIBO assessment tool (Relkin, 2018; Relkin & Bers, 2019), which categorizes children's computational thinking levels by administering programming tasks using KIBO robotics. The other instrument was the TechCheck-K assessment tool (Relkin & Bers, 2021), which describes children's computational thinking skills with multiple-choice questions and does not require programming

engagement with robotics before administration. After the translations were finished, these instruments were sent to researchers who were proficient in the English language. After making the required adjustments to the instruments, a pilot study was carried out utilizing Classical and Item Response Theory to investigate the validity and reliability issues with the TechCheck-K assessment tool (n = 352). Nevertheless, due to its requirement for prior KIBO robotics coding and programming experience, the TACTIC-KIBO assessment tool could not be piloted. To be more precise, the TACTIC-KIBO assessment uses KIBO robotics to conduct programming tasks that center on the seven key concepts of computational thinking skills. Therefore, the children without programming experience through KIBO robotics do not know necessary concepts and skills required to answer tasks in the TACTIC-KIBO and cannot be assessed for their programming-based computational thinking skills.

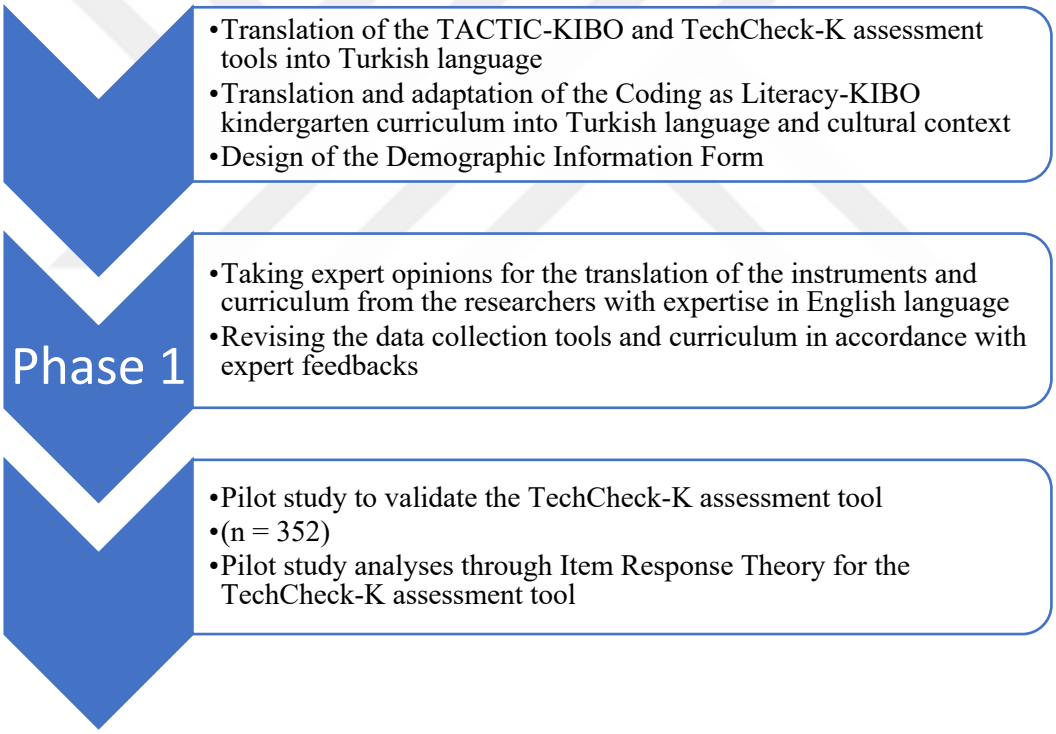

Figure 3. 2. The Procedures Followed in Phase 1

Another translation that was required in this study was the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum which involves coding and programming activities implemented through KIBO-educational robotics. Similarly, necessary permissions were obtained from the original authors of the curriculum, and it was

translated into the Turkish and adapted to the Turkish culture. The curriculum was then reviewed by researchers with experience in the English language for grammatical and clarity errors, and any necessary translation adjustments were made.

As previously mentioned, another goal of this research is to determine and examine preschoolers' background variables to see if they have an impact on their computational thinking abilities. For this purpose, the Demographic Information Form was designed during the first phase of this study. This tool was prepared to be filled out by parents and identifies the children's background factors including ages in month, gender, play experiences with the tangible materials -blocks and Legos- at home, coding experience, screen experience with digital games and average screen time with digital games at home. The form also involves questions about the parents' education level, number of children and age.

3.2.2. Data Collection Tool in Phase 1

The data collection in Phase 1 mainly administered for the pilot study of the TechCheck-K assessment tool. Data collection procedures were explained in detail in the next sections.

3.2.2.1. The TechCheck-K Assessment Tool

The purpose of this study was to investigate preschoolers' abilities in computational thinking. The TechCheck-K assessment tool, which was initially created by Relkin and Bers (2021) for kindergarten students, was translated into Turkish and utilized in this investigation for this reason. The original author's consent was emailed before any work on adapting the TechCheck-K instrument was done. The International Test Commission (ITC) Guidelines for Translating and Adapting Tests (2005) were adhered to during the translation and adaptation process of the English original test. The items were then examined for sentence structure, grammar, clarity, and young readership by two experts with expertise in assessment and evaluation and information and communication technologies, as well as English. The items were examined and changed to make them linguistically appropriate for the Turkish context, following the experts' recommendations.

The TechCheck-K instrument has 15 items and was constructed on seven powerful ideas of computational thinking (Bers, 2018) including hardware/software, control structures, representation, debugging, algorithms, and modularity. However, the instrument does not involve the powerful idea of a design process since it is an iterative and open-ended process that asks for solutions for problems and cannot be assessed with closed-ended and multiple-choice format questions (Relkin & Bers, 2021). In fact, the TechCheck-K instrument is a component of the TechCheck, which was also created to evaluate children's computational thinking abilities between the ages of five and nine (Relkin et al., 2020). The assessments, TechCheck and TechCheck-K, both have fifteen items. However, because of children's limited literacy and working memory development, TechCheck-K has three options per question, while TechCheck has four.

The TechCheck-K instrument does not require prior coding experience, this is because, the instrument does not include coding and programming questions or tasks that are implemented through programming tools (Relkin & Bers, 2021). Thus, it can be concluded that the TechCheck-K tool does not assess and classify children's computational thinking abilities by combining their programming developmental stage. This feature of the tool lets the researchers create pre- and post-test designs and assess how well curricula that include coding activities work. The questions in TechCheck-K involves technological concepts, sequencing tasks, finding the shortest path puzzles, finding the missing symbol in a pattern, object decomposition tasks, obstacle mazes, and symbol shape puzzles (Relkin et al., 2020). The tool can be administered on a tablet or laptop via the online https://www.qualtrics.com platforms. It can also be printed out or projected on a screen and taken with a pencil and paper. The TechCheck-K questions were read aloud to the children in small groups for the current study, and they used pencils to circle the right response on their own assessment sheets.

Examining the psychometric properties of TechCheck-K, the literature reveals that the instrument was first pilot tested with 89 children with a mean age of 5.84 and including 45 boys and 44 girls with no coding experience (Relkin & Bers, 2021). The children's mean scores were averaged in 8.26 and the administration time is around

23.43 minutes. The reliability analysis revealed that the pattern of correct response for the 15 items in TechCheck-K were highly correlated (r = .76) with the response pattern observed in the TechCheck which was pilot tested with first and second grade children. Moreover, items 6, 11, 12 and 13 were found to receive the least correct responses while items 1, 2, 3 and 4 received the highest correct responses in TechCheck-K instrument. In their pilot study (Relkin & Bers, 2021), the researchers did not investigate the validity issues of the instrument through Classical Test theories or Item Response theory. However, before the pilot study of TechCheck-K, they conducted a large-scale study through TechCheck (Relkin et al., 2020) which included data from 5-to 9 years old first- and second-grade children. In that study, the Cronbach's alpha of TechCheck was calculated as .68. Lastly, the item response analyses revealed that the TechCheck instrument is better at providing information about children with moderate and lower ability levels.

3.2.3 Participants and Data Collection Procedure for the TechCheck-K Assessment Tool

Prior to the translation and expert opinions of the instrument, the original authors' consent was emailed in order to start the pilot study of the TechCheck-K assessment tool. After that, the TechCheck-K instrument's pilot testing was started. Additionally, permission from the Provincial Directorate of National Education in Aksaray and the approval of the Middle East Technical University's (METU) Research Ethics Committee were required. Convenience sampling was favored when choosing participants for data collection from those who were nearby and easily reachable by the researcher (Fraenkel, et al., 2011).

Subsequently, negotiations were held with ten kindergarten principals in Aksaray city center, and information was gathered from the participating schools. Following negotiations, six public kindergartens in Aksaray's city center consented to take part in the TechCheck-K assessment tool pilot study. 352 children's data were gathered with their consent during the fall and spring semesters of the 2021–2022 school year. The TechCheck-K assessment tool was administered by the study's researcher to ensure consistency in its administration. This avoided bias in the data collectors as well.

Before starting data collection, the researcher first introduced an assessment tool to the children and then asked for their permission to participate. Verbal consent was obtained from the children to be included in the study. After that, in a classroom where the researcher first began collecting data, the TechCheck-K tool was administered to 10 children at a time, leaving enough space between their desks. However, because the children tended to look at each other's answers, these data were not included in the analysis of the pilot study. In fact, the researcher found that gathering data from ten children at once may also lead to issues with classroom management. As a result, the researcher limited the number of participants to five in each administration and carefully considered how to set up the classroom for each child's proximity. During administration, the researcher read each question to the children twice and gave them up to one minute to answer each question by circling the correct answer on their own assessment papers through pencils. The average administration time was 20 minutes.

The TechCheck-K assessment consists of fifteen items, each of which has three possible answers and one right response. The sixth question (Which shapes can you use to make this? -triangle-) and the seventh question (Which shapes do you need to make this snowman?) require children to select the option that includes the group of correct shapes. However, while some children chose the correct alternative, others preferred to match the shapes by using arrows to form the triangle for the sixth question and the snowman for the seventh question. In this context, all children who chose the correct option and matched the shapes were considered to have given correct answers.

3.2.3.1 The Pilot Study Results of the TechCheck-K Assessment Tool

A total of 163 girls and 189 boys, ages 69 months on average (minimum = 49, maximum = 86), were given the TechCheck-K assessment tool. The TechCheck-K scores were distributed normally, with a minimum score of 0 and a maximum score of 15 out of 15. TechCheck-K scores ranged from 7.33 to 2.60 on average. It is also discovered that the skewness and kurtosis values range from -2 to 2. Preschoolers' scores on computational thinking skills seem to be normally distributed, based on the

histogram shown in Figure 3.3. To control the reliability of TechCheck-K, Cronbach's alpha value was calculated, and alpha value was found to be .63 which is an acceptable level of internal consistency for psychological assessments (Hinton et al., 2004). In the pilot study of TechCheck-K (Relkin & Bers, 2021), the Cronbach's alpha value was not reported, and comparisons could not be made. However, the researchers reported that the pattern of correct responses for the 15 items in TechCheck-K was highly correlated (r = .76) with the pattern of responses observed in the TechCheck, which had a Cronbach's alpha value of .68 (Relkin et al., 2020). This means that the TechCheck-K test assesses the same fundamental capacity for computational thinking. In the pilot study of TechCheck (Relkin et al., 2020), the researchers reported that because computational thinking skill is not a unified instead a single construct skill, it may decrease the internal consistency of the scale. Specifically, the TechCheck-K and TechCheck instruments are constructed on six domains of thinking and problem-solving skills. Despite the fact that other instruments (Zapata-Caceres et al., 2020) had higher alpha values, they were discovered to have fewer categories, which is thought to improve an instrument's internal consistency. Considering these results, it can be concluded that because computational thinking skills do not have a unified construct and measured by six domains, a moderate level of Cronbach's alpha value was found and this result was in line with previous research conducted with TechCheck (Relkin et al., 2020).

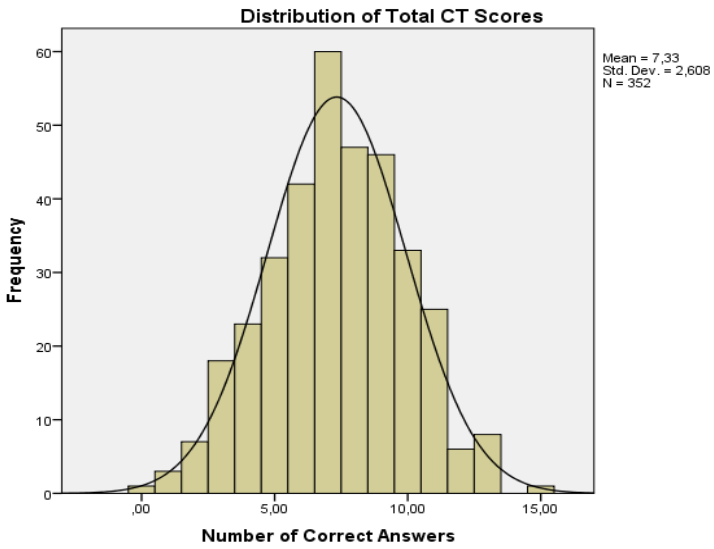

Figure 3. 3. Distribution of Children's Computational Thinking Scores

In addition, the children's response average was checked for each item. The valid, missing, and correct answers were given for each item in Table 2.4. It was found that while the missing questions, which were left blank, were the 10th, 11th, and 13th questions, most answered questions were the 2nd, 3rd, and 7th questions. Moreover, while the 2nd, 3rd, 14th, and 15th questions received the most correct answers; the 10th, 11th, 12th, and 13th questions received the least correct answers (see Figure 3.4 to check the items). The questions numbered 10, 11, 12, and 13 that children did not know the answers to and the ones that had the majority of wrong answers appeared to be difficult for them. In addition, group comparisons were made by an independent sample t-test, and computational thinking scores were examined for boys and girls. Although the girls had better performance in computational thinking scores, there was no significant difference between the computational thinking scores of girls (M = 7.57, SD =2.49) and boys (M = 7.04, SD = 2.64; t (305) = -1.89, p = .058, two-tailed). The magnitude of the difference in the means (mean difference = .53, 95% *CI:* -1.07 to 0.18) was very small (eta squared = .2). The results were presented below (see Tables 3.1. and 3.2.).

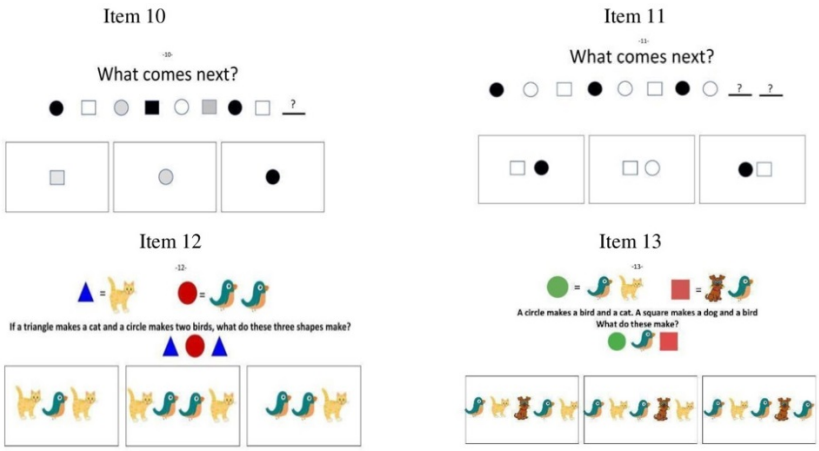

Figure 3. 4. The TechCheck-K items that were not answered by the children and mostly had incorrect answer.

Table 3. 1. Independent Samples T-test Results

	Gender	*n*	*M*	*SD*	*p*
Computational Thinking Skills	Boys	189	7.04	2.64	.058
	Girls	163	7.57	2.55	.058

Table 3. 2. Descriptive Statistics for Each Item

Item Numbers

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Valid Answer	328	344	338	322	337	336	345	329	331	291	284	304	287	331	327
Missing Answer	25	9	15	31	16	17	8	24	22	62	69	49	66	22	26
Correct Answer	142	274	272	222	184	138	176	155	210	82	62	98	82	257	227
Incorrect Answer	185	69	65	99	152	197	168	173	120	208	221	205	204	117	99
M	,43	,79	,8	,68	,54	,41	,51	,47	,63	,28	,21	,32	,28	,77	,69
SD	,49	,4	,39	,46	,49	,49	,5	,49	,48	,45	,41	,46	,45	,41	,46

3.2.3.2 The Pilot Study Results through the Lens of the Item Response Theory

Item Response Theory was also used to examine the TechCheck-K assessment tool's items as a unit of analysis. This is due to the fact that Item Response Theory, particularly when developing or modifying instruments, provides more information than Classical Test Theories (Baker & Kim, 2017). To achieve this, item analyses were performed using the R Studio version 1.2 ITM package, and the Two-parameter Logistic Model (2PLM) was employed.

Before conducting the item response analyses, the sample size is one of the important factors that affect the item parameter estimations in a study (Şahin & Anıl, 2016; Lord, 1968). Another important factor affecting the item parameter estimations is the test length, that is, the number of items (Lord, 1968). For example, a sample of 250 with 15 items (Bentler & Chou, 1987; Harwell & Janosky, 1991), 500 or 750 (Lim & Drasgow, 1990; Stone, 1992) with 20 items are suggested to run a two-parameter logistic model (2PLM). Therefore, a sample of 352 with 15 items was tested for item parameter estimation with 2PLM, as the criteria of sample size and number of items were met (Bentler & Chou, 1987). The two parameters that are controlled when doing 2PLM are discrimination and difficulty of the items. This seeks to clarify each item's capacity for discrimination as well as the challenge of providing an accurate response to every question (Baker & Kim, 2017).

Specifically, the item difficulty provides information about the percentage of students who answered the item correctly, and the discrimination parameter shows the relationship between the student's performance on each item and the overall test score (Baker & Kim, 2017). It is thought that an item's capacity to distinguish between students of high and low ability can be revealed to a greater extent if the discrimination parameter is larger (Baker & Kim, 2017). For example, an easy item will work best with low-ability students, while a hard item will work best with high-ability students. Considering this information, the discrimination and difficulty indices were checked by the item characteristic curves, coefficient values, and test information function.

First, *the item characteristics curve* which provides information about the monotonicity of the instrument was examined. Specifically, the item characteristics curve is the basic building block of item response theory and gives information about the difficulty (*b*) and discrimination (*a*) parameters for each item in an instrument (Baker & Kim, 2017). The item characteristic curve is an S-shaped curve, and each item has its own item characteristic curve. When examining the curves, the steepness in the middle section is checked. Baker and Kim (2017, p.3) report that "the steeper the curve, the better the item can discriminate. The flatter the curve, the less the item can discriminate since the probability of correct response at low ability levels is nearly the same as it is at high ability levels''. The test takers are grouped according to their ability levels, and the left line of the graph indicates the likelihood of a discriminating correct response. The right line indicates the difficulty of the item. As shown in Figure 3.5., except for item 12, other items seem to reveal the true relationship between the trait and the responses to the item. This is because item 12 has a flatter curve on the graph and has less discrimination power to identify the probability of a correct response at low-ability levels and high-ability levels.

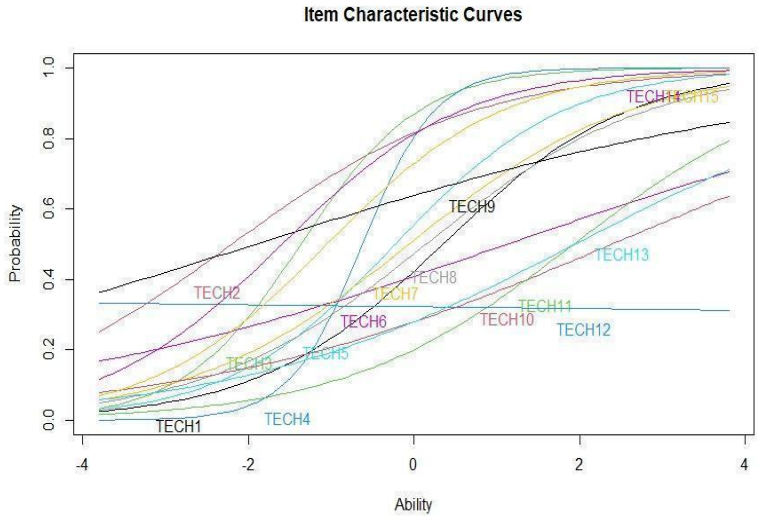

Figure 3.5 The Item Characteristics Curve

Following that, the coefficient values for each item in the TechCheck-K assessment tool were examined (see Table 3.3.). While examining the values, the difficulty indexes were checked between -4 and +4 values. Specifically, while the coefficients

which are near the -4 value are accepted as easy items, the coefficients which are near the +4 value were accepted as difficult. In this context, the coefficient value of item 12 seems to be different among other items. In addition, items 2, 3, 4, 9, 14, and 15 were found to be the easiest, and items 10, 11, and 13 were examined to be the hardest questions among others. These results were found to be similar to the results presented in Table 3.3. Moreover, because the items with the larger discrimination parameter value are better at giving information about the ability of an item to discrimination power, all items were found to be good at discriminating the high and low-ability students except item 12.

Table 3.3 Coefficients of the TechCheck-K Items

Items	Discrimination Index (*a*)	Difficulty Index (*b*)	Items	Discrimination Index (*a*)	Difficulty Index (*b*)
1	0.885	0.349	***9***	0.297	-1.904
2	0.680	-2.196	***10***	0.397	2.391
3	1.393	-1.364	***11***	0.717	1.931
4	2.283	-0.620	***12***	-0.012	-60.551
5	0.980	-0.224	***13***	0.486	1.947
6	0.325	1.130	***14***	0.919	-1.5923
7	0.748	-0.059	***15***	0.942	-1.050
8	0.746	0.148			

The quantity of information gleaned from test items is another piece of information that the item response analysis offers (Baker & Kim, 2017, p.91). Figure 3.6. demonstrates the item information for all items and Figure 3.7. detailly shows the item information curves for each item. The item information curves reveal that all items can estimate the ability over other ability levels and are useful for examinees in the bottom half of ability. Finally, the test information function indicates how much data the test collected at each ability (Baker & Kim, 2017). If the test can estimate ability across the entire ability score range and if the level of test information is higher than a single item information function, these are useful definitions provided

by the test information function. The total of the item information at that level is, in fact, the test information function. “The amount of information has a maximum at an ability level of –1.0 and about 3 for the ability range of -2 ≤ θ ≤ 0.0” (note that θ means *ability*) (Baker & Kim, 2017, p. 90).

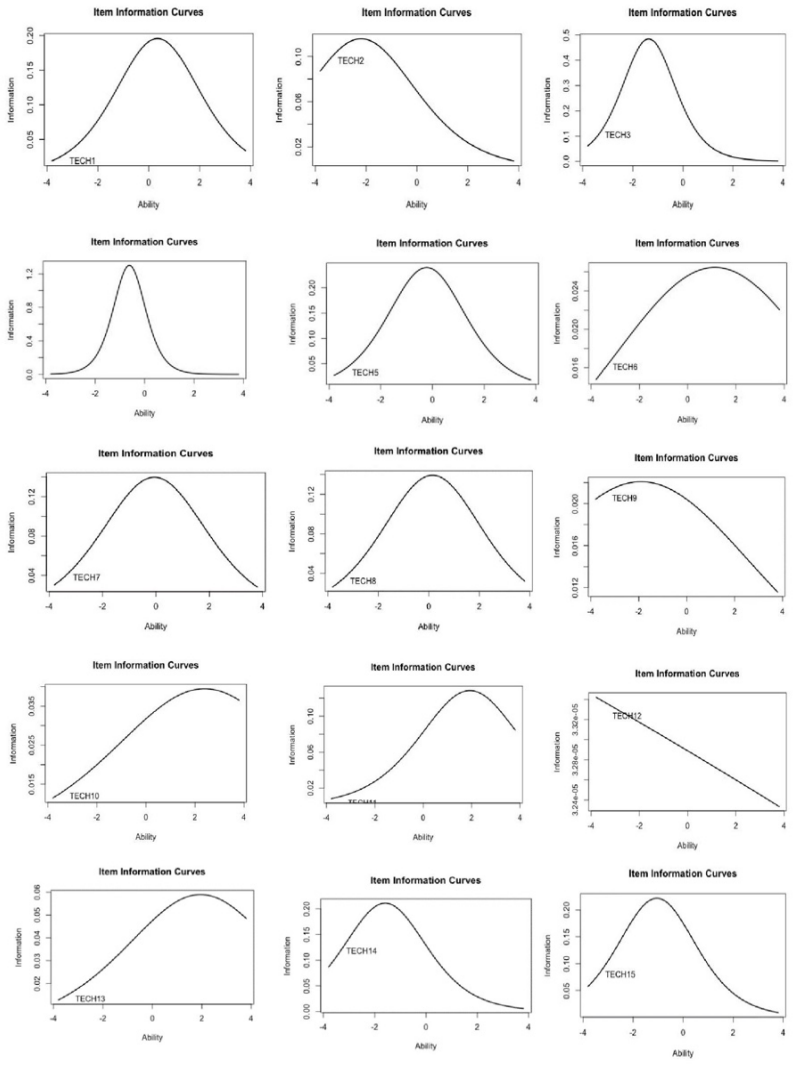

Figure 3.6 The Item Information Curves for All Items in the TechCheck-K

In this context, the TechCheck-K instrument’s test information function indicates that the test yields the most information at ability levels of -1.0 and -2 ≤ θ ≤ 0.0, with the amount of information increasing steadily as the ability levels differ (see Figure 3.8). This is also because the TechCheck-K instrument can estimate the ability levels with some precision near the center and hardly estimate when the

ability level approaches the extremes of the scale since the amount of test information decreases significantly.

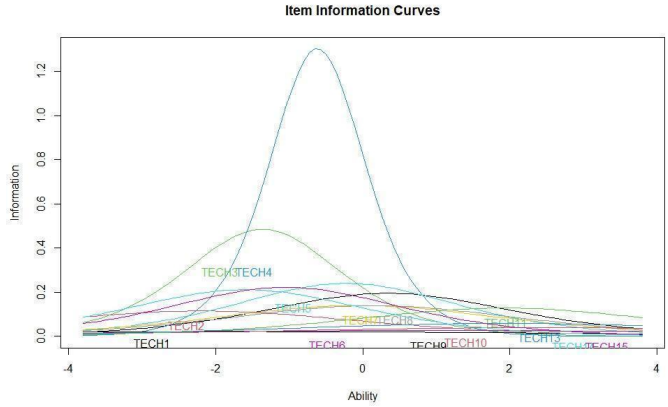

Figure 3.7 Item Information Curves for Each Item in the TechCheck-K

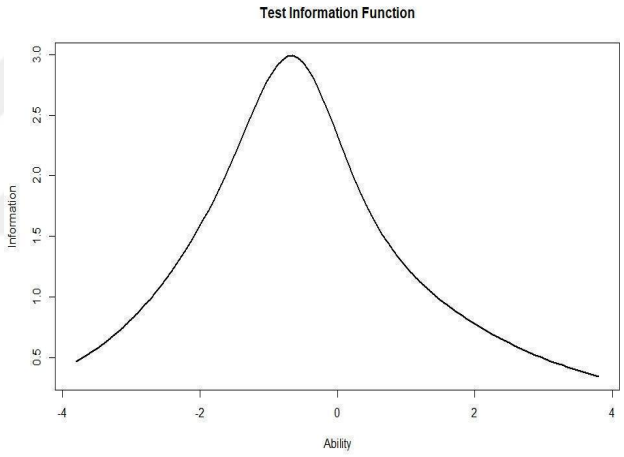

Figure 3.8 Test Information Function of the TechCheck-K

3.2.4. Four Preparatory Activities for Directional Language and Colors

Before conducting the experimental study, the researcher prepared and implemented four activities to reinforce children's knowledge about directions and colors. This is because, the researchers who will implement computational activities are suggested to ensure that the children have some directional language and necessary concepts that are employed during the computational activities (Saxena et al., 2020). For this reason, the researcher prepared four preparatory activities to be implemented before

the Coding as Literacy-KIBO kindergarten curriculum with the experimental group over two days, each lasting a maximum of an hour and a half. The outcomes and indicators, concepts, and educational plans covered in the activities are summarized in Table 3.4.

Table 3.4 The Contents of the Preparatory Activities

Activities		**Outcomes and Indicators**	**Concepts and Words**	**Type of the Activity**
Day 1	Activity 1: I'm learning the directions!	*Cognitive Development Domain:* Outcome 13. Recognizes symbols used in daily life (Indicators: Shows the appropriate symbol for the given description. Says the meaning of the symbol shown.)	Go forward, go backward, turn left, turn right	Whole Group, Game with Song and Literacy Activity
	Activity 2: Dance and freeze!	*Cognitive Development Domain:* Outcome 10. Applies the instructions related to location in space. (Indicators: Takes location in space.) *Motor Development Domain:* Outcome 1. Makes displacement movements (Indicators: Advances a certain distance by jumping with one foot. Advances a certain distance by making a sliding step). Outcome 5. Moves to music and rhythm (Indicators: Dances to music and rhythm. Performs various movements one after the other accompanied by music and rhythm).	Go forward, go backward, turn left, turn right, dance-freeze, slide to the left, slide to the right, on the left foot, on the right foot, right arm, left arm, turn around, spin	Whole Group Integrated Play, Music, and Movement Activity
Day 2	Activity 3: Let's find our home!	*Cognitive Development Domain:* Outcome 10. Applies the instructions related to location in space. (Indicators: Takes location in space. Uses maps and sketches.) Outcome 19. Produces solutions to problem situations (Indicators: States the problem. Suggests various solutions to the problem. Selects one of the solutions. Explains the reason for the solution he/she chooses. Tries the solution he/she chooses. When	Go forward, go backward, turn left, turn right, green, red	Small Group Play Activity

	he/she cannot reach the solution, he/she chooses a new solution).		
Activity 4: I'm playing with the colors!	*Motor Development Domain:* Outcome 6. Matches objects or assets according to their properties (Indicators: Distinguishes and matches objects/assets according to their color).	Green, red, yellow, blue, purple	Whole Group, Integrated Drama Warm-up, and Art Activity

When the developmental areas of preparatory activities are examined, it can be said that the outcomes and indicators focus on the cognitive and motor development of children. Those outcomes and indicators were taken from the preschool education program that the Ministry of National Education has been implementing since 2013. All activities are designed to take children at the center of learning by doing. Furthermore, children learned colors and directed language by using their bodies, according to the prevalent embodied learning approach.

In fact, both short- and long-term memory is supported by embodied learning experiences (Macedonia, 2019). Additionally, it is thought that motor movements facilitate word storage by activating procedural memory (Engelkamp, 2001). This suggests that the body is an effective means of representing and storing knowledge (Ullman & Lovelett, 2018). Hence, in this research, the preparatory activities were mainly conducted by embodied practices that included literacy, game with songs, play, drama, and movement activities.

In addition, the third activity was mainly focused on employing problem-solving skills. Specifically, the children worked in small groups to reach a target provided by the researcher while learning directional language.

In fact, the children found it easier to encode and activate images in working memory when they were using the correct directional cards to express the path to the goal while they were solving problems (Richardson et al., 2003). Lastly, the preparatory activities also served as an opportunity for the researcher and the children to meet and mingle before the robotic coding implication.

3.3. Phase 2: The Main Study

The second quantitative section of this research is called Phase 2. The goal and methods, participant selection, instrumentation, data collection, and analysis procedures that were carried out in the Phase 2 and are compiled in Table 3. 6 are presented in this section.

3.3.1. Purpose and Procedures Followed in Phase 2: The Main Study

After Phase 1, Phase 2 was conducted, and the preschoolers' computational thinking abilities were examined in two steps (see Table 3.5). As mentioned above, the main goal of this study is to elucidate the development of preschool children's computational thinking skills before and after implementing coding/programming activities implemented with educational robotics. For this purpose, two main research questions were formulated and presented in Table 3.6 as second and third main research questions. To this aim, the second main research question was examined in the first step of Phase 2 using a quasi-experimental pretest-posttest control group design to see if the implementation affected the preschoolers' computational thinking abilities. First, the TechCheck-K assessment tool was administered to the experimental and control group children to assess their baseline computational thinking skills. Then, the children were post-tested with the TechCheck-K assessment tool after implementing the programming activities with educational robotics -Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum- to examine whether there was a significant change in the computational thinking skills of the experimental group children.

In the second step of Phase 2, a one-shot case study was carried out (Fraenkel et al., 2011) to investigate the experimental group children's programming-related computational thinking skills by administrating the TACTIC-KIBO assessment tool. By doing this, the experimental group children's programming-related computational thinking skills were detected and categorized concerning their programming

performance. This procedure was investigated with the third main research question and analyzed with frequencies (see Table 3.6).

Table 3.5 The Purposes, Procedures, and Assessment Tools in Phase 2: The Main Study

The Procedures	Assessment Tools	Aim	Administration Time
PRETEST	The TechCheck-K Assessment Tool	To evaluate experimental and control group children's computational thinking skills	Before the intervention of four preparatory activities
The implication of four preparatory activities	-	To teach the experimental group children in terms of directional language and colors	
Intervention Time: ***The Coding as Literacy Curriculum with KIBO Robotics***			
POSTTEST	The TechCheck-K Assessment Tool	To evaluate the difference between pretest and posttest scores in the experimental and control group children's computational skills.	After the Coding as Literacy-KIBO implementation
Demographic Information Forms were sent to the mothers and fathers.	The Demographic Information Form	To describe the background factors of the experimental and control group children and to identify the contribution of background factors to their computational thinking skills.	After the Coding as Literacy-KIBO implementation
TACTIC-KIBO Administration	The TACTIC-KIBO Assessment Tool	To identify the experimental group children's computational thinking levels as proto-programmer, early programmer, and programmer.	After the TechCheck-K administration

Last but not least, as Table 3.6 demonstrates, the fourth primary research question looked into the background characteristics of the children in the experimental and control groups and their potential to influence their computational thinking abilities. In order to achieve this, the children's background circumstances were first described. After that, the contribution of background factors to the posttest computational thinking scores of experimental and control groups was investigated by their adjusted mean scores on the pretest. This allowed the pretest mean scores of the children in the experimental and control groups to be matched, validating the significance test that was carried out to examine the impact of background factors on computational thinking abilities. The background factors involved the children's gender, ages in month, and tangible play experiences at home, screen time with digital games, coding, and unplugged computer science experience, and toys and applications that the children mostly play and tackle at home. However, only gender and age in months were used in the analysis due to number of samples. The data were gathered through the Demographic Information Form.

3.3.2. Data Collection Tools in the Main Study

The data collection tools used in the main study involve the Demographic Information Form, the TechCheck-K assessment, and the TACTIC-KIBO assessment. Since the TechCheck-K instrument was also used in Phase 1 and its psychometric properties were given in the pilot study section, it will not be discussed again in Phase 2.

3.3.2.1 The Demographic Information Form

The demographic information form includes 13 questions about children's background factors, such as date of birth, gender, prior experience with ScratchJr or any other digital manipulatives, screen time with digital games and tangible play experiences at home and demographic information of the parents such as age, number of children and education level (see Appendix A).

Table 3.6 The Research Questions, Purposes, and Related Analyses in the Main Study*

The Main Research Questions		**The Purpose of the Research Question**	
2. Do computational thinking skills of preschool children develop with the educational robotics-based coding implementation?		To identify the effect of Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum on children's computational thinking skills	
The Sub-Research Questions	**The Purposes of the Sub-Research Questions**	**The Instrument**	**The Analysis**
2a. What are the general patterns of experimental and control group preschool children's pretest computational thinking scores?	To identify the experimental and control group preschool children's pretest means, the items with the highest and lowest mean scores and the scores in the domains of computational thinking.	**The TechCheck-K Assessment Tool**	Descriptive Statistics
2b. Is there a significant difference between the experimental and the control group of preschool children in terms of their computational thinking skills before the educational robotics-based coding implementation?	To explain the experimental and control group differences in terms of their baseline computational thinking skills.		Independent Samples T-test
2c. What are the general patterns of experimental and control group preschool children's posttest computational thinking scores?	To identify the experimental and control group preschool children's posttest means, the items with the highest and lowest mean scores and the change in the domains of computational thinking.		Descriptive Statistics and Post-hoc item analysis
2d. Does educational robotics-based coding implementation produce greater computational thinking scores in the experimental group children than in the control group children after adjusting for pretest differences in computational thinking scores?	To detect effect of the CAL-KIBO kindergarten curriculum on the computational thinking skills of experimental group preschool children by equating the compared group on an extraneous variable and using the adjusted mean scores.		One-Way Analysis of Covariance (ANCOVA)
3. What are the programming related computational thinking levels of preschool children in the experimental group after the educational robotics-based coding implementation?	To categorize the experimental group children's programming related computational thinking skills	The TACTIC-KIBO	Descriptive Statistics
4. Do background factors (age in month and gender) of preschool children in the experimental and control group make a difference in their posttest computational thinking scores after adjusting for pretest differences in computational thinking scores?	To identify the contribution of background factors to the experimental and control group preschool children's posttest computational thinking skills	The Demographic Information Form	Three-Way ANCOVA

*The first main research question was investigated in the pilot study of the TechCheck-K instrument

After the implementation, demographic information form was sent to the mothers and fathers of children to fill out. This is due to the fact that the questions regarding coding practices, daily screen time spent playing digital games, and digital gaming experiences in the demographic information form may have an impact on parents, who may then provide experiences that could affect their children's backgrounds during the intervention. One of the questions in the form, for instance, asked about parents' awareness of and involvement with the ScratchJr application by their children. A parent who answered this question before the intervention would question ScratchJr and install it on the tablet for their child to play with. This could result in some children gaining additional coding and programming experience during the robotic coding intervention without the researcher's knowledge, thus invalidating the measurement of the effect of the intervention. This is also due to the fact that, ScratchJr is one of the most popular and widely used coding and programming application in the world and has an interface that includes programming blocks with icons that are similar to the symbols on the KIBO's programming blocks. Given this, the researcher gave the parents the demographic information form following the intervention, which may have allowed them to manage the study's external threats and yield more reliable findings.

3.3.2.2 The TACTIC-KIBO Assessment Tool

The TACTIC-KIBO assessment tool was developed by Relkin (2018) to assess the computational thinking skills of preschool children by administering programming tasks with KIBO robotics. It is administered through one-by-one interviews and the children's success in the programming is scored with 0 (unsatisfactory) and 1 (satisfactory). The TACTIC-KIBO was constructed around *the Seven Powerful Ideas of Computational Thinking* theory which was created by Bers (2018). Specifically, the seven powerful ideas are accepted as core concepts and skills contributing to computational thinking skills and suggested to be developed in young children from their early years (Bers, 2010, 2018). These powerful ideas are *hardware/software, control structures, representation, algorithms, modularity, debugging,* and *design process* (Bers, 2018). Each powerful idea was briefly explained in Table 3.7.

Table 3.7 The Seven Powerful Ideas of Computational Thinking Skills

The Seven Powerful Ideas of Computational Thinking	Description
Hardware/Software	Hardware is the part of the information operating system that is set up to interpret the software and turn the information from the software into processes while software is a part of the information operating system installed to control and direct the hardware.
Control Structures	The order in which a set of commands is started and executed. Iterative actions, conditional actions, loops, and actions observed in a program include control structures.
Representation	It involves understanding that concepts or actions can be represented by symbols.
Algorithms	It is a series of sequential steps followed to solve a problem or achieve a goal and related to sequencing skills.
Modularity	The fulfillment of a complex task by breaking it into small parts is the management of a process by breaking it down into more manageable parts/stages.
Debugging	It includes identifying and solving problems/errors that occur in a task/program.
Design Process	Similar to the engineering design process, it is an iterative process that includes defining a problem or objective, determining the solution ways to be followed, testing, revising and retesting them when necessary.

In addition to the Seven Powerful Ideas of Computational Thinking (Bers, 2021), Relkin (2018) benefitted from the stages of Vizner's (2017) developmental model for programming with KIBO robotics. Vizner (2017) categorized the developmental levels of programming with KIBO robotics and described four levels namely *proto-programmer, early programmer, programmer,* and *fluent programmer*. According to Vizner (2017), while programming via KIBO, the children dominantly display some behaviors that are the criterion for each level. According to the study, a child's dominant behaviors can serve as a sign of the stage they have reached rather than needing to be seen in every child to be accepted in that stage. While using two theories, Relkin (2018) mapped the stages of Vizner's (2017) developmental model of programming with KIBO robotics onto Bers' (2018) Seven Powerful Ideas. Also, the researcher provided the domains for TACTIC-KIBO's questions by Seven Powerful Ideas and the four phases of computational thinking level by Vizner's developmental model of programming which are the levels of the outcome of the

assessment. While each level of programming was described in Table 3.8., the theory, categories, and developmental levels of programming were provided in Figure 3.9.

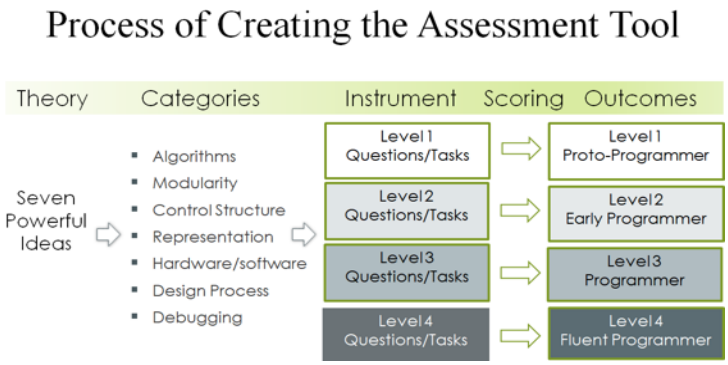

Figure 3.9 The Design Process of the TACTIC-KIBO Assessment Tool (Relkin, 2018, p.22)

This study examined preschoolers at the first, second, and third levels of computational thinking after translating them into Turkish. This is due to the fourth level -fluent programmer- involving "if" blocks, which are known as conditionals and investigated through if-then conditional statements while giving instructions to the children. Actually, fluent programmers are able to write intricate programs without the assistance of an adult and debug software, hardware, or program errors with ease. In addition, fluent programmers can use repeat blocks fluently and understand the conditional tasks. However, it is reported that conditionals that involve if statements can challenge kindergarten children while answering the assessment questions in the fourth level of TACTIC KIBO (Barrouillet & Lucas, 1999; Janveau-Brennan & Markovits, 1999; Relkin, 2018; Rich et al., 2017). These concepts are believed to be more abstract for young children to understand and considered to develop gradually between the ages of 6 to 12 (Barrouillet & Lecas, 1999).

Therefore, the fluent programmer level was not investigated in the current study. Conditional tasks are typically more complicated sequencing patterns and are best taught after children have mastered basic programming tasks. Consequently, the current study did not investigate the fluent programmer level. While identifying the

children's computational thinking levels, seven questions that are based on seven powerful ideas are asked for each level through the TACTIC-KIBO. In the original application of the instrument, children who cannot complete at least three tasks through the second level are categorized as Proto-Programmers. Children who pass the first level but cannot answer four or more questions in the second level are classified as Early Programmers. Lastly, children who can answer four or more tasks at the second level but not at the second level are accepted as Programmers (see Table 3.9.).

Furthermore, when children are unable to meet the lowest levels of the three-correct response criterion, the researcher halts the administration. Nonetheless, by requesting tasks at all levels, the researcher was able to maintain administration in this study. It was thought that children classified as proto programmers might also be able to respond to questions at a higher level. This is because Relkin's (2018) study found that a large number of children who performed incorrectly on level 1 tasks involving algorithms and modularity were able to complete tasks involving these concepts at higher levels. The same study also showed that the children categorized in a lower level, displayed higher level behaviors when they are provided more time and tasks through structured play sessions.

This means that the children who are asked to answer proto-programmer tasks and cannot success necessary number of tasks to jump to the tasks in the next level are limited to show only proto-programmer behaviors and cannot be assessed for their higher-level behaviors. Indeed, the computational behaviors of the children may vary, and not every kid in a given programming level has to exhibit the dominant behaviors listed for each level (Vizner, 2017).

Therefore, administering each level may ease to describe the children's programming related computational behaviors in a broad perspective and identify the children's programming-related computational learning by elucidating the extent of the Coding as Literacy curriculum to contribute all levels of programming.

Table 3.8 The Developmental Levels of Programming by KIBO Robotics and Children's Observed Behaviors for Each Level (Vizner, 2017, p. 52-53; Relkin, 2018, p.21)

The Developmental Level of Programming	**Behaviors**
Proto-Programmer	Proto-programmer children have gross and fine motor skills that enable them to stack KIBO blocks side by side or on top of each other by establishing a whole-piece relationship. They can make predictions about the information on the blocks and create programs without knowing it syntactically. They are aware of the structures found in the body of the KIBO and can assemble the parts into the body. However, they do not know that blocks are the parts that drive KIBO; that is, they cannot establish a relationship between hardware and software. They can discover that the Begin and End blocks come to the beginning and the end of series of blocks by establishing a part-whole relationship. The most important difference of this stage is that they do not know the functions of the symbols or barcodes on the blocks. They also do not yet have the skills to scan and create a purposeful program.
Early Programmer	The children relate hardware and software and know that blocks are the parts that control KIBO. They know why the BEGIN and END blocks are at the beginning and at the end. They can write programs that attempt to move only KIBO, consisting of at least three blocks. However, there is no problem solving or self-expression purpose in the programs they create. They log in to the scan and can learn the full scan with adult support. They can also add repeat and end repeat blocks to their program by establishing a part-whole relationship. They can create the program called THE WORLD'S BIGGEST PROGRAM by using all the blocks they have or using as many blocks as they want. Some children can create programs by scanning blocks one by one and setting them aside (PROGRAM ON THE FLYING).
Programmer	The programmer children create purposeful programs. Purposeful programs may be based on expressing themselves or performing tasks assigned to them. They debug by trial and error or by rewriting the program. They can write programs with repeat and end repeat blocks with adult support. They can enrich it with number parameters when adequate support is provided. They can experience the design process.
Fluent Programmer	The fluent programmers can write complex programs with more than six blocks. They can write programs in which six or more instructions are given and repeat blocks are added. The fluent programmers can write more complex programs which include conditional statements and repetitive actions. They can also debug fluently.

Table 3.9 Original Scoring System of TACTIC-KIBO implemented by Relkin (2018, p. 35)

A number of Satisfactory Responses	Outcome Category
Up to level 2>2	Proto programmer
Level 2>2 and level 3<3	Early programmer
Level 3>2 and level 4<3	Programmer
Level 4>2	Fluent programmer

3.3.3. The Coding as Literacy-KIBO (CAL-KIBO) Kindergarten Curriculum

The Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum was developed by the DevTech Research group (2018), involves 24 activity plans with integrated activities, and is designed for a total of 18 hours. The curriculum is based on Papert's (1980) Constructionism theory. Similar to Piaget's ideas about how children learn, this learning theory focuses on child-centered activities and self-construction of knowledge, but also on active exploration of digital manipulatives and tangible activities that are primarily accomplished with technological tools (Papert, 1993). Papert (1993) emphasized the importance of computationally rich learning environments for learning by doing, which explains why (Bers, et al., 2023). Hence, the CAL-KIBO kindergarten curriculum paid attention to learning by doing through tangible materials that is KIBO educational robotics which enabled children's knowledge construction through teamwork and project-based activities.

The CAL-KIBO kindergarten curriculum intends to enable children to learn computer science and develop their problem-solving and computational thinking skills. It involves powerful ideas of computer science which are integrated with literacy (DevTech, 2018). The powerful ideas embedded in the curriculum are *control structures, representation, hardware/software, algorithms, modularity, debugging,* and *design process.* In addition, the powerful ideas from literacy are *sequencing, the writing process, the alphabet, letter-sound correspondence, editing and audience awareness, literary devices, phonological awareness,* and *tools of communication and language.* The CAL-KIBO kindergarten curriculum (Bers, 2021;

Bers et al., 2023; DevTech, 2018) examined the relationship that was transferred to the curriculum between the powerful ideas from computer science and the powerful ideas from literacy. The findings are summarized in Table 3.10.

As mentioned above, the CAL-KIBO kindergarten curriculum intertwines literacy and coding. Therefore, it is implemented by integrating two storybooks, *Enormous Turnip,* and *Hidden Figures: The True Story of Four Black Women and the Space Race.* The *Enormous Turnip* book tells of a grandfather who plants a turnip, and it grows so big that he cannot pull it up himself. Thus, the grandmother, granddaughter, and pets help him pull the turnip up and there is a specific order to the events in the storybook. The second storybook, *Hidden Figures: The True Story of Four Black Women* focuses on the history of African-American women who worked as mathematicians at NASA. The book also focuses on the discrimination they faced at NASA as women and as African Americans. Before using the storybooks, the researcher examined the appropriateness of the books in terms of Turkish culture and children's development.

Examining the books in terms of Turkish culture and children's development, while The Enormous Turnip was found to be appropriate, the story in Hidden Figures: The True Story of Four Black Women and the Space Race was found to be too long and could be shortened by children's attention to listening. Moreover, as mentioned above, it focuses mostly on American African cultural issues that do not fit into Turkish culture. Therefore, instead of Hidden Figures: The True Story of Four Black Women and the Space Race, the researcher used the storybook *The Little Acorn*, written by Melanie Joyce and translated into Turkish by Anıl Ceren Altunkanat. This book is about the life cycle of an acorn and includes a specific sequence of events that children can easily follow.

In addition, another reason to implement the CAL-KIBO kindergarten curriculum is that it is evidence and research-based in terms of the learning trajectories or paths that would be followed to develop preschool children's computational thinking skills with KIBO educational robotics. Indeed, the curriculum was created using the

Curriculum Research Framework (CRF) (Clements, 2007), a 10-phase process that is recommended by Clements (2007). It is stated that a curriculum built using this framework is legitimately grounded in empirical data. The 10 phases of CRF (Clements, 2007) and how those phases were investigated in the CAL-KIBO design process (Bers et al., 2023) are summarized as follows:

Phase 1: Subject Matter a Priori Foundation: At this stage, content and important concepts were identified that would contribute to the development of computational thinking in the target group of preschool children.

Phase 2: General a Priori Foundation: Teachers' and children's experiences with the curriculum were explained with empirical results and educational philosophies. The ways of knowing computer science concepts and learning through design and programming were explored.

Phase 3: Pedagogical a Priori Foundation: The third phase involves designing pedagogically effective activities that are specific to teaching a particular skill or concept (Clements, 2007). In this sense, this phase explored coding as a literacy approach with literacy activities that included coding and programming with KIBO robotics.

Phase 4: Structure according to Specific Learning Models: This stage requires structuring activities in terms of domain-specific models of learning (Clements, 2007). In this context, the playfulness of the activities, project-based teamwork, coding, and unplugged experiences with KIBO robotics were considered.

The remaining phases are *Phase 5: Market Research, Phase 6: Formative Research*, *Phase 7: Formative Research: Single Classroom*, *Phase 8: Formative Research: Multiple Classrooms*, *Phase 9: Summative Research: Small Scale* and *Phase 10: Summative Research: Large Scale.* During these stages, the curriculum was the subject of formative (pilot testing) and summative research, which involved randomized controlled trials. Thus, early childhood educators can utilize realistic

scenarios that the researchers identified when implementing CAL-KIBO curriculum activities (Bers et al., 2023).

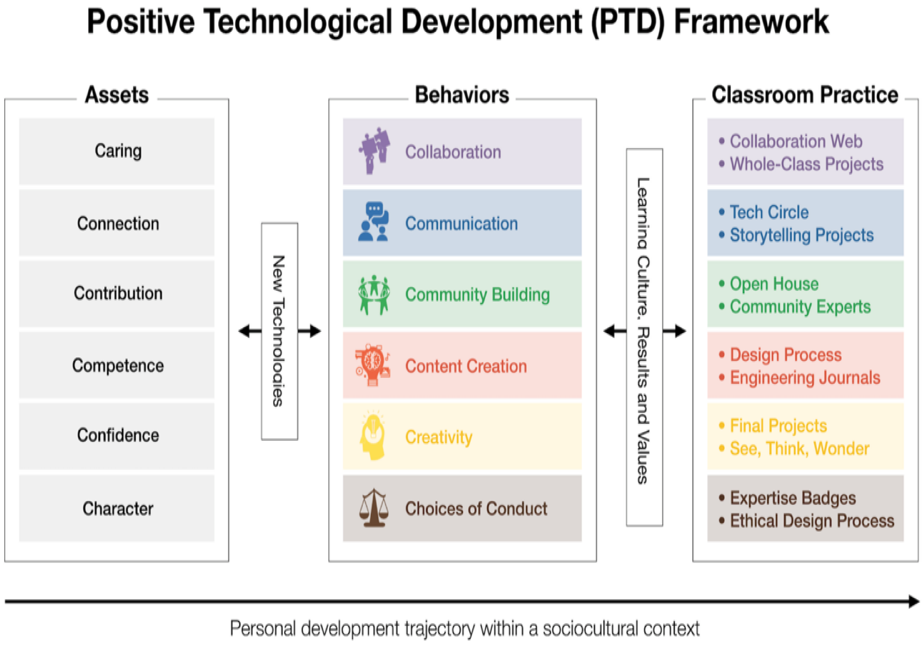

Figure 3.10 The Developmental Assets and Positive Behaviors Embedded in the Coding as Literacy-KIBO Kindergarten Curriculum Activities (Bers, 2021, p.131)

Lastly, the Coding as Literacy-KIBO kindergarten curriculum is grounded by the 6C's of Positive Technological Development framework (Bers, 2021) including *content creation, creativity, collaboration, communication, community building, and choices of conduct*. Positive Technological Development (PTD) is an extension of Papert's Constructionism theory and involves psychosocial, civic, and ethical components (Bers, Blake-West, Kapoor, et al., 2023). Bers (2021, p.131) reported that while *content creation, creativity,* and *choices of conduct* contribute to the intrapersonal domain, *collaboration, communication,* and *community building* are accepted to improve the interpersonal domain. Additionally, the researcher found that these actions are linked to the positive youth development (PYD) personal assets of competence, confidence, caring, connection, and character. (Lerner et al., 2005) (see Figure 3.10). Bers (2021) concluded that when used together, 12Cs can provide insight into how technology can contribute to developmental assets while enhancing

positive behaviors. In addition, 12Cs may inform about how developing technologically rich learning environments can help children develop positive behaviors at the same time. However, Bers (2021) noted that the classroom learning environment, culture, social values, and rituals can make a difference in the activities to be implemented.

3.3.3.1. The Structure of Coding as Literacy-KIBO (CAL-KIBO) Kindergarten Curriculum Activities and KIBO Robotics Platform

The Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum involves 24 activity plans each of which lasts 45 minutes and is completed in 18 hours. Activities are designed to follow a project-based approach. Therefore, each activity schedule begins the day with a warm-up activity to introduce the day's concept or reinforce a previously learned concept. Then, the Opening Tech Circle introduces the topic to be worked on throughout the day, and the Closing Tech Circle concludes the activity with conversation, discussion, and sharing of the information learned.

Activities like Word Time, KIBO Time, and Unplugged Time are typically practiced in between these Tech Circles. Children learn new things and have new experiences related to KIBO during KIBO Time, and they engage in activities where they act and interact with their peers during Unplugged Time. They also learn important knowledge and skills about KIBO and coding through play without using KIBO robots. For instance, each child assumes the role of the KIBO robot and attempts to act out the ways that their friends or peers program them while learning how to code and program using the visuals of the blocks, they receive in the form of colored printouts to teach the KIBO programming language. Moreover, coding and literacy activities are combined during Word Time. In this process, children work individually or in small groups on design journals as needed for each activity, using writing skills, drawing the desired instructions given by the teacher, and expressing themselves. Children in this process collaborate with their group members to plan, design, edit, fix mistakes, and complete the programs they are asked to create by working on the design journals. In this sense, both the potent concepts derived from literacy and computer science that support computational thinking are employed.

Table 3.10 The Link Between the Powerful Ideas from Computer Science and The Powerful Ideas from Literacy

The Powerful Ideas from Computer Science	**The Powerful Ideas from Literacy**	**The Connection between the Powerful Ideas**
Algorithms	Sequencing	Just as in algorithms, the sequence and order of events is important when writing a text. Both writers and programmers provide all the necessary information in an appropriate context, step by step, so that the product they create is meaningful. The skills that come from literacy are therefore thought to be linked to algorithms in computer science.
Design Process	Writing Process	Just as there are processes involved in writing a text, there are also processes involved in writing a program. While the writing process involves planning, producing, revising, and sharing the product with others, the design process followed by programmers is a cyclical and iterative process as well as a creative one. In both processes - the writing and design process - one can go back at any time and make edits and iterations until one reaches the desired point.
Representation	Alphabet and Letter-Sound Correspondence	Just as letters and punctuation marks are used in writing, there are symbols and numbers, such as 0 and 1, that represent certain functions or instructions used in programming in the form of shapes, colors, or sounds.
Debugging	Editing and Audience Awareness	Like writing, programming requires systematic checking, testing, and evaluating to ensure that the product you are working on is free of errors. The result is a useful, understandable, and bug-free product that, when shared with others, achieves the desired result.
Control Structures	Literary Devices	In writing, writers use many techniques to enrich their products, such as metaphor, simile, or flashback; in programming, programmers use techniques involving repetition, patterns, conditionals, or events. This means that both control structures and literary devices can enrich one's creation, making it more complex, but also more original. Like mastery of writing, mastery of programming is related to the use of these techniques.
Modularity	Phonological Awareness	Just as we use the letters, syllables, and sounds of the words in a text before reading or writing it, there are processes that ensure the desired goal is achieved when writing code and programming. When these processes are defined by breaking down a large and complex task into smaller parts, or sub-steps, the goal can be achieved more easily and successfully.
Hardware and Software	Tools of Communication and Language	Just as written and verbal communication allow individuals to express themselves and communicate, the environments that allow thoughts to be expressed while coding and programming include hardware and software.

Table 3.11 provides sample daily activities and objectives of the CAL-KIBO kindergarten curriculum (DevTech, 2018, p.11, 12, 13). In addition, the CAL-KIBO kindergarten curriculum is implemented through the KIBO educational robotics platform developed by the DevTech research group. It is a research-based, screen-free, hands-on robotics platform designed for early childhood computer science and engineering education.

Table 3.11 Sample Activities and Objectives of the CAL-KIBO Kindergarten Curriculum

Activities	**Objectives**
1. Introduction to KIBO	Children will be able to... ● Define a robot. ● Categorize items as “robots” or “not robots”.
5. A New Language	Children will be able to… ● Scan KIBO blocks. ● Identify and use the KIBO Begin and End Blocks. ● Identify and use KIBO Motion Blocks. ● Identify the role of the programmer in programming a robot.
9. Fly Me to the Moon	Children will be able to… ● Write a goal-directed KIBO program. ● Write a KIBO program consisting of the Begin block, the End block, and Motion Blocks.
14. Program the Hokey Pokey!	Children will be able to… ● Use the Lightbulb and Light Block with KIBO. ● Write a KIBO program to accompany a dance
17. Can You Repeat That?	Children will be able to… ● Identify an AAAA pattern in a KIBO program. ● Use the Repeat and End Repeat Blocks in KIBO to create a program with an AAAA repeating pattern. ● Recognize one program output can be represented using multiple approaches.
20. Your Final Project I	Children will be able to… ● Recall and identify the beginning, middle, and end scenes of *The Enormous Turnip.* ● Begin programming their final projects.

Moreover, KIBO has hardware (KIBO’s body, motors, wheels, modules, and sensors) and software (the programming language through programming blocks).

There are 21 programming blocks that have specific symbols and colors on the blocks. To program KIBO, the children arrange the programming blocks after creating their algorithms, and then scan the barcodes on the programming blocks with KIBO's scanner. One of the most useful aspects of KIBO robotics is its ease of integration into math, literacy, music, drama, and art activities (Bers, 2018). KIBO has detachable and pluggable parts that allow young children and early childhood educators to use it for many purposes in many types of activities (Relkin, 2018; Vizner, 2017). Children can see the program they create by sequencing wooden blocks when they program with KIBO. They can also manipulate the blocks to achieve the plans they have in mind, learn how to program, and cultivate their computational thinking skills while doing so.

3.3.4. Population and Sample in the Main Study

The target population of this study involves preschool children attending state kindergartens in Aksaray city center at Türkiye. However, since it is not possible to reach all preschool children attending to the state kindergartens in Aksaray city center and generalize the findings in many preschools with different characteristics, the target population was narrowed and settled to preschool children attending to the one of the state kindergartens located in Aksaray city center. While deciding on a sampling procedure, a convenience sampling method was preferred which is useful when there is a group of persons near and available to the researcher (Fraenkel et al., 2011). For this purpose, four classrooms including two experimental groups (n = 38) and two control groups (n = 33) were chosen from the target population. While selecting the groups, both experimental and control group children's coding experience with robotics and typical development were taken into consideration to be involved in the study. Indeed, the classrooms with typically developed children and the children with no robotics-based coding and programming experience were chosen for the study as the children who have coding experience with robotics and non-typical development were not aimed to be investigated in this study. Other issues that were significant for this study was the child number in each classroom and the children's ages in months. Indeed, there were three KIBO robotics kit to be

used in the intervention and the researcher wanted to study with the classrooms with small numbers to be able to implement small group activities with a maximum of 5-6 children per kit.

3.3.5. Data Collection Procedure in the Main Study

Before conducting the main study, the approval of the research ethics committee of Middle East Technical University (METU) and the necessary permission of the Provincial Directorate of National Education in Aksaray were obtained. Parental consents were also obtained prior to data collection and intervention. Following that the pretest data collection process began with the TechCheck-K assessment to examine the experimental and control group children's existing computational thinking skills in Kindergarten X during the spring semester of the 2022-2023 school year. Before the pretest administration, the play area was prepared in participant children's classrooms by putting the KIBO robotics kit into the play center. Then, each child was introduced to the play area one by one and asked if there are any different materials that they see in the classroom. After taking their answers, each child was informed about the assessment and intervention to take their verbal assent. The children who accepted to participate in the study were involved in the data collection and intervention procedure. The detailed timeline for the main data collection procedure was provided in Table 3.12 as below.

Table 3.12 The Timeline of Data Collection Procedure in Phase 2

Pretest O_1	*Time 1*	*Time 2*	*Posttest* O_2	O_3	O_4
The TechCheck-K Administration	The intervention of four preparatory activities	The intervention of The Coding as Literacy-KIBO Kindergarten Curriculum	The TechCheck-K Administration	Demographic Information Form Administration	The TACTIC-KIBO Administration

Note. 1. While *Time 1 and 2* represent the intervention times, O_1, O_2, O_3, *and* O_4 represent the main data collection times.
Note. 2. While O_1, O_2, *and* O_3 were administered with both the experimental and control groups, O_4 was executed only with the experimental group.

In order to prevent classroom management issues and children's tendency to look others' assessment papers, the researcher arranged the drama room by leaving enough space between children's desks. The data were collected in small groups of up to five children in each administration. To administer the TechCheck-K, the researcher read each question twice and gave the children up to one minute to circle the correct answer on their own assessment papers through pencils. The pretest administration for both control and experimental children took approximately 16 minutes. After the pretest, four preparatory activities were implemented with both experimental groups in their own classrooms over two days to reinforce the children's knowledge of directional language and colors prior to implementation. Then, the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum was implemented between February 2023 and April 2023. The detailed information about the CAL-KIBO kindergarten curriculum intervention process is provided in the next section. Following that, the post-test data were collected from both the control and experimental group children via the TechCheck-K assessment tool with the same conditions in the pretest administration. While the posttest administration of control group children lasted in 13.20 minutes, the experimental group children lasted in 10.4 minutes. Then, the demographic information form was sent to the mother and father of each child in both the control and experimental groups after the intervention was completed. As mentioned above, by sending the demographic information form after the intervention, the researcher aimed to prevent external factors that may occur after the parents read the demographic information form. For instance, parents may engage their children in coding applications during the intervention after reading questions related to coding which could likely lead to invalid results influenced by external factors.

Lastly, the TACTIC-KIBO instrument was administered to the experimental group of children to detect their computational thinking skills through their programming levels. The TACTIC-KIBO was administered as one-by-one, and all the sessions were audio-recorded. The mean time for TACTIC-KIBO was 15 minutes. Because the researcher of this study administered and scored all TACTIC-KIBO sessions after the CAL-KIBO intervention, it was essential to test for rater bias and report inter-rater reliability issues by determining Fleiss' Kappa between the researcher and

other examiners. Fleiss Kappa gives information about the measure of the agreement between more than two raters. For this purpose, two outside raters were recruited. Raters have expertise in information and communication technologies, mathematics education and assessment and evaluation fields and carry out research on these fields. In order to identify Fleiss' Kappa between the researcher and other examiners, the raters were provided a scoring sheet that was also used by the researcher during the TACTIC-KIBO administration. Then, the raters listened to audio recordings of the TACTIC-KIBO administration conducted with eight children.

To calculate the interrater reliability through the Fleiss' Kappa, "Fleiss' Kappa Calculator" was used on "DATAtab" which is an online calculator for statistics. According to Landis and Koch (1977), while kappa value larger than 0.8 show almost perfect level of agreement, the values larger than 0.6 reveal substantial level of agreement, and the value larger than 0.4 means moderate level of agreement. Lastly, the Kappa value which is larger than 0.2 reveals fair agreement level among raters. Therefore, an interrater reliability analysis between the dependent samples of primary rater, rater 2 and rater 3 showed that the Fleiss' Kappa value to be 0.94 which means that there was an almost perfect level of agreement between three TACTIC-KIBO raters.

3.3.5.1. The Coding as Literacy-KIBO Kindergarten Curriculum Intervention Process

The Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum was implemented three days a week in two experimental groups. The period of implementation was from February 2023 to April 2023. Teachers observed the researcher during implementation to ensure classroom management and provided assistance to the researcher as needed. One of the experimental groups (n = 21) was more crowded than the other (n = 17) and it was easier to practice in the class with fewer students. This was because the researcher had three KIBO robot kits, and when the KIBO experiences were conducted in small groups, the turn of the KIBO experience came more quickly for each child in each group.

In addition, both experimental groups participated in this study on the same day in their classrooms. The researcher was able to implement each activity plan in approximately one hour, although the CAL-KIBO curriculum calls for 45 minutes. In the more crowded class, the interventions could take 15 minutes longer than in the other class. In fact, while implementing the activities, each child had to take turns expressing their opinions one by one, especially during the technology circles, and making presentations with their groups while sharing their designs. This resulted in a slightly longer duration in a crowded experimental group.

Figure 3.11 A Design Journal Activity for the Enormous Turnip Story Project

After that, children worked in small groups on three coding and programming projects from the CAL-KIBO kindergarten curriculum. The children were instructed to write programs in the curriculum's design journal for two story-based projects and

one hokey-pokey dance project, and then program the KIBO robot in accordance with the programs they wrote in the design journal. Example from design journal activity was provided in Figure 3.11. These activities also involved children programming KIBOs and decorating their KIBOs with art materials in a contextually appropriate way (Activities 10, 15, and 21) (see Figure 3.12).

Because the projects are conducted in small groups, the researcher determined the small groups to ensure cohesion and to ensure that different children in each project work together in small groups. The number of boys and girls in each group was also considered.

During the first story-based project activity -the Little Acorn project-, the researcher distributed the design journal page to each group. However, the researcher observed that the children did not experience a productive program writing process which could be due to their first time to write programs in small groups. Therefore, the researcher distributed the design journal page to each child and asked them to draw individual programs on the design journal and then come together in small groups to write a common program.

Moreover, the researcher wanted the children to use colored pencils while writing their programs. This allowed the researcher to observe whether the children appropriately used the BEGIN and END blocks in the program and whether they indicated the right color of these blocks in their programs. The researcher also walked among the children as they wrote their programs in their design journals, asked them to name the directional blocks they had drawn which enabled to detect possible gaps in the children's learning.

Furthermore, aside from the directional blocks (turn right, turn left, forward, and backward), some children wanted to look at the symbols on the wait for clap, beep, spin, and shake blocks so they could use them in their program even though they couldn't recall the symbols on those blocks. Hence, the researcher gave the children the programming blocks they had inquired about.

Figure 3.12 Children's KIBO Decorations for the Hokey Pokey Dance

For the second project, Hokey Pokey Dance, the researcher created new small groups that included children who had not worked together on the first project (the Little Acorn project). The researcher asked the children to individually draw the programming blocks in their design journals, since they had become familiar with drawing the images of the blocks in the first project. After they completed their programs, the researcher also distributed the visuals of the programming blocks to the children, and they matched the visuals with their drawings (see Figure 3.13). In the last project, Enormous Turnip, the researcher, formed the last small groups that worked together until the end of the intervention. During the last project, the children worked in small groups without asking to look at any block visuals and programmed KIBO. One of the most useful materials for the researcher in the projects was the *Job Cards*. These cards are one of the CAL-KIBO kindergarten curriculum materials and called as Assistant, Organizer, and Scanner which allowed for the division of labor in small groups as well as for children to wait their turn until their roles changed.

Finally, during the intervention, there were formative and summative activities which enabled the researcher to follow preschool children's knowledge and understanding of KIBO robotics and programming skills. These activities are presented in the CAL-KIBO kindergarten curriculum as assessment tools that may help to detect the

learning gaps in children and to modify or add activities to support children's computational learning (DevTech, 2018).

Figure 3.13 Children's Programming Project for Hokey Pokey Dance on Design Journal and Programming Blocks

These questions are administered as a whole class activity, and the children answered the questions by raising their hands. The children are then involved in the evaluation of the responses. After the questions are posed, the children's learning gaps are thus found and filled. The formative activities are named as *Check for Understanding* and summative activities are named as *Show What You Know*. These activities are conducted at specific times during the intervention. For instance, Check for Understanding is integrated throughout the 6th, 13th, 16th, and 18th activities and inform researchers and early childhood educators about the children's knowledge about the programming blocks. Each Check for Understanding activity involves two questions about the programming blocks. Show What You Know is, however, implemented at the beginning of 22nd activity and evaluates children's programming learning and skills gathered throughout the curriculum.

3.4. Treatment Fidelity of the Study

Treatment fidelity, which is defined as the identification of the strategies to ensure the accuracy and consistency of an intervention that is delivered to participants in a comparable manner, is crucial, particularly during the planning phase of the intervention (Smith et al., 2007). This means that the researchers or educators who wish to implement the same or similar intervention should be able to do so as they have done in previous interventions conducted by other researchers. It is reported that higher levels of treatment fidelity possibly result in better treatment outcomes (Durlak & DuPre, 2008). Treatment fidelity may include participant reporting, preparation of a treatment manual, videotaping the interventions, and regular supervision of practitioners (MTA Cooperative Group, 1999; Smith et al, 2007). Detrich (1999), for example, stated that the research undertaking replication studies would take into account the characteristics of the participants, the intervention conditions—such as the classroom environment, including culture and daily routines—and the resources needed. This is thought to guarantee faithfulness. By providing additional information about the intervention procedure, fidelity is also provided for future researchers who plan to implement the same study on different occasions with different participants. For example, the researcher of this study explained the intervention procedure in the previous section in detail so that the problems or issues encountered in this study can be considered by other researchers. Five key areas of treatment fidelity increase the internal validity of the study, which are *study design, training, treatment delivery, treatment receipt,* and *treatment enactment* that is implementation (Smith et al, 2007). What each fidelity key means and how it is addressed in this paper are as follows.

3.4.1. Study Design

Establishing protocols that minimize implementation setbacks and are in line with pertinent theory, pedagogy, and practice is a crucial aspect of study design. The researchers may take into account the school calendar in terms of field trips, holidays, or assemblies that have an impact on the length of the intervention in order

to ensure this issue. As a result, the study's researcher carried out the activities three days a week while considering the physiological requirements of the children, such as their need for water, food, and the bathroom, which could cause disruptions. The study design also accounts for teacher and child attrition, and the researchers may plan to train practitioners enabling them to reach larger samples. However, due to lack of the equipment -three sets of KIBO robotics kit- was not enough to conduct many interventions at the same time, the researcher of this study conducted the interventions, and given the risk of data loss the study was carried out with two control and two experimental groups.

3.4.2. Training

Training is related to the appraisal of treatment providers who can provide the treatment at an acceptable level of quality and efficiency (Smith et al, 2007). The researcher in this study has completed her high school education in the field of child development in a vocational high school, and in addition to having many internship and teaching opportunities during her high school education, she has also studied in the Department of Elementary and Early Childhood Education in her university education and has done practices in many kindergartens, both as required by the applications of the field courses and as required by the teaching practice courses. In addition, she has had the opportunity to observe and evaluate many preschool classrooms. She can therefore perform the practice of research at a level appropriate for her. Treatment providers receive the training in a systematic manner as well (Hennessey & Rumrill, 2003). In order to achieve this, the study's researcher viewed earlier KIBO robotics intervention videos as well as tutorials that broke down the components of the system.

In addition, the researcher of this study was in contact with the researchers who implemented the Coding as Literacy-KIBO curriculum in their studies and those researchers shared their experiences such as the number of KIBO robotics kits and children which have a big impact on duration and quality of the intervention. In this

way, the researcher was able to anticipate the problems she might encounter during the implementation and take precautions.

3.4.3. Treatment Delivery

Treatment delivery can be ensured through monitoring the intervention and reporting the methods followed for this purpose. This step increases the internal validity of the research and improves the delivery of treatment (Smith et al, 2007). To illustrate, the practitioners can be monitored by other researchers, teachers, or experts in a related field and ensured that the treatment was delivered in an intended manner (Lichstein et al., 1994). The strategies for treatment delivery include the use of a treatment manual, structured training, supervisory monitoring and feedback, and delivery and accuracy checklists (Burgio et al., 2001). Considering these issues, only the researcher of this study conducted the activities and prepared a guide to implement the CAL-KIBO kindergarten curriculum in both experimental groups with the same procedures provided.

The entire guide is based on the information provided in the CAL-KIBO kindergarten curriculum, considering the goals and objectives, materials, and teacher preparation information. Therefore, the number of materials needed for each child, the names and order of the activities, and the administration times to be given at the required times were specified for the current study. In addition, the researcher added a checklist section to the guide to verify that the activities were carried out as planned.

The guide is also intended to describe specific behaviors that should take place during the interventions including the goals to be addressed in the activities. For this purpose, the researcher video-recorded some of the practices in both classes as much as possible to go back and check through the guide after each activity was completed to see whether there were any missing aspects of the intervention. By doing this, the researcher could fill the knowledge that were missed at previous activities. In addition, to ensure that the practices were the same in both classes, the researcher

paid attention to the points she taught in the first class, the concepts, and words she mentioned, and the instructions she gave. For this purpose, the researcher took notes of the points to pay attention to, the instructions, and the techniques she applied throughout the implementation.

3.4.4. Treatment Receipt

Treatment receipt can be considered by researchers conducting experimental research to ensure that the participants in the intervention have acquired the information provided during the implementation (Bellg et al., 2004). For this purpose, the researchers can collect data in many ways to monitor the learning goals and skills taught to them during treatment (Bellg et al., 2004; Smith et al., 2007). In this context, the researcher administered pre- and post-test skill measures (using the TechCheck-K assessment tool) to assess children's computational thinking development.

Also, the preschool children's and their parents' demographic and background factors were gathered through Demographic Information Form and this data allowed researcher to have broad information about the children and to explore the effect of the background factors on the preschool children's computational thinking skills after implementing Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum implementation. Indeed, for this purpose, the experimental and control group children's adjusted pretest means were used as covariate variable which eased to match the groups, and background factors were then checked on the posttest scores which also allowed the researcher to control the effect of implementation by considering children's background factors.

Finally, there was TACTIC-KIBO measurement which explained children's computational thinking skills by interrogating their programming skills through KIBO robotics. This measurement allowed us to look at children's computational thinking skills from a different perspective and to see which dimensions -powerful ideas- of computational thinking skills improved when combined with programming.

3.4.5. Treatment enactment

Treatment enactment is the last step in monitoring treatment fidelity. To achieve this, the participants may be evaluated by the researchers on behaviors or skills related to treatment in pertinent real-life contexts (Borrelli et al., 2005). The focus is on whether the skills taught in the intervention are being employed at appropriate times and situations (Bellg et al., 2004). In other words, the researchers try to learn whether what is learned (delivered), is used in real life (enactment). The enactment can also be treated as a follow-up session to control if the participants maintain and improve skills taught in the intervention (Smith et al., 2007). However, the treatment enactment itself requires the implementation of treatment fidelity steps which means increased staff or staff time, travel time, additional data analysis, and funding (Smith et al., 2007).

Therefore, the enactment could not be ensured in this study. To find out if the children in the experimental groups can still use the intervention skills outside of the implementation context, this study could be expanded to include Structured Play Sessions. By using rubrics that include and measure the target skills and concepts related to computational thinking skills, the children's treatment enactment would be ensured in the next step of this study.

3.5. Data Analysis in the Main Study

It is crucial to remember that this study was initially intended to include two control groups and two experimental groups before moving on to the section on data analysis. However, during one of the researcher's meetings with the teachers in the control group, the researcher was informed that one of the teachers had begun implementing an unplugged coding kit in her classroom, including coding worksheets and a play rug that had previously come as a gift with the educational kits she had received before. This may be due to the fact that the teacher knew that educational robotic coding activities would be implemented in other classrooms, and she wanted to implement coding activities in her classroom. In fact, one of the

control groups was systematically exposed to unplugged computer science activities that could contribute to computational thinking skills, which would prevent the study from providing valid and reliable results if they were included in the study. Consequently, the researcher excluded one of the 13-children control groups from the investigation, and Phase 2 analyses were carried out using 38 children in the experimental group and 20 children in the control group.

The data were entered into SPSS 23 package. Before reporting the results of the research questions, first the demographic information of the parents and the background information of the preschool children in the control and experimental groups were presented. The mean, median, and standard deviation for the continuous variable—the preschoolers' computational thinking abilities—were then supplied by descriptive statistics for the purpose of answering the second main research question and examining the pretest results. The following analysis included Independent Samples T-Test to answer the research question; "Is there a significant difference between the experimental group and the control group of preschool children in terms of their computational thinking skills before the educational robotics-based coding implementation?". Then, to answer the other research question which is "What are the general patterns of experimental group and control group preschool children's posttest computational thinking scores?", descriptive statistics was conducted to investigate the mean changes in the posttest item and mean scores of preschool children in the experimental and control group. Additionally, post-hoc item analysis was done for the same research question in order to look at changes in the percentage of correct answers in the computational thinking domains. Subsequently, One-Way Analysis of Covariance (ANCOVA) was used to explain the answer for the following research question; "Does educational robotics-based coding implementation produce greater computational thinking scores in the experimental group children than in the control group children after adjusting for pretest differences in computational thinking scores?". Before conducting Independent Samples T-Test and One-Way Analysis of Covariance (ANCOVA), required assumptions were checked and ensured. Following that, for the third main research question, the preschool children in the experimental group were examined in terms

of their programming-related computational thinking skills. The research question of "What are the programming related computational thinking levels of preschool children in the experimental group after the educational robotics-based coding implementation (via the TACTIC-KIBO)?" was explored through the frequencies of behaviors observed in each level of computational thinking.

Lastly, the experimental and control group children's computational thinking abilities were determined based on their individual background factors. Initially, the parents' demographic data and the children's background variables in the experimental and control groups were described using descriptive statistics. After adjusting for pretest scores, a Three-Way ANCOVA was performed to examine the impact of background factors on the post-test computational thinking scores of children in the experimental and control group.

3.6. Threats to the internal validity of the study

Internal validity is being clear about what an observed relationship between two or more variables means, and not that it is due to "something else" (Fraenkel et al., 2011). In other words, internal validity means that the results of a study are due to the independent variable and not due to an external variable or threat. In Phase 1, there were threats of mortality, location, and instrumentation decay. A mortality threat occurs when subjects are lost. However, because Phase 1 was not a longitudinal study, the mortality threat did not affect the results of this study. A location threat occurs when the data are collected in different locations. However, in the data collection, the researcher used empty classrooms that were set up in children's kindergarten and was similar to their classrooms. Instrument decay happens particularly in interview survey studies where participants become fatigued during the administration. However, during the administration most children willingly participated in the administration and stated that the activity was so enjoyable for them.

In addition, the administration time was not too long, so the children were not

rushed. In the first step of the main study, the design of the study was a quasi-experimental pretest posttest control group design, and the TechCheck-K instrument was used to measure computational thinking skills of the preschool children during the pretest and posttest administration. The threats to internal validity were location, data collector characteristics, data collector bias, attitude of subjects, and implementation. First, the two experimental groups that took part in this study had different locations for implementing the treatment. Nonetheless, throughout the implementation, the classroom settings for the two experimental groups were essentially the same. Additionally, since the researcher obtained the pretest and posttest results from the same classroom, the data collection process was identical for both groups. Second, because the study's researcher carried out the treatment and curriculum implementation, the data collector's characteristics were the same for the experimental and control groups.

Third, the researcher made an effort to guarantee that the procedures in the two experimental groups were comparable to one another in order to prevent data collector bias. For this purpose, the researcher took notes on the important points and instructions she paid attention to in the first experimental group and used a guide to follow the same procedures. Finally, the attitudes of subjects and implementation including the characteristics of the treatment providers especially affect the post-test results. To avoid these threats, the data collector and treatment provider of the study were the same throughout the study and the children were not treated as receiving special attention compared to the control group.

The second step of Phase 2, the main study, included the one-shot case study in which programming related computational thinking skills of the experimental group children were measured with the TACTIC-KIBO. The control group was not measured with TACTIC-KIBO assessment tool since they did not have a coding and programming experience with KIBO educational robotics or any other educational robotics that may have an impact on their programming related computational thinking skills. In fact, the purpose of the second phase of the main study was to reinforce the findings from the first step by providing more information about the computational thinking skills related to programming of the experimental group of

children who were taught the CAL-KIBO kindergarten curriculum.

In addition, the intention was not to compare the programming related computational thinking skills of the experimental group children with the control group children. Therefore, while conducting the first step of main study, location, data collector characteristics, data collector bias, attitude of subjects, and implementation threats were already controlled, and this enabled the researcher to control the same threats for the second step. It was also assumed that the results revealed in the second step of main study was due to the experimental group children's experience with CAL-KIBO kindergarten curriculum and KIBO educational robotics. This was also due to the fact that the demographic information form used to gather background data on the children in the experimental and control groups showed that none of the children had any prior experience with educational robotics programming, and the majority of the children in the experimental group had never done any screen-based programming.

3.7. External validity

External validity relates to the extent to which the results of a study are generalizable (Fraenkel et al., 2011). For external validity, the data were found to be sufficient for Phase 1 and Phase 2. Nevertheless, the results of this study could only be generalized to the preschool children going to the public schools with the same conditions in ~~the~~ Aksaray city center. Private school children were not participants in this study and were excluded from the generalization of the results. Indeed, the generalizing is limited to the group from that data are obtained (Fraenkel et al., 2011). In order to make a more plausible generalization, the sample was described in detail in related sections.

CHAPTER 4

RESULTS

This chapter began with presenting background data on parents as well as children gathered from the Demographic Information Form, followed by results derived within the scope of the research questions. Firstly, to address the first research question, the TechCheck-K assessment tool's pretest results were given to determine the baseline computational thinking abilities of the children in the experimental and control groups. For this purpose, the general patterns of preschool children's computational thinking skills were explained for the experimental and control groups separately depending on the general mean scores and the TechCheck-K items with the highest and lowest mean scores. In addition, the children's pre-test mean scores were checked in terms of the powerful ideas of computational thinking. In other words, the domain-specific computational thinking skills were checked for each group. In the pretest results, also, experimental and control group children's existing computational thinking skills were checked to investigate whether there was a significant difference between the groups before the intervention.

Following that, the posttest results were reported in terms of the changes in general mean scores and the TechCheck-K items with the highest and lowest mean scores. Then, the children's post-test mean scores were checked in terms of the powerful ideas of computational thinking and post-hoc item analysis was conducted to explore the changes in domain-specific computational thinking skills. Lastly, the posttest scores were controlled to identify whether the robotics-based coding implementation -Coding as a Literacy implementation- produced higher computational thinking scores in the experimental group than the scores before the treatment and compared with the control group children's computational thinking scores whether there is a

significant difference. The impact of Coding as a Literacy-KIBO kindergarten curriculum was also discussed in the second section of the results with reference to the children in the experimental group's computational thinking abilities related to programming performance. The results were provided by the TACTIC-KIBO administration, and the programming levels of children were categorized as proto-programmer, early-programmer, and programmer.

Finally, in response to the last research question, the contribution of background factors to the children's post-test computational thinking scores were reported through the three-way ANCOVA results.

4.1. Background Information of the Pre-School Children and Demographic Information of the Parents

Phase 2 involved 58 preschool-age children who attended a public kindergarten situated in the city center of Aksaray. There were 38 children in the experimental group and 20 in the control group. There were 28 boys and 30 girls among the participants. Furthermore, the ages of the preschoolers in months varied from 62.67 to 79.27.

Table 4.1 provides specifics for the children in the experimental and control groups.

Table 4.1 Background Information of the Children

Group	***Gender***	***F***	***%***
Experimental	Boys	17	44.7
	Girls	21	55.3
Control	Boys	11	55
	Girls	9	45
Whole Group	Boys	28	48.3
	Girls	30	51.7
Group	***Mean Age in Months***	***Min***	***Max***
Experimental	70.74	62.67	76.5
Control	72.14	64.17	79.27
Whole Group	71.22	62.67	79.27

In order to examine the demographic information of parents and background information of children, the data collected through the Demographic Information Form were used. This form was sent to both parents and was completed by 110 (55 mother and 55 father) out of 116 (58 mothers and 58 fathers) parents. There were three families who did not send the Demographic Information Form. This form was filled out by both the mother and the father of each child, and in case of inconsistencies or omissions in the information, information was obtained by asking to the parents. In this way, data loss was also minimized. The data were analyzed through SPSS 23 package through descriptive statistics and presented under three parts. While the first part investigated the demographic information of the parents such as age levels, education levels, socioeconomic status; the second and third part examined the background information of children which explained the children's screen experience with digital games, and tangible play experience at home.

Table 4.2 Demographic Information of the Parents

Number of Children	***1 child***	***2 children***	***2+***
Experimental	7	18	10
Control	2	11	7
Whole Group	9	29	17
Mother Age	***M***	***Min***	***Max***
Experimental	35,48	25	49
Control	35,65	26	45
Whole Group	35.54	25	49

Father Age	***M***	***Min***	***Max***
Experimental	38,82	29	55
Control	37,75	27	47
Whole Group	38.43	27	55

		Experimental Group	**Control Group**
		F	***F***
Education Level of Mother	Elementary School	4	*1*
	High School	10	6
	2-year associate degree	5	1
	Bachelor's degree	13	11
	Master's degree	3	1
	Whole Group	35	20

Education Level of Father	Elementary School	5	1
	High School	7	6
	2-year associate degree	3	0
	Bachelor's degree	17	11
	Master's degree	3	2
	Whole Group	35	20

To begin with the first part, Table 4.2. was created to demonstrate the demographic information of the parents. According to the results, in the experimental group, the number of families with one child was seven, the number of families with two children was 18, and the number of families with more than two children was 10. In the control group, there were two families with one child, 11 families with two children, and seven families with more than two children. It was concluded that most of the children in this study have one (n = 29) or more siblings (n =17). Moreover, the average age of the mothers in the experimental and control group children was approximately 35 years, while the average age of the fathers was almost 38 years. Finally, examining the educational levels of the families, it was found that both mothers and fathers were mostly high school or college graduates. Indeed, 24 mothers were found to graduate from university and 16 mothers were found to graduate from high school out of 55 mothers. Moreover, 28 fathers graduated from university and 13 fathers were found to graduate from high school out of 55 fathers. In addition, in the study group there were five mothers and six fathers who graduated from primary school and four mothers and five fathers with master's degrees. Detailed information about the demographics of the parents were provided in Table 4.2.

Table 4.3 ScratchJr Familiarity of the Parents

		Experimental Group	**Control Group**
		F	***F***
I know the application and my child frequently plays with it.	Mothers	2	0
	Fathers	2	0
I heard the application, but I do not know what ScratchJr is.	Mother	1	2
	Father	0	0
I know the application, but I did not install it to the tablet.	Mother	2	0
	Father	0	0
I do not know the application.	Mother	30	18
	Father	33	20

Next, the parents were asked for their familiarity with the ScratchJr application. Out of 110 parents including 55 mothers and 55 fathers, 48 mothers and 53 fathers reported that they do not know the ScratchJr application. There two mothers in the experimental group who knows the ScratchJr and installed it to the tablet for their children. There was one mother in the experimental group and two mothers in the control group who heard about the application but do not know what ScratchJr is and did not install to the tablet. Indeed, there were only two children in the experimental group who were reported to spend time with ScratchJr (see Table 4.3.).

Subsequently, Table 4.4 was developed. Initially, the background information of the children was given regarding their screen experience with digital games on smartphones or tablets, including the type and duration of the experience (i.e., racing games, mathematical games, coding games). The results indicated that out of 55 children, most of the children have screen experience with digital games at home (*n* = 50) and only five children do not have screen experience with digital games at home. In addition, out of 50 children who have screen experience with digital games at home, only two children were found to have an experience with ScratchJr application.

Table 4.4 Screen Experience of the Children at Home

		Experimental Group	**Control Group**
		F	***F***
Screen Time with Digital Games	1 hour a week	1	3
	30 min a day	5	2
	1 hour a day	18	8
	2 hour a day	7	3
	More than 2 hours a day	3	1
	No digital game experience	2	3

		Experimental Group	**Control Group**
		F	***F***
Type of Digital Game Experience	Racing games	11	10
	Mathematical games	13	2

	Dress Up or Cooking Games	4	1
	Coding Games	3	2
	All type of digital games	2	0

The findings also showed that there was variation in the amount of screen time among the children who played digital games. Moreover, out of 50 children who have screen experience with digital games at home; 26 of children were reported to have one hour of screen experience with digital games per day, ten children have two hours screen experience with digital games per day, seven children spend half an hour of screen experience with digital games per day, four children spend more than two hours of screen experience with digital games a day, and four children have one hour of screen experience with digital games in a week. Parents also reported that out of 55 children, respectively, they most engage in racing games (n = 21), math games (n = 15), dress up or cooking games (n = 5), coding games (n = 5) and all games that were listed in the form (n = 2). Only two children were reported to have screen experience on tablets or smartphones but no experience with digital games. Indeed, those children were noted to prefer watching cartoons or videos from the tablets or smartphones instead of engaging with digital games.

Table 4.5 Tangible Play Experience of Children at Home

The presence of Legos at home	**Yes**	**No**
	F	***F***
Experimental	35	0
Control	19	1
Whole Group	54	1

The presence of blocks at home	**Yes**	**No**
	F	***F***
Experimental	34	1
Control	19	1
Whole Group	54	1

Frequency of Lego Experience at home	***Everyday***	***1 or 2 times/days a week***	***Only with peers***	***Never***
Experimental	7	15	13	0

Control	5	8	5	2
Whole Group	12	23	18	2

Frequency of Block Play Experience at home	***Everyday***	***1-2 days a week***	***Only with peers***	***Never***
Experimental	2	16	13	4
Control	4	9	7	0
Whole Group	6	25	20	4

Finally, Table 4.5. provides information about the children's tangible play experiences at home with blocks and Legos. It was found that all children (n = 35) in the experimental group and 19 children, out of 20 children, in the control group children had Legos at home. Nevertheless, variations were noted in the frequency of Lego play among the children during the day. For instance, parents reported that four children—two from the experimental group and two from the control group—play with Legos every day alone; sixteen from the experimental group and nine from the control group play with Legos once or twice a week alone; thirteen from the experimental group and seven from the control group play with Legos only when they have friends over at home; and four children never play with Legos at home, despite having played with them before.

4.2. The Children's Existing Computational Thinking Skills: The Pretest Results

4.2.1. The Preschool Children's Baseline Computational Thinking Scores

This research aimed to investigate the development of preschool children's computational thinking skills before and after the educational robotics-based programming engagement. For this purpose, the main research question was formulated as "*Do computational thinking skills of preschool children develop with the educational robotics-based coding implementation (via TechCheck-K)?"*. To answer this question, first, the sub-research question of "*What are the general patterns of experimental and control group preschool children's pretest computational thinking scores?"* was addressed. As shown in Table 4.6., descriptive

statistics were applied to the data collected through the TechCheck-K instrument. The mean, median, standard deviation, minimum and maximum values are reported.

Table 4.6 Descriptive Pretest Results for the TechCheck-K Assessment Tool

Computational Thinking Skills	***M***	***SD***	***Mdn***	**Min**	**Max**	**N**
Whole Group	8.60	2.23	9	3	14	58
Experimental Group	8.15	2.29	9	3	13	38
Control Group	9.45	1.87	9	7	14	20
Girls	8.56	2.06	9	3	12	30
Boys	8.64	2.43	8.5	5	14	28

As shown in Table 4.6, the general computational thinking skills of preschool children ranged between 3 to 14. Schwarzer (2011) noted that, if there is no criterion to conclude the scale results of the study, the researcher can examine the median scores and compare them with the mean scores. Hence, to make inferences about the computational thinking scores of preschool children, the mean score of the TechCheck-K was controlled (M = 8.60) and found that it was around the median score of 9. Therefore, it was concluded that preschool children have a moderate level of computational thinking skills.

Table 4.7 Descriptive Pretest Results for Each Item in the TechCheck-K Assessment Tool

Items	***Pretest M***	***St. error of mean***	***SD***	**Correct Response**		**Incorrect Response**	
				f	**%**	**f**	**%**
Item 1	.60	.06	.49	35	60.3	23	39.7
Item 2	.79*	.05	.40	46	79.3	12	20.7
Item 3	.82*	.05	.38	48	82.8	10	17.2
Item 4	.70	.06	.45	41	70.7	17	29.3
Item 5	.58	.06	.49	34	58.6	24	41.4
Item 6	.60	.06	.49	35	60.3	23	39.7
Item 7	.43	.06	.49	25	43.1	33	56.9
Item 8	.56	.06	.49	33	56.9	25	43.1

Item 9	.65	.06	.47	38	65.5	20	34.5
Item 10	.37**	.06	.48	22	37.9	36	62.1
Item 11	.18**	.05	.39	11	19	47	81
Item 12	.29**	.06	.45	17	29.3	41	70.7
Item 13	.25**	.05	.44	15	25.9	43	74.1
Item 14	.86*	.04	.34	50	86.2	8	13.8
Item 15	.84*	.04	.36	49	84.5	9	15.5

*Note 1. * Represents the items with the highest mean scores*

*Note 2. ** Represents the items with the lowest mean scores*

Moreover, the computational thinking (CT) skills of preschool children were examined along with experimental and control groups. The CT skills of the experimental group ranged from 3 to 13 with a mean score of 8.15 and a median score of 9 which was slightly below the average. This means that the experimental group children have a moderate level of CT skills. Similarly, the control group scores ranged from 7 to 14 with a mean score of 9.45 and a median score of 9 which was slightly above the average. In conclusion, the control group children also appeared to have a moderate level of CT skills but have higher mean scores than the experimental group (M = 8.15) since the mean score of control group (M = 9.45) was above whole group mean score (M = 8.60).

Table 4.8 The Items with the Highest and Lowest Mean Scores in Pretest

		Correct Response		**Incorrect Response**	
	M	**f**	**%**	**f**	**%**
Item 14: Mice CANNOT pass through walls or red lights. Which mouse will get the cheese?	.86	50	86.2	8	13.8
Item 15: Mice CANNOT pass blue walls or red lights but CAN pass black tunnels. Which mouse will get the cheese?	.84	49	84.5	9	15.5
Item 3: This seesaw isn't going up and down. How can it be changed so it works?	.82	48	82.8	10	17.2
Item 2: Which works the most like a computer?	.79	46	79.3	12	20.7
Item 11: What comes next?	.18	11	19	47	81
Item 13: A circle makes a bird and a cat. A square makes a dog and a bird. What do these make?	.25	15	25.9	43	74.1
Item 12: If a triangle makes a cat and a	.29				

circle makes two birds, what do these three shapes make?		17	29.3	41	70.7
Item 10. What comes next?	.37	22	37.9	36	62.1

Tables 4.7 and 4.8 were generated in order to explain the computational thinking skills with the mean scores, including the lowest and highest mean scores. According to the descriptive statistics, item 11 (*M* = .18) which is "What comes next?" had the lowest mean scores which revealed to challenge children. This question asks children to find the next shape in a pattern and there were only 11 children, out of 58 children, who correctly answered this question. Other items with the lowest mean scores were item 13 (*M* = .25), which is "A circle makes a bird and a cat. A square makes a dog and a bird. What do these make?" and item 12 (*M* = .29) which is "If a triangle makes a cat and a circle makes two birds, what do these three shapes make?". Respectively, 74.1% and 70.7% of children gave incorrect answers to those questions. Basically, those questions are about the *representation* domain of computational thinking skills and aim at children to match the animals corresponding to the shapes and find out what they represent when they put them together, that is, when they add. These questions also consist of some *algorithms* dimension of computational thinking since it requires sequencing the animals after matching them with the correct shapes. Similarly, item 10 was one of the questions that challenged the children with a mean score of .37. Similar to item 11, this question also asks, "What comes next?" about algorithms, pattern recognition and sequencing skills and requires recognizing the pattern in a series of shapes and find the next shape in the pattern through employing sequencing and algorithmic thinking skills.

When it comes to the items with the highest mean scores, the descriptive results indicated that the items with the highest mean score were items 14 (*M* = .86), which is "Mice CANNOT pass through walls or red lights. Which mouse will get the cheese?", and 15 (*M* = .84) "Mice CANNOT pass blue walls or red lights but CAN pass black tunnels. Which mouse will get the cheese?" respectively. Those items are based on determining the correct way to reach the desired result in the tasks, *which is getting the cheese*. Children try to find the path that allows the mouse to reach the cheese, paying attention to the obstacles and the paths that the mouse should pass.

These questions are related to *control structures* of computational thinking. Similar to algorithms, the control structures mainly focus on sequential structures or repetition and cause effect relationships between the events accordingly. Other items with the highest mean scores were items 3 (M = .82) and 2 (M = .79).

While item 3 is related to *debugging* and asks children to solve seesaw tasks, the second item is related to *hardware/software* and detects children's awareness that computers work like our brains. Based on the study group's highest scores, it can be inferred that most children understand computers and possess some knowledge of both hardware and software. In addition, they can recognize and solve the bugs in a problem when it is provided with concrete tasks, such as *maze tasks*.

Table 4.9 Domain-Specific Pre-Test Results for Experimental and Control Groups

Domains	***Groups***	***M***	***SEM***	***Mdn***	***SD***	***Min***	***Max***
Hardware/ Software	Experimental	1.34	.1	1	.62	0	2
	Control	1.5	.13	2	.6	0	2
	Whole Group	**1.39**	.08	1	.61	0	2
Debugging	Experimental	1.47	.11	2	.72	0	2
	Control	1.65	.12	2	.58	0	2
	Whole Group	**1.53**	.08	2	.68	0	2
Algorithms	Experimental	2.1	.22	1	1.8	0	5
	Control	2.9	.28	1	1.29	0	5
	Total	**2.37**	.18	2	1.38	0	5
Modularity	Experimental	1	.12	1	.77	0	2
	Control	1.1	.16	1	.71	0	2
	Whole Group	**1.03**	0.98	1	.74	0	2
Representation	Experimental	.6	.1	1	.63	0	2
	Control	.45	.13	.0	.6	0	2
	Whole Group	**.55**	.08	.0	.62	0	2
Control Structures	Experimental	1.63	.09	2	.58	0	2
	Control	1.85	.08	2	.36	1	2
	Whole Group	**1.7**	.06	2	.53	0	2

Note 1. The domains represent the powerful ideas of computational thinking.

In addition to item-by-item analysis, the children's pretest computational thinking skills were investigated as domain-specific. While computing the domain-specific mean scores, children's scores for each domain were summed and divided by the

number of items in each domain. In addition, median, standard error of the means, median, standard deviation, minimum and maximum values were provided in Table 4.9. As shown in the Table 4.9., the children in the experimental and control groups received the lowest average score in the *representation* dimension out of two points (*M* = .6) and the highest average score in the *algorithms* dimension out of five points (*M* = 2.37). However, since there were five questions in algorithms dimension, this was not the actual highest mean score among all dimensions in the TechCheck-K instrument. Therefore, the dimensions with two questions were examined and the highest mean score was observed in *control structures* (*M* = 1.7). The experimental group had higher mean scores in *representation* dimension, whereas the control group had higher mean scores than the experimental group in *hardware/software, control structures, debugging, algorithms,* and *modularity* dimensions.

4.2.2. Baseline Differences in Preschool Children's Computational Thinking Scores

An independent sample t-test was performed to determine whether the differences between the experimental and control groups' baseline computational thinking skills could account for the differences. To assure the homogeneity of variance assumption, the results of Levene's test for equality of variances were checked.

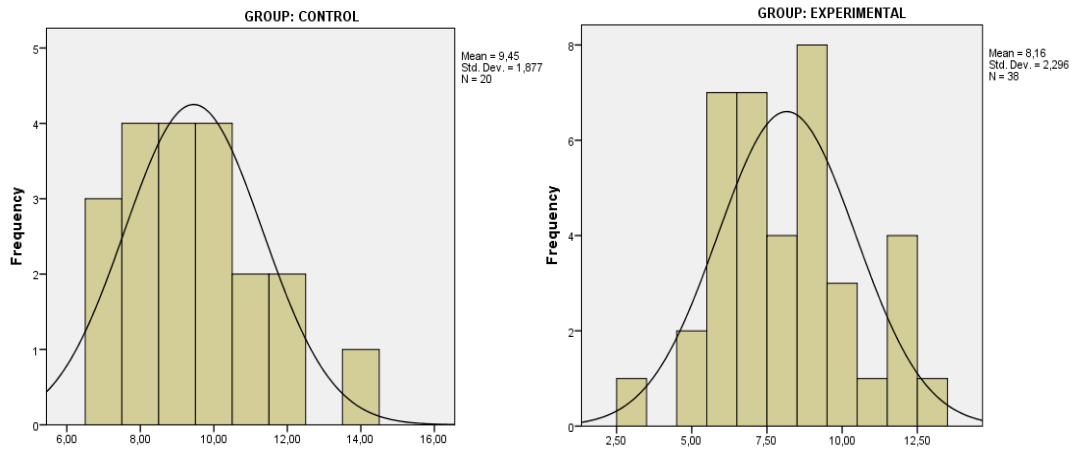

Figure 4.1 The Normality of the Distribution of Computational Thinking Scores for Control and Experimental Groups

The results revealed that the variance of computational thinking scores for experimental and control groups is the same (p =.29). Therefore, the assumption of equal variances was not violated. In addition, the distribution of scores for each of the groups were checked using histograms. The histograms revealed that the scores for each group are reasonably normally distributed, and the assumption of normal distribution was ensured (see Figure 4.1.).

The independent samples *t*-test result indicated that there was a significant difference in computational thinking scores for experimental (M= 8.16, SD=2.29) and control group of children (M= 9.45, SD= 1.87; t (56) = -2.1, p = .03, two-tailed). In other words, the control group children were found to have statistically significant higher mean scores than experimental group. The magnitude of the differences in the means (mean difference = 1.29, 95% *CI:* -2.48 to -0.9) was moderate (eta squared = .08). The results were provided in Table 4.10.

Table 4.10 Independent Samples T-test Results

	Group	***n***	***M***	***Mean Difference***	***SD***	***p***
Computational Thinking Skills	Experimental	38	8.16	-1.29	2.29	.03
	Control	20	9.45	-1.29	1.87	.03

4.3. The Children's Computational Thinking Skills: The Posttest Results

Because this research aimed to investigate the effects of KIBO educational robotics-based activities on preschool children's computational thinking skills, the children's posttest scores were examined. For this purpose, after the implementation, the post-test data were gathered through the TechCheck-K assessment tool. First, descriptive analysis was performed to identify the children's post-computational thinking scores and new answers to each item in the test. The preliminary descriptive posttest results were provided in Table 4.11.

Table 4.11 Descriptive Posttest Results for the TechCheck-K Assessment Tool

Computational Thinking Skills	***Pretest M***	***Posttest M***	***SD***	***Mdn***	**Min**	**Max**	**N**
Whole Group	8.60	9.05	2.13	9	4	13	58
Experimental Group	8.15	9.36	1.90	9	5	13	38
Control Group	9.45	8.45	2.45	8.5	4	12	20
Girls	8.56	8.96	2.18	9	4	12	30
Boys	8.64	9.14	2.12	9	5	13	28

As shown in Table 4.11., the preschool children's computational thinking scores increased in the whole research group (M = 9.05). Although the control group children had higher mean scores in the pretest results (M = 9.45), the posttest results revealed that control group children to have lower scores (M = 8.45) than the experimental group (M = 9.36). The findings indicate that following the intervention, the scores of the children in the experimental group increased by one point, while the scores of the children in the control group decreased by one point. Moreover, the children's posttest scores in each item and the items with the highest and lowest mean scores were tabulated in Tables 4.12 and 4.13. According to the results in Table 4.12., it was observed that there was a decrease in the children's scores on some items and an increase in their scores on others. While items with an increase were 1, 2, 3, 4, 5, 7, 11, 12 and 14, the items with decreased scores were 6, 8, 9, 10, 13 and 15.

Table 4.12 Descriptive Posttest Results for Each Item in the TechCheck-K Assessment Tool

Items	***Pretest M***	***Posttest M***	***SD***	**Correct Response**		**Incorrect Response**	
				f	**%**	**f**	**%**
Item 1	.60	.86*	.34	50	86.2	8	13.8
Item 2	.79*	.93*	.25	54	93.1	4	6.9
Item 3	.82*	.87*	.32	51	87.9	7	12.1
Item 4	.70	.86*	.34	50	86.2	8	13.8
Item 5	.58	.62	.48	36	62.1	22	37.9
Item 6	.60	.60	.49	35	60.3	23	39.7
Item 7	.43	.48	.5	28	48.3	30	51.7

Item 8	.56	.36**	.48	21	36.2	37	63.8
Item 9	.65	.58	.49	34	58.6	24	41.4
Item 10	.37**	.36**	.48	21	36.2	37	63.8
Item 11	.18**	.27**	.45	16	27.6	42	72.4
Item 12	.29**	.41**	.49	24	41.4	34	58.6
Item 13	.25**	.24**	.43	14	24.1	44	75.9
Item 14	.86*	.89*	.3	52	89.7	6	10.3
Item 15	.84*	.75*	.43	44	75.9	14	24.1

*Note 1. * Represents the items with the highest mean scores*

*Note 2. ** Represents the items with the lowest mean scores*

Similar to pretest results, item 13 (*M* = .24) and item 11 (*M* = .27) had the lowest scores on the TechCheck-K assessment. Respectively, 75.9% and 72.4% of children gave incorrect answers to those questions. In addition, unlike the pretest results, children's scores on item 8 dropped by .2 points. Moreover, similar to pretest results, item 10 was one of the hard questions that challenged the children in also a posttest administration with a mean score of .36. Lastly, in posttest results, item 12 was one of the items with lowest mean scores (*M* = .41). However, it was also observed that there was an increase of .12 points in the posttest results.

Table 4.13 Posttest Descriptive Statistics for the TechCheck-K Assessment Tool

		Correct Response		**Incorrect Response**	
	M	**f**	**%**	**f**	**%**
Item 2: Which works the most like a computer?	.93	54	93.1	4	6.9
Item 14: Mice CANNOT pass through walls or red lights. Which mouse will get the cheese?	.89	52	89.7	6	10.3
Item 3: This seesaw isn't going up and down. How can it be changed so it works?	.87	51	87.9	7	12.1
Item 1: Which CANNOT be programmed?	.86	50	86.2	8	13.8
Item 4: This seesaw isn't going up and down. How can it be changed so it works?	.86	50	86.2	8	13.8
Item 13: A circle makes a bird and a cat. A square makes a dog and a bird. What do these make?	.24	14	24.1	44	75.9

Item 11: What comes next?	.27	16	27.6	42	72.4
Item 8: The bunny can only hop one white square at a time. What is the fastest way for the bunny to get ONE carrot?	.36	21	36.2	37	63.8
Item 10: What comes next?	.36	21	36.2	37	63.8
Item 12: If a triangle makes a cat and a circle makes two birds, what do these three shapes make?	.41	24	41.4	34	58.6

Examining the Table 4.13., the descriptive results also revealed that the items with the highest mean score were item 2 (*M* = .93) "Which works the most like a computer?", item 14 (*M* = .89), item 3 (*M* = .87) "This seesaw isn't going up and down. How can it be changed so it works?", item 1 (*M* = .86) "Which CANNOT be programmed?", and item 4 (*M* = .86) "This seesaw isn't going up and down. How can it be changed so it works?". Unlike the pretest results, items 1 and 4 found to be one of the items with high mean score among others. While item 1 is related to *hardware and software* domains of computational thinking, item 4 is based on *debugging* and children solve seesaw tasks as in the third item. According to the highest scores in study group, it can be concluded that after the treatment the items in hardware/software and debugging domains showed increase in most of the children. In order to see the change in computational thinking domains, Table 4.14 and Figure 4.2 were created.

To begin with the Table 4.14, the descriptive results for computational thinking domains indicated that after the Coding as Literacy-KIBO (CAL-KIBO) kindergarten implementation, the experimental group children improved their scores at hardware/software domain with a mean score of 1.89; debugging domain with a mean score of 1.73; algorithms domain with a mean score of 2.23; representation domain with a mean score of .76 and control structures domain with a mean score of 1.68, and modularity with a mean score of 1.05. Similarly control group children increased their scores in hardware/software (*M* = 1.6), modularity (*M* = 1.15) and debugging (*M* = 1.73) domains. The results also indicated that there was a decline in algorithms (*M* = 2.15), and control structures (*M* = 1.6) and no change in representation domains (*M* = .45).

Table 4.14 Domain-Specific Post-Test Results for Experimental and Control Groups

Domains	***Groups***	***Pretest M***	***Posttest M***	***SEM***	***Mdn***	***SD***	***Min***	***Max***
Hardware/ Software	Experimental	1.34	1.89	.05	2	.31	1	2
	Control	1.5	1.6	.13	2	.59	0	2
	Whole Group	1.39	1.79	.05	2	.44	0	2
Debugging	Experimental	1.47	1.73	.08	2	1.73	0	2
	Control	1.65	1.75	.12	2	.12	0	2
	Whole Group	1.53	1.74	.07	2	.54	0	2
Algorithms	Experimental	2.1	2.23	.19	2	1.17	0	5
	Control	2.9	2.15	.28	2	1.29	0	4
	Total	2.37	2.20	.15	2	1.15	0	5
Modularity	Experimental	1	1.05	.11	1	.69	0	2
	Control	1.1	1.15	.15	1	.67	0	2
	Whole Group	1.03	.65	.08	1	.66	0	2
Representation	Experimental	.6	.76	1	1	.67	0	2
	Control	.45	.45	.15	0	.6	0	2
	Whole Group	.55	.65	.08	1	.66	0	2
Control Structures	Experimental	1.63	1.68	.09	2	.57	0	2
	Control	1.85	1.6	.15	2	.68	0	2
	Whole Group	1.7	1.65	.07	2	.6	0	2

Note 1. The domains represent the powerful ideas of computational thinking.

Following that the figure 4.2. reveals a post-hoc item analysis which examined the change in the percentage of correct responses in the computational thinking domains. While creating the figure, the children's correct responses for each TechCheck-K item was calculated for pretest and posttest scores and then averaged within each computational thinking domain. After that, pretest percentages for each domain were subtracted from posttest percentages to examine the change over the course of eight-week intervention for the preschool children who received CAL-KIBO kindergarten curriculum and did not receive the CAL-KIBO kindergarten curriculum. In addition, while experimental group was named as CAL, the control group children was named as no-CAL in the figure. The post-hoc item analysis results showed that after the treatment algorithms, modularity, and control structures revealed slightly less change in the CAL group children than the other domains. In addition, the largest percentage of improvement in the CAL group was in hardware/software, debugging, and representation domains respectively. Furthermore, the no-CAL group received the biggest percentage changes in hardware/software, debugging, and modularity, while

algorithms and control structures revealed a decline in the percentage of correct responses. Finally, for the no-CAL group, there was no percentage change in the correct answers for the representation domain (see Figure 4.2.).

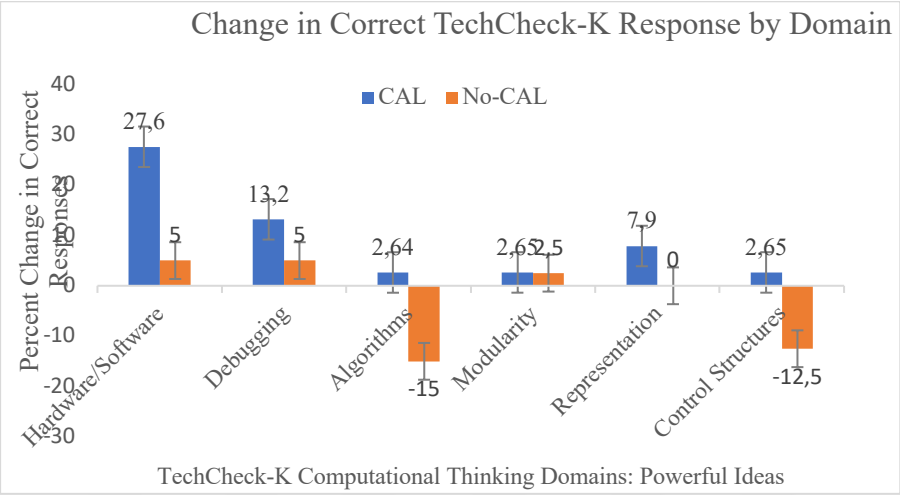

Figure 4.2 Changes in Percent Preschool Children Making Correct Responses in the Six Domains of TechCheck-K

4.3.1. The Effect of the Coding as Literacy-KIBO Kindergarten Curriculum on Computational Thinking Skills: One-Way ANCOVA Results

Since this research focused on the developmental change in computational thinking skills of preschool children who received robotics-based coding instruction and did not receive robotics-based coding instruction, the preschool children's pretest and posttest scores were compared to address the question "*Does educational robotics-based coding implementation produce greater computational thinking scores in the experimental group children than in the control group children after adjusting for pretest differences in computational thinking scores*?". Therefore, following the descriptive results, a one-way between-groups analysis of covariance was conducted to compare the effect of the educational robotics-based coding implementation on preschool children's computational thinking skills. The independent variable was the educational robotics-based coding implementation namely Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum, and the dependent variable consisted of scores on the TechCheck-K assessment tool administered after the implementation

was completed. Preschool children's scores on the pretest administration of the TechCheck-K assessment tool were used as the covariate in this analysis. Preliminary checks were conducted to ensure that there was no violation of the assumptions of normality, linearity, homogeneity of regression slopes, homogeneity of variances and reliable measurement of the covariate.

Table 4.15 One-way ANCOVA Results

Source of the Variance	Sum of Squares	df	Mean Square	F	*p*	η^2
Corrected Model	58,708[a]	2	29,354	7,987	,001*	,225
Pretest	47,655	1	47,655	12,967	,001*	,191
Group	26,115	1	26,115	7,106	,010	,114
Error	202,137	55	3,675			
Total	5013,000	58				
Corrected Total	260,845	57				

a. R Squared = .225 (Adjusted R Squared = .197)

After adjusting for pretest scores, it was found that there was a significant difference between the experimental and control groups on posttest scores on the TechCheck-K assessment tool, F (1, 55) = 7.106, p = .01, partial eta squared = .114. This means that robotics-based coding implementation via the CAL-KIBO kindergarten curriculum made a significant effect on preschool children's computational thinking skills. Considering the mean difference in the pretest scores of experimental and control group children (mean difference = .91), the results showed the mean scores of the experimental group to increase from 8.16 to 9.36 while the control group scores decreased from 9.45 to 8.45. The partial eta squared value indicated a small effect size and only 11.4 percent of the dependent variable is explained by the independent variable. In addition, there was a significant relationship between the pre-test and post test scores on the TechCheck-K assessment tool [F (1, 55) = 12.967; p = .001]. This means that the pretest scores of preschool children are a predictor of posttest scores. As indicated by a partial eta squared value of .19, the pretest scores of preschool children explained 19 percent of the variance in the independent variable. Finally, it is seen that the obtained pretest scores and the

group variable together explain a significant part of the change in the adjusted posttest mean scores with 23% (η^2 = .225), and the ANCOVA model is statistically significant. Detailed information about one-way ANCOVA results and estimated means was provided in Table 4.15 and 4.16.

Table 4.16 Estimated Marginal Means

Groups	***M***	**Std. Error**	**95 % Confidence Interval**	
			Lower Bound	**Upper Bound**
Experimental	9,558[a]	,315	8,926	10,191
Control	8,089[a]	,440	7,207	8,971

a. Covariates appearing in the model are evaluated at the following values Pretest = 8,6034

4.4. The Children's Programming Related Computational Thinking Skills: The TACTIC-KIBO Results

One of the aims of this research was also identifying preschool children's programming related computational thinking skills. For this purpose, the children's computational thinking skills were measured by programming tasks with the TACTIC-KIBO assessment tool and the children's programming levels were categorized as proto-programmer, early programmer, and programmer. In addition, this assessment was only administered to the children in the experimental group since the TACTIC-KIBO requires programming experience with KIBO robotics prior to administration and mainly measures preschool children's computational thinking skills by their programming performance. The TACTIC-KIBO administration was carried out individually for each child. In the original application of the tool, preschool children are measured step by step. In order to pass the first level (proto programmer), the child should correctly answer three questions. Otherwise, the second level (early programmer) questions are not asked to the child. Similarly, the children who cannot pass the second level are not asked questions of the third level (programmer). However, in this study, the researcher asked all questions in each level to the children and wanted to see which behaviors the

children could gain from each programming level after the CAL-KIBO kindergarten curriculum implementation but categorized the children in accordance with the original scoring system. Thus, children who could complete three tasks or more were labeled as proto programmers; children who finished four questions or more in the second level but not the third level were labeled as programmers; children who finished four questions or more in the second level but not the third level were labeled as programmers. Table 4.17 shows the preschool children's computational thinking levels that were derived from the TACTIC-KIBO assessment tool.

Table 4.17 The TACTIC-KIBO Results

The Programming Level	Number of Children	Gender	
		Boys	Girls
Proto programmer	3	2	1
Early programmer	4	1	3
Programmer	31	14	17
N	*38*	*17*	*21*

The TACTIC-KIBO results indicated that most of the children (n = 31) reached at the programmer level after implementing the CAL-KIBO kindergarten curriculum. There were only three proto-programmer and four early programmer children after the intervention. This may reveal that CAL-KIBO kindergarten curriculum has a great potential to contribute preschool children's programming related computational thinking skills and enable children to reach at programmer level. Following that, the children's correct and incorrect answers were examined for each level and the powerful ideas of computational thinking. The children's success and failure for each powerful idea were investigated in the following sections.

4.4.1 The Children's Proto-Programmer Behaviors in terms of Powerful Ideas

Figure 4.3. reveals that the first program in the TACTIC-KIBO assessment involves begin, forward, and end blocks. The administrators are supposed to use this sequence to ask all tasks on this level. Therefore, before administration, the researcher prepared this program and asked the questions without using other motion blocks.

However, since the design process task involves the light sensor and white light on block, the researcher also had them ready to use.

Figure 4.3 The First Program Used in the Proto-Programmer Level

Table 4.18 reveals that after the TACTIC-KIBO proto-programmer administration most children were successful at hardware/software (n = 38), control structures (n = 32), representation (n = 38), algorithms/modularity (n = 35), debugging (n = 36) and design process (n = 27) dimensions. This result revealed that all children in the experimental group accomplished the hardware, software, and representation tasks correctly.

Table 4.18 The TACTIC-KIBO Results for the Proto Programmer Level

The Powerful Ideas of Computational Thinking	***Control Structures***	***Hardware***	***Software***	***Representation***	***Algorithms/ Modularity***	***Debugging***	***Design Process***
Number of Correct Answers	32	38	38	38	35	36	27
Number of Incorrect Answers	6	-	-	-	3	2	11

This means, most children in this study learned the programming blocks and the symbols on the blocks. Moreover, almost the entire group were successful at control structures, debugging and algorithms/modularity domains. This indicates that most of the children in this research acquired the related proto-programmer skills regarding control structures, debugging, algorithms, and modularity.

Considering the incorrect responses, the most challenging task at the first level was the **design process** which is an iterative process that includes defining an objective, the solution ways to be followed, testing, revising, and retesting them if necessary. The design process task in TACTIC-KIBO is: "*Let's pretend it is nighttime and KIBO needs a light to see how to get home. The light has to turn on before it moves forward. Here is a light sensor [examiner puts it in] and a "white light on" block. How can you make the light come on before KIBO moves?"*. This task requires putting the white light on block before the forward block so that KIBO turns on the light before moving forward. During the task, there were children who immediately ordered the blocks and corrected their program when made a mistake. However, 11 children, out of 38, were confused about where to put the light block. Some children arranged the forward block after the white light on block and some only used the white light on block to program KIBO. Additionally, some children did not realize that they had made a mistake by failing to fix their blocks. Although this task is based on a powerful idea of the design process, it also involves the powerful idea of **algorithms/modularity**. This is because it requires breaking the task into smaller parts (choosing the right blocks in a set of blocks) and ordering the tasks in a correct sequence to solve the problem or achieve a desired goal (putting the white light on block before the forward block, putting begin and end blocks to the correct places in the order). Therefore, it can be stated that during this task the children who could not accomplish this task were also challenged by algorithms and modularity related skills.

Another powerful idea that challenged 6 children out of 38 children was the **control structures**. This task requires answering the KIBO's behavior/s in order and the reason behind its behavior. While most of the children correctly answered the question, some children either could not accomplish the task or gave an unrelated answer. Also, there were children who explained the behavior of KIBO but could not recognize the reason. This result could mean that 6 children in this study do not know or recognize that KIBO moves since it was programmed with programming blocks. Other tasks that were incorrectly answered by children were **algorithms/modularity** (n =3) and **debugging** (n = 2). The algorithms/modularity task asks "*[Take away begin block and show remaining 2] "What would happen if I*

just used these two blocks?". During the algorithms/modularity task, the children should remember that the order in the programming blocks matters and there should be begin block at the beginning of the program. The children are also supposed to recognize the begin block is missing and there is an error in the program. Therefore, KIBO will beep with error sound and do not move. In this sense, it can be stated that the algorithms/modularity task also requires debugging skill and one who cannot accomplish the algorithms/modularity may have a difficulty to complete debugging task. In fact, debugging task in the proto-programmer level requires children to control the order of begin and end blocks and fix it so that KIBO works; and, examining the children who could not accomplish algorithms/modularity task (n = 3), two of them were found to also fail at debugging task. In addition, the same children also could not complete the design process task which necessitates to design a program that needs a light sensor before the forward block and control structures. These results revealed that the algorithms/modularity, debugging and design process tasks complement each other and have an interrelated relationship.

4.4.2 The Children's Early Programmer Behaviors in terms of Powerful Ideas

Figure 4.4 The Second Program Used in the Early Programmer Level

Figure 4.4. indicates that the second program in the TACTIC-KIBO assessment includes begin, white light on, forward and end blocks. The administrators are supposed to use this sequence and blocks to ask all questions on this level. Therefore, before administration, the researcher prepared the program in Figure 4.3 and asked the questions without using other blocks. In addition, since the design process task involves the number parameters and repeat blocks, the researcher also had them ready to use. Table 4.19 shows that most children were successful at hardware (n = 35), software (n = 38), control structures (n = 37), representation (n = 32),

algorithms/modularity (n = 34), debugging (n = 33) and design process (n = 23) dimensions. This result indicated that almost all children gave satisfactory answers to the second level tasks and were more successful at software, control structures, and hardware. However, compared to the number of correct answers in the first level, there is a decrease in the number of correct answers in each domain.

Table 4.19 The TACTIC-KIBO Results for the Early Programmer Level

The Powerful Ideas of Computational Thinking	***Control Structures***	***Hardware***	***Software***	***Representation***	***Algorithms/ Modularity***	***Debugging***	***Design Process***
Number of Correct Answers	37	35	38	32	34	33	23
Number of Incorrect Answers	1	3	-	6	4	5	15

Considering the incorrect answers to the second level tasks, early programmer results revealed the **design process** task was the most challenging task for the preschool children in this study. Actually, 15 children, out of 38, could not complete this task. The task was *"KIBO was able to move but it didn't go very far forward. If I only have 1 block that tells it to move forward, how can I make it move farther forward the next time?"*. While answering this question, the child can use repeat blocks with parameters or scan the forward block for many times. The children can also push the green button on the KIBO robot many times so that the program repeats many times. During the administration, none of the children pushed the green button on the KIBO robot many times.

Moreover, there were children who remembered to use repeat blocks and parameters but mostly suggested to scan forward block many times since they could not program the KIBO with a synthetically correct algorithm. Although it was limited, there were also children who used repeat blocks and parameters correctly. Those answers were scored as satisfactory as it was suggested in TACTIC-KIBO administration tool.

Following that, **representation (*n* = 6), debugging (*n* = 5)** and **algorithms/modularity (*n* = 4)** were other tasks that challenged the children. In fact, most of the children seemed to be not understanding the **representation** task and could not answer the question. For instance, some of the children replied, "How so? I did not understand the question.". Then, the researcher said, "Are these blocks the same?". The children stated that the blocks are not the same. Then the researcher added that "Then, what is the difference between the blocks?". Following that, some children could understand and answer the question. In fact, the representation task is stated as "*What is the difference between the blue blocks and the yellow blocks?*". In the Turkish version of the instrument, the sentence was translated as *"Mavi bloklar ile sarı bloklar arasındaki fark nedir?"*. The word "difference" was translated as "fark" and some children could not understand that word. Whereupon the researcher used the word *"*farklılık" instead of the word *"*fark*",* and most of the children who could not understand the task were able to answer.

Moreover, the **debugging** task is as follows: *[Rescan without white light on block] "I want the robot to put a white light on, go forward, and end."* During this task, the children should be aware that white light on block is missing in the algorithm. In addition, they should put this block right after the begin block. Most of the children (*n* = 33) realized the missing block and put it to the right order and only five children put the white light on block after the forward block. This means that most children in this study could create a goal-oriented program, find the error in a program, and fix it.

In addition, four children could not achieve the **algorithms/modularity** task. This task requires explaining what is changed in the program and is asked as follows: *[Show blocks, switch order to begin, forward, white light, end] "What will be different about this program if I do this?".* Examining the children who could not accomplish this task, it was found that the children who could not achieve the debugging, also failed at the algorithms/modularity task which was in line with the results gathered in the proto-programmer level. These findings also showed that the domains of debugging and algorithms/modularity complement one another, and

children who excel in these areas are likely to excel in debugging and design process domains.

4.4.3 The Children's Programmer Behaviors in terms of Powerful Ideas

Figure 4.5 The Third Program Used in the Programmer Level

Figure 4.5. indicates that the third program in the TACTIC-KIBO assessment includes begin, repeat, number parameter, white light on block, forward block, end repeat block and end block. Throughout the third level, the administrators should use this sequence and blocks to ask all questions on this level. Therefore, before administration, the researcher prepared this program and asked the questions without using other blocks. In addition, since the design process task involves the distance sensor, and until near sticker, it was necessary to those materials before administration. However, the design process of the programmer level was not evaluated in this study. This is due to the fact that developing a program with numerous parameters becomes more complex for young children as the number of parameters increases and is not covered in the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum.

To examine the programmer level, table 4.20 was created and the TACTIC-KIBO results for programmer tasks indicated that the **hardware** question could not be correctly answered by 27 children. There were only seven children who remembered *"What does the green board do?"*. Most children could not remember the name of the green board or its function. Instead, while some children said the main board makes the robot work, others stated they do not know. This showed that most children in this study do not know that it is KIBO's brain or circuit board or computer in KIBO.

Table 4.20 The TACTIC-KIBO Results for the Programmer Level

The Powerful Ideas of Computational Thinking	***Control Structures***	***Hardware***	***Software***	***Representation***	***Algorithms/ Modularity***	***Debugging***	***Design Process***
Number of Correct Answers	38	11	33	27	30	31	*
Number of Incorrect Answers	-	27	5	11	8	7	*

* *The design process was not measured because it includes an if conditional statement task and the associated task is not taught in the CAL-KIBO kindergarten curriculum.*

Following that, the **representation** question was the next challenging task which was not accomplished by 11 children out of 38 children. Like the representation task in the early programmer level, the programmer level task requires explaining the difference between two kinds of blocks that are yellow light blocks and gray repeat blocks. The question is as follows: *What is the difference between the gray block and the yellow block?"*. Similar to the procedure followed in the early programmer level, the researcher used the word "farklılık" instead of the "fark", and most of the children were able to answer the question correctly. This result shows that 27 children out of 38 were able to discriminate the yellow and gray blocks and their functions in programming KIBO.

Other domains that were not correctly completed by children were **algorithms/modularity** (n = 8) and **debugging** (n = 7). The **algorithms** task asks, "*What would I change if I wanted it to repeat 4 times?".* In this task, children can suggest changing or change the sticker on the repeat block with number 4, scanning the forward block twice or scanning the white light on block twice since the repeat block has number parameter 2. This means that the children should be aware of the number parameter -number 2- and suggest or change it on the repeat block with number parameter 4. The children may also recognize that the sum of two twos is four and may scan the number parameter on the repeat block twice. Most importantly, the children were asked to repeat the KIBO four times, but they were not told which block should repeat four times. Therefore, they should also decide

which action KIBO will repeat and choose either the white light on block or the motion block. During the administration, the children who successfully completed this task (n = 30) suggested or changed the sticker on the repeat block with number 4, but there were no children who scanned the forward block twice or the white light on block twice. This means that the children were not aware of the sum of two twos is four. For the children who suggested to change the sticker with number 4, the researcher also asked how they could scan programming blocks so that KIBO repeat four times. By doing this, these children's recognition about the repeat blocks would be better understood and children could be observed if they move the repeat blocks from the program and scan the light and motion blocks for four times or if they keep the number parameter 4 on the repeat block and scan each block once. After the question, most children kept the number parameter 4 and scanned each block once, but there were also children who scanned all the blocks in the program four times. This means that although these children suggested to change number sticker on the repeat block with 4, they are not aware of the function of repeat blocks and the number parameter.

Following that, the debugging task asks *[Move the white light on block to before the repeat blocks.] "I want the white light on to flash twice. How can I do that?"*. The children should be able to identify during the task that the repeat loop has been altered and that, after also changing the sticker to represent number 2, the white light on the block can be put back in the loop. Additionally, they can exit the repeat loop by scanning the white light on the block twice; they do not need to modify the repeat block's number parameter. They can also use a third option, which is to use an additional "white light on block" and scan two white light on blocks in the program by placing it next to the other one. Luckily, in the debugging task most of the children were able to recognize the repeating action in the program and change the program for a desired purpose (n = 31). A small percentage of children carried the white light on block in the loop, despite the majority of them preferring to scan it twice rather than moving it. The data indicates that a majority of the children in this study are able to repeatedly scan the programming blocks and understand that doing so causes KIBO to repeat its actions. However, given that only five children

proposed moving the white light on block in the loop and changing the number parameter 4 to 2, it is possible that these children lack knowledge about repeat loops. The results also showed that the children in this study are lack in providing alternative solutions for the same purpose, because none of the children proposed to use a second white light on block so that KIBO flashes twice. Lastly, seven children out of 38 could not accomplish the debugging task in programmer level. This showed that those children could not recognize the repeat loop in the program and could not arrange the algorithm so that KIBO flashes twice. Those children either did not respond to the question or arranged syntactically incorrect program.

4.5. The Preschool Children's Computational Thinking Skills in Terms of Background Factors: Three-Way ANCOVA Results

Table 4.21 Three-Way ANCOVA Results for Preschool Children's Background Factors

	Type III Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**	**Partial Eta Squared**
Intercept	88,377	1	88,377	23,803	,000	,310
Pretest	38,930	1	38,930	10,485	,002	,165
Group	27,498	1	27,498	7,406	,009	,123
Gender	1,030	1	1,030	,278	,601	,005
Age in months	4,211	1	4,211	1,134	,292	,021
Error	196,779	53	3,713			
Total	5013,000	58				
Corrected Total	260,845	57				

This research also sought to examine the preschool children's background factors if they contribute to their computational thinking scores. For this purpose, a demographic information form was sent to the mothers and fathers. There were questions related to children's block time, Lego time, screen time with digital games, gender, age in months, ScratchJr time, parent's familiarity with ScratchJr, mother's and father's age, and education level. While collecting this data, it was aimed to have a broad knowledge of the children participating in the study and examine the

impact of screen time with digital games, gender, and age in months on their posttest computational thinking scores. However, as the demographic results revealed, the sample was not large enough to have enough participants in the subgroups regarding children's screen time with digital games, ScratchJr time, and block and Lego experience, which prevented comparative analyses to examine differences in computational thinking skills in terms of the background factors. Therefore, only age in month and gender variables were examined as background factors.

In this sense, it was essential to conduct a three-way analysis of covariance (ANCOVA). However, before conducting the analysis, examining interaction effects between covariate (pretest), dependent variable (posttest) and group variable (experimental and control groups) was necessary. This is because, if there is an interaction effect between these variables, extra analyses would be required and the reporting of the three-way ANCOVA analysis would change. In the preliminary analysis, therefore, Univariate General Linear Model was performed, and the interaction effects were checked by assigning the posttest scores to dependent variable, pretest scores to the covariate variable and group to the fixed factor. Following that, in the model section the interaction between pretest and group was selected and analysis was performed. The results revealed that there was no interaction effect between the covariate and group variables. This means that main effects can be controlled in the next step by examining the contributor roles of gender and age in months on the computational thinking skills.

Before conducting the three-way ANCOVA, the categories of age in month variable were recoded. The children's ages in months were categorized as "62 to 72 months old children" and "72.01 to 80 months old children". The number of samples and the developmental level of the children were considered when deciding on the month groups. The 72-month-old preschoolers were chosen as the cut point for grouping the children. While 28 children were included in the 62-to-72-month group, 27 children out of 55 were included in the 72.01-to-80-month group. Once the groups were established, preliminary checks were made to ensure that the assumptions of normality, linearity, homogeneity of regression slopes, homogeneity of variances,

and reliable measurement of the covariate which were not violated. When setting the analysis options, the Bonferroni-based confidence interval adjustment was selected to control the Type I error, and the analysis was performed.

Table 4.22 Unadjusted Mean Change in Control and Experimental Group Children regarding Age in Month

Group	Age in months		Pretest	Posttest
Experimental	62 to 72 months old children	Mean	7,7143	9,2381
		N	21	21
		Std. Deviation	1,82052	1,78619
		Std. Error of Mean	,39727	,38978
	72.01 to 80 months old children	Mean	8,7059	9,5294
		N	17	17
		Std. Deviation	2,73324	2,09516
		Std. Error of Mean	,66291	,50815
	Total	Mean	8,1579	9,3684
		N	38	38
		Std. Deviation	2,29602	1,90903
		Std. Error of Mean	,37246	,30968
Control	62 to 72 months old children	Mean	8,8889	7,2222
		N	9	9
		Std. Deviation	1,53659	2,48886
		Std. Error of Mean	,51220	,82962
	72.01 to 80 months old children	Mean	9,9091	9,4545
		N	11	11
		Std. Deviation	2,07145	2,01810
		Std. Error of Mean	,62457	,60848
	Total	Mean	9,4500	8,4500
		N	20	20
		Std. Deviation	1,87715	2,45967
		Std. Error of Mean	,41974	,55000

The three-way ANCOVA was performed on the computational thinking skills of the preschool children to control the effect of background factors. The independent variables consisted of gender and age in months (62 to 72 months old children and

72.01 to 80 months old children). These variables were entered as fixed factors to the model. Another fixed factor was research group including the experimental and control groups to control the changes in accordance with group. The covariate variable was the pretest scores on the computational thinking skills of the preschool children. The results indicated that after adjusting for the pretest scores, there was not a significant difference between the experimental and control group children F (1, 53) = 1.134, p = .2, partial eta squared = .021 on the posttest scores of the TechCheck-K assessment tool in terms of age in months.

Moreover, there was not a significant difference between the experimental and control group children F (1, 53) = .278 p = .6, partial eta squared = .005 on the posttest scores on the TechCheck-K assessment tool in terms of gender. This means that age in months and gender are not predictors of the computational thinking skills of the preschool children in this study. The results of three-way ANCOVA were tabulated in Table 4.21.

In addition, it was observed that there was a one-point difference between the pretest scores of the children in both the experimental and control groups with respect to their age in month groups. However, after the intervention, a decrease was observed in the difference between the posttest scores of the children in the experimental group, although the difference between the posttest scores of the children in the control group increased.

This may indicate that the implementation of the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum contributed on the computational thinking scores of the experimental group children and reduced the score gap between the age in month groups.

In addition, the unadjusted mean comparisons showed that the group that increased their scores the most in the experimental group were the children in the 60–72-month age group (see Table 4.22).

Line graphs showing the pretest scores on the computational thinking mean scores and posttest scores based on estimated marginal means were created and presented for age in months and gender variables in order to compare the changes in the mean pretest and posttest scores of the preschoolers.

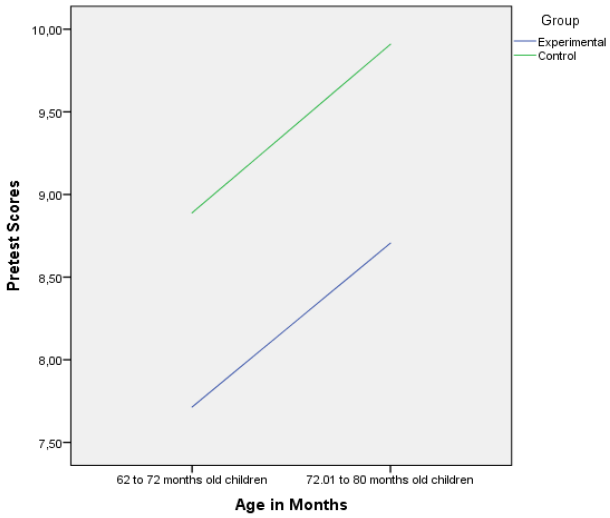

Figure 4.6 Preschool Children's Pretest Computational Thinking Scores regarding Age in Months

Starting with age in months, it was observed that children 72.01 to 80 months old children had higher computational thinking scores than children 62 to 72 months old children in both the control and experimental groups (see Figure 4.6. and 4.7.).

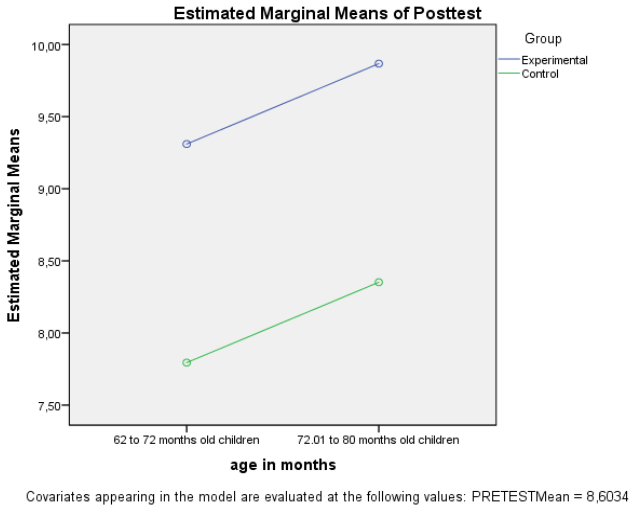

Figure 4.7 Preschool Children's Estimated Marginal Means of Posttest Scores regarding Age in Months

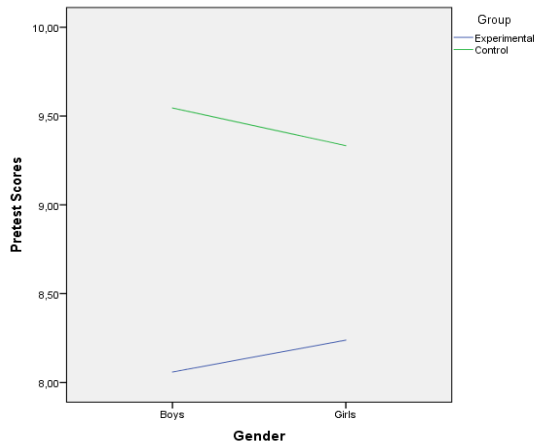

Figure 4 8 Preschool Children's Pretest Computational Thinking Scores regarding Gender

Subsequently, in the pretest scores it was observed that girls in the experimental group had the higher mean scores than boys, while boys in the control group had higher scores than girls. In addition, the difference in mean scores between the girls and boys were the same for both the control and experimental groups. After the treatment, the girls and boys in the experimental group were observed to increase their computational thinking scores.

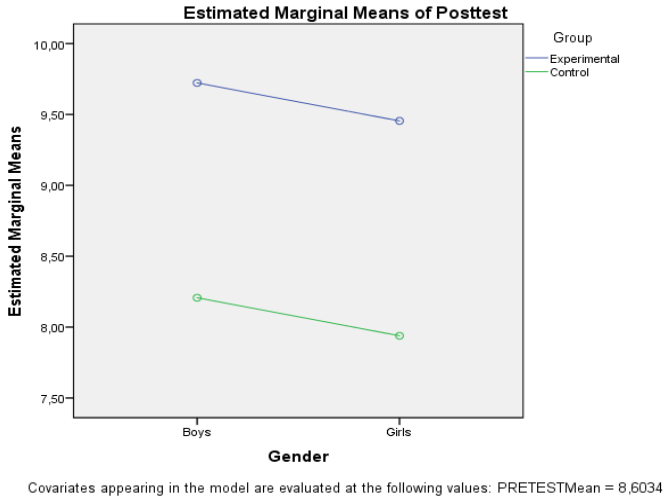

Figure 4.9 Preschool Children's Estimated Marginal Means of Posttest Scores regarding Gender

However, similar to the pretest results the girls in the experimental and the boys in the control group had higher mean scores than others. In addition, the mean

difference between boys and girls remained similar in the control and experimental groups (see Figure 4.8. and 4.9.).

4.6. Summary of the Findings

One of the main objectives of this study was to investigate the effect of educational robotics-based coding and programming activities on the computational thinking skills of preschool children. Therefore, the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum was implemented with preschool children and its effects were controlled in two steps. In the first step, the development of computational thinking skills of children in the experimental group was compared with a control group that did not receive any treatment to improve their computational thinking skills. Mean pretest and posttest scores for each group were checked using the TechCheck-K assessment tool. Independent samples t-test results showed that there was a significant difference between the pretest scores of preschool children in favor of control group children. This means that before the treatment, the control group children reported higher mean scores than the experimental group and the difference was significant.

After the implementation of CAL-KIBO kindergarten curriculum, one-way ANCOVA was performed, and a significant difference was found in the computational thinking skills of the experimental and control group children. Specifically, the experimental group children showed higher mean scores than the control group. This indicates that the CAL-KIBO kindergarten curriculum, which includes KIBO educational robotics-based coding and programming activities, can lead to significant and positive changes in preschool children's computational thinking skills. In addition, post-hoc analyses based on the percentage changes in pre- and post-test scores in the control and experimental groups revealed that the largest percentage of improvement in the experimental group was in the hardware/software, debugging, and representation domains. Algorithms, modularity, and control structures showed slightly less change in the experimental group children. Furthermore, the control group's percentage changes showed that the hardware/software, debugging, and modularity domains received the biggest

percentage changes, whereas algorithms and control structures showed a decline in the percentage of correct answers. Lastly, the control group's percentage of correct answers for the representation domain did not change.

Another aim of this study was to examine the effect of the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum on the programming-related computational thinking skills of the children in the experimental group. For this purpose, the children's programming-related computational thinking skills were assessed using the TACTIC-KIBO assessment tool, which includes coding and programming-based tasks performed by KIBO robotics. The majority of the children in the experimental group (n = 31) reached programmer level, according to the results. Out of 38 children, only four were early programmers and three were proto-programmers. Examining computational thinking domains, it was found that the most challenging domain for the preschool children was the design process at the proto-programmer and early programmer levels.

During the administration of the programmer level, the design process could not be elaborated because it involves a distance sensor and parameters that are not taught in CAL-KIBO kindergarten curriculum. The TACTIC-KIBO results also showed that the children's skills in algorithms, modularity and sequencing are important to complete the computational tasks. This was because most of the tasks in the TACTIC-KIBO comprised the use of these skills, and the children who could not perform the algorithms/modularity tasks, which involve coding a set of ordered instructions and considering the specific order in the program, also failed the debugging, control structures or design process tasks. In other words, the tasks in the TACTIC-KIBO administration complement each other and the children who cannot accomplish a debugging task can also fail at algorithms/modularity or vice versa. To illustrate, a child who can recognize an error in the order of the start and end blocks (algorithms/modularity and debugging) would also know that KIBO does not move (control structures). Therefore, a task that measures the control structures of computational thinking also requires debugging and algorithms/modularity. This study provided evidence that the various domains of computational thinking skills

are interconnected, dynamic, and multidimensional, with a lack of knowledge or proficiency in one domain potentially impacting success in other dimensions.

Finally, this research aimed to explore the effect of background factors on the computational thinking skills of preschool children. For this purpose, a demographic information form was sent to the mothers and fathers. While collecting this data, the researcher aimed to have a broad knowledge of the children participating in the study and aimed to examine the impact of screen time with digital games, gender, and age in months on posttest computational thinking scores. However, the predictor roles of gender and age in month was investigated due to the insufficient data. The three-way ANCOVA results showed that there was no significant difference between the children in the control and experimental groups in terms of their gender and age in months.

Examining the line graphs through the pretest means and estimated marginal posttest means, although there was no significant difference in the computational thinking skills of the experimental and control group children in terms of age in months, the mean scores in the experimental group showed that the gap between the 62–72-month-old children and the 72.01-80-month-old children decreased after the treatment. This may show the effect of the CAL-KIBO kindergarten curriculum on the computational thinking skills of the preschool children who participated in this study. This was also because the mean difference between the pre- and post-test scores of the children in the control group was the same for the 62- to 72-month-old children and the 72.01- and 80-month-old children who did not take any treatment to advance their computational thinking skills.

Moreover, there was no difference in pretest mean scores between the girls and boys for both the control and experimental groups. After the implementation, the girls and boys in the experimental group were observed to increase their computational thinking scores. However, similar to the pretest results, the girls in the experimental group and the boys in the control group were found to have higher posttest mean

scores than others. Lastly, the posttest mean difference between boys and girls was similar and not large in both research groups.

4.6.1 Key Findings

The key findings of the current study are summarized below:

- The item response results revealed that the TechCheck-K is more powerful in discriminating Turkish preschool children at the moderate and below average level of computational thinking.

- Before the implementation of Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum, there was a significant difference in the computational thinking skills between the pretest scores of preschool children in favor of the children in control group.

- After the implementation of CAL-KIBO kindergarten curriculum, there were differences in the powerful ideas of the experimental group depending on their percentage of development. The post-hoc item analysis showed that algorithms, modularity, and control structures (with a ceiling effect) showed slightly less change, while the largest percentage of improvement was found in hardware/software, debugging, and representation.

- After the implementation of CAL-KIBO kindergarten curriculum, there was a significant difference in the computational thinking skills of the control group and experimental group children in favor. This may mean that KIBO educational robotics-based coding and programming activities, can lead to significant and positive changes in the computational thinking skills of preschool children.

- After the implementation of CAL-KIBO kindergarten curriculum, the plugged-in TACTIC-KIBO results showed that most of the children in the

experimental group reached the programmer level (n = 31 out of 38 children).

- The TACTIC-KIBO results also indicated that the most challenging domain for the preschool children was the design process at the proto-programmer and early programmer levels.

- The TACTIC-KIBO results also showed that the most challenging task for the preschool children was the repetition loops as most of the children could not create syntactically correct repeat loop programs.

- Each domain of computational thinking skills is interrelated; that is, they are dynamic, multidimensional, and a lack of knowledge or skills in one domain of computational thinking could affect success in other dimensions.

- After the implementation of CAL-KIBO kindergarten curriculum, there was no significant difference between the computational thinking skills of the control and experimental groups in terms of age in months (via TechCheck-K). However, the gap between the 62–72-month-old children and the 72.01-80-month-old children in the experimental group decreased after the treatment, which may indicate the contribution of the CAL-KIBO kindergarten curriculum to the computational thinking skills of preschool children.

- After the implementation of CAL-KIBO kindergarten curriculum, there was no significant difference in the post-test scores of girls and boys in the experimental group were observed to increase their computational thinking scores (via TechCheck-K).

CHAPTER 5

DISCUSSION

The current study aimed to investigate the effect of educational robotics-based coding and programming activities on the computational thinking skills of preschool children. In such a manner, the TechCheck-K, an unplugged assessment tool, and the TACTIC-KIBO, a plugged-in assessment tool, were used in this study to examine the computational thinking skills of preschoolers. This allowed for an examination of the preschoolers' computational thinking abilities in both programming and non-programming contexts. In light of this, the TechCheck-K psychometric results, baseline (pre-test) results, and end-point (post-test) results with practical findings comprise the first two sections of the results discussion. The TACTIC-KIBO results are then examined in relation to the experimental group children's computational thinking skills related to programming. The findings of the background factors are then covered since this study also sought to ascertain how these factors affected the computational thinking skills of preschoolers. Lastly, suggestions for additional research are made along with the study's limitations and implications for educational practice.

5.1. The Psychometric Issues of the TechCheck-K Instrument

The results of the pilot study (n = 352) revealed that the TechCheck-K instrument has an acceptable level of internal consistency (α = .63) (Hinton et al., 2004). However, examining the literature the Cronbach's alpha value appeared to be lower than other computational assessment tools designed for preschool children (Roman-Gonzalez et al., 2017). For instance, Zapata-Caceres et al. (2020) designed an unplugged assessment of Beginners Computational Thinking Test (BCTt). Their instrument was constructed upon four dimensions namely sequences, simple loops,

nested loops, and conditionals. It was reported that BCTt has a high level of internal consistency (α = .82). Specifically, the number of dimensions they examined to test computational thinking skills may have affected the results they obtained. In particular, BCTt included four computational thinking dimensions, and the TechCheck-K contained six strong concepts that likely reduced internal consistency. Nevertheless, the internal consistency result found in the current study was consistent with the previous research which examined the psychometric issues of the TechCheck-K (Çetin et al., 2022; Relkin et al., 2020).

The TechCheck-K instrument has good to moderate psychometric properties with Turkish kindergarten students, according to the current study's results. Specifically, except for item 12, other items in the TechCheck-K were found to have acceptable levels of difficulty and discrimination parameters which were in line with Relkin et al.'s study (2020). This may reveal that the TechCheck-K instrument is useful in measuring and discriminating the computational thinking skills of the preschool children with different ability levels and most items have acceptable levels of difficulty. The item information and item characteristics curves also revealed that except for item 12, all items can discriminate children with different ability levels. However, the item information and item characteristics curves for item 12 showed that the probability of giving correct information increased for the low ability students, while the probability of giving correct information for item 12 decreased for the high ability students when it should be opposite. In fact, item 12, which measures the representation was found to receive one of the lowest mean scores in both the pilot and main studies of the current study. This item was also one of the items which was left blank by children and may indicate its difficulty for them. Consequently, it is possible that item 12 presented a challenge to the children in the current study, making it impossible for them to know the right response. Nevertheless, because of chance, the majority of the children, who had low ability levels, were able to correctly answer the item (Baker & Kim, 2017). In other words, this result could be attributed to the chance parameter; because low level ability children may have taken a chance and marked this question correctly even though they did not know the correct answer. Lastly, test information function of the

TechCheck-K indicated that the instrument is more powerful in discriminating preschool children with average and low ability levels which also aligns with the previous research (Relkin et al., 2020). In fact, Relkin et al. (2020) used TechCheck which was validated with the first and second grade children and items involved four options in contrast to the TechCheck-K. Nevertheless, their discrimination and difficulty parameters showed consistent results with the current study.

5.2. Investigating Preschool Children's Computational Thinking Skills through Unplugged TechCheck-K Assessment

Using the unplugged assessment tool, the computational thinking abilities of preschoolers in the control and experimental groups were examined in this section. For this reason, the TechCheck-K assessment results were used to discuss the pretest, posttest, and practical results.

5.2.1. The preschool children's baseline computational thinking skills

The results of this study indicated that the TechCheck-K pretest mean score was 8.60, regardless of the study group. Given that the mean score was only marginally lower than the median score of nine, it can be concluded that the preschoolers who took part in the study had a moderate level of computational thinking skills. In examining the related literature, previous findings appear to support the findings of this investigation (Relkin et al., 2021; Çetin et al., 2022; Yang et al., 2023). For instance, while Relkin et al. (2021) reported the first-grade children to get a mean score of 8.34 (M_{age} = 6.23 in the experimental group and M_{age} = 6.28 in the control group), Yang et al. (2023) reported the preschool children to have a mean score of 8.3, which were compatible with the findings of this research. However, Çetin et al. (2022), reported higher mean score for preschool children (M = 10.96 out of 15) who were older than the children participated in this study. Indeed, Çetin et al.'s (2022) results discrimination and difficulty parameters revealed that the TechCheck-K is an easy instrument for preschool children. However, in this study, the TechCheck-K assessment tool was found to be moderately difficult for preschool children.

According to Relkin et al. (2021) and other related literature, children's computational thinking abilities may vary depending on their grade or developmental stage as well as the instrument's degree of difficulty. Second-grade students may receive higher scores than first-grade children due to differences in cognitive development and maturation (Çetin et al., 2022; Relkin et al., 2021). Therefore, considering the fact that the children who participated in this study were at the pre-operational level of cognitive development, the higher mean scores could be due to either the cognitive developmental age of the children or the difficulty level of the instruments for children.

The moderate degree of computational thinking skills is actually not surprising given the domain-specific pretest results, as the current study also revealed that all children had the lowest mean scores in the domains of representation, modularity and algorithms. In light of the related literature, this may mean that the children who participated in this study could be challenged to understand that the symbols represent actions and are executed in specific behaviors (representation) and have difficulty sequencing multiple steps for a solution path (algorithms) by decomposing a large task (modularity) (Bers, 2021; Brennan & Resnick, 2012; Rijke et al., 2017). In the previous research, no study was found to investigate the existing computational thinking skills of preschool children by focusing all powerful ideas in detail. Nevertheless a number of studies revealed that representation, modularity, and algorithms are the fundamentals of computational thinking skills that facilitate the growth of other domains, are required for the acquisition of other fundamentals, and that proficiency in one area may be a prerequisite for success in another (Bers, 2021; Brennan & Resnick, 2012; Grover & Pea, 2014; Nouri et al., 2020; Rijke et al., 2017; Wing, 2008). Specifically, algorithms are the sequencing multiple steps for an algorithm design, or a solution path, and it involves sequencing, abstraction, and representation concepts (Bers, 2021; Wing, 2006; Yang et al., 2023). Algorithms are also constructed through algorithm design, problem specification, problem decomposition, and analytical thinking (Futschek, 2006; Kanaki & Kalogiannakis, 2022; Wong & Jiang, 2018). In fact, algorithms work with modularity, which includes problem decomposition and reported to be difficult to acquire in the early

years, especially in at the age of 5 (Cheng, 2012). Both the algorithms and modularity require the use of abstraction skills during joint processing, which is the ability of to recognize the irrelevant information while establishing a solution path by using sequencing skills (Rijke et al., 2017). This could indicate that abstraction skills and sequencing skills also improve algorithms and modularity to determine the relevant steps of a problem in a logical order (Bers, 2021). Throughout this process, children also benefit from representation because when they create an algorithm, they use symbols (i.e., codes) that have specific functions that lead to a particular action or goal (Bers, 2021; Zeng et al., 2023). This is because, representation is the recognition of the meaning and functions of symbols to create an algorithm that performs in specific behaviors (Zeng et al., 2023). Thus, it can be concluded that the domains of representation, modularity, and algorithms have complex structures, are challenging to develop because of their numerous contents, and require support from the cultivation of their constituent parts.

In addition, this study involved 62- to 72-month-old children who are at the preoperational stage and 72.01- to 80-month-old children who are in the process of transitioning to concrete operational stage (Piaget, 1954). This may show that there could be typical cognitive differences in complex abstract reasoning and logical reasoning among children (Piaget, 1951). Meanwhile, it can be argued that the low level of algorithms, modularity and representation domains would also be due to the developmental age of the children and the difficulty of acquiring abstraction and problem decomposition in the early years (Cheng, 2012; Rijke et al., 2017). Overall, in the context of the current study, it can be concluded that if the preschool children are supported by cultivating the components of algorithms, modularity, and representation, they may acquire higher-level computational thinking skills accordingly (Atmatzidou & Demetriadis, 2016; Bers, 2018; Selby & Woollard, 2013).

In addition, compared to other studies, some works have suggested that young learners' computational thinking can be developed through coding and programming activities using educational robotics and coding applications such as ScratchJr (Bers

et al., 2023; Relkin et al., 2021; de Ruiter & Bers, 2021; Saxena et al., 2020; Tsai et al., 2021; Yang et al., 2023). In the context of the current study, this may reveal that when preschoolers experience coding and programming activities with tangible and digital materials, they can improve their computational thinking skills and may have higher scores in each domain of computational thinking skills. Related research findings include the improvement of computational thinking abilities through at-home coding and programming experiences, as well as differences in the computational thinking abilities of students with and without coding experience (Rum & Ismail, 2017; Tsai et al., 2021). Considering that five children had no screen experience and most of the children had no coding or programming experience with digital games that could contribute to their computational thinking skills, the moderate levels of computational thinking skills are not surprising and consistent with the study by Relkin et al. (2021), who reported that most of the children in their study had little or no coding experience prior to treatment. Therefore, the moderate levels of computational thinking and low levels of core powerful ideas scores can be attributed to the lack of prior coding and programming experience of the preschool children in this study. In addition, children's computational thinking skills are also believed to be developed through tangible experiences with blocks or Legos (Dietz et al., 2019; Yang et al., 2022). It can be claimed that there is also a lack of tangible experiences at home, which may have prevented the preschool children from developing their computational thinking skills and powerful ideas, given that the majority of the children in this study do not play with blocks and Legos frequently. In addition, the highest domain-specific mean scores found in this study were control structures, debugging, and hardware/software. This may mean that the preschoolers in this study seem to be better at control structures, debugging, and hardware/software domains. For example, control structures involve understanding the behavior of the robot and the order in which a set of commands are initiated and executed (Bers, 2018). It also involves understanding repetitive commands, conditionals, and loops in a program. Therefore, it can be said that the preschool children in this study have the potential to understand and succeed in conditional tasks and to know which set of codes or commands will result in certain behaviors. In fact, young children are better at solving problems through trial and they learn better by engaging with concrete and tangible materials (Bers et al., 2023; Papadakis

et al., 2016; Saxena et al., 2020). The children's higher mean scores on the control structure questions may also be due to the concreteness of the tasks in the control structure questions. This is because, when answering control structure questions, children draw paths on the mazes, concretize their solution path on paper, and find the answer through trial-error. While debugging is a high-level skill that preschoolers find challenging to learn from an early age (Misirli & Komis, 2023), the preschoolers who took part in this study were able to identify an error in a problem and solve it correctly. As a matter of fact, they might also possess strong analytical thinking, problem-solving, or mathematical abilities (Cheng, 2012; Doleck et al., 2017; Fessakis et al., 2013; Israel & Lash, 2019). However, as the present study did not focus on these skills, their possible contribution could not be discussed. Subsequently, the higher scores found in the hardware and software domains indicate that the preschool children know that computers and robots can be programmed and that the human brain can work like a computer. It is therefore possible to argue that these children have a greater capacity for understanding ideas and information pertaining to hardware and software.

This research also revealed that there is a significant difference in the pretest computational thinking scores of the control and experimental group children in favor of the control group. More specifically, the control group had significantly higher mean scores than the experimental group. However, this result is due to chance rather than the design of the study as similarly reported in previous research (Relkin et al., 2021). Likewise, previous research examining pre-treatment group differences attributed baseline differences to chance (Munoz-Repiso & Caballero-Gonzalez, 2019; Relkin et al. 2021) or had equivalent groups prior to the educational robotics-based treatment (Bozkurt-Polat, 2023; Yang et al., 2023). For example, Relkin et al. (2021) and Munoz-Repiso & Caballero-Gonzalez (2019) reported a significant difference between the research groups. They reported that in their studies, the experimental group had higher mean scores than the control group. While Munoz-Repiso and Caballero-Gonzalez (2019) conducted extensive analyses to minimize error variance (U of Mann-Whitney, ROC curves, W of Wilcoxon test), Relkin et al. (2021) performed generalized linear mixed model and post hoc analyses

to investigate the relationship between computational thinking skills and the independent variables in order to control for the effect of educational robotics-based activities after treatment. In this study, however, the statistical equation method was utilized to account for group differences in this study because the groups had different baseline scores. By using analysis of covariance (ANCOVA), the groups were equalized by assigning the pretest scores as a covariate and the error variance was minimized to have a powerful look at the relationship between the dependent (posttest mean score) and independent variables. In fact, ANCOVA is reported to be more powerful when random assignment is not possible in experimental research, which was the case in this study, and there was a moderate relationship between pretest and posttest computational thinking skills, which is reported to increase the power of ANCOVA (Tabachnick & Fidell, 2013).

5.2.2. The preschool children's end-point computational thinking skills (via TechCheck-K) and practical results of the Coding as Literacy-KIBO (CAL-KIBO) Kindergarten curriculum

This part of the discussion focuses mainly on the effect of the intervention on the computational thinking skills of the experimental group by comparing it with the posttest scores of the control group and discussing the contribution of the intervention.

In the current study, major posttest findings revealed that the preschool children's computational thinking scores increased in the whole research group with moderate levels, but the experimental group children had significantly higher posttest scores than the control group children. This may show the positive effect of robotics-based coding implementation on preschool children's computational thinking skills. Similar to these findings, other studies have found that coding and programming-based implications contribute to preschoolers' computational thinking skills, but results have been mixed (Angeli & Valanides, 2020; Bers et al., 2023; Del Olmo Munoz et al., 2020; Munoz-Repiso & Caballero-Gonzalez, 2019; Relkin et al., 2021; Saxena et al., 2020; Yang et al., 2023; Yang et al., 2022). For example, Yang et al.

(2023) implemented three curricula that focused on developing computational thinking skills through plugged-in and unplugged computational activities in a culturally responsive story-based environment and an unplugged computational learning environment without storytelling. While one study group was exposed to story-inspired Matatalab robot programming (SIRP), others were exposed to story-inspired ScratchJr tablet programming (SITP) and unplugged computational thinking education (control group). Their results showed that the children in the robotics and story-based programming environment had the highest mean scores among others. Similarly, Yang et al. (2022) investigated the effects of robotics-based activities and block play on preschool children's computational thinking, sequencing, and self-regulation skills. Using the Matatalab coding set, the researchers mainly practiced correspondence, problem decomposition, algorithms, and pattern recognition while coding. Their findings revealed that the robotics-based programming group experienced greater gains in their ability to sequence but no significant difference was reported between the groups in self-regulation and computational thinking skills which was not consistent with this study. However, their result could be due to the sequencing activities that were mostly used in their intervention. Indeed, although sequencing is one of the components of algorithms and contributes to computational thinking (Bers, 2021), it may not be sufficient to develop computational thinking alone it is not a unified skill and has a complex structure (Yang et al., 2022). Therefore, the context and content of the intervention, which primarily concentrated on implementing sequencing skills, may be the reason for their inconsistent results with the current study.

In addition to this, Yang et al. (2022) implemented a six-week curriculum in their study that included fewer activities, which may have resulted in a non-significant outcome in the development of computational thinking skills. This is because the literature emphasizes that regardless of age, the duration and number of training sessions are important factors in the development of computational thinking skills (Angeli & Valanides, 2020; Bers et al., 2014). Considering that the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum lasted for eight weeks in this study and included more computational activities, the significant result found in this

study may also be owing to the appropriateness of the curriculum in terms of duration and number of training sessions.

In another incompatible research (Bers et al., 2023; Papadakis et al., 2016), Bers et al. (2023) reported that after implementing the Coding as Literacy-ScratchJr curriculum, children developed their coding skills through programming activities, but less change was observed in their computational thinking skills. Although not consistent with the results of this study, their findings may be due to the use of a different programming tool, ScratchJr, which was used to improve children's computational thinking skills through coding and programming. Indeed, it is reported that educational robotics would be more useful than the tangible materials with graphical interfaces to develop young children's computational thinking skills (Bers et al., 2023; Papadakis et al., 2016; Pugnali et al., 2017). This is because, graphical interfaces have more complex structure and have many icons, visuals, and programming blocks on the screen that may distract the children's attention on the use of relevant information that is necessary for their programs and may prevent them from using their abstraction skills efficiently.

In addition, the non-significant results reported by other researchers could be because of the decline in children's motivation and attention to participate in repetitive use of plugged-in activities. For instance, Del Olmo Munoz et al. (2023) conducted a two-stage quasi-experimental study with second grade children through Code.org courses. One group (the control group) participated in plugged-in activities during the first phase, while the other group (the experimental group) took part in unplugged activities. Both groups used the plugged-in activities in the second phase. Their results revealed that repetitive implementations through plugged-in tools in the experimental group decreased their motivation and attention scores in the posttest scores. Nonetheless, the preschoolers in the current study participated in math, reading, dance, and music in addition to using plugged-in tools and activities. This likely maintained their motivation and focus throughout the implementation and enhanced their computational thinking skills as a result.

In the current study, the findings also showed that there were differences in the powerful ideas of the experimental group depending on their percentage of development. Specifically, the post-hoc item analysis showed that algorithms, modularity, and control structures showed slightly less change, while the largest percentage of improvement was found in hardware/software, debugging, and representation, respectively. A review of previous research revealed mixed results (Atmatzidou & Demetriadis, 2016; Bozkurt-Polat, 2023; Kanaki & Kalogiannakis, 2022; Misirli & Komis, 2023; Rijke et al., 2018; Wong & Jiang, 2018; Yang et al., 2022). To begin with the control structures, it can be stated that the small improvement for the control structures may be due to the ceiling effect, since the children in the experimental group already had high pretest scores. Certainly, in this study, the difficulty parameters of the TechCheck-K showed that the items measuring control structures (items 14 and 15) were very easy for the kindergarten children, possibly leading to high pre- and post-test scores and therefore less change was observed in post-test scores which is consistent with the related research (Relkin et al., 2021).

To continue with debugging, although the syntactic knowledge and semantic knowledge of preschool children were not examined in this study, the greatest improvement in the debugging skills may be due to their mastery of these aspects (Misirli & Komis, 2023). For instance, Misirli and Komis (2023) used educational robotics to cultivate preschoolers' computational thinking skills and found that and the difference in children's debugging performance was related to their syntactic knowledge and semantic knowledge. Specifically, the children who were able to identify the grammatical errors in a program and the children who were aware of the logical errors, control commands, or spatial reasoning commands in a program were better at debugging.

When it comes to algorithms, the lower improvement could be related to the preschool children's lack of experience with programming or coding experience through educational robotics or coding tools (Wong &Jiang, 2018). For instance, after implementing series of activities, Wong, and Jiang (2018) found that the

activities significantly contributed to the elementary school children's algorithms related skills. In fact, the children who participated in Wong and Jiang's (2018) study received six hours of coding and programming training using Scratch prior to the intervention, which may have significantly increased their scores on algorithms. Considering that the children in this study did not have any programming or coding experience through educational robotics and coding/programming with coding tools, the slight improvement in the algorithms might be due to the lack of coding and programming experience. This comment is because the relevant literature supports the notion that having coding and programming experience significantly contributes to computational thinking skills (Rum & Ismail, 2017; Tsai et al., 2021).

When it comes to the lower improvement in the representation, the related literature revealed that the children's performance at the representation tasks are related to their success in recognizing patterns (Lüken & Kampmann, 2018). In addition, representation is reported to be better developed after experiencing algebraic thinking that involves the generalization and abstraction (Carraher & Schliemann, 2018). Consequently, a lesser improvement in the representation scores may have been seen in this study because algebraic activities were not used. Additionally, it has been observed that repetition loop tasks—that is, repetition patterns—help preschoolers develop representation effectively (Acosta et al., 2023). For example, in their longitudinal study, Acosta et al. (2023) examined children ages 3, 4, and 5 to determine their developmental progress and level of success in representation tasks. The researchers reported that while teaching representation, the repetition patterns were used with technological tools which gradually improved the children's success at representation tasks. Therefore, lower improvements in the representation domain would be attributed to the fact that repetition patterns or repetition loops were not mainly investigated in this study.

In addition to these studies, Relkin et al. (2021) conducted quasi-experimental research with first and second grade children to develop their computational thinking skills through the CAL-KIBO curriculum. Their results showed that the curriculum contributed most to the areas of algorithms, modularity, and representation, while

there was slightly less improvement in hardware/software, debugging, and control structures. Partially consistent with this study, the inconsistent results regarding the higher improvements in algorithms, modularity, and representation could be due to the cognitive developmental stage of the first and second grade children, which is the concrete operational stage. In other words, the preschoolers involved in the current study did not share the cognitive developmental characteristics of the children in Relkin et al.'s (2021) study. In fact, the children in the concrete operational stage differ from the pre-operational children in terms of mastery at decentering and reversibility, which have been reported to be linked to algorithms and modularity (problem decomposition) (Izu et al., 2015). More specifically, decentering involves focusing on more than one aspect of a process or problem at a time, while reversibility is the ability to recall the steps of a process and move back and forth between the steps in any order (Olteanu, 2015). In doing so, the child also benefits from decentering to see the parts and the whole of a process. This suggests that decentering and reversibility are particularly necessary when the child is engaged in a task that involves multiple steps (Olteanu, 2015). When considered in the context of this study, it can be argued that the children without the ability of reversibility and decentering could have had a difficulty in focusing on many aspects of a process, decomposing the problem to describe the necessary steps, and remembering the previous and next steps of a process (Rijke et el., 2017; Sigelman & Rider, 2012) which lead to the lower improvements in algorithms and modularity. Indeed, examining the items related to these domains (the items 6 and 7 for modularity, and the items 10 and 11 for algorithms) it seems that the children are supposed to mainly use decentering and reversibility thinking since they require the use of multiple skills. These items were also reported as challenging for the preschool children in the pilot study section of the TechCheck-K instrument. Thus, it can be inferred that the preschoolers in this study may have struggled with complex abstract thinking, decentering, and reversibility because they were in the preoperational stage or were in the process of moving to the concrete operational stage (Piaget, 1951; Piaget, 1954). This may have prevented them from going on to develop algorithms and modularity.

Lastly, when it comes to the control group children, in the present study, the preschool children were observed to have typical development in hardware/software (M = 1.6), modularity (M = 1.15) and debugging (M = 1.73) domains without taking any intervention. In addition, there was a decline in algorithms (M = 2.15), control structures (M = 1.6) and no change in representation domains (M = .45) which were also in parallel with the control group children's post-hoc results. Examining related research, the findings are partially in line with Relkin et al.'s (2022) study. Similarly, Relkin et al. (2021) reported that although there was no intervention in their control groups, the children showed higher improvement in debugging scores than the experimental group. The control group also showed typical development in modularity, representations, and algorithms respectively, but their scores were lower than those of the experimental group. In this sense, while the positive changes in hardware/software, modularity and debugging at the control group can be attributed to the maturation, testing and daily classroom activities (Relkin et al., 2021; Yang et al., 2022), the decline in algorithms and control structures could be because of their difficulty and complex structure for children (Janveau-Brennan & Markovits, 1999; Relkin, 2018). Consistent with the current study, Relkin et al. (2021) also showed that there was a decline in the control structures of first and second grade control group children after the treatment who did not receive any treatment. Indeed, control structures are hard concepts four preschool children as they involve conditionals (Janveau-Brennan & Markovits, 1999; Relkin, 2018; Rich et al., 2017; Strawhacker & Bers, 2015). For instance, control structure questions involve if conditional statements which are abstract for young children to understand especially when there are many conditions (Sullivan & Bers, 2018) and develop gradually between the ages of 6 to 12 (Barrouillet & Lecas, 1999). Therefore, decline in the control structures could be attributed to the difficulty of acquiring these concepts in the early years.

5.3. Investigating Preschool Children's Computational Thinking Skills through and Plugged-in TACTIC-KIBO Assessment

In this part, the programming-related computational thinking skills of the children in the experimental group were discussed using the results of the plugged-in TACTIC-KIBO assessment.

5.3.1. The effect of educational robotics-based coding and programming activities to the preschool children's programming related computational thinking skills.

The average time for the TACTIC-KIBO with three levels was 12 minutes, according to this study, which is in line with the results of Relkin's (2018) investigation. In fact, Relkin (2018) administered four levels of TACTIC-KIBO instrument and reported an average duration of 16 minutes. Thus, it may be stated that each level of the TACTIC-KIBO would be administered in almost 4 minutes which is suitable for preschool children, considering their short attention spans. In addition, TACTIC-KIBO was enjoyable for most of the children who participated in this study, as they willingly waited for the next question or task throughout the administration. Furthermore, the scoring of the instrument was easy to assess the children's performance throughout the programming tasks.

Major findings of this research also showed that following the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum, the majority of children in the experimental group (n = 31 out of 38) attained the programmer level. There were only four early programmers and three proto-programmers out of 38 children. This may reveal that CAL-KIBO kindergarten curriculum contribute to the programming-related computational thinking skills of preschool children and most children could reach the third level of programming which was consistent with the related literature (Relkin et al., 2021). When programming, programmer children can create purposeful programs to express themselves or solve a problem (Vizner, 2017) and involve in debugging through rewriting their program or finding a missing aspect in the program, such as a programming block, a sensor, or parameter (Relkin, 2018). The programmer children can also design programs including number parameters when adequate support is provided (Vizner, 2017). Finally, programmer children can engage in the design process through experiencing an iterative and open-ended learning process, identifying, and decomposing the problem to find efficient solution paths and testing, revising, and retesting the solution paths until the goal is achieved or problem is solved (Bers, 2021). Consequently, it can be concluded that the

children participating in this study have a potential to succeed at debugging, complex programming, educational robotics, and developing purposeful programs.

Although most preschool children reached programmer level in the current study, the most challenging powerful idea was the design process at the proto-programmer (level 1) and early programmer (level 2) levels. For instance, in level 2, a child is asked to move KIBO farther forward. When a child recognizes that it is a loop activity, s/he may use repeat blocks with number parameters. However, the majority of children were unaware that this task could be accomplished through creating a repeat loop and suggested scanning the forward block many times instead of programming with repeat blocks. In fact, even though certain children recommended using repeat blocks, the majority of them were unable to write a program that followed proper syntax. Similarly, in the programmer level debugging task, repeat loop task was prompted to the children and only five children (out of 38) created a syntactically and semantically correct algorithm involving repeat blocks and parameter. Therefore, as Misirli and Komis (2023) suggested the children's challenge in the design process could be due to their failure at repeat loops and low mastery at syntactic and semantic knowledge (Misirli & Komis, 2023). In addition, it can also be argued that the design process questions in TACTIC-KIBO mostly require the use of algorithms and modularity since the child should first recognize the necessary program, decompose this program to select the right blocks, and sequence them in a correct algorithm. This means that during this process, the preschoolers would benefit from their decentering and reversibility abilities, which are particularly necessary when the child is engaged in a task that involves multiple steps (Olteanu, 2015). Given that these abilities are hardly developed at the pre-operational level (Piaget, 1952, 1954) and are related to the algorithms and modularity (Izu, 2017), and given that the preschoolers' posttest scores in the TechCheck-K showed low development in these domains, it can be argued that the preschoolers had difficulty with design process tasks due to the children's lack of decentering and reversibility. Examining the repeat loops in detail, one reason for children's failure on repeat loop tasks would be the difficulty of acquiring these skills in the early years (Rich et al., 2017). This is because the repeat loops involve

repetitive programs and require more abstract thinking (Brennan & Resnick, 2012), which could be easily acquired when the children have the concrete operations (Relkin, 2018), but would be complex sequencing patterns for children (Rich et al., 2017). Given that older children are capable of performing more complex programs, this actually suggests that age plays a significant role in both basic and complex programming (Batı, 2022; Sullivan & Bers, 2016a). Thus, although the effect of age could not be examined in this study, the children's challenge in programming with repeat loops could be due to their age in months. For instance, in their study, Flannery and Bers (2013) found that older children performed more difficult tasks compared to preschool children and were better at using debugging skills when programming. Dietz et al. (2019) and Rijke et al. (2018) reported that older children are faster and outperform younger children in terms of problem decomposition and abstraction. Finally, it has also been reported that boys are better at complex programming concepts, such as repeat loops (Sullivan & Bers, 2018). However, due to the limited number of children who completed the repeat loops, the gender difference could not be examined in this study to determine whether the children's failure in repeat loops was owing to gender.

Afterwards, taking into account the quantity of right answers for every powerful idea, minor variations were found in the current study; as the tasks grew more difficult, the preschoolers' right answers progressively declined in terms of design process, hardware, algorithms/modularity, and representation-with the exception of software, control structures, and debugging-. Indeed, in the hardware domain of programmer level, children had trouble recalling the function of the green board (circuit board) in KIBO. Because the literature underlines the implementation of adequate number of activities while teaching computational concepts (Angeli & Valanides, 2020; Bers et al., 2014), this result could be attributed to the low number of activities that teach the function of green board in the CAL-KIBO kindergarten curriculum and is not established in other activities like other hardware and software issues of KIBO robotic. Considering other powerful ideas, many studies supported the notion that as programming tasks become more difficult, children need more adult support, show lower performance in sequencing, and make more errors as

parameters increase (Papadakis et al., 2016; Sullivan & Bers, 2016a, 2018; Sullivan et al., 2013). For instance, Sullivan et al. (2013) reported that although the preschool children in their study were exposed to a curriculum about engineers and robotics and engaged in a design process, they were challenged in the design process because they needed adult support while working with the robotics. Papadakis et al. (2016) conducted a small case study with 43 children and examined the impact of ScratchJr activities on the computational thinking skills by teaching basic programming concepts. Their results indicated that as the number of characters increased on the screen, they made more errors and confused the left and right turn blocks. In another study, Sullivan and Bers (2018) conducted a mixed study to investigate the children's success at Solve-Its scale which measures the sequencing skills in different levels. The researchers found that if-conditional tasks were the most complex activities for children and the children's sequencing scores decreased when the tasks became harder. Therefore, in light of the related literature, it may be stated that the decrease in correct responses for design process, algorithms/modularity, and representation could be due to the increase in the complexity and difficulty of programming concepts as the children progress to the next level of TACTIC-KIBO.

5.4. The effects of background factors to the preschool children's computational thinking skills

5.4.1. Age

After the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum, the ages in month comparisons revealed that there was not a significant difference between control and experimental group children in terms of the computational thinking skills. However, while the difference remained the same in the control group, it was almost eliminated in the posttest scores of the experimental group which is most likely due to the intervention. Indeed, this result could be attributed to the developmentally appropriateness of CAL-KIBO kindergarten curriculum since Batı (2022) reported that the educational robotics-based curriculums considering the children's working memory can eliminate the learning gaps among children. Indeed,

the CAL-KIBO kindergarten curriculum was designed according to the Curriculum Research Framework (Bers et al., 2023; Clements, 2007), which involves a 10-phase process to construct an evidence-based curriculum through many design principles, formative research with single and multiple classrooms, and summative research in a small and large context. Although the current research is only summative and was conducted in a small scale, the results support the previous research and the developmental and cultural appropriateness of the curriculum in the Turkish context as well.

Considering the related literature, the results were mixed (Atmatzidou & Demetriadis, 2016; Papadakis et al., 2016; Relkin, et al., 2021; Yildirim & Uluyol, 2022). For instance, Yang et al. (2023) and Relkin et al. (2020) reported that age was not a significant predictor of computational thinking skills of preschool children. However, partially inconsistent with the current study, Relkin et al. (2021) found that there was a significant difference between first and second grade children's computational thinking skills in baseline scores, but although both grades increased their mean scores at the posttest, first grade children improved their scores more than second grade children after treatment which supported the appropriateness of the CAL-KIBO curriculum for the younger children. Another inconsistent finding provided by Yildirim and Uluyol (2022) and the researchers reported that as the grade level increased, the first, second, third and fourth grade students' computational thinking skills increased significantly. In addition, Atmatzidou and Demetriadis (2016) investigated the computational thinking skills of junior and high vocational students and reported a significant difference in decomposition and modularity regarding age. When the contexts of these studies are taken into account, it appears that the researchers found non-significant results in the younger children and significant results in the older children. Therefore, it can be argued that this study is in line with the literature regarding the preschool children.

The results also showed that the group that improved their scores the most were the children in the youngest month group, the 60–72-month-old children. Therefore, it may be argued that CAL-KIBO kindergarten curriculum would be more effective to support younger children's computational thinking skills more. Indeed, the lower

developmental change observed in the 72.01-80-month-old preschoolers may be due to their higher pretest scores, which may have reduced the range of possible development in their computational thinking skills due to the ceiling effect (Relkin et al., 2020). Consistent with this finding, Relkin et al. (2020) stated that the TechCheck instrument, which was validated with first and second graders, had a less than optimal difficulty level for second graders, which could lead to higher pretest and posttest scores and a ceiling effect. In other words, a large percentage of second graders had the highest pretest scores on the TechCheck and they have already achieved the best score on the TechCheck which did not yield large changes in the posttest scores (Relkin et al., 2020), and could also be the case in the current study. In fact, this result could also be due to the limited power of the TechCheck-K assessment tool to discriminate between low, moderate, and high levels of children's computational thinking skills. This is because the item response results found in the current study revealed that the TechCheck-K is more powerful in discriminating Turkish preschool children at the moderate level of computational thinking. This argument actually stems from a similar finding by Çetin et al. (2022), who examined first-graders who had just started elementary school and found that the TechCheck-K test might be simple for young children.

5.4.2. Gender

In this research, no significant difference was found in the computational thinking skills of boys and girls in both the control and experimental groups after the intervention. This would mean that gender does not influence the development of computational thinking in preschool children. In fact, this finding is mostly consistent with previous studies that investigated the effect of gender on preschool children's computational thinking skills (Del Olmo-Munoz et al., 2020; Papadakis et al., 2016; Relkin et al, 2021; Sullivan & Bers, 2013; Yang et al., 2023), but the results were mixed in the related literature (Buitrago-Florez et al., 2017; Cooper, 2006; Master et al., 2017; Master et al., 2021; Master et al., 2023; Sullivan & Bers, 2016b). For example, Yang et al. (2023) conducted a quasi-experimental study with preschool children by implementing educational robotics-, ScratchJr-, and

unplugged-based computational activities that did not yield significant results on their computational thinking skills with respect to the participants' gender. Similarly, Papadakis et al (2016) reported that gender had no effect on the development of computational thinking skills in preschool children. However, inconsistent findings revealed that by the time children reach the first grade, gender differences in computational thinking skills may occur due to the level of interest, gender stereotypes, and individual motivation for computer science and coding and programming (Buitrago-Florez et al., 2017; Doube & Lang, 2012; Master et al., 2023; So et al., 2020). For example, Atmatzidou and Demetriadis (2016) worked with 15- and 18-year-old students and found that there was a significant difference between boys and girls at the initial assessment, but the difference disappeared after the intervention, revealing that girls may need more time than boys to cultivate their computational thinking skills. In addition, Del Olmo Munoz et al. (2020) studied second grade children and reported that while there was no significant difference in the motivation scores of boys and girls in the unplugged group, girls in a plugged-in study group reported less motivation to participate in the computing activities in the posttest scores. This suggests that as the computer science activities get harder, girls might find it harder to stay interested in them in the long run and might start to form negative gender stereotypes about coding and programming. (Del Olmo Munoz et al., 2020; Master et al., 2023). Considering the studies that are inconsistent with the current study, significant changes can be observed in children as they get older and establish gender stereotypes and less motivation to participate in the computer science related engagements, which was not the case in the current study and showed that gender equality can be achieved in early childhood (Yang et al., 2023).

In addition, Angeli and Valanides (2020) reported significant differences between boys and girls in the use of instructional approaches when programming with BeeBot robotics. Specifically, boys were found to benefit more from the individualistic, kinesthetic, and manipulative based coding cards, while girls were observed to benefit more from collaborative writing activities while coding and programming. Considering the CAL-KIBO kindergarten curriculum implemented in this study, there are computational activities that can be implemented individually or in small

and large groups, kinesthetic games that keep children active throughout the curriculum, and collaborative activities that allow them to learn from others. In addition, CAL-KIBO kindergarten curriculum includes writing and drawing activities through design journals at specific times in the curriculum. Therefore, it can be said that the curriculum implemented in this study addresses the developmental needs of preschool children in terms of both boys and girls. As a result, both girls and boys benefited from the computational learning processes and there was no significant difference in their posttest scores after implementation.

Lastly, this study examined the posttest mean scores and profile plots and found that while there was no statistically significant difference between the boys and girls in the experimental group, the girls in the group improved more than the boys at computational thinking. This outcome might have been brought about by the female researcher who taught the preschoolers the Coding as Literacy-KIBO kindergarten curriculum (Angeli & Valanides, 2020; Kalelioğlu, 2015; Sullivan & Bers, 2018). This is because, Sullivan and Bers (2018) reported that the gender of the educators would make a significant difference in the children's success in programming tasks and related skills accordingly. For example, although Sullivan and Bers (2018) implemented the same program with a male and female educator in separate classrooms with similar characteristics, boys had higher scores than girls in a program implemented by a male educator. Their results also showed that in the program conducted with a female educator, there was no difference between boys and girls, but girls' scores increased.

5.5. Educational Implications of the Study

In the current study, the pilot study results of the TechCheck-K revealed that the discrimination and difficulty indices, item characteristics curve, and item information curve for item 12 did not yield efficient results. However, the item was kept in the instrument to protect the internal consistency as when it was omitted from the TechCheck-K, the Cronbach's alpha value decreased. In this sense, Baker, and Kim (2017) suggest that the researchers who want to explore the probability of giving

correct answer to the items regarding chance parameter can increase the sample size. In order to fully comprehend the discrimination difficulty and chance parameter functions of item 12, it is recommended to increase the sample size. In addition to this, item 12 addresses the representation dimension and received one of the lowest mean scores in the pilot study and main study. The representation dimension is understanding that specific symbols execute in particular behaviors or codes when coding and programming (Acosta et al., 2023; Bers, 2021; Brennan & Resnick, 2012). Moreover, representation is related to the repetition loops, algebraic thinking, and patterns (Acosta et al., 2023) as discussed before. Therefore, this result may show that preschool children took part in this study could need support in learning core concepts of representation; but specific educational implications for the representation were provided in the next paragraphs when explaining the implications of pretest and posttest results.

In the current study, baseline results showed that the experimental and control group preschool children had moderate levels of computational thinking skills and low levels of representation, modularity, and algorithms. After the Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum, the experimental group children's overall computational thinking skills improved significantly but were still moderate due to the low levels of development observed in the post-hoc item analysis regarding the algorithms, modularity, and control structures -with a ceiling effect-. In addition, the representation domain was still one of the powerful ideas with the lowest mean scores after the treatment. These findings suggest that the CAL-KIBO kindergarten curriculum contributed to the computational thinking skills of the preschool children participated in this study, but they may be challenged in understanding that the symbols represent actions and are executed in specific behaviors (representation), and in having difficulty sequencing multiple steps for a solution path (algorithms) and decomposing a large task or problem (modularity) (Bers, 2021; Brennan & Resnick, 2012; Rijke et al., 2017). In fact, computational thinking is a multidimensional and interrelated construct (Selby & Woollard, 2013; Zhang & Nouri, 2019). Therefore, it is suggested to implement computational activities that involve multiple steps and multiple skills such as analytical thinking,

decomposition, abstraction, sequencing, and problem solving (Angeli & Valanides, 2020; Bers et al., 2023; Brusoni, 2007; Dietz et al., 2019; Olteanu, 2015). In addition, the preschool children in the current study were in the preoperational level and the intuitive thinking sub-stage, when they move toward logical thinking through logical reasoning and recognize the cause-and-effect relationships in the event (Batı, 2022; Piaget, 1951; Saxena et al., 2020). The related literature states that the preschool children with high-level computational thinking skills could be more successful in science, technology, engineering, and mathematics (STEM) related lessons when they enter to the elementary school, tend to pursue a STEM related careers in the future, address real-world problems, and find practical and creative solutions (Master, 2021; Master et al., 2023). It is therefore increasingly important to support the children's computational thinking skills by focusing on their core skills through developmentally appropriate practices rather than focusing on the overall computational thinking skills, especially since the children in this study were at the pre-operational level and had low levels of algorithms, modularity, and representation (Bati, 2022; Yang et al., 2022). Consequently, the following proposals for educational implications for developing the representation, algorithms, and modularity are made in light of the relevant literature.

To begin with representation, it refers to understanding that the symbols or actions can be represented through symbols (Bers, 2021; Brennan & Resnick, 2012). According to McGarvey (2012), when the children are first taught patterns, they would understand that each symbol in the pattern has a relationship and use symbols to create new patterns. Therefore, in order to develop preschool children's representation knowledge and skills, they would be exposed to the pattern activities with variety of symbols and pattern activities with different difficulty levels. In addition, in their longitudinal study, Acosta et al. (2023) implemented repetition pattern activities by using technological tools. After the longitudinal interventions and measurements, the researchers emphasized the significance of teaching repetition patterns while learning representation concepts. They argued that patterns ease children's transition to early algebraic thinking which has a relationship with computational thinking skills and could be developed through abstraction, and

generalization (Carpenter et al., 2005). The researchers also underlined to learn mathematics and patterns in early years to develop sequences, functions and relationships that transit simple representation to multiple representations. Therefore, in the light of literature to improve the children's performance in representation tasks, algebraic thinking, abstraction, and generalization could be embedded into the computational activities. To illustrate, the children can combine different geometric shapes that represent fruits (Leung et al., 2023), such as circle can represent an apple while square represent a banana. In addition, the combination of representations may become more difficult as children master simple representations.

Algorithms refer to the series of sequential steps that are followed to achieve a goal (Bers, 2021). It also involves sequencing skills, pattern recognition, algorithm design and abstraction skills (Bers, 2021; Futschek, 2006; Kanaki & Kalogiannakis, 2022; Wong & Jiang, 2018). Computer programming is not required to develop algorithms, but when programming and coding, algorithm-related abilities are mainly used. In this sense, related literature reveals that algorithms can be taught through real-life problems, including abstraction and problem decomposition activities (Kanaki & Kalogiannakis, 2022; Lee & Junoh, 2019). For instance, unplugged coding activities could be implemented to develop algorithm design (Lee & Junoh, 2019). Children can first create the algorithms for daily routines like washing their hands and brushing their teeth for this purpose. This way, before learning a new concept through digital and tangible tools, they can make a connection between coding and their prior, familiar knowledge (Lee & Junoh, 2019; Saxena et al., 2020). In addition, when identifying the series of steps for an algorithm and finding the most practical solution/algorithm, they may realize that there are unnecessary steps in their algorithm which prevent them from practical solution. To address this problem, children can work with abstraction. In fact, the abstraction works in tandem with modularity to facilitate problem decomposition and identify irrelevant data or algorithmic errors (Bers, 2021; Rijke et al., 2017; Wing, 2008). Afterwards, plugged-in activities can be implemented through educational robotics, and screen-based programming which have been found to contribute to algorithm-related skills (Cheng et al., 2017; Figueiredo et al., 2021; Wong & Jiang, 2018).

In order to develop algorithms, literacy activities would be implemented through integrating the story books into the coding activities (Bers et al., 2023; Lee & Junoh, 2019). This approach is emphasized in the CAL-KIBO kindergarten curriculum which was implemented in this study through plugged-in tools. However, the coding with literacy activities can also be implemented through paper-pencil activities. Specifically, after reading a storybook, a teacher can ask children to create a program through ScratchJr or educational robotics which describes the beginning, the middle and ending part of the story. The children may also sequence the events in the storybook through picture-sequencing or may draw three scenes of story that represent the beginning, the middle and ending parts (Bers, 2019). Importantly, sequential words, directional words and arrows and coding sheets can be used to improve the algorithm-related skills since they are used in pattern recognition and design, sequencing and problem-solving in a systematic way (Lee & Junoh, 2019).

Additionally, modularity involves problem decomposition and composition, and one of the problem-solving strategies since requires analyzing the problem in a modular way (Wimsatt, 2007). It is noted that modularity can be acquired from the early years by embedding abstraction and decomposition to the activities (Lister, 2011). For instance, the preschool children can be provided two or three problem solution ways and can be asked to select the most practical solution way from the three options (Dietz et al., 2019). However, given preschoolers' limited working memory, consideration should be given to the number of possible solutions (Bati, 2022). To develop modularity, young children can also draw or list the instructions needed to complete a larger project (Bers, 2021). In this process, they create their own symbols, concrete the abstract ideas in their minds, using and developing not only abstraction but also their representation skills. In addition, they can learn about real-world engineering projects that are useful for problem solving and planning skills (Dietz et al., 2019). For instance, children can be given a model and then asked to construct the model through blocks by determining the constraints of the model and working on the blocks until they can copy the model block (Dietz et al., 2019). Given the nature of problem-solving process, modularity, and decomposition; multiple iterations can also be followed through the engineering design process (Bers, 2021). This learning process allows them to set their investigation goals and solution paths,

test, and evaluate them, debug their errors, and implement alternative solutions that they may have interpreted during the problem decomposition process. Through planned problem decomposition procedures, the children may discover the advantages of the decomposition and may use this problem-solving strategy in their daily lives which is likely to contribute greatly to their computational thinking skills.

Continuing with TACTIC-KIBO results and implications, the results of this study revealed that the most challenging powerful idea for preschool children was the design process. Considering the context of design process, the use of debugging strategies, sequencing, algorithms construction, and problem decomposition gain importance (Angeli, 2022; Bers, 2021; Dietz et al., 2019; Flannery & Bers, 2013; Rijke et al., 2017; Saxena et al., 2020; Wong & Jiang, 2018). Actually, it may be stated that since design process is an iterative and open-ended process, it may require the use of all powerful ideas. For example, after setting the solution steps of a particular problem, the child apply trial and error, may revise the solution steps, and retest the solution path until the design produce the intended results (Bers, 2021). During this process, debugging is mainly used. Therefore, to increase preschool children's success in the design process, debugging skills could be cultivated through plugged-in or unplugged activities. For instance, educational robotics can be used to develop debugging skills because they are concrete and tangible materials that make trials, errors and learning visible and have also been reported to develop abstraction skills in the preschool children (Misirli & Komis, 2023). Indeed, Pugnali et al. (2017) and Bers et al. (2023) also reported that educational robotics are useful to concretize the abstract concepts in children's minds and help to actualize their purpose. However, when children engage in graphical interface programming, which is screen-based, and make errors, they tend to create a completely new program or an algorithm (Gomes et al., 2018) or switching to the alternative task or goal, which may lead to higher performance in simpler tasks compared to complex ones (Bers et al., 2014). Therefore, it is suggested that coding toys and educational robotics can be used to improve the children's debugging skills and strategies. Debugging skills can also be developed through daily activities including problem-solving tasks since they are instrumental to problem-solving skills (Blank & Lynch, 2018; Fessakis et al.,

2013). These tasks could be implemented either individually or in small groups. This would allow children to try out different debugging strategies on their own or with their peers.

According to the current study's TACTIC-KIBO findings, most children could not successfully complete the repeat loops by using repeat blocks and number parameters and could not create syntactically and semantically correct programs. In this sense, plugged-in, or unplugged coding activities can be implemented to develop the children's syntactic and semantic knowledge (Komis & Misirli, 2023). For instance, in their study Komis and Misirli (2023) investigated the development of debugging skills in preschool children and examined the syntactic and semantic errors that children make while programming. Their results revealed that the preschool children's syntactic knowledge and semantic knowledge have an impact on the preschool children's debugging skills. Consequently, teaching children semantic knowledge—which includes their comprehension of commands like directional language—is the first step toward helping them succeed in repetition loops. The children can choose the direction arrows to represent the codes in an algorithm by comprehending the directional language in the commands. Then, the children can be taught how to sequence these direction arrows and create a syntactically that is grammatically correct program. In another study, repetition loops were reported to be best with patterns (Acosta et al., 2023). Therefore, the preschool children can also be involved in pattern activities which are mostly implemented through mathematics. However, the patterns can also be taught through gameplays or literacy activities which focus on sequencing the pattern in story books and sequencing the beginning, middle and ending parts of the story.

The results of this study also revealed that there was no significant difference in the computational thinking scores of boys and girls after implementation of the CAL-KIBO kindergarten curriculum, which is consistent with the previous research (Atmatzidou & Demetriadis, 2016; Relkin et al., 2021; Yang et al., 2023). This result may reveal the developmental appropriateness of the CAL-KIBO kindergarten curriculum, since developmental and child-friendly suitable practices can eliminate potential learning gaps between the boys and girls and do not create significant

differences for a particular gender group (Bati, 2022; Del Olmo Munoz et al., 2020). For instance, Angeli and Valanides (2016) examined the interaction effect of scaffolding type when coding and programming with BeeBot robotics. The researchers reported that girls benefited more from adult support when writing and sequencing the commands of a program or an algorithm while boys benefited more from kinesthetic, spatially oriented, and individualistic learning processes while coding and programming. Therefore, it is recommended that the computational experiences include both types of scaffolding or instructional techniques to develop the computational thinking skills of boys and girls equally. Otherwise, teachers who apply one type of instruction or scaffolding technique may prevent gender equality in the development of computational thinking skills. In another study, Atmatzidou and Demetriadis (2016) found that the girls need more time than boys when cultivating their computational thinking skills involving modularity and decomposition. Therefore, it is recommended to provide an efficient amount of training sessions when designing and implementing computational activities for young children. Moreover, although gender stereotypes are believed to occur by the time children reach first grade (Master et al., 2021), teachers may transmit their own biases or stereotypes to the children which may favor boys and cause girls to develop negative attitudes toward STEM, coding, and programming related engagements (Master et al., 2023). Therefore, early childhood teachers are suggested to be careful about their attitudes and beliefs toward the boys' and girls' success at these fields which could prevent them from developing negative gender stereotypes. Lastly, educational implications regarding gender can also be established for the parents so that they do not show gender stereotypes to their children. This is because, the children are gender detectives realizing the preferences and expectations for their gender and negative stereotypes deter girls more than boys in STEM, coding, and programming related fields (Halim & Ruble, 2010; Master, 2021). In this sense, parent education seminars concerning this issue can be provided.

Finally, the findings of the current study also revealed that there was no significant difference in the computational thinking scores of age in month groups after the implementation of CAL-KIBO kindergarten curriculum; but, 60-to-72-month-old

children in the experimental group had more improvement in their computational thinking skills, which supported the notion that the CAL-KIBO kindergarten curriculum is more suitable for developing the computational thinking skills of younger children. The related literature recommends that the age factor is especially important when designing the programming activities for preschool children due to their limited working memory (Batı, 2022; Sullivan & Bers, 2016a). For instance, while older children can recall more instructions and set more complex algorithms when programming, younger children have a difficulty in remembering commands with more than five instructions (Sullivan & Bers, 2016a). Therefore, when implementing computational activities in preschool settings, more complex programming activities that address the development of computational thinking skills in older children can be provided (Rijke et al., 2018). Apart from the CAL-KIBO kindergarten curriculum, additional activities can be designed from the simple to complex contexts to enable the flow of older preschool children and to challenge their cognitive load (Rijke et al., 2018). This is because, when the preschool children feel challenged and capable, flow can occur (Csikszentmihalyi, 1975), and not only the younger children, but also the older ones in month groups can show significant improvements in their computational thinking skills.

5.6. Limitations of the Study and Recommendations for Future Research

To begin with, this study involved 58 preschool children attending one of the state kindergartens in Aksaray city center. In the same context, the effect of Coding as Literacy-KIBO (CAL-KIBO) kindergarten curriculum could be implemented with preschool children by increasing the sample size. For this purpose, it is suggested to further study in different regions of Türkiye with various social, economic, and cultural aspects.

Secondly, this study investigated the psychometric issues of the TechCheck-K instrument through item response theory and due to the sample size (n = 352), a two-parameter logistic model could have been investigated to identify the instrument's discrimination and difficulty parameters. Therefore, future research may focus on

increasing the sample size and controlling the chance parameter of the TechCheck-K items. By doing this, the probability of correctly answering the items could also be detected.

In this study, although the overall computational thinking skills and powerful ideas were examined in detail, lower improvements were observed in particular domains after the CAL-KIBO kindergarten curriculum, making it difficult to reason for the lower improvements. Several experimental and case research were found to focus on one or two dimensions of computational thinking in their studies (Misirli & Komis, 2023; Rijke et al., 2017; Sullivan & Bers, 2016a). These studies could have provided in-depth information about the developmental process and progress in the preschool children after the intervention. For instance, Misirli and Komis (2023) investigated the debugging skills while Rijke et al. (2017) examined the problem decomposition and abstraction in detail. Therefore, further studies can investigate one or two dimensions of computational thinking and the progress in the dimensions of computational thinking which would also ease to explain the developmental differences in children and provide efficient educational implications.

One of the study's limitations was the small sample sizes for children without screen time and children without experience playing digital games. As a result, it was not possible to conduct comparative analyses following the intervention in order to look at how preschoolers' experiences with digital games varied in terms of computational thinking development. The number of children was not equally distributed when it came to their screen time and experiences with digital games due to the small sample size. Even though it was the goal, this study was unable to examine the possible impact of screen time spent playing digital games on the preschool children's computational thinking skills. Consequently, after plugged-in or unplugged computational activities are implemented, future research may concentrate on the variations in preschoolers' computational thinking abilities between those who have never used a screen and those who have, as well as how much time they spend playing digital games on screens. By doing this, the combined

effect of computational practices and screen experience and time with digital games could be examined on the development of computational thinking.

Another limitation of this study was related to the methodology since the experimental group children's programming related computational thinking skills were examined through the TACTIC-KIBO instrument but could not be compared with the control group since they did not have coding and programming experience prior to administration. Therefore, it is suggested to implement CAL-KIBO kindergarten curriculum and a different educational robotics curriculum to compare the effect of the CAL-KIBO kindergarten curriculum to the programming related computational thinking skills with the other computational thinking curriculum. By doing this, comparative analysis could be performed to indicate its significant or non-significant effect on the programming related computational thinking skills. In addition, semi-structured interviews could be conducted with classroom teachers to examine the CAL-KIBO kindergarten curriculum's effect or reflection on the preschool children's daily engagements.

In this study, the effect of gender and age in months on the posttest computational thinking skills of preschool children were examined using the TechCheck-K instrument. However, due to the small sample size, the children's programming related computational thinking skills could not be investigated through the TACTIC-KIBO instrument in the same context. As a result, it is recommended to increase the sample size, which can comprise children of various ages and programming levels. This would allow the computational thinking skills related to programming in children to be evaluated in relation to their age in months. In addition, since related literature mainly highlights the effect of gender on the development of complex programming tasks, such as repetition loops, the future research could also focus on the programming related developmental differences in the computational thinking skills of boys and girls through longitudinal studies.

This study was also limited by the limited number of TACTIC-KIBO tasks for seven powerful ideas. When designing the TACTIC-KIBO, the researchers considered the

short attention span of preschool children. Therefore, they presented one question per powerful idea and programming level in the TACTIC-KIBO. By doing this, the TACTIC-KIBO provided overall information about the programming related computational thinking skills of preschool children. In the same context, it may be suggested to expand the tasks in the instrument to address hardware, software, algorithms, modularity, debugging, control structures, representation, and design process in detail. The new tasks may also solely focus on the design process by embedding other powerful ideas. By doing this, more information about children's programming related computational thinking development can be gathered.

Finally, this research only involved preschool children and their teachers, and the participants were not asked about their roles or beliefs about the coding education in early childhood. For instance, in their large-scale study Koshy et al. (2021) examined the primary school teachers' self-confidence to teach computer science and found that they had lower level of confidence in teaching computer science issues compared to middle and high school teachers. Therefore, it is also possible for the early childhood teachers in Türkiye to have low level of self-efficacy, knowledge, or self-confidence to teach computer science related issues that may prevent them contributing to the computational thinking scores of children through plugged-in and unplugged coding activities. Therefore, a further study is suggested to investigate their understanding, knowledge, attitudes, or teaching efficacy beliefs regarding coding education. In addition, future research can also focus on professional development of early childhood teachers for coding education in the early years.

REFERENCES

Acosta, Y., Alsina, Á., & Pincheira, N. (2023). Computational thinking and repetition patterns in early childhood education: Longitudinal analysis of representation and justification. *Education and Information Technologies*, 1-26.

Akyol Altun, C. (2018). Okul öncesi öğretim programına algoritma ve kodlama eğitimi entegrasyonunun öğrencilerin problem çözme becerisine etkisi. [Yüksek lisans tezi, Ankara Üniversitesi].

Alimisis, D., & Kynigos, C. (2009). Constructionism and robotics in education. *Teacher education on robotic-enhanced constructivist pedagogical methods*, 11-26.

Alsancak-Sırakaya, D. (2020). Investigating computational thinking skills based on different variables and determining the predictor variables. *Participatory Educational Research*, *7*(2), 102-114.

American Academy of Pediatrics Council on Communications and Media (APA). (2016). Media and Young Minds. *Pediatrics, 138* (5). Doi: https://doi.org/10.1542/peds.2016-2591

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction, 31,* 100329.

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and

scaffolding strategy. *Computers in human behavior, 105*. https://doi.org/10.1016/j.chb.2019.03.018

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, *19*(3), 47-57.

Arfé, B., Vardanega, T., Montuori, C., & Lavanga, M. (2019). Coding in primary grades boosts children's executive functions. *Frontiers in psychology*, *10*, 2713. https://doi.org/10.3389/fpsyg.2019.02713

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661-670. https://doi.org/10.1016/j.robot.2015.10.008

Baker, F. B., & Kim, S. H. (2017). *The basics of item response theory using R* (pp. 17-34). New York: Springer.

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. http://dx.doi.org/10.1145/1929887.1929905.

Barrouillet, P., & Lecas, J. F. (1999). Mental models in conditional reasoning and working memory. *Thinking & Reasoning,* 5(4), 289–302.

Batı, K. (2021). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies, 1-24.* https://doi.org/10.1007/s10639-021-10700-2

Becker, K. (2002). Constructivism and the use of technology. *The Technology Teacher*, 1.

Bellg, A. J., Borrelli, B., Resnick, B., Hecht, J., Minicucci, D. S., Ory, M., ... & Czajkowski, S. (2004). Enhancing treatment fidelity in health behavior change studies: best practices and recommendations from the NIH Behavior Change Consortium. *Health psychology*, *23*(5), 443.

Bentler, P. M., & Chou, C. P. (1987). Practical issues in structural modeling. *Sociological methods & research, 16*(1), 78-117.

Bers, M. U. (2008). *Blocks, robots and computers: Learning about technology in early childhood*. New York: Teacher's College Press.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157. https://doi.org/10.1016/j.compedu.2013.10.020

Bers, M. U. (2018). Coding, playgrounds, and literacy in early childhood education: the development of KIBO robotics and ScratchJr. In *Global Engineering Education Conference (EDUCON)*, 2018 IEEE (pp. 2094-2102).

Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, *6*(4), 499-528. https://doi.org/10.1007/s40692-019-00147-3

Bers, M. U., González-González, C., & Armas–Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education, 138,* 130-145. https://doi.org/10.1016/j.compedu.2019.04.013

Bers, M. U. (2020). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge.

Bers, M. U., Strawhacker, A., & Sullivan, A. (2022). *The state of the field of computational thinking in early childhood education*. OECD Education Working Papers, No. 274, OECD Publishing, Paris, https://doi.org/10.1787/3354387a-en.

Bers, M. U., Blake-West, J., Kapoor, M. G., Levinson, T., Relkin, E., Unahalekhaka, A., & Yang, Z. (2023). Coding as another language: Research-based curriculum for early childhood computer science. *Early Childhood Research Quarterly*, *64*, 394-404.

Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller- Ricci, M., & Rumble, M. (2012). Defining twenty-first century skills. In *Assessment and teaching of 21st century skills* (pp. 17-66). Springer.10.1007/978-94-007-2324-5_2

Brackmann, C., Barone, D., Casali, A., Boucinha, R., & Muñoz-Hernandez, S. (2016). Computational thinking: Panorama of the Americas. In *2016 international symposium on computers in education* (pp. 1–6).

Borrelli, B., Sepinwall, D., Ernst, D., Bellg, A. J., Czajkowski, S., Breger, R., ... & Orwig, D. (2005). A new tool to assess treatment fidelity and evaluation of treatment fidelity across 10 years of health behavior research. *Journal of consulting and clinical psychology*, *73*(5), 852.

Bozkurt-Polat, E. (2023). The Effect of Tangible Robotic Coding Education Program on the Computational Thinking Skills and Learning Behavior of Preschool Children. [Doctoral Dissertation, Gazi University].

Brennan, K., & Resnick, M. (2013). Stories from the scratch community: Connecting with ideas, interests, and people. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 463-464).

Bruni, F., & Nisdeo, M. (2017). Educational robots and children's imagery: a preliminary investigation in the first year of primary school. *Research on Education and Media, 9*(1), 37-44.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834-860. https://doi.org/10.3102/0034654317710096

Burgio, L., Lichstein, K. L., Nichols, L., Czaja, S., Gallagher-Thompson, D., Bourgeois, M., ... & REACH Investigators. (2001). Judging outcomes in psychosocial interventions for dementia caregivers: The problem of treatment implementation. *The Gerontologist*, *41*(4), 481-489.

Caeli, E. N., & Bundsgaard, J. (2020). Computational thinking in compulsory education: A survey study on initiatives and conceptions. Educational Technology *Research and Development, 68*(1), 551–573. https://doi.org/10.1007/s11423-019-09694-z

Canbeldek, M. (2020). Erken çocukluk eğitiminde üreten çocuklar kodlama ve robotik eğitim programının etkilerinin incelenmesi (Yayımlanmamış doktora tezi). Pamukkale Üniversitesi.

Canbeldek, M., & Işıkoğlu, N. (2023). Exploring the effects of "productive children: coding and robotics education program" in early childhood education. *Education and Information Technologies*, *28*(3), 3359-3379.

Carpenter, T. P., Levi, L., Franke, M. L., & Zeringue, J. K. (2005). Algebra in elementary school: Developing relational thinking. *Zentralblatt Für Didaktik Der Mathematik, 37*(1), 53–59. https:// doi. org/10. 1007/ BF026 55897

Catana Kuleli, S. (2018). Evaluation of pre-service teachers' readiness for online learning and computational thinking skills. [Yüksek Lisans Tezi, Düzce Üniversitesi].

Chapman, M. (1988). *Constructive evolution: Origins and development of Piaget's thought.* New York: Cambridge University Press.

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. *International Journal of STEM Education*, *10*(1), 47.

Cheng, Z. J. (2012). Teaching young children decomposition strategies to solve addition problems: An experimental study. *The Journal of Mathematical Behavior*, *31*(1), 29-47.

Clements, D.H. (1991). Enhancement of creativity in computer environments. *American Educational Research Journal, 28,* 173-187.

Clements, D. H. (2007). Curriculum research: Toward a framework for research-based curricula. *Journal for research in mathematics education*, *38*(1), 35-70.

Cooper, J. (2006). The digital divide: The special case of gender. Journal of Computer Assisted Learning, 22 (5), 320–334. https://doi.org/10.1111/j.1365-2729.2006.00185.x

Durlak, J. A., & DuPre, E. P. (2008). Implementation matters: a review of research on the influence of implementation on program outcomes and the factors affecting implementation. *American Journal of Community Psychology, 41,* 327–350.

Cortesa, C. S., Jones, J. D., Hager, G., Khudanpur, S., Landau, B., & Shelton, A. (2018). Constraints and development in children's block construction. In *40th Annual Meeting of the Cognitive Science Society: Changing Minds, CogSci 2018* (pp. 244-249). The Cognitive Science Society.

Cowan, N., & Alloway, T. (2008). Development of working memory in childhood. *The development of memory in infancy and childhood*, 303.

Critten, V., Hagon, H., & Messer, D. (2021). Can pre-school children learn programming and coding through guided play activities? A case study in computational thinking. *Early Childhood Education Journal.* https://doi.org/10.1007/s10643-021-01236-8

Csikszentmihalyi, M. (1975). Beyond Boredom and Anxiety: Experiencing Flow in Work and Play. *The Jossey-Bass Behavioral Science Series*, 231. doi:10.2307/2065805

Cuneo, D. O. (1986). Young Children and Turtle Graphics Programming: Generating and Debugging Simple Turtle Programs. *Annual meeting of the American Educational Research Association.*

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs.cmu.edu/~ CompThink/resources/TheLinkWing.pdf*.

Çakır, R., Korkmaz, Ö., İdil, Ö., & Uğur Erdoğmuş, F. (2021). The effect of robotic coding education on preschoolers' problem solving and creative thinking skills. *Thinking Skills and Creativity, 40, 1*-13. https://doi.org/10.1016/j.tsc.2021.100812

Çetin, İ., Şendurur, P., & Tarık, O. T. U. (2022). Tech Check İsimli Bilgi İşlemsel Düşünme Testlerinin Türkçeye Uyarlanması. *Journal of Instructional Technologies and Teacher Education*, *11*(2), 16-27. https://doi.org/10.51960/jitte.1102904

Davies, A., Fidler, D., & Gorbis, M. (2011). Future work skills 2020. *Institute for the Future for University of Phoenix Research Institute*, *540*.

De Ruiter, L. E., & Bers, M. U. (2022). The Coding Stages Assessment: development and validation of an instrument for assessing young children's

proficiency in the ScratchJr programming language. *Computer Science Education*, *32*(4), 388-417.

Değirmenci, Ş. (2022). Investigation of the Effects of the Coding Education Program on Cognitive Flexibility and Computational Thinking Skills of Children Aged 5. (Doctoral Dissertation, Marmara University].

del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education*, *150*, 103832. https://doi.org/10.1016/j.compedu.2020.103832

Demircan, H. Ö. (2022). "How am I supposed to do this on my own?": A case study on perspectives of preschool teachers regarding integrative STEM practices. *Journal of Early Childhood Research, 20*(1), 93–112. doi:10.1177/1476718X211052749

Detrich, R. (1999). Increasing treatment fidelity by matching interventions to contextual variables within the educational setting. *School Psychology Review*, *28*(4).

DevTech Research Group. (2018). A KIBO Coding Curriculum for Emergent Readers. Retrieved from https://sites.tufts.edu/codingasliteracy/files/2019/06/CAL-KIBO-Emergent-Final-Curriculum.pdf

Dietz, G., Landay, J. A., & Gweon, H. (2019). Building blocks of computational thinking: Young children's developing capacities for problem decomposition. In *CogSci* (pp. 1647-1653).

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: Exploring the relationship between computational thinking skills and

academic performance. *Journal of Computers in Education, 4*(4), 355–369. https://doi.org/10.1007/s40692-017-0090-9

Doubé, W., & Lang, C. (2012). Gender and stereotypes in motivation to study computer programming for careers in multimedia. *Computer Science Education*, *22*(1), 63-78.

Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education. Innovations in Practice*, *13*, 153.

Elkin, M., Sullivan, A., & Bers, M. U. (2016). Programming with the KIBO robotics kit in preschool classrooms. *Computers in the Schools, 33*(3), 169-186. https://doi.org/10.1080/07380569.2016.1216251

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63,* 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Figueiredo, M., Gomes, C. A., Amante, S., Gomes, H., Alves, V., Duarte, R. P., & Rego, B. (2021, September). Play, Algorithmic Thinking and Early Childhood Education: Challenges in the Portuguese Context. In *2021 International Symposium on Computers in Education (SIIE)* (pp. 1-4). IEEE.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. Proceedings of the 12th International Conference on Interaction Design and Children, 1-10. https://doi.org/10.1145/2485760.2485785

Flannery, L. P., & Bers, M. U. (2013). Let's dance the "robot hokey-pokey!" children's programming approaches and achievement throughout early

cognitive development. *Journal of research on technology in education*, *46*(1), 81-101.

Fraenkel, J. R., Wallen, N. E. & Hyun, (2012). *How to design and evaluate research in education.* (8th ed.). New York, NY: McGraw- Hill, Inc.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In *International conference on informatics in secondary schools-evolution and perspectives* (pp. 159-168). Berlin, Heidelberg: Springer Berlin Heidelberg.

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *Informatics in Schools. Contributing to 21st Century Education: 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011, Bratislava, Slovakia, October 26-29, 2011. Proceedings 5* (pp. 155-164). Springer Berlin Heidelberg.

Gerosa, A., Koleszar, V., Tejera, G., Gómez-Sena, L., & Carboni, A. (2022). Educational robotics intervention to foster computational thinking in preschoolers: Effects of children's task engagement. *Frontiers in Psychology*, *13*, 904761. https://doi.org/10.3389/fpsyg.2022.904761

Gomes, T. C. S., Falcão, T. P., & Tedesco, P. C. D. A. R. (2018). Exploring an approach based on digital games for teaching programming concepts to young children. In *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems*. Porto Alegre:SBC. 10.5753/ihc.2018.4227.

Goodgame, C. (2018, March). Beebots and Tiny Tots. In *Society for Information Technology & Teacher Education International Conference* (pp. 1179-1183). Association for the Advancement of Computing in Education (AACE).

Grover, S., Pea, R., & Cooper, S. (2015). *"Systems of assessments" for deeper learning of computational thinking in K-12.* Paper presented at the Annual Meeting of the American Educational Research Association, Chicago, United States of Amerika.

Gonzalez, Y. A. C., & Munoz-Repiso, A. G. (2017). Educational robotics for the formation of programming skills and computational thinking in childish. *In 2017 international symposium on computers in education* (pp. 1–5).

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Guggemos, J. (2021). On the predictors of computational thinking and its growth at the high-school level. *Computers & Education*, *161*, 104060.

Halim, M. L., & Ruble, D. (2010). Gender identity and stereotyping in early and middle childhood. In J. C. Chrisler, & D. R. McCreary (Eds.), *Handbook of gender research in psychology* (pp. 495–525). Springer. 10.1007/978-1-4419-1465-1_24.

Harwell, M. R., & Janosky, J. E. (1991). An empirical study of the effects of small datasets and varying prior variances on item parameter estimation in BILOG. *Applied Psychological Measurement, 15*(3), 279–291. http://dx.doi.org/10.1177/014662169101500308

Hassenfeld, Z. R., Govind, M., De Ruiter, L. E., & Bers, M. U. (2020). If you can program you can write: Learning introductory programming across literacy levels. *Journal of Information Technology Education*, *19*.

Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12

education. In *Frontiers in Education Conference* (FIE) 2016 (pp. 1–9). Pennsylvania, US: IEEE.

Hennessey, M. L., & Rumrill Jr, P. D. (2003). Treatment fidelity in rehabilitation research. *Journal of Vocational Rehabilitation*, *19*(3), 123-126.

Highfield, K. (2010). *Robotic toys as a catalyst for mathematical problem solving. Australian Primary Mathematics Classroom, 15*(2), 22–27.

Hinton, P., Brownlow, C., Mcmurray, I., & Cozens, B. (2004). *SPSS explained. Abingdon-on-Thames: Taylor & Francis.* https://doi.org/10.4324/9780203642597.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Hsu, Y. C., Irie, N. R., & Ching, Y. H. (2019). Computational thinking educational policy initiatives (CTEPI) across the globe. *TechTrends*, *63*, 260-270.

Hunsaker, E., West, R.E. Designing Computational Thinking and Coding Badges for Early Childhood Educators. *TechTrends, 64*, 7–16 (2020). https://doi.org/10.1007/s11528-019-00420-3

Israel-Fishelson, R., Hershkovitz, A., Egu´ıluz, A., Garaizar, P., & Guenaga, M. (2021). The associations between computational thinking and creativity: The role of personal characteristics. *Journal of Educational Computing Research, 58*(8), 1415–1447. https://doi.org/10.1177/0735633120940954

Izu, C., Pope, C., & Weerasinghe, A. (2017). On the ability to reason about program behaviour: A think-aloud study. In *Proceedings of the 2017 ACM Conference*

on Innovation and Technology in Computer Science Education (pp. 305-310).

Janveau-Brennan, G., & Markovits, H. (1999). The development of reasoning with causal conditionals. *Developmental Psychology, 35*(4), 904-911.

Kafai, Y. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM, 59*(8), 26–27. https://doi.org/10.1145/2955114

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior*, *52*, 200-210.

Kanaki, K., & Kalogiannakis, M. (2022). Assessing algorithmic thinking skills in relation to age in early childhood STEM education. *Education Sciences*, *12*(6), 380.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245-255. https://doi.org/10.1007/s10643-012-0554-5

Kirçali, A. Ç., & Özdener, N. (2022). A comparison of plugged and unplugged tools in teaching algorithms at the K-12 level for computational thinking skills. *Technology, Knowledge, and Learning*, 1-29.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72,* 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Korucu, A. T., Gencturk, A. T. & Gundogdu, M. M. (2017). Examination of the Computational Thinking Skills of Students. *Journal of Learning and Teaching*

in Digital Age, 2 (1), 11-19. Retrieved from https://dergipark.org.tr/en/pub/joltida/issue/55466/760079

Kyllonen, P. C. (2012). Measurement of 21st century skills within the common core state standards. In *Invitational Research Symposium on Technology Enhanced Assessments* (pp. 7-8).

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, 159-174.

Lavigne, H., Presser, A. L., Rosenfeld, D., Wolsky, M., & Andrews. J. (2020). Creating a preschool computational thinking learning blueprint to guide the development of learning resources for young children. *Connected Science Learning, 2* (2). https:// www.nsta. org/ connected-science-learning/connected-science-learningapril-june- 2020/ creating- preschool.

Lee, J., Joswick, C., & Pole, K. (2023). Classroom play and activities to support computational thinking development in early childhood. *Early Childhood Education Journal*, *51*(3), 457-468.

Lee, J., & Junoh, J. (2019). Implementing unplugged coding activities in early childhood classrooms. *Early Childhood Education Journal*, *47*, 709-716.

Leung, S. K., Wu, J., & Li, J. W. (2023). Children's knowledge construction of computational thinking in a play-based classroom. *Early Child Development and Care*, 1-22. https://doi.org/10.1080/03004430.2023.2299405

Lichstein, K. L., Riedel, B. W., & Grieve, R. (1994). Fair tests of clinical trials: A treatment implementation model. *Advances in Behaviour Research and Therapy*, *16*(1), 1-29.

Lim, R. G., & Drasgow, F. (1990). Evaluation of two methods for estimating item response theory parameters when assessing differential item functioning.

Journal of Applied Psychology, 75(2), 164–174. http://dx.doi.org/10.1037/0021-9010.75.2.164

Lister, R. (2011). Concrete and other neo-piagetian forms of reasoning in the novice programmer. *Conferences in Research and Practice in Information Technology Series, 114,* 9–18.

Lord, F.M. (1952). The relationship of the reliability of multiple-choice test to the distribution of item difficulties. *Psychometrika, 18,* 181-194.

Lord, F. M. (1968). An analysis of the verbal scholastic aptitude test using Birnbaum's three-parameter logistic model. *Educational and Psychological Measurement, 28*(4), 989–1020. http://dx.doi.org/10.1177/001316446802800401

Lüken, M. M., & Kampmann, R. (2018). The Influence of Fostering Children's Patterning Abilities on Their Arithmetic Skills in Grade 1. In I. Elia, J. Mulligan, A. Anderson, A. Baccaglini-Frank, & C. Benz (Eds.), *Contemporary Research and Perspectives on Early Childhood Mathematics Education* (pp. 55–66). Springer International Publishing. https:// doi. org/ 10. 1007/ 978-3-319- 73432-3_4

Marini, Z., & Case, R. (1994). The development of abstract reasoning about the physical and social world. *Child Development*, *65*(1), 147-159.

Martins, E. C., da Silva, L. G. Z., & de Almeida Neris, V. P. (2023). Systematic mapping of computational thinking in preschool children. *International Journal of Child-Computer Interaction*, 100566.

Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *Journal*

of experimental child psychology, *160*, 92-106. https://doi.org/10.1016/j.jecp.2017.03.013

Master, A., Meltzoff, A. N., & Cheryan, S. (2021). Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *Proceedings of the National Academy of Sciences*, *118*(48), e2100030118.

Master, A., Tang, D., Forsythe, D., Alexander, T. M., Cheryan, S., & Meltzoff, A. N. (2023). Gender equity and motivational readiness for computational thinking in early childhood. *Early Childhood Research Quarterly*, *64*, 242-254. https://doi.org/10.1016/j.ecresq.2023.03.004

McGarvey, L. M. (2012). Is it a pattern? *Teaching Children Mathematics, 19*(9), 564–571. https:// doi. org/10. 5951/ teacchilmath. 19.9.0564

Melro, A., Tarling, G., Fujita, T., & Kleine Staarman, J. (2023). What else can be learned when coding? A configurative literature review of learning opportunities through computational thinking. *Journal of Educational Computing Research*, *61*(4), 901-924.

Metz, S. S. (2007). *Attracting the engineering of 2020 today.* In R. Burke, M. Mattis, & E. Elgar (Eds.), Women and minorities in science, technology, engineering and mathematics: Upping the numbers (pp. 184–209). Edward Elgar Publishing.

Metzger, R. (2004). *Debugging by thinking: A multidisciplinary approach*. Burlington, MA: Elsevier Digital Press.

Metin, S. (2020). Activity-based unplugged coding during the preschool period. *International Journal of Technology and Design Education.* https:// doi. org/ 10. 1007/ s10798- 020- 09616-8

Misirli, A., & Komis, V. (2023). Computational thinking in early childhood education: The impact of programming a tangible robot on developing debugging knowledge. *Early Childhood Research Quarterly*, *65*, 139-158.

Monteiro, R., Fernandes, S., & Rocha, N. (2022). What do preschool teachers and parents think about the influence of screen-time exposure on children's development? Challenges and opportunities. *Education Sciences*, *12*(1), 52.

MTA Cooperative Group (1999). Moderators and mediators of treat- ment response for children with attention-deficit/ hyperactiv- ity disorder. *Archives of General Psychiatry, 56, 1088.*

Munoz-Repiso, A. G. V., & Caballero-Gonzalez, Y. A. (2019). Robotics to develop computational thinking in early childhood education. *Media Education Research Journal, 27*(59), 63–72.

Murcia, K. J., & Tang, K. S. (2019). Exploring the multimodality of young children's coding. *Australian Educational Computing, 34*(1) http://journal.acce.edu.au/index.php/AEC/article/view/208.

Müller, U., Ten Eycke, K., & Baker, L. (2015). Piaget's theory of intelligence. *Handbook of intelligence: Evolutionary theory, historical perspective, and current concepts*, 137-151.

National Academies of Sciences, Engineering and Medicine [NASEM]. (2000). *How People Learn*, Vol.11. Washington, DC: National Academy Press.

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1–17.

Olteanu, L. L. (2015). Psychological aspects of teaching in primary school. *Journal of Preschool and Elementary School Education*, *7*(1), 53-67.

Önal, N. T., & Ardıç, M. (2020). Okul öncesi öğrencileri için makey makey ile bir fen etkinliği tasarımı. *İnsan ve Toplum Bilimleri Araştırmaları Dergisi, 9*(3), 2225-2236. https://doi.org/10.21923/jesd.672232

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organisation*, *10*(3), 187-202. https://doi.org/10.1504/IJMLO.2016.077867

Papert, S. (1980). Computers for children. *Mindstorms: Children, computers, and powerful ideas*, 3-18.

Papert, S. (1991). *Situating constructionism.* In I. Harel & S. Papert (Eds.). Constructionism (pp. 1-11). Ablex publishing Corporation.

Papert, S. (1996). Computers in the classroom: Agents of change. *The washington post education review*, *27*.

Peng, J., Wang, M., & Sampson, D. (2017). Visualizing the complex process for deep learning with an authentic programming project. Journal of Educational Technology & Society, 20(4), 275–287.

Piaget, J. (1951). Principal factors determining intellectual evolution from childhood to adult life. In *Organization and pathology of thought: Selected sources* (pp. 154-175). Columbia University Press.

Piaget, J. (1954). The development of causality (M. Cook, Trans.). In J. Piaget & M. Cook (Trans.), *The construction of reality in the child* (pp. 219–319). Basic Books. https://doi.org/10.1037/11168-003

Piaget, J. (1965). *The moral judgment of the child*. New York:Free Press

Piaget, J. (1977). *The development of thought: Equilibration of cognitive structures. (Trans A. Rosin)*. Viking.

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2016). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education*, *26*, 489-504.

Relkin, E. (2018). *Assessing Young Children's Computational Thinking Abilities*. (Master thesis). Tufts University, Eliot-Pearson Department of Child Study & Human Development, Massachusetts.

Relkin, E., de Ruiter, L., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*, *29*(4), 482-498. https://doi.org/10.1007/s10956-020-09831-x

Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021). Learning to code and the acquisition of computational thinking by young children. *Computers & education*, *169*, 104222. https://doi.org/10.1016/j.compedu.2021.104222

Relkin, E., & Bers, M. U. (2019). Designing an assessment of computational thinking abilities for young children. In L. E. Cohen & S. Waite-Stupiansky (Eds.), *STEM for early childhood learners: how science, technology, engineering, and mathematics strengthen learning*. New York: Routledge. https://doi.org/10.4324/9780429453755-5.

Relkin, E., & Bers, M. (2021, April). Techcheck-k: A measure of computational thinking for kindergarten children. *In 2021 IEEE global engineering education conference (EDUCON)* (pp. 1696-1702). IEEE.

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. In *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition* (pp. 1-6).

Rich K., Pokimica J., Wherfel Q., Strickland C., & Moran C. (2017). *Building Mathematics + Computational Thinking Trajectories from Existing Literature.* American Educational Research Association.

Rijke, W. J., Bollen, L., Eysink, T. H., & Tolboom, J. L. (2018). Computational thinking in primary school: An examination of abstraction and decomposition in different age groups. *Informatics in education*, *17*(1), 77-92.

Robledo-Castro, C., Castillo-Ossa, L. F., & Hederich-Martínez, C. (2023). Effects of a computational thinking intervention program on executive functions in children aged 10 to 11. *International Journal of Child-Computer Interaction*, *35*, 100563.

Rum, S. N. M., & Ismail, M. A. (2017). Metocognitive support accelerates computer assisted learning for novice programmers. *Journal of Educational Technology & Society, 20*(3), 170–181.

Rusk, N., Resnick, M., Berg, R. et al. New Pathways into Robotics: Strategies for Broadening Participation. *Journal of Science Education and Technology, 17,* 59–69 (2008). https://doi.org/10.1007/s10956-007-9082-2

Sahin, A., & Anil, D. (2017). The effects of test length and sample size on item parameters in item response theory. *Educational Sciences: Theory & Practice, 17,* 321–335. http://dx.doi.org/10.12738/estp.2017.1.0270

Saritepeci, M. (2020). Developing computational thinking skills of high school students: Design-based learning activities and programming tasks. *The Asia-Pacific Education Researcher*, *29*(1), 35-54.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education. *The Asia-Pacific Education Researcher, 29*(1), 55-66. https://doi.org/10.1007/s40299-019-00478-w

Schwarzer, R. (2011). *Everything you wanted to know about the General Self-Efficacy Scale but were afraid to ask*. Retrieved September 15, 2023, from http://userpage.fu-berlin.de/health/selfscal.htm.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, 59-66.

Selby, C. C., & Woollard, J. (2013). *Computational thinking: The developing definition.* Paper Presented at the 18th annual conference on innovation and Technology in Computer Science Education, Canterbury. Retreived from https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Sigelman, C. K., & Rider, E. A. (2012). *Life-span human development* (7th ed.). Belmont, CA: Wadsworth, Cengage Learning.

Smith, S. W., Daunic, A. P., & Taylor, G. G. (2007). Treatment fidelity in applied educational research: Expanding the adoption and application of measures to ensure evidence-based practice. *Education and treatment of children*, *30*(4), 121-134.

So, HJ., Kim, D. & Ryoo, D. Trajectories of Developing Computational Thinking

Competencies: Case Portraits of Korean Gifted Girls. *Asia-Pacific Edu Res* 29, 85–100 (2020). https://doi.org/10.1007/s40299-019-00459-z

Soleimani, A., Herro, D., & Green, K. E. (2019). CyberPLAYce—A tangible, interactive learning tool fostering children's computational thinking through storytelling. *International Journal of Child-Computer Interaction*, *20*, 9-23.

Somuncu, B., & Aslan, D. (2021). Effect of coding activities on preschool children's mathematical reasoning skills. *Education and Information Technologies, 1-14.* https://doi.org/10.1007/s10639-021- 10618-9

Smith, A., & Johnson, B. (2020). *Fundamentals of Software Development*. Publisher ABC.

Stone, C. A. (1992). Recovery of marginal maximum likelihood estimates in the two-parameter logistic response model: An evaluation of Multilog. *Applied Psychological Measurement, 16*(1), 1–16. http://dx.doi.org/10.1177/014662169201600101

Strawhacker, A., & Bers, M. U. (2015). "I want my robot to look for food": Comparing kindergartner's programming comprehension using tangible, graphic, and hybrid user interfaces. *International Journal of Technology and Design Education, 25*(3), 293-319. http://doi.org/10.1007/s10798-014-9287-7

Strawhacker, A., & Bers, M. U. (2018). Promoting positive technological development in a Kindergarten makerspace: A qualitative case study. *European Journal of STEM Education*, *3*(3), 9.

Su, J., & Yang, W. (2023). A systematic review of integrating computational thinking in early childhood education. *Computers and Education Open*, 100122. https://doi.org/10.1016/j.caeo.2023.100122

Sullivan, A., & Bers, M. U. (2013). Gender differences in kindergarteners' robotics and programming achievement. *International journal of technology and design education*, *23*, 691-702. https://doi.org/10.1007/s10798-012-9210-z

Sullivan, A., & Bers, M. U. (2016a). Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *International Journal of Technology and Design Education,* 26, 3–20. https://doi.org/10.1007/s10798-015-9304-5

Sullivan, A., & Bers, M. U. (2016b). Girls, boys, and bots: gender differences in young children's performance on robotics and programming tasks. *Journal of Information Technology Education: Innovations in Practice, 15*, 145–165. https://doi.org/10.28945/3547.

Sullivan, A., & Bers, M. U. (2018). The impact of teacher gender on girls' performance on programming tasks in early elementary school. *Journal of Information Technology Education. Innovations in Practice*, *17*, 153.

Sullivan, A., & Bers, M. U. (2019). Investigating the use of robotics to increase girls' interest in engineering during early elementary school. *International Journal of Technology and Design Education*, *29*, 1033-1051. https://doi.org/10.1007/s10798-018-9483-y

Tabachnick, B. G., & Fidell, L. S. (2013). *Using multivariate statistics* (Vol. 6, pp. 497-516). Boston, MA: Pearson.

Tabesh, Y. (2017). Computational thinking: A 21st century skill. *Olympiads in Informatics*, *11*(2), 65-70.

Tedre, M., & Denning, P. J. (2021). Computational thinking: A professional and historical perspective. In *Computational Thinking in Education* (pp. 1-17). Routledge.

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, 104083.

Tsai, M. J., Liang, J. C., Lee, S. W. Y., & Hsu, C. Y. (2022). Structural validation for the developmental model of computational thinking. *Journal of Educational Computing Research*, *60*(1), 56-73. https://doi.org/10.1177/07356331211017794

Vizner, M. Z. (2017). *Big Robots for Little Kids: Investigating the Role of Scale in Early Childhood Robotics Kits.* (Published master's thesis). Tufts University, Eliot-Pearson Department of Child Study & Human Development, Massachusettss.

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

Wang, D., Wang, T., & Liu, Z. (2014). A tangible programming tool for children to cultivate computational thinking. *The Scientific World Journal*. https://doi.org/10.1155/2014/428080

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of science education and technology*, *25*, 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wimsatt, W. C. (2007). *Re-engineering philosophy for limited beings: Piecewise approximations to reality*. Harvard University Press.

Wing, J. M. (2014). Computational thinking benefits society. *40th anniversary blog of social issues in computing*, 26.

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. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). Research notebook: computational thinking -what and why? The Link Magazine, 20-23. Retrieved January 10, 2022, from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-andwhy

Wong, G. K., & Jiang, S. (2018, December). Computational thinking education for children: Algorithmic thinking and debugging. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 328-334). IEEE.

Yang, W., Ng, D. T. K., & Gao, H. (2022). Robot programming versus block play in early childhood education: Effects on computational thinking, sequencing ability, and self-regulation. *British Journal of Educational Technology*, *53*(6), 1817-1841.

Yang, W., Ng, D. T. K., & Su, J. (2023). The impact of story-inspired programming on preschool children's computational thinking: A multi-group experiment. *Thinking Skills and Creativity*, *47*. https://doi.org/10.1016/j.tsc.2022.101218

Yıldırım, E., & Uluyol, Ç. (2023). Developing Computational Thinking Scale for Primary School Students and Examining Students' Thinking Levels According to Different Variables. *Journal of Learning and Teaching in Digital Age*, *8*(1), 113-123. https://doi.org/10.53850/joltida.1176173

Yılmaz, F. (2022). 60-72 Month – Old Children's Computational Thinking Skills: An Adaptation Study of the TechCheck-K Scale. [Master's Thesis, Cukurova University]. https://hdl.handle.net/20.500.14114/4801

Zakaria, N. I., & Iksan, Z. H. (2020). Computational thinking among high school students. *Universal Journal of Educational Research*, *8*(11).

Zapata-Cáceres, M., Martín-Barroso, E., & Román-González, M. (2020, April). Computational thinking test for beginners: Design and content validation. In *2020 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1905-1914). IEEE.

Zeng, Y., Yang, W., & Bautista, A. (2023). Computational thinking in early childhood education: Reviewing the literature and redeveloping the three-dimensional framework. *Educational Research Review*, 100520.

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, *141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zurnacı, B., & Turan, Z. (2022). Türkiye'de okul öncesinde kodlama eğitimine ilişkin yapılan çalışmaların incelenmesi. *Kocaeli Üniversitesi Eğitim Dergisi*, *5*(1), 258-286.

APPENDICES

A. DEMOGRAPHIC INFORMATION FORM

Kişisel Bilgi Formu

1) Yaşınız:
2) Cinsiyetiniz: ( ) Kadın ( ) Erkek
3) En son mezun olduğunuz eğitim/öğretim kurumu
1. ilköğretim 2. lise 3. iki yıllık yüksek okul 4. lisans. 5. yüksek lisans 6. Doktora

4) Mesleğiniz
1. Çalışmıyorum 2. Devlet kurumu 3. Özel kurum 4. Serbest

5) Gelir düzeyiniz Alt ( ) Orta ( ) Üst ( )
6) Ailedeki çocuk sayısı:…… Formu doldurduğunuz kaçıncı çocuğunuz.....

7) Çocuğunuzla nasıl vakit geçiriyorsunuz? Aşağıdaki seçenekleri 5'ten 1'e doğru verilen sıklık derecelerine göre puanlayınız.

(5= Neredeyse her gün, 4= 2-3 günde bir, 3= Haftada 1 gün, 2=10 günde 1, 1=Ayda 1-2 kez)

…….. Hikâye kitapları okuyorum.
…….. Birlikte televizyon izliyoruz.
…….. Oyun oynuyorum.
…….. Telefonda/tablette birlikte oyun oynuyoruz.
…….. Çocuğum telefonda veya tablette oyun oynarken yanında oturuyorum.
…….. Çocuğumla bilgisayarda birlikte oyun oynuyoruz.
…….. Birlikte market alışverişine çıkıyoruz.
…….. Alışveriş merkezlerinde geziyoruz.
…….. Parka gidiyoruz.
…….. Aile büyüklerini ziyarete gidiyoruz.
…….. Diğer (Lütfen belirtiniz: .....)

Yukarıda belirtilen seçeneklerin dışında faaliyetleriniz varsa lütfen aşağıdaki boşluğa sıklık derecesini (5'ten 1'e kadar bir sıklık) belirterek yazınız.

Etkinlik Türü ve sıklığı
(5= Neredeyse her gün, 4= 2-3 günde bir, 3= Haftada 1 gün, 2=10 günde 1, 1=Ayda 1-2 kez)

- Etkinlik: ..... Sıklığı: .....
- Etkinlik: ..... Sıklığı: .....
- Etkinlik: ..... Sıklığı: .....

8) Çocuğunuza ne tür oyuncaklar alırsınız? Aşağıda verilen seçeneklerden ilk üçünü, 1'den 3'e doğru sıralayarak işaretleyiniz

……oyuncak almam, evdeki materyallerle veya doğal materyallerle kendi oyuncaklarını yapıp oynamasını isterim.

……Araba ……Bebek ……Mutfak oyuncakları ……Ahşap veya plastik bloklar

……Bilgisayar/tablet oyunları ……Kinetik kum ……Oyun hamuru ……Legolar

……Puzzle. ……eğitici oyuncaklar ..... Diğer (varsa belirtiniz)

9) Evinizde lego oyuncaklar var mı? ☐ Evet ☐ Hayır
Cevabınız evet ise, çocuğunuz ne kadar sıklıkla legolarla oynuyor?
……Her gün
……Haftada 1-2 defa
……Eğer yanında akranları var ise onlarla birlikte oynuyor.
……Oynamıyor.

10) Evinizde ahşap veya plastik bloklar var mı? ☐ Evet ☐ Hayır
Cevabınız evet ise, çocuğunuz ne kadar sıklıkla ahşap veya plastik bloklarla oynuyor?
……Her gün okuldan dönünce 1-2 saat oynar.
……Haftada 1-2 defa
……Eğer yanında akranları var ise onlarla birlikte oynuyor.
……Oynamıyor.

11) Çocuğunuzla oyun oynadığınız zamanlarda genellikle oyunu siz mi çocuğunuz mu yönlendirir/yönetir?

1. Çoğunlukla ben yönetirim.
2. Çoğunlukla çocuğum yönetir.
3. Çocuğumla pek oyun oynamam.
4. Çocuğum benimle oyun oynamak istemez.
5. Çocuğum her zaman tek başına oynamayı tercih eder.

☐

12) Çocuğunuz cep telefonu veya tablet ile oyun oynar mı? Evet Hayır

Cevabınız evet ise, günde kaç saat oyun oynar?
1) 1 saat 2) 2 saat 3) 2 saatten fazla 4) 3-4 saat 5) Diğer ……

Cevabınız evet ise, çoğunlukla ne tür oyunlar oynar?
1. Rekabet/yarış oyunları
2. Sayı sayma, sıralama, eşleştirme gibi görevleri olan matematiksel oyunlar
3. Kodlama oyunları
4. ……
5… ….

13) ScratchJR uygulamasını biliyor musunuz? Evet Hayır

1. Evet biliyorum, çocuğum ScratchJR ile sıkça vakit geçiriyor.
2. Evet duydum ancak ne olduğunu bilmiyorum
3. Evet duydum ancak çocuğumun oynaması için henüz tablete/bilgisayara yüklemedim.
4. Evet biliyorum çocuğumun oynaması için tablete/bilgisayara yükledim ancak ilgisini çekmedi.
5. Hayır bilmiyorum/daha önce hiç duymamıştım.

B. TURKISH SUMMARY/TÜRKÇE ÖZET

GİRİŞ

Bilgi işlemsel düşünme ilk olarak Papert (1980, 1996) tarafından ortaya atılmıştır. Papert, bilgi işlemsel düşünmenin odağının makineler veya bilgisayarlar gibi düşünmek değil, önceden belirlenmiş hedeflere ulaşmak ve hata ayıklamak için zihni bilgisayara benzer şekilde işlemek olduğunu belirtmiştir. Papert, programlama ve bilgi işlemsel düşünme becerileri arasındaki ilişkiye değinmiş ve süreç odaklı düşünmenin tüm akademik alanlarda programlama yoluyla geliştirilebileceğini savunmuştur (Nouri vd., 2020). Papert, Piaget ile uzun süre çalıştığı için onun eğitim felsefesinden ve özellikle küçük çocuklarda öğrenme ve öğretme anlayışından etkilenmiştir (Relkin, 2018). Papert'in öğrenme teorisine göre; çocuklar özgürce keşfetmelerine ve gerçek dünya problemleri için çözümler üretmelerine olanak tanıyan somut etkinlikler deneyimlediğinde daha verimli bir öğrenme gerçekleştirirler (1980). Papert'e göre "anlamlı bir ürün oluşturmak" da önemlidir (Tabesh, 2017). Nitekim yaparak öğrenmek gibi anlamlı bir ürün inşa etmek de çocukların kendini ifade etmesini ve bu yolla öğrenmesini sağlar. Papert'i takiben, bilgi işlemsel düşünme becerileri Wing tarafından "bir problemin formüle edilmesi ve çözüm (ler)inin bir bilgisayarın- insan veya makine- etkili bir şekilde yürütebileceği şekilde ifade edilmesiyle ilgili düşünce süreçleri" olarak yeniden yorumlanmıştır (2014, s. 5). Wing'e göre (2014) insanlar problemlerini bilgi işlemsel düşünme stratejileri aracılığıyla verimli bir biçimde çözebilirler. Bunlara ek olarak, Wing'e göre bilgi işlemsel düşünme hem matematiksel hem de mühendislik fikirlerini de kapsamaktadır.

Bilgi işlemsel düşünmenin eğitim uygulamalarıyla bütünleştirilmesi ilk, orta ve yüksek öğretimde daha fazla ilgi görmesine rağmen, son yıllarda erken yıllardan itibaren geliştirilmesi gereken önemli bir beceri olarak ele alınmaya başlanmıştır (Bati, 2022; Bers et al., 2022; Saxena et al., 2020; Su & Yang, 2023). Bu amaçla

erken çocukluk dönemi, bilgi işlemsel düşünme becerilerini geliştiren uygulamalarınhayata geçirilmesi açısından kritik önem taşımaktadır (Flannery & Bers, 2013). Nitekim Buitrago-Florez ve arkadaşları (2017), öğrenilmesi zor olsa da bilişsel gelişimi güçlendirmek için okul öncesi çocukların erken yaşlardan itibaren bilgi işlemsel düşünme becerilerini edinmeleri gerektiğini öne sürmüştür. Bu sayede, okul öncesi dönemdeki çocukları işlem öncesi evrede ve ikinci alt evre olan sezgisel düşünme alt evresinde olduklarından, mantıksal akıl yürütme yoluyla mantıksal düşünmeye kolayca geçebilirler (Piaget, 1951). Örneğin, okul öncesi dönemindeki bir çocuk bir durumu rasyonel inançlarla açıklayabilir, sebep-sonuç ilişkilerini fark edebilir, tahminlerde bulunabilir ve hedefe ulaşmak için adım adım bir yaklaşım izleyebilir. Bu aynı zamanda, çocukların eylemlerinin nedenlerini ve sonuçlarını ve neden bazı eylemlerin başarılı olurken diğerlerinin olmadığını anlamalarını sağlayan semiyotik işlevden kaynaklanmaktadır (Piaget 1954; Saxena vd., 2020). Bu dönemdeki çocuklar bilgi işlemsel öğrenme deneyimleri sayesinde somut işlemlerden soyut işlemlere daha kolay geçebilir (Su ve Yang, 2023). Örneğin, ayrıştırma yöntemini kullanarak problem çözme becerilerini kullanırken, karmaşık problemleri daha küçük adımlara bölebilirler. Ayrıca, sıralama ve algoritmik düşünme gibi bilgi işlemsel düşünmenin temel içeriklerini edinerek, problem çözme becerilerini geliştirebilirler (Angeli, 2022). Nitekim Piaget'nin (1954) bilişsel gelişim kuramına göre, okul öncesi dönemindeki çocuklarda örüntü tanıma ve algoritma tasarlama becerileri kolayca geliştirilebilir ve bunlar da bilgi işlemsel düşünmenin temel kavramlarıdır (Saxena vd., 2020).

Gelişimsel olarak uygun uygulamalar ve öğrenme araçları sayesinde, çocuklar bilgi işlemsel düşünme becerilerini geliştirirken, başka birçok beceri de edinebilirler. Örneğin, okul öncesi dönemdeki çocuklar dijital ve robotik cihazlarla etkileşime girdiklerinde üst biliş becerileri (Highfield, 2010), örüntü tanıma, sıralama ve algoritma tasarlama becerileri (Saxena vd., 2020), modülerlik, algoritmalar, temsil etme (Relkin vd., 2021) ve sıralama ve algoritma tasarımı (Saxena vd., 2020) gibi bilişsel becerilerin yanı sıra dijital platformlarda işbirlikçi çalışmalarını gerektiren ve iletişim kurmayı gerektiren etkinliklere dahil olarak sosyal-duygusal becerilerini de geliştirilebilir (Critten vd., 2021). Sonuç olarak, erken yaşlarda bilgi işlemsel deneyimler, okul öncesi çocuklarının entelektüel gelişimine, problem çözme

becerilerine, yaratıcılıklarına ve muhakeme becerilerine güçlü bir temel sağlamanın yanı sıra hayatları boyunca onlara fayda sağlayabilecek, teknoloji odaklı bir dünyaya uyum sağlayarak kendilerini geliştirmelerini sağlayacak sosyal-duygusal becerilerle donatabilecektir.

Weintrop ve arkadaşlarına göre (2015), bilgi işlemsel düşünme becerileri incelenirken iyi tanımlanmış, öğretilebilir ve ölçülebilir becerilere ayrıştırılmalıdır. Ayrıca, bilgi işlemsel düşünme becerilerinin yapılarını açıklamak için gelişimsel olarak uygun bir çerçeve üzerinde çalışılmalıdır. Ancak, bilgi işlemsel düşünmenin kavramları ve temel becerilerine yönelik çerçeveler veya yaklaşımlar çoğunlukla daha büyük çocuklar için uygun olduğu görülmüştür (Atmatzidou ve Demetriadis, 2016; Brennan ve Resnick, 2012; Selby ve Woollard, 2013; Weintrop vd., 2016). Bu sebeple Bers (2018) küçük çocukların gelişimlerine uygun tanımladığı *yedi güçlü fikre* odaklanarak, bilgi işlemsel düşünme becerilerinin bu yedi güçlü fikirle edinecekleri deneyimler yoluyla kazanılabileceğini belirtmiştir.

Aslında, güçlü fikirler terimi ilk olarak Papert (1980) tarafından ele alınmıştır. Ancak Bers (2018) bilgi işlemsel düşünmenin yedi güçlü fikrini donanım/yazılım, temsil, kontrol yapıları, algoritmalar, modülerlik, hata ayıklama ve tasarım süreci olarak ayrıştırmış ve tanımlamıştır. Weintrop ve meslektaşlarının (2015) çağrısına yanıt olarak araştırmacı, bu güçlü fikirlerin küçük çocuklar için bilgi işlemsel düşünme becerilerinin gelişimsel olarak uygun temelleri olduğunu, öğretilebilir ve ölçülebilir kavramlar içerdiğini belirtmiştir (Bers, 2018). Bu nedenle, bu çalışmada okul öncesi çocuklarının bilgi işlemsel düşünme becerileri incelenirken yedi güçlü fikre ve okul öncesi çocuklarındaki gelişimlerine odaklanılmıştır.

Okul öncesi çocuklarının gelişimlerine uygun uygulamalar üzerine yapılan araştırmalar incelendiğinde, kodlama ve programlama etkinliklerinin çocukların bilgi işlemsel düşünme yeteneklerini geliştirmelerine yardımcı olduğu; programlamanın bilgi işlemsel düşünmenin gelişmesine yardımcı olan bir beceri veya öğrenme yolu olduğu belirtilmiştir (Arfe vd., 2019; Bers, 2008). Nitekim çocuklar kodlama yoluyla programlama yaparken, bilgi işlemsel düşünme becerilerine katkıda bulunan becerilerden; sıralama, algoritmalar, hata ayıklama ve tasarım süreci, problem

temsili, problem çözme, keşif ve çoklu çözümlerin yinelenmesi gibi çeşitli düşünme süreçleri ve becerilerini deneyimlemektedir (Bers, 2018; Papert, 1980). Bu nedenle, çocukların bilgi işlemsel düşünme becerilerini geliştirmek için kodlama ve programlama uygulamalarından faydalanılması ve hem dijital hem de somut materyallerle öğrenme fırsatları verilmesinin daha faydalı olacağı önerilmektedir (Strawhacker ve Bers, 2018; Bers, 2018; Futschek ve Moschitz, 2011; Koning, Faber ve Wierdsma, 2017).

Buna ek olarak, çocukların bilgi işlemsel etkinliklerdeki başarısını bağımsız olarak etkileyen yaş ve cinsiyet gibi değişkenler vardır. İlgili alan yazın, cinsiyetin küçük çocukların bilgi işlemsel düşünme becerileri üzerinde bir etkisi olmadığını ortaya koymaktadır (Bati, 2022; Papadakis vd., 2016; Sullivan ve Bers, 2013; Yang vd., 2023). Yaştan ziyade motivasyon, toplumsal cinsiyet kalıp yargıları ve öğretmenin cinsiyetinin kız ve erkek çocuklar arasında fark yarattığı ve bilgi işlemsel etkinliklerdeki başarılarını etkilediği bildirilmektedir (Doube ve Lang, 2012; Master vd., 2021; Master vd., 2023; Sullivan ve Bers, 2018; Sullivan ve Bers, 2019). Örneğin, erkek öğretmenlerin kızlara kıyasla erkek çocuklarının bilgi işlemsel düşünme becerileri üzerinde daha olumlu bir etkisi olduğu bildirilmektedir (Sullivan ve Bers, 2018; Sullivan ve Bers, 2019).

Bununla birlikte, erkek çocuklarının yaşları ilerledikçe kodlamayla ilgili görevlerde kızlardan daha iyi olduklarını düşündükleri ve programlamaya kızlardan daha fazla ilgi duydukları belirtilse de (Master vd., 2021), okul öncesi döneminden itibaren gelişimlerine uygun bilgi işlemsel etkinlikler yardımıyla, kız ve erkek çocukların kodlama ve programlama görevlerinde eşit derecede başarılı olabilecekleri belirtilmiştir (Bati, 2022).

Cinsiyete benzer şekilde, yaş faktörünün de çocukların bilgi işlemsel etkinliklerdeki başarısı üzerinde bir etkisi olmadığı açıklanmıştır (Papadakis vd., 2016; Relkin vd., 2021), ancak çocukların büyüdükçe, beş ve daha fazla talimat içeren daha karmaşık görevleri tamamlayabildikleri bildirilmiştir (Sullivan ve Bers, 2016). Bu bağlamda, okul öncesi çocuklarda bilgi işlemsel düşünme becerilerine katkıda bulunan faktörlerin araştırılması gerekmektedir.

Çalışmanın Önemi

Bilgi işlemsel düşünme becerileri, birçok problem çözme stratejisini ve tasarım süreci prosedürlerini kullanan sıralama, algoritmik ve analitik düşünme, soyut ve yaratıcı düşünme gibi birçok beceriyi kapsamaktadır (Bers, 2018; Selby ve Woollard, 2013; Wing, 2008). Çok boyutlu bir yapıya sahip olması, bilgi işlemsel düşünmeyi oluşturan gelişimsel süreçleri, yaklaşımları ve teknikleri keşfetmeyi zorlaştırmaktadır. Bununla birlikte, hangi becerilerin okul öncesi çocuklarının bilgi işlemsel düşünme becerilerine katkıda bulunduğu konusunda çok az şey bilinmektedir (Su ve Yang, 2023; Wing, 2008). "İnsanlar bu tür düşünmeyi nasıl ve ne zaman öğrenmeli ve bunu nasıl ve ne zaman öğretmeliyiz?" sorularına cevap bulmak ve bu yetenekleri geliştirmenin sağlam temelini kavramak için erken çocukluk dönemi çocuklarında uygulamalı araştırmaların yürütülmesi gerekmektedir (Wing, 2008, s. 3720). Bu konuların araştırılması, küçük çocukların bilgi işlemsel düşünme becerilerine katkıda bulunan uygulamaların yanı sıra bilgi işlemsel düşünme becerilerinin bileşenlerini belirlemeyi sağlayabilir. Bu çalışmada da okul öncesi dönemindeki çocukların bilgi işlemsel düşünmenin yedi güçlü fikri (temel içerik ve beceriler) incelenmiştir. Ayrıca Bers (2018), çocukların güçlü fikirlere ilişkin anlayışlarının gelişiminin kültürel geçmişlerine bağlı olarak farklılık gösterdiğini ve farklı bağlamlarda güçlü fikirlerin gelişim sürecini açıklamak için gelecekteki araştırmalara ihtiyaç olduğunu belirtmiştir. Alan yazın incelendiğinde, yedi güçlü fikir (Relkin, 2018; Relkin vd., 2021) üzerinden bilgi işlemsel düşünmenin gelişimini inceleyen sınırlı sayıda araştırma olduğu ve ulusal alan yazında okul öncesi çocuklarda bilgi işlemsel düşünmenin yedi güçlü fikri üzerine herhangi bir araştırma bulunmadığı görülmüştür. Dolayısıyla bu araştırma, güçlü fikirlerin Türkiye'deki eğitim ortamlarına etkili bir şekilde dahil edilip edilemeyeceğinin yanı sıra Türk okul öncesi çocuklarındaki var olan gelişimini inceleyerek, güçlü fikirlerinin nasıl ve ne ölçüde geliştirildiğini belirleyebilir ve sonuçlar gelecekteki araştırmalar için hem uluslararası hem de ulusal alanlara katkıda bulunabilir.

Daha önce de belirtildiği gibi, eğitsel robotikler bilgi işlemsel düşünme becerilerine programlama yoluyla katkıda bulunmaktadır. Erken yaşlarda programlama

öğrenmenin matematik ve fen başarısı (Metin, 2020), sosyo-duygusal ve dil gelişimi (Bers, 2008), yaratıcılık (Canbeldek ve Işıkoğlu, 2023; Clements, 1991) ve problem çözme (Fessakis vd., 2013) üzerinde olumlu bir etkisi olduğu bildirilmiştir. İlgili alan yazın incelendiğinde, Munoz-Repiso ve Caballero-Gonzalez'in (2019) eğitsel robotik aracılığıyla yürüttüğü TangibleK programının uygulanması sonrasında, okul öncesi çocukların bilgi işlemsel düşünme becerilerinin önemli ölçüde geliştiği görülmüştür. Ayrıca Angeli ve Valadines (2020), eğitsel robotiklerle uygulanan programlama etkinliklerinin çocukların bilgi işlemsel düşünme ve uzamsal ilişki becerilerini geliştirmedeki faydalarını bildirmiştir. Son yıllarda çalışmaların sayısının arttığı bildirilse de (Zurnacı ve Turan, 2022), erken yaşlarda kodlama ve programlama etkinliklerinin bilgi işlemsel düşünme üzerindeki etkisini araştıran kısıtlı sayıda ulusal araştırma olduğu gözlemlenmiştir. Aslında, Bozkurt Polat (2023) ve Değirmenci (2022) dışında, ulusal çalışmalar çoğunlukla bilgisayarsız kodlama veya programlama becerilerini (Metin, 2020), bilgisayarsız kodlamanın analitik düşünme, yaratıcılık ve problem çözme üzerindeki etkisini (Akyol-Altun, 2018; Çakır vd., 2021), yaratıcı düşünme (Önal ve Ardıç, 2020) ve matematiksel akıl yürütme becerileri (Canbeldek ve Işıkoğlu, 2023) üzerindeki etkisini ortaya koymuştur. Mevcut çalışmanın da eğitsel robotik ile programlama deneyimi ve bilgi işlemsel düşünme becerileri arasındaki ilişkiye ışık tutacağı düşünülmektedir.

Yukarıda belirtilenlerin yanı sıra, Su ve Yang (2023) bir derleme çalışması yürütmüş ve burada bilgi işlemsel düşünme süreçlerine ilişkin ölçümlerin geçerli ve güvenilir olması gerektiği sonucuna varmıştır. Bunun nedeni, birçok araştırmanın bilgi işlemsel düşünmeyi ölçen araçların geçerliliğini ve güvenilirliğini gerektiği gibi rapor etmede eksik kalmasıdır (Su ve Yang, 2023). Okul öncesi çocukların bilgi işlemsel düşünme becerilerine yönelik bir değerlendirme aracı oluşturmak amacıyla ulusal ve uluslararası literatür taranarak bir dizi ölçüm aracı keşfedilmiştir. Ancak, bu araştırmada güçlü fikirlerin gelişiminin incelenmesi hedeflendiği için, güçlü fikirler bileşenlerini içeren TechCheck-K ölçme aracı Türkçeye çevrilip, geniş bir örneklem ile geçerlik ve güvenirlik çalışması yapılarak ulusal alan yazına kodlama ve programlama deneyimi gerektirmeyen bir ölçme aracının da kazandırılması hedeflenmiştir. Önceki araştırmalardan (Çetin vd., 2022; Yılmaz, 2022) farklı olarak, bu çalışmada, özellikle daha büyük örneklemlere göre araç geliştirirken veya

uyarlarken klasik test teorilerine göre bir araçtaki her bir madde hakkında daha fazla bilgi sağladığı düşünülen madde tepki teorisi uygulanmıştır (Baker ve Kim, 2017). Bu nedenle, mevcut araştırma, TechCheck-K hakkında bildirilen önceki sonuçları, madde tepki kuramı aracılığıyla daha büyük bir anaokulu çocuğu örneklemi aracılığıyla genişletme ve zenginleştirme potansiyeline sahiptir.

Bu çalışmada, okul öncesi çocuklarının programlamaya dayalı olmayan bir ölçme aracı ile (TechCheck-K) bilgi işlemsel düşünme becerilerinin incelenmesinin yanı sıra programlamaya dayalı bilgi işlemsel düşünme becerilerinin incelenmesi de hedeflendiğinden, çocukların programlama performanslarını ölçmeyi sağlayacak bir ölçme aracına ihtiyaç olmuştur. Okul öncesi çocuklarının programlamaya dayalı bilgi işlemsel düşünme becerilerini ölçen ve tüm robotik platformlarda kullanılabilecek evrensel bir değerlendirme aracı bulunmamaktadır. Bu nedenle, TACTIC-KIBO değerlendirme aracı (Relkin ve Bers, 2019) Türkçeye çevrilmiştir. TACTIC-KIBO ölçme aracı, çocukların programlama temelli bilgi işlemsel düşünme becerilerini ölçmek için bilgi işlemsel düşünme becerilerini programlama ile iç içe geçirmektedir. Ayrıca, çocukları programlama performanslarına dayalı olarak proto-programcı, erken programcı, programcı ve akıcı programcı seviyelerinde olmak üzere gruplamaktadır (Relkin, 2018). Ancak bu çalışmada, akıcı programcı seviyesindeki sorular okul öncesi çocukları için karmaşık ve soyut kavramlar içerdiği için, ilk üç düzey Türkçeye çevrilerek kullanılmıştır.

Araştırmalara göre, bilgi işlemsel düşünme süreçleri çocukların demografik özelliklerinin yanı sıra bilgi birikimlerinin de göz önünde bulundurulmasıyla ve etkinliklerin bu doğrultuda hazırlanarak uygulanmasıyla geliştirilebilmektedir (Batı, 2022; Tikva & Tambouris, 2021). Bu doğrultuda hazırlanan kodlama ve programlama temelli bir müfredatın, bilgi işlemsel düşünme becerilerine katkısının belirlenmesinin yanı sıra demografik ve diğer potansiyel faktörlerin katkısının da yordanması verimli uygulamaların tespiti için faydalı olabilir. Örneğin, Yıldırım ve Uluyol (2022) ilkokul çocuklarının bilgi işlemsel düşünme becerilerinde sınıf düzeyine göre anlamlı bir fark olduğunu bildirmişlerdir. Benzer şekilde, Korucu ve diğerleri (2017) ve Catana-Kuleli (2018) de sınıf düzeyinin bilgi işlemsel düşünme becerilerinde anlamlı bir fark yarattığını bulmuştur. Atmatzidou & Demetriadis

(2016), çocukların yaşının modülerlik ve ayrıştırma, algoritmalar ve genelleme ve soyutlama gibi bilgi işlemsel düşünmenin belirli boyutlarında fark yarattığını belirtmiştir. Buna ek olarak, Bers ve diğerleri (2022) ile Relkin ve diğerlerinin (2021) sonuçları, birinci sınıf ve ikinci sınıf çocukları arasında bilgi işlemsel düşünme becerileri açısından anlamlı bir fark olduğunu göstermiştir. Relkin ve diğerleri de (2021) birinci ve ikinci sınıf çocuklarıyla uygulanan programlama etkinliklerinin birinci sınıf çocuklarına anlamlı bir katkı sağladığını ancak ikinci sınıf çocuklarına bir katkı sağlamadığını bildirmiştir. Bu çalışmalar göz önünde bulundurulduğunda, bildiğimiz kadarıyla, eğitsel robotik tabanlı programlama uygulamalarının farklı aylardaki okul öncesi çocukların bilgi işlemsel düşünme becerileri üzerindeki etkisine ilişkin sınırlı sayıda araştırma bulunmaktadır (Atmatzidou ve Demetriadis, 2016; Değirmenci, 2022; Relkin ve Bers, 2021). Bu nedenle, ilgili alan yazına katkıda bulunmak amacıyla bu çalışmada da yaşın bilgi işlemsel düşünme becerilerinin gelişmesindeki etkisi araştırılmıştır.

Bilgi işlemsel düşünme için etkili öğretme ve öğrenme stratejileri belirlenirken yaşa ek olarak cinsiyet farklılıkları da dikkate alınabilir. Nitekim bu becerilerin geliştirilebilmesi için cinsiyet faktörüne özellikle hassasiyet gösterilmesi gerektiği ve etkinliklerin hem kız hem de erkek çocuklarının ilgi alanlarına hitap etmesi gerektiği belirtilmiştir (Metz, 2007). Olumsuz kalıp yargılarının kız çocuklarının ileride STEM alanlarına yönelmesini zorlaştırdığı belirtilerek, erken yaşlardan itibaren eğitsel robotik ve programlama deneyimlerinin sunulması gerektiği ve bu sayede kız çocuklarının bu alanlara olan ilgisini artırılabileceği belirtilmiştir (Master vd., 2023). Cinsiyetin bilgi işlemsel düşünme becerilerinde bir fark yaratıp yaratmadığına ilişkin çalışmalar incelendiğinde, sonuçların farklılık gösterdiği bulunmuştur. Örneğin, Tsai ve diğerleri (2021), erkek öğrencilerin *ayrıştırma* konusunda kız öğrencilerden daha iyi olduğunu belirtmiştir. Bir başka çalışma, okur yazarlık olarak kodlama temelli bir programlama deneyimi aldıktan sonra kız ve erkek öğrenciler arasında bir fark olmadığını ortaya koymuştur (Yang vd., 2023). Relkin ve Bers (2021) ise kız ve erkek çocuklarının bilgi işlemsel düşünme becerilerinde anlamlı bir fark olmadığını bildirmiştir. Ancak Relkin ve diğerlerine (2021) göre ise, ikinci sınıf çocuklarının bilgi işlemsel düşünme becerileri robotikle uygulanan programlama etkinliklerinin uygulanmasında sonra önemli ölçüde değiştiği ve cinsiyete göre farklılık gösterdiği,

birinci sınıf çocuklarının bilgi işlemsel düşünme becerilerinde ise cinsiyete bağlı bir farklılık olmadığı ortaya çıkmıştır. Bu çelişkili sonuçlar, çocukların STEM disiplinlerine ilişkin kalıp yargılarının yaşları ilerledikçe değişmesinden ve bu kalıp yargıların erken yıllardan itibaren gelişimsel olarak uygun uygulamalarla ele alınmasının tespit edilmesinden kaynaklanıyor olabilir (Master vd., 2023; Sullivan ve Bers, 2018; Sullivan ve Bers, 20189). Bununla birlikte, sonuçlar kültürel açıdan farklılık gösterebileceğinden, cinsiyetin Türk okul öncesi çocuklarının bilgi işlemsel düşünme becerileri üzerindeki olası etkilerini açıklamak için daha fazla araştırmaya ihtiyaç vardır. Bu sebeple bu çalışmanın sonuçları, uygulanan müfredatın her çocuğa eşit davranarak hem kız hem de erkek çocukların ilgilerini karşılayıp karşılamadığını da gösterebilir. Zira kız ve erkek çocukların ilgi alanlarına göre hazırlanmış erken çocukluk programlarının katılımcıların bilgi işlemsel düşünme becerileri üzerinde herhangi bir etkisinin olmayacağı düşünülmektedir (Batı, 2022).

Sonuç olarak, bu çalışmada, kodlama ve programlama etkinliklerini uygulamak için KIBO eğitsel robotları kullanılarak alan yazın ışığında okul öncesi çocuklarının bilgi işlemsel düşünme becerilerinin gelişiminin programlamaya dayalı olmayan bir ölçme aracının yanı sıra programlamaya dayalı bir ölçme aracı yoluyla incelenmesi hedeflenmiş; okul öncesi dönemdeki çocukların bilgi işlemsel düşünme becerilerinin gelişimini etkileyebilecek değişkenler araştırılmıştır.

YÖNTEM

Çalışma deseni

Bu çalışmada 1. aşama pilot çalışma ve 2. Aşama ana çalışma olmak üzere iki aşamada nicel araştırma yürütülmüştür. 1. aşamada, araştırmada kullanılacak veri toplama araçlarını (TechCheck-K ve TACTIC-KIBO araçları) Türkçeye çevirerek TechCheck-K aracının pilot çalışmasını yürütmek; ayrıca, ana çalışmada uygulamak üzere Coding as Literacy-KIBO (CAL-KIBO) anaokulu programını Türkçeye çevirmek amaçlanmıştır. Çeviriler yapıldıktan ve veri toplama araçları ile uygulanacak program için uzman görüşleri alındıktan sonra TechCheck-K ölçme aracının geçerlik ve güvenirlik özellikleri araştırılmıştır (Relkin ve Bers, 2021). Bu

amaçla, katılımcılardan belirli bir zamanda veri toplamak için faydalanılan kesitsel tarama araştırmasından yararlanılmıştır (Fraenkel vd., 2011).

Çalışmanın 2. aşamasında ise iki adımdan oluşan deneysel araştırma yürütülmüş; ilk adımda robotik tabanlı kodlama uygulamasından sonra okul öncesi çocuklarının bilgi işlemsel düşünme becerilerinde bir değişiklik olup olmadığını anlamak için yarı deneysel ön test-son test kontrol grubu tasarımı yürütülmüştür. Ön test ve son testi uygulamak için ise kodlama ve programlama deneyimi gerektirmeyen TechCheck-K değerlendirme aracı hem deney hem kontrol grubuna uygulanmıştır. İkinci adımda ise, deney grubu çocuklarının programlama temelli bilgi işlemsel düşünme becerilerini TACTIC-KIBO değerlendirme aracı ile incelemek için tek gruplu son test içeren durum çalışması gerçekleştirilmiştir (Fraenkel vd., 2011). TACTIC-KIBO ile çocukların programlamaya dayalı bilgi işlemsel etkinliklerdeki performansları proto-programcı, erken programcı ve programcı olarak kategorize edilmektedir.

İkinci aşamada, ayrıca ebeveynlerin demografik verileri ile kontrol ve deney gruplarındaki çocuklarının bilgileri de toplanmıştır. Bu amaçla kontrol ve deney gruplarındaki çocukların ailelerine demografik bilgi formu gönderilerek elde edilen bilgiler çocukların bilgi işlemsel düşünme becerileri üzerinde ne kadar etkiye sahip olduklarını belirlemek için kullanılmıştır.

1. Aşama: Pilot Çalışmadaki Veri toplama araçları

TechCheck-K Ölçme Aracı

TechCheck-K 15 maddeden oluşmaktadır ve donanım/yazılım, kontrol yapıları, temsil etme, hata ayıklama, algoritmalar ve modülerlik olmak üzere yedi güçlü fikir üzerine inşa edilmiştir (Bers, 2018). TechCheck-K yinelemeli ve açık uçlu bir süreç olduğu ve kapalı uçlu ile çoktan seçmeli sorularla değerlendirilemeyeceği için tasarım süreci fikrini içermemektedir (Relkin ve Bers, 2021). TechCheck-K aracı uygulamadan önce kodlama deneyimi gerektirmeyen, çocukların programlama gelişim aşamalarını değerlendirmeyen ve sınıflandırmayan bir ölçme aracıdır. TechCheck-K'deki sorular teknolojik kavramlar, en kısa yolu bulma bulmacaları, bir

örüntüdeki eksik sembolü bulma ile nesne ayrıştırma görevleri ile engelli labirentler ve sembollerle örüntü oluşturmaya yönelik sorular içermektedir (Relkin vd., 2020). Araç, çevrimiçi https://www.qualtrics.com platformları aracılığıyla bir tablet veya dizüstü bilgisayarda uygulanabilir, çıktısı alınabilir veya bir ekrana yansıtılarak kalem ve kâğıt ile uygulanabilir. Bu çalışmada TechCheck-K soruları çocuklara küçük gruplar halinde yüksek sesle okunmuş ve çocuklar kendi değerlendirme kağıtlarında doğru yanıtı kalemleriyle daire içine almışlardır.

Katılımcılar

TechCheck-K ölçme aracının pilot çalışmasına başlamak için aracın çevirisi ve uzman görüşlerinden önce orijinal yazarlara e-posta yoluyla izin gönderilmiştir. Ardından, Aksaray İl Milli Eğitim Müdürlüğünden ve Orta Doğu Teknik Üniversitesi (ODTÜ) Araştırma Etik Kurulundan onay alınmıştır. Bu nedenle, veri toplamak için katılımcıları seçerken yakınlarda bulunan ve araştırmacı tarafından kolayca ulaşılabilen kişilerden ulaşılabilir örneklem tercih edilmiştir (Fraenkel, vd., 2011). Daha sonra, Aksaray şehir merkezindeki on anaokulu müdürü ile görüşmeler yapılmış ve katılımcı okullardan bilgi toplanmıştır. Görüşmelerin ardından Aksaray il merkezindeki altı devlet anaokulu TechCheck-K ölçme aracı pilot çalışmasına katılmayı kabul etmiştir. 352 çocuğun verileri, 2021-2022 eğitim-öğretim yılının güz ve bahar dönemlerinde kendi rızalarıyla toplanmıştır. TechCheck-K değerlendirme aracı, uygulamada tutarlılığı sağlamak için çalışmanın araştırmacısı tarafından uygulanmıştır.

Pilot Çalışma Sonuçları

Ortalama 69 aylık (minimum = 49, maksimum = 86) toplam 163 kız ve 189 erkek çocuğa TechCheck-K değerlendirme aracı uygulanmıştır. TechCheck-K puanları, 15 üzerinden minimum 0 ve maksimum 15 puan olmak üzere normal dağılım göstermiştir. TechCheck-K puanları ortalama 7.33 ile 2.6 arasında değişmektedir. TechCheck-K'nin güvenirliğini kontrol etmek için Cronbach alfa değeri hesaplanmış ve .63 olarak bulunmuştur. TechCheck-K'nin orijinal pilot çalışmasında (Relkin ve Bers, 2021) Cronbach alfa değeri rapor edilmediği için karşılaştırma yapılamamıştır.

Ancak araştırmacılar, TechCheck-K'deki 15 madde için doğru yanıt örüntüsünün, Cronbach alfa değeri .68 olan TechCheck ölçme aracında gözlemlenen yanıt örüntüsüyle yüksek oranda korelasyon gösterdiğini (r = .76) bildirmiştir (Relkin vd., 2020).

Bu, TechCheck-K testinin bilgi işlemsel düşünme için aynı temel kapasiteyi değerlendirdiği anlamına gelmektedir. TechCheck'in pilot çalışmasında (Relkin vd., 2020) araştırmacılar, bilgi işlemsel düşünme becerisinin tek bir yapı yerine bir beceri seti olmasından ve altı boyut içermesinden dolayı ölçeğin iç tutarlılığını azaltabileceğini bildirmiştir. Bilgi işlemsel düşünme becerilerini ölçen diğer araçların (Zapata-Caceres vd., 2020) daha yüksek alfa değerlerine sahip olduğu ancak daha az boyuta sahip oldukları görülmüştür. Bu sonuçlar göz önünde bulundurulduğunda bilgi işlemsel düşünme becerilerinin birleşik bir yapıya sahip olmaması ve altı alanla ölçülmesi nedeniyle orta düzeyde bir Cronbach alfa değeri elde edildiği ve bu sonucun TechCheck ile yapılan önceki araştırmalarla uyumlu olduğu sonucuna varılabilir (Relkin vd., 2020).

TechCheck-K ölçme aracının betimsel sonuçlarına bakarak yapılan madde ortalama puanlarına bakıldığında en düşük ortalama puanların 10, 11, 12 ve 13. maddelerde alındığı; bu sonuçların madde tepki analizinin sonuçlarıyla da örtüştüğü görülmüştür. Madde tepki analizlerinin sonucuna göre TechCheck-K ölçme aracı orta düzeyde başarılı olan ve ortalamanın biraz altında olan okul öncesi çocuklarının puanlarını ayırt etmede daha başarılı olduğu ve Türkiye'deki okul öncesi çocuklarıyla uygulamak üzere iyi psikometrik özellikler gösterdiği bulunmuştur.

2. Aşama: Ana Çalışmadaki Veri toplama araçları

TACTIC-KIBO Ölçme Aracı

TACTIC-KIBO değerlendirme aracı Relkin (2018) tarafından KIBO robotları ile programlama görevleri uygulanmasıyla okul öncesi çocuklarının bilgi işlemsel düşünme becerilerini değerlendirmek amacıyla geliştirilmiştir. Bire bir görüşmeler yoluyla uygulanmakta ve çocukların programlamadaki başarısı 0 (yetersiz) ve 1

(yeterli) ile puanlanmaktadır. TACTIC-KIBO, Bers (2018) tarafından oluşturulan Bilgi iş Düşünmenin Yedi Güçlü Fikri teorisi etrafında inşa edilmiştir.

Relkin (2018), TACTIC-KIBO ölçme aracını tasarlarken Vizner'in (2017) KIBO robotik ile programlamaya yönelik gelişimsel modelinin aşamalarından da yararlanmıştır. Vizner (2017), KIBO robotik ile programlamanın gelişimsel seviyelerini kategorize etmiş ve proto-programcı, erken programcı, programcı ve akıcı programcı olmak üzere dört seviye tanımlamıştır. Vizner'e (2017) göre, çocuklar KIBO ile programlama yaparken, her bir seviyenin ölçütü olan bazı davranışları baskın bir şekilde sergilemektedir. Vizner'e göre (2017), bir çocuğun baskın davranışları, o aşamada kabul edilmek için her çocukta görülmesi gerekmek yerine, ulaştıkları aşamanın bir işareti olarak hizmet edebilir.

Demografik Bilgi Formu

Demografik bilgi formu, doğum tarihi, cinsiyet, ScratchJr veya diğer dijital manipülatiflerle önceki deneyimler, dijital oyunlarla geçirilen günlük ekran süresi ve evdeki somut oyun deneyimleri gibi çocukların bilgi birikimine dayalı faktörleri ve ebeveynlerin yaş, çocuk sayısı ve eğitim seviyesi gibi demografik bilgileri hakkında 13 soru içermektedir (bkz. Ek A).

Araştırmada uygulanan program tamamlandıktan sonrasında çocukların anne ve babalarına doldurmaları için demografik bilgi formu gönderilmiştir. Bunun nedeni, demografik bilgi formunda yer alan bazı soruların (kodlama uygulamaları, dijital oyun oynamak için harcanan günlük ekran süresi ve dijital oyun deneyimleri ile ilgili sorular) ebeveynleri etkileme ihtimalinin azaltılmasıdır. Örneğin formdaki sorulardan biri, ebeveynlerin ScratchJr uygulamasından haberdar olup olmadıklarını ve çocuklarının bu uygulamaya katılımını sormaktadır.
Programın uygulanmasından önce bu soruyu hayır diyerek yanıtlayabilecek bir ebeveyn, formu doldurduktan sonra ScratchJr'ı sorgulayarak çocuklarının oynaması için tablete yükleyebilir. Bu durum, bazı çocukların robotik kodlama müdahalesi sırasında araştırmacının bilgisi dışında ek kodlama ve programlama deneyimi kazanmasına neden olabilir ve dolayısıyla müdahalenin etkisinin ölçülmesini

geçersiz kılabilir. Bu aynı zamanda ScratchJr'ın dünyada en popüler ve yaygın olarak kullanılan kodlama ve programlama uygulamalarından biri olması ve KIBO'nun programlama bloklarındaki sembollere benzer simgeler içeren programlama bloklarının olduğu bir ara yüze sahip olmasından kaynaklanmaktadır. Bu durum göz önünde bulundurulduğunda, araştırmacının programın uygulanmasının ardından ebeveynlere demografik bilgi formunu vermesi, çalışmanın dış tehditlerini yönetmelerine ve daha güvenilir bulgular elde etmelerine olanak sağlamış olabilir.

Bir Okuryazarlık Olarak Kodlama KIBO Anaokulu Programı (Coding as Literacy KIBO Kindergarten Curriculum)

Okuryazarlık Olarak Kodlama-KIBO (CAL-KIBO) anaokulu programı DevTech Research grubu (2018) tarafından geliştirilmiştir, bütünleştirilmiş 24 etkinlik planı içerir ve toplam 18 saat sürmektedir. Program, Papert'in (1980) Piaget'nin çocukların nasıl öğrendiğine dair fikirlerine benzer şekilde, bu öğrenme teorisi çocuk merkezli etkinliklere ve bilginin kendi kendine yapılandırılmasına odaklanır, aynı zamanda dijital manipülatiflerin keşfine ve öncelikle teknolojik araçlarla gerçekleştirilen somut etkinliklere de odaklanır (Papert, 1993).

CAL-KIBO anaokulu programı, çocukların takım çalışması ve proje tabanlı etkinlikler yoluyla bilgi oluşturmasını sağlayan somut materyaller, yani KIBO eğitim robotları aracılığıyla yaparak öğrenmeye dayanmaktadır. CAL-KIBO anaokulu programı, çocukların bilgisayar bilimlerini öğrenmelerini, problem çözme ve bilgi işlemsel düşünme becerilerini geliştirmelerini amaçlamaktadır. Bilgisayar bilimi etkinlikleri okuma-yazmaya yönelik etkinliklerle bütünleştirilerek güçlü fikirleri pekiştirmeye odaklanmaktadır (DevTech, 2018).

Katılımcılar

Bu çalışmadaki örneklemin evreni, Aksaray şehir merkezinde bulunan devlet anaokullarından birine – X anaokulu- devam eden 60-72 aylık tüm okul öncesi çocuklar olarak kabul edilmiştir.

Tablo 1. *2.Aşamadaki Veri Toplama Süreci*

Ön test O_1	*Uygulama 1*	*Uygulama 2*	*Son test* O_2	O_3	O_4
TechCheck-K ölçme aracının uygulanması	Hazırlık etkinliklerinin uygulanması	CAL-KIBO anaokulu programının uygulanması	TechCheck-K ölçme aracının uygulanması	Demografik bilgi formunun ailelere gönderilmesi	TACTIC-KIBO ölçme aracının uygulanması

Araştırmacıya yakın ve ulaşılabilir bir grup insan olduğunda faydalı olan ulaşılabilir örneklem tercih edilmiştir (Fraenkel vd., 2011). Bu amaçla, evrenden – X anaokulu- iki deney grubu (n = 38) ve iki kontrol grubu (n = 33) olmak üzere dört sınıf seçilmiştir. Gruplar seçilirken tipik gelişim gösteren çocukların bulunduğu sınıflar ve robotik tabanlı kodlama ve programlama deneyimi olmayan çocuklar çalışma için seçilmiştir. Çalışma grubunun belirlenmesinden sonra izlenen prosedürler aşağıdaki tabloda özetlenmiştir.

SONUÇLAR VE TARTIŞMA

TechCheck-K Ölçme Aracıyla Edinilen Betimsel Ön-test Sonuçları

Schwarzer (2011), çalışmanın ölçek sonuçlarına dair bir çıkarımda bulunmak için bir ölçüt yoksa, araştırmacının medyan puanları inceleyebileceğini ve bunları ortalama puanlarla karşılaştırabileceğini belirtmiştir. Dolayısıyla, okul öncesi çocukların bilgi işlemsel düşünme puanları hakkında çıkarımlarda bulunmak için TechCheck-K'nin ortalama puanı kontrol edilmiş (M = 8.60) ve ortanca puan olan 9 civarında olduğu görülmüştür. Bu nedenle, okul öncesi çocukların orta düzeyde bilgi işlemsel düşünme becerilerine sahip oldukları sonucuna varılmıştır. Deney grubunun bilgi işlemsel düşünme becerileri 3 ile 13 arasında değişmekte olup ortalama puan 8,15 ve ortanca puan 9 ile ortalamanın biraz altındadır. Bu da deney grubundaki çocukların orta düzeyde bilgi işlemsel düşünme becerilerine sahip olduğu anlamına gelmektedir.

Benzer şekilde, kontrol grubu puanları 7 ila 14 arasında değişmekte olup ortalama puan 9,45 ve ortanca puan 9 ile ortalamanın biraz üzerindedir. Sonuç olarak, kontrol grubu çocuklarının da orta düzeyde bilgi işlemsel düşünme becerilerine sahip oldukları, ancak kontrol grubunun ortalama puanı (M = 9.45) tüm grubun ortalama puanının (M = 8.60) üzerinde olduğu için deney grubundan (M = 8.15) daha yüksek ortalama puanlara sahip oldukları görülmektedir.

Güçlü fikirlere yönelik ortalama puanlar hesaplanırken, çocukların her bir alana ilişkin puanları toplanmış ve her bir alandaki madde sayısına bölünmüştür. Sonuçlara göre, deney ve kontrol grubundaki çocuklar temsil boyutunda iki puan üzerinden en düşük ortalama puanı (M = .6), algoritmalar boyutunda ise beş puan üzerinden en yüksek ortalama puanı (M = 2.37) almışlardır. Ancak, algoritmalar boyutunda beş soru olduğu için, bu TechCheck-K aracındaki tüm boyutlar arasındaki gerçek en yüksek ortalama puan değildir. Bu nedenle, iki soru içeren boyutlar incelenmiş ve en yüksek ortalama puan kontrol yapılarında gözlenmiştir (M = 1.7). Deney grubu temsil boyutunda daha yüksek ortalama puanlara sahipken, kontrol grubu donanım/yazılım, kontrol yapıları, hata ayıklama, algoritmalar ve modülerlik boyutlarında deney grubundan daha yüksek ortalama puanlara sahiptir.

Ön test puanları arasında anlamlı bir fark olup olmadığına bakmak için bağımsız örneklem t-test analizi yürütülmüş olup, analiz sonucu deney (M= 8.16, SD=2.29) ve kontrol grubu (M= 9.45, SD= 1.87; t (56) =-2.1, p = .03, iki kuyruklu) çocuklarının bilgi işlemsel düşünme puanları arasında anlamlı bir fark olduğunu göstermiştir. Bir başka deyişle, kontrol grubu çocuklarının deney grubuna göre istatistiksel olarak anlamlı düzeyde daha yüksek ortalama puanlara sahip olduğu görülmüştür. Ortalamalardaki farkların büyüklüğü ise (ortalama fark = 1.29, %95 GA : - 2.48 ila - 0.9) orta düzeydedir (eta kare = .08).

TechCheck-K Ölçme Aracıyla Edinilen Betimsel Son-test ve Güçlü Fikirlere Yönelik Post-Hoc Madde Analizi Sonuçları

Son test sonuçlarına bakıldığında, robotik kodlama uygulaması sonrasında yapılan ölçümlerde hem kontrol hem de deney grubunda okul öncesi çocuklarının bilgi

işlemsel düşünme puanları arttığı görülmüştür (*M* = 9.05). Ancak, ön test sonuçlarında kontrol grubu çocuklarının ortalama puanları daha yüksekken (*M* = 9.45), son test sonuçları kontrol grubu çocuklarının deney grubundan (*M* = 9.36) daha düşük puanlara (*M* = 8.45) sahip olduğunu ortaya koymuştur. Bulgular, uygulamanın ardından deney grubundaki çocukların yaklaşık puanlarının bir puan arttığını, kontrol grubundaki çocukların puanlarının ise yaklaşık bir puan azaldığını göstermektedir.

Bilgi işlemsel düşünmenin güçlü fikirlerine ilişkin betimsel sonuçlar, CAL-KIBO anaokulu uygulamasından sonra deney grubu çocuklarının donanım/yazılım alanında ortalama 1,89; hata ayıklama alanında ortalama 1,73; algoritmalar alanında ortalama 2,23; temsil alanında ortalama .76 ve kontrol yapıları alanında ortalama 1,68 ve modülerlik alanında ortalama 1,05 puan geliştirdiklerini göstermiştir. Benzer şekilde kontrol grubu çocukları donanım/yazılım, modülerlik ve hata ayıklama alanlarındaki puanlarını artırmıştır. Sonuçlar ayrıca algoritmalar ve kontrol yapılarında düşüş olduğunu ve temsil etme alanında ise değişiklik olmadığını göstermiştir.

Bunlara ek olarak bilgi işlemsel düşünme alanlarındaki doğru yanıt yüzdesindeki değişimi inceleyen bir post-hoc madde analizi yapılmış, çocukların her bir TechCheck-K maddesi için doğru yanıtları ön test ve son test puanları için hesaplanmış ve ardından her bir güçlü fikir için ortalaması alınmıştır. Daha sonra, CAL-KIBO anaokulu programını alan ve almayan okul öncesi çocukların sekiz haftalık müdahale süresince değişimini incelemek için ön test yüzdeleri son test yüzdelerinden çıkarılmıştır.

Post-hoc madde analizi sonuçlarına göre, uygulama sonrasında algoritma, modülerlik ve kontrol yapılarının deney grubu çocuklarında diğer güçlü fikirlere göre biraz daha az değişim gösterdiğini ortaya koymuştur. Buna ek olarak, deney grubundaki en büyük gelişme yüzdesi sırasıyla donanım/yazılım, hata ayıklama ve temsil etme alanlarında olmuştur. Ayrıca, kontrol grubunda en büyük yüzdelik değişimler donanım/yazılım, hata ayıklama ve modülerlik alanlarında görülürken, algoritmalar ve kontrol yapıları doğru yanıt yüzdesinde düşüş göstermiştir. Son olarak, kontrol

grubunun temsil etmeye yönelik doğru cevaplarında herhangi bir yüzde değişikliği olmadığı görülmüştür.

TechCheck-K Ölçme Aracıyla Edinilen Tek Yönlü ANCOVA Sonuçları: Okuryazarlık Olarak Kodlama KIBO Anaokulu Programının Bilgi İşlemsel Düşünme Becerilerine Katkısı

Betimsel sonuçların ardından, eğitsel robotik tabanlı kodlama uygulamasının okul öncesi çocukların bilgi işlemsel düşünme becerileri üzerindeki etkisini karşılaştırmak için tek yönlü gruplar arası kovaryans analizi yapılmıştır. Ön test puanları için düzeltme yapıldıktan sonra, deney ve kontrol grupları arasında TechCheck-K değerlendirme aracının son test puanları arasında anlamlı bir fark olduğu bulunmuştur, F (1, 55) = 7.106, p = .01, kısmi eta kare =.114. Bu, CAL-KIBO anaokulu programı aracılığıyla verilen robotik tabanlı kodlama uygulamasının okul öncesi çocuklarının bilgi işlemsel düşünme becerileri üzerinde anlamlı bir etki yarattığı anlamına gelmektedir.

Deney ve kontrol grubu çocuklarının ön test puanlarındaki ortalama fark (ortalama fark = .91) dikkate alındığında, sonuçlar deney grubunun ortalama puanlarının 8.16'dan 9.36'ya yükseldiğini, kontrol grubu puanlarının ise 9.45'ten 8.45'e düştüğünü göstermektedir. Kısmi eta kare değeri küçük bir etki büyüklüğüne işaret etmektedir ve bağımlı değişkenin sadece yüzde 11,4'ü bağımsız değişken tarafından açıklanmaktadır.

TACTIC-KIBO Ölçme Aracından Edinilen Sonuçlar

CAL-KIBO anaokulu programının uygulanmasından sonra, TACTIC-KIBO sonuçlarına göre deney grubundaki çocukların çoğunun programcı seviyesine ulaştığını göstermiştir (38 çocuktan 31'i). TACTIC-KIBO sonuçları ayrıca okul öncesi çocuklar için en zorlayıcı güçlü fikirin proto-programcı ve erken programcı seviyelerindeki tasarım süreci olduğunu göstermiştir. Son olarak TACTIC-KIBO sonuçları ayrıca okul öncesi çocuklar için en zorlayıcı görevin tekrarlama döngüleri

olduğunu, çocukların çoğunun söz dizimsel olarak doğru sıralanmış tekrarlama döngüsü içeren programları oluşturamadığını göstermiştir.

TechCheck-K Ölçme Aracıyla Edinilen Yaş ve Cinsiyetin Bilgi İşlemsel Düşünme Becerilerine Etkisine Yönelik Sonuçlar

Üç yönlü ANCOVA sonuçlarına göre, CAL-KIBO anaokulu programının uygulanmasından sonra kontrol ve deney gruplarındaki çocuklar arasında cinsiyet ve ay olarak yaş bakımından bilgi işlemsel düşünme becerilerinde anlamlı bir fark olmadığı bulunmuştur. Ancak, ön test ortalamaları ve tahmini marjinal son test ortalamaları üzerinden çizgi grafikleri incelendiğinde, deney ve kontrol grubu çocuklarının bilgi işlemsel düşünme becerilerinde ay olarak yaş açısından anlamlı bir fark olmamasına rağmen, deney grubundaki ortalama puanlar 62-72 aylık çocuklar ile 72,01-80 aylık çocuklar arasındaki farkın uygulama sonrasında azaldığını göstermiştir. Bu durum, CAL-KIBO anaokulu uygulamasının bu çalışmaya katılan okul öncesi çocuklarının bilgi işlemsel düşünme becerileri üzerindeki etkisini gösterebilir. Çünkü kontrol grubundaki çocukların ön ve son test puanları arasındaki ortalama fark, bilgi işlemsel düşünme becerilerini geliştirmek için herhangi bir tedavi almayan 62 ila 72 aylık çocuklar ile 72,01 ve 80 aylık çocuklar için aynıdır. Ayrıca hem kontrol hem de deney grupları için kız ve erkek çocuklar arasında ön test ortalama puanları açısından bir fark bulunmamıştır. Uygulama sonrasında deney grubundaki kız ve erkek çocukların bilgi işlemsel düşünme puanlarında artış gözlenmiştir. Ancak ön test sonuçlarına benzer şekilde deney grubundaki kızların ve kontrol grubundaki erkeklerin son test puan ortalamalarının diğerlerine göre daha yüksek olduğu görülmüştür. Son olarak, erkekler ve kızlar arasındaki son test ortalama farkı her iki araştırma grubunda da benzerdir ve büyük değildir.

Tartışma ve Öneriler

Mevcut çalışmanın sonuçlarına göre, TechCheck-K aracı Türk anaokulu öğrencileriyle iyi ila orta düzeyde psikometrik özelliklere sahiptir. Özellikle, 12. madde hariç, TechCheck-K'deki diğer maddelerin Relkin ve diğerlerinin (2020) çalışmasıyla uyumlu olarak kabul edilebilir zorluk seviyelerine ve ayırt edicilik

parametrelerine sahip olduğu bulunmuştur. Bu durum, TechCheck-K aracının farklı yetenek düzeylerine sahip okul öncesi çocukların bilgi işlemsel düşünme becerilerini ölçmede ve ayırt etmede yararlı olduğunu ve çoğu maddenin kabul edilebilir güçlük düzeyine sahip olduğunu ortaya koyabilir. Ancak, madde 12 için madde bilgisi ve madde özellikleri eğrileri, düşük yetenekli öğrenciler için doğru bilgi verme olasılığının arttığını, yüksek yetenekli öğrenciler için ise tam tersi olması gerekirken doğru bilgi verme olasılığının azaldığını göstermiştir. Aslında, temsil etme güçlü fikrini ölçen 12. madde, mevcut çalışmanın hem pilot hem de ana çalışmalarında en düşük ortalama puanlardan birini almıştır. Bu madde aynı zamanda çocuklar tarafından boş bırakılan maddelerden biridir ve onlar için zorluğuna işaret ediyor olabilir. 12. Madde ve diğer maddelerin de ayırt edicilik ve zorluk parametreleri kontrol edilmiş, ancak şans parametresi örneklem küçüklüğünden dolayı test edilememiştir. Örneklem sayısı artırılarak TechCheck-K maddelerinin şans parametresi göz önünde bulundurularak psikometrik özelliklerinin araştırılması önerilmektedir.

Bu çalışmada okul öncesi çocuklarının kontrol ve deney grubu gözetmeksizin ön test sonuçlarında orta düzeyde bilgi işlemsel düşünme becerilerine sahip olduğu bulunmuştur. Alan yazınla tutarlı olan bu sonuç, çalışmaya katılan çocukların bulundukları bilişsel gelişim evresine bağlı olarak ortaya çıkmış olabilir. Relkin ve diğerleri (2021) ve diğer ilgili literatüre göre, çocukların bilgi işlemsel düşünme becerileri, sınıflarına veya gelişimsel aşamalarına ve aracın zorluk derecesine bağlı olarak değişebilir. Örneğin ikinci sınıf öğrencileri, bilişsel gelişim ve olgunlaşmadaki farklılıklar nedeniyle birinci sınıf çocuklarından daha yüksek puanlar alabilmektedir (Çetin vd., 2022; Relkin vd., 2021). Dolayısıyla, bu çalışmaya katılan çocukların işlem öncesi bilişsel gelişim düzeyinde oldukları göz önünde bulundurulduğunda, daha yüksek ortalama puanlar ya çocukların bilişsel gelişim yaşından ya da araçların çocuklar için zorluk derecesinden kaynaklanıyor olabilir. Bu sonuç çocukların güçlü fikirlere özgü ön test sonuçları göz önüne alındığında aslında şaşırtıcı değildir; zira mevcut çalışma aynı zamanda tüm çocukların temsil etme (M = .55), modülerlik (M = 1,03) ve algoritma (M = 2,37) alanlarında en düşük ortalama puanlara sahip olduğunu ortaya koymuştur. İlgili literatür ışığında bu durum, bu çalışmaya katılan çocukların sembollerin eylemleri temsil ettiğini ve belirli davranışlarda

yürütüldüğünü anlamakta (temsil etme) ve büyük bir görevi ayrıştırarak (modülerlik) bir çözüm yolu için birden fazla adımı sıralamakta (algoritmalar) zorlandıkları anlamına gelebilir (Bers, 2021; Brennan ve Resnick, 2012; Rijke vd., 2017). Bununla birlikte, bir dizi çalışma, temsil etme, modülerlik ve algoritmaların, diğer alanların gelişimini kolaylaştıran, diğer temellerin kazanılması için gerekli olan ve bir alandaki yeterliliğin diğerinde başarı için bir ön koşul olabileceğini ortaya koymuştur (Bers, 2021; Brennan ve Resnick, 2012; Grover ve Pea, 2014; Nouri vd., 2020; Rijke vd., 2017; Wing, 2008).

Ayrıca, bazı çalışmalar ScratchJr gibi eğitsel robotik ve kodlama uygulamaları kullanılarak yapılan kodlama ve programlama etkinlikleri aracılığıyla küçük yaştaki öğrencilerin bilgi işlemsel düşünme becerilerinin geliştirilebileceğini öne sürmüştür (Saxena vd., 2020; Tsai vd., 2021; Yang vd., 2023). Mevcut çalışma bağlamında bu durum, okul öncesi dönemdeki çocukların somut ve dijital materyallerle kodlama ve programlama etkinliklerini deneyimlediklerinde bilgi işlemsel düşünme becerilerini geliştirebileceklerini ve bilgi işlemsel düşünme becerilerinin her bir alanında daha yüksek puanlara sahip olabileceklerini ortaya koyabilir. Araştırma bulgularına örnek olarak, evde kodlama ve programlama deneyimleri yoluyla bilgi işlemsel düşünme becerilerinin geliştirilmesi ve kodlama deneyimi olan ve olmayan öğrencilerin bilgi işlemsel düşünme becerilerindeki farklılıklar verilebilir (Rum ve Ismail, 2017; Tsai vd., 2021).

Bu çalışmada beş çocuğun ekran deneyiminin olmadığı ve çocukların çoğunun bilgi işlemsel düşünme becerilerine katkıda bulunabilecek dijital oyunlarla kodlama veya programlama deneyimi olmadığı düşünüldüğünde, bilgi işlemsel düşünme becerilerinin düşük seviyeleri şaşırtıcı değildir ve Relkin ve arkadaşlarının (2021) çalışmasındaki çocukların çoğunun uygulamadan önce çok az kodlama deneyimi olduğunu veya hiç olmadığını bildiren çalışmasıyla tutarlıdır. Buna ek olarak, çocukların bilgi işlemsel düşünme becerilerinin bloklar veya Legolarla somut deneyimler yoluyla da geliştiğine inanılmaktadır (Dietz vd., 2019; Yang vd., 2022). Bu çalışmadaki çocukların çoğunluğunun bloklar ve legolarla sık sık oynamadığı göz önünde bulundurulduğunda, evde somut deneyimlerin eksikliğinin de okul öncesi

çocukların bilgi işlemsel düşünme becerilerini ve güçlü fikirlerini geliştirmelerini engellemiş olabileceği iddia edilebilir.

Ayrıca, bu çalışmada bulunan en yüksek güçlü fikirlere özgü ortalama puanlar kontrol yapıları, hata ayıklama ve donanım/yazılım alanlarıdır. Bu durum, bu çalışmadaki okul öncesi çocukların kontrol yapıları, hata ayıklama ve donanım/yazılım alanlarında daha iyi oldukları anlamına gelebilir. Aslında, küçük çocuklar deneme yoluyla problem çözmede daha iyidir ve somut ve elle tutulur materyallerle etkileşime girerek daha iyi öğrenirler (Bers vd., 2023; Papadakis vd., 2016; Saxena vd., 2020). Çocukların kontrol yapısı sorularında daha yüksek ortalama puanlar alması, kontrol yapısı sorularındaki görevlerin somutluğundan da kaynaklanıyor olabilir. Örneğin, çocuklar kontrol yapısı sorularını yanıtlarken labirentler üzerinde yollar çizmekte, çözüm yollarını kâğıt üzerinde somutlaştırmakta ve cevabı deneme-yanılma yoluyla bulmaktadır. Çocukların hata ayıklama puanlarına bakıldığında, her iki grubun da hata ayıklama görevlerinde hatayı tespit etme ve çözümü bulma konusunda iyi olduğu görülmektedir. Nitekim güçlü analitik düşünme, problem çözme veya matematiksel yeteneklere de sahip olabilirler (Cheng, 2012; Doleck vd., 2017; Fessakis vd., 2013). Ancak, mevcut çalışma bu becerilere odaklanmadığından, olası katkıları tartışılamamıştır.

Bu çalışmada, robotik kodlama uygulamasından sonra deney ve kontrol grupları arasında, deney grubu lehine anlamlı bir fark bulunmuştur ve benzer şekilde yürütülmüş çalışmaların çoğuyla tutarlı sonuçlar sunmuştur. Örneğin, Yang ve arkadaşları (2023) robotik kodlama uygulamasının, ekran tabanlı ve bilgisayarsız kodlama tabanlı etkinliklerden daha yüksek ortalama puanları ve anlamlı bir fark oluşturduğunu belirtmiştir. Yang ve arkadaşları (2022) ise sıralama becerilerine robotik kodlama tabanlı etkinliklerin sıralama becerisine katkıda bulunduğunu ama bilgi işlemsel becerilere katkıda bulunmadığını bulmuştur. Bununla birlikte, elde ettikleri sonuç, etkinliklerinde çoğunlukla kullanılan sıralama etkinliklerinden kaynaklanıyor olabilir. Sıralama becerileri algoritmaların bileşenlerinden biri olmasına ve bilgi işlemsel düşünmeye katkıda bulunmasına rağmen (Bers, 2021), tek başına bilgi işlemsel düşünmeyi geliştirmek için yeterli olmayabilir, çünkü bilgi işlemsel düşünme karmaşık bir yapıya sahiptir (Yang vd., 2022). Bu nedenle,

öncelikle sıralama becerilerinin uygulanmasına odaklanan müdahalenin bağlamı ve içeriği, mevcut çalışma ile tutarsız sonuçlarının nedeni olabilir. Ayrıca Yang ve diğerleri (2022) çalışmalarında daha az etkinlik içeren altı haftalık bir müfredat uygulamış ve bu da bilgi işlemsel düşünme becerilerinin gelişiminde anlamlı olmayan bir sonuçla sonuçlanmış olabilir. Çünkü alan yazın eğitim oturumlarının süresi ve sayısının bilgi işlemsel düşünme becerilerinin gelişiminde önemli faktörler olduğunu vurgulamaktadır (Angeli ve Valanides, 2020; Bers vd., 2014; Clark vd., 2015). Örneğin, kızların bilgi işlemsel düşünme becerilerini geliştirmek ve yeterli eğitim oturumlarıyla öğrenme boşluğunu doldurmak için erkeklerden daha fazla zamana ihtiyacı vardır (Atmatzidou ve Demetriadis, 2016). Bu çalışmada Okuryazarlık Olarak Kodlama-KIBO (CAL-KIBO) anaokulu müfredatının sekiz hafta sürdüğü ve daha fazla etkinlik içerdiği göz önünde bulundurulduğunda, bu çalışmada elde edilen anlamlı sonuç, müfredatın süre ve eğitim oturumlarının sayısı açısından uygunluğundan da kaynaklanıyor olabilir.

TACTIC-KIBO sonuçlarına dayalı olarak, mevcut çalışmada okul öncesi çocuklarının çoğu programcı seviyesine ulaşmış olsa da en zorlayıcı güçlü fikir proto-programcı ve erken programcı seviyelerindeki tasarım süreciydi. Örneğin, seviye 1'deki tasarım süreci görevi, çocuktan KIBO'yu ileriye doğru hareket ettirmeden önce ışığı açmasını istemektedir. Bu görevi başarmak için çocuğun "beyaz ışık açık" bloğunu "ileri git" bloğundan önce koyması gerekir, böylece KIBO ilerlemeden önce ışığı yakar. Seviye 2'de çocuktan KIBO'yu daha ileri götürmesi istenir. Eğer çocuk bunun bir döngü aktivitesi olduğunu fark ederse, sayı parametreli tekrar bloklarını kullanabilir veya KIBO'yu birçok kez hareket ettirmek için üzerindeki yeşil düğmeye basabilir. Ancak, çocukların çoğunluğu ileri bloğu tekrar tekrar taramayı önermiştir. Bu, araştırmadaki çocukların çoğunun bu görevin bir döngü içerdiğinin farkında olmadığı anlamına gelebilir. Aslında, bazı çocuklar tekrar bloklarını kullanmayı önermiş olsa da çoğunluğu doğru söz dizimi içeren bir program yazamamıştır. Benzer şekilde, programcı düzeyinde hata ayıklama görevinde, çocuklara tekrar döngüsü görevi sorulmuş ve sadece beş çocuk (38 çocuktan) tekrar blokları ve parametre içeren söz dizimsel ve anlamsal olarak doğru bir algoritma oluşturabilmiştir. Dolayısıyla, çocukların tasarım sürecindeki zorluğu, tekrar döngülerinden ve çocukların söz dizimsel ve anlamsal bilgideki ustalık

düzeyinden kaynaklanıyor olabilir. Misirli ve Komis (2023) söz dizimsel ve anlamsal bilginin programlama yaparken hata ayıklama kullanımı ile ilişkili olduğu fikrini desteklemiştir. Ayrıca, Fessakis ve diğerleri (2013) ile Blank ve Lynch (2018) hata ayıklamanın programlama sırasında problem çözme becerileri için önemli bir araç olduğunu vurgulamıştır. Bu durum, hata ayıklama, anlamsal ve söz dizimsel bilgi ile problem çözme becerileri arasında bir ilişki olduğunu ve tüm bunların çocukların programlama başarısını etkileyebileceğini düşündürmektedir. Sonuç olarak, bu çalışmada incelenmemiş olsa da tekrar döngüsü görevleri, okul öncesi dönemdeki çocukların söz dizimsel ve anlamsal bilgilerine, hata ayıklama ve problem çözme becerilerine ek olarak tasarım sürecindeki zorlukları olabilir. Yukarıdakilere ek olarak, TACTIC-KIBO'daki tasarım süreci sorularının çoğunlukla algoritma ve modülerlik kullanımını gerektirdiği, çünkü çocuğun öncelikle gerekli programı tanıması, doğru blokları seçmek için bu programı ayrıştırması ve bunları doğru bir algoritmada sıralaması gerektiği söylenebilir (Rijke vd., 2017; Saxena vd., 2020). Bu süreçte, okul öncesi çocuklar, özellikle çocuk birden fazla adım içeren bir görevle meşgul olduğunda gerekli olan merkezileştirme ve tersine çevrilebilirlik yeteneklerinden faydalanacaktır (Olteanu, 2015). Bu becerilerin işlem öncesi düzeyde pek gelişmediği (Piaget, 1952, 1954) ve algoritmalar ve modülerlikle ilişkili olduğu (Izu, 2017) ve okul öncesi dönemdeki çocukların TechCheck-K'deki son test puanlarının bu alanlarda düşük gelişim gösterdiği göz önüne alındığında, çocukların bu nedenlerden dolayı tasarım süreci görevlerinde zorluk yaşadıkları söylenebilir.

Tekrar döngüleri ayrıntılı olarak incelendiğinde, çocukların tekrar döngüsü görevlerinde başarısız olmalarının diğer bir nedeni, bu becerileri erken yıllarda edinmenin zorluğu da olabilir (Rich vd., 2017). Bunun nedeni, tekrarlanan döngülerin tekrarlayan programlar içermesi ve daha soyut düşünmeyi gerektirmesidir (Brennan ve Resnick, 2012); bu beceriler çocuklar somut işlemlere sahip olduklarında kolayca edinilebilir (Relkin, 2018), ancak çocuklar için karmaşık sıralama kalıpları olacaktır (Rich vd., 2017). Daha büyük çocukların daha karmaşık programlar yapabildiği düşünüldüğünde, bu durum aslında yaşın hem temel hem de karmaşık programlamada önemli bir rol oynadığını göstermektedir (Bati, 2022; Sullivan ve Bers, 2016). Dolayısıyla, bu çalışmada programlamaya dayalı bilgi işlemsel düşünme becerileri incelenirken yaşın etkisi incelenememiş olsa da

çocukların tekrar döngülerinde zorlanmaları ay olarak yaşlarından kaynaklanıyor olabilir. Örneğin, Flannery ve Bers (2013) çalışmalarında, daha büyük çocukların okul öncesi çocuklara kıyasla daha zor görevleri yerine getirdiklerini ve programlama yaparken hata ayıklama becerilerini kullanmada daha iyi olduklarını bulmuşlardır. Dietz ve diğerleri (2019) ve Rijke ve diğerleri (2018), büyük çocukların problem ayrıştırma ve soyutlama açısından daha hızlı olduklarını ve küçük çocuklardan daha iyi performans gösterdiklerini bildirmiştir. Son olarak, erkek çocukların tekrar döngüsü gibi karmaşık programlama kavramlarında daha iyi olduğu da bildirilmiştir (Sullivan ve Bers, 2018). Ancak, tekrar döngülerini tamamlayan çocuk sayısının sınırlı olması nedeniyle, bu çalışmada cinsiyet farklılığı incelenememiştir.

Bu çalışmada, genel bilgi işlemsel düşünme becerileri ve güçlü fikirler ayrıntılı olarak incelenmesine rağmen, uygulamadan sonra belirli alanlarda daha düşük gelişmeler gözlenmiştir. Bazı deneysel ve vaka araştırmalarının, çalışmalarında bilgi işlemsel düşünmenin bir ya da iki boyutuna odaklandığı görülmüştür (Misirli ve Komis, 2023; Rijke vd., 2017; Sullivan ve Bers, 2016). Bu çalışmalar, uygulama sonrasında okul öncesi dönem çocuklarındaki gelişimsel süreç ve ilerleme hakkında derinlemesine bilgi sağlayabilmiştir. Örneğin, Misirli ve Komis (2023) hata ayıklama becerilerini araştırırken, Rijke ve diğerleri (2017) problem ayrıştırma ve soyutlama becerilerini ayrıntılı olarak incelemiştir. Bu nedenle, ileride yapılacak çalışmalarda bilgi işlemsel düşünmenin bir ya da iki boyutu ve bu boyutlardaki ilerleme derinlemesine incelenebilir; bu da çocuklardaki gelişimsel farklılıkları açıklamayı kolaylaştıracak ve etkili eğitimsel çıkarımlar sağlayacaktır.

Çalışmanın sınırlılıklarından biri de ekran süresi olmayan ve dijital oyun oynama deneyimi olmayan çocuklar için örneklem boyutlarının küçük olmasıdır. Sonuç olarak, okul öncesi çocukların dijital oyunlarla olan deneyimlerinin bilgi işlemsel düşünme gelişimi açısından nasıl değiştiğine bakmak için müdahalenin ardından karşılaştırmalı analizler yapmak mümkün olmamıştır. Çocuk sayısı, ekran süreleri ve dijital oyun oynama deneyimleri bakımından eşit dağılmamıştır. Çalışmanın amaçlarından biri olsa da bu çalışma dijital oyun oynayarak geçirilen ekran süresinin bilgi işlemsel düşünme becerilerine olası etkisini inceleyememiştir. Sonuç olarak,

fişe takılı veya fişe takılı olmayan bilgi işlemsel etkinlikler uygulandıktan sonra, gelecekteki araştırmalar okul öncesi çocukların bilgi işlemsel düşünme becerilerinde hiç ekran kullanmayanlar ile kullananlar arasındaki farklılıklara ve ekranlarda dijital oyun oynayarak ne kadar zaman geçirdiklerine odaklanabilir. Bu şekilde, bilgi işlemsel düşünmenin gelişimi üzerinde bilgi işlemsel uygulamaların, ekran deneyiminin ve dijital oyunlarla geçirilen zamanın birleşik etkisi incelenebilir.

C. ETHICAL PERMISSION OF METU HUMAN SUBJECTS ETHICS COMMITTEE

UYGULAMALI ETİK ARAŞTIRMA MERKEZİ
APPLIED ETHICS RESEARCH CENTER

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 22 91
F: +90 312 210 79 59
ueam@metu.edu.tr
www.ueam.metu.edu.tr

Sayı:

Konu : Değerlendirme Sonucu

Gönderen: ODTÜ İnsan Araştırmaları Etik Kurulu (İAEK)

İlgi : İnsan Araştırmaları Etik Kurulu Başvurusu

Sayın Doç. Dr. Refika OLGAN

Danışmanlığını yürüttüğünüz Hasret KÖKLÜ YAYLACI'nın "Okul Öncesi Çocuklarının Bilgi İşlemsel Düşünme Becerilerine Yönelik Varsayımsal Öğrenme Süreçlerinin İncelenmesi" başlıklı araştırmanız İnsan Araştırmaları Etik Kurulu tarafından uygun görülmüş ve **309-ODTU-2021** protokol numarası ile onaylanmıştır.

Saygılarımızla bilgilerinize sunarız.

Prof.Dr. Mine MISIRLISOY
İAEK Başkan

D. APPROVAL OF MINISTRY OF NATIONAL EDUCATION

T.C.
AKSARAY VALİLİĞİ
İl Millî Eğitim Müdürlüğü

Sayı :
Konu :Veri Toplama ve Anket İzni

AKSARAY ÜNİVERSİTESİ REKTÖRLÜĞÜNE

İlgi: Valilik Makamının ... tarihli ve ... sayılı Onayı.

Üniversiteniz Temel Eğitim Bölümü Okul Öncesi Anabilim Dalı Araştırma Görevlisi Hasret KÖKLÜ YAYLACI, "Okul Öncesi Çocukların Bilgi İşlemsel Düşünme Becerilerine Yönelik Varsayımsal Öğrenme Süreçlerinin İncelenmesi" konulu doktora tez çalışması ile ilgili ilgi Valilik Makamı Onayı yazımız ekinde gönderilmiştir.

Bilgilerinizi arz ederim.

Hacı Ömer KARTAL
İl Millî Eğitim Müdürü

Ek: İlgi Onay (1 Adet)

Bu belge güvenli elektronik imza ile imzalanmıştır.
Adres : İl Milli Eğitim Müdürlüğü Yeni Sanayi Mah. E-90 Bulv. No:47 Belge Doğrulama Adresi : https://www.turkiye.gov.tr/meb-ebys
Valilik Ek 3 Nolu Hizmet Binası Bilgi için: Nurdan ANDAÇ
Telefon No : 0 (382) 213 68 40 Unvan : Veri Hazırlama ve Kontrol İşletmeni
E-Posta: arge68@meb.gov.tr İnternet Adresi: http://aksaray.meb.gov.tr/ Faks:3822136814
Kep Adresi : meb@hs01.kep.tr

E. CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Köklü Yaylacı, Hasret

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Early Childhood Education	2018
BS	METU Early Childhood Education	2014
High School	Kalaba Anadolu High Vocational School, Ankara	2009

WORK EXPERIENCE

Year	Place	Enrollment
2015- Present	Aksaray University	Research Assistant

PROJECT EXPERIENCE

1. Geometri School, TUBITAK Project, Educator. 01/07/2019- 05/07/2019

WORKSHOP EXPERIENCE

1. Ministry of National Education, School Profile Assessment Workshop (OPD), April 12-16, 2021, Activity Writing Moderator.

ADMINISTRATIVE DUTIES

1. Aksaray University / Faculty of Education / Department of Primary Education and Preschool Education / Department of Preschool Education Erasmus Coordinator 2017-Continuing.
2. IV. International Aksaray Symposium, (2019). Organizing Committee Member.
3. TEMA / Aksaray TEMA Provincial Representative. 2020-2022.

4. Aksaray University/Faculty of Education/Faculty Quality Commission 14.03.2022-ongoing.

PUBLICATIONS

A. Articles published in international refereed journals:

1. Köklü-Yaylacı, H. & Olgan, R. (2020). Investigating early childhood preservice teachers' personal teaching efficacy and outcome-expectancy beliefs regarding Education for Sustainable Development in Turkey. *The Teacher Educator, 56*(1), 4-24. Doi: 10.1080/08878730.2020.1767742

B. Abstracts presented at international scientific meetings and published in proceedings:

1. Köklü-Yaylacı, H., & Olgan, R. (2023). Investigating the Psychometric Issues of Techcheck-K Assessment Tool Through Item Response Theory. 8th International Early Childhood Education Congress.
2. Terzioğlu, T. E., Köklü-Yaylacı, H., Mert, Z., Tantekin-Erden, F., & Yalçın, F. (2019). A Systematic Review of Fundamental Approaches to Early Childhood Education Research. VIIth International Euresian Educational Research Congress. (Yayın No:5420196)
3. Köklü-Yaylacı, H., & Olgan, R. (2018). Early childhood teachers' views regarding Education for Sustainable Development: A focus group study. 16th International JTEFS/BBCC Conference Sustainable Development, Culture, Education. (Yayın No:4487013)
4. Yaylacı, M. E., Gelbal, S., & Köklü-Yaylacı, H. (2018). 2003-2015 PISA applications of information technologies application comparison of the competence scale by years and the relationship between Turkish students' success. 16th International JTEFS/BBCC Conference Sustainable Development, Culture, and Education.
5. Köklü-Yaylacı, H., & Olgan, R. (2018). Pre-service early childhood teachers' Education for Sustainable Development teaching beliefs and attitudes towards Sustainable Development: A pilot study. 16th International JTEFS/BBCC

Conference Sustainable Development, Culture, and Education. (Yayın No:4487010)

6. Akıncı-Coşgun, A., Kılıç, Ş., Çeken, R., & Köklü, H. (2018). Türkiye'de okul öncesi döneme yönelik çocuk edebiyatı alanında lisansüstü tezlerin incelenmesi. 27. Uluslararası Eğitim Bilimleri Kongresi. (Yayın No:5420126)
7. Köklü, H., Günindi, Y., Kılıç, Ş., Ertürk-Kara, H. G., & Akıncı-Coşgun, A. (2017). Küçük çocukların gözüyle hayvanat bahçeleri. 5th International Early Childhood Education Congress. (Yayın No:4020174)
8. Köklü, H. (2016). Investigation of pre-service teachers' opinions about environmental education. 5. International Conference on Education (Yayın No:2969622)
9. Köklü, H., & Tekin, N. (2016). The opinions of in-service teachers about sustainable energy consumption: Aksaray case. 14th International Teacher Education for Sustainable Development, Culture and Education. (Yayın No:2969709)
10. Köklü, H. (2016). Investigating environmental attitudes and nature relatedness of pre-service teachers in Aksaray University. 14th International Teacher Education for Sustainable Development, Culture and Education. (Yayın No:2969738).

C. Authored national/international books or chapters in books:

C1. Translated national/international books:

1. Bers, M. U. (2021). *Bir oyun alanı olarak kodlama: Erken çocukluk sınıfında programlama ve bilgi işlemsel düşünme.* (Hasret Köklü-Yaylacı, Mehmet Özkaya, & Muhammed Emre Yaylacı, Çev.). (2022). Pegem Akademi, Ankara.

C2. Authored national/international book chapters:

1. Köklü-Yaylacı, H., & Demircan, H. (2023). Bölüm 14: Erken Çocuklukta Stem ve Matematik Eğitimi. Ayşegül Akıncı Coşgun, Hilal Genç Çopur, İlkay Ulutaş (Ed.), *Erken Çocukluk Matematik Eğitiminde Güncel Konular.* Ankara: Nobel Akademik Yayıncılık.
2. Köklü-Yaylacı, H., & Özyıldırım-Gümüş, F. (2022). Chapter 8: Teacher Education in Turkey from Past to Present. Myint Swe Khine (Ed.), *Handbook*

of Research on Teacher Education: Pedagogical Innovations and Practices in the Middle East, pp.121-137. Springer Singapore. Doi: https://doi.org/10.1007/978-981-19-2400-2

3. Köklü-Yaylacı, H. & Feriver-Gezer, Ş. (2020). Bölüm 2: Erken Çocukluk Döneminde Çevre Eğitimi ve Sürdürülebilir Kalkınma için Eğitim. Refika Olgan (Ed.), *Erken Çocukluk Döneminde Çevre Eğitimi.* Ankara: Pegem Akademi Yayıncılık.
4. Köklü, H. (2018). Bölüm 13: Kanada Eğitim Sistemi. Süleyman Yılmaz (Ed.). *Yerelden Evrensele Eğitim Sistemleri (Jeopolitik ve Jeostratejik Perspektifle),* s. 283-303. Ankara: Vize Yayıncılık (Yayın No: 4777802)

D. Full text papers presented at national scientific meetings and published in proceedings books:

1. Köklü, H. & Ünlü-Çetin, Ş. (2014). Okul öncesi öğretmenlerinin yaratıcı dramaya yönelik tutumlar ile öz-yeterliliklerinin incelenmesi. Başkent Üniversitesi 9. Okul Öncesi Eğitimi Öğrenci Kongresi. (Yayın No:1326363)

E. CERTIFICATIONS

Ministry of National Education, Private Doğaç Creative Drama Leadership/Training Program, Contemporary Drama Association and Youth Club

1. Creative Drama Leadership 1st Stage Certificate; Trainers: "Güney Çınar and Hasan Hüseyin Altınova" (27.01.2013-03.02.2013)
2. Creative Drama Leadership 2nd Stage Certificate; Trainers: "Ferhunde Aytaç and Nevin Gümüş" (23.02.2013-12.05.2013)
3. Creative Drama Leadership 3rd Stage Certificate; Trainers: "Songül Başbuğ and Assoc. Prof. Dr. Pınar Özdemir Şimşek" (24.06.2013-02.07.2013)
4. Creative Drama Leadership 4th Stage Certificate; Trainer: "Ebru Turan Vural" (28.07.2013- 19.08.2013)
5. Creative Drama Leadership 5th Stage Certificate; Trainer: "Zeki Özen" (01.10.2013- 21.01.2014)
6. Creative Drama Rapporteur/Assistant Leader in Stage 3, with Creative Drama Leader Sevim Mimtaş (02.2014-05.2014)

F. THESIS PERMISSION FORM / TEZ İZİN FORMU

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural and Applied Sciences ☐

Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences ☒

Uygulamalı Matematik Enstitüsü / Graduate School of Applied Mathematics ☐

Enformatik Enstitüsü / Graduate School of Informatics ☐

Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences ☐

YAZARIN / AUTHOR

Soyadı / Surname : Köklü Yaylacı
Adı / Name : Hasret
Bölümü / Department : Temel Eğitim, Okul Öncesi Eğitimi / Early Childhood Education

TEZİN ADI / TITLE OF THE THESIS (**İngilizce** / English): Effectiveness of KIBO Educational Robotics-Based Activities on Preschoolers' Computational Thinking Skills

TEZİN TÜRÜ / DEGREE**:** **Yüksek Lisans** / Master ☐ **Doktora** / PhD ☒

1. **Tezin tamamı dünya çapında erişime açılacaktır. /** Release the entire work immediately for access worldwide. ☐
2. **Tez iki yıl süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **two years. *** ☐
3. **Tez altı ay süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **six months**. ***** ☒

***** *Enstitü Yönetim Kurulu kararının basılı kopyası tezle birlikte kütüphaneye teslim edilecektir. / A copy of the decision of the Institute Administrative Committee will be delivered to the library together with the printed thesis.*

Yazarın imzası / Signature ..... **Tarih** / Date .....
(Kütüphaneye teslim ettiğiniz tarih. Elle doldurulacaktır.)
(Library submission date. Please fill out by hand.)

Tezin son sayfasıdır. / This is the last page of the thesis/dissertation.