IMPROVING VOCATIONAL STUDENTS' CONCEPTUAL UNDERSTANDING AND COMPUTATIONAL THINKING WITH DIGITAL AND UNPLUGGED PROGRAMMING ACTIVITIES

SELİN SOYDAN

BOĞAZİÇİ UNIVERSITY

2024

IMPROVING VOCATIONAL STUDENTS' CONCEPTUAL UNDERSTANDING AND COMPUTATIONAL THINKING WITH DIGITAL AND UNPLUGGED PROGRAMMING ACTIVITIES

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Educational Technology

by

Selin Soydan

Boğaziçi University

2024

Improving Vocational Students' Conceptual Understanding

and Computational Thinking with Digital and Unplugged Programming Activities

The thesis of Selin Soydan

has been approved by:

Assoc. Prof. Günizi Kartal _____
(Thesis Advisor)

Assist. Prof. Oğuz Ak _____

Prof. Arif Altun _____
(External Member)

September 2024

DECLARATION OF ORIGINALITY

I, Selin Soydan, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature.....

Date .....

ABSTRACT

Improving Vocational Students' Conceptual Understanding

and Computational Thinking with Digital and Unplugged Programming Activities

This study explored and assessed the extent to which vocational students' conceptual understanding and computational thinking can be developed through digital and unplugged activities that focused on specific Computer Science (CS) concepts aligned with the curriculum. Another aim was to obtain deeper understanding of how vocational high school students interacted with and defined CS concepts in a context-specific English vocabulary-integrated learning setting. Six activities were adapted from Grover (2019)'s VELA (Variables, Expressions, Loops, Abstraction) curriculum, and one activity was designed from scratch to include a seventh concept, algorithms, complete with lesson plans. Thus, a total of 7 digital and unplugged activities were carried out at a vocational high school for 3 weeks, as an introduction to the 9th grade fundamentals of programming course at the beginning of the school year. Using the Turkish version of the Computational Thinking Test (CTt) (Polat et al., 2021) and the Computational Thinking Scale developed by Korkmaz et al. (2017) for self-assessment of CT skills, participants' CT skills were assessed before and after the intervention. Grover's (2020) summative assessment tool was adapted for use in a Turkish vocational high school. Semi-structured interviews were conducted to learn more about students' comprehension of CS concepts. The findings showed a significant change in the scores for CTt, but not for the Computational Thinking scale. Students demonstrated a favorable progression in their grasp and application of CS concepts. The implications are explored in terms of computer science education in vocational high schools and programming education in general.

ÖZET

Meslek Lisesi Öğrencilerinin Kavramsal Anlayışlarının ve Bilgi İşlemsel Düşünmelerinin Dijital ve Bilgisayarsız Programlama Etkinlikleriyle Geliştirilmesi

Bu çalışma, müfredatla uyumlu belirli Bilgisayar Bilimleri (CS) kavramlarına odaklanan dijital ve bilgisayarsız etkinlikler aracılığıyla meslek lisesi öğrencilerinin kavramsal anlayışlarının ve bilgi işlemsel düşünmelerinin ne ölçüde geliştirilebileceğini araştırmış ve değerlendirmiştir. Bir diğer amaç ise, meslek lisesi öğrencilerinin bağlama özgü İngilizce kelime dağarcığı ile entegre bir öğrenme ortamında CS kavramlarıyla nasıl etkileşime girdiklerini ve bu kavramları nasıl tanımladıklarını derinlemesine incelemekti. Altı etkinlik Grover'ın (2019) VELA müfredatından uyarlanmış ve bir etkinlik yedinci bir konsept olan algoritmaları içerecek şekilde ders planlarıyla birlikte sıfırdan tasarlanmıştır. Böylece, bir meslek lisesinde, 9. Sınıf programlamanın temelleri dersine giriş niteliğinde, 3 hafta boyunca toplam 7 dijital ve bilgisayarsız etkinlik gerçekleştirilmiştir. Bilgi İşlemsel Düşünme Testi'nin (CTt) Türkçe versiyonu (Polat vd., 2021) ve Bilgi İşlemsel Düşünme Ölçeği (Korkmaz vd., 2017) kullanılarak katılımcılar uygulama öncesinde ve sonrasında değerlendirilmiştir. Grover'ın (2020) sonuç değerlendirme aracı bir Türk meslek lisesinde kullanılmak üzere uyarlanmıştır. Öğrencilerin CS kavram anlayışları hakkında daha fazla bilgi edinmek için yarı yapılandırılmış görüşmeler yapılmıştır. Sonuçlar, CTt puanlarında önemli bir değişiklik olduğunu, ancak öz değerlendirme ölçeğinde bir değişiklik olmadığını göstermiştir. Öğrenciler, CS kavramlarını anlama ve uygulama konusunda olumlu bir ilerleme göstermişlerdir. Bulguların meslek liselerindeki bilgisayar bilimleri eğitimi ve genel olarak programlama eğitimi açısından çıkarımları incelenmiştir.

ACKNOWLEDGMENTS

I would like to convey my heartfelt thanks to Assoc. Prof. Günizi Kartal, as she not only guided me but also provided vital support during this important period of my life. Her valuable recommendations and consistent composure during challenges were priceless.I am grateful and fortunate to have crossed paths with you, ma'am.

I am grateful to Assist. Prof. Oğuz Ak, who let me be part of my first research project. His detailed feedback on the A-VELA test is also greatly appreciated. I want to thank Assist. Prof. Mutlu Şen Akbulut, also my project mentor, for her feedback on the A-VELA test. Thanks to our project team and mentors, I have published my first paper, a significant milestone in my academic journey. Finally, I am thankful to Assist. Prof. Duygu Umutlu for her constructive feedback on the A-VELA test.

I am also grateful to Prof. Arif Altun for providing important resources and perspectives that helped in the creation of this thesis. The guidance and encouragement were instrumental in shaping the direction of my research.

I am as thankful to the fantastic IT teachers Gamze and Özge, who enhanced my learning experience with their openness and support during and after the implementation process. I must remember to thank my friends for their encouragement that helped me finish this thesis. A special thank you also goes out to Ali Eren Kayhan, whose designs made the A-VELA interface possible.

I am eternally grateful to my family – my father, Fahrettin Soydan, my mother, Ebru Soydan, and my sister, Pelin Soydan – for their unwavering support throughout my academic journey and the writing of this thesis. Finally, I want to say how happy I am to be a student at Boğaziçi and how much I value the experience and personal growth it has given me. Hopefully, we'll meet again.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1

1.1 Purpose 4

CHAPTER 2: LITERATURE REVIEW 6

2.1 Computational thinking 6

2.2 Computational thinking and programming 13

2.3 The role of CS conceptual understanding in learning to program 16

2.4 Learners' self-efficacy in programming 24

2.5 Programming instruction in the Turkish vocational high school 26

2.6 Research questions 31

CHAPTER 3: METHOD 33

3.1 Research design 33

3.2 Context and participants 34

3.3 Data collection instruments 35

3.4 Procedures and data collection 37

3.5 Digital & unplugged learning materials 39

3.6 Design framework 43

3.7 Data coding and analysis 44

CHAPTER 4: FINDINGS 47

4.1 Demographic questionnaire 47

4.2 Students' learning gains in CT 48

4.3 Students' perceptions of their own CT competence 50
4.4 Students' scores on A-VELA 51
4.5 Improvements in students' conceptual understanding 52
4.6 Experiences and Perceptions of A-VELA 60
CHAPTER 5: DISCUSSION AND CONCLUSION 63
5.1 Students' learning gains in CT 63
5.2 Students' perceptions of their own CT competence 64
5.3 Improvements in students' conceptual understanding 66
5.4 Student experiences of A-VELA 68
5.5 Limitations & delimitations 69
5.6 Implications and recommendations 70
APPENDIX A: PARTICIPANT INFORMATION FORM 72
APPENDIX B: CONSENT FORM 76
APPENDIX C: SEMI-STRUCTURED INTERVIEW PROTOCOL 79
APPENDIX D: COMPUTATIONAL THINKING SCALE (TURKISH) 81
APPENDIX E: ADAPTED VELA ASSESSMENT (TURKISH) 82
APPENDIX G: SAMPLE A-VELA LESSON PLAN 92
APPENDIX H: EXAMPLE OF A-VELA ACTIVITY (TURKISH) 96
APPENDIX I: DEMOGRAPHIC FORM 96
APPENDIX J: ETHICS COMMITTEE APPROVAL 102
REFERENCES 103

LIST OF TABLES

Table 1. Data Collection and Implementation Process.....50

Table 2. Lesson Procedure……..…….……..….52

Table 3. Details on the A-VELA Activities…….…….…….53

Table 4. Wilcoxon Signed Ranks Test Result for CTt Scores ……..…….60

Table 5. Students' CTt Mean Ranks ……..….61

Table 6. Wilcoxon Signed Ranks Test Result for CT Scale…..…….……62

Table 7. Students' CT Scale Mean Ranks…….…….…….…….63

Table 8. Students' A-VELA Scores……...……...…..64

Table 9. Students' Descriptions for Loops…….…..…….……...65

Table 10. Students' Descriptions for Conditionals…….…….…..……66

Table 11. Student Descriptions for Boolean Operators…….…..…….68

Table 12. Student Descriptions for Algorithms …….……..69

Table 13. Student Descriptions for Variables.…….……70

Table 14. Number of Students with Correct Answers to the Questions..……72

CHAPTER 1

INTRODUCTION

Computational thinking was defined as a "fundamental skill for everyone, not just for computer scientists," (Wing, 2006, p. 33) almost two decades ago, and has since become a widely studied domain of learning. While humans perform computational thinking such as generalizing, or abstracting in everyday activities, learning to program at least at a rudimentary level seems necessary for its development (Grover & Pea, 2013). Computational thinking (CT) has been defined as the thought processes involved in formulating problems and their solutions in a way that enables their representation as computational steps and algorithms (Aho, 2012, p. 832; Wing, 2011). It involves recognizing aspects of computation in the world and applying tools and techniques from Computer Science (CS) to understand and reason about both natural and artificial systems and processes (Royal Society, 2012, p. 29).

CT encompasses several skills for problem-solving; Abstraction, i.e. simplifying complex problems by focusing on essential details while ignoring unnecessary ones; algorithm design to provide step-by-step instructions or procedures to solve problems efficiently; decomposition to break down complex problems into smaller, more manageable sub-problems; automation for developing and utilizing automated processes and tools for efficient, accurate task execution; debugging, i.e. systematic error detection for identifying and fixing errors in algorithms and programs; and generalization, i.e. determining patterns and similarities to reach a solution based on a previous problem. These skills collectively form the foundation of CT, enabling individuals to approach problems systematically

and effectively (Barr & Stephenson, 2011; Lee et al., 2011; Grover & Pea, 2013; Selby & Woollard, 2013; Angeli, 2016).

In general, CT gives a conceptual framework for problem-solving that may be applied outside of the discipline of CS, which provides the technical underpinning for CT (Huang & Looi, 2021). Providing programming instruction in schools will not only facilitate the development of CT skills but also establish a strong foundation in CS and informatics (Belmar, 2022; Bocconi et al., 2016, 2022).

A line of research in CT instruction has concentrated on how CS concepts are taught and learned, as well as the difficulties and misconceptions student comprehension (Grover & Pea, 2013; Grover et al., 2015; Grover et al., 2019). Persistent misconceptions encountered in novice programmers, stemming from the inherent complexity of programming concepts, underscore the need for appropriate pedagogical support to address the challenges in learning to program (Grover et al., 2015; Mladenović et al., 2018). For example, variables and variable assignment have been identified as an area of misconceptions (Mladenović et al., 2018; Žanko et al., 2019; Bayman & Mayer, 1983; Sirkiä & Sorva, 2012; Sorva & Karavirta, 2013; Kohn, 2017). Similarly, Boolean operators, expressions, loops, and arrays seem to challenge, particularly, students in introductory programming courses (Grover & Basu, 2017; Grover et al., 2019; Yukselturk & Altiok, 2017). It has been observed that the main hurdle in learning to program lies not in the syntax and semantics of programming languages, but in comprehending the logic of "putting the pieces together" (Ezeamuzie & Leung, 2022; Soloway & Spohrer, 1989). Insufficient conceptual understanding of these abstract notions seems to hinder the acquisition of procedural knowledge (Almadhoun & Parham-Mocello, 2023; Grover et al., 2019; Hwang et al., 2023). Programming instruction on the other hand, often prioritizes

coding over a thorough understanding of foundational concepts. However, beginner programmers who are unable to acquire essential conceptual understanding, will not be able to master more complex programming techniques (Gallego-Durán et al., 2021; Yeomans et al., 2019). As with other areas of learning, it is important to identify difficulties early on and intervene to provide remediation (Flores & Rodrigo, 2020 ; Kallia & Sentance, 2021) for strengthening beginner programmers' comprehension of ideas and principles (Ma et al., 2023).

Another factor that requires attention at the early stages of programming is that CS jargon and other field specific terminology can be exclusionary for struggling learners, leading to difficulties in concept comprehension. A further difficulty for non-native speakers of English is actually the language itself (Prasad et al., 2022). In countries where English is not the native/primary language, students may face additional challenges as programming languages and platforms are English-based (Alhamami, 2021; Guo, 2018; Yadav et al., 2016). Even if translated into one's native language, the primary commands in a programming platform will still be in English, which leaves the teacher with little ability to use the student's knowledge of everyday concepts and vocabulary to structure instruction around. (Jacob & Warschauer, 2018).

Since the industry standards are also based on English, it is essential to acquire a minimum understanding of vocational English, which makes it necessary to include the teaching of some English vocabulary in the programming curricula and instructional materials. For example, the programming textbooks currently in use in the Turkish vocational high schools inevitably included English terms and their meanings in Turkish to explain how to program using script based (e.g. Python) as

well as block based (e.g. Scratch) platforms (MEB 2020, *Programlamanın Temelleri*).

1.1 Purpose

This study aimed to investigate how conceptual understanding can be developed in beginning programmers, specifically, vocational high school students, through unplugged and digital activities that focus on foundational concepts of CS, and integrate teaching of English vocabulary, whenever needed. It was also important to find out whether or not such an effort would help increase students' CT skills, as well as their perceptions of their own CT competence. For that reason, the study made use of the unplugged and digital activities adopted from Grover et al. (2019)'s VELA curriculum.

As the main focus was improving students' understanding of CS concepts, and not their English language skills, the study assessed CT skills through a Computational Thinking test (CTt) (Román-González et al., 2017) and perceptions of their own CT competence through Computational Thinking Scale which was designed by Korkmaz et al., (2017). In addition, semi-structured interviews was used to assess students' understanding of CS concepts, considering both the cognitive and non-cognitive components of computational thinking (Grover et al., 2015) while also recognizing the need to enhance evaluation methods to investigate diverse modes of knowing (Brennan & Resnick, 2012).

This study further intended to provide an understanding of the acquisition of foundational concepts in programming and the development of CT skills in learners whose mother tongue is not English. The integration of CT and English vocabulary instruction may present a promising approach to addressing the challenges faced by

novice programmers in general, and Turkish vocational high schoolers as they struggle to understand concepts and commands that are not in their native language. It is hoped that this research will help unveil the extent to which focusing on conceptual understanding will help improve learning of CS concepts and CT and will offer insights for educators and teachers for integrating conceptual understanding of CS in the programming curriculum.

CHAPTER 2

LITERATURE REVIEW

This section presents various perspectives and findings on CT, and its role in learning to program, as well as the role of CS conceptual understanding in learning to program. A final section will report findings from research on the learning and instruction of programming in the Turkish vocational high schools.

2.1 Computational thinking

Computational thinking, which has more recently become a popular topic of study, was initially introduced by Papert in 1980's, and as pointed out by Hsu et al. (2018), has been defined, taught, and evaluated by researchers since then. Wing (2006) established the groundwork for the description of CT as a process of solving problems, designing systems, and understanding human behavior by relying on the fundamental concepts of computer science. Needless to say, CT has now been considered to be essential for everyone, not only for computer scientists. (Barr & Stephenson, 2011; Grover et al., 2015; Wing, 2006).

Even though CT has undergone much examination and investigation over the past fifteen years, researchers have not yet agreed on a single definition of CT (Bocconi et al., 2022). The same could be said about the skills and abilities that make up CT. Nonetheless, many researchers defined the composite skills used in formulating and solving a problem which can have a computing solution: abstraction, algorithmic thinking, automation, decomposition, debugging, and generalization (Bocconi et al., 2016; Csizmadia et al., 2015; Grover & Pea, 2013; Kalelioglu et al., 2016; Selby & Woollard, 2013; Wing, 2008). In addition to the composite skills

mentioned, individuals employ logic and utilize various computational concepts such as sequences, loops, events, parallelism, conditionals, operators, and variables to explore and address issues when engaged in computational thinking (Yadav et al., 2016; Allsop, 2019; Román-González et al., 2019; Pugnali et al., 2017).

According to Wing (2008) abstraction is the essence of CT, since CT is based on the abstraction process, which involves determining what components to bring forth and what to disregard. Algorithmic thinking, which is also a vital component of CT, was described by Csizmadia et al., (2015) as the process of reaching a solution by clearly defining the phases. Lee et al. (2011) described automation as a labor-saving method in which a computer is programmed to perform a set of repetitive tasks more rapidly and efficiently than a human. Decomposition is a means of considering artifacts in terms of their constituent elements (Csizmadia et al., 2015). Debugging is the systematic use of analytical and assessment abilities such as testing, tracing, and logical thinking to predict and validate outputs (Csizmadia et al., 2015). Finally, generalization is described as a cognitive process for addressing new problems by recognizing and reusing common parts of a solution and capitalizing on prior knowledge (Selby & Woollard, 2013).

CT is vital in an increasingly information-based environment for various reasons. It entails the mental activity of abstracting problems and constructing automated solutions (Yadav et al., 2014), which can develop the critical thinking abilities required for problem-solving in various academic fields (Tran, 2018; Yadav et al., 2014). CT extends beyond coding and information technology, and involves an analytical style of thinking, similar to mathematical, engineering, and scientific thinking (Corradini et al., 2017; Pugnali et al., 2017). In addition to problem-solving,

CT improves logical reasoning, creativity, predictive thinking, and conceptualization skills across domains and settings (Falloon, 2016; Scherer et al., 2019).

Tran (2018) conducted a mixed-methods study with elementary students (N=132) to assess their STEM perspectives, career choices, and the effects of a CS intervention (N=183) for three months. The computer science coding project lasted ten weeks, with each lesson lasting about 60 minutes. These lessons integrated hands-on CS concepts with online coding puzzles adapted from Course 2 of the elementary framework from code.org and CSUnplugged. The researchers developed and used a 10-item evaluation to examine students' CT skills before and after the intervention on five CS concepts: sequence, algorithm, looping, debugging, and conditionals. They also adapted and developed pre- and post-test surveys based on 44 items on a five-point scale that measured aspirations, beliefs, and goals in STEM; attitudes from parents and peers about STEM; and enjoyment in STEM-related tasks. Semi-structured student interviews were also employed.

The results revealed significant and positive changes in students' perceptions and attitudes toward STEM subjects after participating in the CS coding project. The intervention led to a notable improvement in students' CT skills, as evidenced by pre- and post-test assessments. Qualitative themes from student interviews indicated increased enjoyment and interest in CS lessons, with students personally connecting CS concepts to their future aspirations and potential careers. For instance, one student recognized the relevance of learning algorithms for a future career as a doctor searching for a cure, and another student saw the importance of debugging in her aspirations to become an anesthesiologist. The CS coding project also fostered problem-solving abilities, enhanced mathematical skills, and encouraged creative design, aligning well with operational definitions of CT. Regression analysis showed

significant subject-specific variations in STEM perceptions over time, particularly in science, technology, engineering, and math.

Yadav et al. (2014) conducted a quasi-experimental study examining the impact of computational-thinking modules on preservice teachers' understanding of CT and their attitudes toward computing. The study involved preservice teachers (n=357) at a large mid-western university, with 200 students in the control group and 157 students in the treatment group. The treatment group received a CT module, while the control group did not. The CT module had two classes. The first class introduced students to the definition of computational thinking and five CT concepts: problem identification and decomposition, abstraction, logical thinking, algorithms, and debugging. The second class demonstrated the application of computational thinking in educational settings, showcasing its benefits in K-12 classrooms and encouraging creativity and problem-solving. The researchers collected the data using a computational-thinking quiz with three open-ended questions and a computing attitude questionnaire with 21 Likert-type scale questions, both of which were developed and adapted. The quiz tested students' comprehension of computational thinking, while the questionnaire explored their attitudes toward computing. The responses were categorized into three categories based on participants' perspectives on computational thinking, integrating it into the classroom, and its link to other disciplines.

The results showed that the computational-thinking module had positively influenced preservice teachers' understanding of computational-thinking concepts. Those who received the module viewed CT as a problem-solving approach using algorithmic thinking and abstraction to generate general principles. In contrast, the control group tended to associate computational thinking solely with computers. The

treatment group also recognized the broader applicability of computational thinking beyond computer science and science disciplines, embracing its relevance to other fields like art and English. However, the results regarding preservice teachers' attitudes towards computing were mixed, with no significant differences between the control and experimental groups in comfort and interest in computing. The researchers suggested that the module increased interest in CT concepts, but participants might have felt less comfortable learning these concepts due to their realization of their lack of understanding.

In a recent meta-analysis, Scherer et al. (2019) investigated the transferability of learning computer programming to cognitive skills and found positive transfer effects, indicating that learning programming can have cognitive benefits. The researchers suggested two potential explanations for these findings. Firstly, the cognitive skills assessed in the primary studies shared conceptual commonalities with programming skills, which indicates that learning programming could also support the development of other skills (Liao & Bright, 1991; Shute et al., 2017). These shared subskills may explain the positive transfer effects observed in creative thinking, mathematical skills, and spatial skills. For example, programming was conceptualized as a form of problem-solving, and both programming and mathematical modeling require similar abstraction and application of strategies (Shute et al., 2017). Secondly, the positive effects on cognitive skills might be attributed to the development of metacognitive skills during programming interventions, such as evaluation and monitoring of problem-solving processes (Spohrer & Soloway, 1986).

However, the transfer effects were not present across all cognitive skills, with the far transfer being less pronounced for school achievement and literacy (Scherer et

al., 2019). The positive transfer effects were stronger for creative thinking and mathematical and spatial skills. The effectiveness of learning computer programming in enhancing creativity was comparable to other interventions targeting creativity. Additionally, the transfer effects on mathematical skills were larger than those found in similar meta-analyses focusing on chess instruction, technology-based instruction, music education, or working memory training. Nevertheless, the meta-analysis did not show significant differences in effect sizes between cognitive skills, indicating that the hierarchy of far-reaching transfer effects was not stringent. The findings support that learning computer programming can positively affect certain cognitive skills. Yet, the transferability may vary depending on the cognitive domain and the distinctness of the situations in which the skills are applied.

The studies by Pugnali et al. (2017) and Falloon (2016) also reveal that computational thinking (CT) skills and computer science (CS) concepts benefit a variety of student profiles. Young children (4-7 years) engaging with coding interfaces improved their understanding of sequencing, repeat loops, and conditionals, while 5-6-year-olds integrated coding into their primary school curriculum to enhance general and higher-order thinking skills.

CT has gained recognition in national curricula in compulsory education in many countries, as researchers and educators increasingly agree on its importance as a key objective in K-12 education (Bocconi et al., 2022, 2016; Yadav et al., 2014; Voogt et al., 2015). The integration may vary across educational systems and countries. In countries like the United Kingdom and several European countries, computational thinking has become a mandatory subject in the curriculum (Garrido et al., 2023). Nevertheless, in developing countries, integration might depend more

on individual efforts from organizations and charities rather than being guided by governmental policies.

CT integration into the curriculum is widely recognized as crucial for improving students' problem-solving abilities, critical thinking skills, and computational thinking capacity (Voogt et al., 2015). This requires incorporating computational thinking within the pedagogical framework and integrating it with subject knowledge and pedagogical content knowledge (Yadav et al., 2014).

Voogt et al. (2015) emphasized the incorporation of Computational Thinking (CT) into curricular frameworks in a literature review. They provided an example of a CT framework developed by the College Board for US high schools offering the Computer Science Principles course. The CT framework offers six computational thinking practices that might improve CT skills and benefit learning in computer science. These include; connecting computing, creating computational artifacts, abstracting, assessing problems and artifacts, communicating and collaborating. On the other hand, The UK National Curriculum for Computing aims to foster students' understanding and application of fundamental CS concepts and problem-solving skills through computer programming, and evaluating and applying information technology (Voogt et al., 2015). It also focuses on developing students' abilities, creativity, and understanding of computer science, digital media, and information technology.

The US and UK curricula underscore the importance of incorporating CT into education to enhance students' problem-solving abilities, critical thinking skills, and computational thinking capacity. The development of CT requires not just knowledge of computer science and mathematics but also imaginative capacities like

innovative thinking and a deep, wide-ranging curiosity (Voogt et al., 2015; as quoted in Lohr, 2012).

2.2 Computational thinking and programming

While programming requires the application of CT in practice, CT encompasses more than just programming and includes computational perspectives, human-computer interaction, and ethical considerations (Scherer et al., 2019). To teach and nurture CT, various activities and approaches that emphasize problem-solving, abstraction, algorithmic thinking, and modeling can be employed (Lu & Fletcher, 2009). However, programming is a tangible and practical means to apply and reinforce CT skills, providing a hands-on experience that allows individuals to witness the direct application of CT concepts in software development (Selby, 2012). Programming also plays a vital role in applying and reinforcing computational concepts, such as sequences, loops, parallelism, events, conditionals, operators, and data, making it indispensable for learning and enhancing CT (Lu & Fletcher, 2009; Selby, 2012).

Brennan and Resnick (2012) put forward a thorough theoretical framework for computational thinking. They aimed to explore its practical application and evaluate its adaptability across various programming languages and settings. Their research focused on interactive media designers using Scratch to create this framework. The result was a multi-dimensional approach to understanding and implementing computational thinking, comprising three core components: computational concepts, computational practices, and computational perspectives.

This study placed its emphasis on the Computational Concepts dimension. This dimension encompasses a range of programming concepts, including events, sequences, loops, parallelism, data, operators, and conditionals. The main focus was exploring the intricacies of three core concepts: operators, conditionals, and loops, using a blend of digital and unplugged activities. The objective was to enable students to grasp these fundamental concepts efficiently and effectively, empowering them to apply them to real-world situations.

Research has demonstrated a reciprocal relationship between programming and computational thinking skills, indicating that improvements in one area led to enhancements in the other (Angeli & Giannakos, 2020; Lye & Koh, 2014; Oluk & Korkmaz, 2016). Furthermore, studies have shown that learning programming through visual programming environments or physical robots enhances students' computational thinking abilities, underscoring the interconnectedness of computational thinking and programming skills, as well as the potential transfer of programming skills to foster computational thinking (Brennan & Resnick, 2012 ; Scherer et al., 2019). Integrating computational thinking into education provides an opportunity to enhance critical thinking and problem-solving, emphasizing the problem-solving nature of programming rather than merely writing code (Lockwood & Mooney, 2018; Tran, 2018).

In their research review, Åkerfeldt et al. (2024) indicated that research in programming and computational thinking in education is cross-disciplinary. A substantial portion of the research focused on technologies and how they might be utilized to improve computational thinking and programming skills. However, the ways in which computational thinking can be taught more effectively in the context of K-12 programming teaching remain unclear (Xu et al., 2023).

Sun et al., (2024) sought to evaluate the programming attitudes, computational thinking skills, and programming behaviors of high school students in China, comparing block-based programming modality (BPM) to text-based programming modality (TPM). Utilizing a quasi-experimental design, the study gathered various types of data, such as pre- and post-tests and surveys, to assess students' CT skills and attitudes. The findings indicated that TPM learners made additional syntax errors and had longer pauses between debugging tries, while BPM learners tended to edit code, test various debugging methods, and get involved in other activities. BPM students outperformed TPM students in critical thinking skills, particularly in algorithmic reasoning and collaboration, making the most of the user-friendly features of BPM platforms. And, a small decrease in confidence and enjoyment in both modalities. Yet, BPM students maintained a high level of interest in programming.

Similar findings were highlighted by Tran (2018), Scherer et al. (2019), Brennan and Resnick (2012) as they emphasize the vital role of computer science education and programming in developing computational thinking skills. Tran (2018) illustrated the positive impact of a ten-week coding project on elementary students' attitudes toward STEM subjects and their computational thinking abilities. Scherer et al. (2019) confirmed the transferability of programming skills to cognitive domains, particularly in fostering creativity and mathematical thinking. Brennan and Resnick (2012) revealed the multi-faceted nature of computational thinking development through design-based learning activities like programming with Scratch.

In sum, research findings demonstrated that middle school students' unplugged CS activities improved their CTs using various sub-components and conceptual frameworks (Delal & Oner, 2020; Rodriguez, 2015; Vieira, Gomez, Canu

& Duque, 2023). Studies including programming or informatics may show similar effects (Zhang et al., 2024). Integrating computer science education into curricula can be instrumental in nurturing computational thinking skills and enhancing students' attitudes, cognitive abilities, and long-term aspirations.

2.3 The role of CS conceptual understanding in learning to program

It has been stated that beginning learners struggle with certain foundational concepts in the context of programming (Grover et al., 2015, 2016; Weintrop et al., 2017). Earlier research also pointed out the potential problems of little pedagogical attention to and engagement with conceptual ideas, causing difficulties for beginners on the path to a robust comprehension of programming (Pea & Kurland, 1984).

A poor mental model or conceptual understanding is bound to limit deeper learning of core CS topics (Mayer, 2004; Robins et al., 2003). McGill and Volet (1997) argued that an overemphasis in computer programming education on coding and knowledge of programming syntax, would frequently result in less understanding of computer programming principles and little development of problem-solving skills in CS.

Mladenović et al. (2024) explored the impact of the bridging method pedagogical approach on programming misconceptions occurrence via mediated transfer. The scientists conducted a quasi-experimental study with 163 sixth-grade students in educational settings while they were in computer science lessons. The control group received traditional Python programming lectures, which used a text-based programming language. In contrast, the experimental group utilized a mediated transfer teaching approach, starting with MakeCode for micro:bit and subsequently transitioning to Python programming. The results indicated that the experimental

group was able to reduce confusion in fundamental programming ideas - variables, sequencing, selection, and loops - more effectively than the control group.

In another study, Kwon et al. (2021) investigated the effects of a problem-based CS curriculum on elementary students' CS learning and their attitudes, through a pretest/posttest design. The participants were 200 sixth graders and their four sixth-grade teachers. The curriculum followed a problem-based learning (PBL) approach, and comprised two units, spanning a total of 8 weeks. The first unit focused on introducing block-based coding, while the second unit involved a PBL coding project where students had to find solutions for social issues encountered in their daily school lives. The curriculum emphasized CS concepts such as Sequences, Loops, Events, Conditions, Parallelism, Data, and Operator, and CT practices such as deciding topics, decomposing tasks, developing programs, demonstrating programs. During the first unit, students observed demonstrations of coding tasks by their teacher and then had the opportunity to create their versions of the programs. The PBL project included a warm-up lesson on the topic, brainstorming and research, program design and creation, and a final presentation of the student's teamwork. To assess the student's knowledge of basic computer science and coding principles, the researchers and teachers jointly developed a 14-item multiple-choice assessment. Eight of the test items focused on fundamental CS concepts, such as sequences, loops, events, conditionals, variables, and parallelism, while the remaining six focused on CS practices, specifically debugging errors related to one of the six concepts. After the PBL project, a 16-item survey was conducted to evaluate students' self-efficacy in programming, attitudes toward computer science, and confidence in their CT skills.

The study revealed a positive impact of the problem-based CS curriculum on elementary students' CT skills. Students showed significant improvement in CT after the Scratch lessons, and this progress was sustained throughout the problem-based learning (PBL) activities. The concepts Event and Parallelism were particularly challenging for novice programmers, as evidenced by the lower percentage of correct answers in multiple-choice assessment test. Furthermore, the results indicated distinct learning trajectories based on students' prior knowledge levels. Students with a stronger foundation in CT performed better on all content tests. Although the differences in conceptual understanding between students with high and low prior knowledge decreased after the Scratch lessons, the gaps widened when assessing debugging skills. Additionally, the study revealed that students generally had moderate levels of self-efficacy, positive attitudes toward computer science, and confidence in their CT abilities. However, students with higher prior knowledge consistently scored higher on the attitudinal survey.

Weintrop and Wilensky (2019) investigated the effects of the introductory programming instruction modality (block-based versus text-based) on learners' transition to a professional text-based programming language in a quasi-experimental classroom study in an urban public high school. The study specifically focused on high school introductory computer science classes and consisted of 60 high school students. As part of this research, a specific version of Pencil Code known as Pencil.cc was created. The study included two introductory programming classes that used the same curriculum but in different modalities: a block-based version of Pencil.cc or a text-based version of Pencil.cc. The intervention was implemented in two sections of an already existing Introduction to Programming course. Previously, the course began teaching Java on the first day. The introduction to Java was

postponed until the sixth week of class, with the first five weeks spent utilizing Pencil.cc and the designed curriculum. Following the five-week introduction, students in both modalities moved on to Java. All students utilized the same basic text editor for their Java programming tasks, which did not include standard programming editor features like syntax highlighting or autocompletion.

During the 15-week study, students completed a customized version of Commutative Assessment, which consisted of 30 multiple-choice questions on programming concepts covered in the introductory course, including conditional and iterative logic, variables, functions, program comprehension, and algorithms. Additionally, students completed an attitudinal survey at the beginning, middle, and end of the study. The survey was based on items from the Georgia Computes project and the Computing Attitudes Survey, and it included 10-point Likert scale questions and short response questions. For this study, the researchers focused on questions related to confidence, enjoyment, and students' interest in future computer science courses. Finally, the researchers recorded the programs students wrote and the error messages they received while compiling their programs.

The study revealed that at the beginning of the intervention, students initially achieved greater learning gains in block-based environments compared to text-based alternatives. However, these gains did not lead to significant differences later in text-based instruction regarding conceptual learning, attitudinal shifts, or programming practices since the gap in learning concepts between the two conditions diminished after ten weeks in Java. Additionally, there were no notable differences in programming practices between the conditions during the Java phase, and attitudinal measures converged after ten weeks, resulting in no significant distinction after 15 weeks.

Sun et al. (2021) sought to investigate effective teaching strategies that promote junior high school students' participation in programming activities and the development of their computational thinking (CT) skills. Additionally, their study aimed to assess the influence of unplugged programming activities on improving students' CT skills. They conducted a quasi-experimental study for 8 weeks in the information technology course of junior high school in China. The participants were 93 7th-graders. The participants were divided into three groups. Experimental Groups 1 (n=29) and 2 (n=33) engaged in unplugged programming activities, while Group 3 (n=31), the control group, followed conventional information technology courses based on national curriculum standards. These groups were compared to explore the effectiveness of the unplugged programming teaching approach in promoting CT skill acquisition in junior high school students and examine the influence of various background factors (e.g., gender, the number of ICT devices available, parents' occupation, programming experience, and mathematics performance).

The research comprised three stages: early, middle, and late, with 8 courses developed over two months. Each course lasted 40 minutes, and the same information technology teacher conducted all activities in the three groups. The intervention followed the teaching process of ‘programming knowledge guidance, unplugged programming activity implementation, and computational thinking integration’. The programming knowledge involved sequence, cycle, direction, and condition. The teacher explained the content to students, followed by unplugged programming activities. Then, the teacher guided the students through problem abstraction, algorithmic design, iterative implementation, and summary. Furthermore, the teacher elaborated on the CT-related knowledge involved in these

activities to instill CT concept knowledge in the students, with concepts such as abstraction, decomposition, algorithmic thinking, evaluation, and generalization.

The researchers collected background information factors such as gender, the number of available ICT devices, parents' occupation, programming experience, and mathematics performance. They also constructed The CT skills test using the 2018 and 2019 Bebras Challenge Cadets (12–14 years old) questions. Two sets of CT test questions were selected and designed, with each set containing 12 questions to assess the CT skills of the sample students in the pretest and posttest.

The findings indicate that the unplugged programming significantly improved the CT skills of participating 7th-grade students in both Experimental Group 1 and Experimental Group 2. However, there was no significant improvement in the control group students' CT skills. Students reported during the unplugged activities that their problem-solving abilities improved, leading to better problem-solving skills in their daily lives and enhanced logical thinking. Regarding the factors influencing CT skill development, the study revealed that students' mathematics performance and programming experience played significant roles. Students with previous programming experience demonstrated higher CT skill levels than those without such experience. Similarly, mathematics performance positively impacted students' CT skills, indicating a strong relationship between CT and mathematics. Yet, when analyzing the influence of students' background factors on CT skill improvement during the unplugged intervention process, the study found that these factors, such as gender, parents' occupation, and the number of ICT devices available, did not have a significant influence.

Grover et al. (2019) conducted a mixed-methods study to examine the effectiveness of VELA (Variables, Expressions, Loop, Abstraction) curriculum in

supporting conceptual understanding and engagement with programming concepts among students in three public schools in the US involving grades 6-8 (n=71). Additionally, the study investigated the relationship between student learning and demographic factors such as prior math and English preparation, gender, ethnicity, and grade level. The design of the VELA activities incorporated a constructivist pedagogical approach, which emphasized engaging students with programming concepts before coding. This approach involved using design principles derived from dynamic geometry and leveraging research in the learning sciences to provide scaffolding for students' guided inquiry and exploration of programming concepts (Grover et al., 2019, p. 6, p. 27).The curriculum emphasized abstraction, which integrated elements from looping activities such as execution flow and blocks or chunks of functionality and activities focusing on variables and expressions such as derived and compound data types.

The VELA intervention was integrated into a 20-day introductory programming curriculum in middle schools. It encompassed six VELA activities, four digital activities, and two unplugged activities. Supporting materials such as teacher lesson plans, student worksheets for pair work, and formative assessment review sheets were provided alongside the activities. Additionally, lesson plans were developed to introduce students to foundational Scratch constructs and establish connections with relevant concepts covered in the VELA activities to facilitate the transfer.

The pre-post assessments included nine items and 29 individually gradable sub-items to assess students' comprehension of Scratch code snippets and real-world scenarios through multiple-choice and open-ended questions. Student interviews were conducted twice, both during and after the intervention, involving four students

from each class. Post-VELA surveys utilized Likert-scale questions to evaluate students' engagement and comprehension levels. Additionally, the researchers collected final free-choice Scratch projects to evaluate student performance. These projects encompassed creations by both VELA students, individually and in pairs, and projects from non-VELA students. Teacher interviews and focus group sessions were also conducted with the three teachers involved in the intervention, providing feedback on the VELA activities.

The findings showed that earlier math scores had a substantial predictive effect on pre-scores, showing students with higher math scores did better on the pre-assessment. However, when controlled for pre-scores, this association became non-significant, implying that pre-scores were a good predictor of post-scores independent of prior math performance. Grades 7 and 8 displayed higher average pre-scores when compared to Grade 6. Even though Grades 6 and 8 were in public schools identified as low performing, all three grade levels demonstrated significant improvements.

The analysis of Scratch projects provided compelling evidence of the superior performance of students participating in the VELA intervention compared to their non-VELA counterparts. VELA students' programs exhibited increased complexity and demonstrated a higher utilization of VELA constructs, particularly Boolean operators. The utilization of foundational constructs and relevant VELA concepts in the VELA projects was significantly higher compared to non-VELA projects, further highlighting the effectiveness of the intervention in enhancing students' programming proficiency. However, Post VELA surveys and student interviews revealed that students encountered challenges with abstraction and Boolean variables

and expressions, some of which appeared to be related to experiences with the activities.

2.4 Learners' self-efficacy in programming

Exploring the challenges the students face in programming has offered insights into the various factors that determine programming skills. For example, Özmen and Altun (2014) used a qualitative phenomenological approach to investigate the issues that undergraduate students confront in programming classes. Observations and semi-structured interviews revealed key barriers including lack of practice, incomplete knowledge, and personal issues such as poor motivation and anxiety. The study highlighted the significance of training, foundational knowledge, and positive attitudes towards programming. Students with more prior programming experience showed better performance and self-efficacy.

Similarly, Lung-Chun Chang et al. (2024) investigated the challenges undergraduate students have in programming courses due to the anxiety generated by the perceived complexity of programming concepts and syntax. To address these issues, the researchers provided CT- focused instruction that increased student engagement, decreased anxiety, and improved academic performance. The teaching approach included breaking down the integration of CT into a programming lesson into five steps: identifying the issue, designing a resolution, structuring and assessing, making a flowchart, and creating the program. 197 students were divided into two groups: one group received CT-centered instruction while the other group was taught using a traditional approach. Both pre-tests and post-tests were utilized to assess learning outcomes, along with a questionnaire on motivation and anxiety. Results showed that CT-focused instruction led to better learning outcomes,

increased motivation, and reduced anxiety, especially for female students who were initially found to have lower logical thinking skills than male students.

Yurdakök and Kalelioğlu (2024) studied how teaching text-based programming using a physical programming tool impacted seventh and eighth grade students' computational thinking skills and perceptions on self-efficacy. The study employed a sequential explanatory design. The participants were 85 secondary school students. Within the scope of the study, a physical programming tool called Micro:bit was used to teach Python programming for a period of 6 weeks. Data were collected using the Self-Efficacy Perception Scale for Computational Thinking Skill, Bebras: International Challenge on Informatics and Computational Thinking Tasks, tests focused on programming tools, concepts, and processes, and through semi-structured interview. According to the findings obtained from pretests and posttests, a significant and positive difference was found in the students' computational thinking skills and self-efficacy perceptions towards computational thinking skill. As a result of having received instruction in programming, the students learnt the required programming concepts and processes. Students acquired not just the skills to write correct syntax, and to test and fix code, but also grasped programming concepts like variables, conditional statements, loops, and functions by learning Python programming using a physical programming tool, based on the results of the end of the lesson assessment and post application achievement test.

In line with the above studies, Altun and Mazman (2015) identified patterns in undergraduate students' programming profiles, building on previous observations. They discovered two unique profiles for high and low programming performance groups after examining cognitive and non-cognitive variables. Variables such as verbal memory, mathematical achievement, and perceived programming self-

efficacy were strongly linked to programming ability in both groups. Verbal memory consistently influenced working memory across performance groups, highlighting its importance in programming education.

The studies reported above highlight the complexity of programming problems and the demand for a range of focused instructional approaches. Effective programming training requires strong practice, underlying knowledge, cognitive processes, and positive self-efficacy. Recognizing these aspects can help build instructional techniques that cater to different learning levels and handle unique challenges in programming education.

2.5 Programming instruction in the Turkish vocational high school

Programming instruction is the main area of study in the vocational high schools that specialize in Informatics in Turkey. The Informatics high schools aim to equip students with the necessary knowledge, skills, and competencies over four years in software development and network operations (MEB, 2022). Students are expected to develop skills in algorithms, programming, computer assembly, networking, technical drawing, and three-dimensional design.

The software development branch focuses on block-based programming, object-oriented programming, project creation, database usage, and website design.

In the software development branch, the novice learners start with a course on the Fundamentals of Programming, where they are expected to acquire essential programming knowledge and skills. The course begins with problem-solving techniques, enabling students to grasp the problem-solving process by identifying different problem types, formulating effective solutions using various techniques and constructing algorithms. These algorithms are then meticulously debugged and

visually represented using flowcharts to enhance understanding and facilitate effective problem-solving processes. The course moves on with block-based programming, where students explore the structure and functionalities of a block-based platform. The rest of the course is dedicated to object-oriented programming, where programming fundamentals are covered such as variables, constants, operators, diverse data types, decision and loop structures, functions, string operations, error handling procedures, and file operations.

Although the programming curriculum seems to encompass all that is needed for the 9th graders enrolled in the software department of an Informatics high school, the students do not necessarily graduate with the knowledge and skills aimed for by the Ministry of National Education. While there are multiple reasons associated with this, an important factor seems to be the low level of readiness of 9th graders upon entry, as these schools admit students from the lowest quartile of the general high school entrance exam (Aslan & Küçüker, 2022; ERG, 2022).The dropout rates are among the largest in high schoolers in Turkey, and university entrance percentage is among the lowest (Milli Eğitim İstatistikleri, 2022; ÖSYM YKS Yerleştirme, 2022; Eryılmaz- Ballı & Kartal, 2020). Several studies were conducted to address the learning needs of students enrolled in the Informatics vocational high schools (Binici & Necdet, 2004; Özer, 2020).

In a recent study with 10th grade Informatics high schoolers, Cevahir et al. (2022) focused on student achievement, motivation, and attitudes and examined the impact of using animation-based worked examples (ARAWEs) created with augmented reality (AR), as opposed to traditional paper-based worked examples (TWEs). Pre and posttest data were collected from 94 participants via an achievement test developed by researchers for programming skills, the Turkish

version of The Instructional Materials Motivation Survey by Kutu and Sözbilir (2011), and the AR Applications Attitude Scale by Küçük et al. (2014).

The teaching materials focused on the concept of the "for-loop" and involved animations created in MS PowerPoint, converted to video format with voice-over narration. VUFORIA and UNITY served as the augmented reality software for presenting animations. Additionally, a traditional paper-based material was developed for the control group. The intervention spanned two weeks, with a 3-hour study session for the experimental and control groups each materials and another for the control group using traditional materials. The results demonstrated a clear advantage for the experimental group using the ARAWE materials, over the control group. While step-by-step animations in the ARAWE materials led to improved learning outcomes, it also helped increase the students' level of motivation. The study also revealed a positive correlation between students' attitudes toward AR and academic achievement based on the test scores. Specifically, the level of satisfaction in using ARAWE materials had a significant improvement in students' academic performance. These findings highlight the beneficial effects of innovative and engaging learning environments using current technologies on students' academic performance.

In a study of assessment development, Akçay and Altun (2020) focused on debugging skills of the students in grades 10-11 to address the need for a valid and reliable measurement of debugging performance in Informatics vocational high schools. Debugging was emphasized as an important skill to write efficient and error-free code. Not surprisingly, students with low levels of achievement in general were the one who scored the lowest on their debugging skill performance test.

In a descriptive study, Demir and Cevahir (2020) investigated the relationship between algorithmic thinking and problem-solving skills among participants (n=60) who had completed the programming fundamentals course and enrolled in the Computer Technologies program at Vocational and Technical Anatolian High Schools. The dependent variables included students' problem-solving and algorithmic thinking scores, while the independent factors encompassed gender, programming fundamentals, and mathematics grades. In addition to applying the scanning model research framework, the researchers administered a problem-solving inventory, adapted into Turkish by Taylan (1990) based on the work of Hepner and Petersen (1982), and an algorithm knowledge test developed by themselves.

The findings showed that students majoring in information technology had low levels of algorithmic thinking and problem-solving skills. Notably, algorithmic thinking skills differed according to programming fundamentals course grades and gender but not according to mathematics grades. Problem-solving abilities, on the other hand, did not show significant differences based on gender, mathematics, or programming basics. The researchers emphasized that algorithmic thinking is not confined to program development; it extends to solving challenges across various domains. Thus, enhancing problem-solving skills through applications in academic subjects, especially mathematics, can empower students as active learners.

Cevahir and Özdemir (2017) conducted a study employing the phenomenology analysis method to investigate the difficulties students encounter in programming education. The participants were 17 information technology teachers from four vocational and technical schools. The data were collected through four semi-structured interview questions that facilitated discussions with teachers regarding the challenges observed in programming education, specific problems

encountered by students during the learning process, difficulties associated with teaching programming concepts, and strategies for effective programming instruction.

The thematic analysis of the interview data revealed five themes: "education centered on rote learning," "challenges in comprehending arrays and loops," "repetition and lack of practice," "deficiency in abstract thinking skills," and "inadequacy in arithmetic, mathematical, and analytical thinking." Researchers suggested that students relying on rote learning might encounter programming difficulties. Repetition and practical application gaps have been highlighted by teachers as hindrances to effective learning in programming classes. Moreover, educators widely believed that students' deficiencies in arithmetic, math, and analytical thinking hindered programming instruction. According to the results, teachers have indicated that students struggle with learning array and loop commands in programming instruction. Based on the research findings, teachers have emphasized that students' deficiency in abstract thinking skills constitutes a significant obstacle in programming education.

Despite efforts to address the needs of Turkish vocational high school students, there is a significant gap in the literature regarding their conceptual understanding, learning, and specific requirements. Building upon the curriculum framework outlined by the Ministry of National Education (MEB, 2022) and drawing inspiration from the VELA curriculum (Grover et al., 2019), this study aimed to investigate the development of conceptual understanding among novice programmers, with a specific focus on 9th grade on vocational high school students. The VELA curriculum was selected in this study, because with its focus on variables,

expressions, loops, abstraction and conditionals, it aligned well with the content of the 9th grade Fundamentals of Programming course objectives.

The VELA activities were translated, localized, and modified for the Turkish vocational high school context. The modified activities included the first five modules in VELA, and an additional module on algorithms, so that the content was further aligned with the national curriculum. Both digital and unplugged activities were included, with warm-up exercises before each activity. The language of instruction was Turkish, but context-specific English terminology was included, in preparation for programming instruction based on the Turkish version of Python, with key commands preserved in the original English. While the language barrier was expected to present challenges, the primary focus of the study was the acquisition of fundamental concepts in CS and enhancing CT skills. Consequently, English vocabulary instruction was limited to the commands covered within the curriculum.

2.6 Research questions

1. Is there a significant difference in the mean scores of vocational high school students' Computational Thinking skills before and after participating in unplugged and digital activities that focus on conceptual understanding with incorporated English vocabulary?
2. Is there a significant difference in the mean scores of vocational high school students' perception of their own Computational Thinking competence before and after they participate in these activities?
3. To what extent was conceptual understanding of CS concepts supported through these activities?

4. What are the experiences and perceptions of vocational high school students who participate in the A-VELA activities?

CHAPTER 3

METHOD

The primary objective of this study is to examine the implementation of unplugged and digital activities focusing on conceptual understanding that incorporates English vocabulary tailored to the context of programming fundamentals for vocational high schoolers. The study investigated how this implementation might help beginners at a vocational high school to improve their understanding of CS concepts, and CT skills, as well as perceptions of their own CT competence. The secondary objective is to explore how student artifacts from paper and digital activities, alongside their performance in a summative test, offer insights into their conceptual understanding of computer science concepts throughout the implementation.

3.1 Research design

This research study used a single-group pretest-posttest design and qualitative data to evaluate the effectiveness of English vocabulary integrated A-VELA Activities in enhancing students' computational thinking (CT) skills and programming conceptual understanding. The data was collected in the form of pre and post-test scores using the Turkish version of the CT test (Polat et al.,2021), which was initially designed by Román-González et al. (2017), as well as the Computational Thinking Scale developed by Korkmaz et al. (2017). The VELA Assessment, designed by Grover (2020), was used as a summative measure. Given the multi-faceted nature of conceptual understanding in programming, collecting qualitative data alongside pre- and post-test learning gains was essential. Consequently, semi-structured interviews were carried out with six students specifically chosen based on their scores on the

CTt and the summative test. The interviews were intended to gather valuable insights into how students of varying competence engage with and comprehend the CS concepts in a context-related English vocabulary-integrated learning environment that involves unplugged and digital activities.

3.2 Context and participants

The study was conducted during lesson hours in the Informatics department of a vocational high school in Istanbul. The participants consisted of twelve 9th grade students enrolled in the Informatics department of a vocational high school in Istanbul, and their IT teacher. There were originally 19 students in the classroom; however, one student left after the demographic survey and pre-tests, leaving 18 students. Due to logistical limitations, such as the size of the computer lab and limited number of computers, teachers conducted the classes in groups to optimize their resources. As a result, the number of participants was reduced to12 students who were able to fit the lab at once, even though pre-test data were initially collected from 18 students.

The participants for this study were selected using a combination of purposeful and convenience sampling methods. Initially, six IT teachers from three vocational high schools in the area were contacted and interviewed to determine their interest and willingness to participate in a research study focusing on conceptual understanding. After considering the teachers' alignment with the research goals and their insights into students' learning needs, one school, and one teacher were selected for inclusion in the study. The participants for interviews were also purposefully selected, based on their scores on the CTt test.

3.3 Data collection instruments

Students' CT skills were assessed using the Turkish version of the CT test (Polat et al., 2021) developed by Román-González et al. (2017 before and after the implementation). The test assesses CT skills such as algorithmic thinking, abstraction, automation, generalization, decomposition, and debugging, consisting of multiple-choice questions with four possible answers and only one correct response.

Each question of the test concentrates on one or more of seven computational concepts, arranged in ascending order of difficulty: Basic directions and sequences (4 questions); Loops repeat times (4 questions); Loop Repeat until (4 questions); If simple conditional (4 questions); If/else complex conditional (4 questions); While conditional (4 questions); Simple functions (4 questions). The reliability coefficient for the Turkish version was .80 (Polat et al., 2021), indicating excellent agreement, according to Fleiss (1981).

The students' perceptions of their own computational thinking competence were assessed before and after the implementation using the Computational Thinking Scale developed by Korkmaz et al. (2017), a five-point Likert scale with 29 items and five factors: Creativity, Algorithmic Thinking, Cooperativity, Critical Thinking, and Problem-Solving. Its internal consistency coefficient was 0.822, based on responses from 1,306 university students. This scale was tested with 144 ninth and tenth graders to assess whether the scale can be used with high school students and a Cronbach's Alpha of 0.78 was found, indicating sufficient internal consistency.

The VELA Assessment (Grover, 2020) was used as a summative assessment after the implementation, which included multiple-choice and open-response items, originally based on the block-based programming platform, Scratch. The test consisted of nine questions and sub-questions covering a range of concepts addressed

in VELA Activities. The concepts include Variables (the basic idea of variation, naming, and data types), Arithmetic Expressions, Boolean Logic, Boolean Expressions, Arithmetic and Boolean Expressions, Data Types, String Data, and Modeling. Rubrics for the assessment provided guidelines for grading open-response items, with each question carrying points ranging between 1 and 5. Additionally, the assessment items demonstrated high reliability (82%) based on the factor analyses on the classroom use data (Grover, 2020).

To assess concepts not covered by the Computational Thinking Test, six of the VELA questions were chosen and translated into Turkish, and adapted for the current context. A question about Algorithms and flowcharts was added as a new question to make sure that all concepts in A-VELA are covered in this summative assessment. Five expert opinions were obtained regarding the adaptation of the VELA questions for the A-VELA implementation—two subject area teachers connected to the same high school where the study was conducted and three faculty members from the researcher's university specializing in teaching computing. The first version of the adapted A-VELA assessment was pilot tested at the research site with three 11th-grade students (neither involved in the study). Based on their feedback, customary blocks were included in the A-VELA digital activities, so the students could acquire some familiarity with a Scratch-like interface before they took the A-VELA assessment. The adapted A-VELA was piloted with 60 9th and 10th grade students from various high schools, who were learning to program in Python. A standardized Cronbach's Alpha of 0.70 was obtained based on this pilot data, indicating acceptable internal consistency.

In addition to the quantitative data collected through the above instruments, qualitative data were collected through a semi-structured interview protocol with

selected students, based on their scores on CTt. To ensure that the semi-structured interviews were as accurate and reliable as possible, a protocol was developed (Appendix C). The interview questions sought students' understanding of computer science concepts in the A-VELA program. Their scope was about variables and their utility in programming, the thought process behind examples provided in activities, expectations during exercises, the role of algorithms in programming, examples of logical conditions, initial impressions of questions, and problem-solving strategies.

3.4 Procedures and data collection

The A-VELA implementation was carried out in the 9th grade at a vocational high school in Istanbul during lesson hours. Before the data collection phase, a pilot study was conducted with three 11th grade students. All pilot studies were conducted with voluntary students who were enthusiastic enough to try A-VELA, and who were not going to take part in the intervention.

After the first version was designed, a pilot was conducted in a high school that was selected for this purpose. In total, two pilot studies were conducted, and after each study, feedback was taken from both students and teachers. Based on the feedback received from the two pilots, the last version of the program was developed and implemented.

Teachers and administrators were briefed about the implementation and study objectives, and parental consent for the research. The A-VELA activities were implemented in collaboration with the teacher during the first three weeks of the Fundamentals of Programming course, 2-3 hours per week, a total of 8 lesson hours (40 minutes each). Each A-VELA activity was conducted in two parts: A warm-up activity and a main activity, consisting of unplugged and digital activities. Lesson

plans were prepared for each activity, and discussed with the teacher before the implementation. The researcher provided the teacher and students hard-copy handouts for the unplugged activities, while user data was compiled during the students' interaction with the digital activities.

Due to physical constraints such as the limited size of the computer lab and inadequate number of computers available, only 12 students could participate in the A-VELA implementation, even though 19 participants were initially available. Before collecting data, the study and procedures were explained to the students, and parental consent (Appendix A, Appendix B) was secured. All participants completed the pre-tests: CTt (Polat et al. 2021), CT scale (Korkmaz et al. 2017) and a demographic form (Appendix I) the day before the implementation. After the intervention, students completed the same pretest measures except for the adapted VELA Assessment (Grover et al. 2015), used only in the post-test as a summative measure (see Table 1). The students completed the post-CT competence perception scale (Appendix D) one week after the implementation and the post-CTt.

Finally, six students who scored low, middle, or high (two students for each distinction) in the Computational Thinking Test were selected and participated in semi-structured post-interviews. These interviews focused on students' answers to the unplugged materials and adapted VELA assessment (Appendix E).

Table 1. Data Collection and Implementation Process

Time Schedule	Procedure	Instrument	Duration
Week 1	*Pre-Test*	*Demographic Form,* *Computational Thinking Test,* *Computational Thinking Scale*	10 minutes 45- 50 minutes 15 minutes
Week 2	*Intervention*		3 lesson hours
Week 3	*Intervention*		2,5 lesson hours
Week 4	*Intervention*		2,5 lesson hours
Week 5	*Post Test*	*VELA Assessment* *Computational Thinking Test,*	45-50 minutes 45-50 minutes
Week 6	*Post Test* *Interviews*	*Computational Thinking Scale* *Semi-structured interview protocol*	15 minutes 30 minutes

3.5 Digital & unplugged learning materials

The VELA curriculum designed by Grover et al. (2019) to enhance conceptual understanding was adapted for use with Turkish 9th-grade vocational high school students. As in Grover et al. (2019), the adapted activities focused on specific CS concepts, such as *variables, expressions, loops, and abstraction*. In addition to the original VELA activities, algorithms and flowcharts were included to align with the 9th-grade programing curriculum in the Turkish vocational high schools. With the additional module on Algorithms, the adapted version was called A-VELA.

The instruction aimed to assist students in reconciling the CS concepts with everyday examples, and English instruction was restricted to context-related

vocabulary for programming (See Table 2 for a brief explanation of how each type of activity was implemented in each lesson).

A-VELA consists of seven activities. The unplugged activities include four tasks, each aimed at evaluating different concepts using different types of questions. Task 1, which focuses on variables, consists of sixteen open-ended questions, whereas Task 2, which concentrates on algorithms, consists of two open-ended questions. Task 3 includes three open-ended questions regarding boolean operators with twenty two items in total. In conclusion, Task 4, which also centers on boolean operators, consists of five multiple-choice questions. Therefore, unplugged exercises consist of 11 questions in total for students to complete before they start the digital activity (see Table 3 for details on the activities, programming concepts and assessment).

Each activity was examined twice by an expert and assessed for instructional design, interface, and usability. Based on this feedback, design changes were made, and each activity was re-examined by the same expert to check compliance.

Table 2. Lesson Procedure

Stage	Description
Warm-up	The teacher explains the concept using examples from the students' daily lives, and encourages students to give similar examples. English programming terminology is introduced.
Unplugged Activity	Starts with a “hands-on” activity to provide practice with the concept covered. Students carry out paper-based activities, and check their answers with the teacher and explain their answers.
Digital Activity	Students carry out digital activities on the computer individually. If they have a question, they ask out loud, and other students or the teacher provide help.
Wrap-up	The teacher summarizes the concept covered that day, and links the lesson’s topic with the more general programming concept in A-VELA.

Table 3. Details on the A-VELA Activities

VELA Activity	*Corresponding A-VELA Activity*	*Type*	*Introductory Computing Concepts*	*Data Source for Assessment*
Story Variables	Stories from Space	Unplugged	Variables (basic idea of variation, naming, data types)	Adapted VELA Assessment, Semi - structured interviews
Cats and Ladders	Space Ladder Space Restaurant	Digital	Variables, Arithmetic Expressions	Adapted VELA Assessment Semi - structured interviews
Graphical Looping	Space Loop	Digital	Repeating pattern, Loops, Modeling	CTt (Turkish version) Adapted VELA Assessment, Semi - structured interviews
Dice Conditionals	Weather for Space Launch	Digital	Conditionals	CTt (Turkish version), Adapted VELA Assessment, Semi - structured interviews
Three Toggles	Repair Spaceship System	Digital & Unplugged	Boolean Logic + Boolean Expressions Boolean Outcomes (ON/OFF TRUE/FALSE)	Adapted VELA Assessment, Semi - structured interviews
Alarm Clock	Programming the Space Schedule	Digital & Unplugged	Boolean Logic + Boolean Expressions	Adapted VELA Assessment, Semi - structured interviews

3.6 Design framework

To enhance the learning experience, the digital A-VELA activities were designed based on Mayer's (2014) Multimedia Design Principles. Mayer identifies three cognitive processes required for comprehending scientific explanations using multimedia: selection, organization, and integration. These processes assist in managing many forms of cognitive load during multimedia learning, including extraneous processing, essential processing, and generative processing. Ineffective instructional design leads to extraneous processing, which wastes cognitive resources without promoting learning. Essential processing involves structuring relevant information in working memory, based on the material's complexity. Generative processing includes rearranging information and integrating it with existing knowledge to improve comprehension.

A-VELA's digital activities were designed to manage essential processing, based on the multimedia learning principles relevant for this purpose. The segmenting principle recommends that multimedia content be presented in learner-paced chunks rather than a continuous unit. This allows learners to comprehend each stage at their own pace before going on to the next. As a result, A-VELA digital activities were offered in many segments (levels) that addressed various programming ideas with increasing complexity (see Figure 1). Furthermore, the pre-training principle was utilized, which posits that learners retain multimedia messages more successfully when they are previously acquainted with the names and concepts of important topics. Pre-training focuses on preparing learners to grasp the causal relationships in the content presented. Consequently, in accordance with the pre-training principle, warm-up exercises were incorporated to present programming

terminology in English, the language employed by the Python programming language students were about to begin to learn.

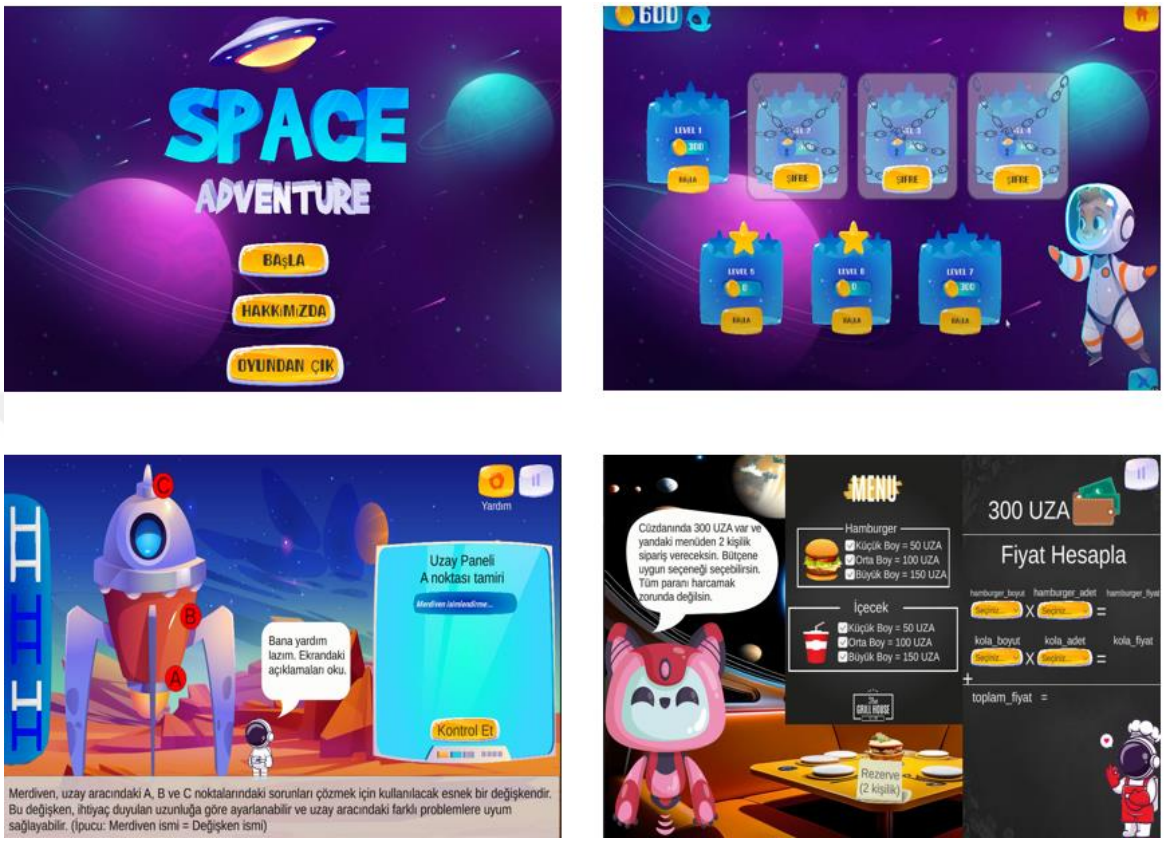


Figure 1. Screenshots from A-VELA digital activities

3.7 Data coding and analysis

To answer the first research question, the scores from the Turkish version of the Computational Thinking test (CTt) were used. The CTt (Turkish) consisted of 28 questions, each worth 1 point, with a maximum possible score of 28. Each correct answer was scored 1 point, while blank or incorrect responses were scored 0. The normality assumption for the differences in CTt scores was assessed through histograms, z-scores for skewness and kurtosis, Q-Q plots, and the Shapiro-Wilk test.

A Wilcoxon signed-rank test was conducted, because the assumption of normal distribution was not met for the CTt This non-parametric test evaluates the

differences between paired observations (i.e., pretest and posttest scores) while considering the ranks of these differences. Upon conducting the Wilcoxon signed-rank test in SPSS, the output provided the test statistic (W), the associated p-value, and effect size measures.

Students' responses to the VELA Assessment, a summative assessment corresponding to VELA curriculum concepts, were also examined. The assessment, initially developed by Grover et al. (2019), was translated and adapted for Turkish students. Due to limited access to all questions in the original test, the researcher made adaptations to ensure alignment with the curriculum, including algorithms and flowcharts. Each response was scored on a 1 to 4 scale using a rubric comprising 19 questions. Descriptive statistics were employed to analyze students' scores, providing students' comprehension levels across the programming concepts.

To address the second research question, the data collected using the Computational Thinking (CT) scale (Appendix C) were used. The CT scale comprised 29 items rated on a five-point Likert scale, ranging from 1 to 5, with the highest rating indicating the strongest agreement and the lowest indicating the weakest agreement.

The scale yields a maximum score of 145 points, with six items being reverse-coded to measure problem-solving skills. Adjustments were made for reverse-coded items to ensure consistency in scoring. A Wilcoxon signed-rank test was conducted, since the CT scale data were not normally distributed.

To address the third research question, selected students' responses (n = 6) to semi-structured interview questions around the A-VELA concepts were analyzed. The responses were then coded to identify students' understanding of Computer Science Concepts.

Each student's data was gathered into a spreadsheet file for coding. In reviewing and categorizing the students' responses the concepts from the A-VELA activities were used, to track any signs of conceptual understanding in student responses to the relevant interview questions. These questions focused on what students think a variable is and how it is used in programming, whether they can define and give an example of a loop, what the concept of an algorithm reminds them of and how it is used in programming, their own description of the algorithm they have designed, how this algorithm works, and whether they can give example of conditionals and statements containing one of the IF functions (AND, OR, NOT).

In accordance with Grover et al. (2019), the responses were coded as definitions of a concept and its application. The teacher who carried out the A-VELA implementation also assessed the students' responses for each concept. Each response was coded for full (1 point) or partial (0.5 points) understanding, and the responses were thus quantified. The researcher and the teacher then met to discuss their rating of each response, and consensus was reached for each response, with 100% agreement. Students were asked whether they had found A-VELA interface useful and if so in what ways. In reviewing and categorizing the students' responses the concepts from the A-VELA activities were used, to track any signs of conceptual understanding in student responses to the relevant interview questions.

To answer the final research question, the students' responses to the interview questions about the A-VELA intervention were tallied, and coded. The three main issues addressed in the interview regarding the activities, usefulness of activities, in what ways, and suggestions for improvement served as the categories for coding the students' responses.

CHAPTER 4

FINDINGS

In this chapter, first, the demographic data will be summarized, and then the findings for each research question will be presented.

4.1 Demographic questionnaire

The demographic questionnaire of 18 students provided information about the students' computer usage habits, level of programming expertise, and attitudes toward going into programming as a career. About 61.1% (11/18) of the students said they had computers at home, but only 5.6% (1) said they used them often, compared to 44.4% (8/18) who said they used them occasionally and 38.9% (7/18) who said they used them seldom. Notably, 88.9% (16/18) of the students used computers for watching TV shows or movies, homework, and academic purposes. A lower percentage of students 22.2% (4/18) reported being involved in programming, compared to the majority 55.6% (10/18) who were gaming. A sizable percentage 83.3% (15/18) felt they did not know enough about programming, even though 72.2% (13/18) said they would like to work in the programming field. Remarkably, of the participants, 44.5% (8/18) knew nothing about programming languages, but 33.3% (6/18) knew Python and 27.8% (5/18) knew Scratch.55.6% (10/18) of respondents who stated they had studied programming languages did so in school. 83.3% (15/18) of the participants had less than a year of experience in programming. Regarding the difficulties in learning programming, there were various responses to the open-ended question; Some students brought up terms like "code" 16.8% (3 out

of 18) "English wording," 5.6% (1 out of 18), and comprehension issues classified as "numerical" 5.6% (1 out of 18) or "difficult to understand" 16.8% (3 out of 18).

On the other hand, 50.4% (9/18) said they had no problems, and one student (5.6%) said it was hard to remember difficulties.

4.2 Students' learning gains in CT

Research Question 1: Is there a significant difference in the mean scores of vocational high school students' Computational Thinking skills before and after participating in unplugged and digital activities that focus on conceptual understanding with incorporated English vocabulary?

The results of the Wilcoxon signed-rank test showed a statistically significant difference in the participants' CTt scores, with a large effect size (r = -0.742), after attending the A-VELA implementation. The Z value was -2.57, with $p < .050$. Median values obtained from the CTt increased from pre-intervention (Md = 12,7) to post-intervention (Md = 14,8) (See Table 4).

Table 4. Wilcoxon Signed Ranks Test Result for CTt Scores

	Post-Score - Pre-Score
Z	-2,571[b]
Asymp. Sig. (2-tailed)	,010

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

As shown in Table 5, 11 students increased their test scores from pre to posttest, while one students' score slightly decreased. There were no ties found, scores improved steadily after the intervention.

Table 5. Students' CTt Mean Ranks

		N	*Mean Rank*	*Sum of Ranks*
CTtPostScore-	*Negative Ranks*	*1*[a]	*6,50*	*6,50*
CTtPreScore	*Positive Ranks*	*11*[b]	*6,50*	*71,50*
	Ties	*0*[c]		
	Total	*12*		

a. Post-Score < Pre-Score

b. Post-Score > Pre-Score

c. Post-Score = Pre-Score

4.3 Students' perceptions of their own CT competence

Research Question 2: Is there a difference in the mean scores of vocational high school students' perception of their own Computational Thinking competence before and after participating in unplugged and digital activities that focus on conceptual understanding with incorporated English vocabulary?

A Wilcoxon signed-rank test conducted on the Computational Thinking (CT) scale scores showed no significant difference before and after the A-VELA implementation. The test resulted in a Z statistic of -1.780, with a two-tailed asymptotic significance (p-value) of. 075 (See Table 6).

Table 6. Wilcoxon Signed Ranks Test results for CT Scale

	CTScalePostTest - CTScalePreTest
Z	-1,780[b]
Asymp. Sig. (2-tailed)	,075

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

Despite a negative trend in the mean scores indicating a possible loss in perception of CT skills competence following the intervention (Md = 100,92) to (Md = 95,08) the test result failed to reach statistical significance at the .05 level. Two students had greater post-scores than pre-scores, while nine had lower post-scores. There was one tie (See Table 7).

Table 7. Students' CT Scale Mean Ranks

		N	*Mean Rank*	*Sum of Ranks*
CTScalePostTest-	*Negative Ranks*	*9*[a]	*5,89*	*53,00*
CTScalePreTest	*Positive Ranks*	*2*[b]	*6,50*	*13,00*
	Ties	*1*[c]		
	Total	*12*		

a. CTScalePostTest < CTScalePreTest

b. CTScalePostTest > CTScalePreTest

c. CTScalePostTest = CTScalePreTest

4.4 Students' scores on A-VELA

Students' scores on summative assessment A-VELA were calculated. To ensure the reliability of the assessment process, two independent raters graded the responses with an agreement of 0.81. This is within the range that suggests high agreement between raters according to Kappa statistics. Based on the analysis of the data from 12 students, the scores ranged from 13 to 38, with an average of 20,56, all students showed some development, at varying degrees, and room for improvement regarding A-VELA concepts (See Table 8).

Table 8. Students' A-VELA Scores

	N	Range	Minimum	Maximum
Students' scores on A-VELA	12	25	13	38
Valid N (listwise)	12			

4.5 Improvements in students' conceptual understanding

Research question 3. To what extent was conceptual understanding of CS concepts supported through these activities?

Interviewees' responses to semi-structured interview questions and classroom observations revealed various insights into their definition and application of Computer Science Concepts. During the lesson, it was observed that four of the students were of the idea that "every repetitive action could be a loop", based on the comments made when asked "what a loop could be".

Table 9 summarizes interviewees' definition of the loop concept in programming. Students were asked how they could "define" loop or give examples regarding what a loop is. Four of the six students interviewed were able to define the concept, and two of them were able to give partial definitions. Of the four students who gave a complete definition, one scored high on the CTt, one scored medium, and the other two scored low. Of the two students who partially defined the concept, one scored medium, whereas the other scored low on the CTt.

Table 9. Student Descriptions for Loops

Concept	Full description	Number of students	Partial description	Number of students
Loops	S102: “Going to school at the same time every morning.”	4	S105: “Things we do all the time.”	2
	S103: “Doing the same things over and over again. Every day, for example, we get up, wash our hands and face and come to school... for example, I take the subway every day…”		S106: “Coming to school because it is something I do every day”	
	S107:”Repeating something several times. When we attend classes, we have a 40-minute class period in the same way. This repeats every time.”			
	S104: “Repetition. Jumping, swimming, the same strokes all the way there.”			

During the lesson, it was observed that three students thought that "then" and "else" meant the exact opposite and were unsure which statement to run in the digital exercise based on their statement. Table 10 examines students' definitions of the concept of "Conditionals”. Four of the six students interviewed were able to define the concept, and two of them were able to give partial definitions. Two of the four students who gave a complete definition scored high on the CTt, one scored medium, and one scored low. Of the two students who partially defined the concept, one scored medium, and the other scored low on the CTt.

Table 10. Student Descriptions for Conditionals

Concept	Full Description	Number of students	Partial Description	Number of students
Conditionals	S102: “If it's raining, take your umbrella, if not, don't.”	4	S103: “If you study, I can buy you a present.”	2
	S105:“Use the first key to open the music cabinet in our room, and if that doesn't work, use the second key.”		S104: “If you've done your homework, you can go out.”	
	S106h: “If the paper is pink, it can be written. If not, it cannot be written.”			
	S107l: “If I have money to buy ice cream, I will look at the price of ice cream and buy ice cream. If not, I will not buy ice cream.”			

During the lesson, it was observed that four students had difficulty selecting the correct operator and understanding its impact on the result, as they couldn't recall the English representations for the operators and sought clarification from the teacher. Table 11 examines Boolean operator examples provided by students, which are logical operators in programming that combine or negate conditions, such as "AND," "OR," and "NOT." The responses are categorized into three levels of description. Of the 6 students who were interviewed, 4 students provided full definitions regarding Boolean operators. One gave partial definition and another student provided an unrelated example. Two of the four students who gave a complete definition scored high on the CTt, one scored medium, and one scored low.

The student who partially defined the concept, scored medium, and the student who provided an unrelated example scored low on the CTt.

Table 11. Student Descriptions for Boolean Operators

Concept	Full Description	Number of students	Partial Description	Number of students	None Irrelevant	Number of students
Boolean Operators	S102: "Take your umbrella if it is dark AND rainy, OR don't take your umbrella if it is sunny."	4	S103: “If you study hard AND get a good grade in the exam, I can buy you a toy OR a chocolate bar.”	1	S107: “I have a dress AND perfume. I have a dress, but if I don't have the dress I want, I'll look for another dress.”	1
	S104: “….. it's not, it's a warning if the seat belt is NOT fastened.”					
	S105: "You need to use key number one OR 7 to open the closet in our room."					
	S106: “The values A AND B turn on the light."					

During the activities, it was observed that a student misunderstood algorithms as the actual coding implementation, as indicated by their response in class. Two students also felt unsure about the relationship between algorithm length and effectiveness, as they pointed out from their classroom examples.

Table 12 explores students' definitions of the concept of "Algorithms," which includes a systematic process for solving a problem or completing a task. The student responses are sorted into three different levels of definition in Table 11.

Three of the six students interviewed were able to define the concept, while two students gave partial definitions. One student gave an irrelevant definition. Of the three students who gave a complete definition, one scored high on the CTt, one scored medium, and one scored low. Of the two students who partially defined the concept, one scored medium, and the other scored high on the CTt. The student who gave an irrelevant answer scored low on the CTt.

Table 12. Student Descriptions for Algorithms

Concept	Full Description	Number of students	Partial Description	Number of students	None Irrelevant	Number of students
Algorithms	S102:“An algorithm, a code, a pseudo-code that is made to solve a problem in everyday life or to solve another problem.”	3	S103: “It could be something in a row.”	2	S104:“I remember what we did in the class. There are shapes”	1
	S107: "Starting something and continuing it step by step. Has a start and a finish."		S106:“Things that go step by step.”			
	S105: “An algorithm is something that shows how to solve a problem and how to do it.”					

During the activities, five students appeared to be unsure about the ranges of variables and giving them a "meaningful" name, as they asked the teacher, "What would it mean for a variable to have a meaningful name?" and stated that they could not decide the range for some of the examples in the unplugged activity. Table 13 gives an overview of students' definition of the programming concept "variables.". Two of the six students interviewed gave a full description of the variables. The other two gave partial definitions. Meanwhile, the remaining two students gave irrelevant definitions. Of the two students who gave full definitions, one scored medium and one scored low on CTt. Of the two students who partially defined the concept, both

scored high on the CTt. Of the two students who provided irrelevant answers, one received a middle score and one received a low score on the CTt.

Table 13. Students' Descriptions for Variables

Concept	Full Description	Number of students	Partial Description	Number of students	None/Irrelevant	Number of students
Variables	S103:"Expressions that are not definite and can change continuously."	2	S106:"Code that can change. To change the code we write what we do wrong and so on."	2	S107:"Variable changeable, can be replaced by something else."	2
	S104:"Variable? She's talking about the same thing, that the price is going up. I mean, you know. Well. Uh-huh. 6 UZA, 7 UZA, 80 UZA a month, which is a bit more fuel price increase."		S102:"It could be storage. Interchangeable storage."		S105: "A variable is something that changes, not a constant ... that is."	

The interviewees' answers to the tasks in the unplugged activities were also evaluated (see Table 14). In Task 1 (variables), 5 out of 12 students answered all 16 questions correctly. The remaining 7 students had a mix of correct and partially correct answers, with 2 students having 10 correct answers, 2 students having 11 correct answers, 1 having 12 correct answers, and the remaining two having 15 correct answers. In Task 2 (Algorithms), 10 students answered both questions

correctly, while 2 students answered neither question correctly due to incomplete algorithms that led nowhere. For Task 3 (Boolean Operators), six students answered all 22 questions correctly. The other 6 students gave partially correct answers, with one student giving 7 correct answers, one giving 8, 2 students giving 12, and 2 students giving 21 correct answers. In Task 4 (Boolean Operators: Multiple Choice), 5 students answered all 5 questions correctly. The remaining 7 students had a mix of correct and incorrect answers, with 6 students answering 4 questions correctly and one student answering 3 questions correctly.

Table 14. Number of Students with Correct Answers to the Questions

Tasks	Total Questions	Correct Answers	Partially correct Answers	Wrong Answers
Task 1 (Variables)	16 open ended	5 students (16/16)	2 students (10/16), 2 students (11/16) 1 student (12/16) 2 students (15/16)	-
Task 2 (Algorithms)	2 open ended	10 students (2/2)	-	2 students (0/2)
Task 3 (Boolean Operators)	22 open ended	6 students (22/22)	1 student (7/22) 1 student (8/22) 2 students(12/22) 2 students (21/22)	-

4.6 Experiences and Perceptions of A-VELA

What are the experiences and perceptions of vocational high school students who participate in the A-VELA activities?

Regarding the interviewed students' experiences and perceptions of the digital A-VELA activities, 4 themes emerged: Prior knowledge, usefulness, programming concepts and interface feedback.

Prior Knowledge

The students' perceptions of their programming knowledge were mostly similar. Five out of the six students had no prior exposure to Scratch or other block-based programming platforms. Even though they were not familiar with it, they still participated in the activities and thought the platform was easy to use. One student (S102), who hadn't used Scratch since 5th-6th grade, mentioned that the platform was easy to use, saying, "Yes, I mean, even someone with no prior knowledge can benefit from it."

Usefulness

The students found the digital activities helpful in improving their understanding of programming concepts. One student (S102) highlighted the platform's usefulness: "Even those with no prior knowledge can benefit from it." Another student (S103) repeated this statement, emphasizing that the activities in class had reinforced their understanding of the fundamental concepts of programming: "Yes, for example, we see the basics of programming with our teacher, and, I understood them, but now I see myself better thanks to the games or tests you do, for example." A student (S104) also expressed a liking for learning through games and stating that they believe it is

more effective to learn in a game format "I think it is - I think it teaches better in game form". Another student (S105) further emphasized the fun and educational benefits of the activities, finding them both enjoyable and useful for enhancing engagement in programming lessons: "It was fun for me, both to show what we understand and also to make the lessons or programming...it could be to make the lesson fun."

Furthermore, a student (S107) appreciated the range of new ideas gained from the activities, stating: “We did good things in the activities, I mean, I learned a lot of different things”.

Programming Concepts

Students commented that practicing A-VELA helped them in grasping concepts regarding programming, particularly loops and algorithms. According to one student (S104), the loops in A-VELA made the programming lesson easier to follow, "Having loops in both the lesson and the game made it easier to understand, especially with the examples provided." The same student also commented on the inclusion of English terminology. "There were also English words present. I believe they were good as well. I did not know all of it." Furthermore, other students mentioned that they appreciated how studying algorithms in A-VELA was useful for them in their programming lesson. A student (S105) remarked, "Algorithms, yes, we had seen them there, and it helped me to do better here." Another student (S106) expressed, "It assisted with the algorithm and more. It also aided in the programming lesson." Building on this, another student (S107l) highlighted how the activities helped with grasping the order and processes of algorithms, especially when going over material covered in previous classes: “Yes, it was easier when we did

algorithms in programming basics. We were doing algorithms here already, and when I came across it in these activities, I could understand it in class more easily.“

Interface Feedback

Students shared their thoughts on the interface, pointing out its strengths as well as its shortcomings. One student (S102) described it as "enjoyable, similar to playing a game". One other participant (S103) mentioned that "the space missions were interesting", while another (S104) expressed "I wish the activities did not end". Some recommendations for improving comprehension and accessibility were given. For instance, a different student (S102) suggested eliminating the need for passwords to make access easier. Another student (S103) reflected on the challenges they faced due to their limited programming skills, suggesting that more time in a programming fundamentals course would have made completing the tasks easier: "I struggled because we had not seen all of programming yet. Maybe with extra time, I could manage it, but..." Moreover, the student S103 suggested adding more interactive elements like visual cues to make the tasks more accessible for peers who have difficulty understanding. "They might get it better using mouse-guided actions or similar functionalities. It could be developed." Another student (S104) expressed a desire for a larger variety of options. "I think there should be more games. I think there should be more content with the games. We enjoy doing them, and we learn something more. It could be longer." Another student (S106) recommended starting these activities in middle school and making them more challenging: "We could have started this a little earlier, in middle school, and so on. It would be a preferable choice to begin sooner. It might also be slightly challenging and not too difficult."

CHAPTER 5

DISCUSSION AND CONCLUSION

The purpose of this study was to find out how novice programmers, specifically vocational high school students, can strengthen their conceptual knowledge using digital and unplugged activities that emphasize the fundamental ideas of computer science and incorporate teaching English vocabulary where necessary. Determining whether or not such an endeavor would contribute to the improvement of students' CT skills and perceptions of their own CT competence was also examined.

The results showed that participation in the implementation of the A-VELA activities helped students significantly improve students' computational thinking skills. In addition, improvement on the programming concepts was visible in the coded student interview data. Regarding students' perceptions of computational thinking competence, there was no significant difference before and after the A-VELA implementation despite the negative trend in scores. This section will discuss the findings of each research question along with limitations and implications.

5.1 Students' learning gains in CT

The results showed that the three week A-VELA implementation improved student learning in CT. Even though the curriculum and activities did not specifically delve into separate CT skills, focusing on foundational CS concepts and computational skills seemed to help students to reinforce both (Tran, 2018 ; Lu & Fletcher, 2009; Selby, 2012). This is in line with the previous research, which found that learning gains in one of these areas could lead to learning gains in the other (Oluk & Korkmaz, 2016; Angeli & Giannakos, 2020; Lye & Koh, 2014).

The increase in the students' CT scores from pre (Md = 12,7) to post-intervention (Md = 14,8) can be assumed as evidence that working on CS concepts can help with CT skills, especially given the fact that the intervention directly addressed only two of the concepts tested on the CT test.

Despite the fact that the A-VELA modules directly addressed only, loops and conditionals, the measured gains provided evidence of A-VELA's possible influence on the learning of other concepts as well. This can be attributed, in part, to the intervention's alignment with the curriculum, with each session reinforcing foundational CS concepts that underlie programming skills. The duration and intensity of the three-week intervention, comprising 2-3 hour sessions per week, seem to have allowed students to engage and practice with the material, and strengthen their understanding. Thus, while the test may not have captured the entirety of the intervention's impact, its focus on some of the fundamental concepts provides a valid measure of students' progress and highlights the intervention's role in enhancing CT skills within the specified scope.

5.2 Students' perceptions of their own CT competence

The results from the CT scale on the students' perception of their own CT skills did not show a significant change after the A-VELA implementation. The analysis revealed a trend of slightly lower scores following the intervention, which did not reach statistical significance. Among the 12 students, only two increased their scores from pre- to posttest, while nine scored lower in the posttest, and one remained the same. This might be due to the challenges in attitudes and perceptions toward CS and programming, influenced by numerous factors. Korkmaz et al (2017)'s CT scale is based on self-report, and requires genuine responses. It is noteworthy that these

scores showed no difference or a slight decrease, while the CTt scores improved significantly, and the results from summative assessment and interviews showed improvement. It can be speculated that the students started realizing what they can and cannot do in terms of CT skills, as they learned more about fundamental concepts of CS, which underlie CT skills. This finding provides support to the argument that assessment of CS and CT skills cannot depend on a single method of assessment, but a holistic perspective and multiple strategies are needed (Tang et al., 2020 ; Román-González et al., 2019 ; Grover et al., 2015 ; Brennan & Resnick, 2012) Also, relying solely on programming contexts to assess self-perceptions of CT skills might not capture the whole picture of skill development, a notion supported by previous research (Lai, 2021 ; Polat et al., 2021; Allsop, 2019 ; Kalelioglu et al., 2016).

When the demographic data are considered it can be seen that the majority, 83.3% (15/18), of the students felt they lacked sufficient programming knowledge despite the fact that a considerable ratio 72.2% (13/18) expressed interest in pursuing a career in the field. These findings require further investigation into potential underlying factors, including the complex nature of CT, differences in the learning styles among individuals, and the necessity for more inclusive instructional strategies to improve self-confidence, particularly among traditionally underrepresented groups such as female students from low SES backgrounds, such as the population of the school where this research was conducted.

5.3 Improvements in students' conceptual understanding

Differences in the comprehension and application of CS concepts were noted in the answers provided by students in semi-structured interviews. Four out of six students were able to give detailed explanations of Loops and Conditionals, showing their grasp of the concepts. Four students showcased their comprehension of Boolean Operators, with one providing an incomplete explanation and another offering a non-relevant description. There were challenges with algorithms; three students provided full descriptions two provided partial descriptions, and one provided an irrelevant description. Variables were the most challenging concept for most students, with only two able to provide full definitions and two others giving incomplete or irrelevant ones. High-scoring CTt students demonstrated conceptual understanding of Boolean Operators and Conditionals, whereas low-scoring students demonstrated conceptual understanding of Loops. Furthermore, one of the two students who scored low on the CTt gave irrelevant responses in Boolean Operators, Algorithms, and Variables, showing either a lack of knowledge or difficulty conveying what they knew. Algorithms were interpreted differently among score groups. Finally, variables were identified as the most difficult concept for students across all score levels, indicating an area that may require more instructional support.

Although the students grasped some programming concepts (such as boolean operators, conditionals and loops), as understood from the fact that they were able to provide definitions or examples regarding the concepts, others (such as variables and algorithms) seemed only partially comprehended or posed problems for self-expression when giving definitions or examples about them.

While variables and algorithms are essential concepts in programming, students may find it difficult to fully understand what they really stand for, even

though examples from their daily lives are provided. It could be because understanding these fundamental ideas requires a higher level of abstract reasoning. It is also possible that linking the daily examples to an abstract concept can be cognitively loaded, as students may have problems at the repacking tail of a semantic wave, after a concept has been unpacked (see NCCE, 2020 for semantic waves). Previous studies has also shown that beginner learners typically struggle with fundamental programming concepts (Grover et al., 2015, 2016 ;Weintrop et al., 2017), as inadequate mental models and conceptual understanding might be barriers to learning more complicated CS concepts (Mayer, 2004; Robins et al., 2003).

Notably, the majority of students were new to programming, they lacked prior knowledge in programming languages, which likely contributed to the observed differences in understanding. Hence, it might be difficult for them to express their conceptual understanding despite successfully applying concepts, algorithms, and variables in paper-based exercises and examples. Instructors have observed that students studying programming struggle with array and loop instructions; they attribute this to a lack of abstract thinking abilities (Cevahir & Özdemir, 2017).

Individual scores on the adapted VELA assessment ranged from 13 to 38, showing that students' conceptual growth is uneven across different levels. This data demonstrates a dynamic learning process where students gradually engage with complex ideas. The differences in accomplishment highlight the intrinsic difficulties in teaching and learning computer science, impacted by educational strategies, past experiences, and individual cognitive abilities (Akçay & Altun,2020 ; Mladenović et al., 2018; Grover et al., 2015; Cevahir & Özdemir, 2017). Targeted instructional interventions meant to close knowledge gaps and improve conceptual mastery require identification of particular areas of difficulty and misconception through a

more thorough study of student data. In order to strengthen a more encouraging learning environment and enhance students' comprehension and application of computer science, additional research into efficient teaching strategies and interventions is necessary.

5.4 Student experiences of A-VELA

The favorable experiences and opinions given by vocational high school students who engaged in digital and unplugged CS activities demonstrated that A-VELA activities were engaging for learners. According to one of the participants, the learning content was accessible, implying that even people with less prior knowledge may benefit from such interactive activities, carefully crafted for the proper level and learner profile. The participants saw that knowledge transfer from A-VELA activities to other programming learning situations was useful. Another participant's comment on how algorithms learned in A-VELA helped them understand the topic in programming lessons more clearly highlights that it has been possible to interconnect the learning content. It is possible to support learning across related areas through digital activities, as indicated by a participant's assessment of the usefulness of concepts acquired through A-VELA activities. This is in line with research that highlights the significance of interesting and user-friendly interfaces in fostering successful learning experiences, and hands-on, applied learning approaches in facilitating conceptual understanding and skill acquisition in CS education (e.g. Cevahir et al., 2022; Tran, 2018; Yadav et al., 2014; Binici & Necdet, 2004).

The importance of continuous iterative design and refinement in digital learning environments to maximize user experience and learning outcomes is highlighted by the participants' suggestions for improving the digital learning

interface, which include eliminating password-protected access and a wider variety of digital activity options. All things considered, the positive experiences and feedback received from high school students who attend vocational schools offer insightful information on how to create and use digital learning platforms that are specifically suited to the varied demands of computer programming students in vocational education settings.

5.5 Limitations & delimitations

A major limitation of the study was the absence of a control group, preventing direct comparison of scores. The lack of a control group would limit the ability to evaluate the overall effectiveness of the A-VELA intervention. A pretest/posttest design was employed to help offset this limitation. Focusing on one group provided in-depth insights into the impact of the intervention on student understanding and use of CS concepts. In addition, due to time constraints and the small number of students, it was not possible to conduct further design iterations, which made it difficult to address certain aspects of the effectiveness of the intervention.

To overcome limitations, a variety of data were collected in order to gain a comprehensive understanding of students' progress in CS thinking. Physical barriers, such as access to a computer lab, posed challenges for some participants, revealing practical barriers associated with implementing technology-enhanced learning environments. It is important to acknowledge that considerable time and effort is required to successfully internalize A-VELA concepts. The A-VELA curriculum emphasizes the complementary role of teacher and the interface, with teachers assuming an active role in curriculum building and making adaptations as

appropriate. This new workload can present challenges for educators, especially those already dealing with demanding schedules and responsibilities.

5.6 Implications and recommendations

The findings of this study have several implications for introducing computer science education in vocational high schools and for teaching computational thinking skills. To begin, the A-VELA program led to a significant improvement in the participants' CT skills, showing the effectiveness of incorporating both online and unplugged activities to reinforce computer science concepts and CT skills for vocational high school students. Nevertheless, it is advisable to lengthen the intervention's timeframe and incorporate additional digital activities into A-VELA in vocational high schools, as per the feedback from certain participants. Extending the duration of the program could lead to students gaining more experience and a better understanding of CS concepts, ultimately improving their CT skills.

Additionally, conflicting results on the participants' perceptions of their computational thinking abilities highlight the need for a more comprehensive curriculum to assess and improve the attitudes of vocational high school students toward computer science and programming. Furthermore, identifying specific learning challenges and misconceptions in computer science concepts highlights the importance of utilizing effective teaching strategies and offering ongoing assistance to vocational high school students. Teachers need appropriate resources and training to create an inclusive learning environment that caters to all students' diverse learning needs.

To further study the efficacy of interventions like A-VELA, it is crucial to carry out extended research in diverse educational environments with bigger sample

sizes, with varied student profiles. Incorporating a control group for comparison would further strengthen the findings. The findings from this research should be read with a caveat, as it was conducted with a small sample size, with a specific student profile. An important goal of this study was to provide educators and policymakers with valuable information on enhancing computer science education in vocational high schools through the integration of technology and effective teaching strategies. By addressing the known challenges and building upon the results of initiatives such as A-VELA, students can improve their preparedness for a highly digital world driven by progressively sophisticated technologies.

APPENDIX A

PARTICIPANT INFORMATION FORM

Araştırmayı destekleyen kurum: Boğaziçi Üniversitesi

Araştırmanın adı: Meslek Lisesi Öğrencilerinin Kavramsal Anlayışlarının ve Bilgi İşlemsel Düşünmelerinin Dijital ve Bilgisayarsiz Programlama Etkinlikleriyle Geliştirilmesi

Proje Yürütücüsü: Doç. Dr. Günizi Kartal

E-mail adresi:

Telefonu:

Araştırmacının adı: Selin Soydan

E-mail adresi:

Telefonu:

Sayın Veli,

Ben Selin Soydan, Boğaziçi Üniversitesi Eğitim Teknolojisi yüksek lisans öğrencisiyim. Tez danışmanım Doç. Dr. Günizi Kartal yöneticiliğinde bir araştırma projesi yürütüyorum. Bu çalışmanın amacı, bilgisayarsız ve dijital etkinlikler aracılığıyla bilgisayar alanı meslek lisesi öğrencilerinin İngilizce sözcük bilgisini de destekleyerek programlama becerilerini geliştirmektir. Çalışmaya katılması için çocuğunuzu projeye davet ediyorum. Aşağıda detaylı bir açıklama bulunmaktadır. Bu bilgileri okuduktan sonra, çocuğunuzun araştırmaya katılmasına izin vermek için bu formu imzalayarak öğretmenine ulaştırmasını rica ederim.

Bu araştırma çerçevesinde çocuğunuzdan üç aşamada katılım sağlanması beklenmektedir: anket, mülakat ve gözlem. Her bir aşama için ayrı bir izin alınması gerekmektedir. Aşağıda her bir aşama için ayrı bir onay kutucuğu bulunmaktadır. Çalışmaya katılan öğrencilerden programlama becerilerini desteklemek amacıyla üç anketi tamamlamaları istenecektir. Ankete katılım ders saatleri içinde gerçekleşecek, her anketin süresi yaklaşık 15 dakika olacaktır. Ankette alınan puanlar temel alınarak daha sonra bazı öğrencilerle görüşmeler gerçekleştirilecektir. Görüşmeler öğrencinin sınıfında, ancak ders saatleri dışında yapılacaktır. Görüşmenin tahmini süresi yaklaşık 15 dakikadır. Görüşme, çocuğunuzun programlama kavramlarıyla ilgili anlayışına dair fikir edinmeyi amaçlamaktadır. Görüşme sırasında ses kaydı alınması planlanmaktadır. Tüm kayıtlar gizli tutulacak, öğrencinin adı ve elde edilen ses kaydı hiçbir şekilde paylaşılmayacaktır. Ses kaydına izin verilmezse, görüşme sırasında not alınacaktır.Görüşme sırasında ses kaydının alınması araştırmanın geçerliliği ve güvenilirliği için önemlidir.

Bu araştırma bilimsel bir amaçla yapılmaktadır ve katılımcı bilgilerinin gizliliği önemlidir. Çalışma sürecinde alınan notlar ve ses kayıtları aracılığıyla varılan sonuçlar sadece bilimsel sunumlarda ve araştırma projelerinde kullanılacaktır. Onam formları ayrı bir yerde kilitli bir dolapta saklanacak ve araştırma tamamlandığında imha edilecektir.

Çalışmaya katılım tamamen sizin ve çocuğunuzun isteğine bağlıdır. Katılım için herhangi bir teşvik sağlanmayacaktır. Ancak istendiği takdirde, araştırma sonuçlarının tüm katılımcılarla e-posta yoluyla paylaşılması planlanmaktadır. Çalışmanın herhangi bir aşamasında, herhangi bir sebep göstermeden katılımınızı geri çekme hakkına sahipsiniz. Katılımın geri çekilmesi durumunda, o zamana kadar

toplanan veri sadece sizin onayınız varsa saklanmaya devam edilecektir. Aksi takdirde, veriler tamamen silinecektir.

Araştırma projesi hakkında daha fazla bilgi almak için Boğaziçi Üniversitesi'nde görevli olan Doç. Dr. Günizi Kartal ile iletişime geçebilirsiniz.

Ben, (velinin adı)..... , yukarıdaki metni okudum ve çocuğumun katılmaya davet edildiği çalışmanın kapsamını ve amacını, gönüllü olarak üzerime düşen sorumlulukları tamamen anladım. Çalışma hakkında soru sorma imkânı buldum. Çocuğumun bu çalışmayı istediği zaman ve herhangi bir neden belirtmek zorunda kalmadan bırakabileceğini ve bıraktığı takdirde herhangi bir olumsuz tutum ile karşılaşmayacağımı anladım.

Bu koşullarda söz konusu araştırmaya kendi isteğimle, hiçbir baskı ve zorlama olmaksızın çocuğumun katılımını kabul ediyorum. Bana anlatılanları ve yukarıda yazılanları anladım. Formun bir örneğini aldım / almak istemiyorum (bu durumda araştırmacı bu kopyayı saklar). Çocuğumun çalışmaya katılmasını kabul ediyorum.

Katılımcının VELİSİNİN

Adı-Soyadı .....

İmzası .....

Tarih (gün/ay/yıl): ...../...../ .....

☐ Çocuğumla anket yapılmasını kabul ediyorum.

☐ Çocuğumla görüşme yapılmasını kabul ediyorum.

☐ Çocuğumla yapılacak görüşmede ses kaydı alınmasını kabul ediyorum.

☐ Çocuğumla gözlem yapılmasını kabul ediyorum.

APPENDIX B

CONSENT FORM

Değerli Öğrenciler,

Ben Selin Soydan, Boğaziçi Üniversitesi Eğitim Teknolojisi yüksek lisans öğrencisiyim. Tez danışmanım Doç. Dr. Günizi Kartal'ın yöneticiliğinde bir araştırma projesi yürütüyorum. Bu çalışmanın amacı, bilgisayarsız ve dijital etkinlikler aracılığıyla İngilizce sözcük bilgisini destekleyerek programlama becerilerinizi geliştirmektir. Sizin de bu çalışmaya katılarak projeye destek olmanızı ve deneyimlerinizden faydalanmamıza yardımcı olmanızı bekliyoruz. Aşağıda projenin detayları bulunmaktadır. Bu bilgileri okuduktan sonra, araştırmaya katılmak için aşağıdaki formu imzalayarak öğretmeninize teslim etmenizi rica ederim.

Bu araştırma çerçevesinde üç aşamada katılım sağlamanız beklenmektedir: anket, mülakat ve gözlem. Her bir aşama için ayrı bir izin alınması gerekmektedir. Aşağıda her bir aşama için ayrı bir onay kutucuğu bulunmaktadır. Çalışmaya katıldığınızda sizden programlama becerilerini desteklemek amacıyla üç anketi tamamlamanız istenecektir. Ankete katılım ders saatleri içinde gerçekleşecek, her anketin süresi yaklaşık 15 dakika olacaktır.

Ayrıca, anketlerde alacağınız puanlara göre, sizlerle görüşme gerçekleştirilecektir. Görüşme, sınıfınızda, ancak ders saatleri dışında yapılacaktır. Görüşmenin tahmini süresi yaklaşık 15 dakikadır. Görüşme, programlama kavramlarıyla ilgili anlayışınıza dair fikir edinmemizi sağlamayı amaçlamaktadır. Görüşme sırasında ses kaydı alınması planlanmaktadır. Tüm kayıtlar gizli tutulacak, adınız ve elde edilen ses kaydı hiçbir şekilde paylaşılmayacaktır. Ses kaydına izin vermezseniz, görüşme sırasında not alınacaktır. Ancak, görüşme sırasında ses kaydının alınması araştırmanın geçerliliği ve güvenilirliği için önemlidir.

Bu araştırma bilimsel bir amaçla yapılmaktadır ve katılımcı bilgilerinin gizliliği önemlidir. Çalışma sürecinde alınan notlar ve ses kayıtları aracılığıyla varılan sonuçlar sadece bilimsel sunumlarda ve araştırma projelerinde kullanılacaktır. Onam formları, ayrı bir yerde kilitli bir dolapta saklanacak ve araştırma tamamlandığında imha edilecektir.

Çalışmaya katılım tamamen sizin isteğinize bağlıdır. Katılım için herhangi bir teşvik sağlanmayacaktır. Ancak istendiği takdirde, araştırma sonuçlarının tüm katılımcılarla e-posta yoluyla paylaşılması planlanmaktadır. Çalışmanın herhangi bir aşamasında, herhangi bir sebep göstermeden katılımınızı geri çekme hakkına sahipsiniz. Katılımın geri çekilmesi durumunda, o zamana kadar toplanan veri sadece sizin onayınız varsa saklanmaya devam edilecektir. Aksi takdirde, veriler tamamen silinecektir.

Araştırma projesi hakkında daha fazla bilgi almak için Boğaziçi Üniversitesi'nde görevli olan Doç. Dr. Günizi Kartal ile iletişime geçebilirsiniz

Ben, (adınız)..... yukarıdaki metni okudum ve katılım için davet edildiğim çalışmanın kapsamını ve amacını, gönüllü olarak üzerime düşen sorumlulukları tamamen anladım. Çalışma hakkında soru sorma imkânı buldum. Bu çalışmayı istediği zaman ve herhangi bir neden belirtmek zorunda kalmadan bırakabileceğini ve bıraktığı takdirde herhangi bir olumsuz tutum ile karşılaşmayacağımı anladım. Bu koşullarda söz konusu araştırmaya kendi isteğimle, hiçbir baskı ve zorlama olmaksızın katılmayı kabul ediyorum.

Bana anlatılanları ve yukarıda yazılanları anladım. Formun bir örneğini aldım / almak istemiyorum (bu durumda araştırmacı bu kopyayı saklar).

Katılımcının Adı-Soyadı .....

İmzası .....

Tarih (gün/ay/yıl): ...../...../ .....

☐Anket yapılmasını kabul ediyorum.

☐Görüşme yapılmasını kabul ediyorum.

☐Görüşme sırasında ses kaydı alınmasını kabul ediyorum.

☐Gözlem yapılmasını kabul ediyorum.

APPENDIX C

SEMI-STRUCTURED INTERVIEW PROTOCOL

Proje: Meslek Lïsesï Öğrencïlerïnin Kavramsal Anlayışlarının Ve Bilgi İşlemsel Düşünmelerinin Dïjïtal Ve Bïlgisayarsiz Programlama Etkinlikleriyle Geliştirilmesi

Röportaj Zamanı: Okul saatleri içinde ders saatleri dışında öğrenciye uygun olan zaman

Yer: Boş bir sınıf

Tarih: Araştırmanın veri toplama aşamasından hemen sonra

Röportajı yapan işi: Selin Soydan

Katılımcı: Anket Sonuçlarına göre seçilen öğrenciler

Katılımcının Rolü:

[Burada görüşülen kişiye (a) çalışmanın amacını anlatarak projeyi tanımlayınız, (b) toplanan verilerin bireyleri ve kaynakları, (c) verilerle ne yapılacağı görüşülen kişinin gizliliğinin korunması ve (d) görüşmenin ne kadar süreceği hakkında kısa bir bilgilendirme yapınız]. [Görüşülen kişiye rıza formunu okutun ve imzalatın]. [Ses kayıt cihazını açın ve test edin.]

Sorular:

1. Değişken sence nedir? Programlama yaparken ne
işe yarayabilir?

2. Şu etkinlikte döngü kavramıyla ilgili şu örneği vermiştin. Bu örnek nasıl akına geldi?

3. Ne açıdan iyi bir örnek? (Öğrencilerin sınıf içi çalışmalarda doldurdukları çalışma kağıtlarından yola çıkarak sorulacak)

4. Peki, bu alıştırmada senden ne yapman bekleniyor ve bu

alıştırmayı yaparken hangi kavram sana yardımcı olabilir?

(Kavramı yanlış uyguladığı alıştırmayı göstererek)

5. Algoritma kavramı sana neyi hatırlatıyor?

Programlamada nasıl kullanılıyor?

6. Şöyle bir algoritma kurgulamıştın bana tarif eder misin,

bu algoritma nasıl çalışıyor?

7. EĞER işlevlerinden birini (VE, VEYA, YADA) içeren bir

koşul örneği verebilir misin?

8. Bu soruyu okurken aklına ilk gelen ne oldu? (Bu sana neyi çağrıştırdı/ nasıl bir çağrışım yaptı?)

9. Bu soruyu çözerken nasıl bir çözüm yolu izledin?

(Bu röportajın amacı öğrencilerin sınıf içi etkinliklere verdiği cevapları derinleştirmek ve bu cevaplar üzerinde düşünmelerini sağlamak olduğundan yukarıdaki sorular öğrencilerin sınıf içi etkinliklere verdiği yanıtlara göre belli olacaktır.)

(Görüşmenin sonunda katılımcılara bu görüşmeye katıldıkları ve araştırmaya katkıda bulundukları için teşekkür edilecek; Yanıtların gizliliği ve gelecekteki görüşmelerin potansiyeli konusunda bilgilendirilecek).

APPENDIX D

COMPUTATIONAL THINKING SCALE (TURKISH)

Bilgisayarca (Bilgi İşlemsel) Düşünme Ölçeği

Aşağıdaki maddeleri dikkatle okuyup, sizi yansıtma düzeyini en olumludan (5) en olumsuza (1) doğru puanlayınız. Katılımınızdan dolayı şimdiden teşekkür ederiz.

Yaratıcılık	Kararlarının çoğundan emin olan insanları severim	1	2	3	4	5
	Gerçekçi ve tarafsız insanları severim	1	2	3	4	5
	Yeterince zamanım olur ve çaba gösterirsem karşılaştığım sorunların çoğunu çözebileceğime inanıyorum.	1	2	3	4	5
	Yeni bir durumla karşılaştığımda ortaya çıkabilecek sorunları çözebileceğime inancım vardır.	1	2	3	4	5
	Bir sorunumu çözmek üzere plan yaparken o planı yürütebileceğime güvenirim.	1	2	3	4	5
	Hayal kurmak, çok önemli projelerimin ortaya çıkmasına neden olur	1	2	3	4	5
	Bir sorunun çözümünde yaklaştığım zaman sezgilerime ve ‘‘doğruluk’’ veya ‘‘yanlışlık’’ hislerime güvenirim	1	2	3	4	5
	Bir sorunla karşılaştığımda, başka konuya geçmeden önce durur ve o sorun üzerinde düşünürüm.	1	2	3	4	5
Algoritmik Düşünme	Bir problemin çözümünü verecek denklemi hemen kurabilirim	1	2	3	4	5
	Matematiksel işlemlere karşı özel ilgimin olduğunu düşünüyorum	1	2	3	4	5
	Matematiksel sembol ve kavramlar yardımıyla yapılan anlatımları daha kolay öğrendiğimi düşünürüm	1	2	3	4	5
	Sayılar arasındaki ilişkileri kolaylıkla yakalayabildiğime inanırım	1	2	3	4	5
	Güncel yaşamda karşılaştığım sorunların çözüm yollarını matematiksel olarak ifade edebilirim.	1	2	3	4	5
	Sözel olarak ifade edilen bir matematik problemini sayısallaştırabilirim.	1	2	3	4	5
İşbirliklilik	Grup arkadaşlarımla birlikte işbirlikli öğrenme deneyimleri yaşamaktan hoşlanırım.	1	2	3	4	5
	İşbirlikli öğrenmede, grupla çalıştığım için daha başarılı sonuçlar elde ettiğimi/edeceğimi düşünüyorum.	1	2	3	4	5
	İşbirlikli öğrenmede grup arkadaşlarımla birlikte grup projesi ile ilgili problemleri çözmekten hoşlanırım.	1	2	3	4	5
	İşbirlikli öğrenmede daha çok fikir ortaya çıkıyor.	1	2	3	4	5
Eleştirel Düşünme	Karmaşık problemlerin çözümüne yönelik düzenli planlar geliştirmede iyiyimdir.	1	2	3	4	5
	Karmaşık problemleri çözmeye çalışmak eğlencelidir.	1	2	3	4	5
	Zorlayıcı şeyler öğrenmeye istekliyimdir.	1	2	3	4	5
	Büyük bir netlikle düşünebilmekten gurur duyuyorum.	1	2	3	4	5
	Elimdeki seçenekleri karşılaştırırken ve karar verirken kullandığım sistematik bir yöntem vardır.	1	2	3	4	5
Problem Çözme	Problemin çözümünü zihnimde canlandırma konusunda sıkıntı yaşarım.	1	2	3	4	5
	Problem çözümünde X, Y gibi değişkenleri nerede ve nasıl kullanmam gerektiği konusunda sıkıntı yaşarım.	1	2	3	4	5
	Tasarladığım çözüm yollarını sırasıyla aşamalı bir şekilde uygulayamam.	1	2	3	4	5
	Bir soruna yönelik olası çözüm yollarını düşünürken çok fazla seçenek üretemem.	1	2	3	4	5
	İşbirlikli öğrenme ortamında kendi düşüncelerimi geliştiremem.	1	2	3	4	5
	İşbirlikli öğrenme grup arkadaşlarıma bir şeyler öğretmeye çalışmak beni yoruyor.	1	2	3	4	5

APPENDIX E

ADAPTED VELA ASSESSMENT (TURKISH)

Soru 1

Program başladığında, boş bırakılan yerlere sırasıyla "0" ve "5" değerleri girildiğinde tilkinin konuşma balonunda ne yazar?





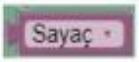
- **A** Hadi sayalım!

  Sayaç!
  Son sayı!

- **B** Hadi sayalım!
  0
  2
  4
  6
  8
  Son sayı!
  8

- **C** Hadi sayalım!
  0
  2
  4
  6
  8
  Son sayı!
  10

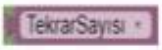
- **D** Hadi sayalım!
  Sayaç!
  Sayaç!
  Sayaç!
  Sayaç!
  Sayaç!
  Son sayı!
  Sayaç!

(b) Programda, döngü içinde 

değeri değişiyor mu?

- **Evet**
- **Hayır**

Evet ise, nasıl değiştiğini açıklayınız.

(c) Programda, döngü içinde 

değeri değişiyor mu?

- **Evet**
- **Hayır**

Evet ise, nasıl değiştiğini açıklayınız.

Soru 2

a. Arda yeni bir bisiklet almak için para biriktiriyor. Arda'nın altı amcası var ve bayram hediyesi olarak her biri ona bir miktar para veriyor. Arda, bisikleti almak için yeterli parası olup olmadığını anlamak için bir plan yapıyor. Aşağıda, Arda'nın planı yazılı olarak verilmiştir:

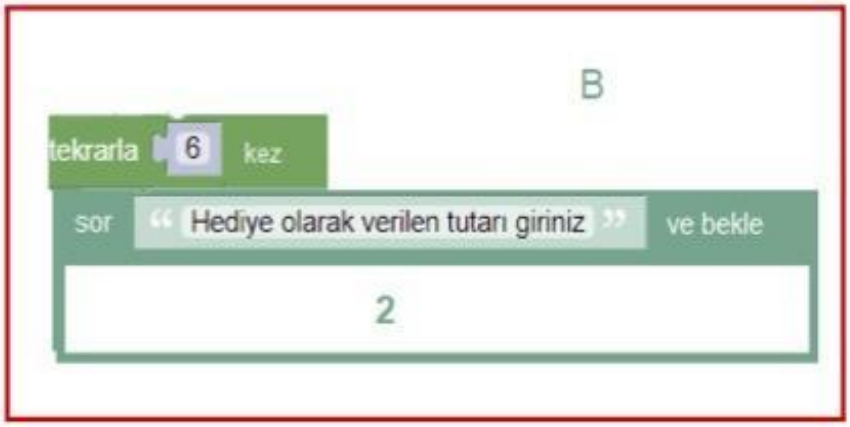
1. Başlangıçta, Arda'nın toplam parası sıfırdır.
2. İlk amcasına gittiğinde amcası ona bir miktar para verir, Arda aldığı parayı toplam parasına ekler.
3. İkinci amcasına gittiğinde o da bir miktar para verir, Arda aldığı parayı toplam parasına ekler.
4. Bu şekilde, tüm amcalarını ziyaret ederek her birinden para alır ve aldığı parayı toplam parasına ekler.
5. Tüm amcalarını ziyaret ettikten sonra, bisikleti almak için yeterli parası olup olmadığını kontrol eder:

- Eğer toplam parası bisikletin fiyatına eşit veya daha büyükse, bisikleti alır.
- Eğer toplam parası bisikletin fiyatından daha küçükse, bisikleti almak için yeterli parası yoktur ve para biriktirmeye devam eder.

Arda'nın planını inceleyin ve algoritmasını akış şeması olarak çizin.

b. Şimdi ise, Arda'nın bisikleti almak için yeterli parası olup olmadığını anlamak için oluşturduğu programa bakalım;





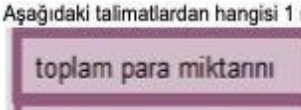


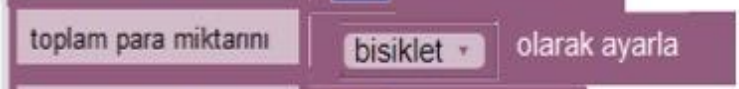
Program üç bölüme ayrılmıştır (A, B ve C). Aşağıdaki tabloda programın hangi bölümünün ne işe yaradığını, karşılık gelen seçenek ile eşleştiriniz.

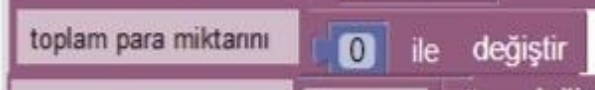
Program ne işe yarıyor:	**Programın hangi bölümü:**
a. Arda'nın amcalarından aldığı para miktarını toplar.	□ A □ B □ C
b. Bu tutarı bisikletin fiyatıyla karşılaştırır ve Arda'ya bisikleti almaya yetecek kadar parası olup olmadığını söyler.	□ A □ B □ C
c. Arda'ya bisikletin fiyatını girmesini söyler.	□ A □ B □ C

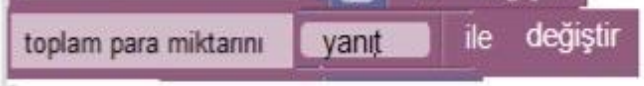
c.Programda hala eksik olan 3 talimat bulunmaktadır (1, 2 ve 3). Şimdi, eksik talimatlara karşılık gelecek doğru seçenekleri işaretleyiniz.

Aşağıdaki talimatlardan hangisi 1 rakamının olduğu yere yerleştirilmelidir?

A. 

B. 

C. 

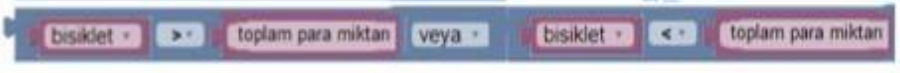
D. 

Aşağıdaki talimatlardan hangisi 2 rakamının olduğu yere yerleştirilmelidir?

A. 

Aşağıdaki talimatlardan hangisi 3 rakamının olduğu yere yerleştirilmelidir?

A. 

B. 

C. 

D. 

Soru 3

Sezen, eğer yağmur yağıyorsa okula giderken yanına bir şemsiye alır. Ayrıca, eğer hava bulutluysa da bir şemsiye alır. Bu havaların dışında (başka havalarda) şemsiye almaz. Bu durumu hangi ifade gösterir?

A. EĞER (yağmurlu VEYA bulutlu) İSE Şemsiye alın
B. EĞER (yağmurlu VE bulutlu) İSE Şemsiye alın

Soru 4

Araç çalışırken emniyet kemeriniz takılı değilse sesli uyarı verir. Bunu hangi ifade gösterir?

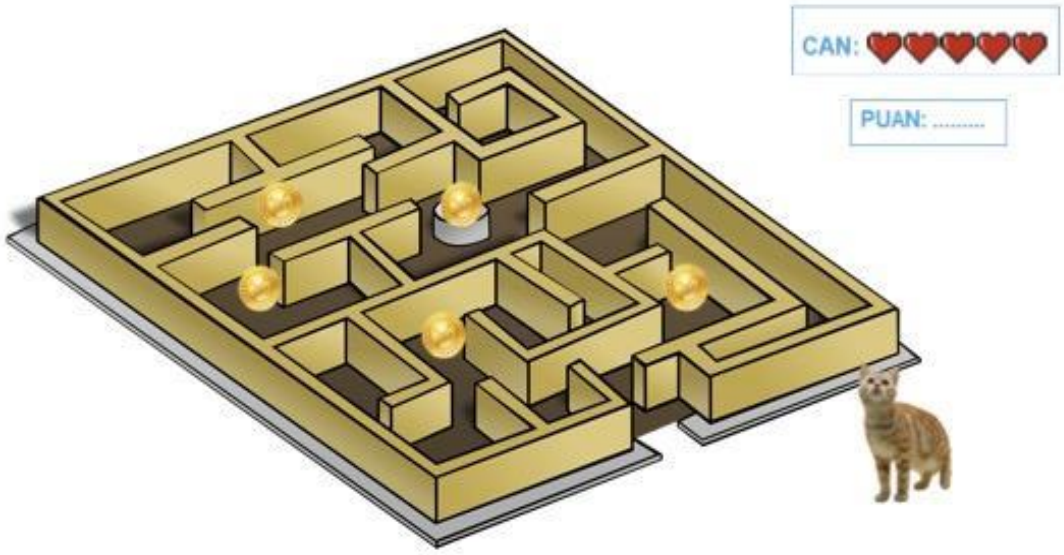
A. EĞER ((Araç çalışıyor) VEYA (Emniyet kemeri takılı DEĞİL)) İSE Uyarı bip sesi
B. EĞER ((Araç çalışıyor) VE (Emniyet kemeri takılı DEĞİL)) İSE Uyarı bip sesi
C. EĞER ((Araç çalışıyor) VEYA (Emniyet kemeri takılı)) İSE Uyarı bip sesi
D. EĞER ((Araç çalışıyor) VE (Emniyet kemeri takılı)) İSE Uyarı bip sesi

Soru 5

Bir labirent oyunu programlıyorsunuz. Karşınızdaki ekranda bir kedi, labirent duvarları ve altınlar var. Siz de altınları bulmaya çalışan bir kedinin ekranda hareket etmesini sağlıyorsunuz. Oyunu her oynadığınızda labirent duvarları ve altınlar aynı yerde kalıyor.

- Oyuna 0 puan ile başlıyorsunuz.
- Başlangıçta 5 canınız var.
- Eğer kedi bir altına dokunursa-
  - Bir can kazanırsınız ve
  - Puanınız 100 artar.
- Eğer kedi bir duvara dokunursa-
  - Bir can kaybedersiniz ve
  - Puanınız 100 azalır.

Oyun, 500 puana ulaştığınızda veya hiç canınız kalmadığında sona erer. Oyun sona erdiğinde puanınız 500 ise kazanırsınız. Oyun sona erdiğinde puanınız 500'den az ise kaybedersiniz.



a.Aşağıdakilerden her biri, siz oyun oynadıkça değişir mi?

Puan	• Değişir	• Değişmez
Can	• Değişir	• Değişmez
Duvar	• Değişir	• Değişmez
Altın miktarı	• Değişir	• Değişmez
Kedinin yeri	• Değişir	• Değişmez

b.Bu oyunu programlamak için gereken bir değişken tanımlayın.

Bu değişkene nasıl bir isim verirdiniz?

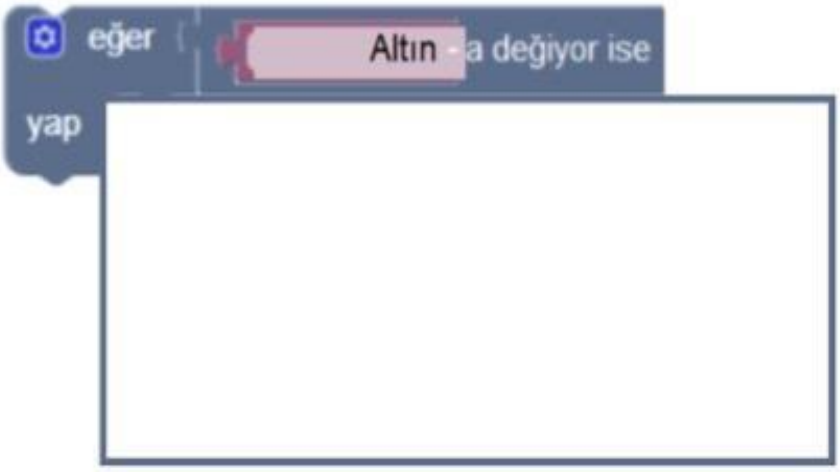
Bu değişkenin amacı ne olurdu?

Bu değişkenin başlangıç değeri ne olurdu?

Değişken İsmi	**Bu değişkenin programda nasıl kullanıldığını açıklayın (bu değişkenin amacı nedir)**	**Değişkenin başlangıç değeri**

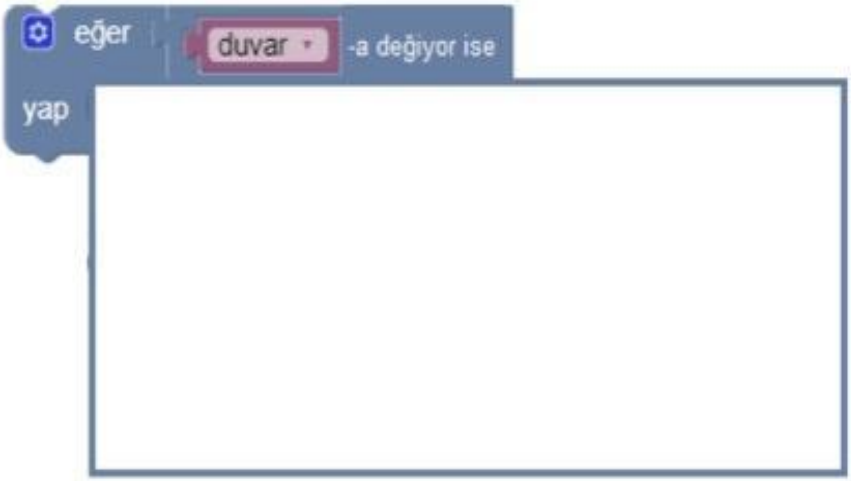
c. Kedi altına dokunduğunda ne olur?

- Aşağıdaki "EĞER" Bloğunu "puan" ve "can" değişkenlerini de kullanarak tanımlayın.
- Oyundaki değişkenlerin değerinin nasıl değiştiğini göstermek için ifadeleri alt alta yazın.



d. Kedi duvara çarptığında ne olur?

- Aşağıdaki "EĞER" Bloğunu "puan" ve "can" değişkenlerini de kullanarak tanımlayın.
- Oyundaki değişkenlerin değerinin nasıl değiştiğini göstermek için ifadeleri alt alta yazın.



e. Oyun ne zaman biter?

Oyunun ne zaman biteceğine karar vermek için aşağıdaki Olana kadar tekrarla döngüsünü tanımlayın.



f. Oyuncunun kazanıp kazanmadığına karar vermek için bir "eğer" ifadesi yazın. (Yazarken değişkenleri kullanın)

Soru 6

Aşağıda bir örüntü gösterilmiştir.



Adım 1 Adım 2 Adım 3

a. Örüntüde bir sonraki adımı (Adım 4) çizin

b. Adım 600 için örüntünün nasıl görüneceğini kelimelerle açıklayın

c. Bu örüntünün 600. Adımını üretmek için bir program yazıyor olsaydınız, aşağıdaki bloklarından HANGİSİNE ihtiyacınız olacağını daire içine alın:

A)



B)



C)



D)



E)



d.Aşağıdaki program, örüntünün 2. Adımını çizer:



Adım 2

Bir yıldız *çiz*
Sağa *taşı*
Bir yıldız *çiz*
Sağa *taşı*
Bir üçgen *çiz*
Sağa *taşı*
Bir üçgen *çiz*

Şimdi bu örüntünün 600.adımı için bir program yazın. Bunun için c.'de seçtiğiniz bloğu kullanabilirsiniz. *Taşı* ve *Çiz* komutlarını yukarıda gösterildiği gibi kullanın.

e. Şimdi, bir kullanıcının girdiği HERHANGİ BİR Adım numarası için örüntü çizmek üzere bir "örüntü oluşturucu" programı yazın. Yukarıda gösterildiği gibi *Taşı* ve *Çiz* komutlarını diğer komutlarla birlikte kullanabilirsiniz.

Programın ilk bölümü şuna benziyor:



Yukarıdaki kod parçasında, AdımSayısı değeri kullanıcının girdiği değere ayarlanan bir değişkendir (Sor komutu ile).
Bu kod bloğunun içinde yer alan komutlarla başlayan programı aşağıda devam ettirin. Programınızda belirli bir adımın değeri yerine AdımSayısı'nı kullanın.

Sor Örüntünün hangi adımını çizmek istiyorsunuz? ve bekle

AdımSayısı değişkenini yanıt olarak ayarla

APPENDIX F

A-VELA RUBRIC

Item	Parts	Concepts	Points
1	a	Variables + Arith. Exp + Loops	2
	b	Variables + Arith Exp + Loops	2
	c	Variables + Loops	1
2	a	Algorithms + Flowcharts	4
	b	Abstraction (Pattern Gen.)	3
	c	For Loop	4
			4
			4
3		Boolean Expression	1
4		Boolean Expression	2
5	a	Variables (V)	5
	b	Variables (V)	3
	c	Variable + Arithmetic Expression	2
	d	Variable + Arithmetic Expression	2
	e	Variable + Boolean Expression	2
	f	Variables + Arithmetic Exp + Loops	2
6	a	Pattern Recognition	2
	b	Abstraction (Pattern Gen.)	2
	c	Loops	1
	d	Loops	4
	e	Variables + Loops	4

APPENDIX G

SAMPLE A-VELA LESSON PLAN

Grade : 9th

Duration: 3 lesson hours (120 mins)

Objectives:

- Students will understand the concept of variables by identifying and grouping them accurately in at least 3 different scenarios within space-themed stories and activities.
- Students will learn about the range and value of numeric data by discussing them in worksheets and digital activities, correctly defining and applying these concepts to given problems with high accuracy.
- Students will practice using meaningful names for variables by naming variables in stories and digital activities, creating names that clearly describe the variable's purpose in at least 2 different contexts.
- Students will gain proficiency in using variables within arithmetic expressions by solving arithmetic problems involving variables, correctly performing operations (addition, subtraction, multiplication) in at least 4 out of 5 instances.
- Students will enhance their programming terminology by discussing and practicing concepts in English, accurately using the terms range, value, and variables in sentences in at least 3 out of 5 instances.

Materials:

- Space-themed images/video about astronauts' life on space
- Worksheet (Includes printed stories + exercises)

- Space Ladder digital activity (students’ computers)

Procedure:

1. Warm-Up (10 minutes)

Teacher: "Good morning, everyone! Are you interested in space? Do you follow the latest developments?"

Show images or a short video about life in space to spark imagination.

Teacher: "Let's imagine life in space... Imagine that it is the year 2093, and a colony has been established on Mars, governed by Elon Musk's son. You can travel to space as easily as traveling from one city to another because you want to celebrate your 85th birthday in space!"

"While exploring, you witness various daily life events. Let's see how these 'Mars-Earthlings' live their daily lives and what they talk about."

2. Story Introduction (10 minutes)

Teacher: "We have four mini-stories about life in space. We'll read and act them out in groups."

Story 1: Fuel Prices in Space

Story 2: Space Shopping

Story 3: Weather News from Earth

Story 4: Earth versus Mars Volleyball Match

3. Unplugged Activity Part I (15 minutes)

Students read all stories one by one and volunteers act out their stories.

Discuss key points and identify variables in each story (e.g., fuel prices, space suit costs).

4. Unplugged Activity Part II (15 minutes)

Distribute worksheets based on the stories.

Students complete worksheets individually or in pairs.

Discuss answers as a class, focusing on variables and their changes.

5. Language Practice (10 minutes)

Review concepts in English: range, value, and meaningful naming.

Write examples on the board and discuss based on the stories.

Example: "The fuel prices range between 2 and 10 UZA."

6. Digital Activity Introduction (5 minutes)

Teacher: "Now, let's help an astronaut complete repairs in space using the Space Ladder activity."

Before moving on to the digital activity, students brainstorm about what a space ladder could be named in a computer program.

7. Digital Activity Part I: Space Ladder (20 minutes)

Stage 1: Numeric Data - Calculate the number of ladders needed for various steps.

Stage 2: Variables - Use different ladder lengths (e.g., 'white_ladder_length = 5' , 'blue_ladder_length= 8') to solve problems.

8. Transition to the Second Digital Activity (2 minutes)

Teacher: " After all the repairing, our astronaut now needs to eat to regain energy. Let's see what the space restaurant is all about."

9. Space Restaurant Menu (5 minutes) + Language Practice (5 minutes)

Discuss the space menu with items priced between 5 and 8 UZA.

Students practice forming sentences about price ranges and variables based on different items.

Example: "The prices range between 5 and 8 UZA for a space burger that includes extra Martian garden sauce."

Example: "The value of the energy drink is 7 UZA."

10. Digital Activity Part II: Space Restaurant (15 minutes)

- Addition - Calculate total prices (e.g., subtotal = LargeBurgerPrice + SmallDrinkPrice).
- Subtraction - Update subtotal when an item is canceled (e.g., subtotal = subtotal - SmallDrinkPrice).
- Multiplication - Calculate total cost for multiple items (e.g., subtotal = (LargeBurgerPrice * numBurgers) + (SmallDrinkPrice * numDrinks)).

Wrap-Up + Real-World Connection (7 minutes)

- Recap the key concepts covered in the lesson: variables, range, value, and meaningful naming.
- Discuss how these concepts apply to programming and real-world situations.
- Teacher: "Think about how you use variables in everyday life. Any examples that come to mind?”

APPENDIX H

EXAMPLE OF A-VELA ACTIVITY (TURKISH)

Hikaye 1: Tüh ya! Sizi Mars'a getiren uzay aracının yakıtının azaldığını fark ettiniz. Sizin uzay aracınız da eski model tabi…bu yüzden bir uzay istasyonuna uğramanız gerekti. Bu istasyonda aracınıza özel uzay yakıtı alırken bir şey fark ediyorsunuz… Yok artık! Uzay yakıtlarına da zam gelmiş! Dünyadan uzaya değişmeyen tek şey…artan fiyatlar olsa gerek Geçen geldiğinizde bir depo uzay yakıtı için ödediğiniz ücret 60 UZA iken şimdi 80 UZA* olmuş. İstasyon görevlisi bunun nedeninin uzay seyahatleri sırasında artan üretim maliyetleri nedeniyle yükseldiğini söylüyor. Siz de değişen fiyatlar hakkında birbirinize dert yanıyorsunuz.

*UZA : Uzay altınımız, Mars'taki para birimimiz.

Hikaye 2: Dünya'dan Mars'a doğum gününüzü farklı bir şekilde kutlamak için o kadar yol geldiniz! Haliyle, doğum gününüzde giymek için, Mars'ın güncel modasına uygun bir giysi ve aksesuar seçmek istiyorsunuz. Kendinize doğum günü hediyesi :) Şansına Mars'ın en popüler giyim mağazası da bugün açıkmış. Çeşitler de oldukça fazla... Tamı tamına 985 çeşit giysi!

Burada farklı tarzda beğendiğiniz uzay elbiseleri olduğunu görüyorsunuz ancak kendi bedeninizi bulamıyorsunuz. Sonuçta bu mağaza sadece insanlara hizmet etmiyor yani. Bu yüzden Mars-Dünyalı satış elemanına bu uzay elbiselerinden biri için önce dünya standartlarına göre kendi bedeninizin kalıp kalmadığını (S), daha sonra da mağazada hangi bedenlerin (XS, M, L,XL) olduğunu soruyorsunuz.

Hikaye 3: Kıyafet tamam! Sıra yemeğe geldi… Mars'ın en Dünya manzaralı restoranı olan Mars Uzay Restoranı'nda kendinize güzel bir yer ayırtmıştınız. Masanıza oturdunuz ve manzaranın tadını çıkarırken mekandaki dev ekranda "Dünyadan Havadisler" adlı programının izlendiğini fark ediyorsunuz. Bu program uzaya Dünyadan haberler getiriyor. Şansınıza, bu haftanın hava durumundan bahsediyorlar. E, siz de uzay yolculuğunuzun dönüşünü Dünyadaki hava durumuna göre planlayacağınız için sesi açıyorsunuz. Görünen o ki hava durumu 7°C ile 15°C arasında değişiyor.

Hikaye 4: Türk Voleybol Milli Takımı, Mars Voleybol Şampiyonası final maçına büyük bir coşkuyla başladı. İki güçlü takımın mücadelesi 0-0 berabere başladı. Ancak Türk takımı, yerçekimsiz ortamı hızla kavrayarak ilk seti 1-0 önde tamamladı.

Rakip takım, pes etmeyip ikinci seti kazanarak skoru 1-1'e getirdi. Ancak Türk voleybol takımı, Ebrar Karakurt'un muhteşem smaçları ve diğer oyuncuların yaratıcı stratejileriyle üçüncü seti 3-1 kazandı. Son sette de yerçekimi avantajını iyi kullanan Türk takımı, şampiyonluğa ulaştı ve Mars'ta bir kez daha Türk bayrağı dalgalanırken milli marşımız çaldı.

Hikayeler:	Hikayede değişen belirli bir öğe veya miktar nedir?	Hikayede değişkenin belirli değerleri nelerdir?	Bu değişken için anlamlı bir isim ne olabilir?	Bu değişkenin alabileceği tüm olası değerler aralığı nedir?
1.Hikaye - Batarya Fiyatları				
2.Hikaye- Aranıyor : Uzay Elbisesi				
3.Hikaye- Dünyadan Hava Durumu				
4.Hikaye- Dünyadan Voleybol				

What's a good name for the story?	Common Words

APPENDIX I

DEMOGRAPHIC FORM

Bu ankette sizin bilgisayar ve programlama geçmişinize yönelik birkaç soru sormak istiyoruz. Katılımınız için şimdiden teşekkür ederiz :)_____

Sınıfınız: _____

Evde bilgisayarınız var mı? ___Evet ___Hayır

Evde İnternet erişiminiz var mı? ___Evet __ Hayır

Bilgisayarı günlük hayatınızda hangi sıklıkta kullanıyorsunuz?

___Her an __ Sık Sık __ Bazen __ Nadiren

Bilgisayarı ilk kez ne zaman kullanmaya başladınız? _____

Bilgisayarı hangi amaçlar için kullanıyorsunuz?

Uygun olanların tümünü işaretleyin.

- Ödevlerimi yapmak için
- Oyun oynamak için
- Dizi/film izlemek için
- Programlama/Kodlama için
- Diğer:

Programlama konusunda yeterli bilgiye sahip olduğunuzu düşünüyor musunuz?

___Evet __ Hayır

Programlamayı daha iyi öğrenmek ister miydiniz? ___Evet __ Hayır

Neden? _____

İleride programla ile ilgili bir meslekte çalışmayı düşünüyor musunuz? ___Evet __ Hayır Diğer: _____

Hangi programlama dillerini biliyorsunuz?

Uygun olanların tümünü işaretleyin.

- Python
- Scratch
- C#
- C++
- Java
- HTML & CSS
- Diğer

Bildiğiniz programlama dillerini nereden öğrendiniz?

- Okulda
- Kursta
- İnternet aracılığıyla
- Diğer: ……

Ne kadar süredir programlama ile uğraşıyorsunuz?

- 1 yıldan daha az
- Yaklaşık olarak 1 yıl
- Yaklaşık olarak 2 yıl
- 2 yıldan daha fazla

Programlama öğrenirken karşılaştığınız en büyük zorluk nedir?

……

…….

Programlama öğrenirken İngilizcenin önemli bir etken olduğunu düşünüyor musunuz?

- Evet
- Hayır
- Diğer:

APPENDIX J

ETHICS COMMITTEE APPROVAL

Evrak Tarih ve Sayısı: 02.07.2023-133978

T.C.
BOĞAZİÇİ ÜNİVERSİTESİ
SOSYAL VE BEŞERİ BİLİMLER YÜKSEK LİSANS VE DOKTORA TEZLERİ ETİK İNCELEME KOMİSYONU
TOPLANTI KARAR TUTANAĞI

Toplantı Sayısı	:	46
Toplantı Tarihi	:	14.06.2023
Toplantı Saati	:	14:00
Toplantı Yeri	:	Zoom Sanal Toplantı
Bulunanlar	:	Prof. Dr. Feyza Çorapçı, Dr. Öğr. Üyesi Ayşegül Metindoğan, Doç. Dr. Senem Yıldız
Bulunmayanlar	:	Doç.Dr. Arhan S. Ertan, Dr. Öğr. Üyesi Yasemin Sohtorik İlkmen

Selin Soydan
Bilgisayar ve Eğitim Teknolojileri Eğitimi Bölümü

Sayın Araştırmacı,

"Developing Vocational High Schoolers' Conceptual Understanding of Programming Through Unplugged/Digital Activities integrated with English vocabulary" başlıklı projeniz ile ilgili olarak yaptığınız SBB-EAK 2023/50 sayılı başvuru komisyonumuz tarafından 14 Haziran 2023 tarihli toplantıda incelenmiş ve uygun bulunmuştur.

Bu karar üyelerin toplantıya çevrimiçi olarak katılımı ve oy birliği ile alınmıştır. Onay mektubu üye ve raportör olarak Yasemin Sohtorik İlkmen tarafından toplantıya katılan bütün üyeler adına e-imzalanmıştır.

Saygılarımızla, bilgilerinizi rica ederiz.

Dr. Öğr. Üyesi Yasemin
SOHTORİK İLKMEN
Öğretim Üyesi

e-imzalıdır
Dr. Öğr. ÜyesiYasemin Sohtorik
İlkmen
Öğretim Üyesi
Raportör

SOBETİK 46 14.06.2023

Bu belge, güvenli elektronik imza ile imzalanmıştır.

REFERENCES

Åkerfeldt, A., Kjällander, S. & Petersen, P (2024) A research review of computational thinking and programming in education, *Technology, Pedagogy and Education*, 33:3, 375-390, DOI: 10.1080/1475939X.2024.2316087

Aho, A. V. (2012). Computation and computational thinking. *Journal of the ACM*, *59*(3), 32:1-32:23.

Akçay, A., & Altun, A. (2020). Test development for debugging performance: Validity and reliability study. *Erzincan Üniversitesi Eğitim Fakültesi Dergisi*, *23*(3), 667-685.

Alhamami, M. (2021). English as a medium of instruction (EMI) in computing undergraduate programs. *Education and Information Technologies*, *26*(5), 6549-6562.

Allsop, Y. (2019). Assessing computational thinking process using a multiple evaluation approach. *International Journal of Child-Computer Interaction*, *19*, 30-55.

Almadhoun, E., & Parham-Mocello, J. (2023). Students' difficulties with inserting and deleting nodes in a singly linked list in the C programming language. *Journal of Computer Languages*, *74*, 101184.

Angeli, C. (2016). A review of the intersection of computational thinking and computer science education research. *In Proceedings of the 2016 ACM* Conference on International Computing Education Research (pp. 9-16).

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*, 106185.

Altun, A., & Mazman, S. G. (2015). Identifying latent patterns in undergraduate Students' programming profiles. *Smart Learning Environments*, *2*, 1-16.

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.

Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of BASIC programming statements. *Communications of the ACM*, *26*(9), 677-679.

Belmar, H. (2022). Review on the teaching of programming and computational thinking in the world. *Frontiers in Computer Science*, *4*.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site).

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M. A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. and Stupurienė, G. (2022). *Reviewing computational thinking in compulsory education*. Inamorato Dos Santos, A., Cachia, R., Giannoutsou, N. and Punie, Y. editor(s), Publications Office of the European Union, Luxembourg, ISBN 978-92-76-47208-7, doi:10.2760/126955, JRC128347.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).

Cevahir, H., Özdemir, M., & Baturay, M. H. (2022). The effect of animation-based worked examples supported with augmented reality on the academic achievement, attitude and motivation of students towards learning programming. *Participatory Educational Research*, *9*(3), 226-247.

Chang, L. C., Lin, H. R., & Lin, J. W. (2024). Learning motivation, outcomes, and anxiety in programming courses—A computational thinking–centered method. *Education and Information Technologies*, *29*(1), 545-569.

Corradini, I., Lodi, M., & Nardelli, E. (2017, August). Conceptions and misconceptions about computational thinking among Italian primary school teachers. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 136-144).

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking-A guide for teachers.

Delal, H., & Oner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. *Informatics in Education*, *19*(1), 1-13.

Ezeamuzie, N. O., & Leung, J. S. (2022). Computational thinking through an empirical lens: A systematic review of literature. *Journal of Educational Computing Research*, *60*(2), 481-511.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576-593.

Fleiss, J. (1981). Statistical methods for rates and proportions. New York: Wiley

Flores, R. M., & Rodrigo, M. M. T. (2020). Wheel-spinning models in a novice programming context. *Journal of Educational Computing Research*, *58*(6), 1101-1120.

Gallego-Durán, F. J., Satorre-Cuerda, R., Compañ-Rosique, P., Villagrá-Arnedo, C. J., Molina-Carmona, R., & Llorens-Largo, F. (2021). A low-level approach to improve programming learning. *Universal Access in the Information Society*, *20*, 479-493.

Garrido, I., Andion, J., & Del Alamo, J. M. (2023). An assessment of challenge-based learning for computer programming at the undergraduate level. In *ICERI2023 Proceedings* (pp. 8299-8305). IATED.

Grover, S. (2020, February). Designing an assessment for introductory programming concepts in middle school computer science. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 678-684).

Grover, S., & Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 267-272).

Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, *29*(2-3), 106-135.

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

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199-237.

Grover, S., Pea, R., & Cooper, S. (2016, February). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 552-557).

Guo, P. J. (2018, April). Non-native English speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI conference on human factors in computing systems* (pp. 1-14).

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.

Huang, W., & Looi, C. K. (2021). A critical review of literature on "unplugged" pedagogies in K-12 computer science and computational thinking education. *Computer Science Education, 31*(1), 83-111.

Hwang, G. J., Tung, L. H., & Fang, J. W. (2023). Promoting students' programming logic and problem-solving awareness with precision feedback: A two-tier test-based online programming training approach. *Journal of Educational Computing Research*, *60*(8), 1895-1917.

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration*, *1*(1).

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

Kallia, M., & Sentance, S. (2021). Threshold concepts, conceptions and skills: Teachers' experiences with students' engagement in functions. *Journal of Computer Assisted Learning*, *37*(2), 411-428.

Kohn, T. (2017). *Teaching Python programming to novices: Addressing misconceptions and creating a development environment*. ETH Zurich.

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.

Kwon, K., Ottenbreit-Leftwich, A. T., Brush, T. A., Jeon, M., & Yan, G. (2021). Integration of problem-based learning in elementary computer science education: effects on computational thinking and attitudes. *Educational Technology Research and Development*, *69*, 2761-2787.

Lai, R. P. (2021). Beyond programming: A computer-based assessment of computational thinking competency. *ACM Transactions on Computing Education (TOCE)*, *22*(2), 1-27.

Lee, Irene & Martin, Fred & Denner, Jill & Coulter, Bob & Allan, Walter & Erickson, Jeri & Malyn-Smith, Joyce & Werner, Linda. (2011). Computational thinking for youth in practice. *ACM Inroads*. 2. 32-37. 10.1145/1929887.1929902.

Liao, Y. K. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of educational computing research*, *7*(3), 251-268.

Lockwood, J., & Mooney, A. (2018). Developing a computational thinking test using Bebras problems.

Lohr, S. (2012). Big Data is great, but don't forget intuition. *The New York Times.* Retrieved from http://www.nytimes.com/2012/12/30/technology/big-data-is-great-but-dont-forget-intuition.html

Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *Proceedings of the 40th ACM technical symposium on Computer Science Education* (pp. 260-264).

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

Ma, N., Qian, J., Gong, K., & Lu, Y. (2023). Promoting programming education of novice programmers in elementary schools: A contrasting cases approach for learning programming. *Education and Information Technologies*, 1-24.

Mayer, R. E. (2004). Should there be a three-strikes rule against pure discovery learning? *American Psychologist, 59*, 14–19.

McGill, T. J., & Volet, S. E. (1997). A Conceptual Framework for analyzing students' knowledge of programming. *Journal of Research on Computing in Education*, 6504(December), 37–41. https://doi.org/10.1080/08886504.1997.10782199

MEB. (2020). Programlamanın temelleri textbook. Retrieved from http://kitap.eba.gov.tr/panel/dosyalar/upload/1375/0/U_0_29_08_2020_19_36_02_368.pdf

MEB. (2022). Retrieved from http://meslek.eba.gov.tr/?p=Ogretim-Programi&tur=mtal&sinif=9

Mladenović, M., Žanko, Ž., & Zaharija, G. (2024). From blocks to text: Bridging programming misconceptions. *Journal of Educational Computing Research*, 07356331241240047.

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23*, 1483-1500.

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' scratch skills with their computational thinking skills in terms of different variables. *Online Submission*, *8*(11), 1-7.

Özmen, B., & Altun, A. (2014). Undergraduate students' experiences in programming: difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 1-27.

Papert, S. (1980). "Mindstorms" Children. *Computers and Powerful Ideas*.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137-168.

Polat, E., Hopcan, S., Kucuk, S., & Sisman, B. (2021). A comprehensive assessment of secondary school students' computational thinking skills. *British Journal of Educational Technology*, *52*(5), 1965-1980.

Prasad, A., Chaudhary, K., & Sharma, B. (2022). Programming skills: Visualization, interaction, home language and problem solving. *Education and Information Technologies*, 1-27.

Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The impact of user interface on young children's computational thinking. *Journal of Information Technology Education. Innovations in Practice*, *16*, 171.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*, 137–172.

Rodriguez, B. R. (2015). *Assessing computational thinking in computer science unplugged activities*. Colorado School of Mines.

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. *Computational Thinking Education*, 79-98.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691.

Royal Society. (2012). Shut down or restart? The way forward for computing in UK schools.

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, *111*(5), 764.

Selby, C., & Woollard, J. (2013). Computational thinking: The developing definition. University of Southampton.

Selby, C. C. (2012, November). Promoting computational thinking with programming. In *Proceedings of the 7th workshop in primary and secondary computing education* (pp. 74-77).

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158.

Sirkiä, T., & Sorva, J. (2012, November). Exploring programming misconceptions: An analysis of student mistakes in visual program simulation exercises. In *Proceedings of the 12th Koli calling international conference on computing education research* (pp. 19-28).

Soloway, E., & Spohrer, J. C. (1989). Studying the novice programmer. Lawrence Erlbaum Associates. https://doi.org/10.4324/9781315808321

Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, *13*(4), 1-64.

Spohrer, J. C., & Soloway, E. (1986, April). Alternatives to construct-based programming misconceptions. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 183-191).

Sun, D., Looi, C. K., Li, Y., Zhu, C., Zhu, C., & Cheng, M. (2024). Block-based versus text-based programming: a comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming. *Educational Technology Research and Development*, *72*(2), 1067-1089.

Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity*, *42*, 100926.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, *148*, 103798.

Tran, Y. (2018). Computer programming effects in elementary: Perceptions and career aspirations in STEM. *Technology, Knowledge and Learning*, *23*(2), 273-299.

Vieira, C., Gómez, R. L., Gómez, M., Canu, M., & Duque, M. (2023). Implementing unplugged CS and use-modify-create to develop student computational thinking skills. *Educational Technology & Society*, *26*(3), 155-175.

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.

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, *142*, 103646.

Weintrop, D., Shepherd, D. C., Francis, P., & Franklin, D. (2017, October). Blockly goes to work: Block-based programming for industrial robots. In *2017 IEEE Blocks and Beyond Workshop (B&B)* (pp. 29-36). IEEE.

Wing, J. (2011). Research notebook: Computational thinking—What and why. The link magazine, 6, 20-23.

Wing, J. (2010). Computational Thinking and Thinking What and Why.

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 A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Wong, G. K. W., & Cheung, H. Y. (2020). Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments*, *28*(4), 438-450.

Xu, E., Wang, W., & Wang, Q. (2023). A meta-analysis of the effectiveness of programming teaching in promoting K-12 students' computational thinking. *Education and Information Technologies*, *28*(6), 6619-6644.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235-254.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)*, *14*(1), 1-16.

Yeomans, L., Zschaler, S., & Coate, K. (2019). Transformative and troublesome? Students' and professional programmers' perspectives on difficult concepts in programming. *ACM Transactions on Computing Education (TOCE)*, *19*(3), 1-27.

Yukselturk, E., & Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self- efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology*, *48*(3), 789-801.

Yurdakök, E. A., & Kalelioğlu, F. (2024). The effect of teaching physical programming on computational thinking skills and self-efficacy perceptions towards computational thinking. *Journal of Educational Computing Research*, *62*(3), 785-815.

Žanko, Ž., Mladenović, M., & Boljat, I. (2019). Misconceptions about variables at the K-12 level. *Education and Information Technologies*, *24*, 1251-1268.

Zhang, Y., Liang, Y., Tian, X., & Yu, X. (2024). The effects of unplugged programming activities on K-9 students' computational thinking: meta-analysis. *Educational Technology Research and Development*, 1-26.